TECHNISCHE UNIVERSITÄT DARMSTADT

Approximate Modified Policy Iteration in Tetris

Jan Köhler

Seminar 2015/16

Professor Fürnkranz

Table of contents

TECHNISCHE UNIVERSITÄT DARMSTADT

Table of contents

TECHNISCHE UNIVERSITÄT DARMSTADT

Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

Modified policy iteration (MPI)

- is a dynamic programming algorithm
- computes optimal policy and value function of Markov Decision Process
- generates a sequence of policy-value pairs

$$\pi_{k+1} = \mathcal{G} v_k \qquad \text{(greedy step)}$$
$$v_{k+1} = (T_{\pi_{k+1}})^m v_k \qquad \text{(evaluation step)}$$

- combines policy iteration (PI, $m = \infty$, fast convergence) and value iteration (VI, $m = 1$, less computation per iteration)
- large state / action spaces: approximate versions API, AVI and AMPI

Background information

TECHNISCHE UNIVERSITÄT DARMSTADT

- **discounted Markov Decision Process** (MDP) $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$
  - $\mathcal{S}$: **state space**
  - $\mathcal{A}$: finite **action space**
  - $P(ds' \mid s, a)$: **probability kernel** on $\mathcal{S}$
  - $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: **reward function** bounded by R_{max}
  - $\gamma \in (0, 1)$: **discount factor**
- **policy** $\pi : \mathcal{S} \to \mathcal{A}$
- **value function** $v_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_\pi(s_t) \mid s_0 = s, s_{t+1} \sim P_\pi(\cdot \mid s_t)\right]$
- **action-value function**
  $Q_\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a, s_{t+1} \sim P(\cdot \mid s_t, a_t), a_{t+1} = \pi(s_{t+1})\right]$
- both bounded by $V_{max} = Q_{max} = R_{max}/(1 - \gamma)$

Background information

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Bellman operator** $[T_\pi v](s) = \mathbb{E}\left[r(s, \pi(s)) + \gamma v(s') \mid s' \sim P_\pi(\cdot \mid s)\right]$

$\rightarrow$ unique fixed-point $v_\pi = (I - \gamma P_\pi)^{-1} r_\pi$

- **greedy policy** $\pi = \mathcal{G}v$ w.r.t. v, if $\forall s \in \mathcal{S}, \quad [T_\pi v](s) = \max_a [T_a v](s)$

$$\iff T_\pi v = \max_{\pi'} [T_{\pi'} v]$$

- **Bellman optimality operator** $T : v \rightarrow \max_\pi T_\pi v = T_{\mathcal{G}(v)} v$

$\rightarrow$ unique fixed-point $v_\star$ is optimal value function

$\rightarrow$ optimal policy $\pi_\star$ is greedy w.r.t. $v_\star$ with value $v_{\pi_\star} = v_\star$

Table of contents

TECHNISCHE UNIVERSITÄT DARMSTADT

AMPI-V algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: Value function space $\mathcal{F} \subset \mathbb{R}^{\mathcal{S}}$, state distribution μ, empirical distribution $\hat{\mu}$
Initialize: Let $v_0 \in \mathcal{F}$ be an arbitrary value function
for $k = 0, 1, \ldots$ **do**

- **Perform rollouts:**

Construct the rollout set $\mathcal{D}_k = \{s^{(i)}\}_{i=1}^{N}$, $s^{(i)} \overset{\text{iid}}{\sim} \mu$
for all states $s^{(i)} \in \mathcal{D}_k$ **do**
Perform a rollout $(s^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, s_m^{(i)})$ using the greedy step
$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \frac{1}{M} \sum_{j=1}^{M} r_a^{(j)} + \gamma v_k(s_a^{(j)})$
with samples of rewards $r_t^{(i)}$, $r_a^{(j)}$ and next states $s_t^{(i)}$, $s_a^{(j)}$
Compute the rollout estimate $\hat{v}_{k+1}(s^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_k(s_m^{(i)})$
end for

- **Approximate value function** by minimizing the empirical error: **(regression)**

$v_{k+1} \in \operatorname{argmin}_{v \in \mathcal{F}} \hat{\mathcal{L}}_k^{\mathcal{F}}(\hat{\mu}; v) = \operatorname{argmin}_{v \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} (\hat{v}_{k+1}(s^{(i)}) - v(s^{(i)}))^2$

end for

AMPI-Q algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: Action-value function space $\mathcal{F} \subset \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, state-action distr. μ, emp. distr. $\hat{\mu}$
Initialize: Let $Q_0 \in \mathcal{F}$ be an arbitrary action-value function
for $k = 0, 1, \ldots$ **do**

- **Perform rollouts:**

Construct the rollout set $\mathcal{D}_k = \{s^{(i)}, a^{(i)}\}_{i=1}^{N}, (s^{(i)}, a^{(i)}) \overset{\text{iid}}{\sim} \mu$
for all state-action pairs $(s^{(i)}, a^{(i)}) \in \mathcal{D}_k$ **do**
Perform a rollout $(s^{(i)}, a^{(i)}, r_0^{(i)}, s_1^{(i)}, a_1^{(i)}, \ldots, s_m^{(i)}, a_m^{(i)})$ using the greedy step
$\pi_{k+1}(s) \in \arg\max_{a\in\mathcal{A}} Q_k(s, a)$
with samples of rewards $r_t^{(i)}$ and next states $s_t^{(i)}$
Compute the rollout estimate $\hat{Q}_{k+1}(s^{(i)}, a^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m Q_k(s_m^{(i)}, a_m^{(i)})$
end for

- **Approximate action-value function** by minimizing the emp. er.: **(regression)**

$Q_{k+1} \in \operatorname{argmin}_{Q\in\mathcal{F}} \hat{\mathcal{L}}_k^{\mathcal{F}}(\hat{\mu}; Q) = \operatorname{argmin}_{Q\in\mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} (\hat{Q}_{k+1}(s^{(i)}, a^{(i)}) - Q(s^{(i)}, a^{(i)}))^2$

end for

Classification-Based MPI algorithm
CBMPI algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: Value function space $\mathcal{F} \subset \mathbb{R}^{\mathcal{S}}$, policy space Π, state distr. μ, emp. distr. $\hat{\mu}$
Initialize: Let $\pi_1 \in \Pi$ be an arbitrary policy and $v_0 \in \mathcal{F}$ an arbitrary value function
for $k = 1, 2, \ldots$ **do**

- **Perform rollouts:**

  Construct the rollout set $\mathcal{D}_k = \{s^{(i)}\}_{i=1}^{N}$, $s^{(i)} \overset{\text{iid}}{\sim} \mu$

  for all states $s^{(i)} \in \mathcal{D}_k$ **do**

    Perform a rollout $(s^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, s_m^{(i)})$ using the policy π_k

    with samples of rewards $r_t^{(i)}$ and next states $s_t^{(i)}$

    Compute the rollout estimate $\hat{v}_k(s^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_{k-1}(s_m^{(i)})$

  end for

Classification-Based MPI algorithm
CBMPI algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

Construct the rollout set $\mathcal{D}'_k = \{s^{(i)}\}_{i=1}^{N'}$, $s^{(i)} \overset{\text{iid}}{\sim} \mu$

for all states $s^{(i)} \in \mathcal{D}'_k$ and actions $a \in \mathcal{A}$ **do**

  for $j = 1$ to M **do**

    Perform a rollout $(s^{(i)}, a, r_0^{(i,j)}, s_1^{(i,j)}, a_1^{(i,j)}, ..., a_m^{(i,j)}, r_m^{(i,j)}, s_{m+1}^{(i,j)})_{j=1}^{M}$

    using the policy π_k with samples of rewards $r_t^{(i,j)}$ and next states $s_t^{(i,j)}$

    Compute $R_k^j(s^{(i)}, a) = \sum_{t=0}^{m} \gamma^t r_t^{(i,j)} + \gamma^{m+1} v_{k-1}(s_{m+1}^{(i,j)})$

  end for

  Compute the rollout estimate $\hat{Q}_k(s^{(i)}, a) = \frac{1}{M} \sum_{j=1}^{M} R_k^j(s^{(i)}, a)$

end for

- **Approximate value function** by minimizing the empirical error: **(regression)**

$$v_k \in \operatorname*{argmin}_{v \in \mathcal{F}} \hat{\mathcal{L}}_k^{\mathcal{F}}(\hat{\mu}; v) = \operatorname*{argmin}_{v \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} (\hat{v}_k(s^{(i)}) - v(s^{(i)}))^2$$

