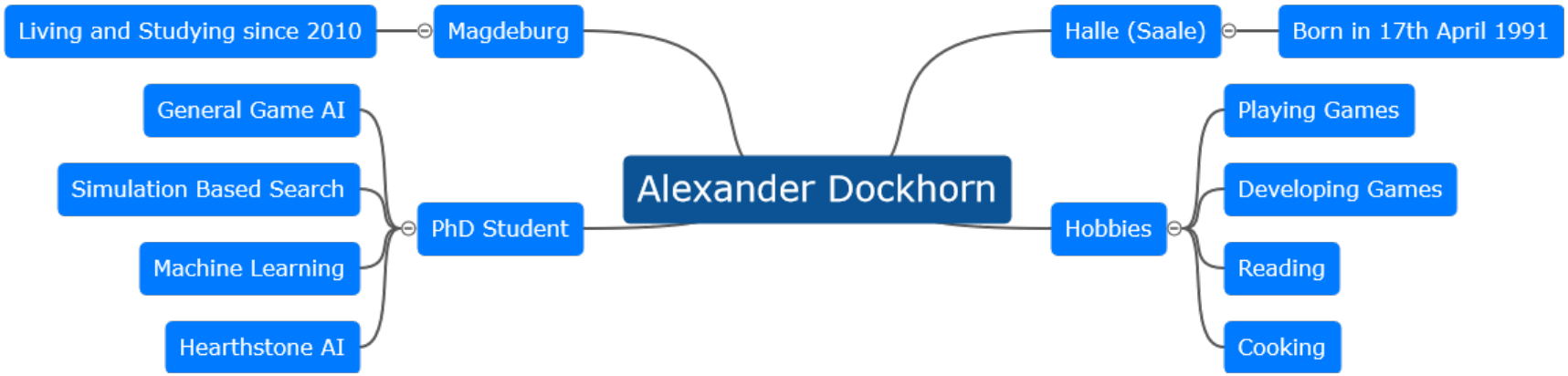
Generalisation of Simulation-Based Search for Autonomous Gameplaying

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

Short Introduction of Myself and My University

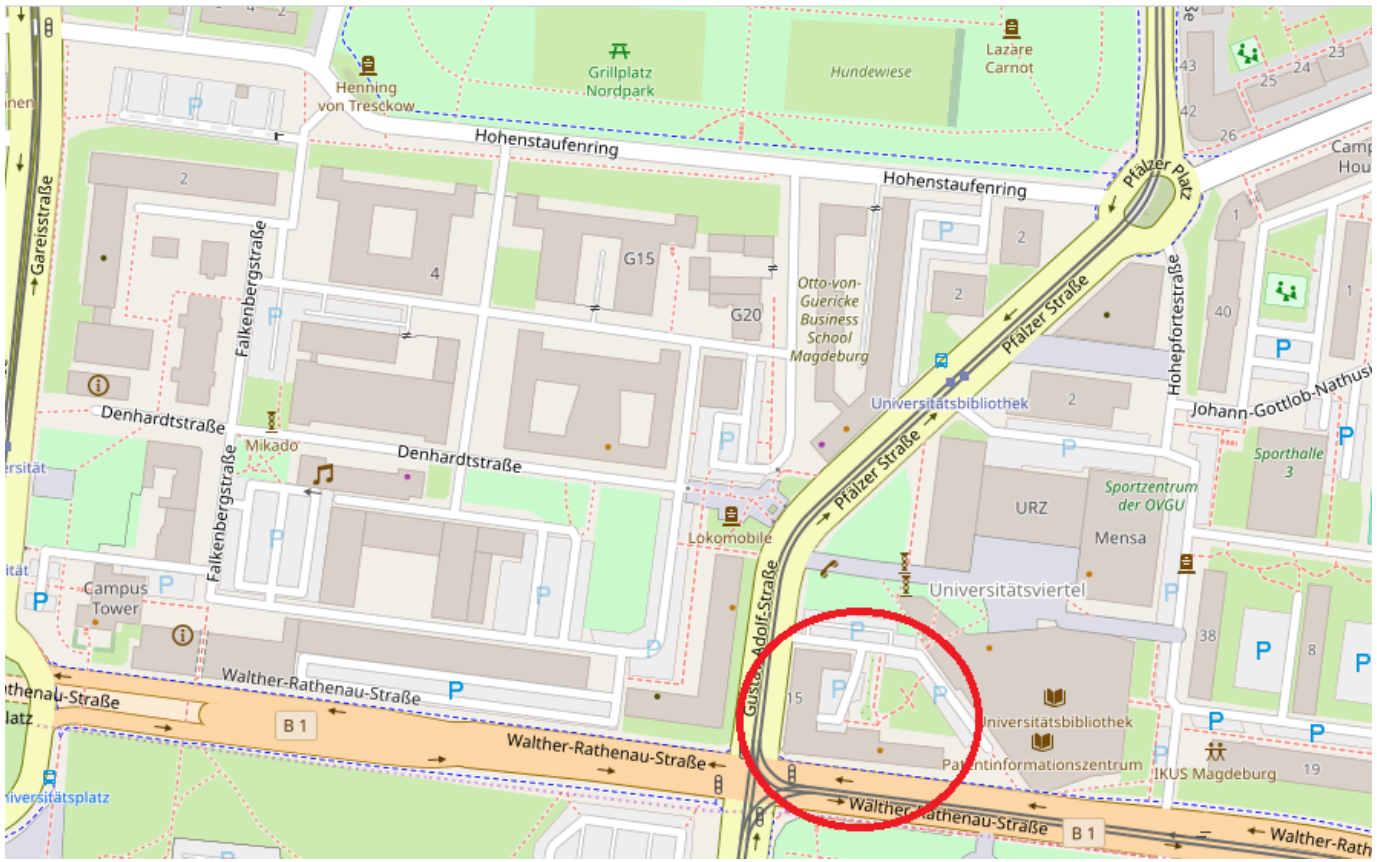


Where Am I From?

Where Am I From?

Where Am I From?

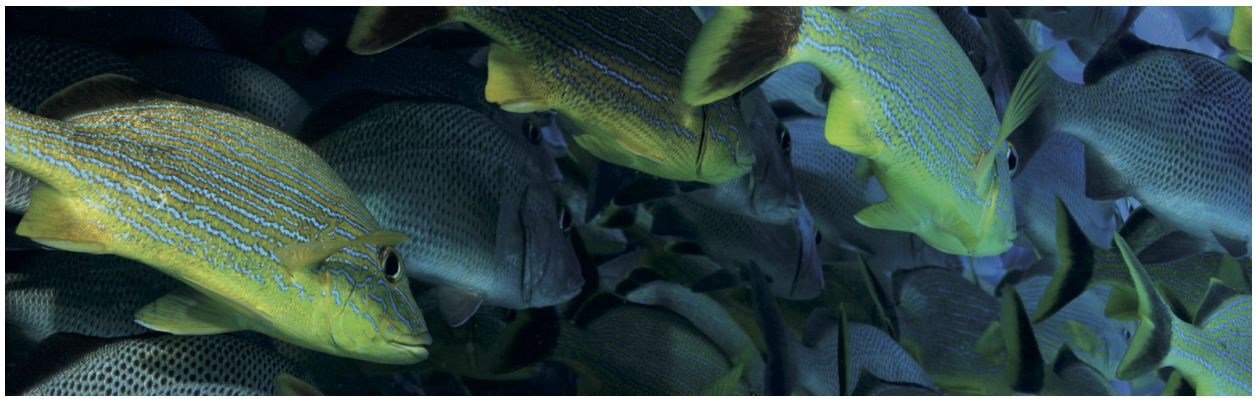
Where Am I From?



Computational Intelligence Research Group

Research Topics:

- Multi-Objective Optimization and Decision Making
- Swarm Intelligence
- Swarm Robotics
- Computational Intelligence in Games



Meet the Team


Sanaz Mostaghim


Rudolf Kruse


Sabine Laube


Michael Preuss


Xenija Neufeld


Christian Braune


Sebastian Mai


Christoph Steup


Heiner Zille


Florian Uhde


Simon Anderer


Palina Bartashevich


Jens Weise


Mahrokh Javadi


Me

Head of the Team

Prof. Dr.-Ing. habil. Sanaz Mostaghim

Chair of Computational Intelligence Research Group

Research topics: Swarm Intelligence, Swarm Robotics, Nature-inspired Optimization, Multi-objective Optimization and CI in Games

Prof. Dr. rer. nat. habil. Rudolf Kruse

Former Chair of Computational Intelligence Research Group

Research topics: Fuzzy Systems, Bayesien Networks, Classification, Clustering, Softcomputing, Data Mining, Exploratory Data Analysis

Generalisation of Simulation-based Search

overcoming classical constraints of simulation-based search

Gameplaying

Gameplaying

Gameplaying

Games and General Game Learning

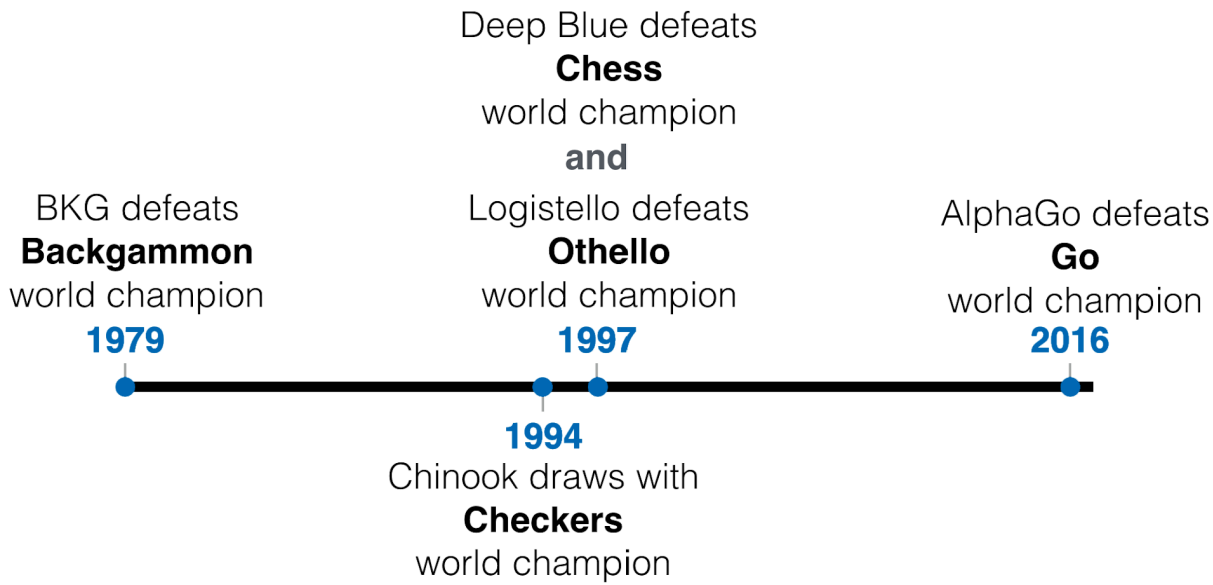
Games can be simulations of real world tasks

- 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

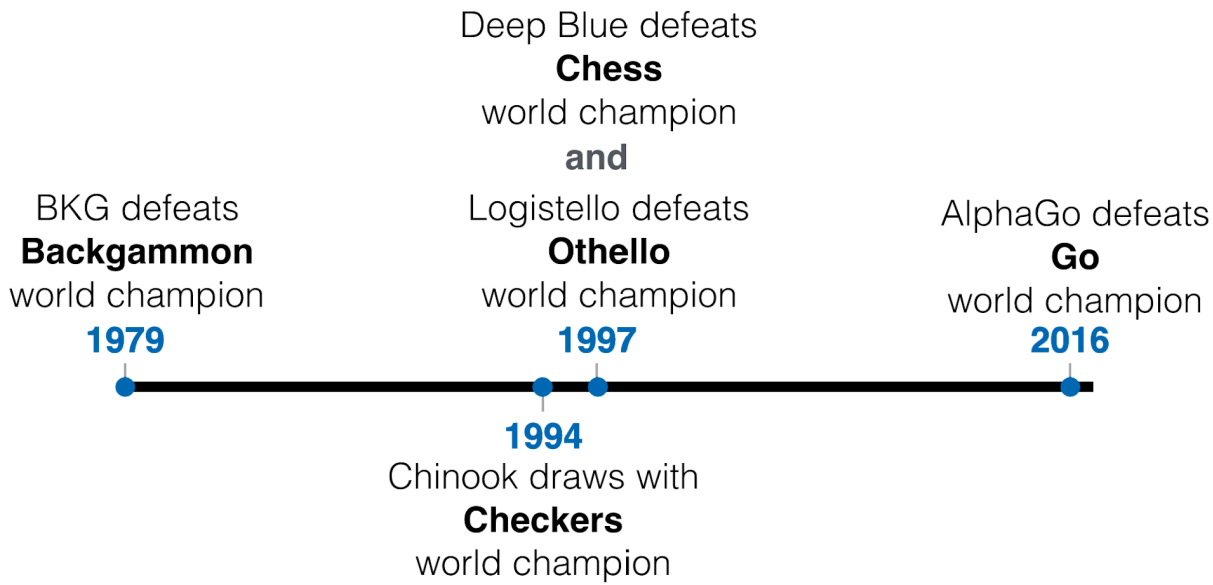
- quantifiable goal, varying difficulty, popular (large data sets)
- digital games are fully accessible to computers



