Swarm Algorithms
Simulation and Generation

Michael A. Kovacina

Masters Thesis Defense

September 02, 2005

Case Western Reserve University

Advisor: Dr. Michael Branicky

Overview

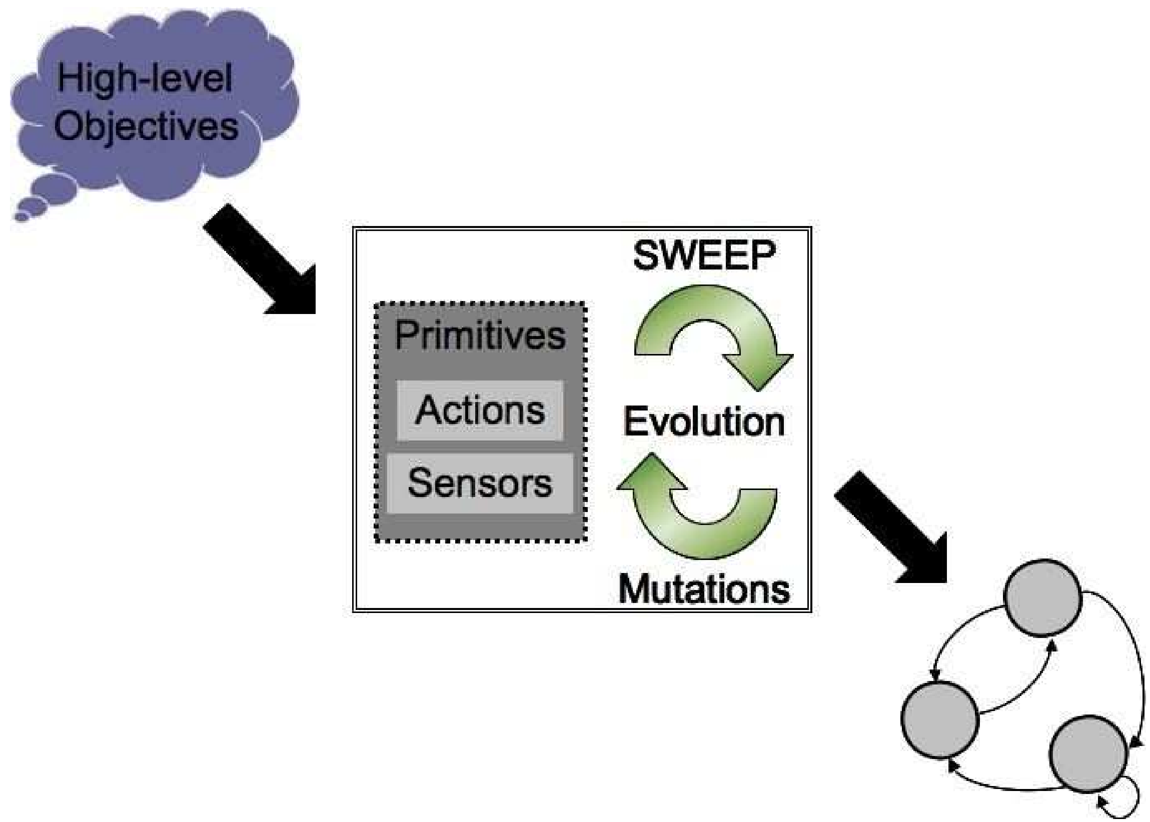
- Rationale
- Swarm Intelligence Overview
- Swarm Simulation Software
- Evolutionary Computing for Swarms
- Results
- Conclusions

Motivations

- Interactions and complexity
- Complexity can grow quickly
- Swarm algorithm development requires simulation
- Trial-and-error programming
- A complete toolchain is needed to streamline swarm algorithm creation
  - Simulator
  - Algorithm generator
  - Identify/classify emergence

Vision

Demonstrate a method for generating swarm behaviors using evolutionary computing.



Swarm Intelligence

Examples of *Swarm Intelligence* found in nature

- Flocking birds
  - A bird flying disrupts airflow
  - Disrupted air flow reduces drag for following birds
  - Reduced drag results in easier flying
  - Distance traveled by the flock is maximized

Swarm Intelligence

Examples of *Swarm Intelligence* found in nature

- Foraging ants
  - An ant leaves a pheromone trail upon finding food
  - Other ants follow and reinforce the trail
  - Each ant is able to find food for the nest
  - Trail laying finds the closest food source

Swarm Intelligence

Examples of *Swarm Intelligence* found in nature

- Termite nests
  - A termite deposits a pheromone-tagged mud ball
  - Local pheromones affect mud ball placement
  - A secure nest for the termite is established
  - A temperature regulated nest emerges

Swarm Intelligence

Why has evolution produced swarming in so many different contexts?