- **Approximate greedy policy** by minimizing the empirical er.: **(classification)**

$$\pi_{k+1} \in \operatorname*{argmin}_{\pi \in \Pi} \hat{\mathcal{L}}_k^{\Pi}(\hat{\mu}; \pi) = \operatorname*{argmin}_{\pi \in \Pi} \frac{1}{N'} \sum_{i=1}^{N'} [\max_{a \in \mathcal{A}} \hat{Q}_k(s^{(i)}, a) - \hat{Q}_k(s^{(i)}, \pi(s^{(i)}))]$$

end for

Additional information

TECHNISCHE UNIVERSITÄT DARMSTADT

- number of samples
  - AMPI-V: $Nm(M|\mathcal{A}|+1)$
  - AMPI-Q: Nm
  - CBMPI: $Nm + M|\mathcal{A}|N'(m+1)$

$\rightarrow$ can be reduced by reusing the rollouts

- AMPI algorithms are generalisations of common algorithms
  - AMPI-V with $m = 1$: fitted value iteration algorithm
  - AMPI-Q with $m = 1$: fitted-Q iteration algorithm
  - CBMPI with $m \rightarrow \infty$: DPI algorithm

Table of contents

TECHNISCHE UNIVERSITÄT DARMSTADT

Error Propagation

TECHNISCHE UNIVERSITÄT DARMSTADT

- wanted: upper bound for performance loss $l_k = v_{\pi_\star} - v_{\pi_k}$
- greedy step error: $\epsilon' = \sup_{j \geq 1} \left\| \epsilon'_j \right\|_\infty$
- evaluation step error: $\epsilon = \sup_{j \geq 1} \left\| \epsilon_j \right\|_\infty$
- upper bound in max-norm:

$$\limsup_{k \to \infty} \left\| l_k \right\|_\infty \leq \frac{2\gamma\epsilon + \epsilon'}{(1-\gamma)^2}$$

- μ-weighted L_p norm $\left\| f \right\|_{p,\mu} = \left[\int |f(x)|^p \, \mu(dx) \right]^{1/p}$
- concentrability coefficients $c_q(j) = \max_{\pi_1, \ldots, \pi_j} \left\| \frac{d(\rho P_{\pi_1} P_{\pi_2} \cdots P_{\pi_j})}{d\mu} \right\|_{q,\mu}$

$$C_q^{l,k,d} = \frac{(1-\gamma)^2}{\gamma^l - \gamma^k} \sum_{i=l}^{k-1} \sum_{j=i}^{\infty} \gamma^j c_q(j+d)$$

$\rightarrow$ measure the stochasticity of an MDP

- upper bound in L_p-norm:

$$\limsup_{k \to \infty} \left\| l_k \right\|_{p,\mu} \leq \frac{2\gamma (C_q^{1,\infty,0})^{\frac{1}{p}} \epsilon + (C_q^{0,\infty,0})^{\frac{1}{p}} \epsilon'}{(1-\gamma)^2}$$

Finite-Sample Analysis of the Algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- upper bounds for greedy and evaluation step errors of the three AMPI algorithms can be computed

$\rightarrow$ new upper bounds for performance loss l_k

- AMPI-Q: $\|l_k\|_{1,\mu} \leq O\left(d_m + \sqrt{\frac{K}{B}} + \gamma^K C_0\right)$
- CBMPI: $\|l_k\|_{1,\mu} \leq O\left(\gamma^m \left(d_m + \sqrt{\frac{m}{B}}\right) + d' + \sqrt{\frac{|\mathcal{A}|mM}{B}}\right)$
- with the variables
  - B: budget (number of samples)
  - K: number of iterations
  - C_0: quality of the initial value / policy
  - $d' = \sup_{g \in \mathcal{F}, \pi' \in \Pi} \inf_{\pi \in \Pi} \mathcal{L}^{\Pi}_{\pi', g}(\mu; \pi)$
  - $d_m = \sup_{g \in \mathcal{F}, \pi \in \Pi} \inf_{f \in \mathcal{F}} \|(T_\pi)^m g - f\|_{2,\mu}$

Table of contents

TECHNISCHE UNIVERSITÄT DARMSTADT

Experimental Results

TECHNISCHE UNIVERSITÄT DARMSTADT

- application of AMPI-Q and CBMPI to
  - mountain car problem
  - Tetris
- comparison with other algorithms' performance
- large m (length of rollouts)

$\rightarrow$ small regressor error, big classifier error

$\rightarrow$ small rollout sets for fixed budget B (number of samples)

$\rightarrow$ useful for poor value function spaces

- best results for $M = 1$ (number of rollouts) and reusing rollouts
- budget
  - DPI, CBMPI: $B = |\mathcal{A}| N(m+1)$
  - AMPI-Q: $B = Nm$

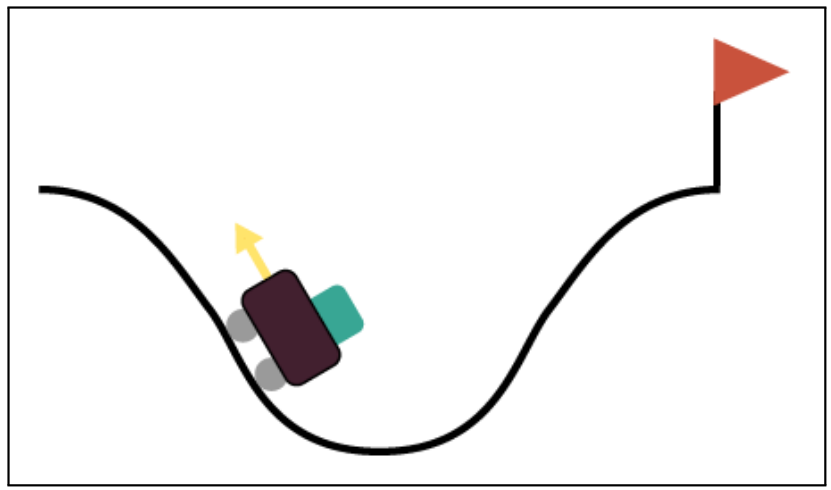
Table of contents

TECHNISCHE UNIVERSITÄT DARMSTADT

Mountain Car: state and action space

TECHNISCHE UNIVERSITÄT DARMSTADT

- goal: driving car on top of hill (reward 0)
- states $s = (x_s, \dot{x}_s)$
  - x_s : position
  - $\dot{x}_s$: velocity
- actions (reward)
  - forward (-1)
  - reverse (-1)
  - stay (-1)
- discount factor $\gamma = 0.99$

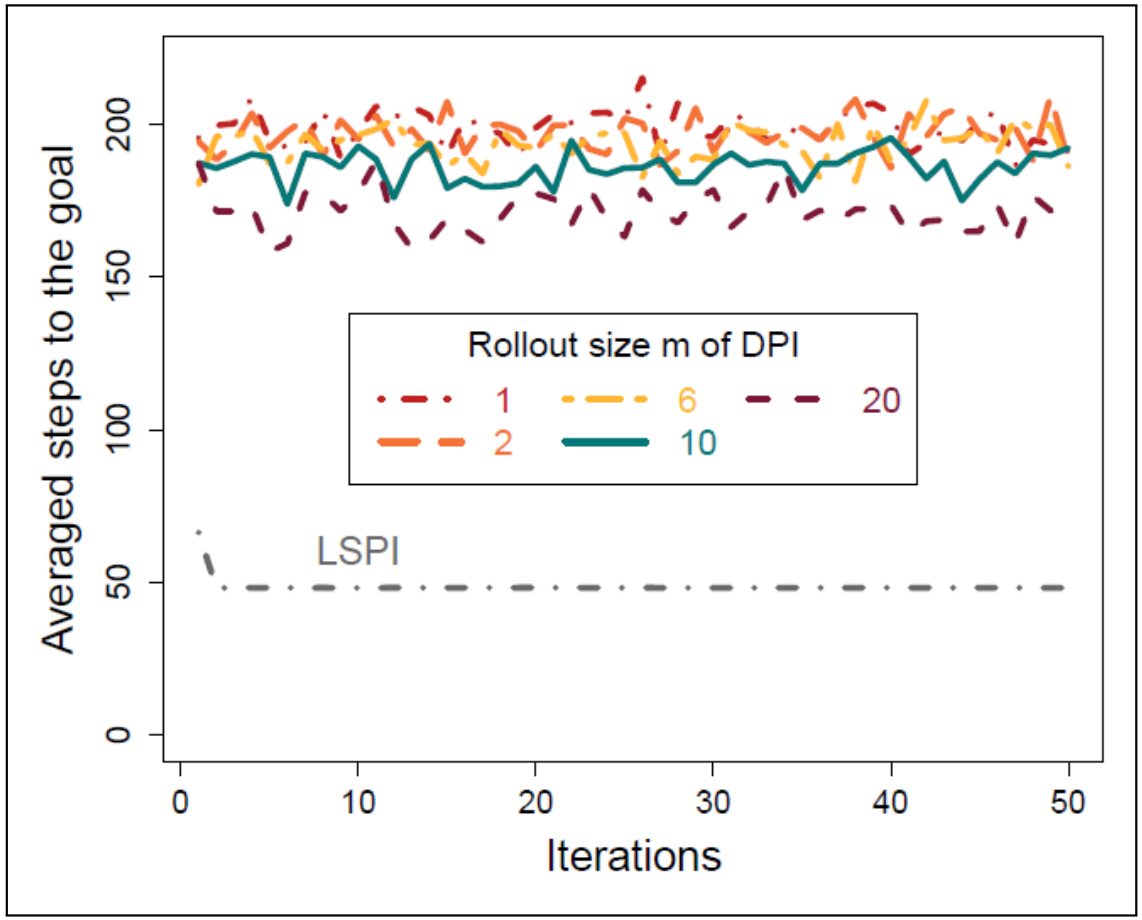


Mountain Car: rich function space, budget $B = 4000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- LSPI converges to 50 steps
- DPI converges to 160 steps ($m = 20$)