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



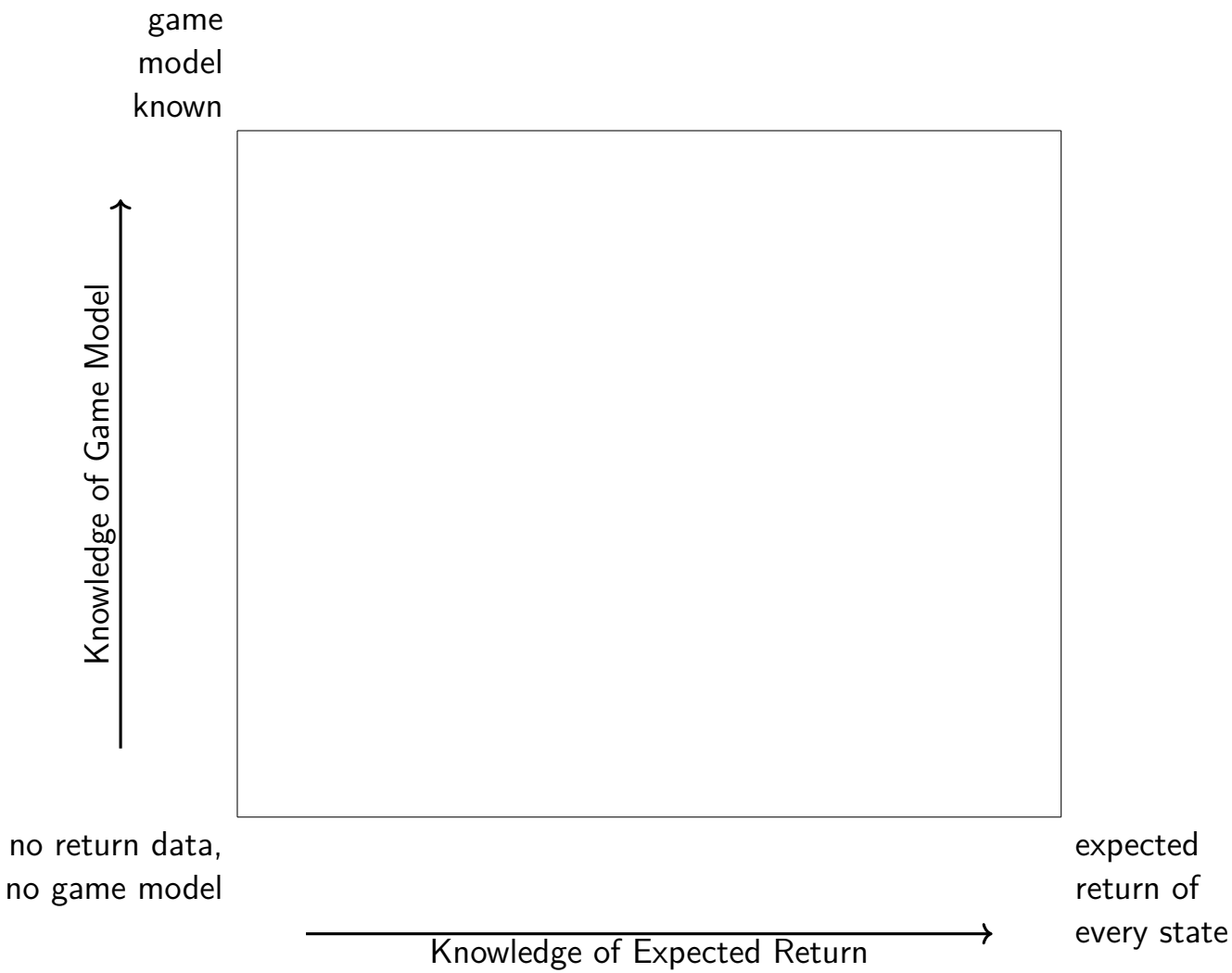
General Game Playing/Learning generalizes learning strategy across games but ignores learning the game's specific representation

