Distributed Intelligent Systems – W11

Machine-Learning Methods Applied to Distributed Robotic Systems
Outline

• Revisiting expensive optimization problems
  – Additional experimental evidence
  – Noise-resistant algorithms in single robot scenarios

• Challenges in multi-robot scenarios
  – Credit assignment problems
  – Co-adaptation strategies
  – Noise-resistance

• Co-adaptation examples in multi-robot obstacle avoidance
Expensive Optimization and Noise Resistance
Expensive Optimization Problems

Two fundamental reasons making robot control design and optimization expensive in terms of time:

1. **Time for evaluation** of candidate solutions (e.g., tens of seconds) >> time for application of metaheuristic operators (e.g., milliseconds)

2. **Noisy performance evaluations** disrupt the adaptation process and require multiple evaluations for actual performance
Expensive Optimization Problems

1. Time for evaluation of candidate >> time for application of metaheuristic operators

- Example: obstacle avoidance
- Robots need to encounter obstacles to learn to avoid them
- Evaluation span 20-60 s depending on size of the arena
- Current processors can execute several million instructions in that time (e.g. ARM Cortex-A9 ~5000 MIPS)

[Di Mario and Martinoli, Robotica, 2014]
Expensive Optimization Problems

2. **Noisy performance evaluations** disrupt the adaptation process and require multiple evaluations for actual performance

- Multiple evaluations at the same point in the search space yield different results
- Example: fitness distribution for obstacle avoidance
- Noise from: sensors, actuators, initial conditions, other robots.
- Noise causes decreased convergence speed and residual error

[Di Mario and Martinoli, *Robotica*, 2014]
Pbest-based Noise-Resistant PSO

Remarks:

• Better assessment of actual performance of a candidate solution through re-evaluation and aggregation of pbest performances over iterations
• Evaluations per iteration:
  – Noise-resistant: 2
  – Regular: 1
• Fair comparison with regular PSO using the same total number of evaluations
Testing Noise-Resistant on Benchmarks

• Benchmark 1: Sphere and Generalized Rosenbrock functions
  – 30 real parameters [Pugh et al., SIS 2005], W10 (biased results)
  – 24 real parameters [Di Mario et al., CEC 2014] today
  – Minimize objective function
  – Expensive only because of noise

• Benchmark 2: obstacle avoidance on a robot
  – 24 real parameters
  – Maximize objective function
  – Expensive because of noise and evaluation time
Benchmark 1: Functions

- Sphere
  \[ f_1(x) = \sum_{i=1}^{D} x_i^2 \]

- Rosenbrock
  \[ f_2(x) = \sum_{i=1}^{D-1} [(1 - x_i^2) + 100(x_{i+1} - x_i^2)^2] \]

- Normalized and bounded to \([0, 1]\)

- Gaussian noise model
  \[ f^g_i(x) = \frac{f_i(x)}{\max f_i} + \mathcal{N}(0, \sigma) \]

- Bernoulli noise model
  \[ f^b_i(x) = \frac{f_i(x)}{\max f_i} + A \cdot \mathcal{B}(p) \]

[Di Mario et al., CEC 2014]
Rosenbrock with Gaussian Noise: Increasing $\sigma$

[Di Mario et al., CEC 2014]

$\sigma = 0$

$\sigma = 0.01$

$\sigma = 0.05$

$\sigma = 0.1$
Increasing Population Size Does Not Help

[Di Mario et al., CEC 2014]
Bernoulli Noise: Positive and Negative Amplitudes

\[ f_i^b(x) = \frac{f_i(x)}{\max f_i} + A \cdot \mathcal{B}(p) \]
Benchmark 2: Obstacle Avoidance on a Mobile Robot

• **Similar** to [Floreano and Mondada 1996]
  – Discrete-time, single-layer, artificial recurrent neural network controller
  – Shaping of neural weights and biases (24 real parameters)
  – fitness function: rewards speed, straight movement, avoiding obstacles

