Reinforcement Learning Methods for Military Applications

Malcolm Strens

Centre for Robotics and Machine Vision
Future Systems Technology Division
Defence Evaluation & Research Agency
U.K.

19 February 2001

RL & Simulation

- Trial-and-error in a real system is expensive
  - learn with a cheap model (e.g. CMU autonomous helicopter)
  - or ...
  - learn with a very cheap model (a high fidelity simulation)
  - analogous to human learning in a flight simulator
- Why is RL now viable for application?
  - most theory developed in last 12 years
  - computers have got faster
  - simulation has improved

RL Generic Problem Description

- States
  - hidden or observable, discrete or continuous.
- Actions (controls)
  - discrete or continuous, often arranged in hierarchy.
- Rewards/penalties (cost function)
  - delayed numerical value for goal achievement.
  - return = discounted reward or average reward per step.
- Policy (strategy/plan)
  - maps observed/estimated states to action probabilities.
- RL problem: "find the policy that maximizes the expected return"

Existing applications of RL

- Game-playing
  - backgammon, chess etc.
  - learn from scratch by simulation (win = reward)
- Network routing and channel allocation
  - maximize throughput
- Elevator scheduling
  - minimize average wait time
- Traffic light scheduling
  - minimize average journey time
- Robotic control
  - learning balance and coordination in walking, juggling robots
  - nonlinear flight controllers for aircraft

Characteristics of problems amenable to RL solution

- Autonomous/automatic control & decision-making
- Interaction (outputs affect subsequent inputs)
- Stochasticity
  - different consequences each time an action is taken
  - e.g. non-deterministic behavior of an opponent
- Decision-making over time
  - a sequence of actions over a period of time leads to reward
  - i.e. planning
- Why not use standard optimization methods?
  - e.g. genetic algorithms, gradient descent, heuristic search
  - because the cost function is stochastic
  - because there is hidden state
  - because temporal reasoning is essential

DERA

Potential military applications of RL: examples

- Autonomous decision-making over time
  - guidance against reacting target
  - real-time mission/route planning and obstacle avoidance
  - trajectory optimization in changing environment
  - sensor control & dynamic resource allocation
- Automatic decision-making
  - rapid reaction
    - electronic warfare
  - low-level control
    - flight control for UAVs (especially micro-UAVs)
    - coordination for legged robots
- Logistic planning
  - resource allocation
  - scheduling

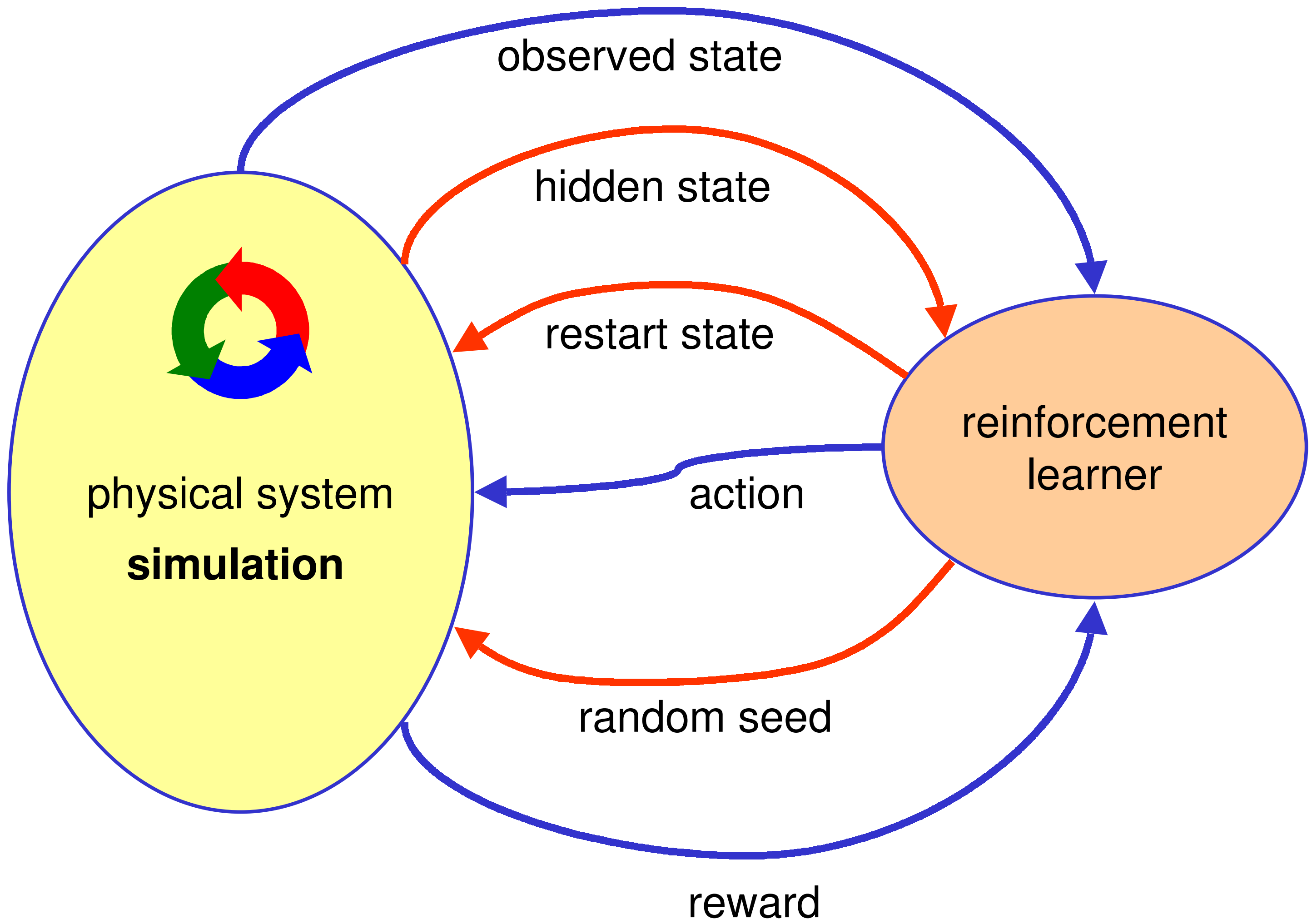
DERA

4 current approaches to the RL problem

- Value-function approximation methods
  - estimate the discounted return for every (state, action)
  - actor-critic methods (e.g. TD-Gammon)
- Estimate a working model
  - estimate a model that explains the observations
  - solve for optimal behavior in this model
  - full Bayesian treatment (intractable) would provide convergence and robustness guarantees
  - certainty-equivalence methods tractable but unreliable
  - the eventual winner in 20 dimensions+ ?
- Direct policy search
  - apply stochastic optimization in a parameterized space of policies
  - effective up to at least 12 dimensions (see pursuer-evader results)
- Policy gradient ascent
  - policy search using a gradient estimate

