High-Dimensional Function Approximation for Knowledge-Free Reinforcement Learning: a Case Study in SZ-Tetris

Wojciech Jaśkowski Marcin Szubert
Pawel Liskowski Krysztof Krawiec

Institute of Computing Science

POZNAN UNIVERSITY OF TECHNOLOGY

July 14, 2015

RL Perspective

1. **direct policy search** (e.g. EAs), good for Tetris, Othello
2. **value function-based methods** (e.g. TD), good for Backgammon

Comparison: Many factors involved: randomness, environment observability, problem structure, etc.

RL Perspective

1. **direct policy search** (e.g. EAs), good for Tetris, Othello
2. **value function-based methods** (e.g. TD), good for Backgammon

Comparison: Many factors involved: randomness, environment observability, problem structure, etc.

Here: Policy Representation

- For High dimensions only value function-based methods?
- Modern EAs capable of searching high-dimensional spaces, e.g., VD-CMA-ES, R1-NES.

RL Perspective

1. **direct policy search** (e.g. EAs), good for Tetris, Othello
2. **value function-based methods** (e.g. TD), good for Backgammon

Comparison: Many factors involved: randomness, environment observability, problem structure, etc.

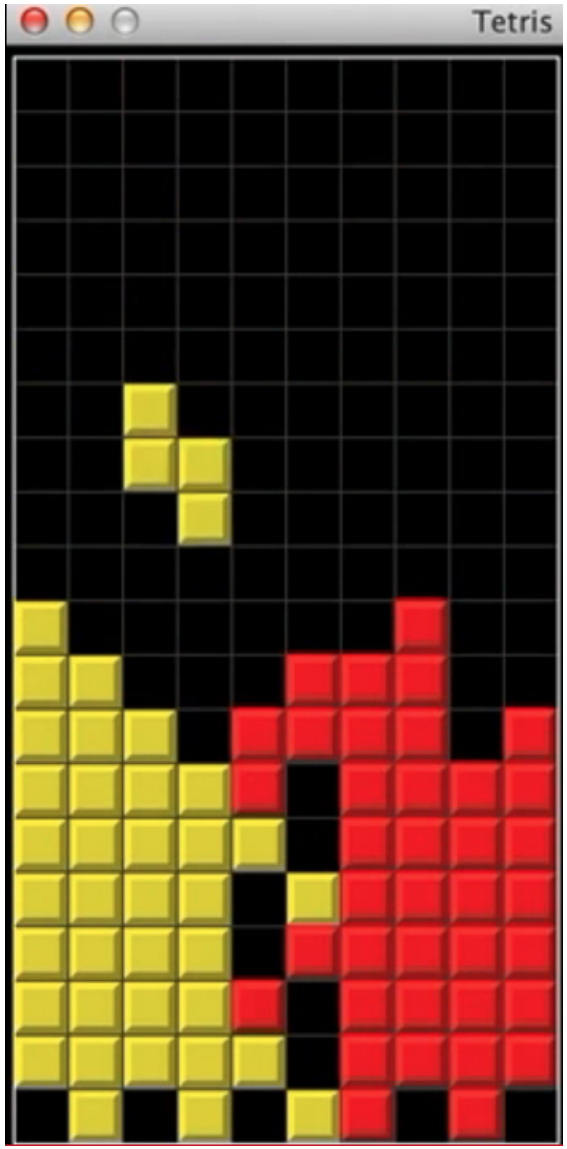
Here: Policy Representation

- For High dimensions only value function-based methods?
- Modern EAs capable of searching high-dimensional spaces, e.g., VD-CMA-ES, R1-NES.

Research Question

How these modern EAs compare to value function-based methods for high-dimensional policy representations?