Survey of Related Methods

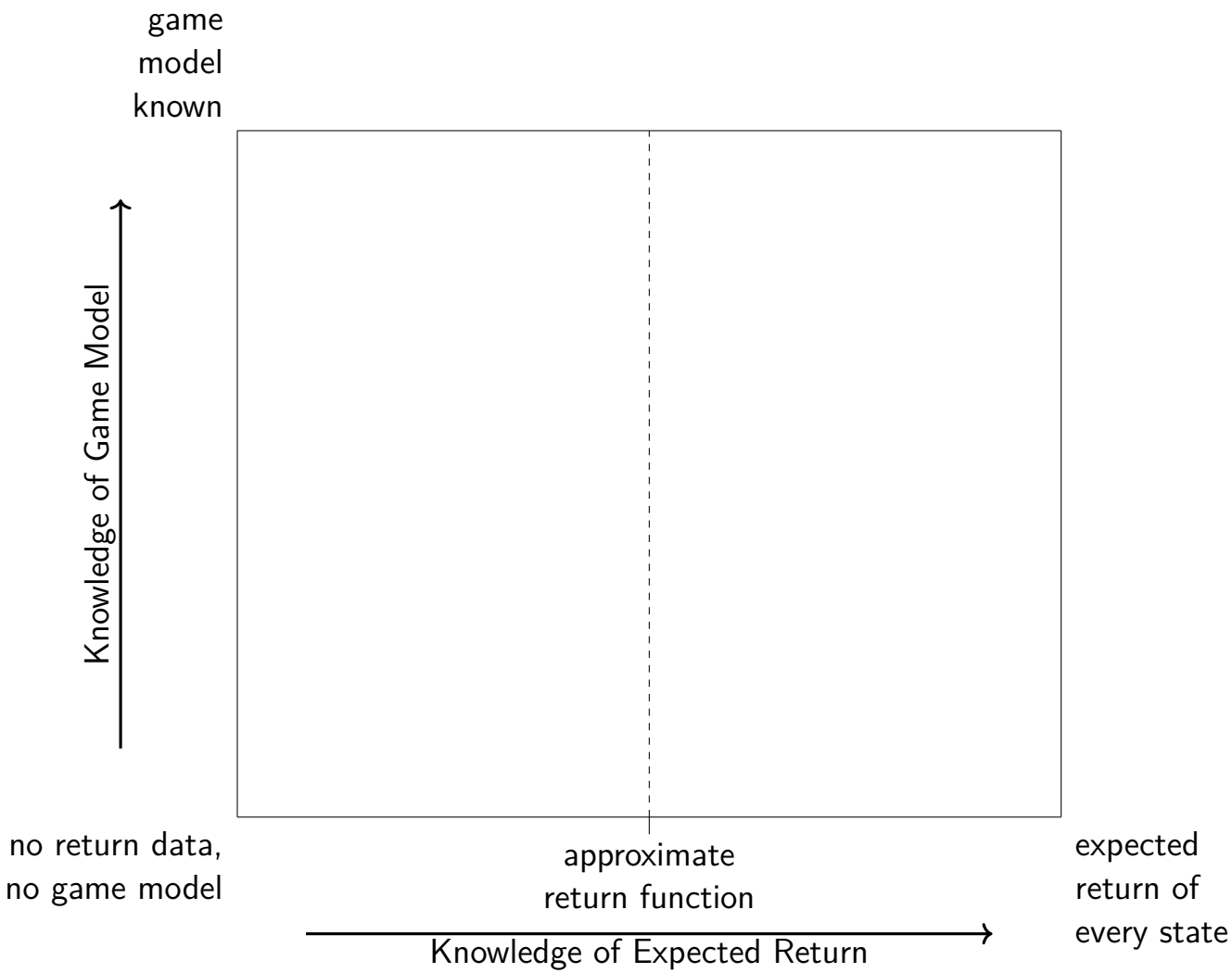
Survey of Related Methods

Survey of Related Methods

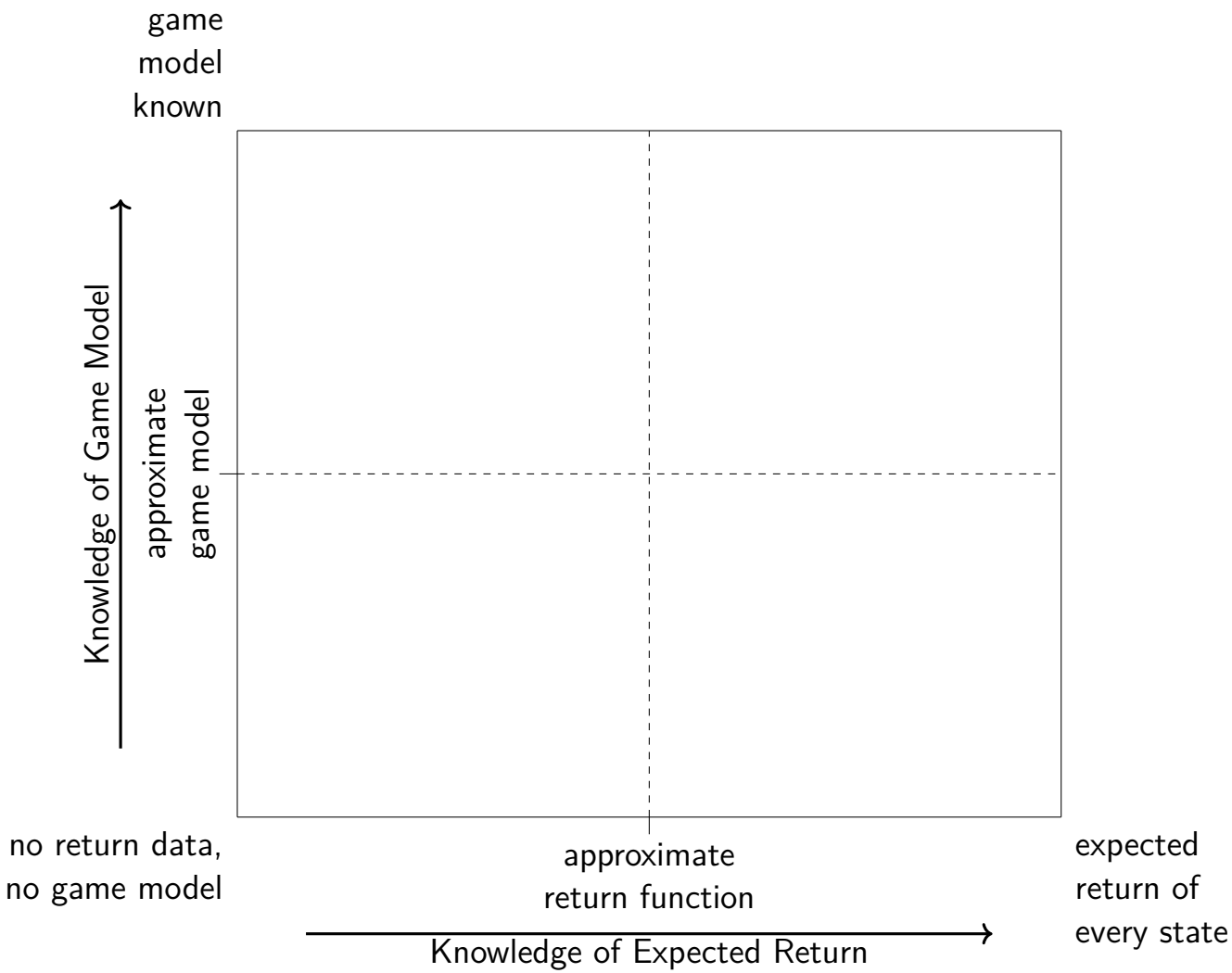
Survey of Related Methods



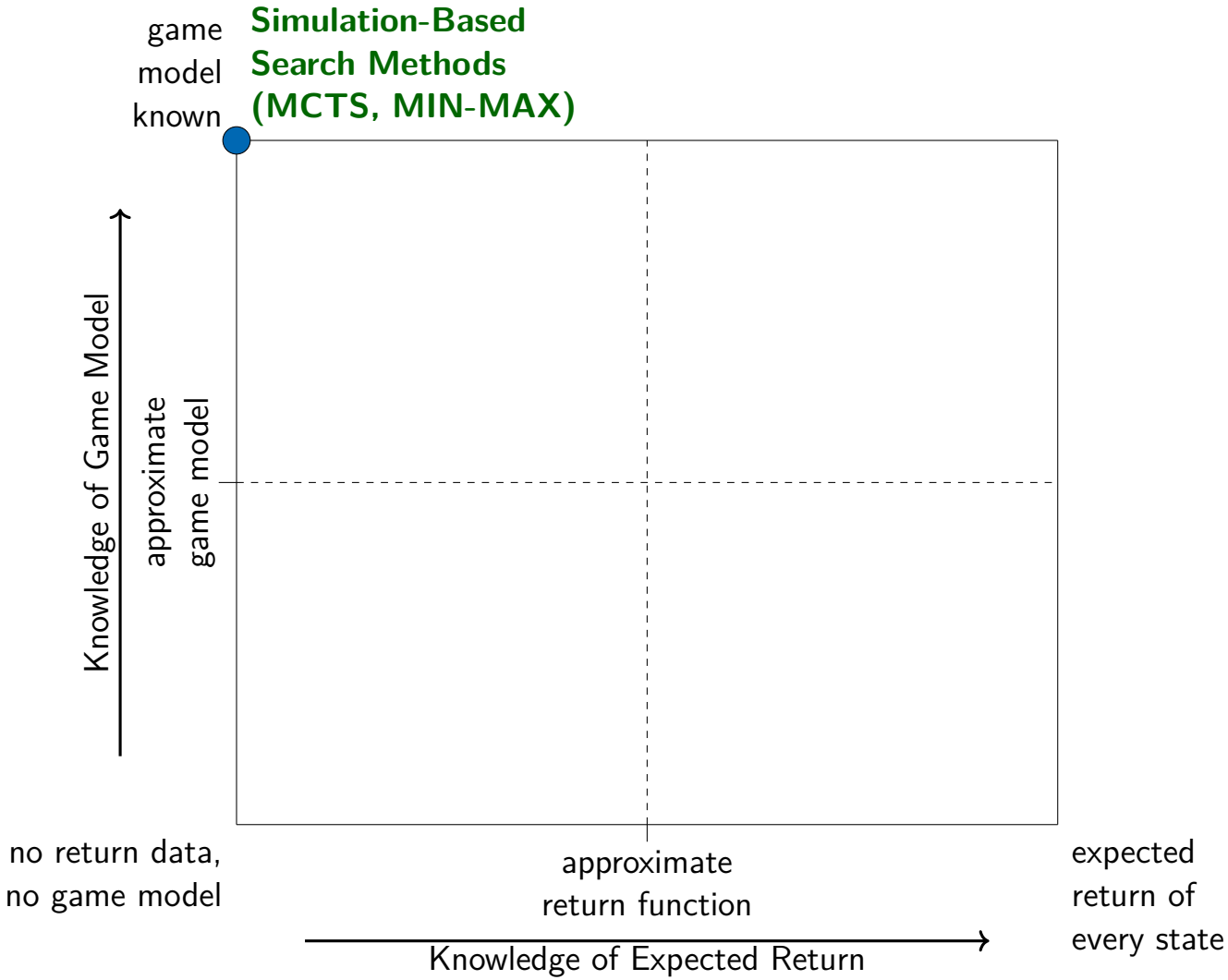
Survey of Related Methods



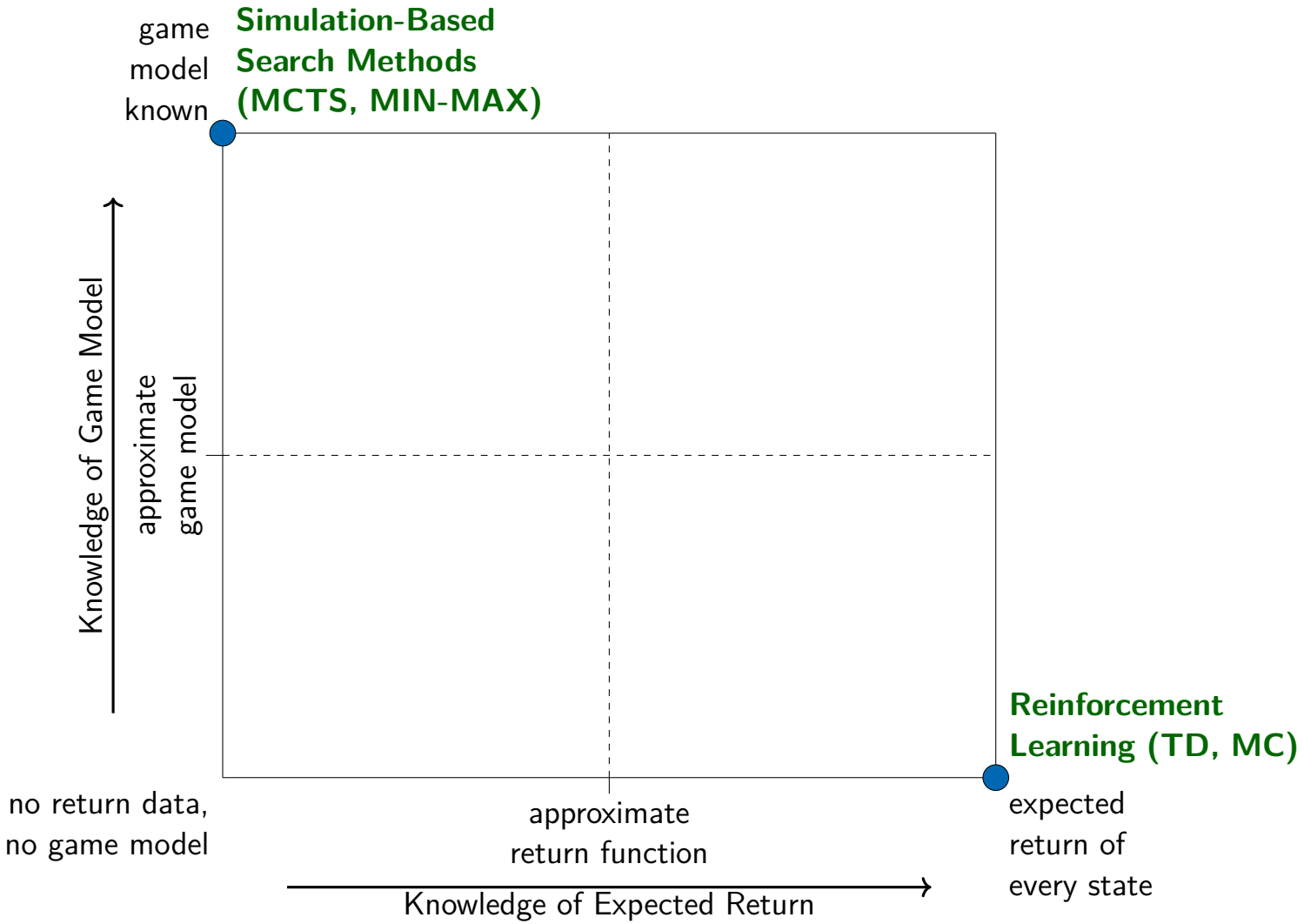
Survey of Related Methods



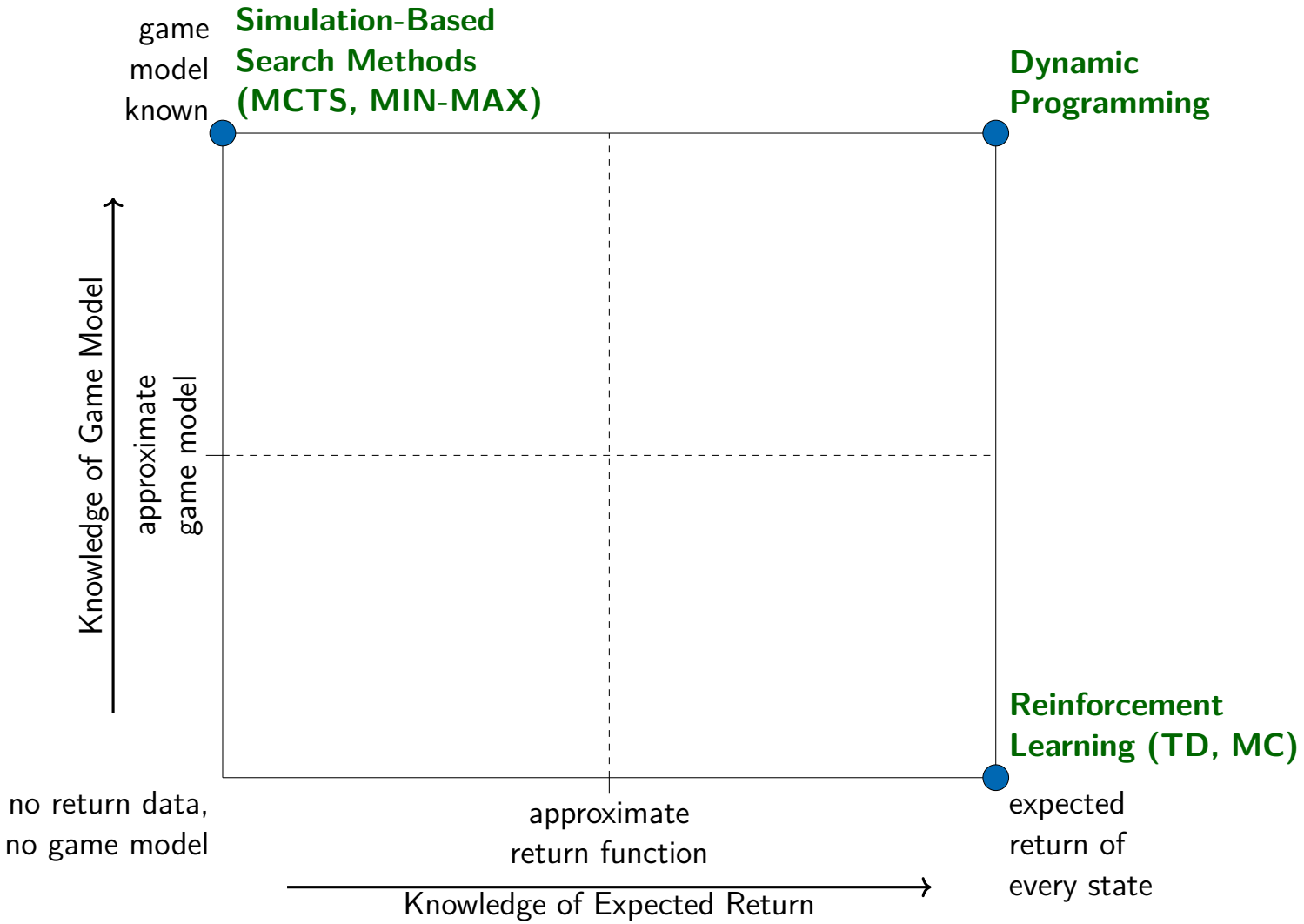
Survey of Related Methods



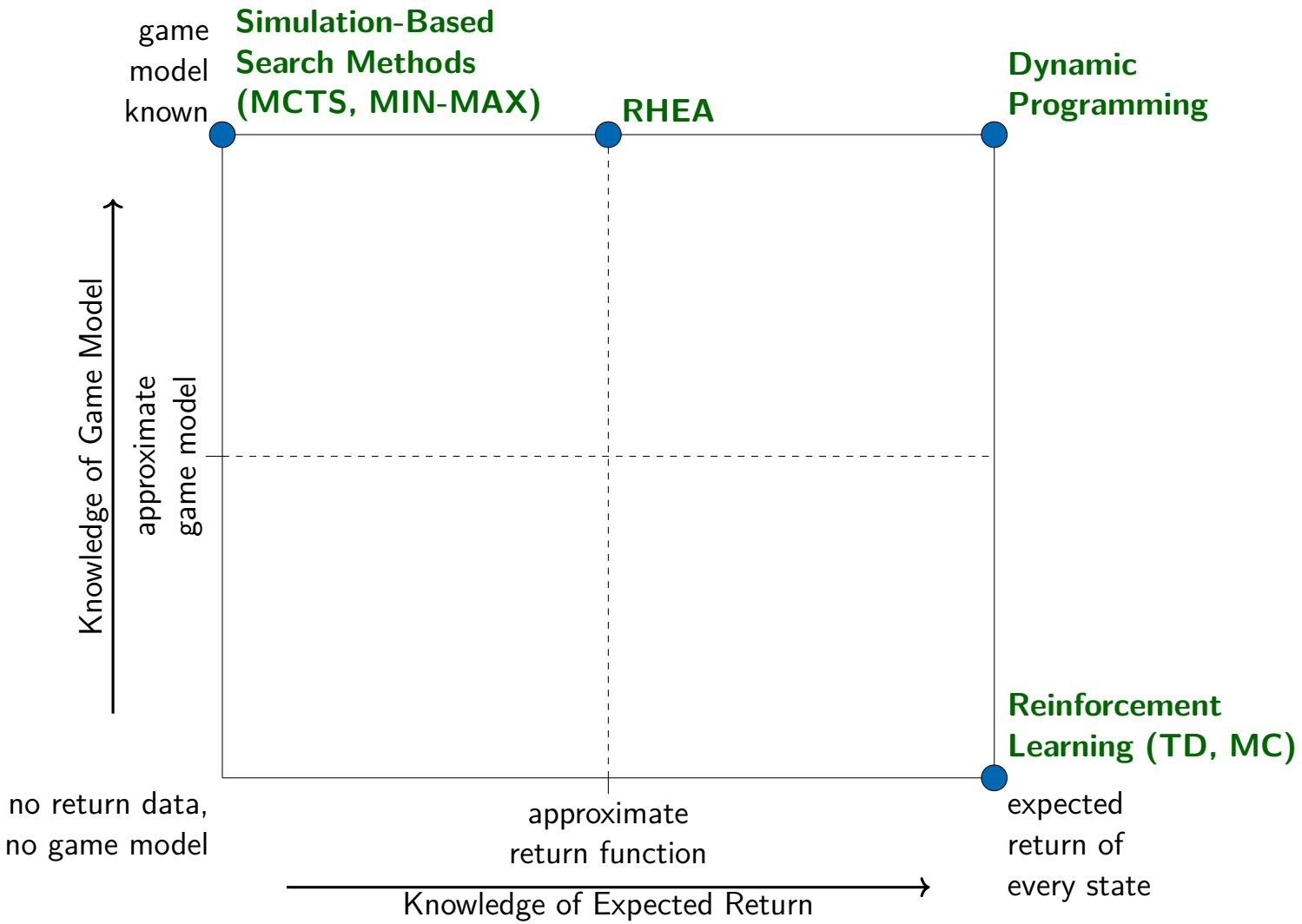
Survey of Related Methods



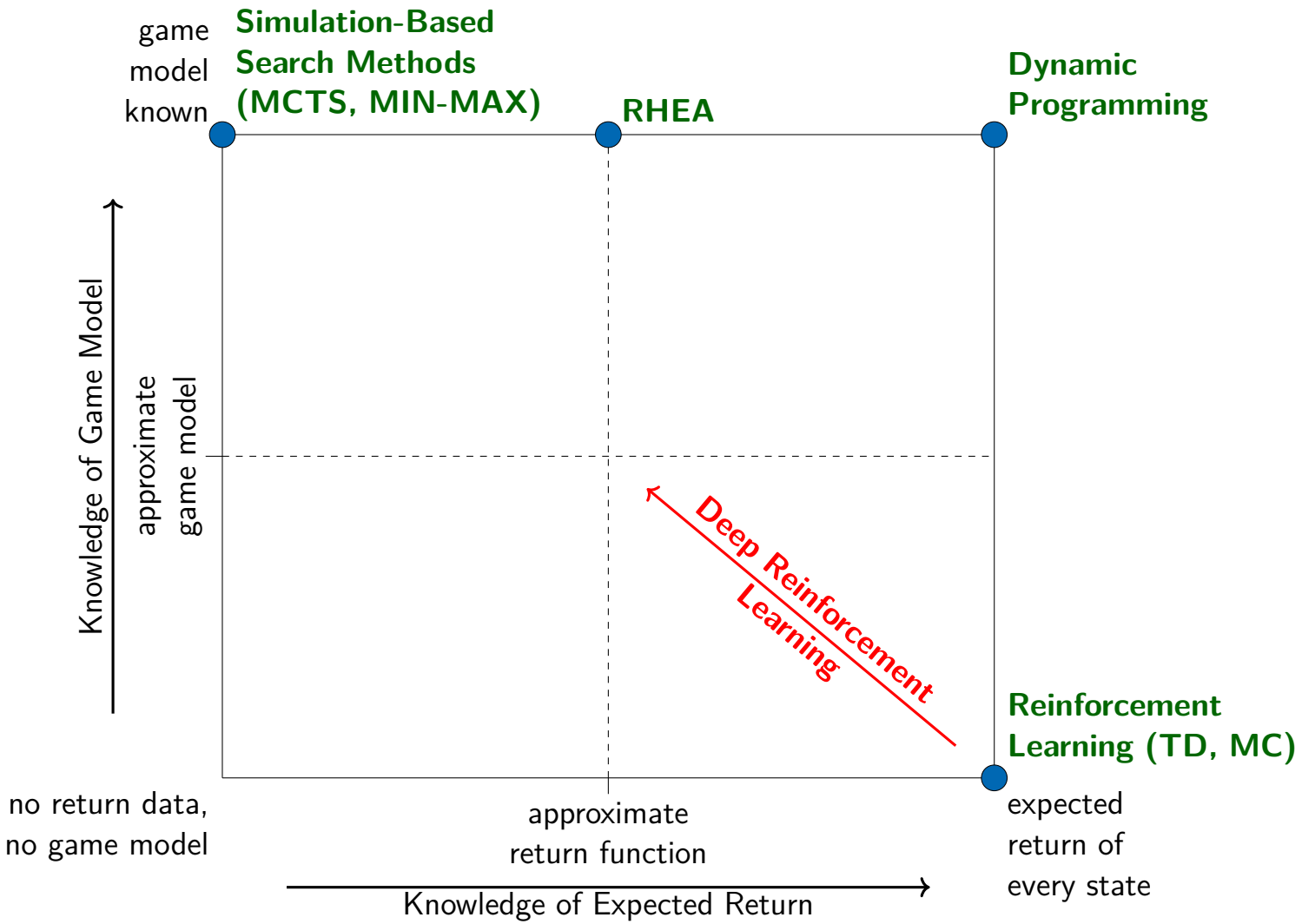
Survey of Related Methods



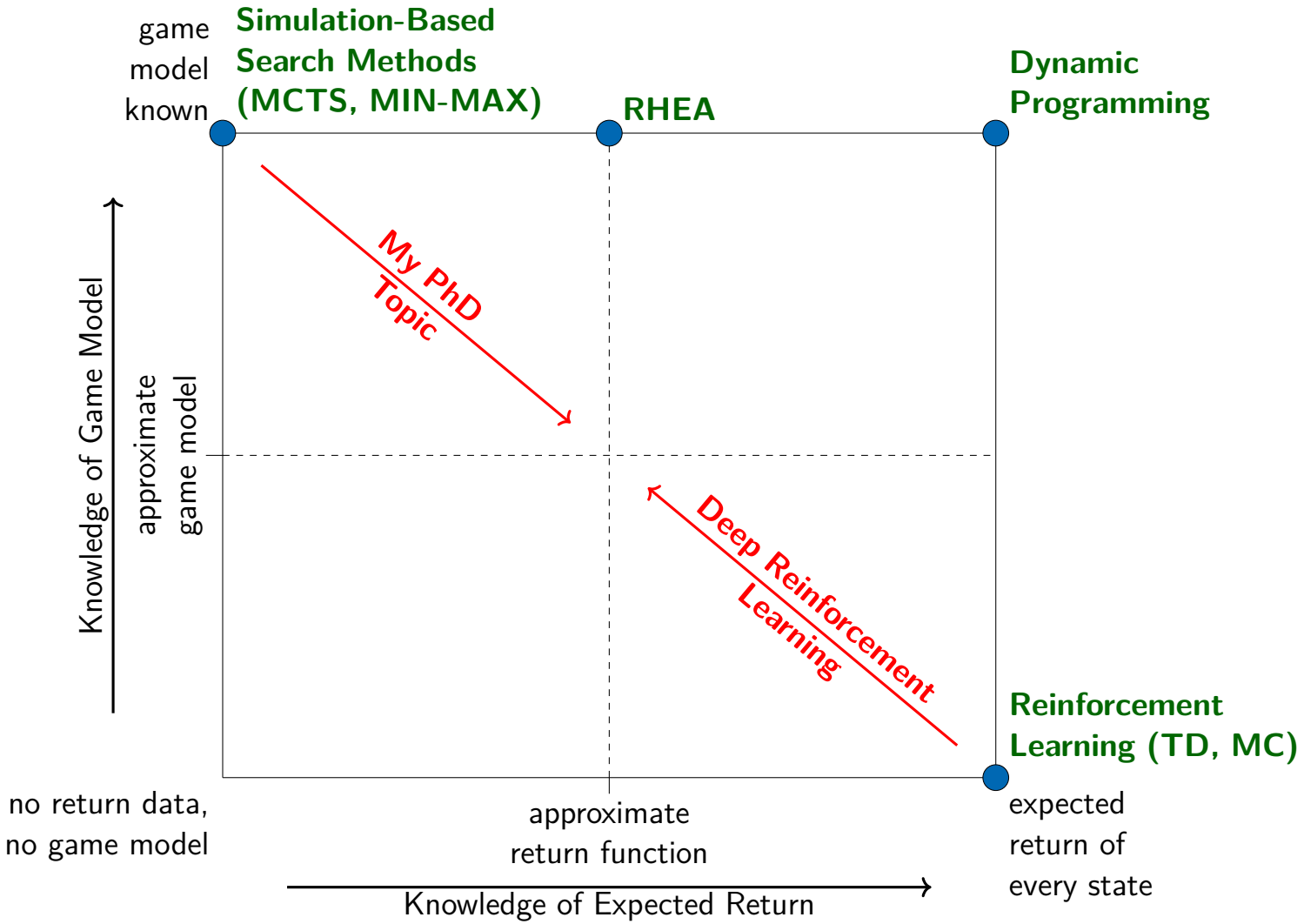
Survey of Related Methods



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

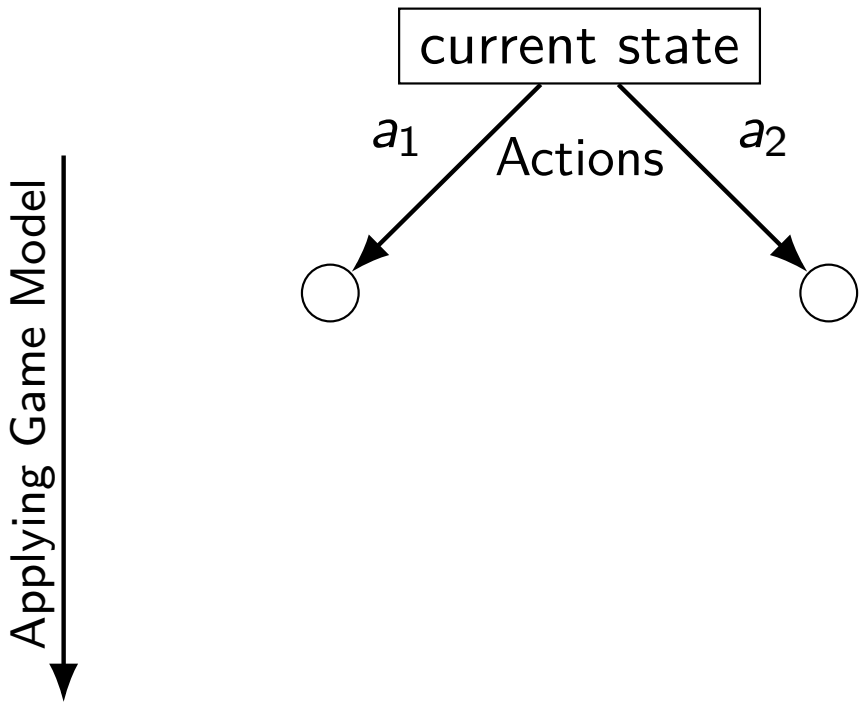
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



