Prediction-based Search for Autonomous Game-Playing

PhD Defense Presentation

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

Motivation

Computational Intelligence refers to the ability of a computer to learn a specific task from data or experimental observation.

Motivation

Computational Intelligence refers to the ability of a computer to learn a specific task from data or experimental observation.

- steering of self-driving cars and parking aids

[1]

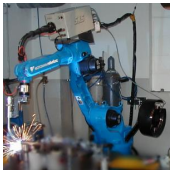
[1] https://commons.wikimedia.org/wiki/File:Tesla_Autopilot_Engaged_in_Model_X.jpg

Motivation

Computational Intelligence refers to the ability of a computer to learn a specific task from data or experimental observation.

- steering of self-driving cars and parking aids
- automating a production pipeline

[1]



[2]

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

[2] https://commons.wikimedia.org/wiki/File:Industrieroboter.jpg

Motivation

Computational Intelligence refers to the ability of a computer to learn a specific task from data or experimental observation.

- steering of self-driving cars and parking aids
- automating a production pipeline

Problems:

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

Motivation

Computational Intelligence refers to the ability of a computer to learn a specific task from data or experimental observation.

- steering of self-driving cars and parking aids
- automating a production pipeline

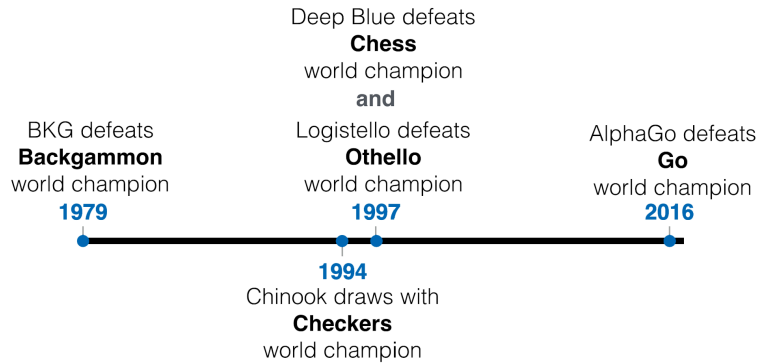
Problems:

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

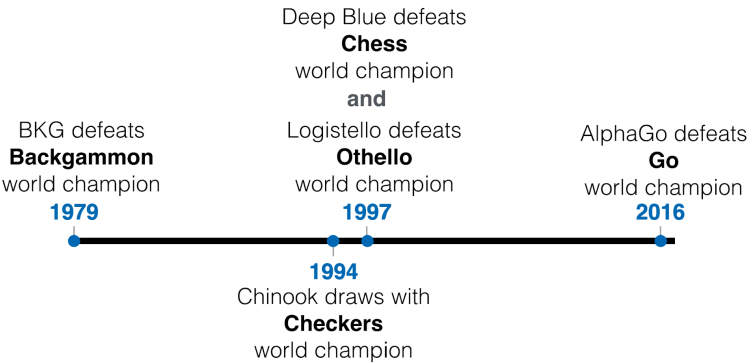
Games can be simulations of real world tasks

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

A Short History of Computational Intelligence in Games



A Short History of Computational Intelligence in Games



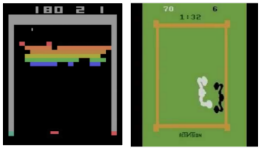
heuristic solutions, planning agents and **reinforcement learning agents**

A Short History of Computational Intelligence in Games

ATARI

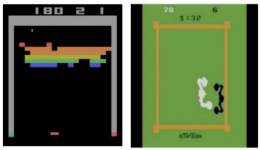
StarCraft II

DOTA 2



A Short History of Computational Intelligence in Games

ATARI

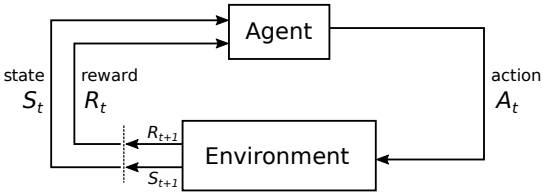


STARCRAFT II

DOTA 2

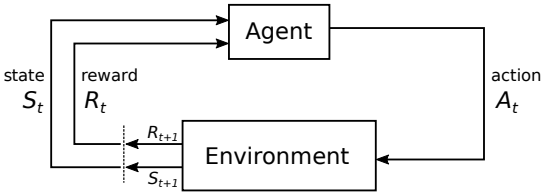
these were achieved using **reinforcement learning** algorithms and **months of training**

Agent-Environment Interface



A general framework for studying games which consists of the elements:

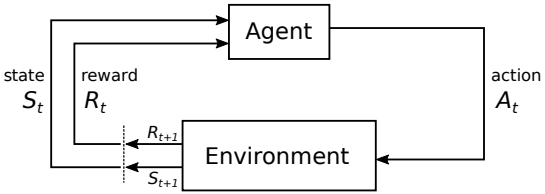
Agent-Environment Interface



A general framework for studying games which consists of the elements:

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

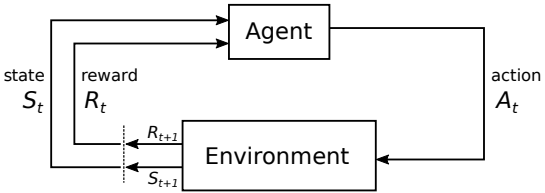
Agent-Environment Interface



A general framework for studying games which consists of the elements:

- **Agent:** the learner and decision-maker
- **Environment:** anything the agent interacts with, e.g. a game

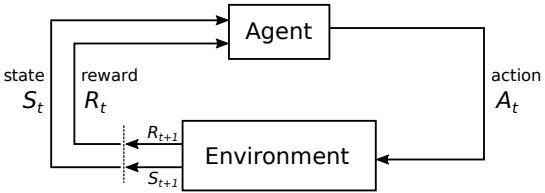
Agent-Environment Interface



A general framework for studying games which consists of the elements:

- **Agent:** the learner and decision-maker
- **Environment:** anything the agent interacts with, e.g. a game
- **Actions:** Agent and environment interact continuously interact with each other.
  - the agent selects an action
  - the environment responds to those actions

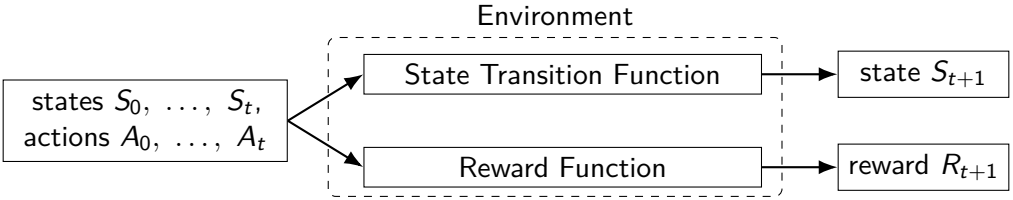
Agent-Environment Interface



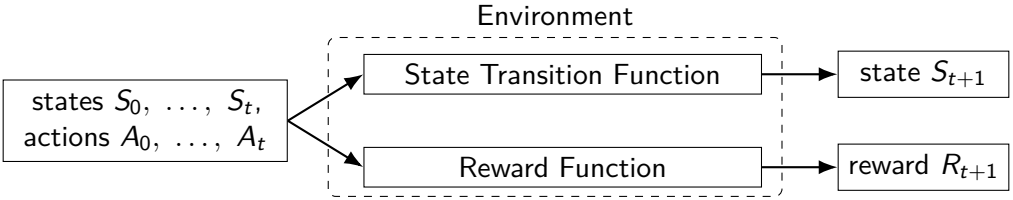
A general framework for studying games which consists of the elements:

- **Agent:** the learner and decision-maker
- **Environment:** anything the agent interacts with, e.g. a game
- **Actions:** Agent and environment interact continuously interact with each other.
  - the agent selects an action
  - the environment responds to those actions
- **Reward:** numerical values provided that the agent tries to maximize over time.

Components of a Game



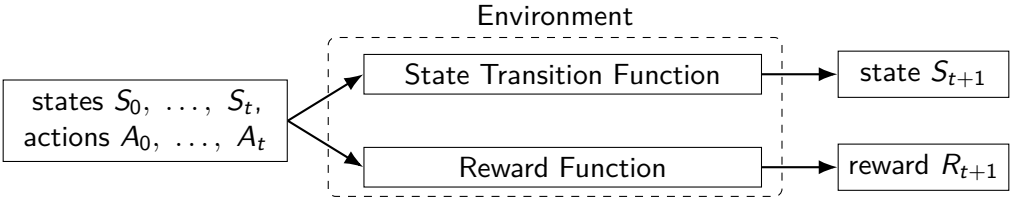
Components of a Game



State $S_t \in \mathcal{S}$ can be perceived through multiple sensors $(S_t^{(1)}, S_t^{(2)}, \ldots, S_t^{(n)})$.

- a state may not be fully observable (partial information game)

Components of a Game



State $S_t \in \mathcal{S}$ can be perceived through multiple sensors $(S_t^{(1)}, S_t^{(2)}, \ldots, S_t^{(n)})$.