SZ-Tetris Domain



SZ-Tetris

- a single-player stochastic game,
- a constrained variant of Tetris, a popular yardstick in RL
- devised to studying 'key problems of reinforcement learning'
- 10×20 board
- 17 actions: position + rotation
- 1 point for clearing a line

Hard for value function-based methods

There are many RL algorithms for approximating the value functions. ***None of them really work on (SZ-)Tetris****, they do not even come close to the performance of the evolutionary approaches.*[1]

[1] I. Szita and C. Szepesvári. SZ-Tetris as a benchmark for studying key problems of reinforcement learning. In Proceedings of the ICML 2010 Workshop on Machine

Hard for value function-based methods

There are many RL algorithms for approximating the value functions. ***None of them really work on (SZ-)Tetris****, they do not even come close to the performance of the evolutionary approaches.*[1]

Not easy for direct search methods

Cross Entrophy Method (ca. 117) $<$ hand-coded policy: (ca. 183.6)

[1] I. Szita and C. Szepesvári. SZ-Tetris as a benchmark for studying key problems of reinforcement learning. In Proceedings of the ICML 2010 Workshop on Machine

Hard for value function-based methods

There are many RL algorithms for approximating the value functions. ***None of them really work on (SZ-)Tetris****, they do not even come close to the performance of the evolutionary approaches.*[1]

Not easy for direct search methods

Cross Entrophy Method (ca. 117) < hand-coded policy: (ca. 183.6)

Need for better function approximator

Challenge #1: ***Find a sufficiently good feature set*** *(...). A feature set is sufficiently good if CEM (or CMA-ES, or genetic algorithms, etc.) is able to learn a weight vector such that the resulting preference function reaches at least as good results as the hand-coded solution.*[1]

[1] I. Szita and C. Szepesvári. SZ-Tetris as a benchmark for studying key problems of reinforcement learning. In Proceedings of the ICML 2010 Workshop on Machine

State-Evaluation Function and Action Selection

- Known model $\rightarrow$ we use **state-evaluation function**

$$V : S \rightarrow \mathbb{R}$$

- **Greedy policy** w.r.t V:

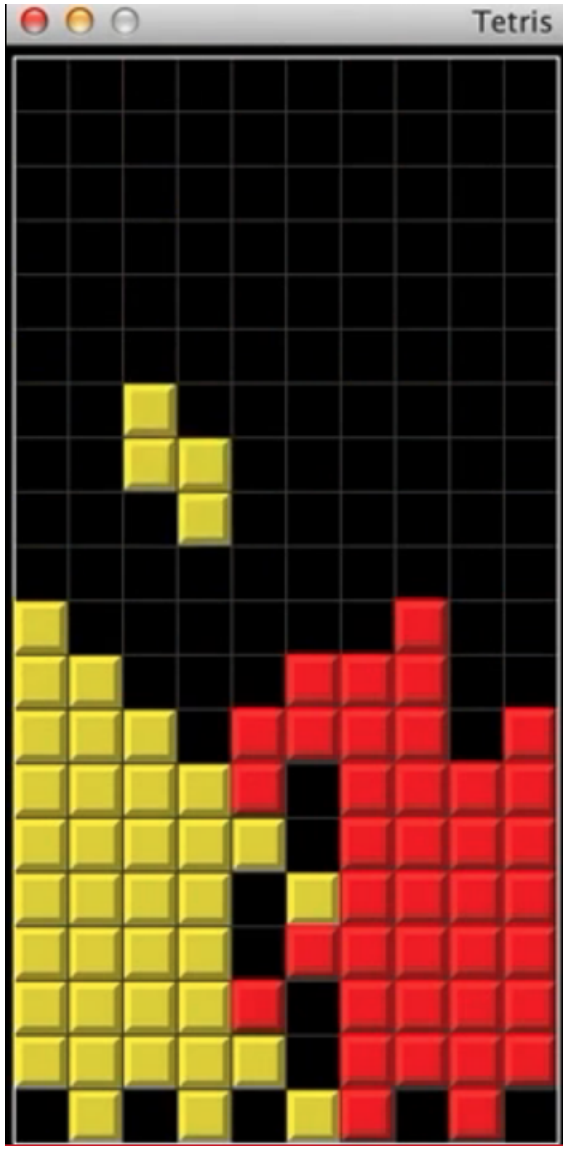
$$\pi(s) = argmax_{a \in A} V(T(s, a)),$$

where T is a transition model.