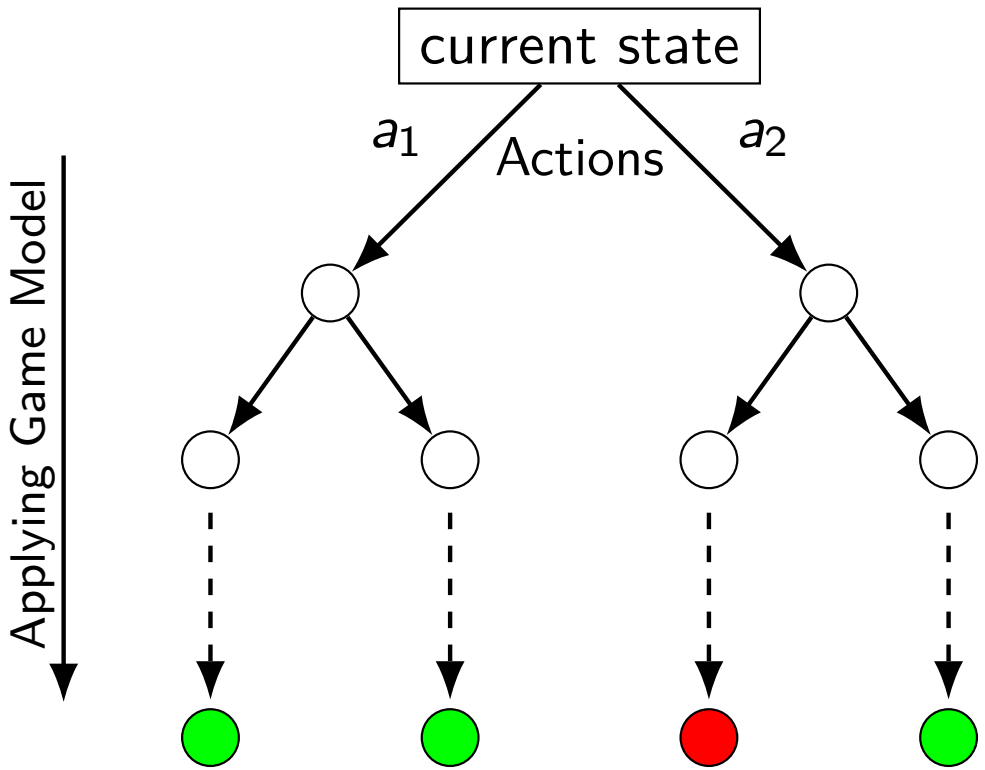
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



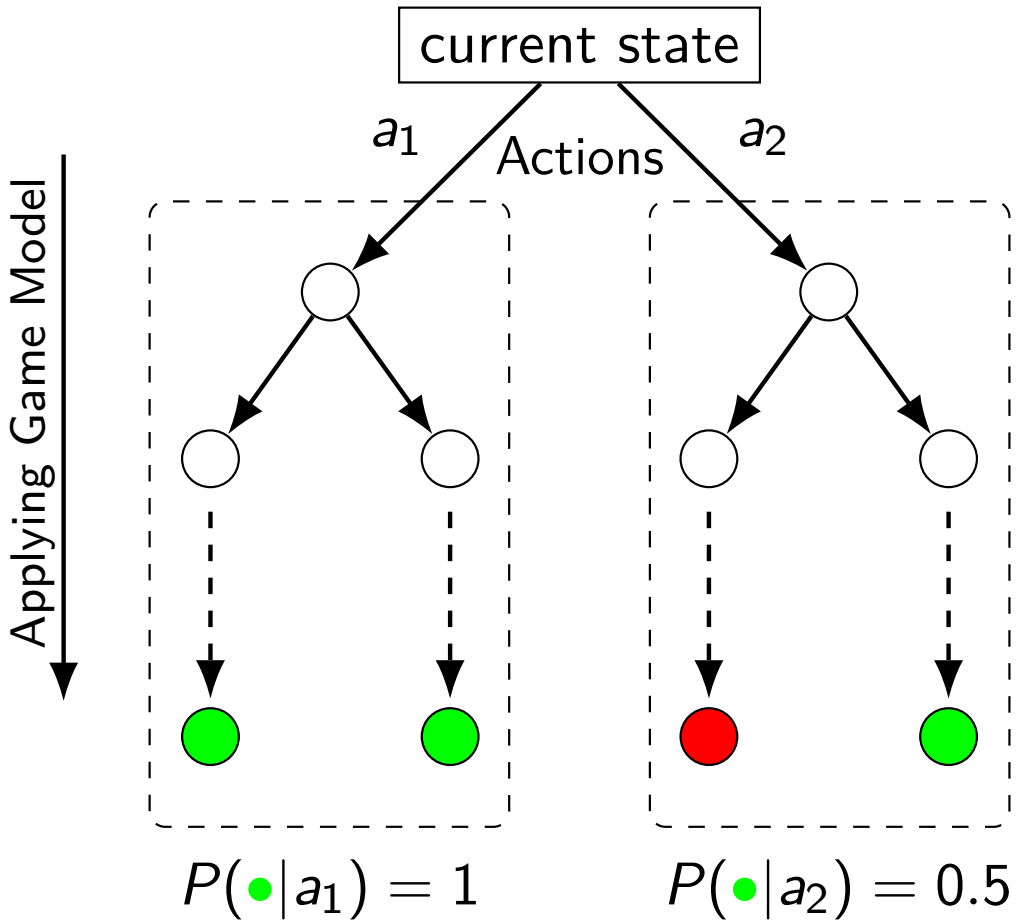
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



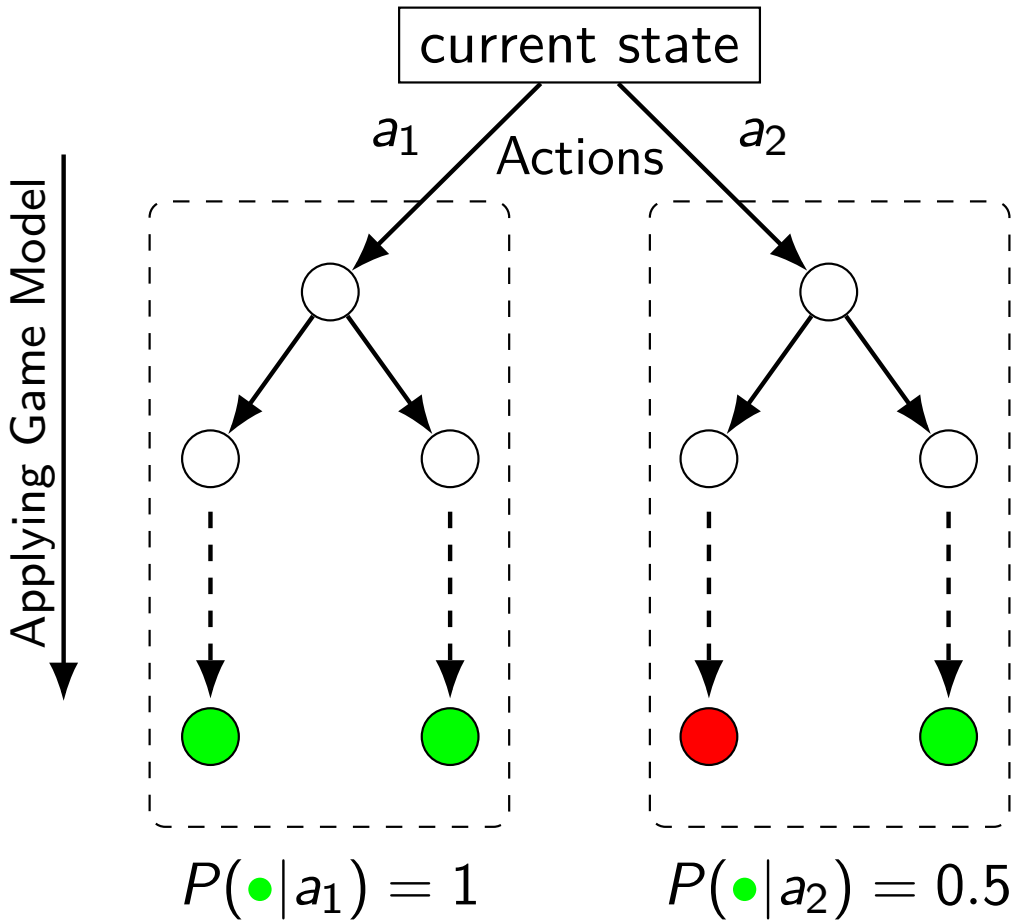
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



Input:

- current state
- game-model

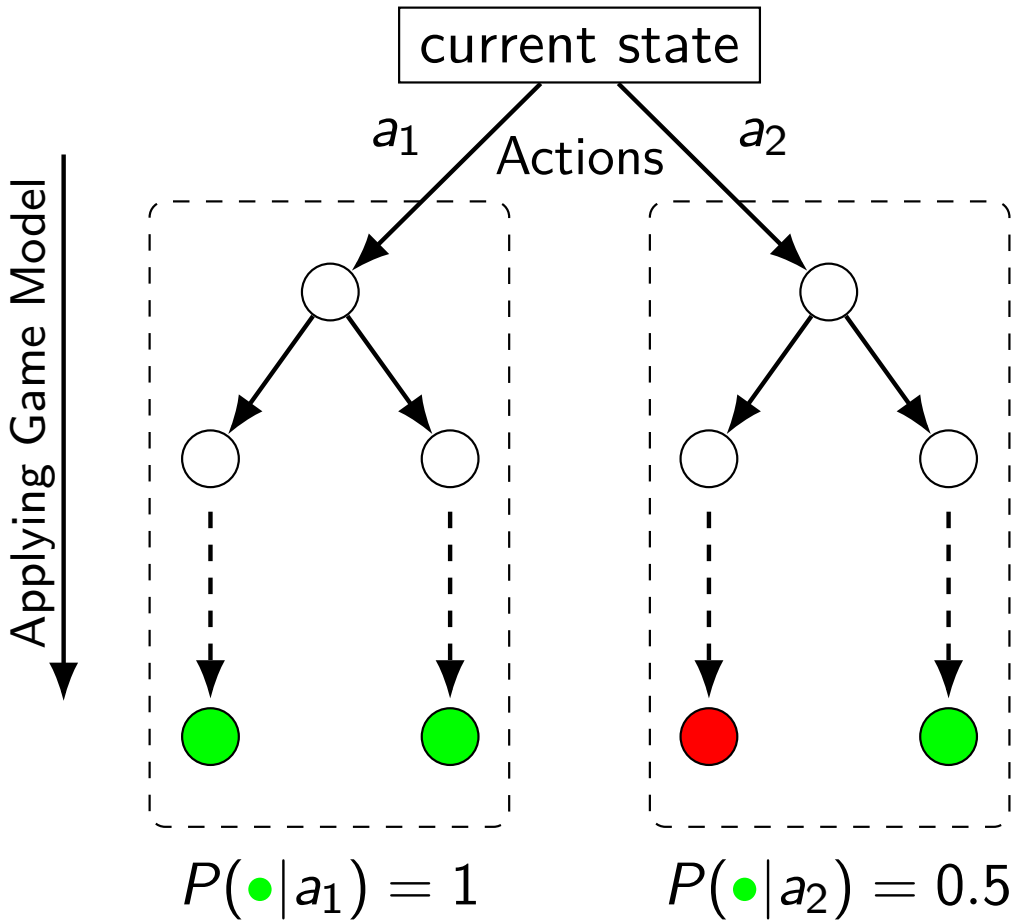
Output:

- 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

Restrictions of Simulation-Based Search

Simulation-Based search has two requirements:

1) the Forward Model needs to be known to the agent for forecasting the outcome of its actions
2) the current gamestate needs to be known, such that the agent can apply the forward model to it

Access to both is often assured in computer games.