• **Different** from [Floreano and Mondada, 1996]
  – Environment: bounded open-space of 2x2 m instead of a maze

\[
F = V \cdot (1 - \sqrt{\Delta v}) \cdot (1 - i)
\]

\[
0 \leq V \leq 1, \quad 0 \leq \Delta v \leq 1, \quad 0 \leq i \leq 1
\]

\( V \) = average wheel speed, \( \Delta v \) = difference between wheel speeds, \( i \) = value of most active proximity sensor

[Pugh J., EPFL PhD Thesis No. 4256, 2008]
Baseline Experiment: Extended-Time Adaptation

- Compare the basic algorithms with their corresponding noise-resistant version
- Population size 100, 100 iterations, evaluation span 300 seconds (150 s for noise-resistant algorithms) $\rightarrow$ 34.7 days
- Fair test: same total evaluation time for all the algorithms
- Realistic simulation (Webots)
- Best evolved solutions averaged over 30 runs
- Best candidate solution in the final pool selected based on 5 runs of 30 s each; performance tested over 40 runs of 30s each
- Similar performance for all algorithms

Where Can Noise-Resistant Algorithms Make the Difference?

- Limited adaptation time
- Hybrid adaptation (simulation/hardware in the loop)
- Large amount of noise (see Pugh et al., SIS 2005 and later in multi-robot systems)

Notes:
- all examples from shaping obstacle avoidance behavior
- best learned/evolved solution averaged over multiple runs
- fair tests: same total amount of evaluation time for all the different algorithms (standard and noise-resistant)
Limited-Time Adaptation Trade-Offs

- 1 robot, 24 parameters
- Total adaptation time = 8.3 hours (1/100 of previous learning time)
- Trade-offs: population size, number of iterations, evaluation span
- Realistic simulation (Webots)

Varying population size vs. number of iterations

No advantage

- 1 robot, 24 parameters
- Total adaptation time = 8.3 hours (1/100 of previous learning time)
- Trade-offs: population size, number of iterations, evaluation span
- Realistic simulation (Webots)

[Pugh J., EPFL PhD Thesis No. 4256, 2008]
Hybrid Adaptation with Real Robots

- Move from realistic simulation (Webots) to real robots after 90% learning (even faster evolution)
- Compromise between time and accuracy
- Noise-resistance helps manage transition

[Pugh J., EPFL PhD Thesis No. 4256, 2008]
From Single to Multi-Unit Systems: Co-Adaptation in a Shared World
Adaptation in Multi-Robot Scenarios

- **Collective**: fitness become noisy due to partial perception, independent parallel actions
Credit Assignment Problem

With limited communication, no communication at all, or partial perception:

- A robot cannot distinguish between the environmental modifications caused by its own actions from those generated by others.
- Punishments and rewards are likely to be inconsistent.
Co-Adaptation in Collaborative Multi-Robot Systems
Aaxes for Co-Adaptation

Three orthogonal axes to consider (extremities and balanced solutions are possible):

1. **Performance evaluation:**
   - individual vs. group fitness

2. **Solution sharing:**
   - private vs. public policies

3. **Team diversity:**
   - homogeneous (identical controller and hardware) vs. heterogeneous learning
Co-Adaptation Strategies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Performance</th>
<th>Sharing</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-pr-he</td>
<td>individual</td>
<td>private</td>
<td>heterogeneous</td>
</tr>
<tr>
<td>i-pr-ho</td>
<td>individual</td>
<td>private</td>
<td>homogeneous</td>
</tr>
<tr>
<td>i-pu-he</td>
<td>individual</td>
<td>public</td>
<td>heterogeneous</td>
</tr>
<tr>
<td>i-pu-ho</td>
<td>individual</td>
<td>public</td>
<td>homogeneous</td>
</tr>
<tr>
<td>g-pr-he</td>
<td>group</td>
<td>private</td>
<td>heterogeneous</td>
</tr>
<tr>
<td>g-pr-ho</td>
<td>group</td>
<td>private</td>
<td>homogeneous</td>
</tr>
<tr>
<td>g-pu-he</td>
<td>group</td>
<td>public</td>
<td>heterogeneous</td>
</tr>
<tr>
<td>g-pu-ho</td>
<td>group</td>
<td>public</td>
<td>homogeneous</td>
</tr>
</tbody>
</table>