Evaluation functions:

1. **state-value function** (estimates the expected future scores from a given state),
2. **state-preference function** (no interpretation, larger is better)

Function Approximation

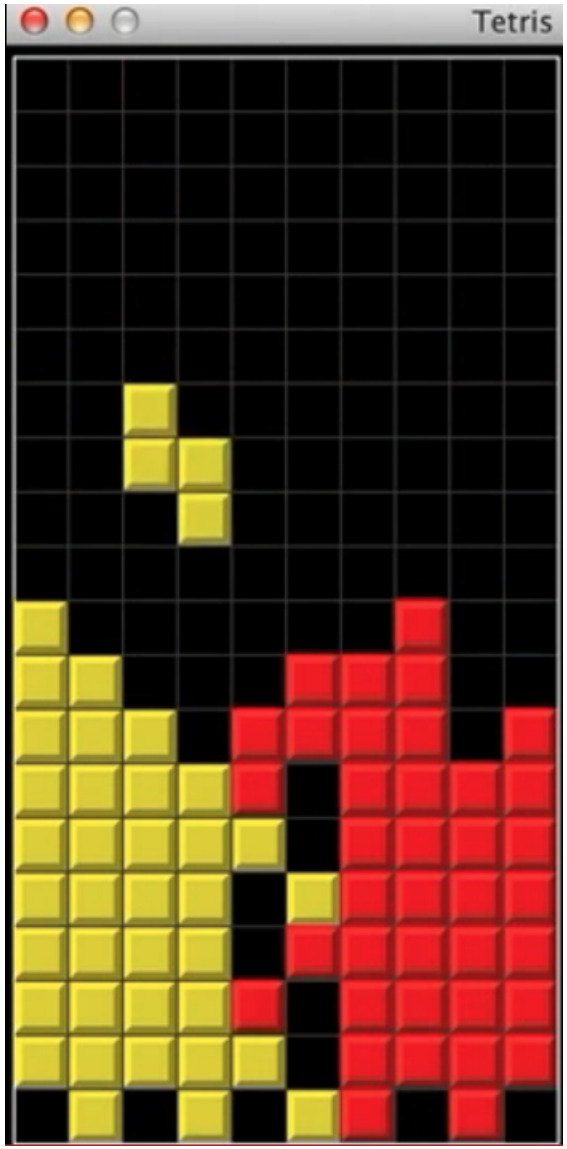


$2^{20\times 10} \approx 10^{60}$ states (upper bound) $\rightarrow$ we need a function approximator:

$$V_\theta : S \rightarrow \mathbb{R}$$

Task: learn the best set of parameters θ.

Weighted Sum of Hand-Designed Features ϕ



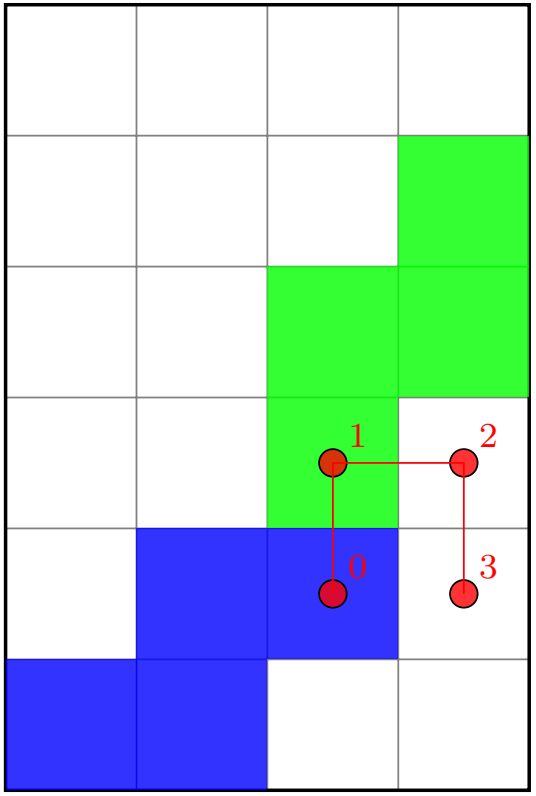
Bertsekas & Ioffe (B&I)

1. Height h_k of the kth column of the board, $k = 1, \ldots 10$.
2. Absolute difference between the heights of the consecutive columns.
3. Maximum column height max_h.
4. Number of 'holes' on the board.