Restrictions of Simulation-Based Search

Simulation-Based search has two requirements:

1) the Forward Model needs to be known to the agent for forecasting the outcome of its actions
2) the current gamestate needs to be known, such that the agent can apply the forward model to it

Access to both is often assured in computer games.

What do we do in case the requirements are not fulfilled?

Do those scenarios exist?

Restrictions of Simulation-Based Search

Simulation-Based search has two requirements:

1) the Forward Model needs to be known to the agent for forecasting the outcome of its actions
2) the current gamestate needs to be known, such that the agent can apply the forward model to it

Access to both is often assured in computer games.

What do we do in case the requirements are not fulfilled?

Do those scenarios exist?

Can simulation-based search still be applied?

Unknown Forward Model

What to do if the agent cannot acces a forward model?

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

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

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.

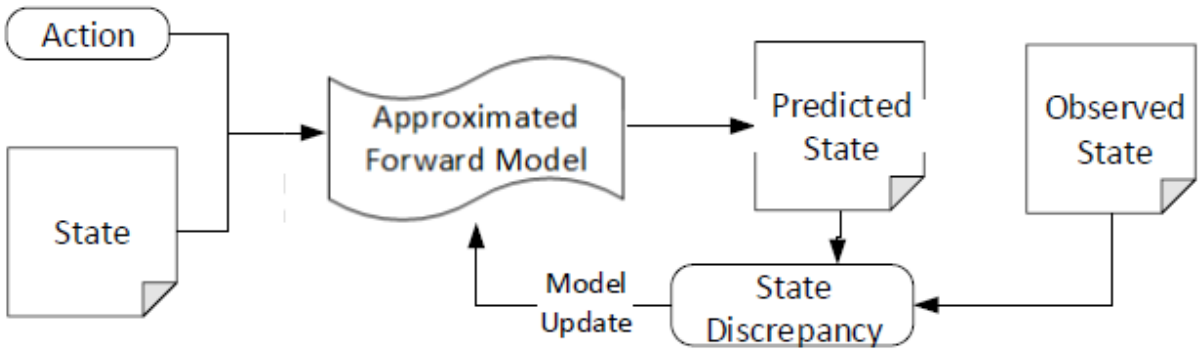
- apply simulation-based search using the learned model

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:

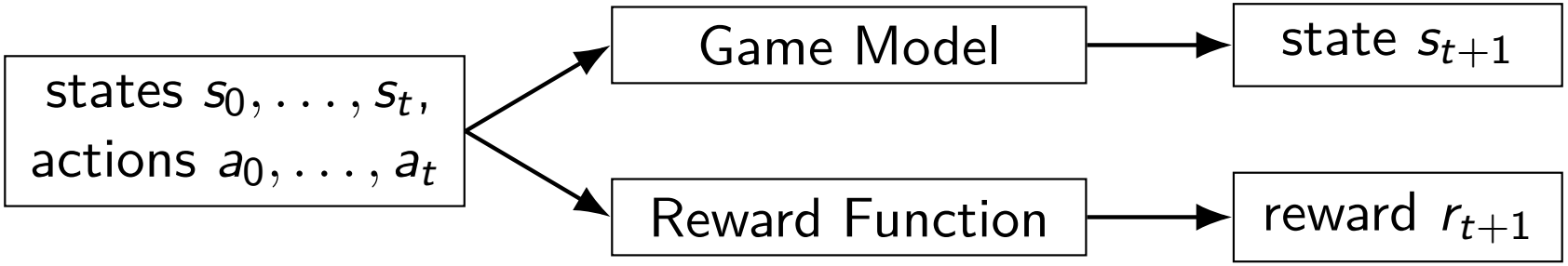
- map the current state and an action to the upcoming state



Part 1: 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:

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

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

Characteristics of the Forward Model

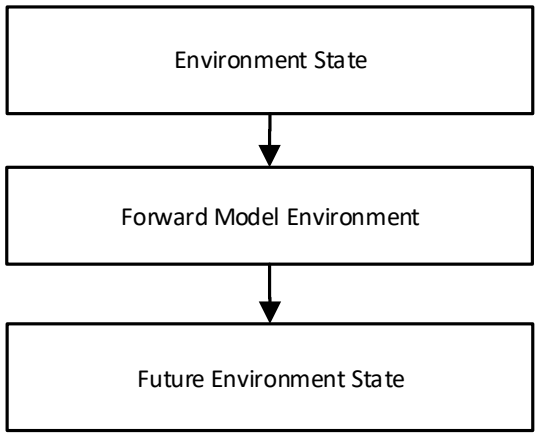
Model the state change as a single transition, considering the history of:

- 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



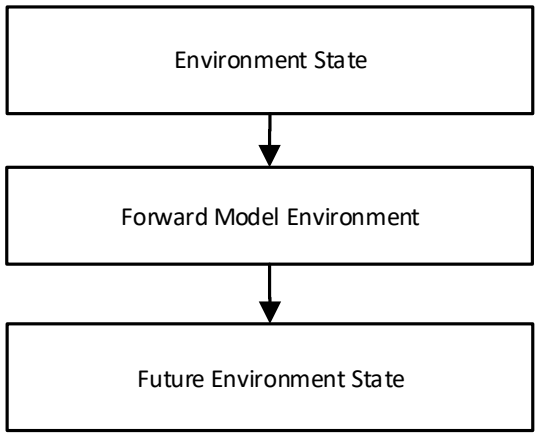
Markov property (order 1):

- 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

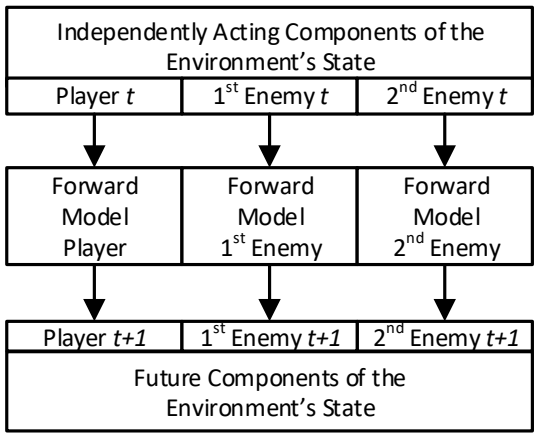
Inherent constraint of the VGDL:

- 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:

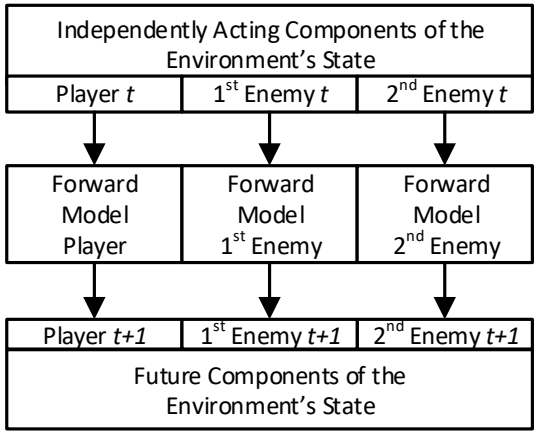
- model components individually and combine the result model
- simple component models are combined to a complex game model



Composite Model using Background Knowledge

Inherent constraint of the VGDL:

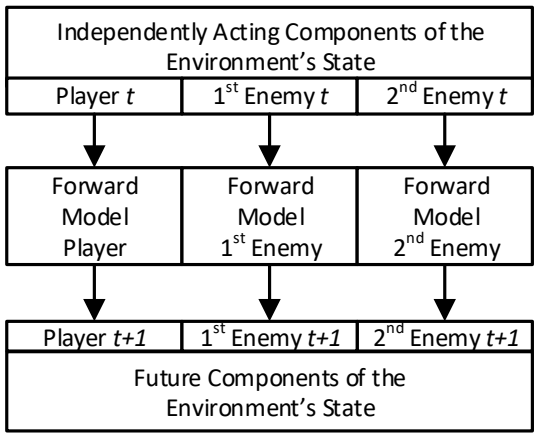
- model components individually and combine the result model
- simple component models are combined to a complex game model
- merge similar models

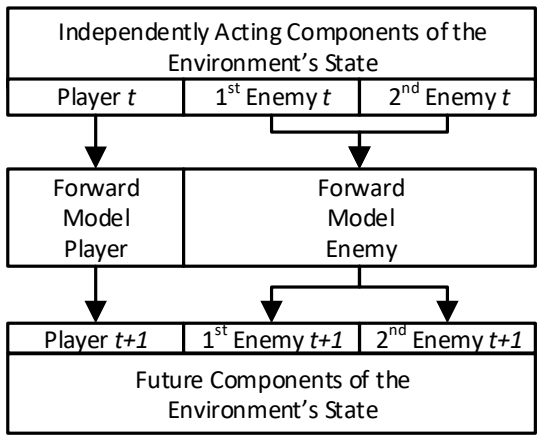


Composite Model using Background Knowledge

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

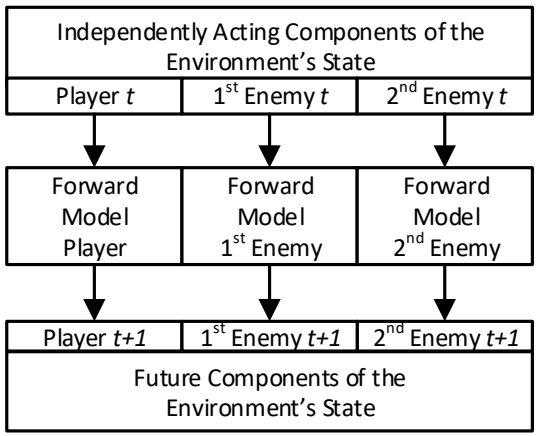




Composite Model using Background Knowledge

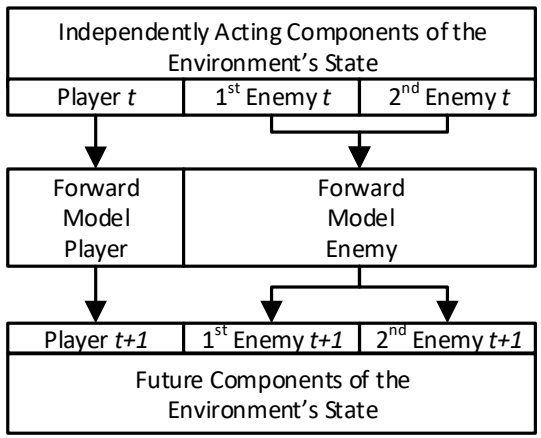
Inherent constraint of the VGDL:

- 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



Evaluation of Prediction Accuracy

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

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: $s_t, a_t \rightarrow s_{t+1}$

[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

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

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

General Video Game AI Competition:

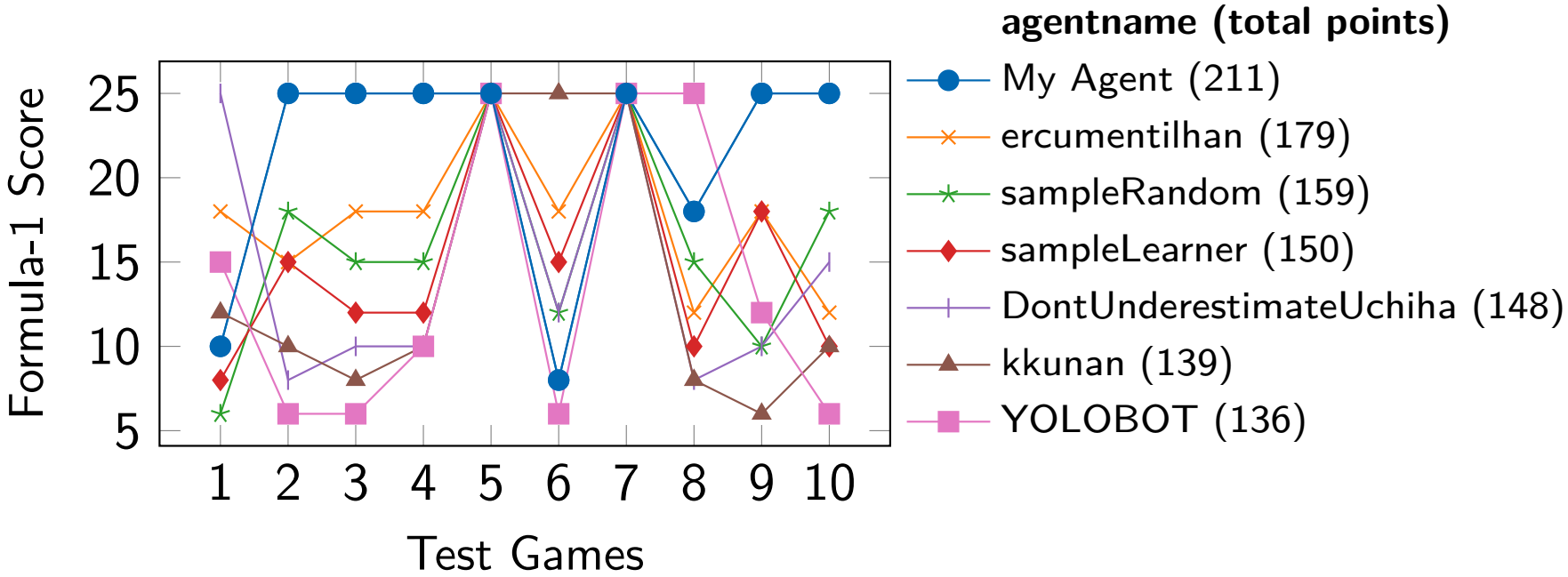
- 6 agents entered in 2017/2018
- evaluation is based on a set of 10 games
- Formula-1 scoring system

