Educating Genghi: A Complexity Perspective on Designing Reactive Swarms

Todd Wareham

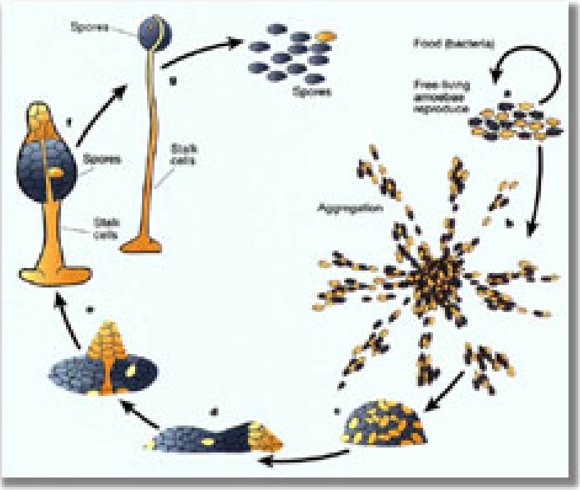
Department of Computer Science
Memorial University of Newfoundland

April 12, 2013

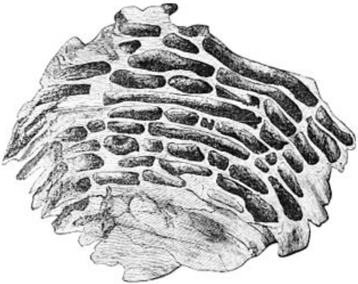
Introduction: Why swarms?

- Swarm $=$ group of active, mobile entities.
- Characteristics of a swarm:
    - Large number of entities (100+)
    - No centralized control or synchronization
    - Composed of few homogeneous groups of entities
    - Entities are simple
    - Entities are autonomous
    - Entities sense and communicate locally
- Swarms are robust (wrt individual failure or disturbances in environment), flexible, and scalable.

EXAMPLE: Slime Mold Aggregation



EXAMPLE: Termite Nest Construction



EXAMPLE: Robot swarm Morphogenesis

Introduction: Why swarms? (Cont'd)

- Many methodologies proposed to design robot swarms (Crespi et al, 2008; Brambilla et al, 2012), *e.g.*,
  - temporal-logic decomposition (Winfield et al, 2005a)
  - dataflow diagram decomposition (Winfield et al, 2005b)
  - interaction-graph decomposition (Wiegand et al, 2006)
  - evolutionary algorithm (Sperati et al, 2011)
- No method to date is both general and efficient.

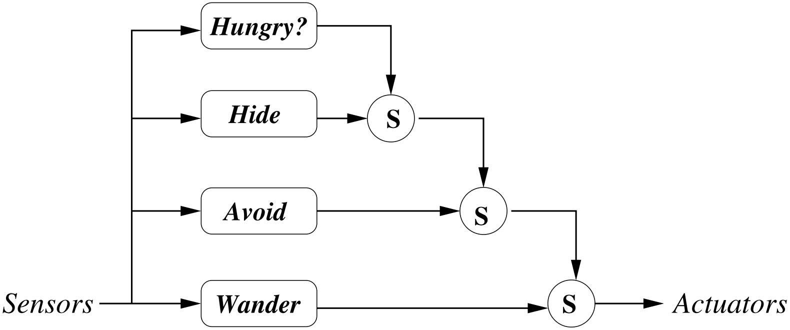
HOW DIFFICULT IS SWARM DESIGN?
WHAT DOES (AND DOESN'T) MAKE
SWARM DESIGN EASY?

Organization of this Talk

1. Defining Swarms
2. Defining Swarm Design
3. Computational Complexity Analysis:
   The *Reader's Digest* Version
4. Complexity of Swarm Design
5. Conclusions and Future Work

Defining Swarms: Swarm Entity Architecture

- Use reactive subsumption architectures (Brooks, 1986).
- Architecture $=$ sensors $+$ layers $+$ total order on layers $+$ layer subsumption interactions (inhibit/override)



Defining Swarms: Swarm Entity Architecture (Cont'd)

- Restrictions (this talk):
  - Sensors as object-existence in perceptual radius
  - One action per layer, triggered by Boolean sensor-formula
  - Layer either outputs action *OR* subsumes, not both
  - Restriction on length of Boolean sensor-formulas
- Modifications:

Reconfiguration: Modify up to c layers and layer-linkages (relative to provided layer library M)

Defining 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: Add / delete up to c entities (relative to provided entity library A)

Defining Swarm Design: The General Picture

SWARM NAVIGATION WITH X *Input*: World W, swarm S, start and finish points s and d in W, integer c. *Output*: A swarm S' derived by at most c modifications of type X from S that can move conflict-free from s to d, if such an S' exists, and special symbol $\perp$ otherwise.

- Focus on:
  - World as finite 2-D map (obstacle/freespace).
  - **area** $=$ region of size $|S|$ in world; **position** $=$ assignment of members of S to squares in an area.
  - Task as navigation between specified start and destination areas / positions in world (no restrictions on path).