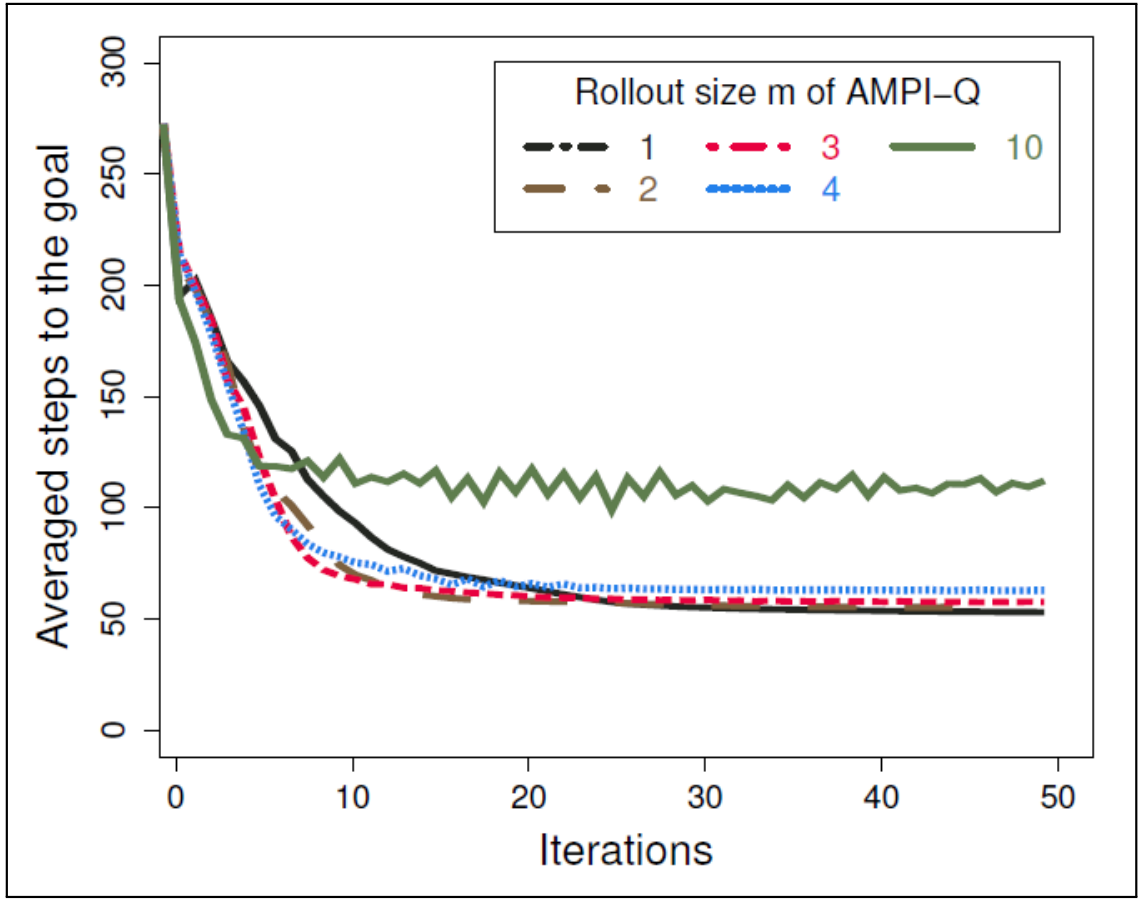
$\rightarrow$ needs big rollout which contains the goal



Mountain Car: rich function space, budget $B = 4000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- AMPI-Q converges to 50 steps ($m < 10$)
- for $m = 10$ the rollout sets are too small (fixed budget B)



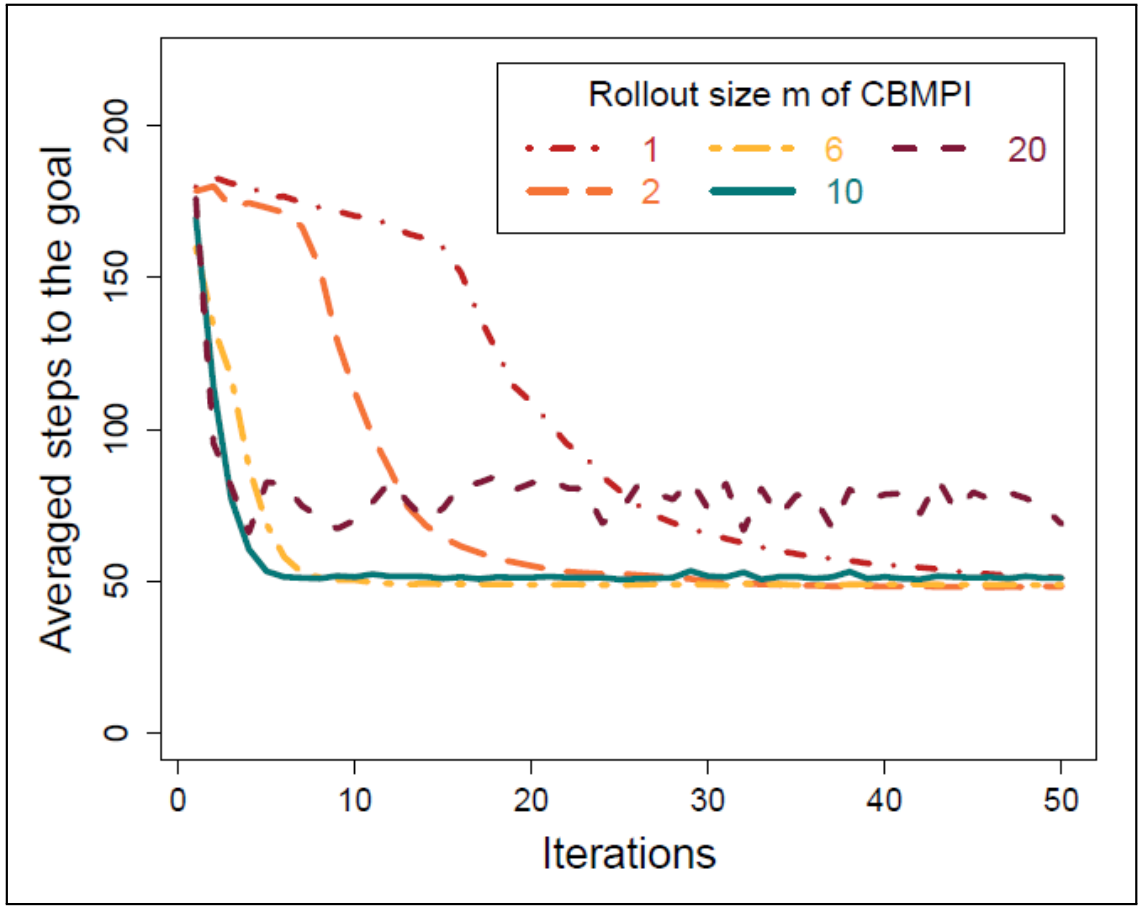
Mountain Car: rich function space, budget $B = 4000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- CBMPI converges to 50 steps ($m < 20$)