[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

General Video Game AI Competition:

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

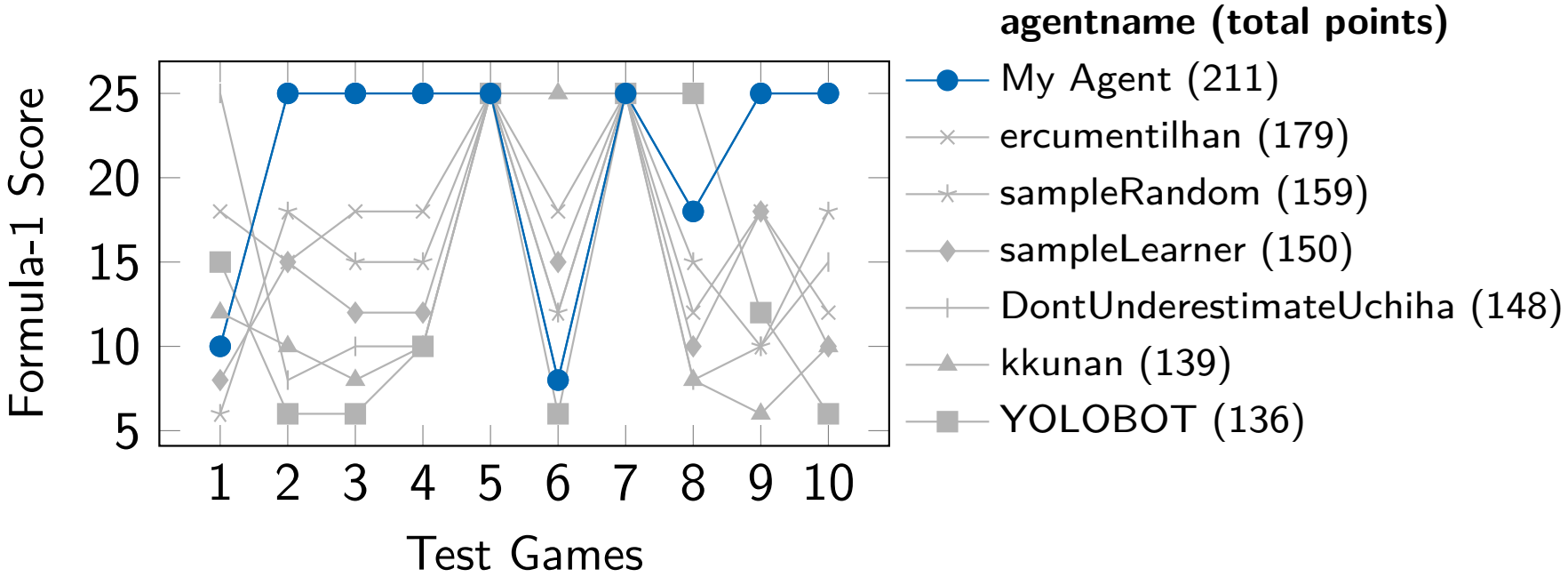


[1] Alexander Dockhorn and Daan Apeldoorn; *Forward Model Approximation for General Video Game Learning*, 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2018, pp. 425-432

Evaluation of Playing Performance

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

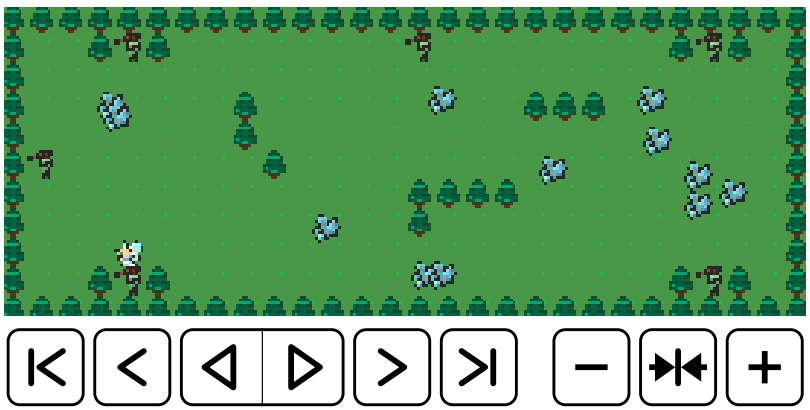
Comparison of Agents

→ Analysis of game model characteristics

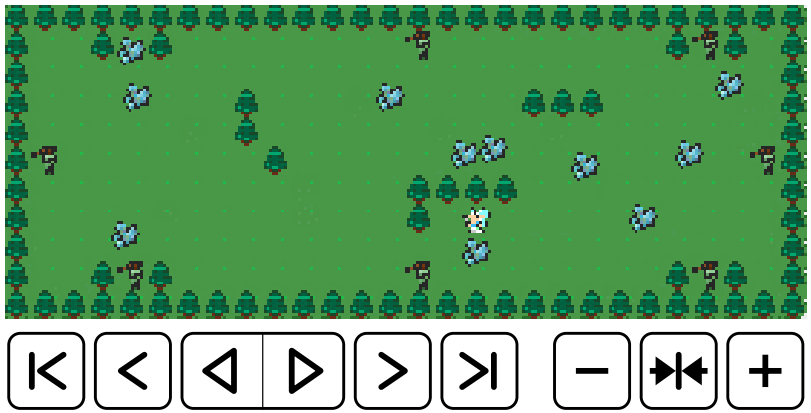
→ Game models can be learned using supervised learning

→ The proposed model is able to outperform other agents

Reinforcement Learning



Proposed Solution



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 naive 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 naive 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 naive Forward Model Approximation

Problem

- **needs background knowledge** (Work in progress)

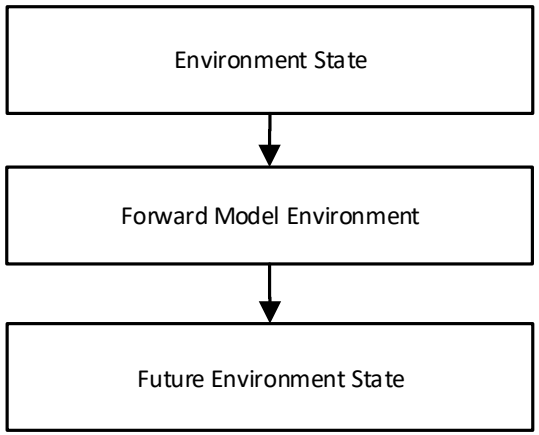
Finding a Composite Model (Work in Progress!)

Can we somehow build composite models?

Finding a Composite Model (Work in Progress!)

Can we somehow build composite models?

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

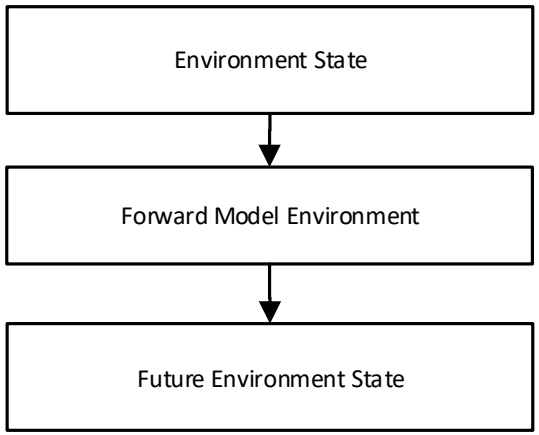


Finding a Composite Model (Work in Progress!)

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

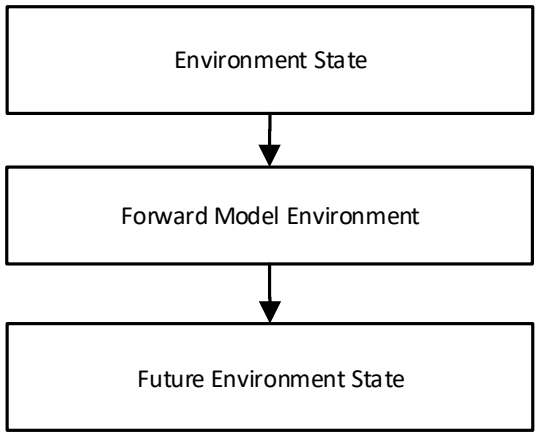


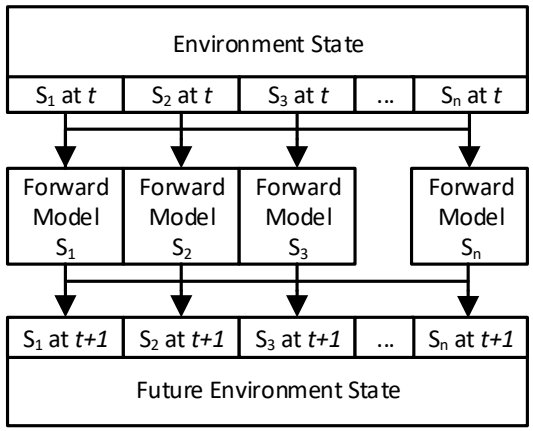
Finding a Composite Model (Work in Progress!)

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





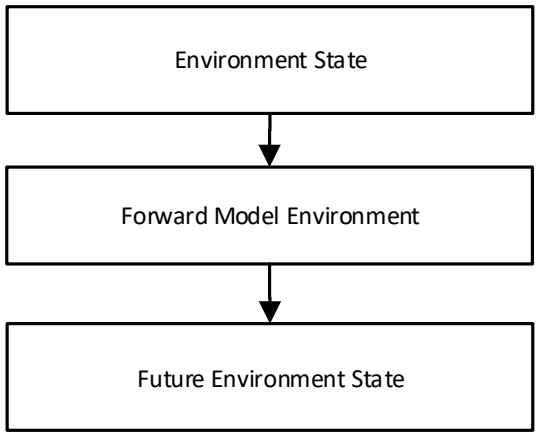
Finding a Composite Model (Work in Progress!)

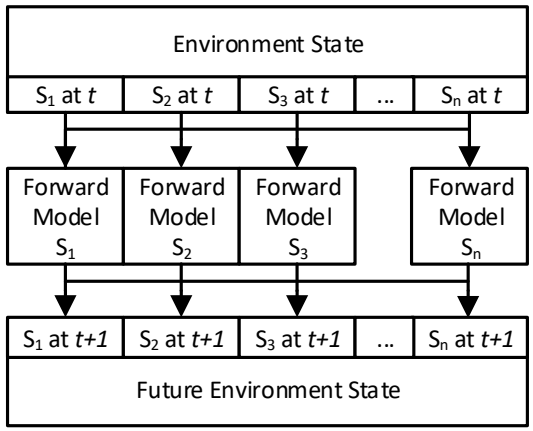
Can we somehow build composite models?

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 (Work in Progress!)

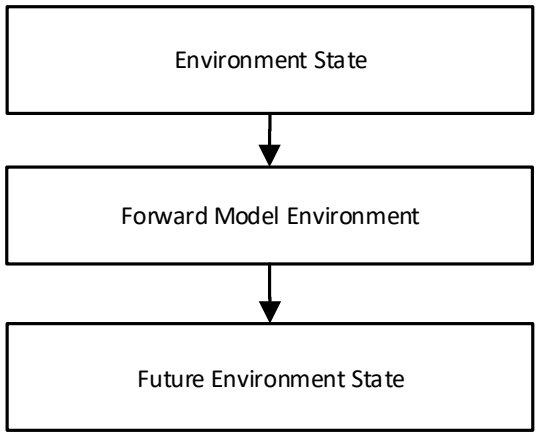
Can we somehow build composite models?

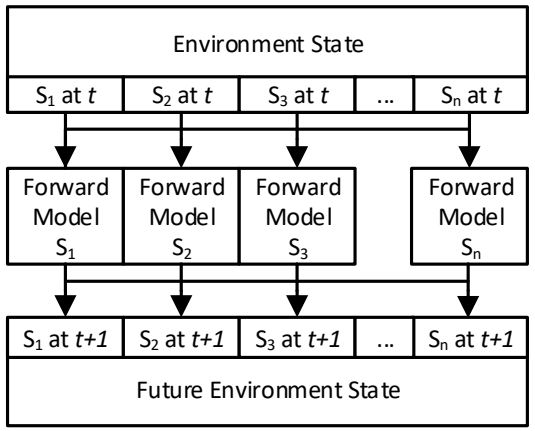
Modelling the complete state transition is very complex for a lot of applications

- ... 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

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

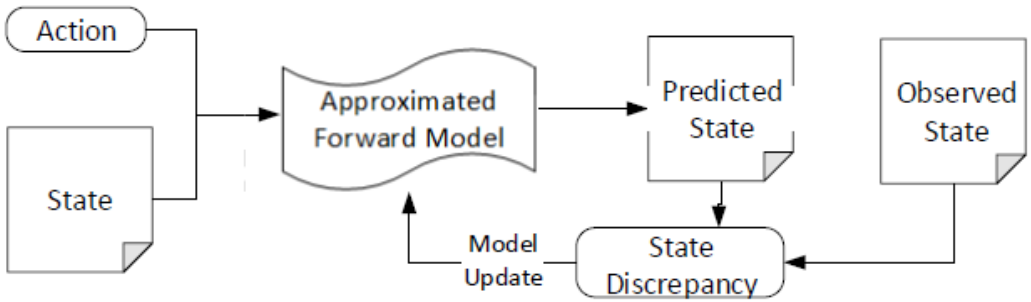
We can do this for all variables at once by learning a belief net structure.

- it encodes the necessary independencies for all involved nodes

Generating a Data Set

Forward Model Approximation is an iterative framework

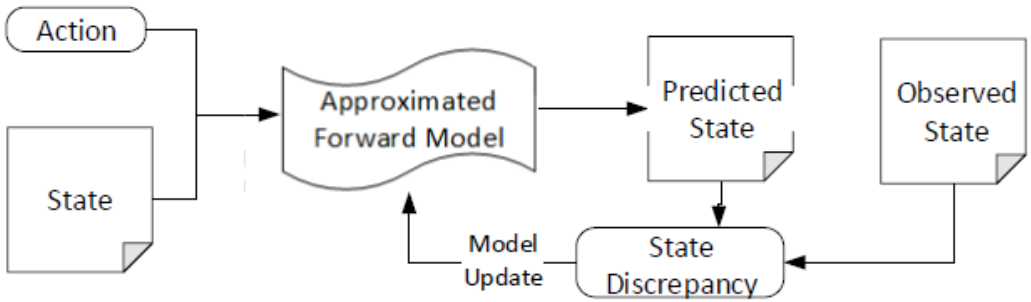
- our data set of observations grows over time
- model building needs to be repeated in case of errors



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.

