Multi-Level Modeling Methods as a Tool for Design and Control of Swarm Robotic Systems and Beyond

Grégory Mermoud and Alcherio Martinoli

School of Architecture, Civil and Environmental Engineering
Distributed Intelligent Systems and Algorithms Laboratory
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
http://disal.epfl.ch/

gregory.mermoud@epfl.ch and alcherio.martinoli@epfl.ch

ICRA 2010, Workshop on Stochasticity in Robotics and Biology
A lot of minimalist agents may offer better performance than a few complex agents.

How to design and control these systems, especially when they involve very resource-constrained robots?
Case studies

- **Collaborative stick-pulling**
  - How to deal with limited *actuation* capabilities of individual robots?
  - **General recipe** for designing and controlling distributed robotics systems

- **Collaborative identification and destruction of spots**
  - How to deal with limited actuation and *sensing* capabilities of individual robots?
  - **More specific insights** into our methodology
A Seminal Case Study: The Collaborative Stick-Pulling Experiment

- [Martinoli and Mondada, ISER, 1995]
- [Ijspeert et al., AR, 2001]
- [Lerman, Galstyan, Martinoli, Ijspeert, Artificial Life, 2001]
- [Martinoli et al., IJRR 2004]
Collaborative Stick-Pulling

Physical Set-Up

- 2-6 robots
- 4 sticks
- 40 cm radius arena

Collaboration via indirect communication

Control parameter: gripping time

Proximity sensors

Arm elevation sensor
Two Model-Based Approaches

1. A top-down (reverse) approach
   – From the **collective system** to the **individual units**
   – Design collective behavior in an **idealized setting**
   – Use (often **deterministic**) global-local mapping techniques for distributed control (e.g., market-based, graph-based)
   – The ideal system performance approximates the actual one if **assumptions** are fulfilled

2. A bottom-up (forward) approach
   – From the **individual units** to the **collective system**
   – Simple robust heuristic control techniques (e.g., behavior-based)
   – Statistically **predict** and **optimize** system performance using **probabilistic modeling**
   – Well-adapted for **highly resource-constrained robots**
Multi-Level Modeling Methodology

Target system (physical reality): information on controller, S&A, communication, morphology and environmental features

Bottom-Up Approach

Abstraction

Common metrics

Experimental time
Recipe: Target System & Metric(s)

1. Perform your basic design choices for the single robot:
   HW (e.g., robot morphology, S&A technology, etc.);
   SW (e.g., control architecture)

2. Define your system performance metric(s)
Multi-Level Modeling Methodology

Realistic – Module-based: intra-robot (e.g., S&A, transceiver) and environment (e.g., physics) details reproduced faithfully.

Target system (physical reality): information on controller, S&A, communication, morphology and environmental features

Approximations

Calibration
Recipe: Realistic simulation & FSM

3. Implement faithfully your design choices (HW & SW) in a realistic simulation (microscopic module-based).

4. Capture the control structure with a finite number of states of interest (semi-markovian properties must be fulfilled) and generate a corresponding FSM.
Multi-Level Modeling Methodology

**Target system (physical reality):** info on controller, S&A, communication, morphology and environmental features

**Realistic – Module-based:** intra-robot (e.g., S&A, transceiver) and environment (e.g., physics) details reproduced faithfully

**Microscopic – Agent-based:** multi-agent models, only relevant robot features captured, 1 agent = 1 robot

**Approximations**

**Calibration**

**Common metrics**

**Experimental time**

**Abstraction**
Recipe : Microscopic-AB Model

5. Approximate local interactions and intra-robot details and develop an agent-based model
Multi-Level Modeling Methodology

\[
\frac{dN_n(t)}{dt} = \sum_n W(n | n', t) N'_n(t) - \sum_n W(n' | n, t) N_n(t)
\]

- **Macroscopic**: rate equations, mean-field approach
- **Microscopic – Agent-based**: multi-agent models, only relevant robot features captured, 1 agent = 1 robot
- **Realistic – Module-based**: intra-robot (e.g., S&A, transceiver) and environment (e.g., physics) details reproduced faithfully
- **Target system** (physical reality): info on controller, S&A, communication, morphology and environmental features

**Approximations**

**Calibration**

**Experimental time**

**Abstraction**

**Common metrics**
Recipe : Macroscopic Model

6. Approximate micro-to-macro mapping (mean field): exploit AB-blueprint and build the macroscopic model

Robotic system (1 PFSM)
- Single representation for the whole swarm
- Aggregation of individual units’ PFSM into a single PFSM
- Rate equations (solved by ODE integration)

Environment (1 PFSM)

Type q
ODE models become inaccurate for small numbers of robots and capture only average trajectory.
A Generalized Case Study: Collective Identification and Destruction of Spots

[Mermoud et al., AAMAS, 2010]
Experimental setup

- Projector
- Camera
- Alice robots
- Augmented reality software package
- Tracking software package (SwisTrack)
- TCP/IP
good spots (e.g., healthy cells)  bad spots (e.g., cancer cells)

Light sensor

Alice robot
Robots can make mistakes!