Defining Swarm Design: The Specific Picture

- GIVEN SWARM NAVIGATION (GSN)
  Given W, S, start-position s and destination-area d, can S get from s to d?
- SELECTED SWARM NAVIGATION(SSN)
  Given W, $|S|$, A, and areas s and d, derive S and position of S in s such that S can get from s to d.
- GIVEN SWARM NAVIGATION WITH REC. (GSN-REC)
  Given W, S, M, start-position s and destination-area d, derive S' from S wrt M such that S' can get from s to d.
- SELECTED SWARM NAVIGATION WITH REC. (GSN-REC)
  Given W, $|S|$, A, M, and areas s and d, derive S wrt A and M and position of S in s such that S can get from s to d.

Computational Complexity Analysis

- A problem Π is **poly-time solvable** if Π is solvable in time n^c for input size n and constant c.
- In Computer and Cognitive Science, efficient solvability $=$ poly-time solvability (see van Rooij (2008) and references).
- Basic questions about a computational problem C:
  1. Is C hard, *i.e.*, is C poly-time solvable?
  2. If so, what can we restrict to make C easy, *i.e.*, (effectively) poly-time solvable?
- Use classical complexity to show problem is not poly-time solvable, *i.e.*, NP-hardness (Garey and Johnson, 1979).

. . . What about Question (2)??? . . .

Computational Complexity Analysis (Cont'd)

- State problem restrictions in terms of the values of problem aspects, *e.g.*, # entities in swarm; a **parameter** is the set of one or more restricted aspects.
- A problem Π is **fixed-parameter (fp-)tractable** relative to a parameter p if Π is solvable in time $f(p) \times n^c$ for some function f, input size n, and constant c.

  $\Rightarrow$ Π is effectively poly-time solvable for small values of p !

  $\Rightarrow$ The aspects in p are responsible for the poly-time unsolvability of Π, in that large values of p result in impractical running times, *i.e.*, p make Π hard!!

  $\Rightarrow$ To get poly-time solvability, *i.e.*, make Π easy, limit values of aspects in p !!!
- Use parameterized complexity to show fp-intractability, *i.e.*, W-hardness (Downey and Fellows, 1999).

Computational Complexity Analysis (Cont'd)
The *Reader's Digest* Version

	good	bad
classical	poly-time solvable	NP-hard
parameterized	fp-tractable	fp-intractable

Complexity of Swarm Design: A Quick Reminder

- GIVEN SWARM NAVIGATION (GSN)
  Can a given positioned swarm get from s to d?
- SELECTED SWARM NAVIGATION(SSN)
  Can a selected swarm be positioned to get from s to d?
- GIVEN SWARM NAVIGATION WITH REC. (GSN-REC)
  Can a given positioned swarm be reconfigured
  to get from s to d?
- SELECTED SWARM NAVIGATION WITH REC. (GSN-REC)
  Can a selected swarm be reconfigured and positioned
  to get from s to d?

Complexity of Swarm Design

- Main results:
  - GSN is poly-time solvable! But . . .
  - SSN, GSN-REC, and SSN-REC are poly-time intractable.
- Implications:
  - Swarm design problems are intractable in general (as GSN is not so much swarm design as swarm verification).
  - 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.
$\|S\|$	# entities in swarm	All
h	# entity-types in swarm (heterogeneity)	All
$\|L\|$	Max (final) # layers per swarm member	All
$\|E\|$	# distinguishable world-square types	All
f	Max length of layer trigger-formula	All
$\|M\|$	# layers in layer library	*-REC
c	Max # modifications	*-REC
$\|A\|$	# architectures in architecture library	SSN*

Complexity of Swarm Design (Cont'd)

	$\lvert S\rvert$	$\lvert L\rvert$	h	$\lvert A\rvert$	$\lvert M\rvert$	c	$\lvert E\rvert$	f
SSN	p	3	p	–	X	X	–	p
	–	3	2	2	X	X	2	–
	c	–	–	–	X	X	–	–
GSN-REC	1	p	1	X	–	p	–	1
	p	3	p	X	–	p	–	p
	–	3	2	X	2	p	2	–
SSN-REC	1	p	1	1	–	p	–	1
	p	3	p	–	–	p	–	p
	–	3	2	1	2	p	2	–

Complexity of Swarm Design (Cont'd)

- What *doesn't* make swarm design hard:
  - (Almost) Everything restricted individually (to constants!)
  - Many, many combinations of restrictions as well ...
- What makes swarm design hard:
  - Several combinations of restrictions that restrict input size are fp-tractable (whoopdeedoo ...).
  - $\langle |E|, f \rangle$-SSN, -GSN-REC, and SSN-REC are fp-tractable.
- Implications:
  - Many restrictions on swarm entity or overall swarm architecture do not matter, cf. natural swarms.
  - What is important is restrictions on the sensory / perceptual complexity of the swarm entities $\Rightarrow$ ignorance is (computational) bliss! (Haselager, van Dijk, and van Rooij, 2008; 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:
  - Extend parameterized analysis to other aspects, *e.g.*, perceptual radius.
  - Analyze swarm design relative to other types of worlds, tasks, and architectures.
  - Investigate approximability of swarm design.
  - Investigate related problems, *e.g.*, reactive morphogenesis.