$\rightarrow$ LSPI converges faster

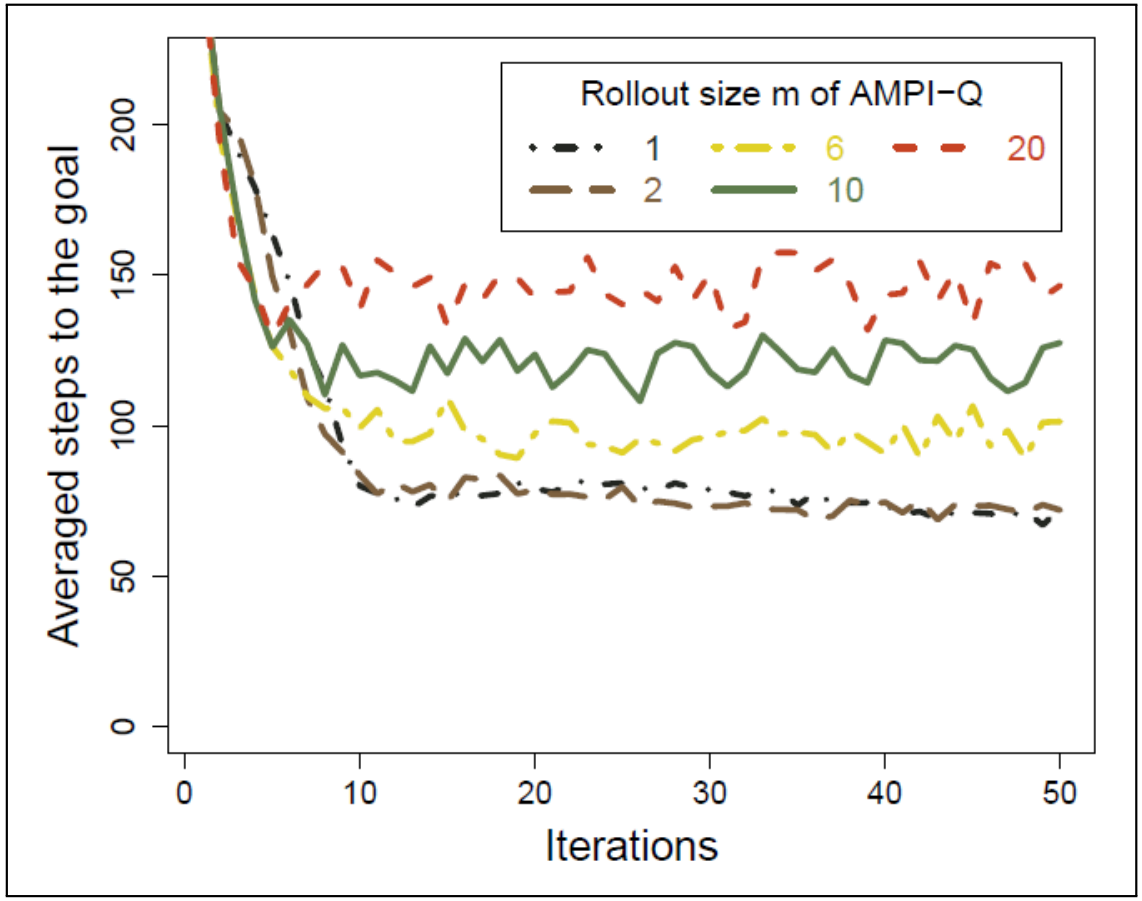
- for $m = 20$ the rollout sets are too small (fixed budget B)



Mountain Car: poor function space, budget $B = 4000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- AMPI-Q converges to 70 steps ($m < 6$)
- for $m \geq 6$ the rollout sets are too small (fixed budget B)



Mountain Car: poor function space, budget $B = 4000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- LSPI converges to 75 steps
- CBMPI converges to 50 steps ($m = 6$, $m = 10$)
- for $m = 20$ the rollout sets are too small (fixed budget B)
- for $m = 1$, $m = 2$ the function space is too poor

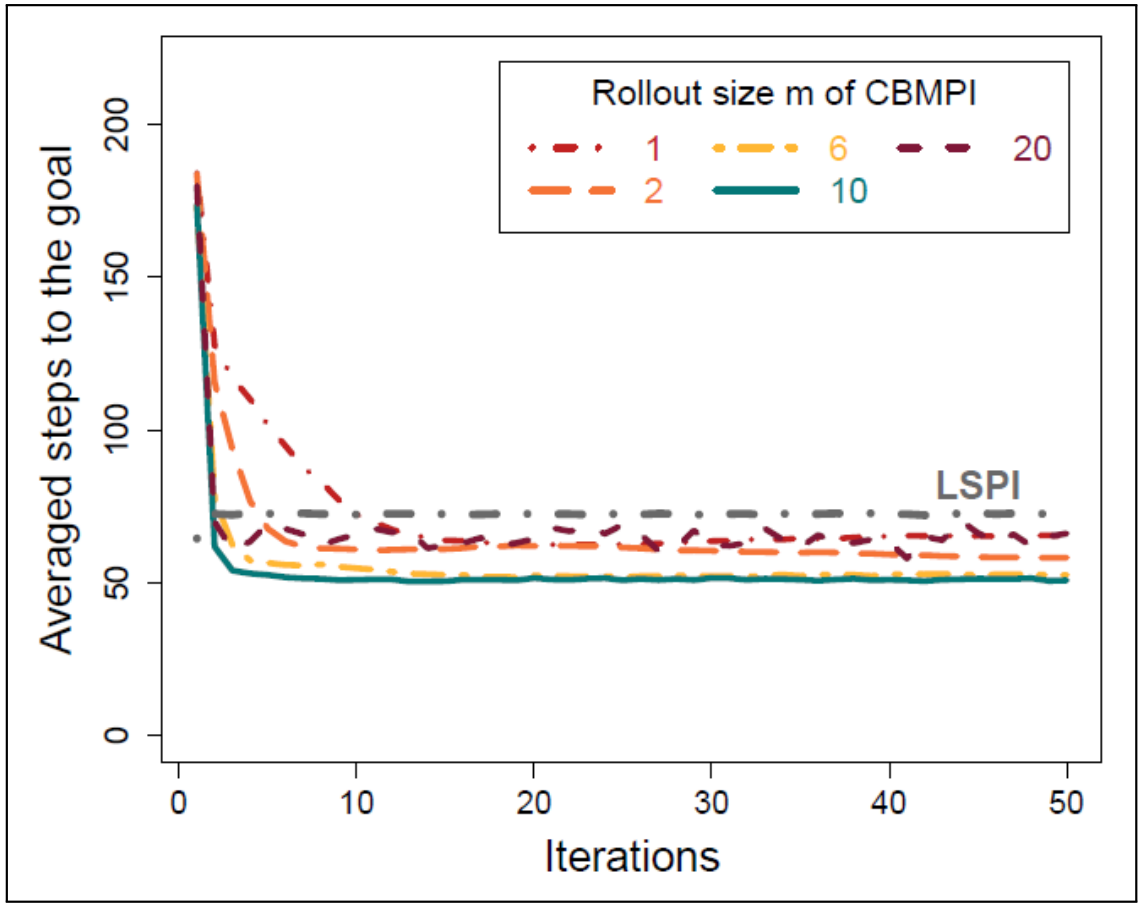


Table of contents

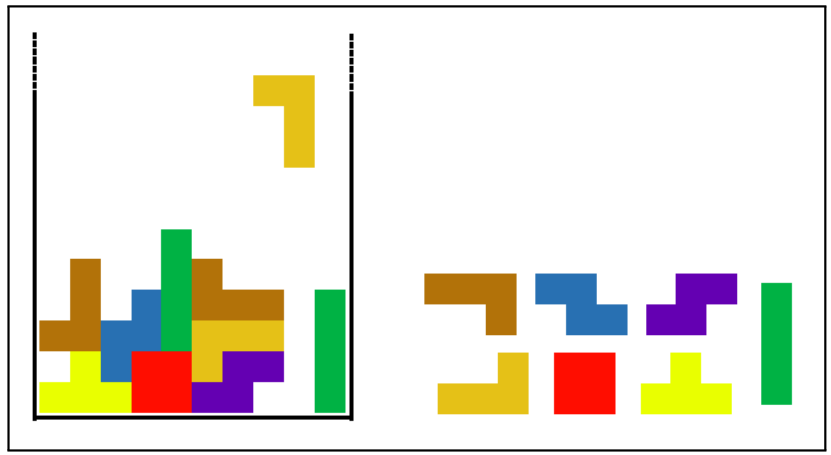
TECHNISCHE UNIVERSITÄT DARMSTADT

Tetris: state and action space

TECHNISCHE UNIVERSITÄT DARMSTADT

- goal: maximize score (removed lines)
- state
  - board configuration b
    (all in all $2^{200} \approx 1.6 \cdot 10^{60}$)
  - falling piece p (all in all 7)
- actions (number depends on shape)
  - orientation of piece
  - location of piece
- reward

  maximizing expected sum of rewards
  $\sim$ maximizing score



Tetris: background information

TECHNISCHE UNIVERSITÄT DARMSTADT

- provides optimization benchmark w.r.t. average number of removed lines

$\rightarrow$ finite number because Tetris ends with probability one

- finding strategy to maximize: NP hard (even if sequence of pieces is known)
- algorithms based on approximating value function (e.g. λ-PI): not successful
- algorithms based on search in space of policies (e.g. CE method, DPI): best results

$\rightarrow$ also CBMPI?

