Exploring Algorithmic Options for the Efficient Design and Reconfiguration of Reactive Robot Swarms

Todd Wareham
Department of Computer Science
Memorial University of Newfoundland

December 5, 2015
Introduction

- Many methods proposed to design robot swarms (Crespi et al, 2008; Brambilla et al, 2013; Doursat et al, 2013), e.g.,
  - temporal-logic decomposition (Winfield et al, 2005a)
  - dataflow diagram decomposition (Winfield et al, 2005b)
  - interaction-graph decomposition (Wiegand et al, 2006)
  - evolutionary algorithms (Sperati et al, 2011)
- No method to date is both general and efficient.

**How difficult is swarm design in general?**
**What restrictions do (and do not) make swarm design easy?**
Organization of this Talk

1. Formalizing Swarms
2. Formalizing Swarm Design
3. Computational Complexity Analysis: The Reader’s Digest Version
4. Complexity of Swarm Design
5. Conclusions and Future Work
Formalizing Swarms: Swarm Entity Architecture

- **Modifications:**

  **Reconfiguration:** Modify up to $c$ layers and layer-linkages relative to layer library $\mathcal{M}$
Formalizing Swarms: Overall Swarm Architecture

- Three policies: individual entity movement $+$ entity communication $+$ movement conflict resolution.
- Restrictions (this talk):
  - Synchronized entity movement.
  - No inter-entity communication.
  - No movement conflict allowed.
- Modifications:
  **Selection**: Select $|S|$ entities from entity library $A$
<table>
<thead>
<tr>
<th></th>
<th>Swarm Members / Positions Given</th>
<th>Swarm Members / Positions Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Swarm Member Reconfiguration</td>
<td>Given Swarm Morphogenesis (GRSM)</td>
<td>Selected Swarm Morphogenesis (SRSM)</td>
</tr>
<tr>
<td>Swarm Member Reconfiguration Allowed</td>
<td>Given Swarm Morphogenesis with Reconfiguration (GRSM-REC)</td>
<td>Selected Swarm Morphogenesis with Reconfiguration (SRSM-REC)</td>
</tr>
</tbody>
</table>
## Computational Complexity Analysis
### The Reader’s Digest Version

<table>
<thead>
<tr>
<th></th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>classical</td>
<td><strong>pt-tractable</strong> $(n^c)$</td>
<td><strong>pt-intractable</strong> $(\text{NP-hard})$</td>
</tr>
<tr>
<td>(unrestricted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>parameterized</td>
<td><strong>fp-tractable</strong> $(f(p) \times n^c)$</td>
<td><strong>fp-intractable</strong> $(\text{W-hard})$</td>
</tr>
<tr>
<td>(restriction $p$)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Complexity of Swarm Design

Main results:

- SRSM, GRSM-REC, and SRSM-REC are \textit{pt-intractable}.
- Complexity of GRSM is not known but evidence suggests it may be \textit{pt-time intractable}.

Implications:

- Swarm design problems are \textit{intractable} in general $\Rightarrow$ these problems cannot have efficient solution-guaranteed deterministic \textit{or} probabilistic algorithms, \textit{e.g.}, evolutionary algorithms.
- Need to restrict these problems if we are to get \textit{tractability}.

\ldots \textit{What restrictions (if any) yield \textit{tractability}?} \ldots
### Complexity of Swarm Design (Cont’d)

<table>
<thead>
<tr>
<th>Param.</th>
<th>Definition</th>
<th>Appl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>Max (final) # layers per swarm member</td>
<td>All</td>
</tr>
<tr>
<td>$E$</td>
<td># distinguishable world-square types</td>
<td>All</td>
</tr>
<tr>
<td>$f$</td>
<td>Max length of layer trigger-formula</td>
<td>All</td>
</tr>
<tr>
<td>$r$</td>
<td>Swarm member perceptual radius</td>
<td>All</td>
</tr>
<tr>
<td>$S$</td>
<td># entities in swarm</td>
<td>All</td>
</tr>
<tr>
<td>$h$</td>
<td># entity-types in swarm (heterogeneity)</td>
<td>All</td>
</tr>
<tr>
<td>$a$</td>
<td>Size of initial swarm positioning area</td>
<td>All</td>
</tr>
<tr>
<td>$A$</td>
<td># entities in entity library</td>
<td>SSN*</td>
</tr>
<tr>
<td>$M$</td>
<td># layers in layer library</td>
<td>*-REC</td>
</tr>
<tr>
<td>$c$</td>
<td>Max # swarm entity modifications</td>
<td>*-REC</td>
</tr>
</tbody>
</table>
• What restrictions don’t make swarm design easy?
  • (Almost) Everything restricted individually (to constants!)
  • Many, many combinations of restrictions as well . . .

• What restrictions do make swarm design easy?
  • Several combinations of restrictions that restrict input size are fp-tractable (whoopdeedoo . . .).
  • \( \langle |E|, f, |a| \rangle \) / \( \langle |E|, r, |a| \rangle \)-SRSM, -GRSM-REC, and SRSM-REC are fp-tractable.

• Implications:
  • Many restrictions on swarm entity or overall swarm architecture do not make swarm design efficient.
  • What does seem to matter is restrictions on the sensory / perceptual complexity of the swarm entities ⇒ ignorance is (computational) bliss! (Wareham et al, 2011).
Conclusions and Future Work

• Swarm design is intractable in general for the simplest types of worlds, tasks, and entity / overall architectures; however, there are plausible restrictions that may allow instances of interest to be solved exactly.

• Future work:
  • Determine computational complexity of GRSM.
  • Extend parameterized analysis to other aspects, e.g., complexity of environment.
  • Analyze swarm design relative to more realistic types of worlds, tasks, and architectures.
  • Investigate related problems, e.g., random start-position morphogenesis.