- a state may not be fully observable (partial information game)

The whole environment can be modelled as a probability distribution of possible outcomes:

$$P(R_{t+1}, S_{t+1} \mid S_0,\ A_0,\ S_1,\ A_1,\ \ldots,\ S_t,\ A_t)$$

- but we can also model both components separately

Methods for Computational Intelligence in Games

Methods for Computational Intelligence in Games

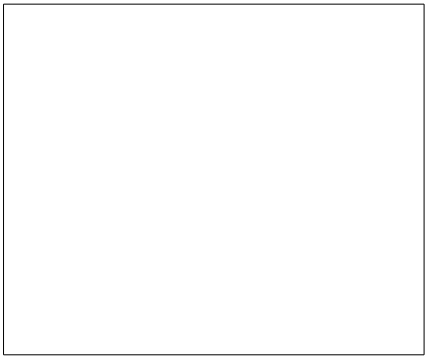
two popular learning approaches:

- learn which actions are good
- learn to anticipate the future

Methods for Computational Intelligence in Games

two popular learning approaches:

- learn which actions are good
- learn to anticipate the future



Methods for Computational Intelligence in Games

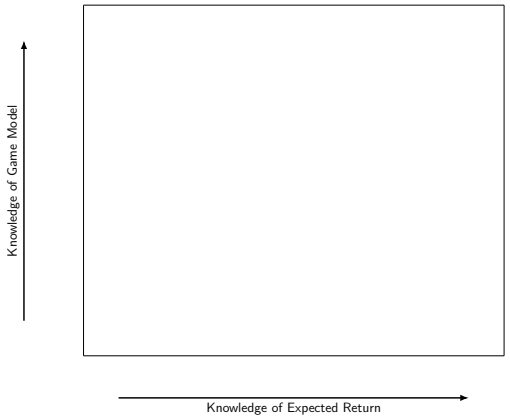
two popular learning approaches:

- learn which actions are good
- learn to anticipate the future

Methods for Computational Intelligence in Games

two popular learning approaches:

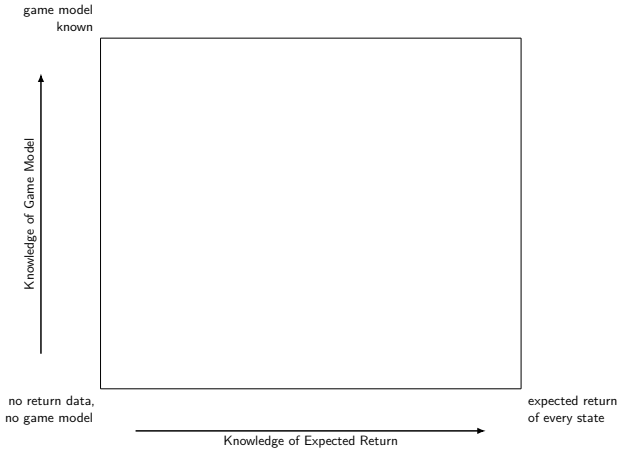
- learn which actions are good
- learn to anticipate the future



Methods for Computational Intelligence in Games

two popular learning approaches:

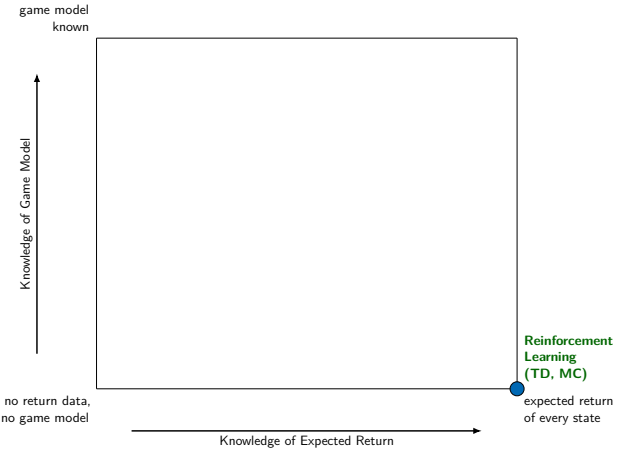
- learn which actions are good
- learn to anticipate the future



Methods for Computational Intelligence in Games

two popular learning approaches:

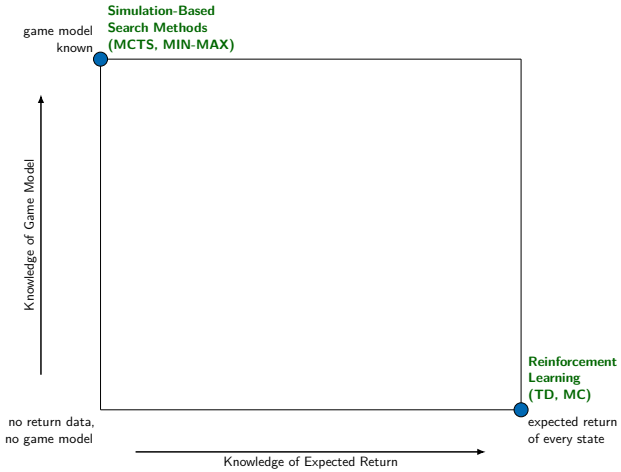
- learn which actions are good
- learn to anticipate the future



Methods for Computational Intelligence in Games

two popular learning approaches:

- learn which actions are good
- learn to anticipate the future



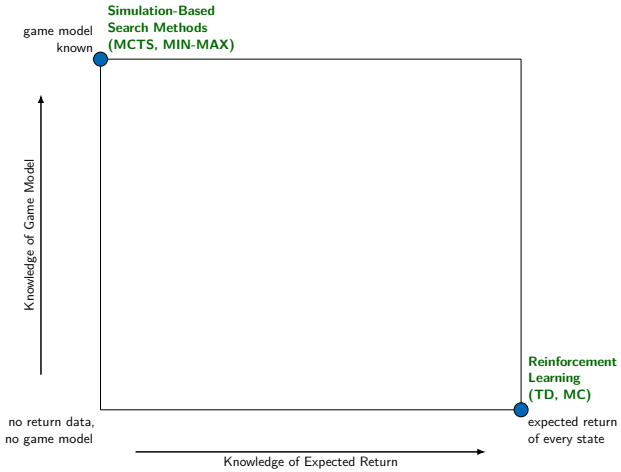
Methods for Computational Intelligence in Games

two popular learning approaches:

- learn which actions are good
- learn to anticipate the future

Reinforcement Learning

- performance depends on the available training time



Methods for Computational Intelligence in Games

two popular learning approaches:

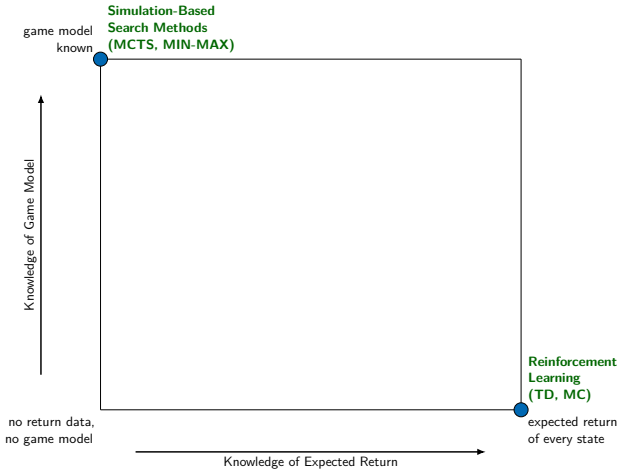
- learn which actions are good
- learn to anticipate the future

Reinforcement Learning

- performance depends on the available training time

Simulation-based Search

- performance depends on the available computation time during evaluation



Methods for Computational Intelligence in Games

two popular learning approaches:

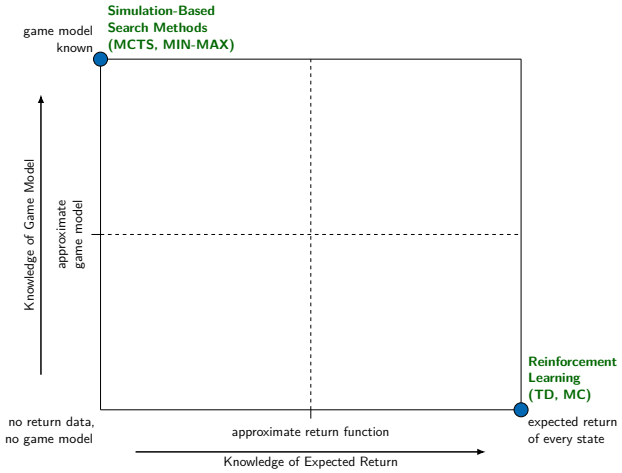
- learn which actions are good
- learn to anticipate the future

Reinforcement Learning

- performance depends on the available training time

Simulation-based Search

- performance depends on the available computation time during evaluation



Methods for Computational Intelligence in Games

two popular learning approaches:

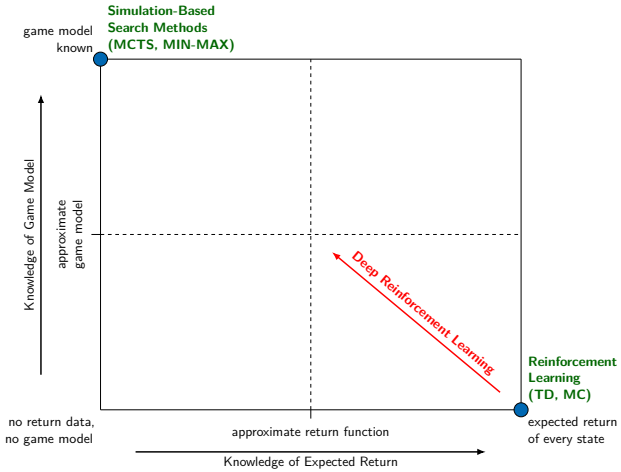
- learn which actions are good
- learn to anticipate the future

Reinforcement Learning

- performance depends on the available training time

Simulation-based Search

- performance depends on the available computation time during evaluation



Methods for Computational Intelligence in Games

two popular learning approaches:

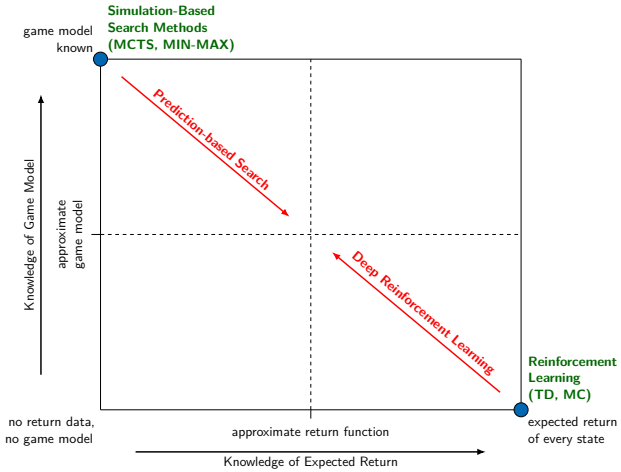
- learn which actions are good
- learn to anticipate the future

Reinforcement Learning

- performance depends on the available training time

Simulation-based Search

- performance depends on the available computation time during evaluation



Simulation-Based Search Algorithms

Simulation-Based Search Algorithms

Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms

current state

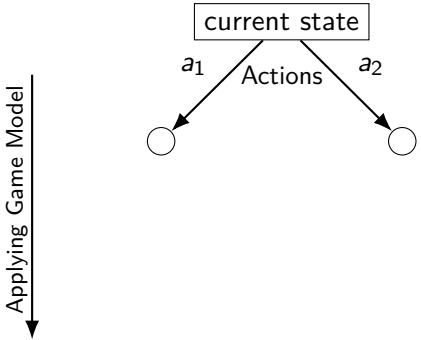
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



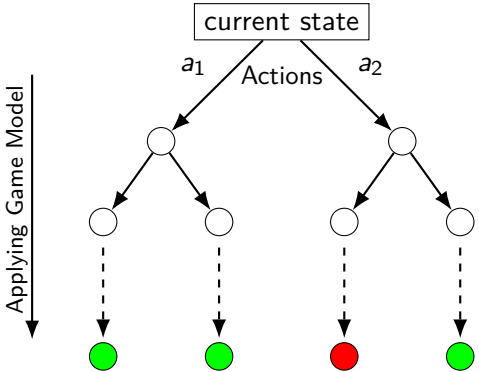
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



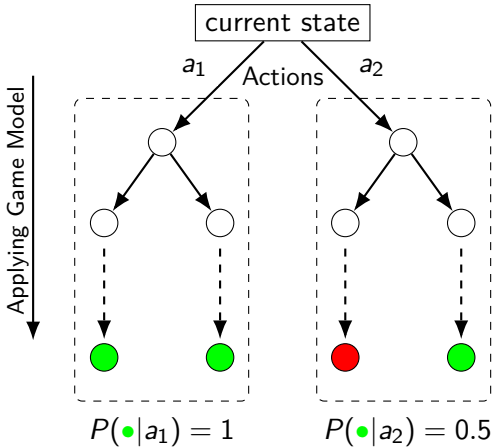
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



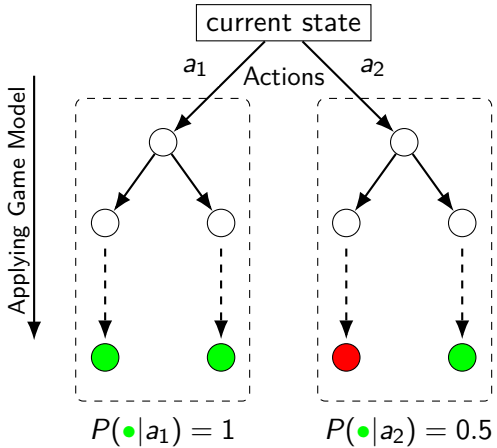
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



Input:

- current state
- game-model

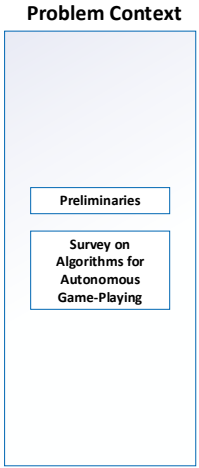
Output:

- action with highest win-rate

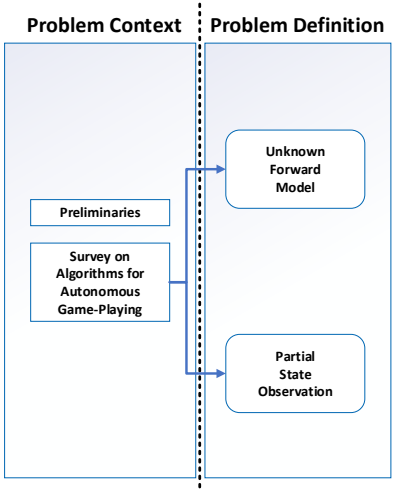
How to handle games for which:

- the game model is unknown?
- the state cannot be fully observed?