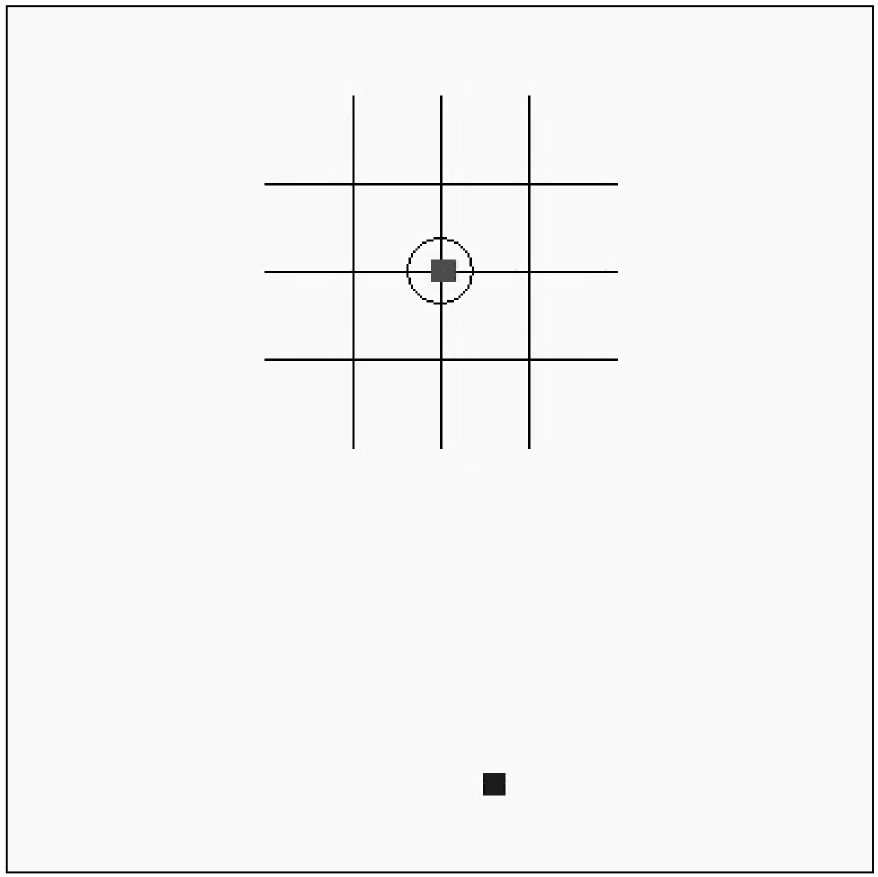
DERA

Learning with a simulation

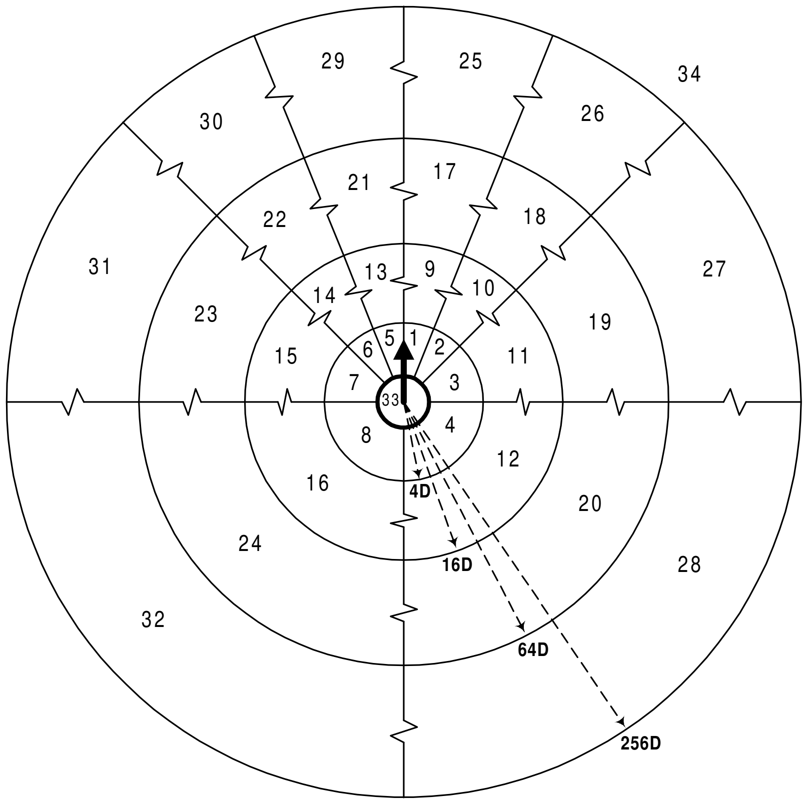


DERA

2D pursuer evader example



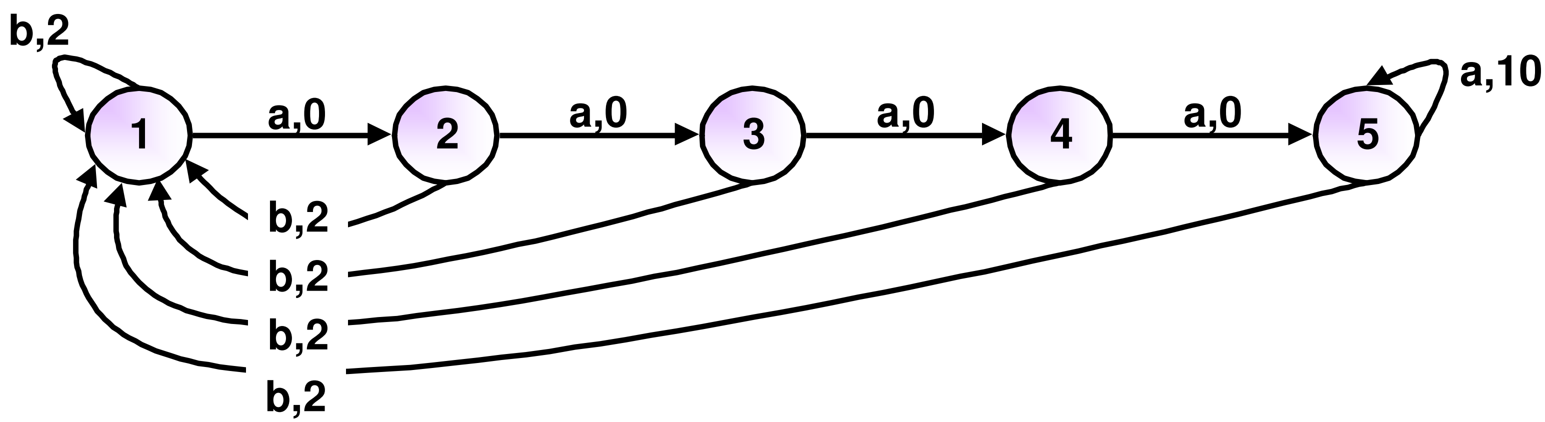
Learning with discrete states and actions



- States: relative position & motion of evader.
- Actions: turn left / turn right / continue.
- Rewards: based on distance between pursuer and evader.

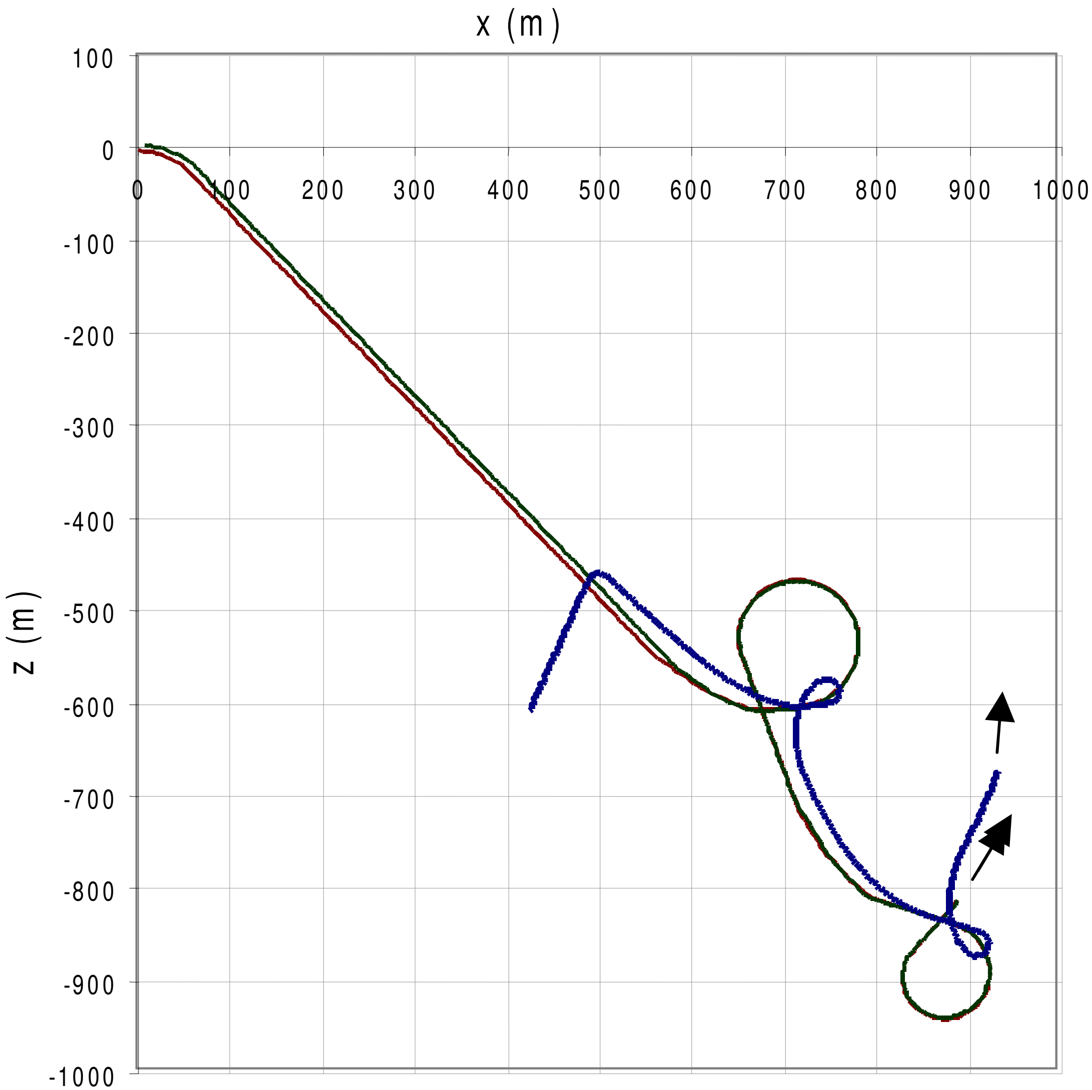
DERA

Markov Decision Process



- (S,A,T,R)
- A: Set of Actions
- S: Set of States
- T: Transition Probabilities $T(s,a,s')$
- R: Reward Distributions $R(s,a)$