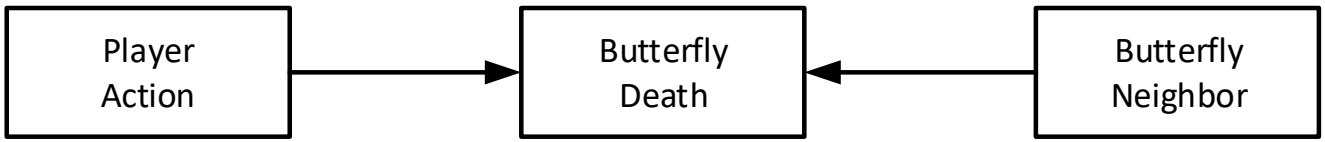
- contains an object's position, neighbors, movement and death

Results I/II

Found substructures contain:

- position changes

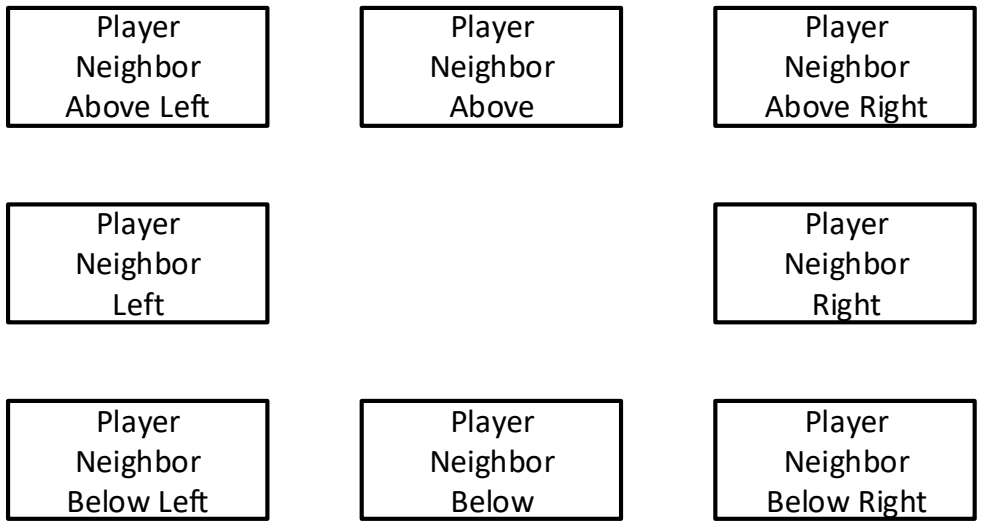
- explanations for an component's death



Results II/II

Some patterns evolve over time when new data becomes available

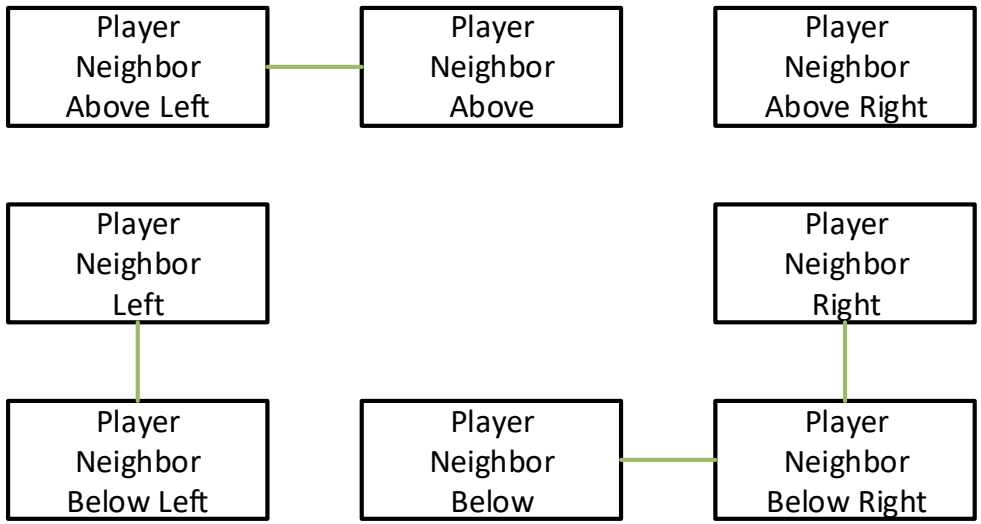
- e.g. neighboring relations
- starting without any connection



Results II/II

Some patterns evolve over time when new data becomes available

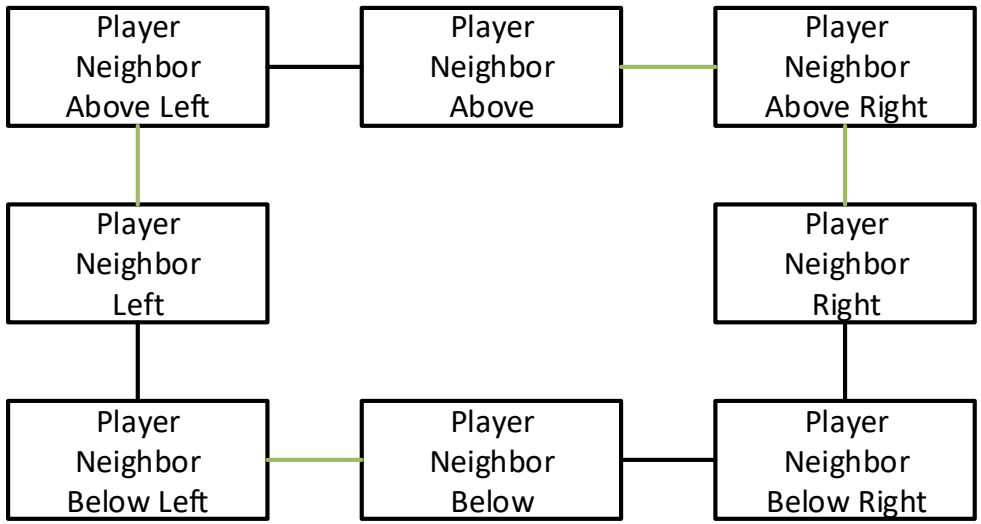
- e.g. neighboring relations
- starting without any connection
- adding edges dependencies over time



Results II/II

Some patterns evolve over time when new data becomes available

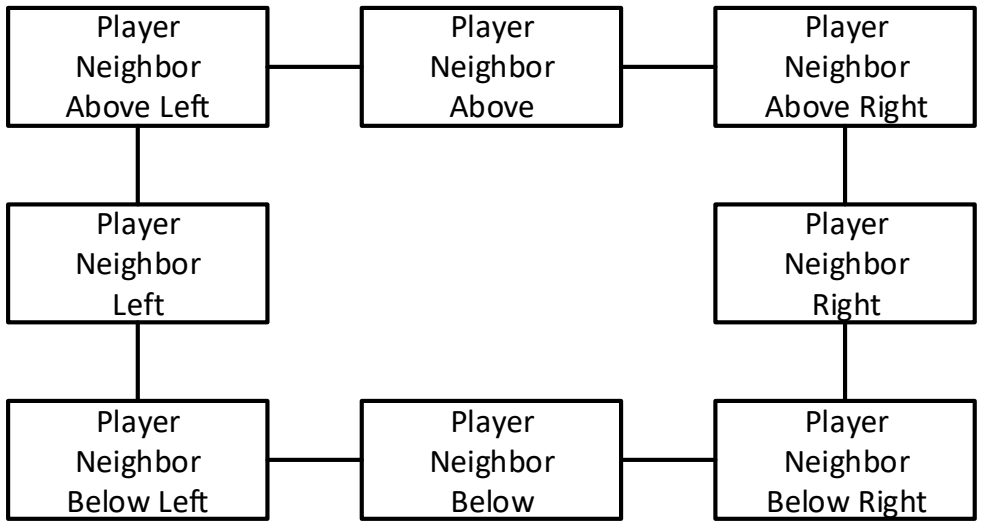
- e.g. neighboring relations
- starting without any connection
- adding edges dependencies over time



Results II/II

Some patterns evolve over time when new data becomes available

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



Limitations and Open Research Questions

Can we merge similar sub-structures?

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

Limitations and Open Research Questions

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

Limitations and Open Research Questions

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

Part 2: Game State Approximation

Developing techniques for approximating the current state by using knowledge of previous gameplays

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.

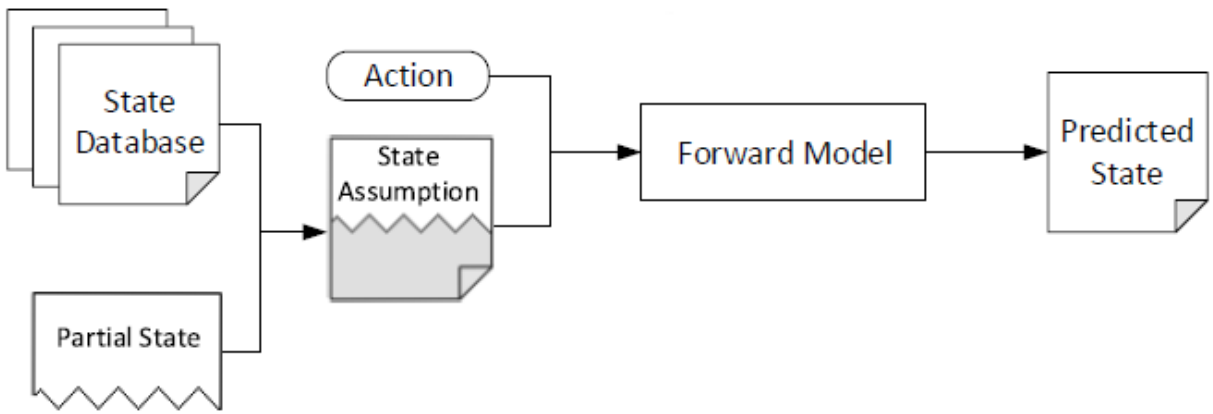
- apply simulation-based search with the estimated state

Idea:

Can observable sensor variables be used to guess the gamestate?

Dependency analysis:

- find dependencies between observable and unobservable variables




HEARTHSTONE
HEROES OF WARCRAFT

Hearthstone – Game Components and States

Why we cannot use SBS effectively?

- Simulation-based search needs a Forward Model for executing many simulations starting from the current state
  - We now the rules of the game, so the Forward Model is available
  - But we do not know the current state of the system!

- The player does not know the real state of the game, which consists of
  - The board state
  - The own and the opponent‘s cards
  - The cards in both decks
  - All previously played cards

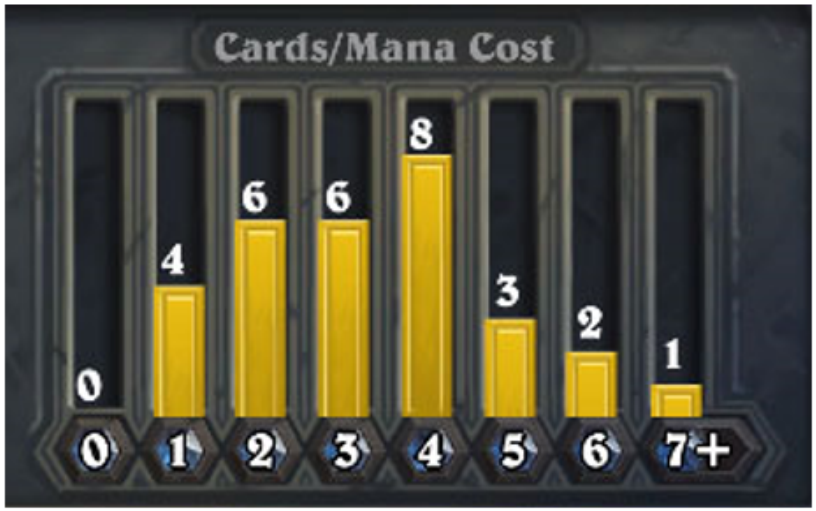
Deckbuilding

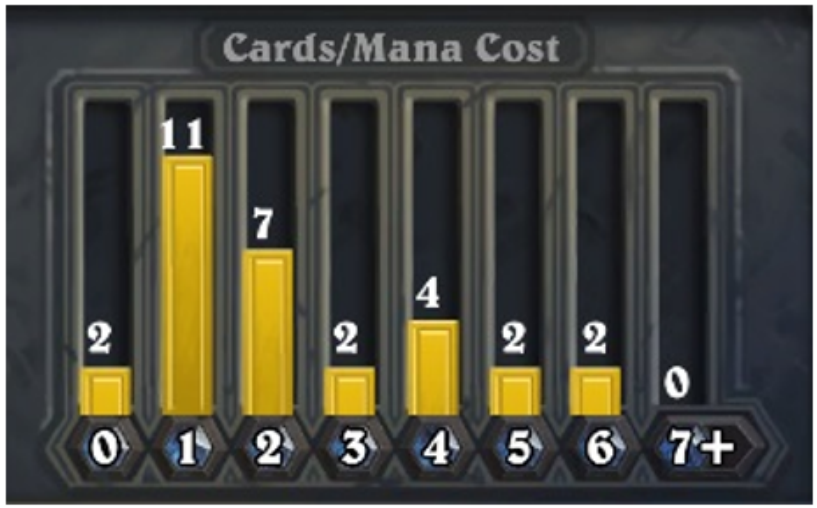
- By common knowledge popular decks often employ critical properties such as:
  - A stable mana curve
  - Strong synergies
  - A strategy to reach the win-condition

- Most decks can be categorized in deck types, which are common to the meta-game
  - Those decks often consist of similar cards, but do not necessarily contain the same 30 cards