Tetris: implementation details of CBMPI

TECHNISCHE UNIVERSITÄT DARMSTADT

- evaluation function $f(\cdot) = \phi(\cdot)^T\theta$ gives value to an action at a state
  - ϕ : features
  - θ : parameter vector
- regressor
  - value function $\hat{v}_k(s^{(i)}) = \phi(s^{(i)})\alpha$
    with feature vector ϕ and weight vector α (initially $\alpha = (0, 0, ..., 0)$)
  - minimize empirical error $\hat{\mathcal{L}}_k^{\mathcal{F}}(\hat{\mu}; v)$ with least-squares method
- classifier
  - policy $\pi_\beta(s) = \arg\max_a \psi(s, a)^T\beta$
    with policy feature vector ψ
    and policy parameter vector β (initially random)
  - minimize empirical error $\hat{\mathcal{L}}_k^{\Pi}(\hat{\mu}; \pi_\beta)$ with covariance matrix adaption evolution strategy
- samples taken from good policy (DU controller), subsampling to uniform board height distribution

Tetris: features

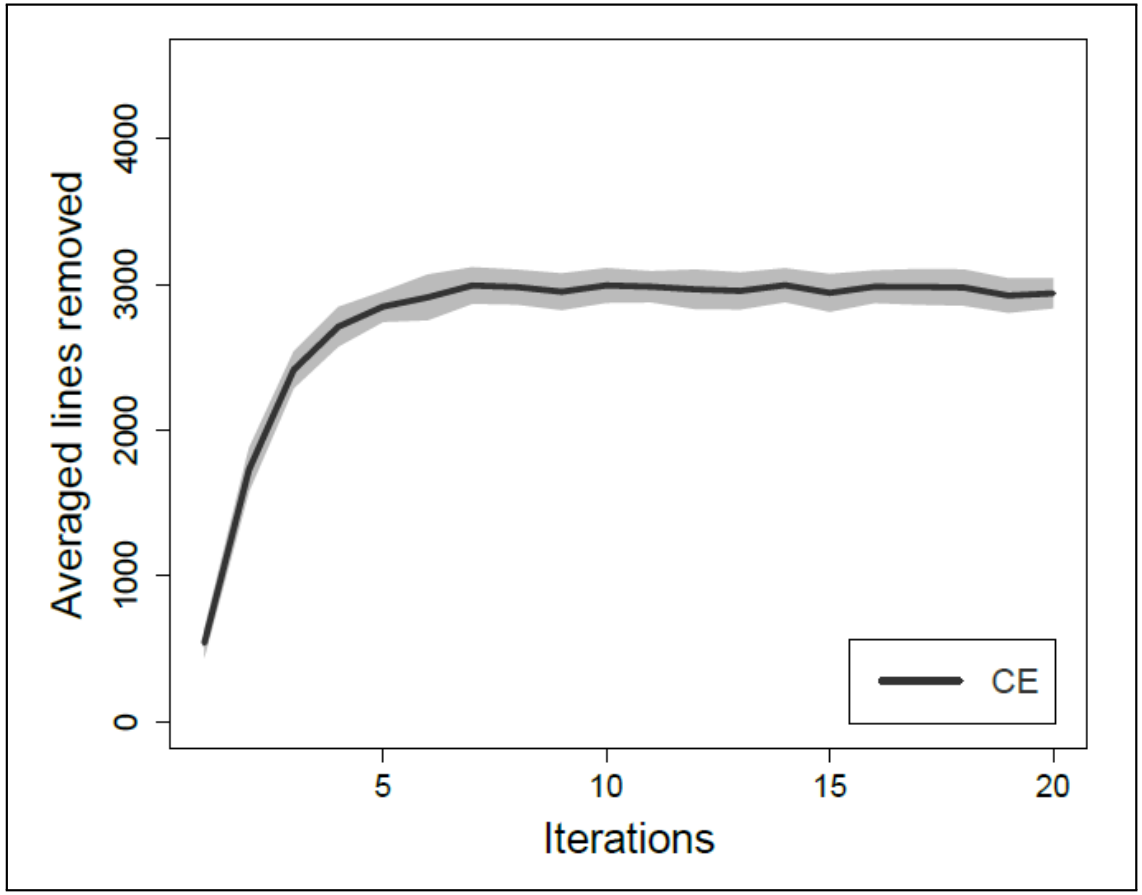
TECHNISCHE UNIVERSITÄT DARMSTADT

- Bertsekas Features: 22 features, e.g.
  - number of holes
  - height of columns
  - difference in height of adjacent columns
  - maximum height
- Dellacherie-Thiery Features: nine features, e.g.
  - landing height
  - number of eroded piece cells
  - row / column transitions
  - hole depth and number of board wells
  - number of rows with holes
  - pattern diversity feature
- RBF Height Features: five features
  - $\exp(\frac{-|c-ih/4|^2}{2(h/5)^2})$, $i = 0, ..., 4$
    with average height of columns c and total number of rows h

Tetris: small 10×10 board, D-T Features, budget $B = 6,500,000$

TECHNISCHE UNIVERSITÄT DARMSTADT

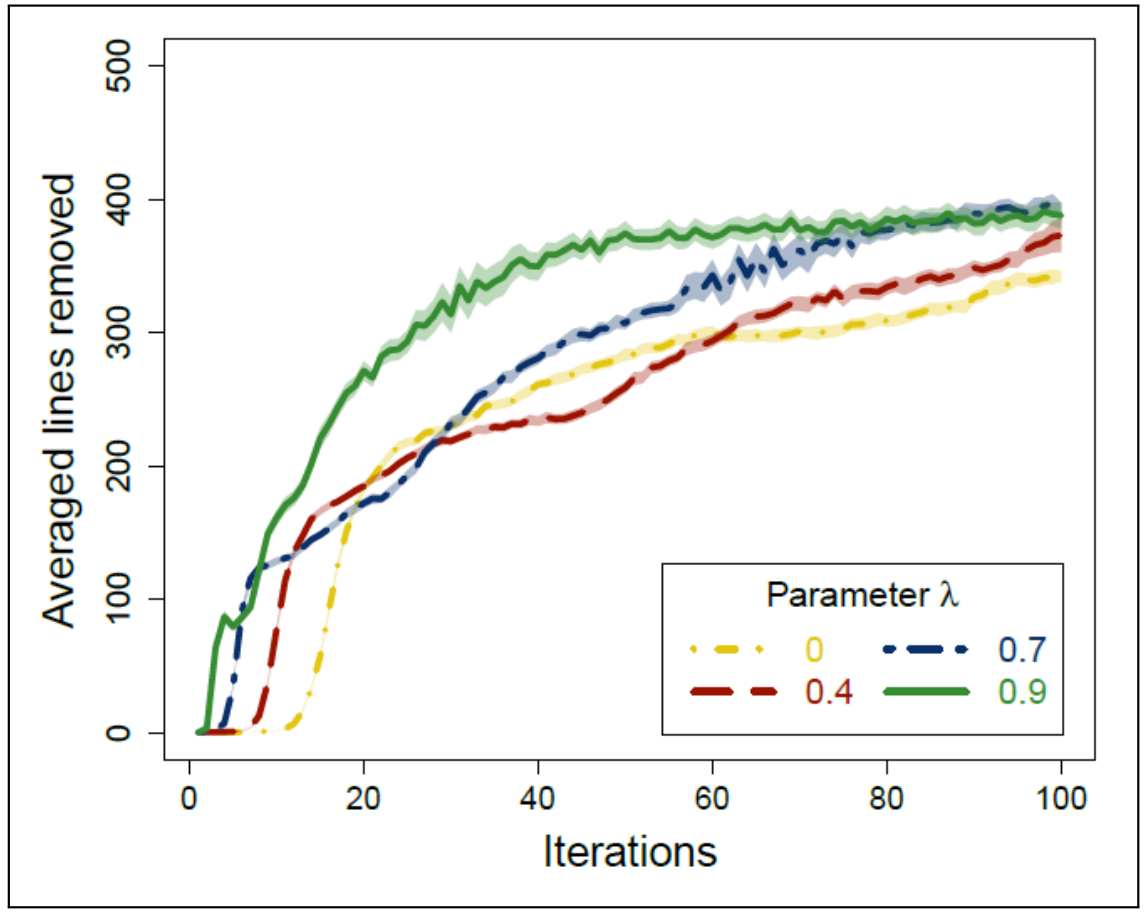
- CE converges to score 3,000 after 10 iterations