- Simultaneously benefits the individual and the whole
- Individuals benefit from the efforts of others
- The survivability of the swarm increases
- Simple rules and behaviors, decentralized
- Replication relatively easy

Swarm Intelligence

Swarm intelligence as defined for this work

> a group of agents whose collective interactions magnify the effects of individual agent behaviors, resulting in the manifestation of swarm level behaviors beyond the capability of a small subgroup of agents

Other properties required for emergent behavior

- Large numbers of agent interactions
- Ability to modify the environment, stigmergy
- Randomness

Swarm Algorithm Development

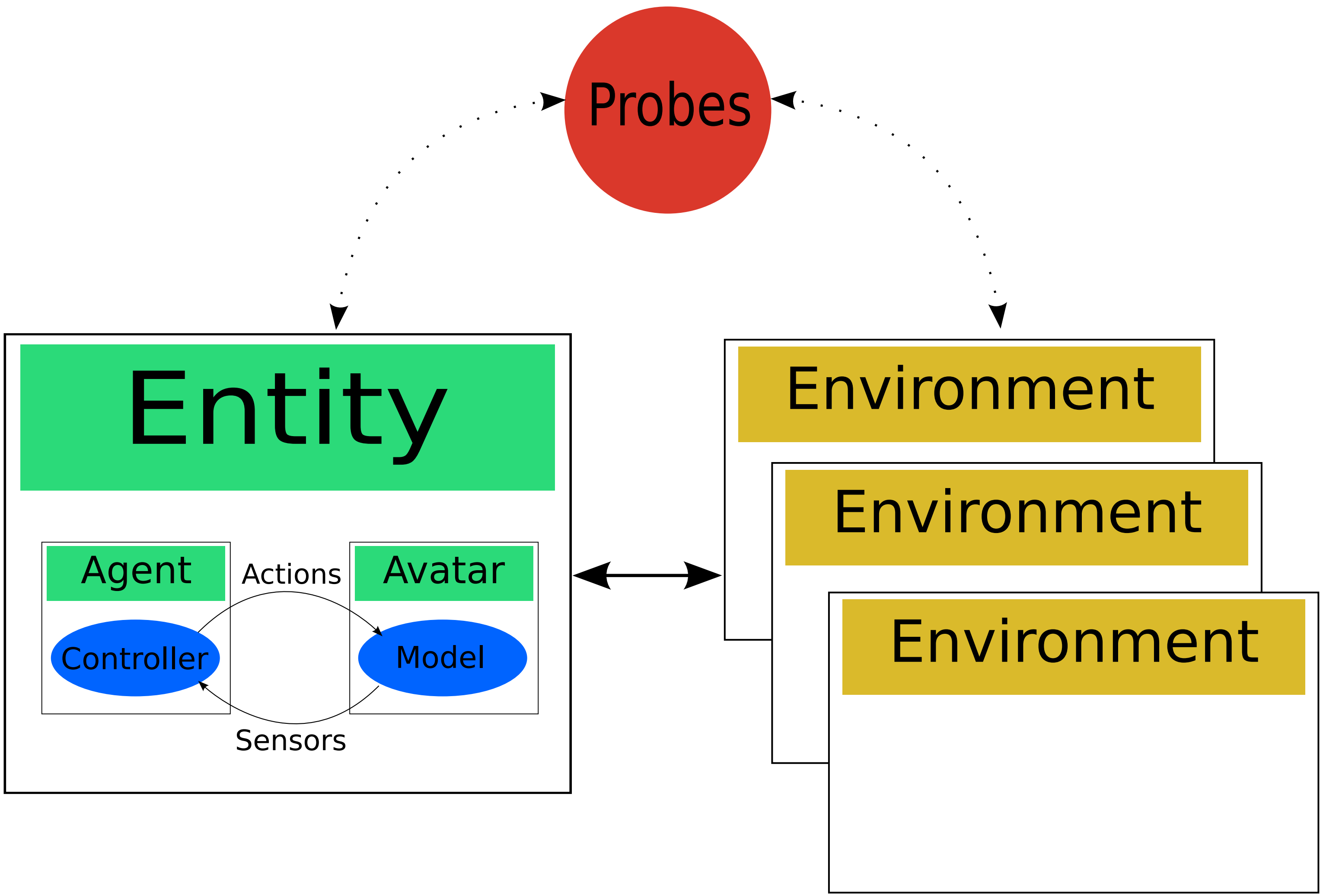
- No methods for direct analysis of swarm algorithms
- Swarm algorithms evaluated through simulation

Thus, a flexible swarm simulation platform is required.