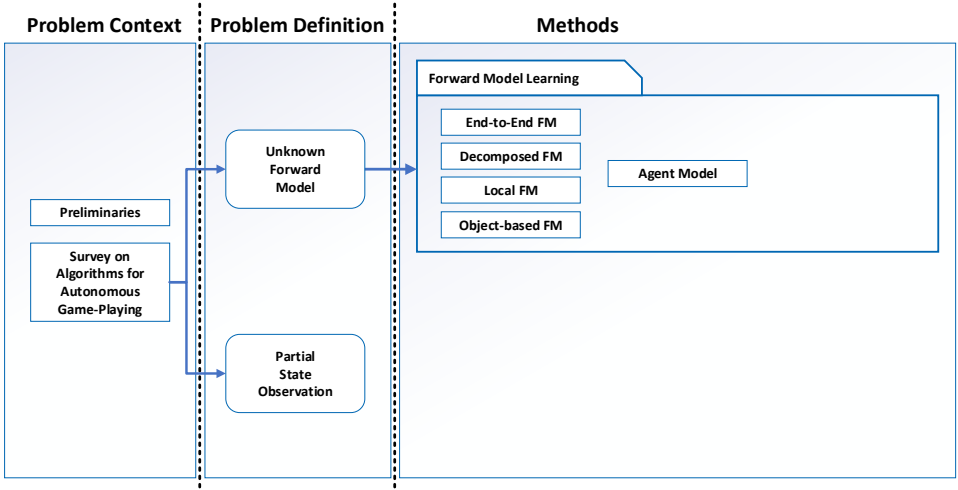
Graphical Overview of Contents



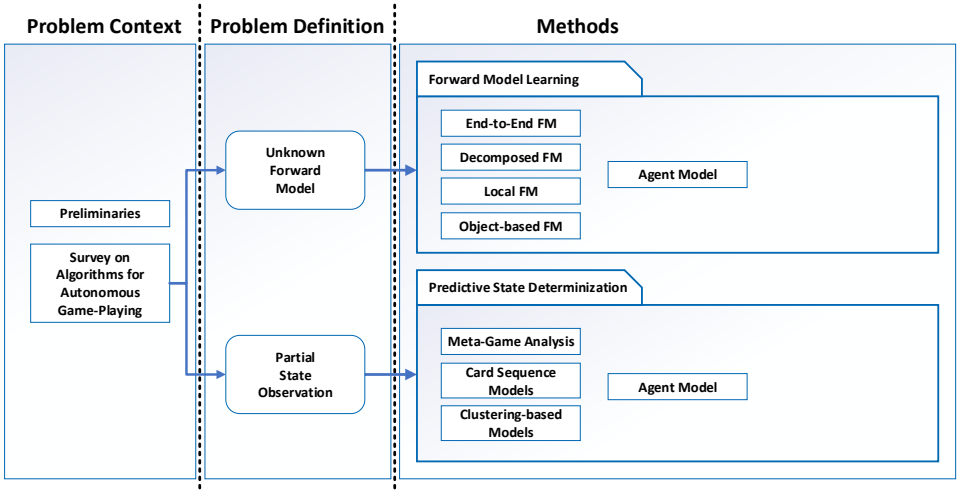
Graphical Overview of Contents



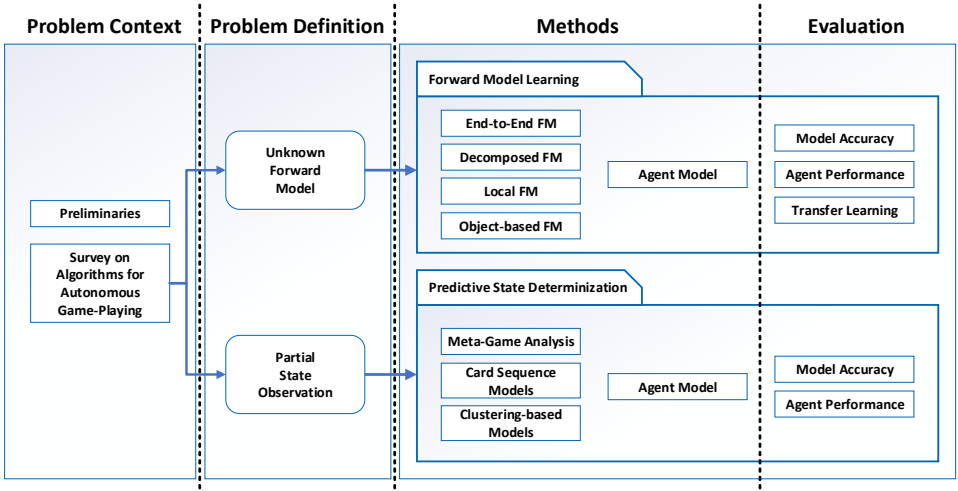
Graphical Overview of Contents



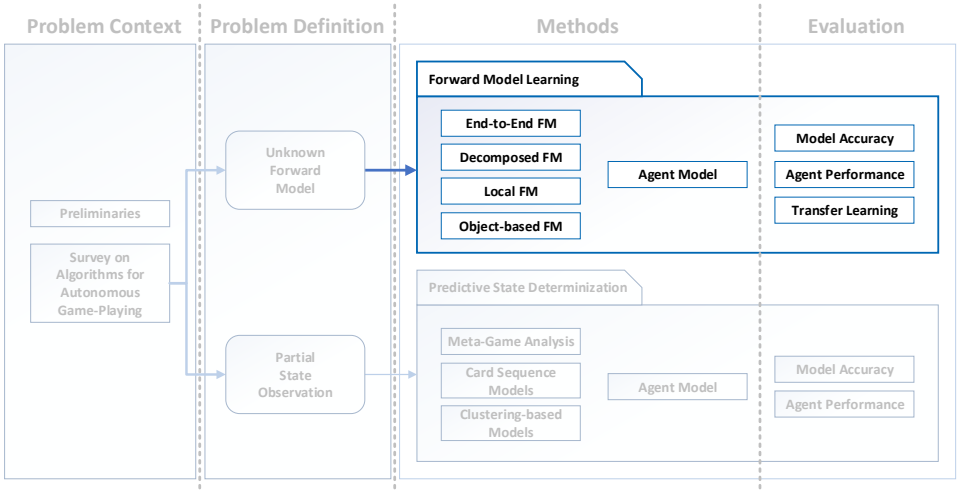
Graphical Overview of Contents



Graphical Overview of Contents



Graphical Overview of Contents



Forward Model Learning

Forward Model Learning

Goal: Learn to predict upcoming states of the environment.

Forward Model Learning

Goal: Learn to predict upcoming states of the environment.

Definition: Forward Model

- A forward model *fm* maps the environment's state S_t and the agent's action A_t at time t to the upcoming state S_{t+1} of the environment:

$$fm : (\mathcal{S} \times \mathcal{A}) \to \mathcal{S} \qquad (S_t, A_t) \longmapsto S_{t+1}$$

Forward Model Learning

Goal: Learn to predict upcoming states of the environment.

Definition: Forward Model

- A forward model *fm* maps the environment's state S_t and the agent's action A_t at time t to the upcoming state S_{t+1} of the environment:

$$fm : (\mathcal{S} \times \mathcal{A}) \to \mathcal{S} \qquad (S_t, A_t) \longmapsto S_{t+1}$$

- This definition only applies to environment models that fulfil the Markov property.

Forward Model Learning

Goal: Learn to predict upcoming states of the environment.

Definition: Forward Model

- A forward model *fm* maps the environment's state S_t and the agent's action A_t at time t to the upcoming state S_{t+1} of the environment:

$$fm : (\mathcal{S} \times \mathcal{A}) \to \mathcal{S} \qquad (S_t, A_t) \longmapsto S_{t+1}$$

- This definition only applies to environment models that fulfil the Markov property.

Markov Property:

- The environment fulfills the Markov property in case the upcoming state is independent of all states but the present state.

$$P(S_{t+1} \mid S_0,\ A_0,\ S_1,\ A_1,\ \ldots,\ S_t,\ A_t) \quad \Rightarrow \quad P(S_{t+1} \mid S_t,\ A_t)$$

End-To-End Forward Models

End-To-End Forward Models

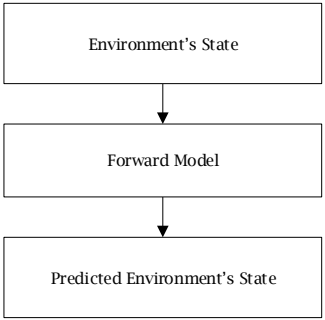
Problem Categorization:

- choose the next state among the entirety of states
- either a classification or regression problem

End-To-End Forward Models

Problem Categorization:

- choose the next state among the entirety of states
- either a classification or regression problem



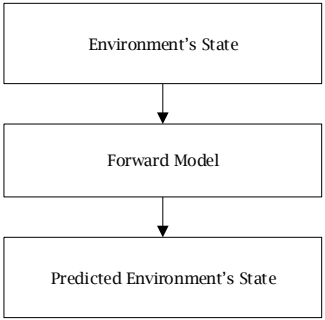
End-To-End Forward Models

Problem Categorization:

- choose the next state among the entirety of states
- either a classification or regression problem

Learning a model of the game:

- Gather experience while playing. Each observed state-transition equals one training example.
- Train a model given all observations.



End-To-End Forward Models

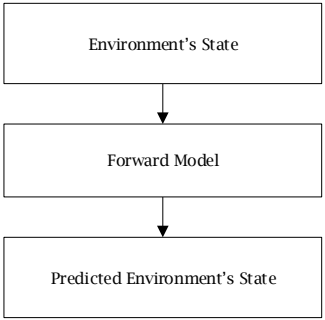
Problem Categorization:

- choose the next state among the entirety of states
- either a classification or regression problem

Learning a model of the game:

- Gather experience while playing. Each observed state-transition equals one training example.
- Train a model given all observations.

$\Rightarrow$ The number of required examples can be dependent on the complexity of the state and action space



End-To-End Forward Models

Problem Categorization:

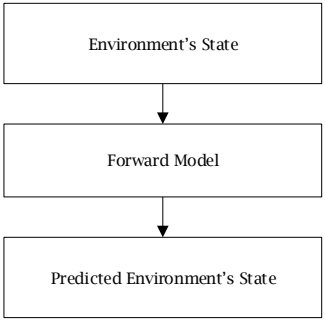
- choose the next state among the entirety of states
- either a classification or regression problem

Learning a model of the game:

- Gather experience while playing. Each observed state-transition equals one training example.
- Train a model given all observations.

$\Rightarrow$ The number of required examples can be dependent on the complexity of the state and action space

The following methods represent attempts to reduce the complexity of the learning problem.



Decomposed Forward Models[1]

Assumptions:

- sensor values can be modelled independently

$$\forall i, j \in 1..n : i \neq j \Rightarrow S_{t+1}^{(i)} \perp\!\!\!\perp S_{t+1}^{(j)} \mid S_t, A_t$$

[1] Dockhorn, A., Tippelt, T., & Kruse, R. (2018). Model Decomposition for Forward Model Approximation

Decomposed Forward Models[1]

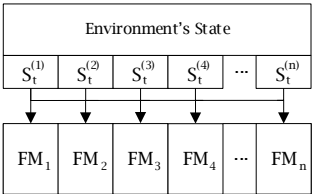
Assumptions:

- sensor values can be modelled independently

$$\forall i, j \in 1..n : i \neq j \Rightarrow S_{t+1}^{(i)} \perp\!\!\!\perp S_{t+1}^{(j)} \mid S_t, A_t$$

Learn one sub-model for each observable sensor value

$$\underset{i}{fm} : (S_t,\ A_t) \longmapsto S_{t+1}^{(i)}$$



[1] Dockhorn, A., Tippelt, T., & Kruse, R. (2018). Model Decomposition for Forward Model Approximation

Decomposed Forward Models[1]

Assumptions:

- sensor values can be modelled independently

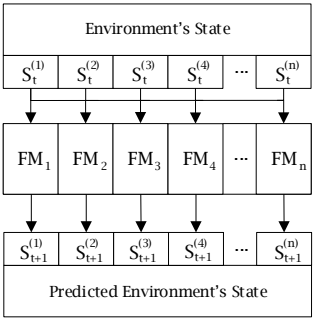
$$\forall i, j \in 1..n : i \neq j \Rightarrow S_{t+1}^{(i)} \perp\!\!\!\perp S_{t+1}^{(j)} \mid S_t, A_t$$

Learn one sub-model for each observable sensor value

$$\underset{i}{fm} : (S_t,\ A_t) \longmapsto S_{t+1}^{(i)}$$

Aggregate the result of each sensor value prediction

$$fm(S_t, A_t) = (fm_1(S_t, A_t),\ fm_2(S_t, A_t),\ \ldots,\ fm_n(S_t, A_t))$$
$$= (S_{t+1}^{(1)},\ S_{t+1}^{(2)},\ \ldots,\ S_{t+1}^{(n)}) = S_{t+1}$$



[1] Dockhorn, A., Tippelt, T., & Kruse, R. (2018). Model Decomposition for Forward Model Approximation

Modelling Local Dependencies

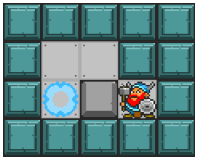
Assumptions:

- structured representation of the state
- requires a similarity or distance function for sensor values
- semantic of a sensor-value is independent of its index

Modelling Local Dependencies

Assumptions:

- structured representation of the state
- requires a similarity or distance function for sensor values
- semantic of a sensor-value is independent of its index

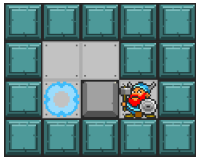


Game-State of Sokoban

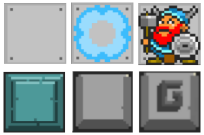
Modelling Local Dependencies

Assumptions:

- structured representation of the state
- requires a similarity or distance function for sensor values
- semantic of a sensor-value is independent of its index



Game-State of Sokoban

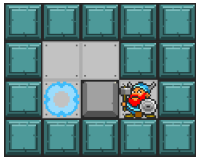


Tilemap Components

Modelling Local Dependencies

Assumptions:

- structured representation of the state
- requires a similarity or distance function for sensor values
- semantic of a sensor-value is independent of its index



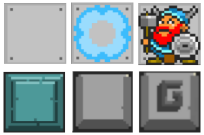
Game-State of Sokoban

Tile-based Representation (of Video Games):

- a state can be represented as a matrix T of size $n \times m$

$$T = \begin{bmatrix} T(1,1) & \dots & T(1,m) \\ \vdots & \ddots & \vdots \\ T(n,1) & \dots & T(n,m) \end{bmatrix}$$

- $T(x,y)$ specifies the observed tile at position (x,y)



Tilemap Components

Local Transition Function

Decompose the forward model into one sub-model per tile:

Local Transition Function

Decompose the forward model into one sub-model per tile:

$$\underset{x,y}{fm} : \Big(N(x,y)_t,\ A_t \Big) \longmapsto T(x,y)_{t+1}$$

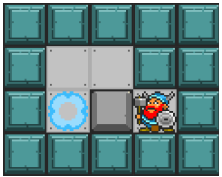
- $N(x,y)_t$ describes the local neighbourhood of tile $T(x,y)$ at time t
- it contains each tile with distance less than a given threshold

Local Transition Function

Decompose the forward model into one sub-model per tile:

$$\underset{x,y}{fm} : \Big(N(x,y)_t,\ A_t \Big) \longmapsto T(x,y)_{t+1}$$

- $N(x,y)_t$ describes the local neighbourhood of tile $T(x,y)$ at time t
- it contains each tile with distance less than a given threshold



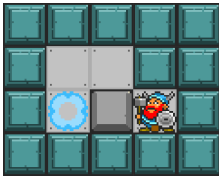
Game-State of Sokoban

Local Transition Function

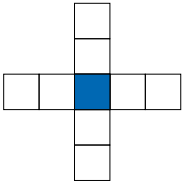
Decompose the forward model into one sub-model per tile:

$$\underset{x,y}{fm} : \Big(N(x,y)_t,\ A_t \Big) \longmapsto T(x,y)_{t+1}$$

- $N(x,y)_t$ describes the local neighbourhood of tile $T(x,y)$ at time t
- it contains each tile with distance less than a given threshold



Game-State of Sokoban



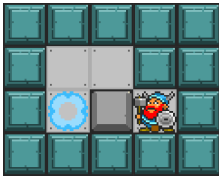
Local Neighborhood

Local Transition Function

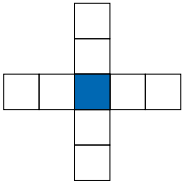
Decompose the forward model into one sub-model per tile:

$$\underset{x,y}{fm} : \Big(N(x,y)_t,\ A_t \Big) \longmapsto T(x,y)_{t+1}$$

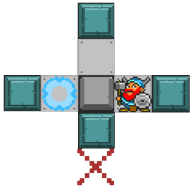
- $N(x,y)_t$ describes the local neighbourhood of tile $T(x,y)$ at time t
- it contains each tile with distance less than a given threshold



Game-State of Sokoban



Local Neighborhood



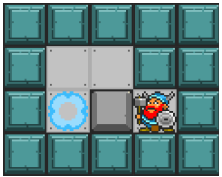
Extracted Pattern

Local Transition Function

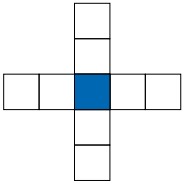
Decompose the forward model into one sub-model per tile:

$$\underset{x,y}{fm} : \Big(N(x,y)_t,\ A_t \Big) \longmapsto T(x,y)_{t+1}$$

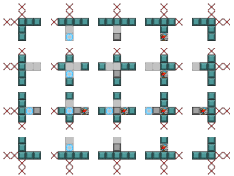
- $N(x,y)_t$ describes the local neighbourhood of tile $T(x,y)$ at time t
- it contains each tile with distance less than a given threshold



Game-State of Sokoban



Local Neighborhood



Extracted Patterns

Local Forward Model

Predict the next state by predicting each tile

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1,1), A_t) & \dots & fm_{1,m}(N(1,m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n,1), A_t) & \dots & fm_{n,m}(N(n,m), A_t) \end{bmatrix}$$

Local Forward Model

Predict the next state by predicting each tile

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1,1), A_t) & \dots & fm_{1,m}(N(1,m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n,1), A_t) & \dots & fm_{n,m}(N(n,m), A_t) \end{bmatrix}$$

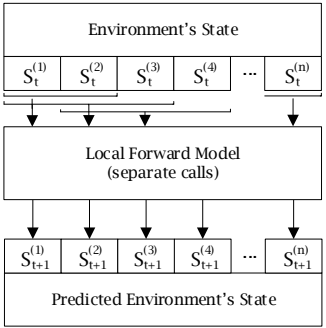
In case the semantic of a tile is independent of its position, only a single model needs to be learned

Local Forward Model

Predict the next state by predicting each tile

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1,1), A_t) & \dots & fm_{1,m}(N(1,m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n,1), A_t) & \dots & fm_{n,m}(N(n,m), A_t) \end{bmatrix}$$

In case the semantic of a tile is independent of its position, only a single model needs to be learned

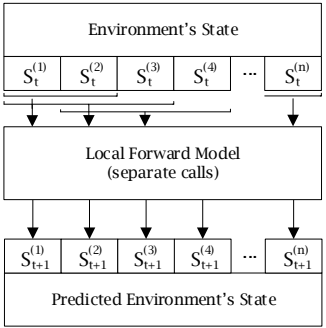


Local Forward Model

Predict the next state by predicting each tile

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1,1), A_t) & \dots & fm_{1,m}(N(1,m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n,1), A_t) & \dots & fm_{n,m}(N(n,m), A_t) \end{bmatrix}$$

In case the semantic of a tile is independent of its position, only a single model needs to be learned



Advantage: higher sampling efficiency

- each observed state transition consists of one observed pattern per tile (in total: $n \times m$ patterns)

Modelling Entities of a Game I/II

Measuring the importance of each tile in the local neighborhood indicates a high importance of the center tile.

$\Rightarrow$ instead of modelling the change of each position, model the change of represented objects

Feature Importance of Neighborhood Tiles

Modelling Entities of a Game I/II

Measuring the importance of each tile in the local neighborhood indicates a high importance of the center tile.

$\Rightarrow$ instead of modelling the change of each position, model the change of represented objects

Object-based Representation

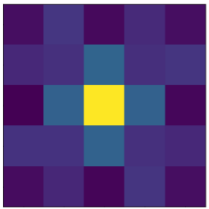
- the state consists of multiple entities of which several attributes can be observed

Feature Importance of Neighborhood Tiles

Modelling Entities of a Game I/II

Measuring the importance of each tile in the local neighborhood indicates a high importance of the center tile.