Tetris: small 10×10 board, D-T Features

TECHNISCHE UNIVERSITÄT DARMSTADT

- $\lambda-$PI converges to score 400



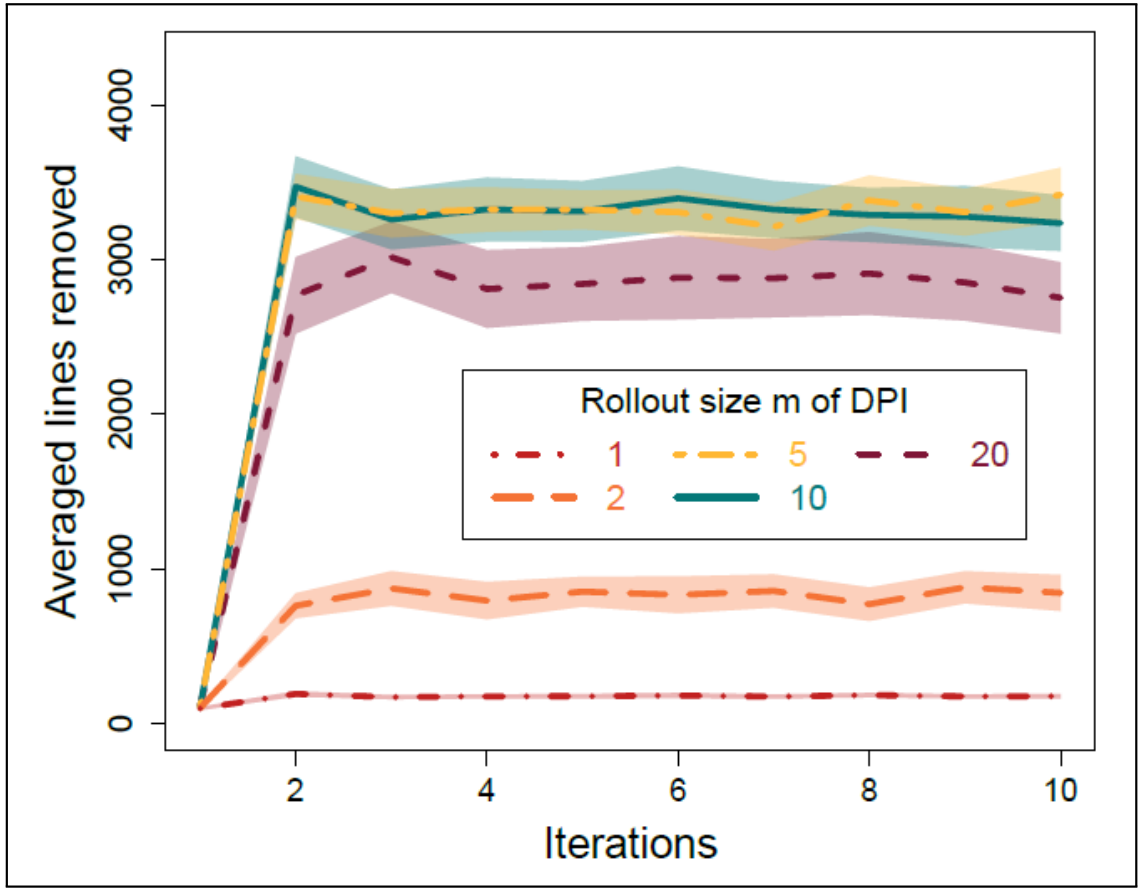
Tetris: small 10×10 board, D-T Features, budget $B = 8,000,000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- DPI converges to score 3,400 ($m = 5, m = 10$)

$\rightarrow$ needs big rollout which contains the goal

$\rightarrow$ for $m = 20$ the rollout sets are too small (fixed budget B)



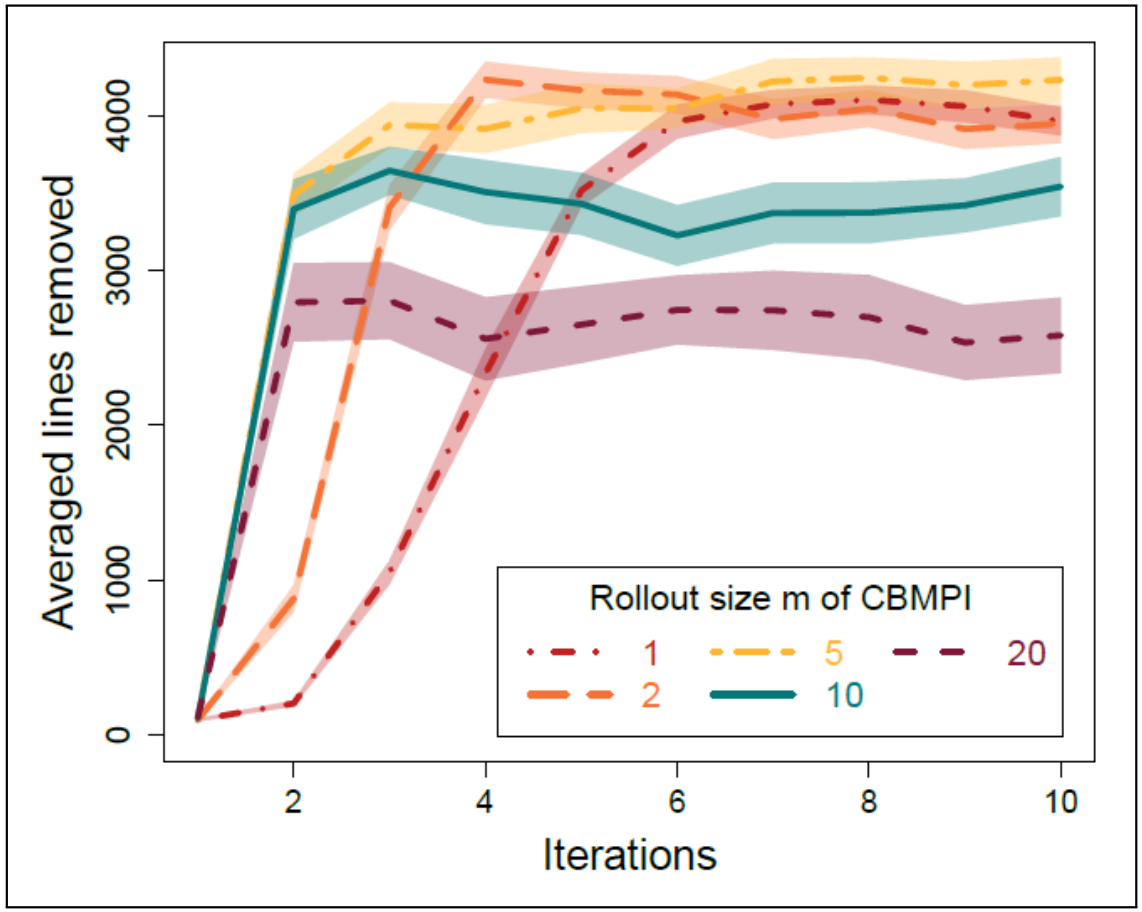
Tetris: small 10×10 board, D-T Features, budget $B = 8,000,000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- CBMPI converges to score 4,200 ($m = 5$)
- also good performance for $m = 1$
- faster convergence than CE