- Multiple agent and swarm types
- Support for various environment types
- Access to all simulation data
- Easy to use
- Portable

SWEEP

SWEEP- **SW**arm **E**xperimentation and **E**valuation **P**latform



SWEEP - Simulation

- XML simulation specification file
- Responsible for constructing the simulation
- Handles update scheduling

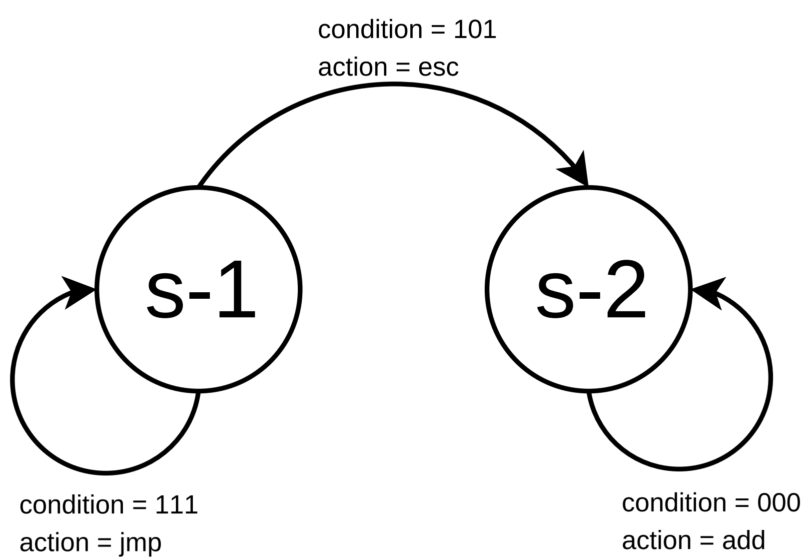
1. Environment
2. Entity::Agent
3. Entity::Avatar
4. Probes

```
<simulation>
  <main/>
  <agent/>
  <controller/>
  <model/>
  <environment/>
  <probes/>
</simulation>
```

SWEEP - `Entity::Agent`

The "mind" of the `Entity`

- `State`: collection of variables that define the agent
- `Controller`: defines the governing logic of the agent
- The default `Controller` is a finite state machine



SWEEP - `Entity::Avatar`

The “body” of the `Entity`

- Conduit between `Agents` and `Environments`
- Separates modeling and algorithm development
- e.g., a UAV

- `Model`: defines characteristics
  e.g., minimum turning radius, maximum thrust
- `Sensor`: defines environmental information available
  e.g., chemical sensor, GPS
- `Action`: defines behavioral abilities
  e.g., plan-path-to, return-to-base

SWEEP - `Environment`

The `Environment` has three core functionalities:

1. Defining fundamental laws that `Avatars` must respect
   e.g., gravity, $F = ma$, bandwidth limits

2. Presenting an information abstraction layer
   e.g., neighborhood on a grid vs. a graph

3. Facilitating direct and indirect communication
   e.g., simulating wireless, pheromone gradients

SWEEP - `Probe`

`Probes` provide the ability to

- Extract information from a running simulation
- Inject information into a running simulation

The current `Probe` implementation uses `Connectors`.

- `Connectors` are data conduits between components
- `Probes` “tap” `Connectors`
- `Connectors` provide access to information injection/extraction

Example `Probe` usage: diagnostic interface

SWEEP Applications

This thesis:

- Dispersion
- Task assignment, CAST Auction
- $\boxed{\text{Chemical cloud tracking}}$

Other works:

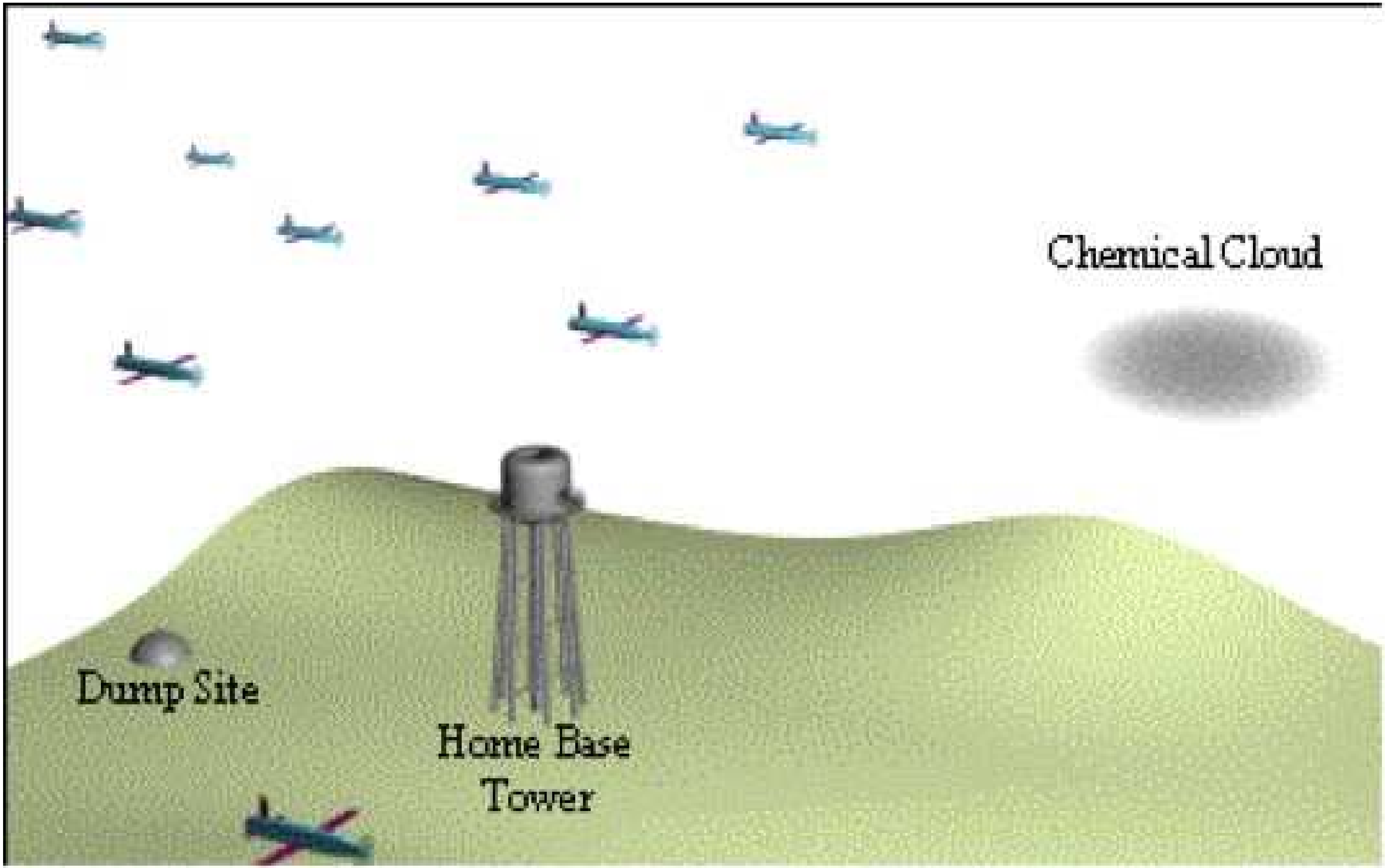
- Swarm reasoning for the four-color mapping problem
- Mars exploration using “tumbleweeds”
- Extending swarm programming with aspect-oriented programming

