Distributed Intelligent Systems – W12
Machine-Learning Methods
Applied to Distributed Robotic Systems
Outline

• Expensive optimization problems
  – Rationale
  – Experimental evidence
  – Noise-resistant algorithms in single robot scenarios

• Challenges in multi-robot scenarios
  – Credit assignment problem
  – Co-optimization strategies
  – Noise-resistance

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

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

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

2. **Noisy performance evaluations** disrupt the optimization process and require multiple evaluations for assessing the actual performance of a candidate solution
Reducing Evaluation Time

Variant 1: exploit more abstracted, calibrated representations (models at various levels)

See also multi-level modeling lectures (W9 and W10)
Reducing Evaluation Time

**Variant 2**: minimal evaluation span for providing enough optimization opportunities

- Example: obstacle avoidance, usual fitness function
- A single robot 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 billions of instructions in that time (e.g. ARM Cortex-A9 ~5000 MIPS)

[Di Mario and Martinoli, *Robotica*, 2014]
Dealing with Noisy Evaluations

- 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]
Dealing with Noisy Evaluations

- Better information about a candidate solution can be obtained by combining multiple noisy evaluations.
- We could evaluate systematically each candidate solution for a fixed number of times → not efficient from a computational perspective.
- In particular in a population-based optimization method (but similar mechanisms are applicable to hill-climbing evaluative learning), we want to dedicate more computational time to evaluate promising solutions and eliminate as quickly as possible the “lucky” ones.
- Idea: re-evaluate and aggregate → each candidate solution might have been evaluated a different number of times → compare the aggregated value.
- For instance, in GA good and robust candidate solutions survive over generations; in PSO they survive in the individual memory.
- Use dedicated functions for aggregating multiple evaluations: e.g., minimum and average or more generalized aggregation functions (e.g., quasi-linear weighted means), perhaps combined with a statistical test for comparing resulting aggregated performances.
Pbest-based Noise-Resistant PSO

- Better assessment of actual performance of a candidate solution through **re-evaluation** and **aggregation** of pbest performances over iterations; aggregation examples: average (avg) and minimum (min)
- Evaluations per particle and per iteration:
  - Noise-resistant: 2
  - Regular: 1
- Fair comparison with regular PSO using the **same total number of evaluations**
Testing Noise-Resistant Algorithms on Benchmarks

• Benchmark 1: Sphere and Rosenbrock functions
  – 30 real parameters [Pugh et al., SIS 2005] (biased results)
  – 24 real parameters [Di Mario et al., CEC 2014]
  – 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: Early Results

\[ f'_j(\bar{x}) = \mathcal{N}(0, \sigma^2) + f_j(\bar{x}) \]

\( f_j = \text{Rosenbrock} \)

Fair test: **same**

number of evaluations candidate solutions for all algorithms

(i.e. \(N\) generations/iterations of standard versions compared with \(N/2\) of the noise-resistant ones)

Biased results: low number of runs (20) and population size (20) < search dimension (30)!

[Pugh et al., SIS 2005]
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**
  \[ f^b_i(x) = \frac{f_i(x)}{\max f_i} + A \cdot \mathcal{B}(p) \]
Rosenbrock with Gaussian Noise: Increasing $\sigma$

$\sigma = 0$

$\sigma = 0.01$

$\sigma = 0.05$

$\sigma = 0.1$

[Di Mario et al., CEC 2014]

Note: Fair number of evaluations for each algorithm
Increasing Population Size Does Not Help

- Results in contrast with best practices in GA
- Other parameters constant (i.e. more evaluations for larger populations)

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

- Negative amplitudes (i.e. good spurious evaluation of bad solutions) have a more significant disruptive impact of positive amplitudes (bad spurious evaluation of good solutions)
- Noise-resistant algorithms can cope with this large negative outliers

\[ f_i^b(x) = \frac{f_i(x)}{\max f_i} + A \cdot B(p) \]

[Di Mario et al., CEC 2014]
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

[Di Mario and Martinoli, *Robotica*, 2014]
Benchmark 2: Early Results

- **Scenario 1**: One robot learning obstacle avoidance
- **Scenario 2**: One robot learning obstacle avoidance, one robot running pre-learned obstacle avoidance
- **Scenario 3**: Two robots co-learning obstacle avoidance

Idea: more robots more noise (as perceived from an individual robot) because there is no explicit communication between the robots (in scenario three even more noisy because both robots are learning independently).

[1x1 m arena, PSO, iteration 50, scenario 3]
Benchmark 2: Early Results

Note: Fair number of evaluations for each algorithm
From Single to Multi-Unit Systems: Co-Optimization in a Shared World
Optimization 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:

If 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-Optimization in Cooperative Multi-Robot Systems
Axes for Co-Optimization

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 controllers) vs. heterogeneous (diverse controllers possible)
## Co-Optimization 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>

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

Example of co-optimization strategies with binary encoding of 100 candidate solutions and 2 robots
From W10 (Stick-Pulling Case Study): Heterogeneous vs. Homogenous Optimization

Performance ratio between heterogeneous (full and 2-castes) and homogeneous groups AFTER optimization

Notes:
• large $T_m$ (long averaging window)
• only private strategies
• global = group
  local = individual