Performance metric

\[ M(\alpha) = \frac{D_{bad}}{(D_{good})^{\alpha} + 1} \]

\( \alpha \) depends on the application!
In our case, \( \alpha = 2 \!

Solution: collective decision-making!

Aggregates of \( k \) robots are required to trigger the destruction of a spot.
Robot controller

**Explore arena**
*Random walk and braitenberg*

Detect border and $X < p_{\text{leave}}$

**Explore spot**
*Straight and u-turns*

Detect mate

$X < p_{\text{leave,agg}}$

**Aggregation**
*Stop*

Detect spot

Spot destroyed

Depends on robot’s belief!
Controller principle \((k = 2)\)
Proof-of-concept (real robots)
Proof-of-concept (real robots)

<table>
<thead>
<tr>
<th>Destruction rate</th>
<th>Without collaboration ($k = 1$)</th>
<th>With collaboration ($k = 2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bad spots</td>
<td>Good spots</td>
</tr>
<tr>
<td>Run 1</td>
<td>4.93</td>
<td>3.85</td>
</tr>
<tr>
<td>Run 2</td>
<td>5.28</td>
<td>2.68</td>
</tr>
<tr>
<td>Run 3</td>
<td>5.12</td>
<td>2.95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance</th>
<th>Without collaboration ($k = 1$)</th>
<th>With collaboration ($k = 2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run 1</td>
<td>$2.9 \cdot 10^{-2}$</td>
<td>2.8</td>
</tr>
<tr>
<td>Run 2</td>
<td>$7.0 \cdot 10^{-2}$</td>
<td>9.0</td>
</tr>
<tr>
<td>Run 3</td>
<td>$5.8 \cdot 10^{-2}$</td>
<td>1.76</td>
</tr>
</tbody>
</table>

- **Collaboration** brings two orders of magnitude improvement on performance metric over non collaborative controller!
- However, **overall destruction rate** is lowered.
- How to find the optimal tradeoff?

**Use model-based optimization!**
Multi-Level Modeling Methodology

Chemical Reaction Network (CRN) Framework

Macroscopic 1: rate equations (ODE), mean field approach, whole population

Macroscopic 2: stochastic simulations, Monte Carlo method

Bottom-Up Approach

Reactor (module-based):
finite representation of the robot (geometry, S&A) and environment (e.g., friction, gravity, inertia)

Common metrics

Abstraction

Experimental time
Building up CRNs (step 1)

**Robot’s states**

- Explore arena: Random walk and braitenberg
- Explore spot: Straight and u-turns
- Aggregation: Stop

**Transitions and Conditions**

- Detect border and $X < p_{\text{leave}}$
- Detect spot
- Detect mate
- $X < p_{\text{leave,agg}}$
- Spot destroyed
Building up CRNs (step 2)

Robots exploring arena: $X_o$

Robots exploring spot $i$: $X_i$

Robots aggregated in spot $i$: $X_i$
Robots exploring arena: $X_o$

Robots exploring spot $i$ with **correct** estimate: $X_{i,c}$

Robots exploring spot $i$ with **wrong** estimate: $X_{i,w}$

Robots aggregated in spot $i$: $X_i$

Reaction rates

$\ell^c_i + s^c_i$

$e^c_i$

$e^w_i$

$\ell^w_i + s^w_i$

$s^c_i$

$s^w_i$
Solving CRNs

- **Stochastic** approach (Monte Carlo method)
  - Outcome: *realizations of trajectory* of the system (discrete quantities, distributions)
  - *Exact* (Gillespie) or *approximate* (e.g., Tau-leaping) methods
  - Can become *computationally expensive!*

- **Mean-field** approach (ODEs integration)
  - Outcome: *average trajectory* of the system (continuous quantities, mean)
  - *ODE approximation* (large number of events with small effect)
  - Extremely computationally efficient!

\[ \dot{y} = S \cdot p(y) \text{ with } S = (s_{ij}) \text{ the matrix of stoechiometries} \]
ODEs can be very inaccurate for some parameter sets!
Take-home messages

- Control and design of distributed robotic systems requires **efficient model-based approaches**.
- Bottom-up multi-level modeling methodology yields both **faithful** and **computationally inexpensive** models.
- It proved very efficient in studying a **large variety of case studies**, including aggregation, foraging, self-assembly, flocking in groups of small robots.

- Stochastic methods can be **exact**. ODE approximation may hold for only a subset of the parameter space.
Thank you for your attention!

Any questions?