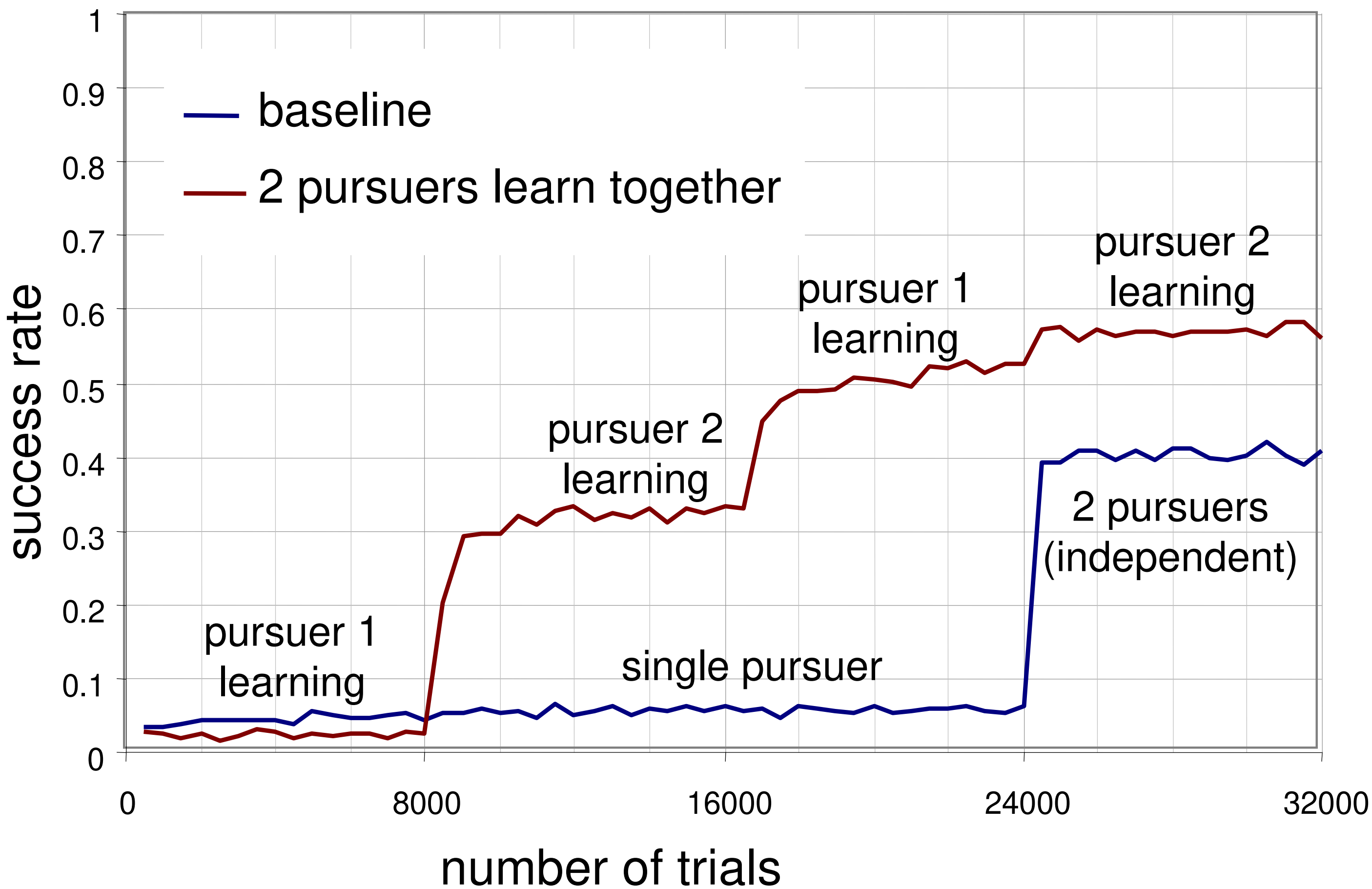
- $Q(s,a)$: Expected discounted reward for taking action a in state s and following an optimal policy thereafter.



2 pursuers - identical strategies

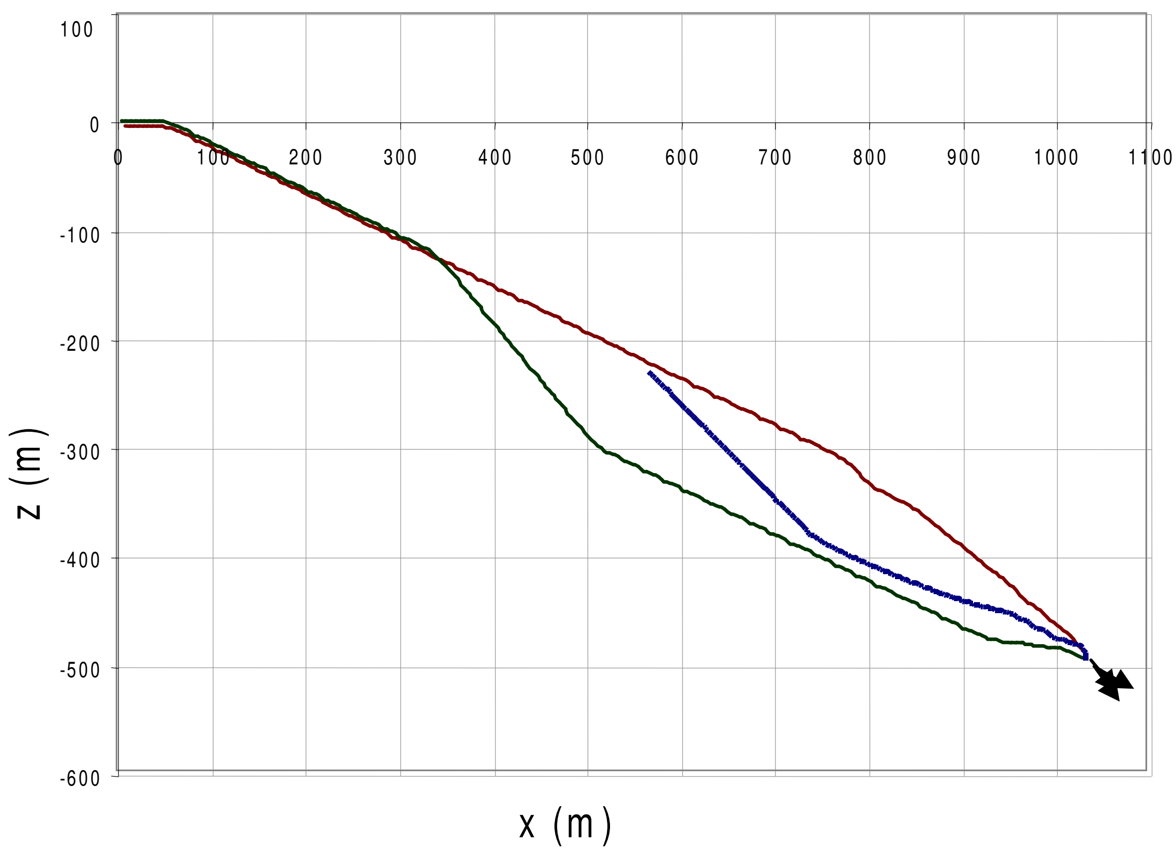
DERA

Learning by policy iteration / fictitious play



DERA

Different strategies learnt by policy iteration (no communication)



DERA

Model-based vs model-free for MDPs

	% of maximum reward (phase 2)			
	Chain	Loop	Maze	
Q-learning (Type 1)	43	98	60	
IEQL+ (Type 1)	*69*	*73*	*13*	*
Bayes VPI + MIX (Type 2)	*66*	*85*	*59*	*
Ideal Bayesian (Type 2)	98	99	94	**