$\rightarrow$ CBMPI is best algorithm

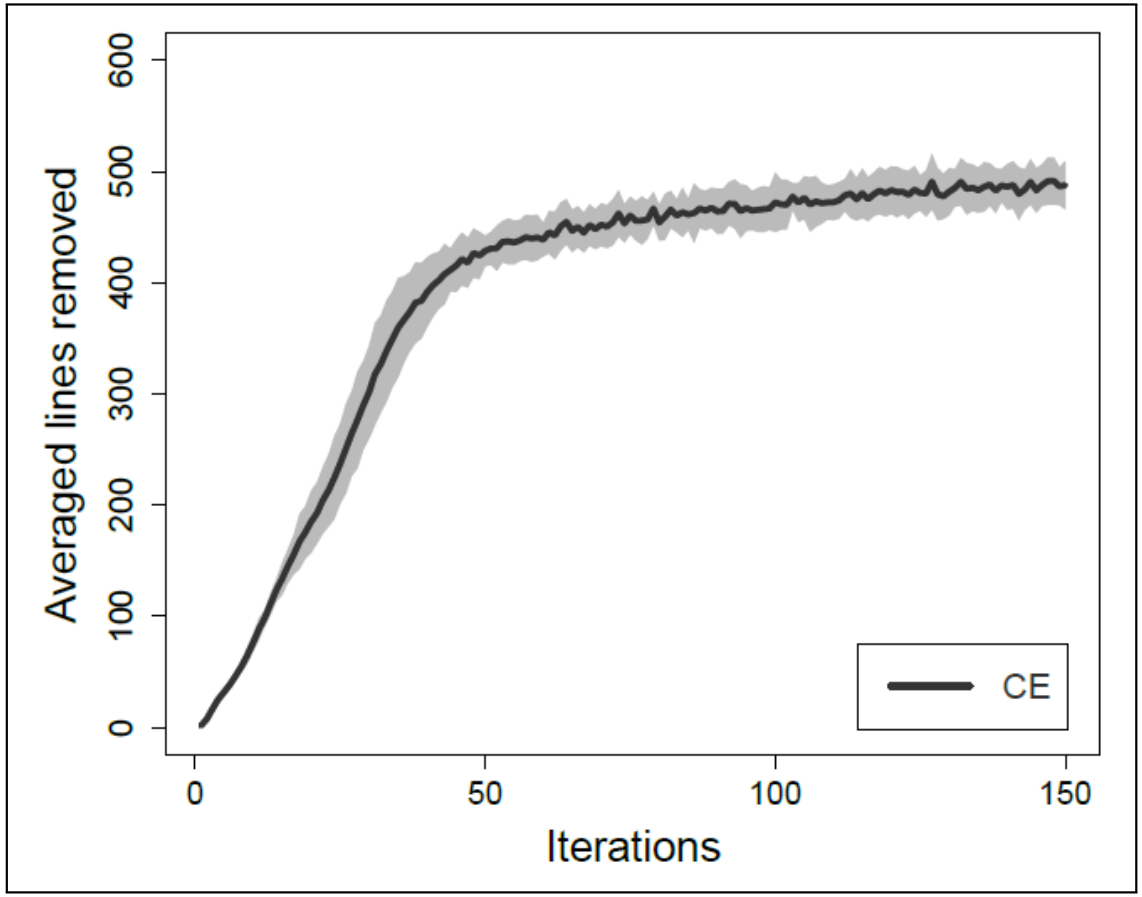
$\rightarrow$ policy search methods perform better than value-function based algorithms (e.g. $\lambda-$PI)



Tetris: small 10×10 board, Bertsekas Features

TECHNISCHE UNIVERSITÄT DARMSTADT

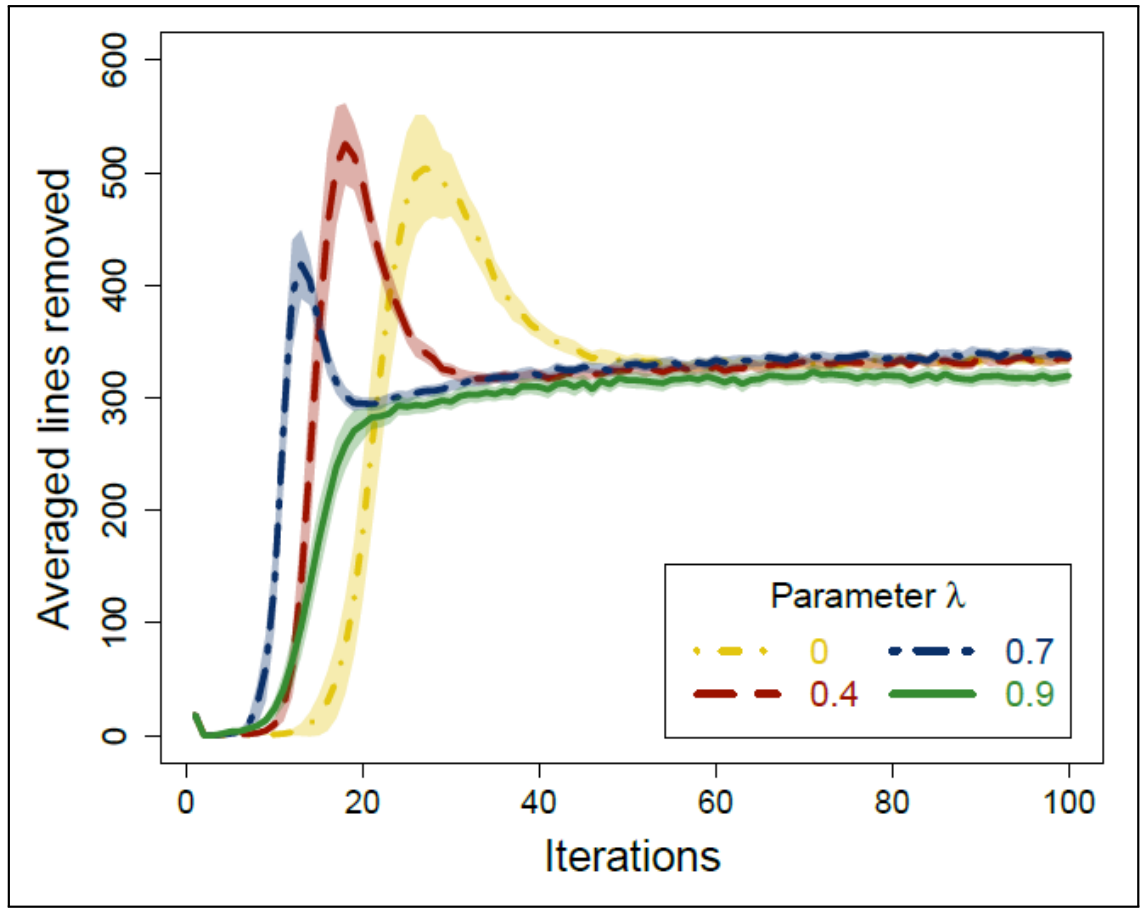
- CE converges to score 500 after 60 iterations



Tetris: small 10×10 board, Bertsekas Features

TECHNISCHE UNIVERSITÄT DARMSTADT

- $\lambda-$PI converges to score 350



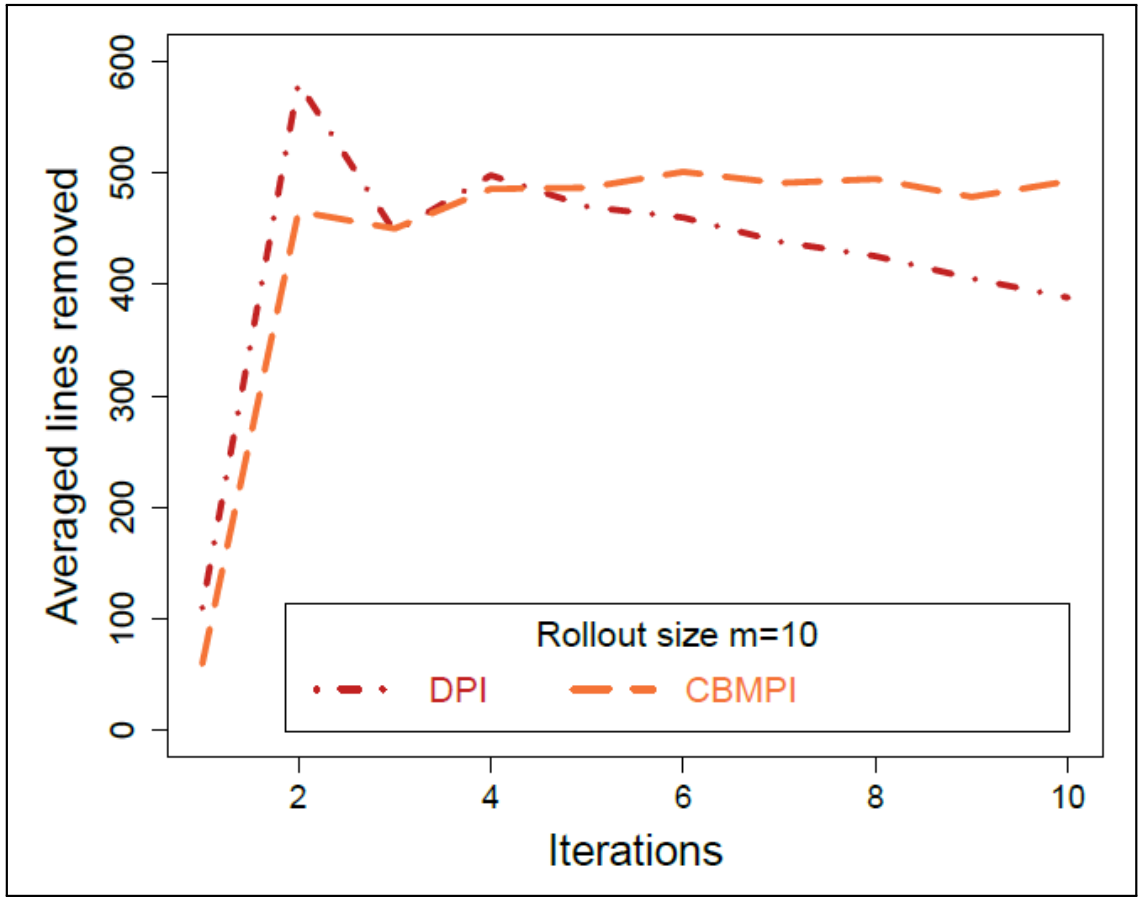
Tetris: small 10×10 board, Bertsekas Features, budget $B = 80,000,000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- DPI converges to score 400 ($m = 10$)
- CBMPI converges to score 500 ($m = 10$) after 2 iterations
- faster convergence than CE