$\Rightarrow$ instead of modelling the change of each position, model the change of represented objects



Feature Importance of Neighborhood Tiles

Object-based Representation

- the state consists of multiple entities of which several attributes can be observed

$$\begin{aligned} \mathcal{S} &= (S^{(1)}, S^{(2)}, \ldots, S^{(n)}) \\ &= (\underbrace{S^{(1,1)}, \ldots, S^{(1,i)}}_{\text{Object 1}}, \underbrace{S^{(2,1)}, \ldots, S^{(2,j)}}_{\text{Object 2}}, \ldots, \underbrace{S^{(m,1)}, \ldots, S^{(m,k)}}_{\text{Object m}}) \end{aligned}$$

Modelling Entities of a Game II/II

Assumptions:

- game components are considered to represent indepently acting entities
- similar looking objects exhibit similar behavior

Modelling Entities of a Game II/II

Assumptions:

- game components are considered to represent indepently acting entities
- similar looking objects exhibit similar behavior

Create one model for each entity or entity type

$$\underset{i}{fm} : \left((S_t^{(i,1)}, \ldots, S_t^{(i,k)}),\ A_t \right) \longmapsto (S_{t+1}^{(i,1)},\ \ldots,\ S_{t+1}^{(i,k)})$$

Modelling Entities of a Game II/II

Assumptions:

- game components are considered to represent indepently acting entities
- similar looking objects exhibit similar behavior

Create one model for each entity or entity type

$$\underset{i}{fm} : \left((S_t^{(i,1)}, \ldots, S_t^{(i,k)}),\ A_t \right) \longmapsto (S_{t+1}^{(i,1)},\ \ldots,\ S_{t+1}^{(i,k)})$$

Complex entities can be modelled using a decomposed forward model

- create one model for each observable sensor value

$$\underset{i,j}{fm} : \left((S_t^{(i,1)}, \ldots, S_t^{(i,k)}),\ A_t \right) \longmapsto S_{t+1}^{(i,j)}$$

Object-based Forward Model

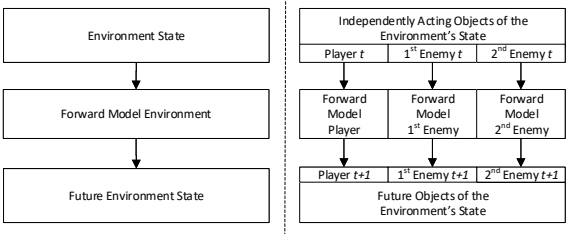
Aggregate the prediction of each object and its associated sensor values:

$$fm(S_t, A_t) = (\underset{1}{fm}(S_t, A_t),\ \ldots,\ \underset{n}{fm}(S_t, A_t))$$

Object-based Forward Model

Aggregate the prediction of each object and its associated sensor values:

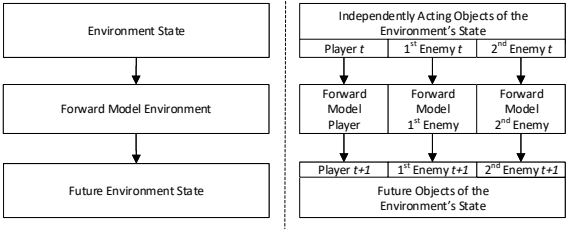
$$fm(S_t, A_t) = (\underset{1}{fm}(S_t, A_t),\ \ldots,\ \underset{n}{fm}(S_t, A_t))$$



Object-based Forward Model

Aggregate the prediction of each object and its associated sensor values:

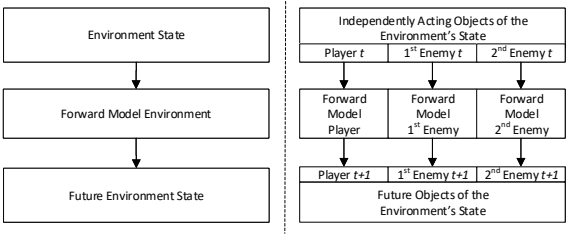
$$
\begin{aligned}
fm(S_t, A_t) &= (\underset{1}{fm}(S_t, A_t),\ \ldots,\ \underset{n}{fm}(S_t, A_t)) \\
&= ((\underset{1,1}{fm}((S_t^{(1,1)}, \ldots, S_t^{(1,k)}), A_t),\ \ldots,\ \underset{m,k}{fm}((S_t^{(m,1)}, \ldots, S_t^{(m,k')}), A_t)))
\end{aligned}
$$



Object-based Forward Model

Aggregate the prediction of each object and its associated sensor values:

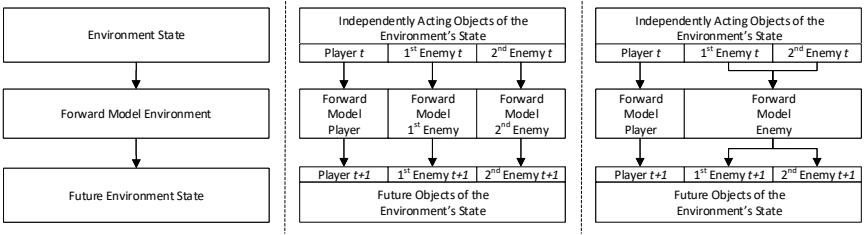
$$
\begin{aligned}
fm(S_t, A_t) &= (\underset{1}{fm}(S_t, A_t),\ \ldots,\ \underset{n}{fm}(S_t, A_t)) \\
&= ((\underset{1,1}{fm}((S_t^{(1,1)}, \ldots, S_t^{(1,k)}), A_t),\ \ldots,\ \underset{m,k}{fm}((S_t^{(m,1)}, \ldots, S_t^{(m,k')}), A_t))) \\
&= (S_{t+1}^{(1)},\ S_{t+1}^{(2)},\ \ldots,\ S_{t+1}^{(n)}) = S_{t+1}
\end{aligned}
$$



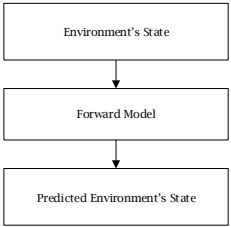
Object-based Forward Model

Aggregate the prediction of each object and its associated sensor values:

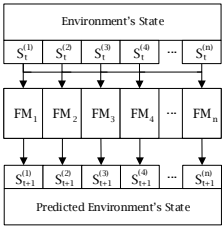
$$
\begin{aligned}
fm(S_t, A_t) &= (\underset{1}{fm}(S_t, A_t),\ \ldots,\ \underset{n}{fm}(S_t, A_t)) \\
&= ((\underset{1,1}{fm}((S_t^{(1,1)}, \ldots, S_t^{(1,k)}), A_t),\ \ldots,\ \underset{m,k}{fm}((S_t^{(m,1)}, \ldots, S_t^{(m,k')}), A_t))) \\
&= (S_{t+1}^{(1)},\ S_{t+1}^{(2)},\ \ldots,\ S_{t+1}^{(n)}) = S_{t+1}
\end{aligned}
$$



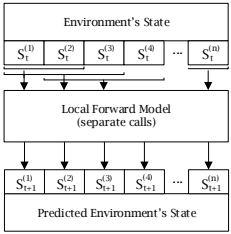
Comparison of Proposed Models



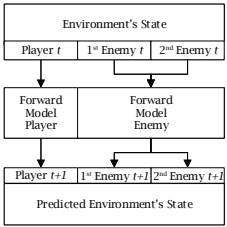
End-to-End FM



Decomposed FM



Local FM



Object-based FM

Comparison of Proposed Models

Qualitative comparison of proposed forward models architectures;

(+) well suited, (~) neutral, (—) poorly suited

Forward Model	#Models	Model Complexity	Interpret-ability	Transfer across levels
End-To-End				
Decomposed				
Local				
Object-based				

Comparison of Proposed Models

Qualitative comparison of proposed forward models architectures;

(+) well suited, (~) neutral, (—) poorly suited

Forward Model	#Models	Model Complexity	Interpret-ability	Transfer across levels
End-To-End	+			
Decomposed	—			
Local	+			
Object-based	~			

Comparison of Proposed Models

Qualitative comparison of proposed forward models architectures;

(+) well suited, (~) neutral, (—) poorly suited

Forward Model	#Models	Model Complexity	Interpret-ability	Transfer across levels
End-To-End	+	—		
Decomposed	—	~		
Local	+	+		
Object-based	~	~		

Comparison of Proposed Models

Qualitative comparison of proposed forward models architectures;

(+) well suited, (~) neutral, (—) poorly suited

Forward Model	#Models	Model Complexity	Interpret-ability	Transfer across levels
End-To-End	+	—	—	
Decomposed	—	~	~	
Local	+	+	+	
Object-based	~	~	+	

Comparison of Proposed Models

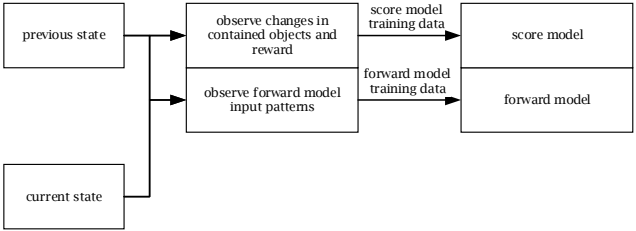
Qualitative comparison of proposed forward models architectures;

