Optimized Navigation Behaviour of Multiple Robots based on Motor-Schema

Professor:
Alcherio Martinoli
Distributed Intelligent Systems and Algorithms Laboratory

Assistant:
Zeynab Talebpour
Distributed Intelligent Systems and Algorithms Laboratory

Students:
Ondine Chanon
Computational Science and Engineering

Nicolas Hubacher
Computer Science

Stefano Savarè
Computational Science and Engineering
Content

- Introduction, goals and motivation
- Motor-schemas based navigation behaviour
- Particle Swarm Optimization (PSO)
- Results
- Conclusions
Introduction, goals and motivation

• Set-up
  - 1 goal
  - 4 robots
  - some obstacles

• Two-step project
  - basic navigation control, based of 4 motor-schemas
  - PSO implementation

• Aim
  - find the good balance between the objectives to get the best possible behaviour
Motor-schematas

General concept

• Each motor-schema generates a vector field
• The global vector field is the sum of all the computed individual vector fields
• Difficulty: presence of local minima

Sphere of influence

• Most of the implemented motor-schematas use this concept
• Example equation
  \[
  0, \quad \text{if } d < R \\
  \frac{d-R}{S-R}, \quad \text{if } R < d < S \\
  1, \quad \text{if } d > S
  \]
Motor-schemas

- Move to goal
- Keep formation
  - unit centered
- Robot avoidance
  - local communication
- Obstacle avoidance
  - supervisor
  - distance sensors
- Random walk
  - noise generation frequency
  - fading

Expected

Goal

Position expected
Actual position

Line
Column
Wedge
Diamond
Particle Swarm Optimization

• 15 parameters to optimize:
  - Weights for each motor-schema
  - Minimum and maximum thresholds
  - Parameters for random walk

• Manual setting

• Computationally expensive problem

• ‘Particle Space’: one dimension per parameter
  - normalized dimension ranges

• Problems with Webots ®
  → We have only optimized the motor-schema weights
PSO Formula

\[ v^{(i)} = w \cdot v^{(i-1)} + c_n \cdot r\text{nd}_n \cdot (b_n - x^{(i-1)}) + c_p \cdot r\text{nd}_p \cdot (b_p - x^{(i-1)}) \]

<table>
<thead>
<tr>
<th>( w )</th>
<th>Inertia</th>
<th>( v^{(i-1)} )</th>
<th>Previous Iteration’s Velocity</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_n )</td>
<td>Neighbourhood Weight</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c_p )</td>
<td>Personal Best Weight</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( r\text{nd}_n, r\text{nd}_p )</td>
<td>Random numbers</td>
<td>[0, 1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( b_n )</td>
<td>Best Solution so Far in Neighbourhood</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( b_p )</td>
<td>Personal Best Solution so Far</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x^{(i-1)} )</td>
<td>Previous Iteration’s Position</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PSO Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensionality</td>
<td>4</td>
</tr>
<tr>
<td>Flock Size</td>
<td>5</td>
</tr>
<tr>
<td>Neighbourhood Size</td>
<td>2</td>
</tr>
<tr>
<td>Neighbourhood Type</td>
<td>‘Ring’</td>
</tr>
<tr>
<td>Ranges</td>
<td>[0, 10]</td>
</tr>
<tr>
<td>Maximum Velocity</td>
<td>1</td>
</tr>
</tbody>
</table>
Optimally Computing Budget Allocation (OCBA)

- Randomness: Problem of basic PSO
- OCBA
  - Iteration Budget: Number of Evaluations to be performed
  - New positions are evaluated $n_0$ times
  - Promising positions with high variance are re-evaluated
  → Results are more reliable
  → More efficient than other re-evaluating PSO methods

<table>
<thead>
<tr>
<th>Iteration Budget</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_0$</td>
<td>2</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>4</td>
</tr>
</tbody>
</table>
Particle Swarm Optimization

\[ F = \alpha_v \cdot V \cdot \left(1 - \sqrt{\alpha_\Delta \cdot \Delta V}\right) \cdot \frac{1}{D_o + \alpha_o} \cdot \frac{1}{D_f} + \alpha_g \cdot G \]