UAV Chemical Cloud Tracking

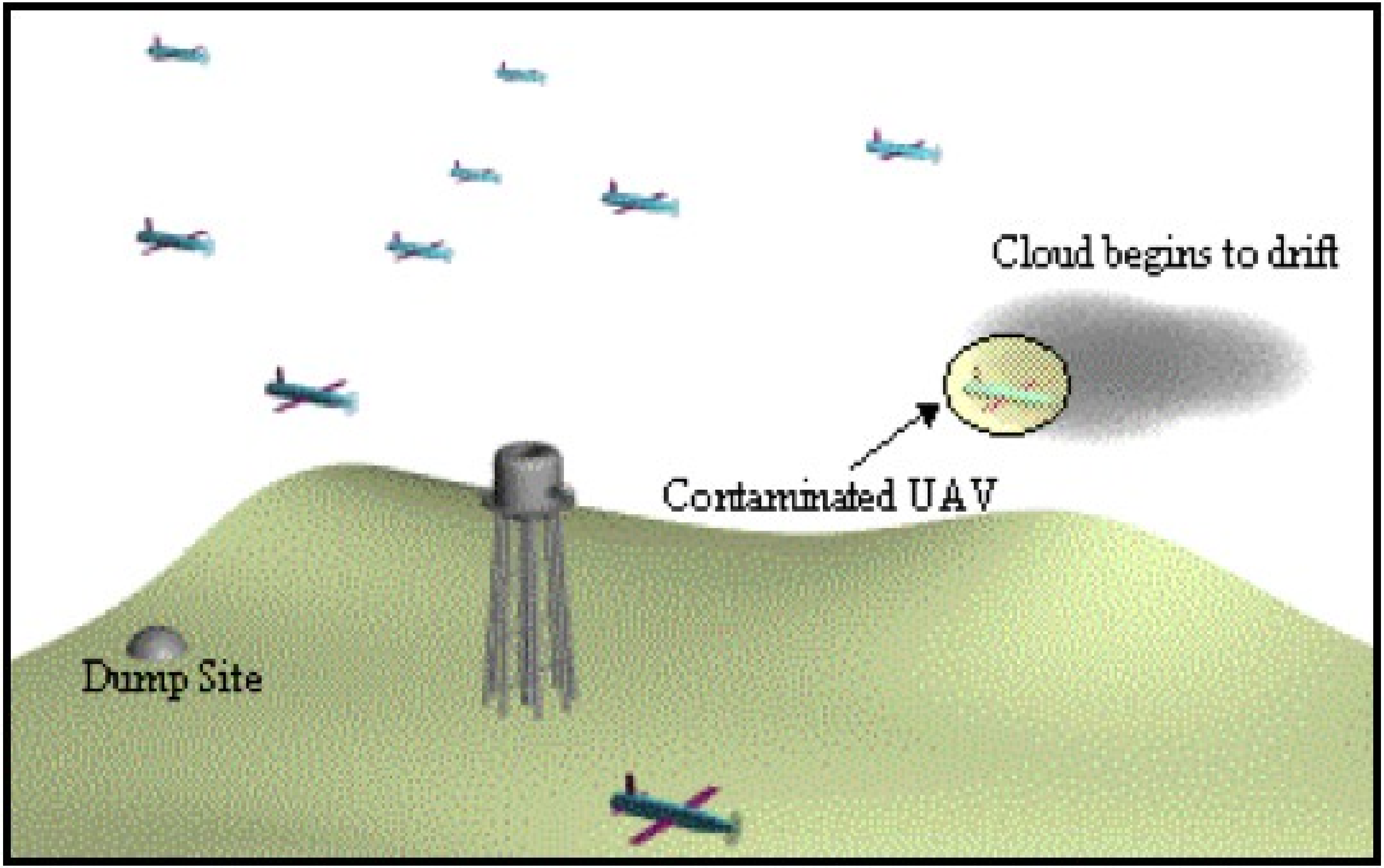
Highlights

- Develop decentralized algorithms for small collections of UAVs
- Constrained vehicles
  - Limited communication
  - Limited flight capabilities
  - Binary sensors
- Explore potential emergent behavior of small collections of agents