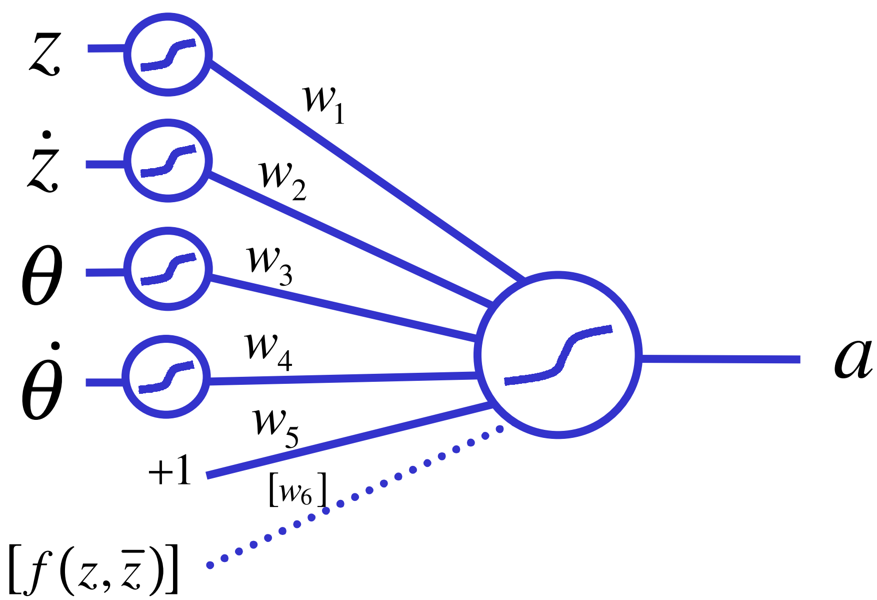
** Dearden, Friedman & Russell (1998)*

*** Strens (2000)*

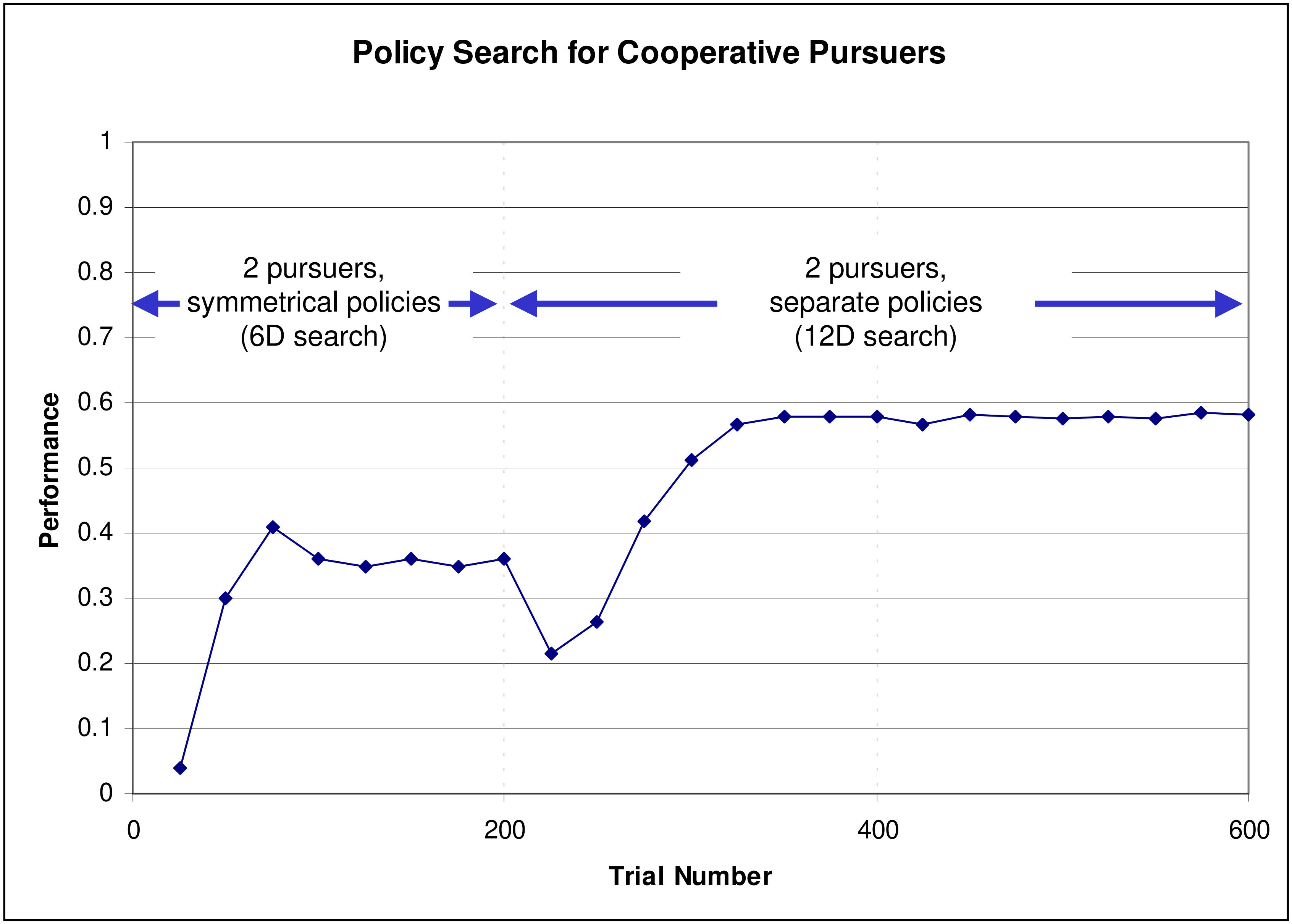
DERA

Direct policy search for pursuer evader

- Continuous state: measurements of evader position and motion
- Continuous action: acceleration demand
- Policy is a parameterized nonlinear function
- Goal: find optimal pursuer policies



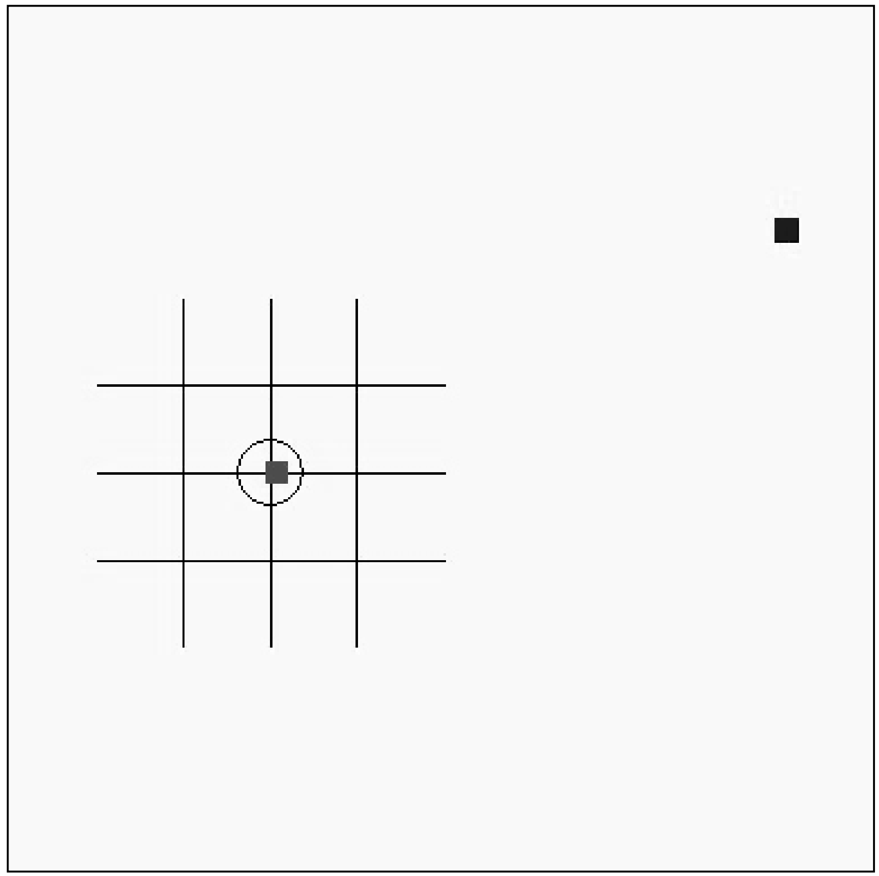
DERA



Policy Search for Cooperative Pursuers
Performance
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2 pursuers,
symmetrical policies
(6D search)
2 pursuers,
separate policies
(12D search)
0
200
400
600
Trial Number