[Li et al., Adaptive Behavior, 2004]
Stick-Pulling Case Study: Homogeneous Optimization

- See W10 lecture
- Optimization of a single GTP for the whole team
Stick-Pulling Case Study: Heterogeneous Optimization

- See W10 lecture
- Shaping specialization of the team members (multiple GTPs)

Viable for exploring heterogeneous solutions

Heterogeneity allowed but eventually roughly homogeneous solution via shuffle around of candidate solutions

Homogeneity enforced

Not scalable

Local on s. 26

Global on s. 26
Co-Optimization of Obstacle Avoidance using PSO
Population-Based Optimization Strategies for Multi-Robot Systems

Chosen for obstacle avoidance optimization
Distributed Optimization using PSO

- Standard approach: evaluate candidate solutions on robots but centralize swarm manager (off-board)
- New approach: distributed also the swarm 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 – Optimization vs. Validation Environment

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

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

More obstacles

Optimized as single robot, final evaluation as single robot

[Pugh and Martinoli, *Swarm Intelligence J.*, 2009]
Distributed Optimization with Real Robots

[Pugh and Martinoli, *Swarm Intelligence J.*, 2009]

Before optimization (5x speed-up)
Distributed optimization with Real Robots

[Pugh and Martinoli, *Swarm Intelligence J.*, 2009]

After optimization (5x speed-up)
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 tight coordination)
• Limited optimization time
• Hybrid optimization (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 Optimization

- 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 Optimization 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 (Obstacle avoidance, 4 robots)

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

Noise-Resistant PSO vs. A-Posteriori evaluations (Obstacle avoidance, 4 robots)

[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 (D= 24)
- Usual fitness function [Floreano and Mondada, 1996]
- [Di Mario et al., ICRA 2015 and CEC 2015]
Standard PSO: no re-evaluations

- \( D = 24 \)
- 24 particles
- 500 iterations
  (24 eval./iteration)
- Over-estimation
- Stagnation
- Iterations wasted
Approach 1: PSO rep

1: Initialize particles
2: for $N_i$ iterations do
3: for $N_p$ particles do
4: Evaluate particle position
5: Update personal best
6: Update neighborhood best
7: Update particle position
8: end for
9: 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 (24x10 = 240 evaluations/iteration)
- Less over-estimation
- Less stagnation
Approach 2: PSO $p_{best}$

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

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

- 2x less iterations (2x24 = 48 evaluations/iteration)
- No stagnation
- Random drops: poor estimates of new candidates
- Still overestimation
Approach 3: OCBA

• Chen et al [1] : select the number of samples (evaluations) $N_i$ per candidate $i$ according to:

\[
\frac{N_i}{N_j} = \left( \frac{\sigma_i / \delta_{b,i}}{\sigma_j / \delta_{b,j}} \right)^2, \quad i \neq j \neq b \quad N_b = \sigma_b \sqrt{\sum_{i=1, i \neq b}^k \frac{N_i^2}{\sigma_i^2}}
\]

• Intuition: more samples for candidates with:
  – higher variance
  – mean closer to the best (low delta) $\delta_{i,j} = \bar{X}_i - \bar{X}_j$

• Proven that maximizes probability of correct selection of best candidate $b$ for infinite total number of samples (evaluations) $T$

• But “works well in practice” for finite number of samples (evaluations) $T$

OCBA in Practice

- 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: remainder budget := iteration budget - $n_0 \cdot N_p$
7: while remainder 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: remainder budget := remainder budget $- \Delta$
12: end while
13: for $N_p$ particles do
14:   Update personal best
15:   Update neighborhood best
16:   Update particle position
17: end for
18: end for

$n_0 = 2$, 24 particles $\rightarrow$
$2 \times 24 = 48$ evaluations

Number of candidate solutions is twice the total number of particles

$\Delta = 4 \rightarrow$
$4 \times 2 \times 24 = 196$ evaluations

$48 + 196 = 240$ eval./iteration
Centralized PSO OCBA

- 10x less iterations (240 evaluations/iteration)
- No stagnation, no overestimation
- Can distribute on multi-robots with global networking
Approach 4: Distributed PSO OCBA

- 10x less iterations (240 evaluations/iteration)
- No stagnation
- Very little overestimation, still higher than centralized OCBA
- Can distribute on multi-robot with local networks
Summary of Results

Despite a significant better estimation ("red" lines in the previous slides) of OCBA techniques, all noise-resistant algorithms lead in this scenario to only a slight increase of the absolute performance ("blue" lines in the previous slides)

[Di Mario et al., CEC 2015]
Conclusion
Take Home Messages

• The cost of an optimization problem is heavily influenced by the dimension of the parameter space but also by the amount of noise in the evaluation function and the time needed for evaluating a candidate solution.
• Optimization problems in robotics are computationally expensive because of long and noisy evaluation of candidate solutions.
• Cooperative co-optimization strategies can be differentiated along three axes: public/private strategies; homogeneous/heterogeneous optimization, individual/group performance.
• Multi-robot platforms can be exploited for evaluating 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 optimization and fairly robust to small pools of candidate solutions.
• A series of noise-resistant techniques have been presented for dealing with noisy optimization problems in multi-robot systems.
Additional Literature – Week 12

Books

Papers