(+) well suited, (~) neutral, (—) poorly suited

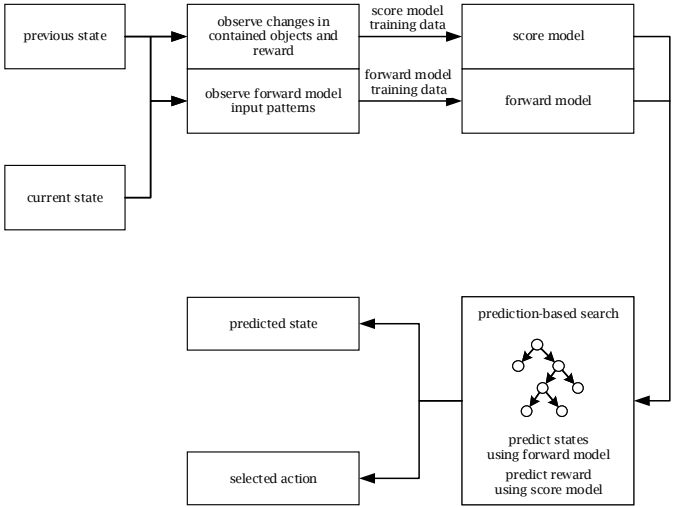
Forward Model	#Models	Model Complexity	Interpret-ability	Transfer across levels
End-To-End	+	—	—	—
Decomposed	—	~	~	—
Local	+	+	+	+
Object-based	~	~	+	+

Prediction-based Search

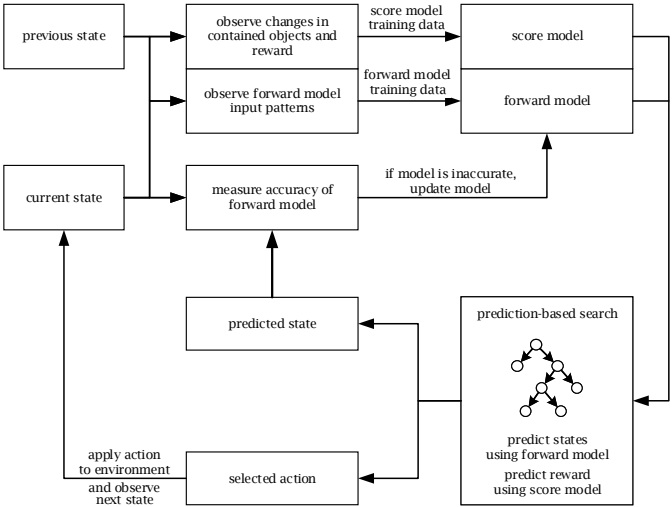
Prediction-based Search



Prediction-based Search



Prediction-based Search



Evaluation Setup I/II

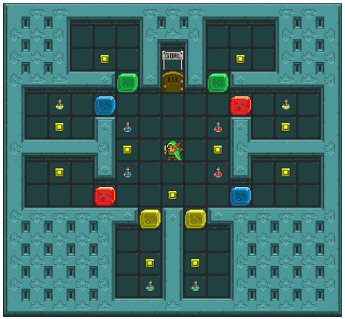
The evaluation is based on 30 games of the General Video Game AI (GVGAI) framework

Evaluation Setup I/II

The evaluation is based on 30 games of the General Video Game AI (GVGAI) framework

Varying Game Characteristics:

- types and number of NPCs
- use of a ressource system
- reward style (dense/sparse)
- the number and types of termination conditions
- determinism vs. non-determinism
- the number of actions available



Evaluation Setup II/II

Local and Object-based Forward Models (LFM/OBFM) have been used in conjunction with:

Evaluation Setup II/II

Local and Object-based Forward Models (LFM/OBFM) have been used in conjunction with:

- Breadth First Search (BFS)
- Rolling Horizon Evoluationary Algorithm (RHEA)
- Monte Carlo Tree Search (MCTS)

Evaluation Setup II/II

Local and Object-based Forward Models (LFM/OBFM) have been used in conjunction with:

- Breadth First Search (BFS)
- Rolling Horizon Evoluationary Algorithm (RHEA)
- Monte Carlo Tree Search (MCTS)

Trained agents are compared to a random agent

- in previous research competitions no agent performed significantly better

Evaluation Setup II/II

Local and Object-based Forward Models (LFM/OBFM) have been used in conjunction with:

- Breadth First Search (BFS)
- Rolling Horizon Evoluationary Algorithm (RHEA)
- Monte Carlo Tree Search (MCTS)

Trained agents are compared to a random agent

- in previous research competitions no agent performed significantly better

Scenarios for Evaluating the Game-Playing Performance

- Constant Model
- Continuous-Learning
- Transfer-Learning

Evaluation Setup II/II

Local and Object-based Forward Models (LFM/OBFM) have been used in conjunction with:

- Breadth First Search (BFS)
- Rolling Horizon Evoluationary Algorithm (RHEA)
- Monte Carlo Tree Search (MCTS)

Trained agents are compared to a random agent

- in previous research competitions no agent performed significantly better

Scenarios for Evaluating the Game-Playing Performance

- Constant Model
- Continuous-Learning
- Transfer-Learning

Learning a Constant Forward Model for each Game

Data Set Generation

- collection of observed state transitions of a random agent
- all 5 levels were played 10 times for a maximum of 200 ticks each
- 9 different local neighborhood patterns were extracted (one data set each)

Learning a Constant Forward Model for each Game

Data Set Generation

- collection of observed state transitions of a random agent
- all 5 levels were played 10 times for a maximum of 200 ticks each
- 9 different local neighborhood patterns were extracted (one data set each)

Accuracy Evaluation and Model Selection

- evaluation of 5 classifiers, multiple parameters

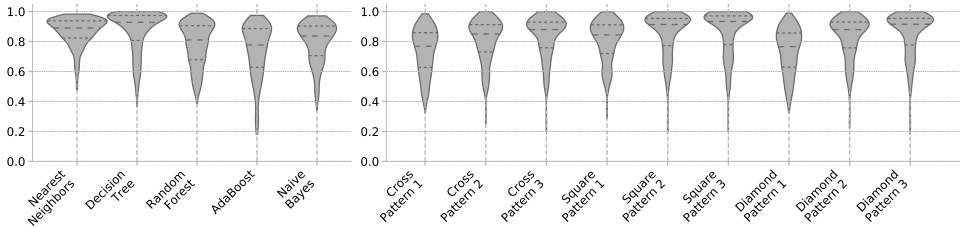
Learning a Constant Forward Model for each Game

Data Set Generation

- collection of observed state transitions of a random agent
- all 5 levels were played 10 times for a maximum of 200 ticks each
- 9 different local neighborhood patterns were extracted (one data set each)

Accuracy Evaluation and Model Selection

- evaluation of 5 classifiers, multiple parameters



Evaluating Game-Playing Performance

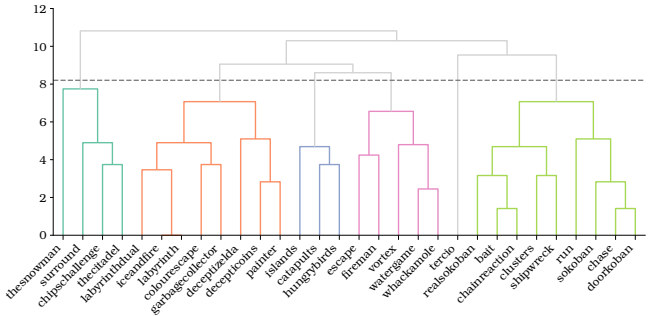
agents are ranked according to their:

- average win-rate, average score, average ticks of won and lost games
- results are clustered to find groups of games in which the agent performs similar

Evaluating Game-Playing Performance

agents are ranked according to their:

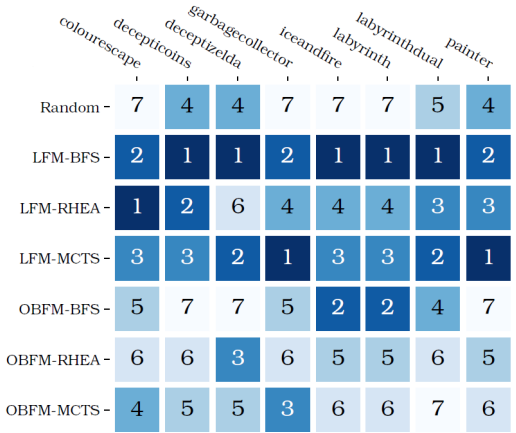
- average win-rate, average score, average ticks of won and lost games
- results are clustered to find groups of games in which the agent performs similar



sparse reward, maze-like, long-term planning, randomness and ressources, puzzles

Detailed Game Results

Results on Maze-Like Games



Rank Comparison:

$$\text{BFS} = \text{MCTS} > \text{RHEA} > \text{Random}$$

$$\text{LFM} > \text{OBFM} > \text{Random}$$

all effects are local

decepticoins, deceptizelda and painter involve spawning elements, which OBFM cannot model