- **Do not make sense (inconsistent)**
- **Possible but not scalable**
- **Interesting (consistent)**
Population-Based Learning Strategies for Multi-Robot Systems

Example of collaborative co-learning with binary encoding of 100 candidate solutions and 2 robots.
Stick-Pulling Case Study: Homogeneous Learning

- See W9 lecture
- Optimization of a single GTP for the whole team
Stick-Pulling Case Study: Heterogeneous Learning

- See W9 lecture
- Learning to specialize the team members (multiple GTPs)
Co-Learning Obstacle Avoidance using PSO
Population-Based Learning Strategies for Multi-Robot Systems
Distributed Learning using PSO

• Standard approach: evaluate candidate solutions on robots but centralize population manager (off-board)
• New approach: distributed also the population manager on the robots (on-board) and share candidate solutions within the neighborhood through communication channels
• Currently: synchronization at the end of an iteration/generation
Varying the Robotic Group Size
Varying the Robotic Group Size
Varying the Robotic Group Size
Varying the Robotic Group Size
Varying the Robotic Group Size

- Same control architecture as [Floreano & Mondada, 1996] (ANN, 24 weights to tune, Khepera III has 9 proximity sensors)
- Same fitness function as [Floreano & Mondada, 1996]
- Similar Webots world as [Pugh et al., 2005] but 3x3 m
- Robot group size: 1, 2, 5, 10
- PSO parameters
  - Swarm size: 10
  - pw = nw = 2.0
  - w = 0.6

Works but bias the results as in [Pugh et al, 2005]

[Pugh and Martinoli, *Swarm Intelligence J.*, 2009]
Varying the Robotic Group Size – Learning vs. Testing Environment

- Gradually increase number of robots on team
- Up to 10x faster learning with little performance loss
- Arena 3x3 m

Learned in a group of 10 robots (10x faster), final evaluation as single robot

Learned as single robot, final evaluation as single robot

[Pugh and Martinoli, Swarm Intelligence J., 2009]
Distributed Learning with Real Robots (Pugh, 2008)

Before learning (5x speed-up)
Distributed Learning with Real Robots (Pugh, 2008)

After learning (5x speed-up)
Co-Learning Obstacle Avoidance – Communication-Based Neighborhoods
Standard Index-Based Neighborhood

• Default neighborhood: ring topology, 2 fixed index-based neighbors for each particle

• Problem for real robots: neighbors could be very far away
Index-Based Neighborhood

A possible robot distribution

Ring Topology - Standard
Communication-Based Neighborhoods

• Default neighborhood - ring topology, 2 fixed index-based neighbors for each particle

• Problem for real robots: neighbor could be very far away

• Possible solutions:
  – use two closest robots in the arena (capacity limitation)
  – use all robots within some radius r (range limitation)

• Reality is affected often by both capacity and range
Communication-Based Neighborhoods

Model A: 2-closest robots (capacity limitation)

Model B: robots within range $r$ (range limitation)
Performance Comparison
using Different Neighborhoods for 10 Robots

Simulation (Webots)  
Real robots

[Pugh and Martinoli, *Swarm Intelligence J.*, 2009]
Re-Assessing Noise-Resistant Algorithms in Multi-Robot Systems
Where Can Noise-Resistant Algorithms Make the Difference?