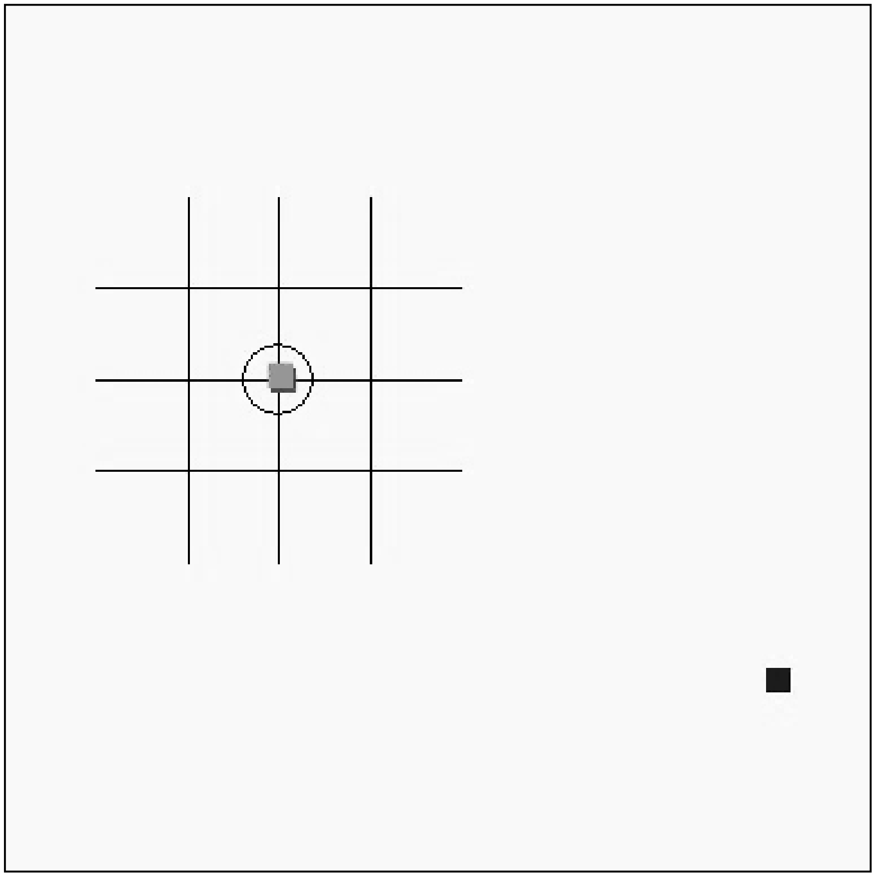
DERA

Policy search: single pursuer

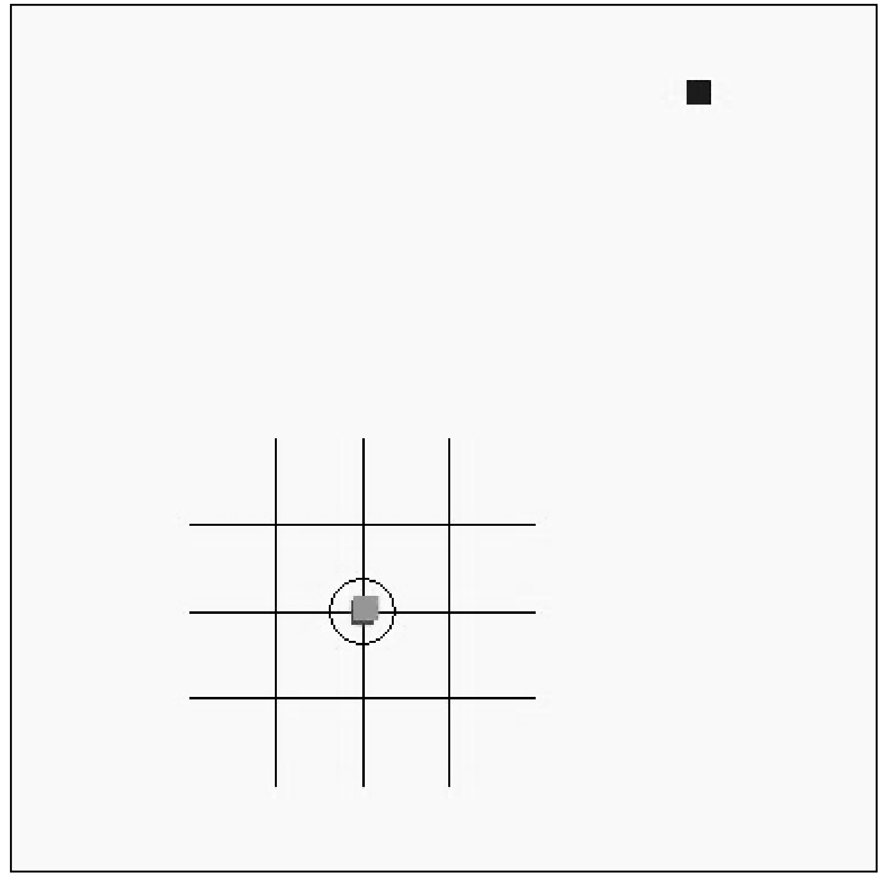


after 200 trials

2 aware pursuers, symmetrical policies



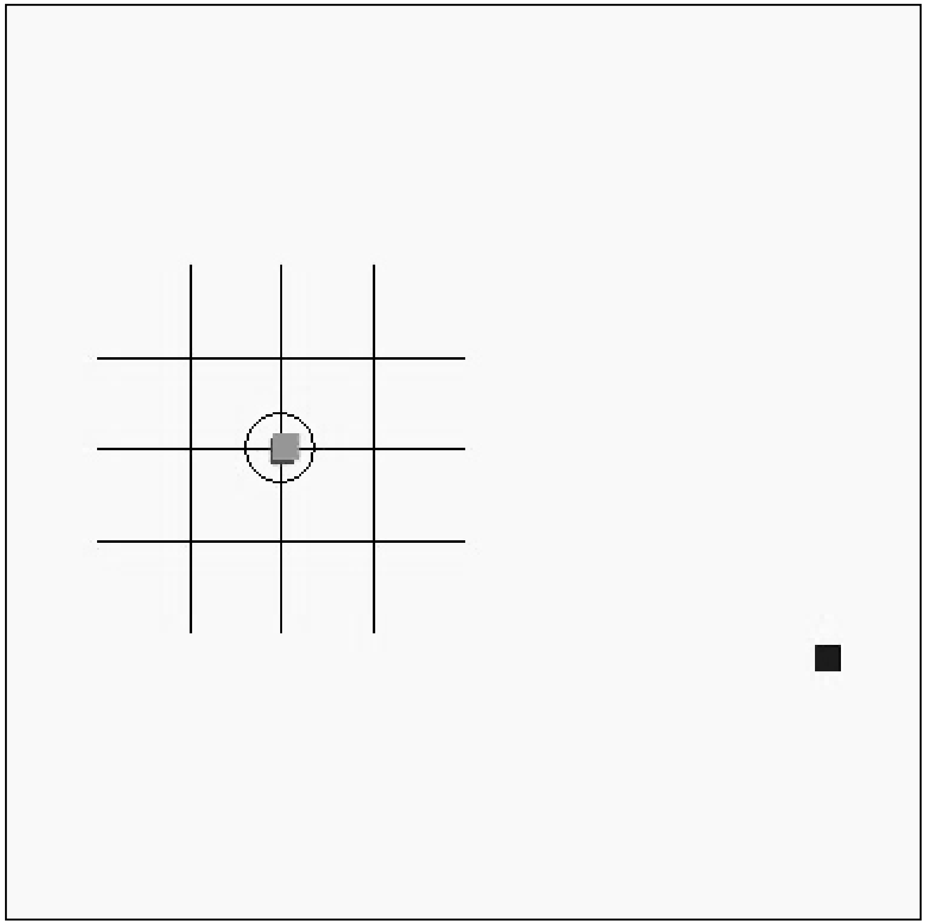
untrained



after 200 trials

DERA

2 aware pursuers, asymmetric policies

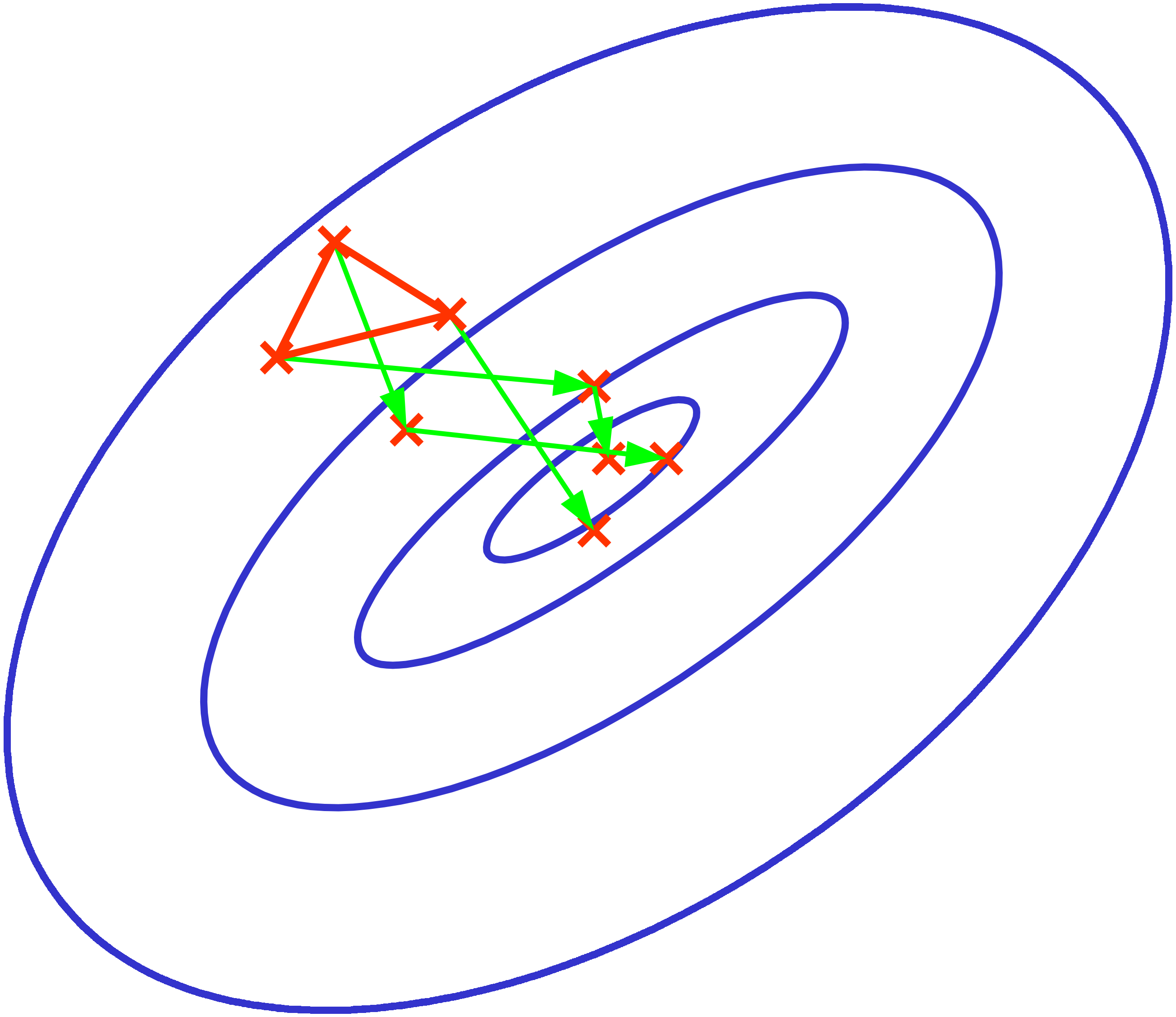


after a further 400 trials

How to perform direct policy search

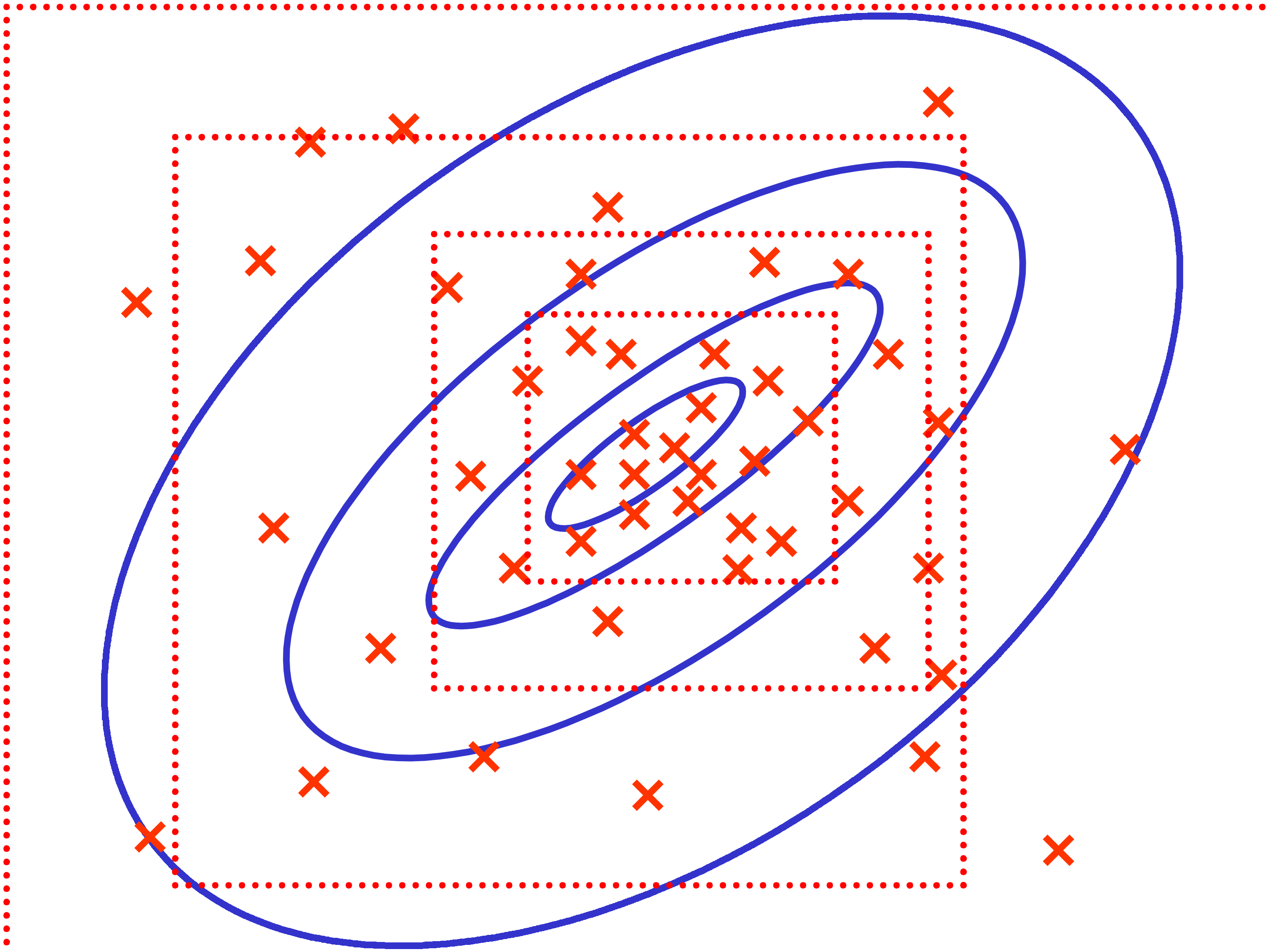
- Optimization Procedures for Policy Search
  - Downhill Simplex Method
  - Random Search
  - Differential Evolution
- Paired statistical tests for comparing policies
  - Policy search = stochastic optimization