- \( \alpha_v, \alpha_\Delta, \alpha_o, \alpha_g \): appropriate weights
- \( V \): average velocity of the unit center
- \( \Delta V \): average difference of wheel velocity
- \( D_o \): mean distance of the robot to its nearest obstacle
- \( D_f \): mean distance to the expected position in the formation
- \( G \): decreases with the final distance to the goal
Results

3 different worlds considered for PSO

1. Random
2. Wall of obstacles
3. Difficult configuration
### Results

**Parameters manually tuned**

<table>
<thead>
<tr>
<th>Motor-schema</th>
<th>Weight</th>
<th>Min threshold</th>
<th>Max threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>move_to_goal</td>
<td>0.07</td>
<td>0.10</td>
<td>0.50</td>
</tr>
<tr>
<td>keep_formation</td>
<td>0.35</td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>avoid_robot</td>
<td>0.07</td>
<td>0.05</td>
<td>0.10</td>
</tr>
<tr>
<td>avoid_obstacle</td>
<td>0.35</td>
<td>60</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Motor-schema</th>
<th>Weight</th>
<th>Noise gen. frequency</th>
<th>Fading</th>
</tr>
</thead>
<tbody>
<tr>
<td>random_walk</td>
<td>0.33</td>
<td>10</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Weights found by PSO**

<table>
<thead>
<tr>
<th>Motor-schema</th>
<th>World 1</th>
<th>World 2</th>
<th>World 3</th>
<th>World 1 sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>move_to_goal</td>
<td>0.21</td>
<td>0.28</td>
<td>0.29</td>
<td>0.18</td>
</tr>
<tr>
<td>keep_formation</td>
<td>0.33</td>
<td>0.30</td>
<td>0.15</td>
<td>0.10</td>
</tr>
<tr>
<td>avoid_robot</td>
<td>0.14</td>
<td>0.11</td>
<td>0.19</td>
<td>0.30</td>
</tr>
<tr>
<td>avoid_obstacle</td>
<td>0.27</td>
<td>0.26</td>
<td>0.32</td>
<td>0.15</td>
</tr>
<tr>
<td>random_walk</td>
<td>0.04</td>
<td>0.06</td>
<td>0.05</td>
<td>0.26</td>
</tr>
</tbody>
</table>
Results

- Wedge formation; obstacles known through the supervisor
- Better convergence: world with a difficult configuration
- More iterations and more particles would be needed in the other 2 worlds to perform a better analysis
Results

Different formations
Results

PSO optimization
Conclusions

- Hand tuned parameters already provide a good controller
- PSO – OCBA is a powerful tool: better results but higher simulation time
- Unit-centered formation control is more robust to different formations than the leader referenced formation control
- Formation does not have a big influence on PSO, but worlds do.
- Noisy measures lead to suboptimal results: sensor measures and local communications
- Further work:
  - odometry based positioning
  - computation of fitness metrics in the robot’s controllers
  - combine PSO worlds
Bibliography

Arkin, Ronald C.  

Balch, Tucker and Arkin.  

Di Mario, Navarro and Martinoli.  
Analysis of fitness noise in particle swarm optimization: From robotic learning to benchmark functions. Evolutionaly Computation (CEC).  
*IEEE Congress on. IEEE, 2014.*

Eberhart, Kennedy.  
*IEEE 1995.*

Di Mario, Navarro and Martinoli.  
A Distributed Noise-Resistant Particle Swarm Optimization Algorithm for High-Dimensional Multi-Robot Learning.  

Floreano and Mondada.  

Pugh, Zang and Martinoli.  
Particle swarm optimization for unsupervised robotic learning.  
*Swarm Intelligence Symposium. No. SWIS-CONF-2005-004. 2005*
The end... almost!

Thank you for your attention
Questions?