- Large amount of noise (typically accentuated in multi-robot systems without centralized coordination)
- Limited adaptation time
- Hybrid adaptation (simulation/hardware in the loop)

Notes:
- all examples from shaping obstacle avoidance behavior
- best learned solution averaged over multiple runs
- fair tests: same total amount of evaluation time for all the different algorithms (standard and noise-resistant)
Increasing Number of Robots: Impact of Noise Resistance

- Webots experiments
- 1x1 m arena (high density!)
- Fair test: same amount of total evaluation time for each bar
- Performance decreases with number of robots (more difficult to avoid in overcrowded arenas)
- Noise-resistance make the difference in high density (i.e. noisier) scenarios

[Di Mario and Martinoli, *Robotica*, 2014]
Impact of Limited Time Adaptation

- Webots experiments
- 1x1 m arena (high density!)
- full-time adaptation: 417 h
- limited time adaptation: 8h
- 52 times smaller evaluation time, 17% max drop in performance
- same obstacle avoidance strategy

**Recipe:**
1. Evaluation span include at least 1 interaction
2. Swarm size = dimension of parameter space
3. Use noise-resistant algorithms
4. Dedicate max time budget to iterations

[Di Mario and Martinoli, *Robotica*, 2014]
Hybrid Adaptation vs. Only Real Robots

- Noise-resistant PSO
- 4 robots
- Hybrid: 30 iterations in simulation, then 30 iterations on real robots
- Achieves similar fitness as running 60 iterations on real robots
- Requires half the real robot evaluation time

[Di Mario and Martinoli, *Robotica*, 2014]
Why Noise-Resistant Algorithms Make the Difference?

Standard PSO vs. A-Posteriori evaluations

[Di Mario et al., CEC 2014]
Why Noise-Resistant Algorithms Make the Difference?

Noise-Resistant PSO vs. A-Posteriori evaluations

[Di Mario et al., CEC 2014]
Noise-Resistant Algorithms in Multi-Robot Systems: From Pbest-based strategies to OCBA
Benchmark Task: Obstacle Avoidance

- 24 parameters of Artificial Neural Network
- Usual fitness function [Floreano and Mondada, 1996]
Standard PSO: no re-evaluations

- Over-estimation
- Stagnation
- Iterations wasted
Approach 1: PSO rep

1: Initialize particles
2: \textbf{for} \ N_i \ \textbf{iterations do}
3: \hspace{1cm} \textbf{for} \ N_p \ \textbf{particles do}
4: \hspace{2cm} \textbf{Evaluate particle position}
5: \hspace{1cm} Update personal best
6: \hspace{1cm} Update neighborhood best
7: \hspace{1cm} Update particle position
8: \hspace{1cm} \textbf{end for}
9: \textbf{end for}

- Each function evaluation replaced by average of $k$ evals
- PSO algorithm not changed
- If noise is Gaussian, std dev reduced by a factor of $\sqrt{k}$
PSO rep10: 10 re-evaluations

- 10x less iterations
- Less over-estimation
- Less stagnation
Approach 2: PSO \textit{pbest}

1: Initialize particles
2: \textbf{for} $N_i$ iterations \textbf{do}
3: \hspace{1em} \textbf{for} $N_p$ particles \textbf{do}
4: \hspace{2em} Update particle position
5: \hspace{2em} Evaluate particle
6: \hspace{2em} \textbf{Re-evaluate personal best}
7: \hspace{2em} Aggregate with previous best
8: \hspace{2em} Share personal best
9: \hspace{1em} \textbf{end for}
10: \textbf{end for}

[Pugh and Martinoli, \textit{Swarm Intelligence J.}, 2009]
PSO *pbest*

- No stagnation
- Random drops: poor estimates of new candidates
- Still overestimation
Approach 3: OCBA

- Chen et al [1]: select the number of samples $N_i$ according to:

$$\frac{N_i}{N_j} = \left( \frac{\sigma_i}{\delta_{b,i}} \right)^2, \quad i \neq j \neq b$$

$$N_b = \sigma_b \sqrt{\sum_{i=1,i\neq b}^{k} \frac{N_i^2}{\sigma_i^2}}$$

- Proven that maximizes probability of correct selection of best candidate $b$ for infinite $T$
- But “works well in practice” for finite $T$
- Intuition: more samples for candidates with:
  - higher variance
  - mean closer to the best (low delta) $\delta_{i,j} = \bar{X}_i - \bar{X}_j$