Linear evaluation function of features:

$$V_\theta(s) = \sum_{i=1}^{21} \theta_i \phi_i(s),$$

Systematic n-Tuple Network



LUT

0123	value
0000	3.04
0001	−3.90
0010	−2.14
⋮	⋮
1100	**−2.01**
⋮	⋮
1110	6.12
1111	3.21

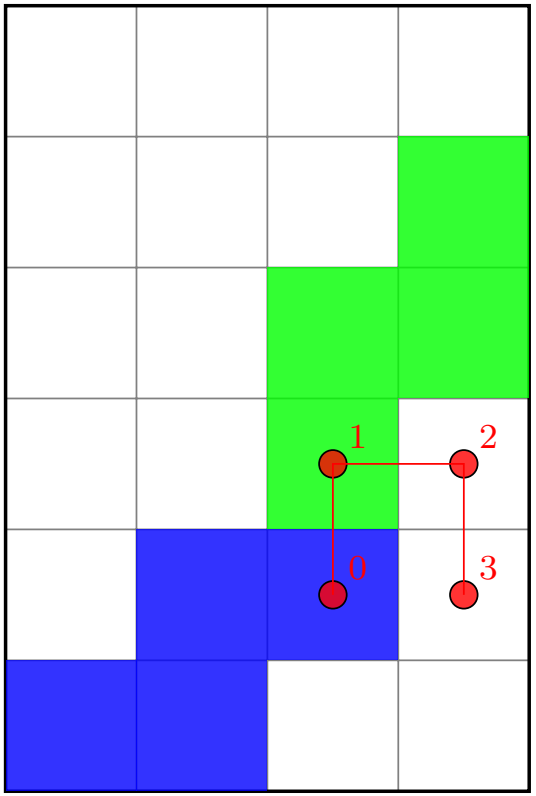
Successful for

1. Othello [Lucas, 2007, Jaśkowski 2014], League
2. Connect-4 [Thill, 2012],
3. 2048 [Szubert, 2015]

- Linear weighted function of (a large number of) binary features
- Computationally efficient

$$V_\theta(s) = \sum_{i=1}^{m} V^i(s) = \sum_{i=1}^{m} \mathrm{LUT}^i \left[\mathrm{index}\left(s_{loc_{i1}}, \ldots, s_{loc_{in_i}}\right)\right]$$

Systematic *n*-tuple Network



LUT

0123	value
0000	3.04
0001	−3.90
0010	−2.14
⋮	⋮
1100	**−2.01**
⋮	⋮
1110	6.12
1111	3.21

Systematically cover the board with:

1. 3×3-tuples (size $= 9$), $|\theta| = 72 \times 2^9 = 36\,864$
2. 4×4-tuples (size $= 16$), $|\theta| = 68 \times 2^{16} = 4\,456\,448$

Direct search methods

ESs maintaining a multi-variate Gaussian probability distribution: $\mathcal{N}(\mu, \Sigma)$:

1. Cross-Entrophy Method [**CEM**, Rubinstein, 2004]:
2. Covariance Matrix Adaptation Evolution Strategy [**CMA-ES**, Hansen 2001]
   - full matrix Σ, smart self-adaptation ($O(n^2)$)
3. CMA-ES for high dimensions [**VD-CMA-ES**, Akimoto, 2014]
   - $\Sigma = D(I + vv^{\mathrm{T}})D$, where D – diagonal matrix, $v \in \mathbb{R}^n$ ($O(n)$)

Learning of V

- After a move the agents gets a new experience $\langle s, a, r, s' \rangle$
- Modify V in response to the experience by Sutton's TD(0) update rule:

$$V(s) \leftarrow V(s) + \alpha(r + V(s') - V(s))$$

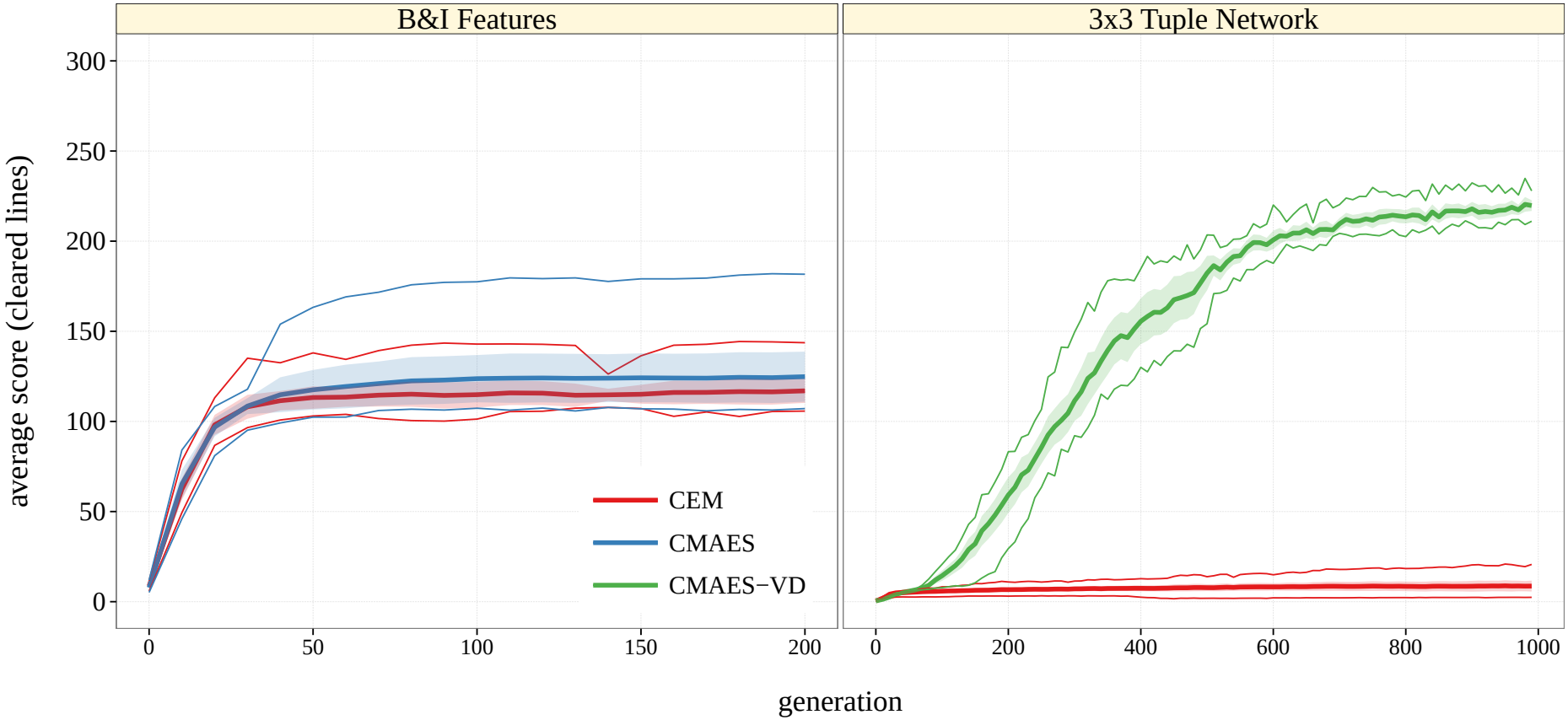
α — learning rate

General Idea

- Reconcile values of neighboring states $V(s)$ and $V(s')$, to make in the long run Bellman equation hold:

$$V(s) = \max_{a \in A(s)} \left(R(s, a) + \sum_{s' \in S} P(s, a, s') V(s') \right)$$