  - Pegasus
  - Parametric & non-parametric paired tests
- Evaluation
  - Assessment of Pegasus
  - Comparison between paired tests
  - How can paired tests speed-up learning?
- Conclusions

Downhill Simplex Method (Amoeba)



DERA

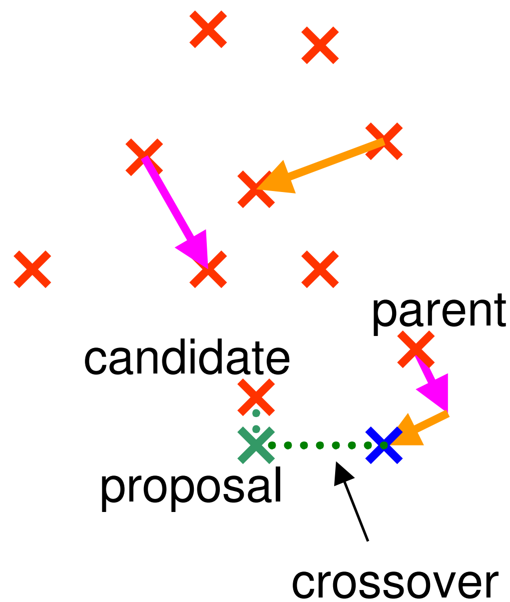
Random Search



DERA

Differential Evolution

- State
  - a population of search points
- Proposals
  - choose candidate for replacement
  - take vector differences between 2 more pairs of points
  - add the weighted differences to a random parent point
  - perform crossover between this and the candidate
- Replacement
  - test whether the proposal is better than the candidate