OCBA in practice

\[ \{ \bar{X}_1, \ldots, \bar{X}_k \} \]
\[ \{ \sigma_1^2, \ldots, \sigma_k^2 \} \]

\[ \text{OCBA} \]
\[ \{ N_1, N_2, \ldots, N_k \} \]

- Use empirical means and std devs as estimates for OCBA
  1) Sample all candidates \( n_0 \) times
  2) Calculate initial empirical means and std devs
  3) While there is budget left:
     - Allocate \( \Delta \) additional samples using OCBA
     - Evaluate the new samples
     - Update means and std devs
     - Reduce budget by \( \Delta \)
- Parameter \( \Delta \) controls the number of allocation steps
Approach 3: Centralized PSO OCBA

1: Initialize particles
2: for $N_i$ iterations do
3:   for $N_p$ particles do
4:     Evaluate new particle position $n_0$ times
5:   end for
6: remaining budget := iteration budget - $n_0 \cdot N_p$
7: while remaining budget $> 0$ do
8:   Allocate $\Delta$ samples among current positions and personal bests using OCBA
9:   Evaluate allocated samples
10:  Recalculate mean and variance for new evaluations
11:  remaining budget := remaining budget - $\Delta$
12: end while
13: for $N_p$ particles do
14:   Update personal best
15: end for
16: Update neighborhood best
17: Update particle position
18: end for

Number of candidates is twice the total number of particles
- No stagnation, no overestimation
Approach 4: Distributed PSO OCBA

- Each particle conducts its own OCBA allocation
- Candidates for OCBA are new positions and pbests in neighborhood
- N candidates = 2 * Neighborhood size
- Mean and standard deviation can be calculated online by storing only the previous values and the number of samples
- Memory and communication overhead is small and constant
Distributed PSO OCBA

1: Initialize particle
2: for $N_t$ iterations do
3: Evaluate new particle position $n_0$ times
4: Share evaluation results in neighborhood
5: Receive and store evaluation results from neighborhood
6: remaining budget := iteration budget - $n_0 \cdot N_p$
7: while remaining budget $> 0$ do
8: Allocate $\Delta$ samples among current positions and personal bests in neighborhood using OCBA
9: Evaluate allocated samples
10: Recalculate mean and variance for new evaluations
11: Share evaluation results in neighborhood
12: Receive and store evaluation results from neighborhood
13: remaining budget := remaining budget - $\Delta$
14: end while
15: Update personal best
16: Update neighborhood best
17: Update particle position
18: end for

Number of candidates is twice the PSO neighborhood size.
PSO OCBA D

- No stagnation
- Very little overestimation, still higher than centralized OCBA
Summary of Results

- Despite a significant better estimation ("red" lines) of OCBA techniques, all noise resistant algorithms lead in this scenario to only a slight increase of the absolute performance ("blue" lines).
- Performance of learned controller very competitive with an engineered approach based on potential fields (see below for PSO OCBA D).
Conclusion
Take Home Messages

• The cost of an optimization problem is heavily influenced by the amount of noise in the evaluation function, the time needed for evaluating a candidate solution, and the dimension of the parameter space.

• Collaborative co-adaptation strategies can be differentiated along three axes: public/private solutions; homogeneous/heterogeneous system, individual/group performance.

• Multi-robot platforms can be exploited for testing in parallel multiple candidate solutions.

• One way to bypass the credit assignment problem in multi-robot contexts is to enforce homogeneity and reward group performance.

• PSO appears to be well suited for fully distributed on-board operation and fairly robust to small pools of candidate solutions.

• A series of noise-resistant techniques have been presented for dealing with noisy problems in multi-robot systems.
Books


Papers