Results for evolutionary methods

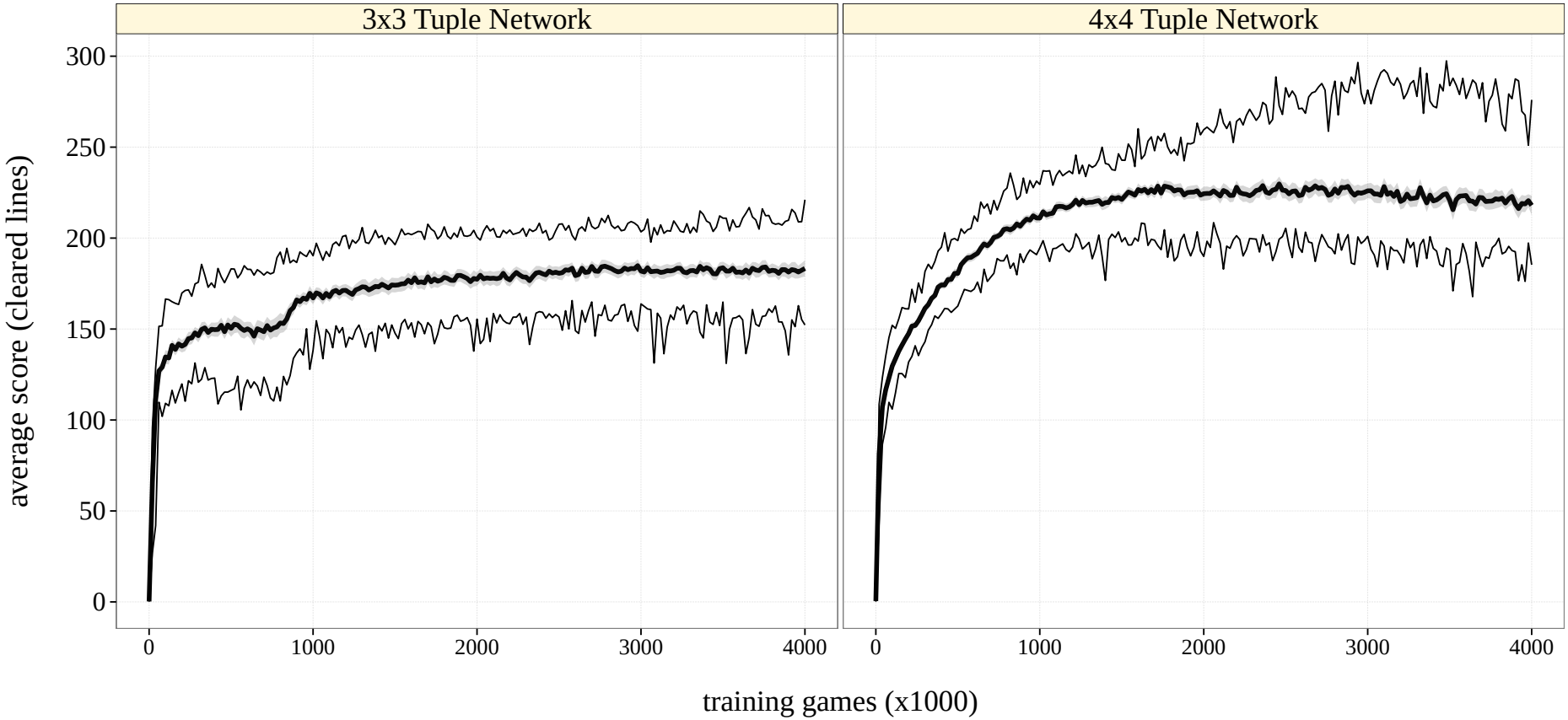


117.0 ± 6.3 CEM

$124.8 \pm 13.$ CMA-ES

219.7 ± 2.8 VD-CMA-ES for 3×3

Results for TD(0)