DERA

Policy search = stochastic optimization

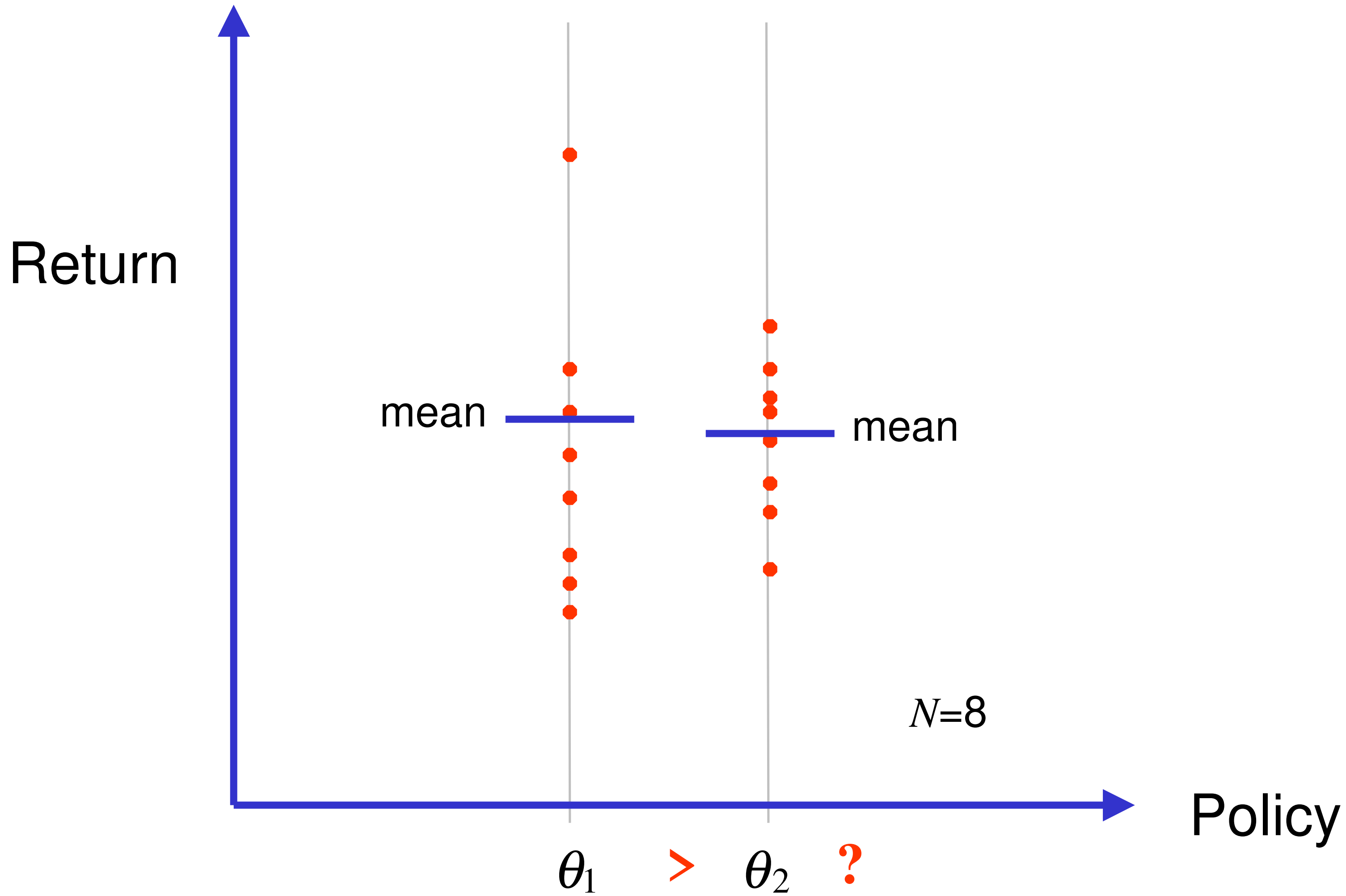
- Modeling return
  - Return from a simulation trial: F_θ
  - + (hidden) starting state $\boldsymbol{x}$: $F_{\theta,\boldsymbol{x}}$
  - + random number sequence y: $f_{\theta,\boldsymbol{x},y}$

- True objective function: $V(\theta) = E[F_\theta] = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} f_i$

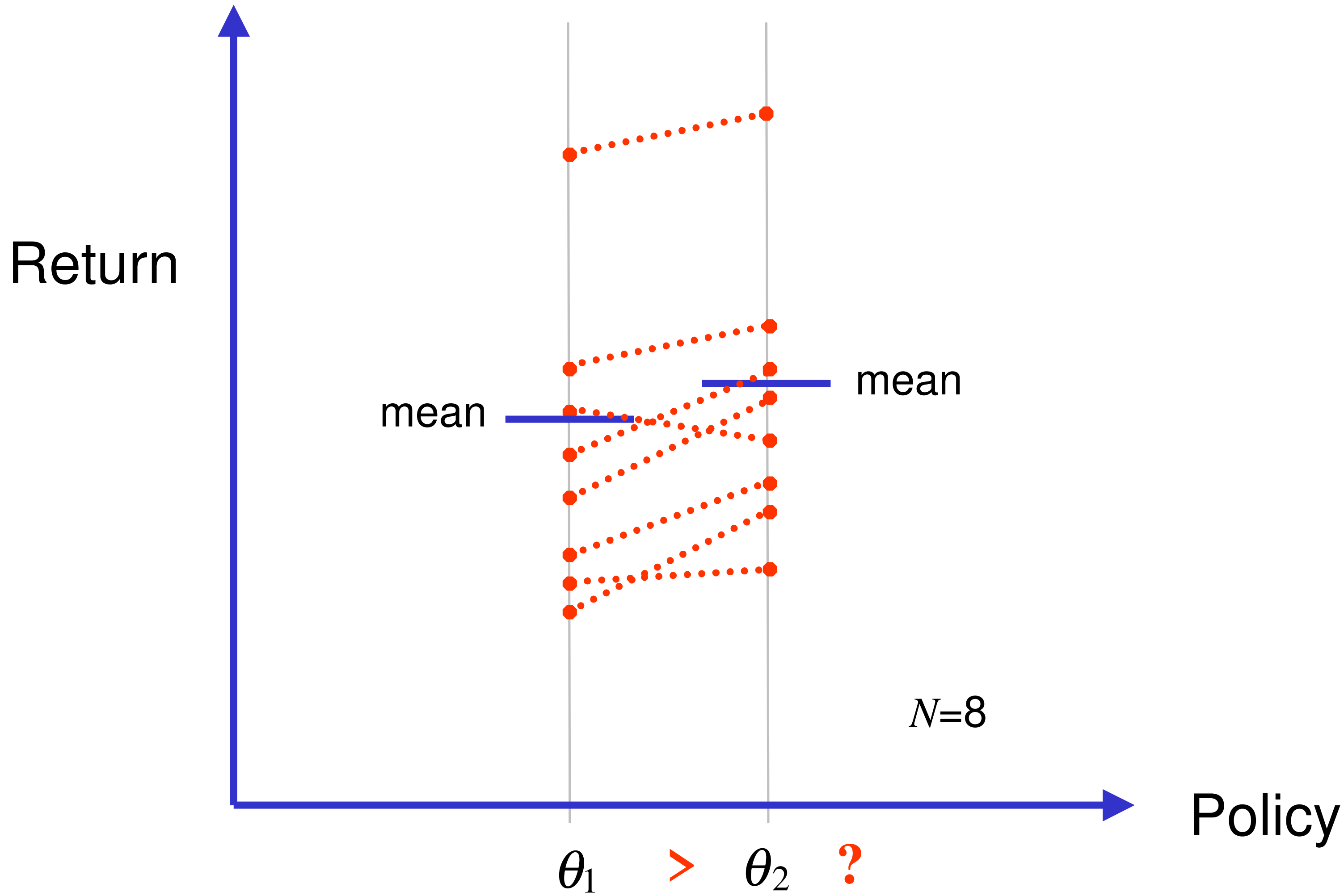
- Noisy objective: N finite

- PEGASUS objective: $V_{PEG}(\theta | \{(\boldsymbol{x}_i, y_i)\}) = \frac{1}{N}\sum_{i=1}^{N} f_{\theta,\boldsymbol{x}_i,y_i}$

Policy comparison: N trials per policy



Policy comparison: paired scenarios



DERA

Policy comparison: Paired statistical tests

- Optimizing with policy *comparisons* only
  - DSM, random search, DE, grid search
  - but not quadratic approximation, simulated annealing, gradient methods
- Paired statistical tests
  - model changes in individuals (e.g. before and after treatment)
  - or the difference between 2 policies evaluated with same start state:

$$D_{\boldsymbol{x}}(\theta_1, \theta_2) \sim F_{\theta_1,\boldsymbol{x}} - F_{\theta_2,\boldsymbol{x}}$$

  - allows calculation of a significance or automatic selection of N
- Paired t test:
  - “is the expected difference non-zero?”
  - the natural statistic; assumes Normality
- Wilcoxon signed rank sum test:
  - non-parametric: “is the median non-zero?”
  - biased, but works with arbitrary symmetrical distribution

Experimental Results (Downhill Simplex Method & Random Search)