Detailed Game Results

Results on Puzzle Games

	bait	chainreaction	chase	clusters	doorkoban	realsokoban	run	shipwreck	sokoban
Random	7	7	6	7	6	7	6	7	6
LFM-BFS	3	4	7	2	7	4	5	4	7
LFM-RHEA	6	6	4	6	4	6	4	6	5
LFM-MCTS	4	3	3	3	3	5	7	1	4
OBFM-BFS	1	1	2	4	1	2	1	3	2
OBFM-RHEA	5	5	5	5	5	3	3	5	3
OBFM-MCTS	2	2	1	1	2	1	2	2	1

Rank Comparison:

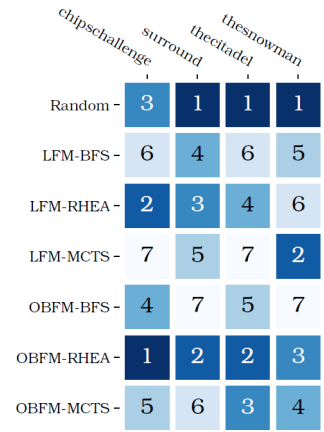
BFS $>$ MCTS $>$ RHEA $>$ Random

OBFM $>$ LFM $>$ Random

most but not all effects are local
(e.g. Doorkoban)

Detailed Game Results

Results on Games with Sparse Reward



Rank Comparison:

$$\text{Random} > \text{RHEA} > \text{BFS} = \text{MCTS}$$

$$\text{Random} > \text{OBFM} = \text{LFM}$$

rare rewards hinder in agent in distinguishing good and bad actions

more training data is required to create a reliable model

Aggregated Results: Game-Playing Performance

Aggregated ranks over all tested games and final score per agent

Agents		Rank 1st	2nd	3rd	4th	5th	6th	7th	Formula-1 Score
Random		4	0	1	3	3	4	15	303
LFM	BFS	10	5	4	4	2	2	3	502
	RHEA	3	3	3	8	3	10	0	380
	MCTS	5	3	8	4	2	4	4	423
OBFM	BFS	6	8	2	4	4	1	5	450
	RHEA	3	5	6	2	10	4	0	411
	MCTS	5	7	4	3	5	4	2	441

Formula-1 Scoring System: $1^{st} = 25$, $2^{nd} = 18$, $3^{rd} = 15$, $4th = 12$, $5th = 10$, $6th = 8$, $7th = 6$

Aggregated Results: Game-Playing Performance

Aggregated ranks over all tested games and final score per agent

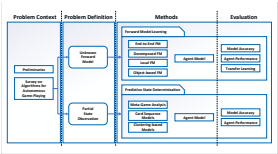
Agents		Rank 1st	2nd	3rd	4th	5th	6th	7th	Formula-1 Score
Random		4	0	1	3	3	4	15	303
LFM	BFS	10	5	4	4	2	2	3	502
	RHEA	3	3	3	8	3	10	0	380
	MCTS	5	3	8	4	2	4	4	423
OBFM	BFS	6	8	2	4	4	1	5	450
	RHEA	3	5	6	2	10	4	0	411
	MCTS	5	7	4	3	5	4	2	441

Formula-1 Scoring System: $1^{st} = 25$, $2^{nd} = 18$, $3^{rd} = 15$, 4th $= 12$, 5th $= 10$, 6th $= 8$, 7th $= 6$

Conclusion

Simulation-based search requires extensions to be used in case:

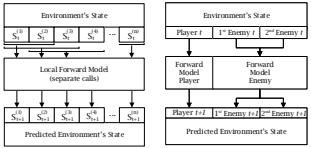
- the environment's forward model is inaccessible
- the environment's state is partial observable



Conclusion

Simulation-based search requires extensions to be used in case:

- the environment's forward model is inaccessible
- the environment's state is partial observable



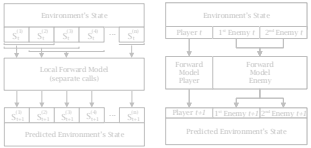
Four types of forward models were introduced.

- The underlying independency assumptions reduced the size of the model space and the required training time.
- A prediction-based search agent has been proposed.

Conclusion

Simulation-based search requires extensions to be used in case:

- the environment's forward model is inaccessible
- the environment's state is partial observable

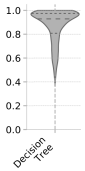


Four types of forward models were introduced.

- The underlying independency assumptions reduced the size of the model space and the required training time.
- A prediction-based search agent has been proposed.

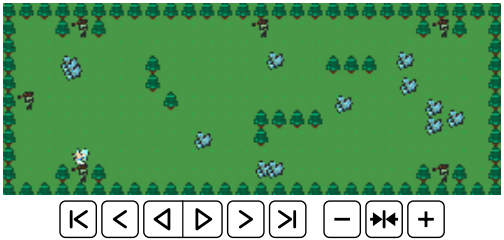
Agents were succesfully trained to play GVGAI games.

- trained agents achieved a high state prediction accuracy and game-playing performance
- learned models can be transferred to unobserved levels

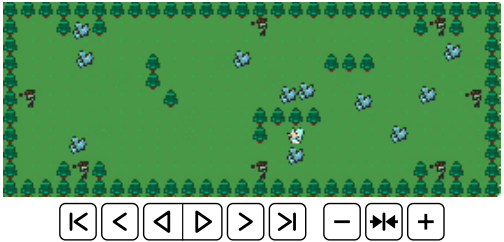


Agents		Formula-1 Score
Random		303
LFM	BFS	502
	RHEA	380
	MCTS	423
OBFM	BFS	450
	RHEA	411
	MCTS	441

State-of-the-Art



Proposed Solution



Thank you for your attention!

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

References

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

Publications - Forward Model learning I/II

Book Chapter

Alexander Dockhorn, Chris Saxton, and Rudolf Kruse; *Association Rule Mining for Unknown Video Games*, A fuzzy dictionary of fuzzy modelling. Common concepts and perspectives, (Accepted), 2020

Journal Paper

Daan Apeldoorn and **Alexander Dockhorn**; *Exception-Tolerant Hierarchical KnowledgeBases for Forward Model Learning*, IEEE Transactions on Games (TOG) (Submitted)

Conference Paper

Alexander Dockhorn and Rudolf Kruse; *Forward Model Learning for Motion Control Tasks*, 10th IEEE International Conference on Intelligent Systems IS'20 (Accepted)

Alexander Dockhorn and Simon Lucas; *Local Forward Model Learning for GVGAI Games*, 2020 IEEE Conference on Games (Submitted)

Publications - Forward Model learning II/II

Conference Paper

Simon Lucas, **Alexander Dockhorn**, Vanessa Volz, Chris Bamford, Raluca Gaina, Ivan Bravi, Diego Perez-Liebana, and Rudolf Kruse; *A Local Approach to Forward Model Learning: Results on the Game of Life Game*. In 2019 IEEE Conference on Games (CoG) (pp. 1–8). IEEE.

Alexander Dockhorn, Simon Lucas, Vanessa Volz, Ivan Bravi, Raluca Gaina, and Diego Perez-Liebana; *Learning Local Forward Models on Unforgiving Games*. In 2019 IEEE Conference on Games (CoG) (pp. 1–4). IEEE.

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

Workshop Paper

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

Publications Predictive State-Determinization

Journal Paper

Alexander Dockhorn, Rudolf Kruse; *Metagame-based Prediction of Cards via Fuzzy Multiset Clustering*, International Journal of Computational Intelligence Systems (Submitted)

Conference Paper

Alexander Dockhorn, Tony Schwensfeier, Rudolf Kruse; *Fuzzy Multiset Clustering for Metagame Analysis*, in Proceedings of the 2019 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (EUSFLAT 2019). Paris, France

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

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

Publications on other Topics I/II

Journal Papers

Pascal Held, **Alexander Dockhorn**, and Rudolf Kruse; *n Merging and Dividing Social Graphs*. Journal of Artificial Intelligence and Soft Computing Research, 5(1), 23–49.

Conference Papers

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

Tim Sabsch, Christian Braune, **Alexander Dockhorn**, and Rudolf Kruse; *Using a Multiobjective Genetic Algorithm for Curve Approximation*. In 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE.

Alexander Dockhorn, Christian Braune, Rudolf Kruse; *Variable density based clustering*. In 2016 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1–8). IEEE.

Publications on other Topics II/II

Conference Papers

Pascal Held, **Alexander Dockhorn**, Benjamin Krause, and Rudolf Kruse; *Clustering Social Networks Using Competing Ant Hives*. In 2015 Second European Network Intelligence Conference (pp. 67–74). IEEE.

Alexander Dockhorn, Christian Braune, and Rudolf Kruse; *An Alternating Optimization Approach based on Hierarchical Adaptations of DBSCAN*. In 2015 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 749–755).

Pascal Held, **Alexander Dockhorn**, and Rudolf Kruse; *Generating Events for Dynamic Social Network Simulations*. 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2014.

Pascal Held, **Alexander Dockhorn**, and Rudolf Kruse; *On Merging and Dividing of Barabasi-Albert-graphs*. In 2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS) (Vol. 444, pp. 17–24).

Preprints

Alexander Dockhorn, Sanaz Mostaghim; *Introducing the Hearthstone-AI Competition*, 1–4. Arxiv ID: 1906.04238