183.3 ± 4.3 TD(0) for 3×3

218.0 ± 5.2 TD(0) for 4×4

219.7 ± 2.8 VD-CMA-ES for 3×3

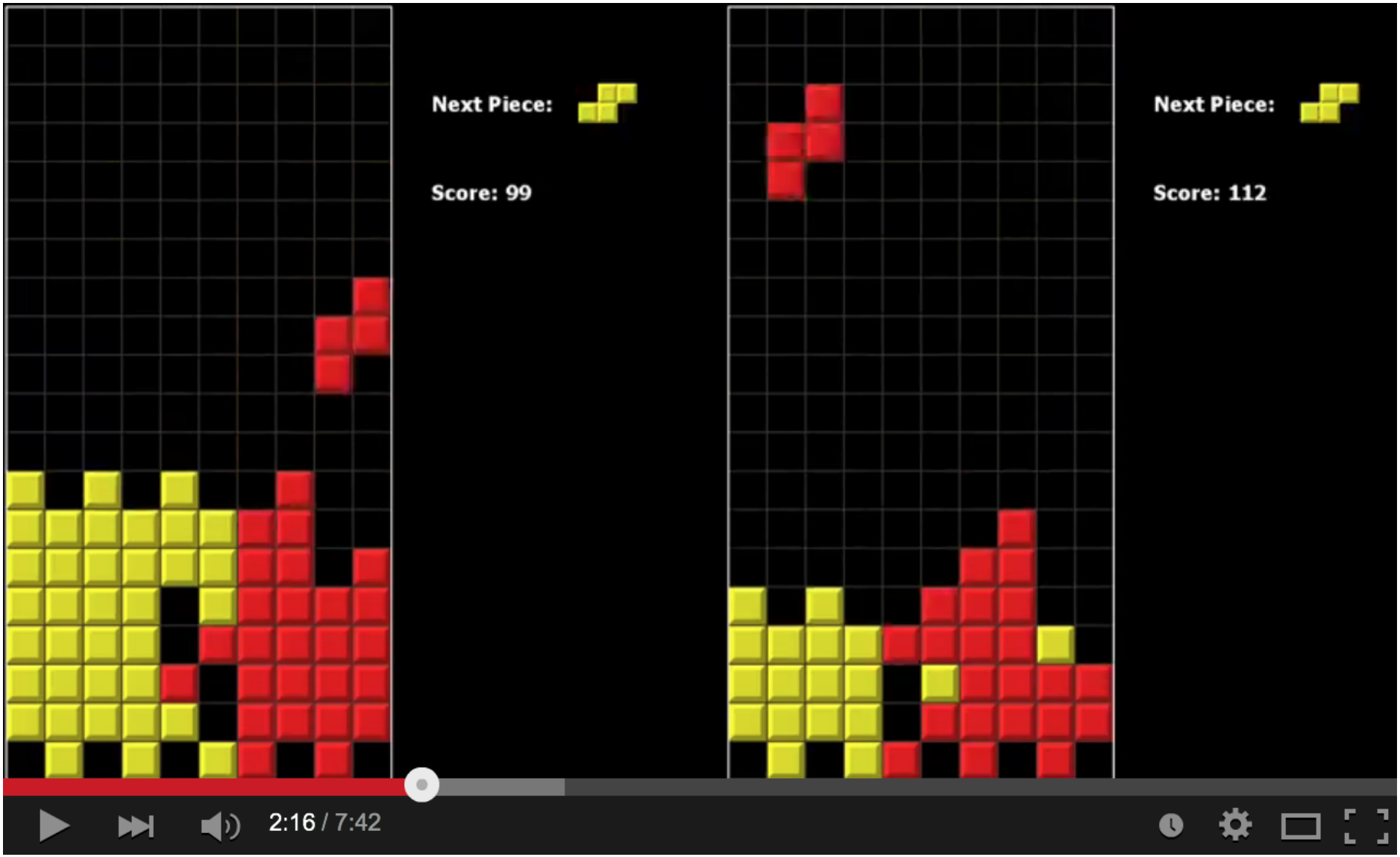
Results Summary

Algorithm	Function	Features	# Games	Result
Hand-coded	-	-	-	183.6 ± 1.4
CEM	B&I	21	20 mln	117.0 ± 6.3
CMA-ES	B&I	21	20 mln	124.8 ± 13.1
VD-CMA-ES	3×3-tuple network	36 864	100 mln	219.7 ± 2.8
TD(0)	3×3-tuple network	36 864	4 mln	183.3 ± 4.3
TD(0)	4×4-tuple network	4 456 448	4 mln	218.0 ± 5.2

Larger variance with TD(0) $4 \times 4 \rightarrow$ best strategy (nearly 300 points on average).