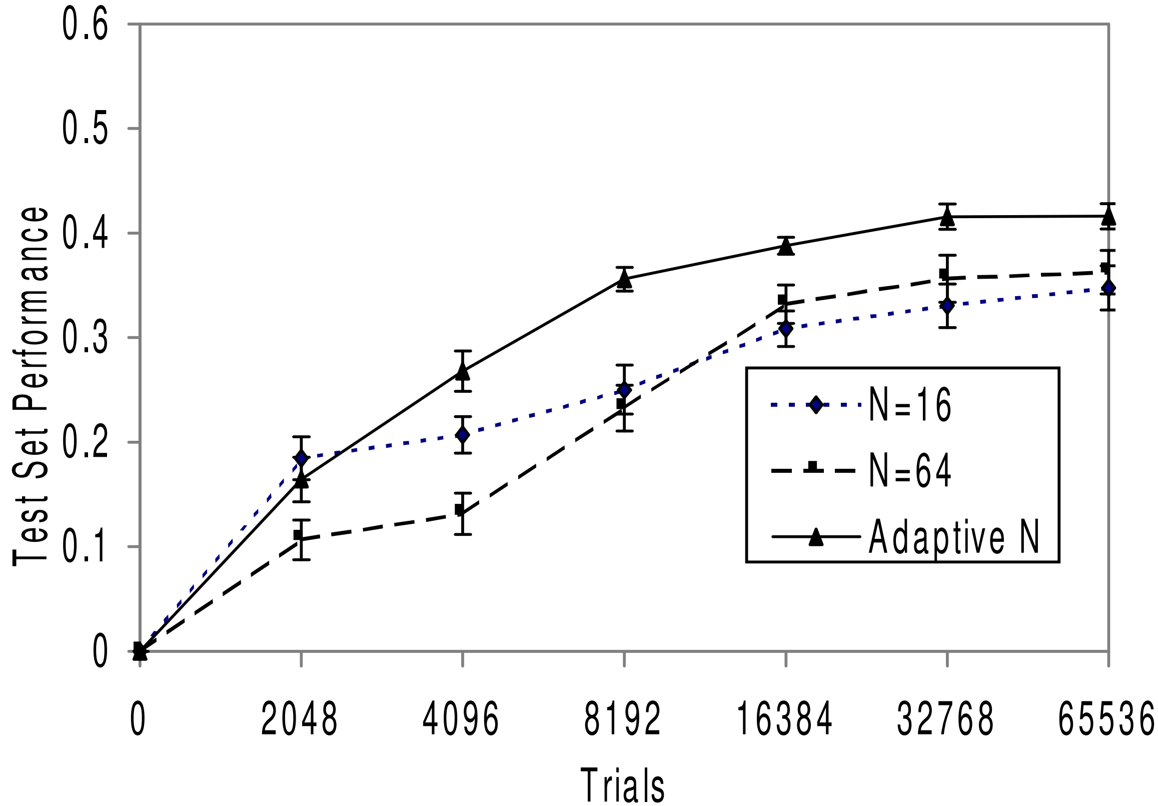
Return (%) $N = 64$	2048 Trials		65536 Trials	
	Training	Test	Training	Test
Random Search	1.5 ± 0.2	7.2 ± 2.0	13 ± 0	28 ± 2
Pegasus	28 ± 1	33 ± 1	60 ± 3	34 ± 2
Pegasus (wx)	5.3 ± 0.1	34 ± 3	46 ± 2	42 ± 1
Scenarios (wx)	4.3 ± 0.2	17 ± 0	44 ± 1	41 ± 1
Unpaired	4.6 ± 0.2	20 ± 1	40 ± 2	40 ± 2

- Pairing accelerates learning
- Pegasus overfits (to the particular random seeds)
- Wilcoxon test reduces overfitting
- Only the start states need to be paired

DERA

Adapting *N* in Random Search

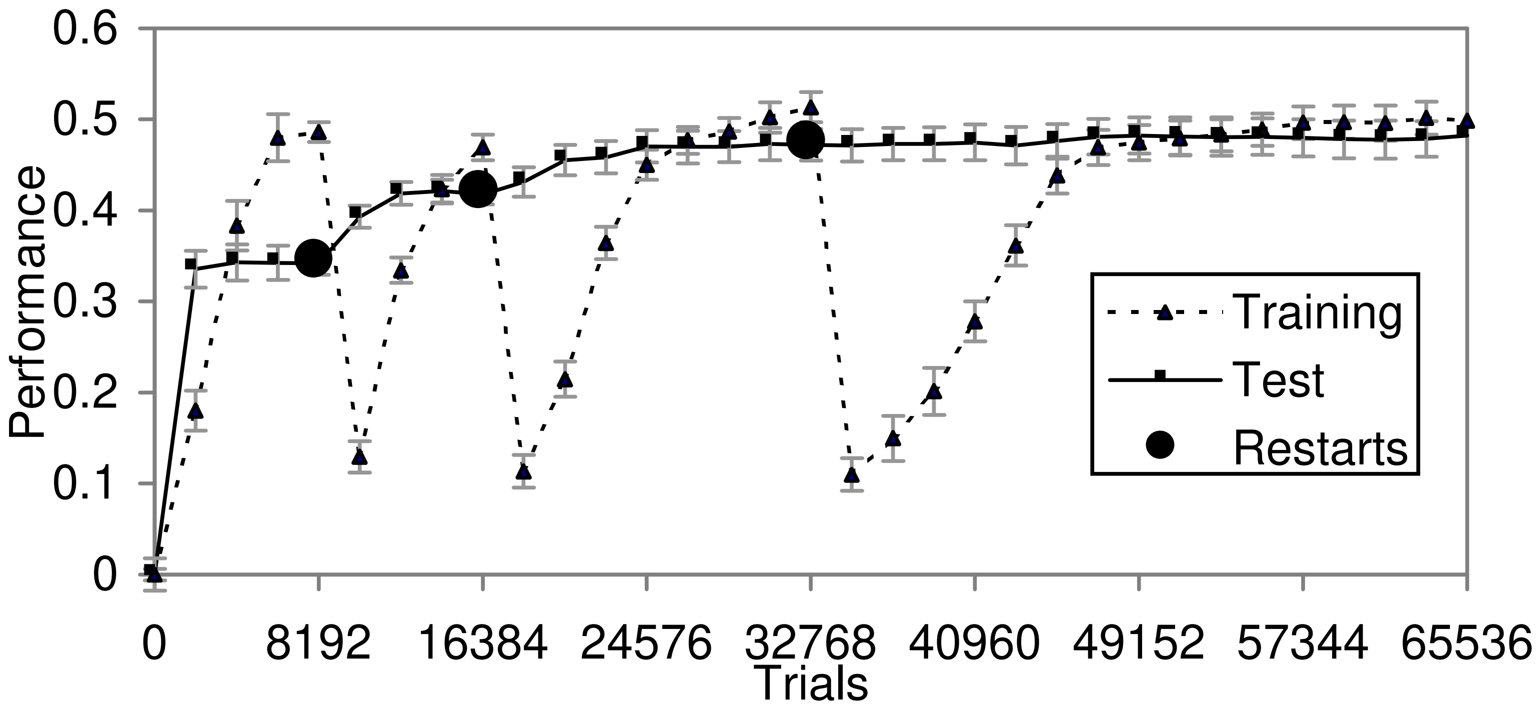
- Paired t test: 99% significance (accept); 90% (reject)
  - Adaptive *N* used on average 24 trials for each policy



DERA

Adapting *N* in the Downhill Simplex Method

- Paired t test: 95% confidence
- Upper limit on *N* increases from 16 to 128 during learning



DERA

Differential Evolution: N=2

- Very small N can be used
  - because population has an averaging effect
  - decisions only have to be >50% reliable
- With unpaired comparisons: 27% performance
- With paired comparisons: 47% performance
  - different Pegasus scenarios for every comparison

- The challenge: find a stochastic optimization procedure that
  - exploits this population averaging effect
  - but is more efficient than DE.

DERA

2D pursuer evader: summary

- Relevance of results
  - non-trivial cooperative strategies can be learnt very rapidly
  - major performance gain against maneuvering targets compared with ‘selfish’ pursuers
  - awareness of position of other pursuer improves performance
- Learning is fast with direct policy search
  - success on 12D problem
  - paired statistical tests are a powerful tool for accelerating learning
  - learning was faster if policies were initially symmetrical
  - policy iteration / fictitious play was also highly effective
- Extension to 3 dimensions
  - feasible
  - policy space much larger (perhaps 24D)

DERA

Conclusions

- Reinforcement learning is a practical problem formulation for training autonomous systems to complete complex military tasks.
- A broad range of potential applications has been identified.
- Many approaches are available; 4 types identified.
- Direct policy search methods are appropriate when:
  - the policy can be expressed compactly
  - extended planning / temporal reasoning is not required
- Model-based methods are more appropriate for:
  - discrete state problems
  - problems requiring extended planning (e.g. navigation)
  - robustness guarantee

DERA