$\rightarrow$ CBMPI is best algorithm

$\rightarrow$ D-T Features are more suitable than Bertsekas Features

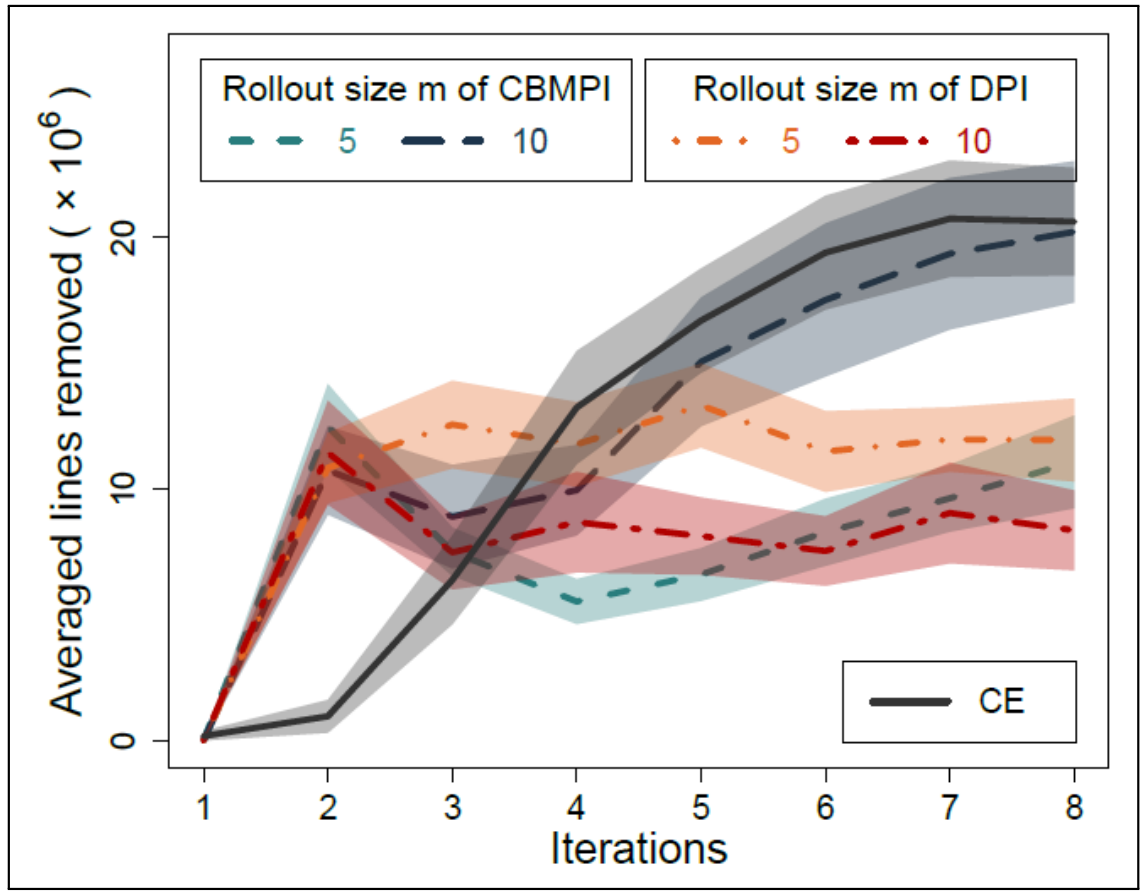


Tetris: large 10×20 board, D-T Features, budget $B = 32,000,000$ for CBMPI and DPI budget $B = 210,000,000$ for CE

TECHNISCHE UNIVERSITÄT DARMSTADT

- CBMPI converges to score 20,000,000 ($m = 10$) after 8 iterations
- DPI converges to score 12,000,000 ($m = 5$) after 3 iterations
- CE converges to score 20,000,000 after 8 iterations
- needs $\approx$ 6 times more samples than CBMPI

$\rightarrow$ CBMPI is best algorithm

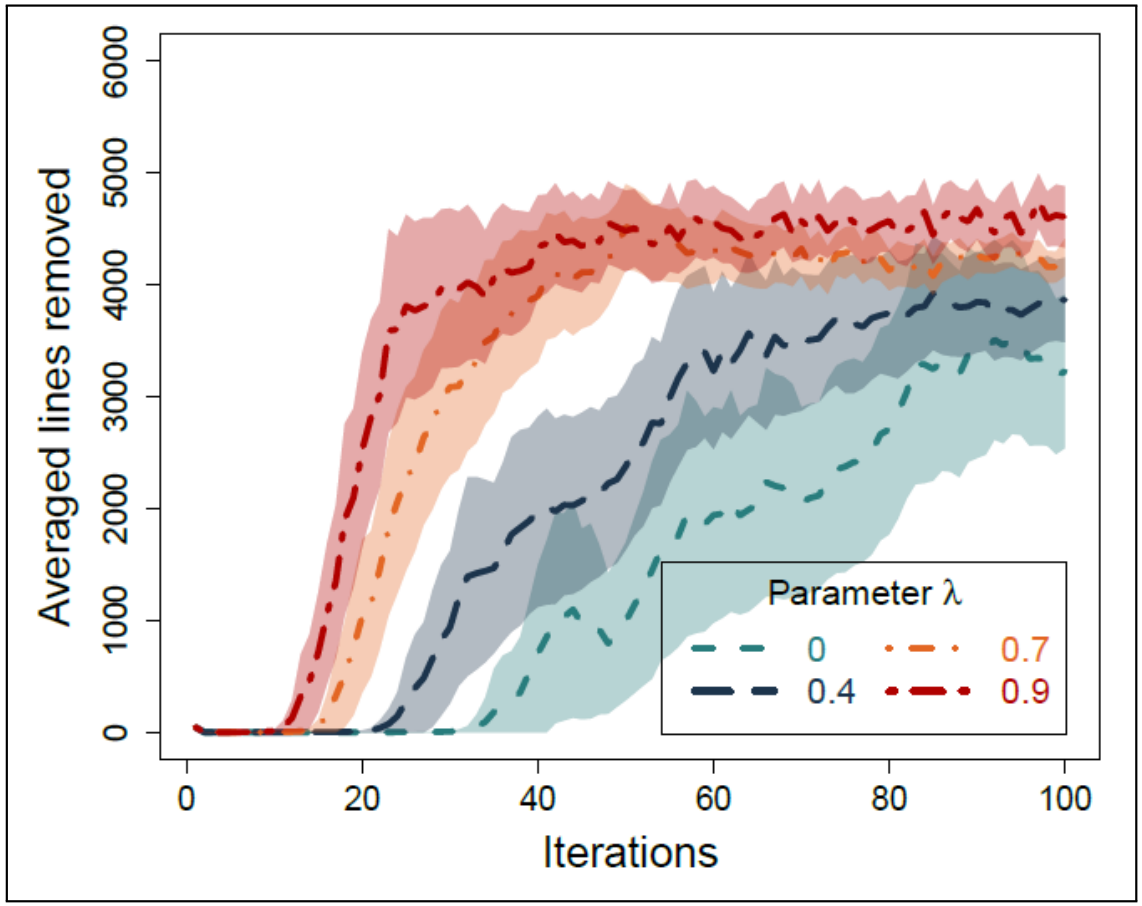


Tetris: large 10×20 board, Bertsekas Features, budget $B = 100,000$

TECHNISCHE UNIVERSITÄT DARMSTADT

- $\lambda-$PI converges to score 4,500

$\rightarrow$ policy search methods perform better than value-function based algorithms (e.g. $\lambda-$PI)



Tetris: best policies

TECHNISCHE UNIVERSITÄT DARMSTADT

- best policies averaging over 10,000 games

Boards | Policies	DU	BDU (by CE)	DT-10	DT-20
Small 10×10 board	3800	4200	**5000**	4300
Large 10×20 board	31,000,000	36,000,000	29,000,000	**51,000,000**

- DT-10 and DT-20: policies learned by CBMPI with D-T Features
- DT-10 is best policy on small board
- DT-20 is best policy on large board

$\rightarrow$ best reported result in the literature (published in August 2015)

B. Scherrer; M. Ghavamzadeh; V. Gabillon; B. Lesner; M. Geist.
Approximate modified policy iteration and its application to the game of tetris.
2015.

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Thanks for your attention!