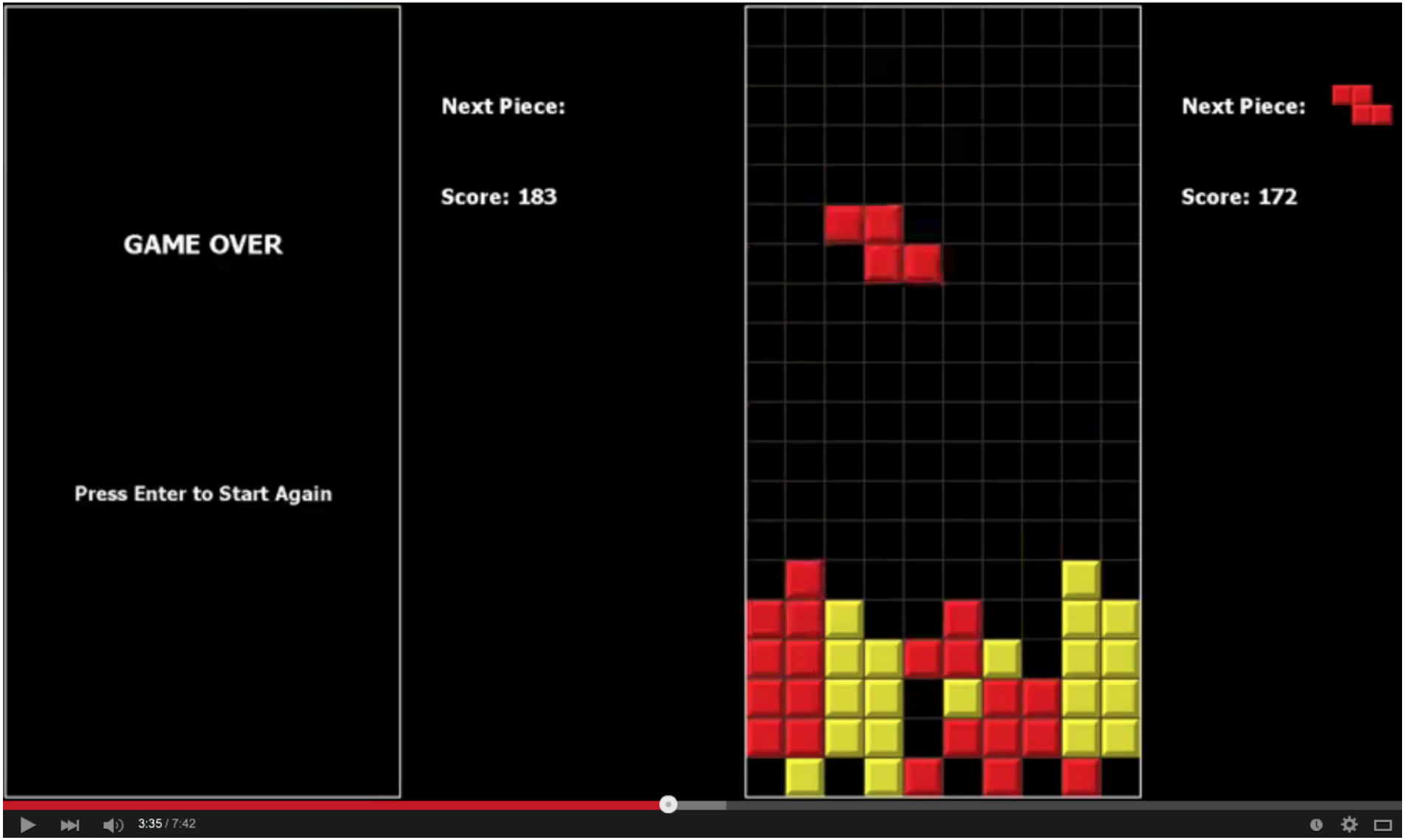
Best agent

play



4x4 TDL agent

play



RL Perspective

1. High-dimensional representation (systematic n-tuple network) to:
   - Make TD work at all on this problem
2. VD-CMA-ES vs. TD:
   - VD-CMA-ES can work with tens of tousands parameters (**needs large populations**)
   - CEM $<$ TD $<$ VD-CMA-ES (on $3x3$)
   - TD vs. VD-CMA-ES $\rightarrow$ **memory vs. time trade-off**

[1] Source code: `http://github.com/wjaskowski/gecco-2015-sztetris`

RL Perspective

1. High-dimensional representation (systematic n-tuple network) to:
   - Make TD work at all on this problem
2. VD-CMA-ES vs. TD:
   - VD-CMA-ES can work with tens of tousands parameters (**needs large populations**)
   - CEM $<$ TD $<$ VD-CMA-ES (on $3x3$)
   - TD vs. VD-CMA-ES $\rightarrow$ **memory vs. time trade-off**

SZ-Tetris Perspective

1. The best player to date[2](nearly 300 lines on average) $>$ hand-coded strategy (183 lines).
2. Systematic n-Tuple Networks solve the Challenge #1 posed by Szita and Szepesvári

[1]Source code: `http://github.com/wjaskowski/gecco-2015-sztetris`