Mana Curves

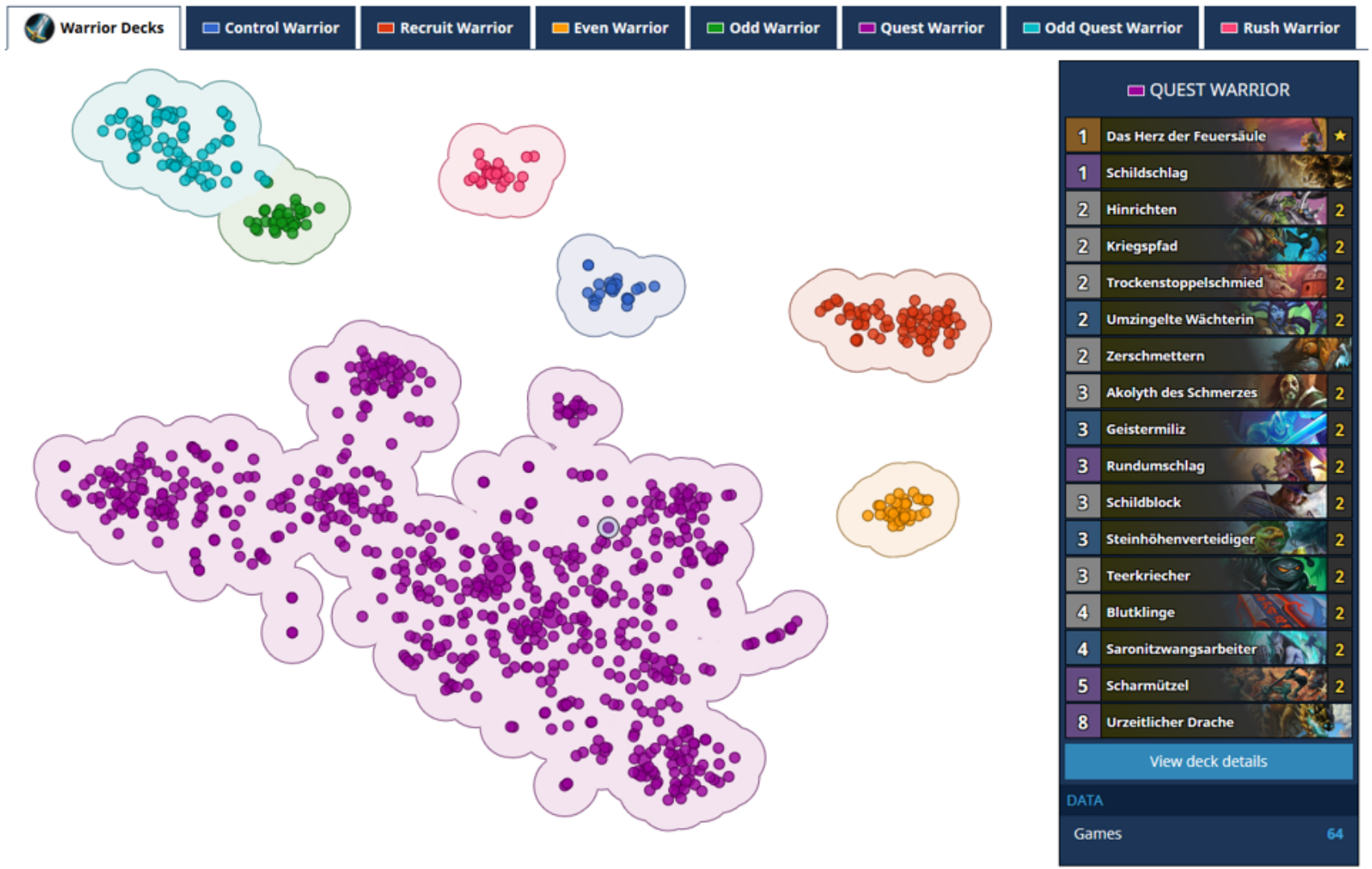
- The players can play their cards using mana
- Each turn the pool of available mana increases by one
- The mana curve of a deck describes how much cards per cost are included in the deck





Card Synergies

Meta-Decks


Warrior Decks
Control Warrior
Recruit Warrior
Even Warrior
Odd Warrior
Quest Warrior
Odd Quest Warrior
Rush Warrior
QUEST WARRIOR
1 Das Herz der Feuersäule ★
1 Schildschlag
2 Hinrichten 2
2 Kriegspfad 2
2 Trockenstoppelschmied 2
2 Umzingelte Wächterin 2
2 Zerschmettern
3 Akolyth des Schmerzes 2
3 Geistermiliz 2
3 Rundumschlag 2
3 Schildblock 2
3 Steinhöhenverteidiger 2
3 Teerkriecher 2
4 Blutklinge 2
4 Saronitzwangsarbeiter 2
5 Scharmützel 2
8 Urzeitlicher Drache
View deck details
DATA
Games 64

Finding Probable Card-Combinations

- Meta-decks and card synergies tell us that we have strong dependencies between cards in a deck
- We want to predict likely cards by observing frequent card-combinations in a database of played games
- Counting bi-grams of co-occuring cards:
  - **isolated**: cards need to be played at the same turn
  - **successive**: cards need to be played in successive turns
  - **combined**: isolated and successive counts are added
- How to incorporate the knowledge into the search process?

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS): Adding Monte Carlo simulations (or rollouts) after adding a new node to the tree.

Two different policies are used on each episode:

- **Tree Policy:** Selecting a node for expansion in the search tree (e.g. Greedy, UCB).
- **Default Policy:** Control the simulations outside the tree (e.g. picking actions at random)

UCB1

$$a_t = \arg\max_{a \in A} \underbrace{Q_t(a)}_{\text{Exploitation}} + C \underbrace{\sqrt{\frac{\ln N(s)}{N(s,a)}}}_{\text{Exploration}}$$

Monte Carlo Tree Search - Tree Policy

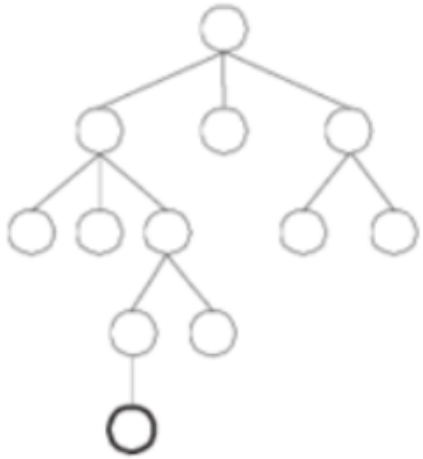
1. Tree Selection:

Following the **tree policy** (i.e. UCB1), navigate the tree until reaching a node with at least one child state that was not explored yet.

2. Expansion:

Add a new node in the tree, as a child of the node reached in the tree selection step.

- this node resembles an action

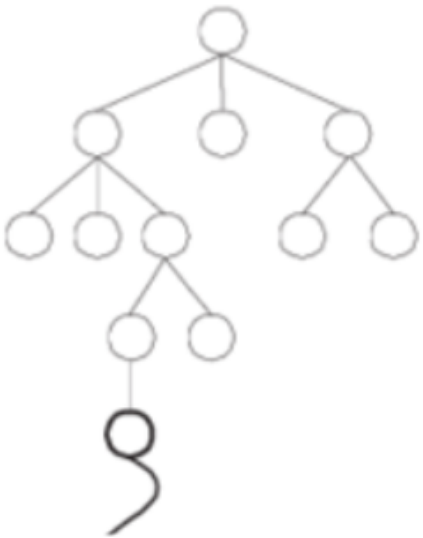


Monte Carlo Tree Search - Default Policy

3. Monte Carlo Simulation/Rollout:

Following the **default policy**, advance the state until a terminal state or a pre-defined maximum number of steps.

- calculate the return of this episode.



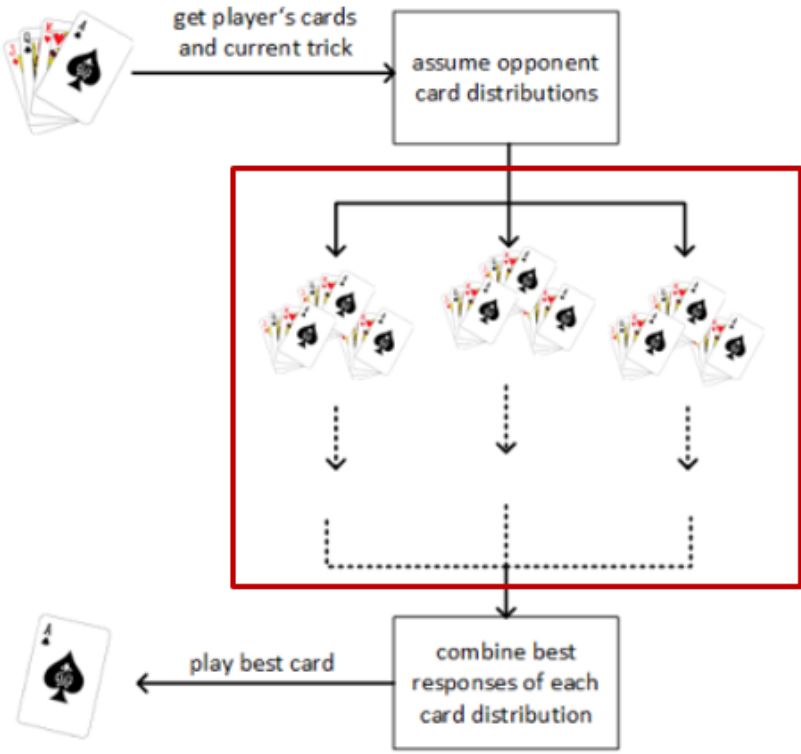
4. Back-propagation:

Update the values of $Q(s, a)$, $N(s)$ and $N(s, a)$ of the nodes visited in the tree during steps 1 and 2.

- Values of nodes visited during the simulation will not be updated.

MCTS for an Unknown Card Distribution

- If the real gamestate is unknown we can still guess multiple times:
  - information about the opponent's hero and cards that were already played restrict the open card pool
- Use and ensemble of MCTS agents with majority vote
- **This does not suffice in Hearthstone!**
  - **Too many possible card distributions**



Method used for Doppelkopf

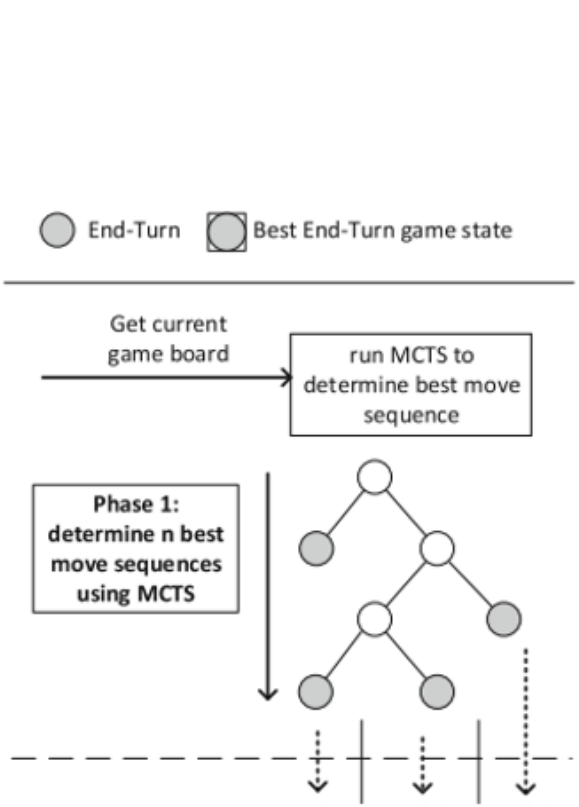
Adapting MCTS Gamestate Sampling

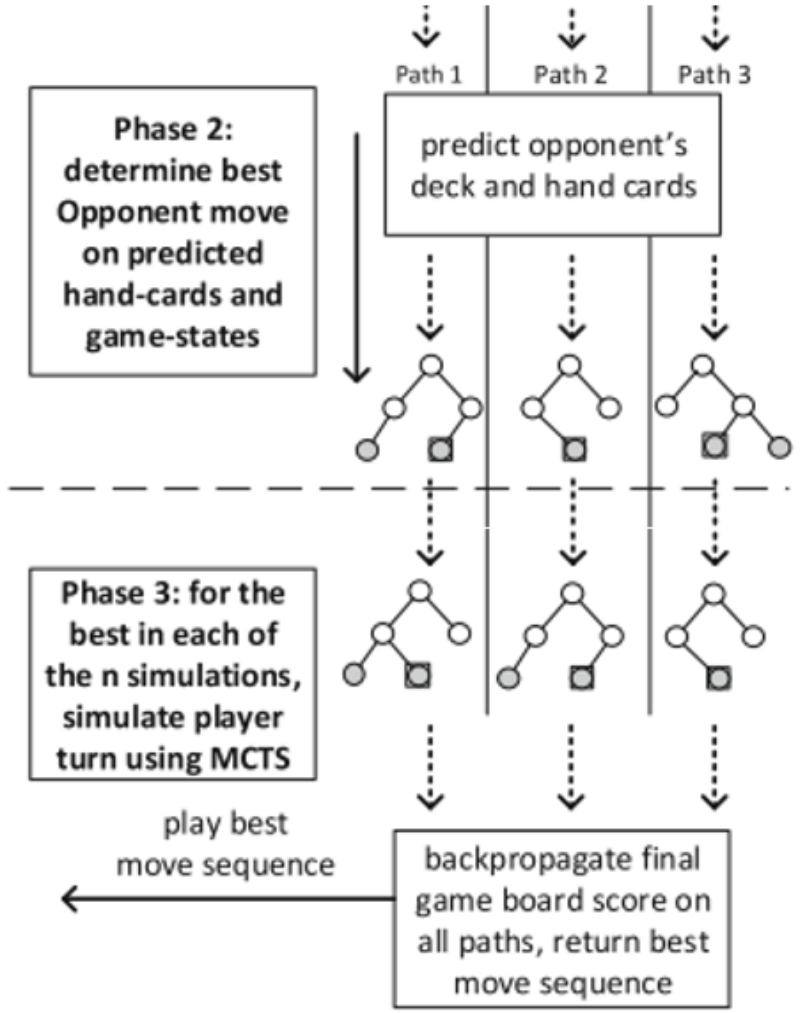
- A set of hand-cards is determined using the bi-gram database and all previously seen cards
  1. determine the number of co-occurences of each card
  2. filter possible cards by the rules of the deckbuilding process
  3. sample the opponent‘s hand cards based on the normalized co-occurence values

- 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

The overall Process





Evaluation

- 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

Conclusions

- Applying MCTS to Hearthstone is very limited due to missing information
  - Guessing a gamestate lets us apply MCTS with weak play strength
- Combining multiple of such guesses by creating an ensemble of MCTS helps but does not solve the problem efficient enough
- In this work we proposed a card-prediction to enhance the accuracy of sampling possible card distributions
  - An ensemble using such predicted gamestates is nearly as powerful as knowing the real gamestate

Limitations and Open Research Questions

- Our tests using an MCTS agent with full information on the current game-state show that a perfect prediction would yield slightly better results
  - Further improving the prediction accuracy promises to yield a stronger agent
- ➢ Explore the tradeoff between a complex prediction or more simulations/predicted card sets
- ➢ Can fuzzy sets and dempster-shafer theory help us to solve this problem more efficiently?

- We want to further explore meta-decks
  - How do they develop over time?
  - Can we detect changes in the meta and react accordingly?
- ➢ Important for deck-building and balancing!

Hearthstone AI Competition

Interested in trying it yourself? Check out our Hearthstone Competition:

http://www.is.ovgu.de/Research/HearthstoneAI.html

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