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

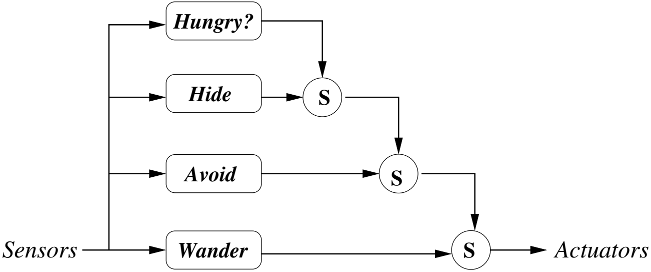
- Many methods proposed to design robot swarms (Crespi et al, 2008; Brambilla et al, 2013; Doursat et al, 2013), *e.g.*,
  - temporal-logic decomposition (Winfield et al, 2005a)
  - dataflow diagram decomposition (Winfield et al, 2005b)
  - interaction-graph decomposition (Wiegand et al, 2006)
  - evolutionary 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 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

Formalizing Swarm Design

	Swarm Members / Positions Given	Swarm Members / Positions Selected
No Swarm Member Reconfiguration	Given Swarm Morphogenesis (GRSM)	Selected Swarm Morphogenesis (SRSM)
Swarm Member Reconfiguration Allowed	Given Swarm Morphogenesis with Reconfiguration (GRSM-REC)	Selected Swarm Morphogenesis with Reconfiguration (SRSM-REC)

Computational Complexity Analysis
The *Reader's Digest* Version

	good	bad
classical (unrestricted)	pt-tractable (n^c)	pt-intractable (NP-hard)
parameterized (restriction p)	fp-tractable ($f(p) \times n^c$)	fp-intractable (W-hard)

Complexity of Swarm Design

- Main results:
  - SRSM, GRSM-REC, and SRSM-REC are pt-intractable.
  - Complexity of GRSM is not known but evidence suggests it may be pt-time intractable.
- Implications:
  - Swarm design problems are intractable in general $\Rightarrow$ these problems cannot have efficient solution-guaranteed deterministic *or* probabilistic algorithms, *e.g.*, evolutionary algorithms.
  - Need to restrict these problems if we are to get tractability.

... What restrictions (if any) yield tractability? ...

Complexity of Swarm Design (Cont'd)

Param.	Definition	Appl.
$\|L\|$	Max (final) # layers per swarm member	All
$\|E\|$	# distinguishable world-square types	All
f	Max length of layer trigger-formula	All
r	Swarm member perceptual radius	All
$\|S\|$	# entities in swarm	All
h	# entity-types in swarm (heterogeneity)	All
$\|a\|$	Size of initial swarm positioning area	All
$\|A\|$	# entities in entity library	SSN*
$\|M\|$	# layers in layer library	*-REC
c	Max # swarm entity modifications	*-REC

Complexity of Swarm Design (Cont'd)

- 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 $\Rightarrow$ 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.