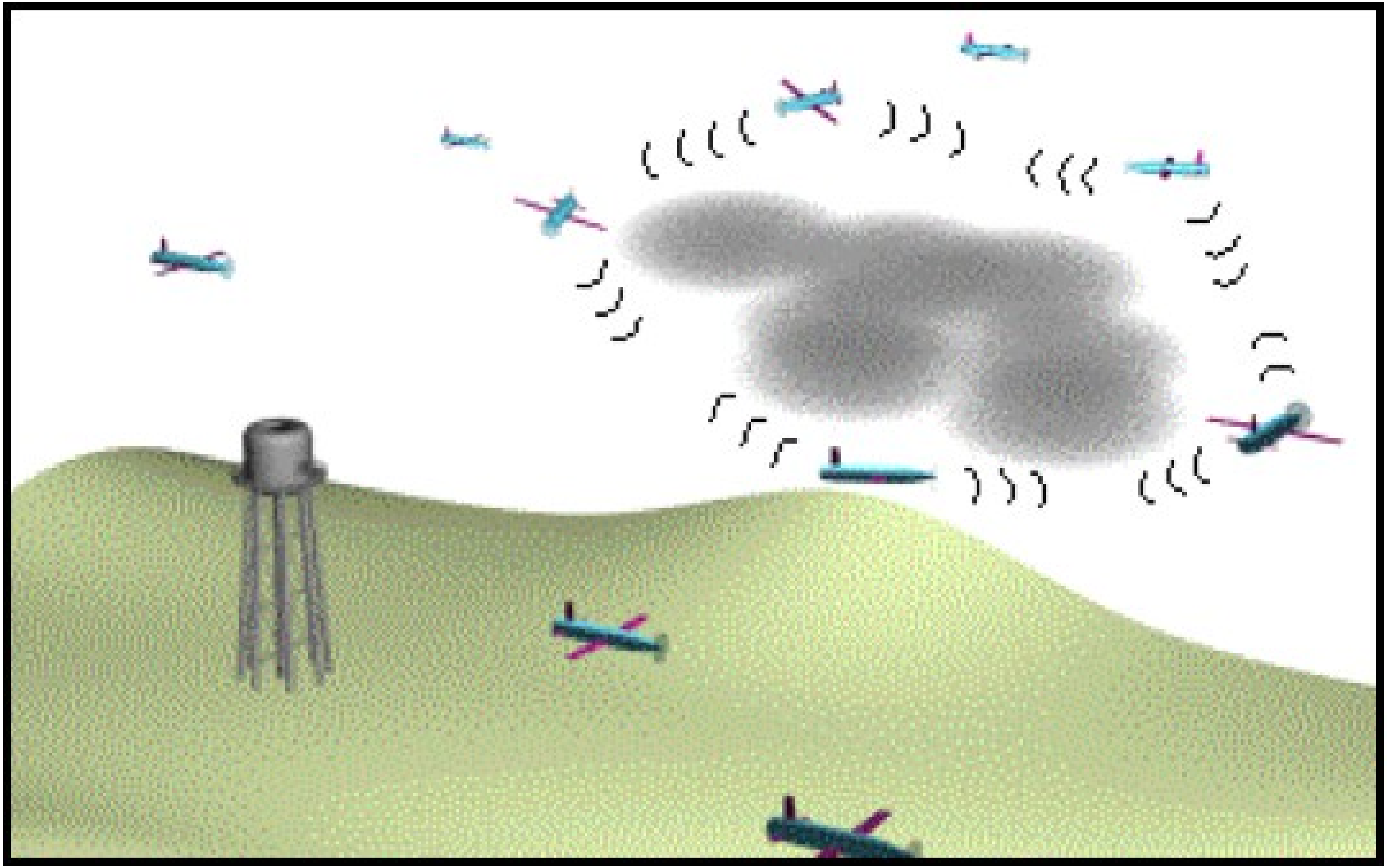
UAV Chemical Cloud Tracking



UAV Chemical Cloud Tracking



UAV Chemical Cloud Tracking



UAV Chemical Cloud Tracking

UAV Characteristics

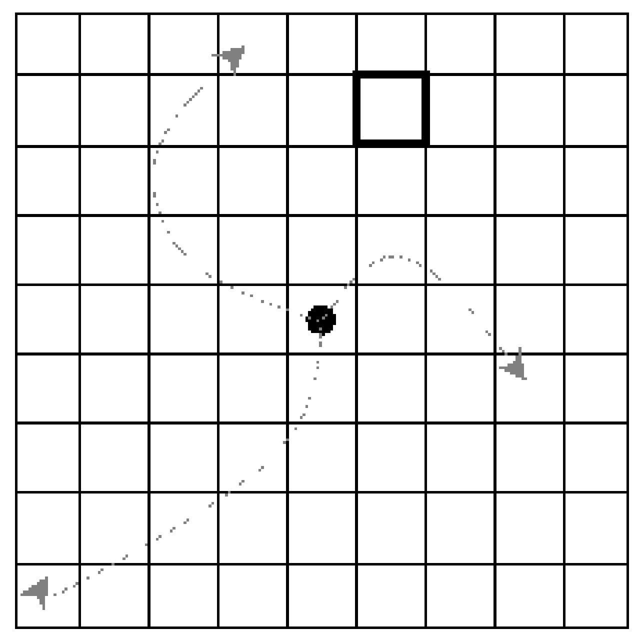
- UAV flight model
  - Point mass
  - $$\begin{cases} \dot{x} = v_C \cos(\theta) \\ \dot{y} = v_C \sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$
  - Fixed turning radius
  - Constant speed
  - Geometric path planner
- 30 minute power supply
- Binary chemical sensor

UAV Chemical Cloud Tracking

Data Collection

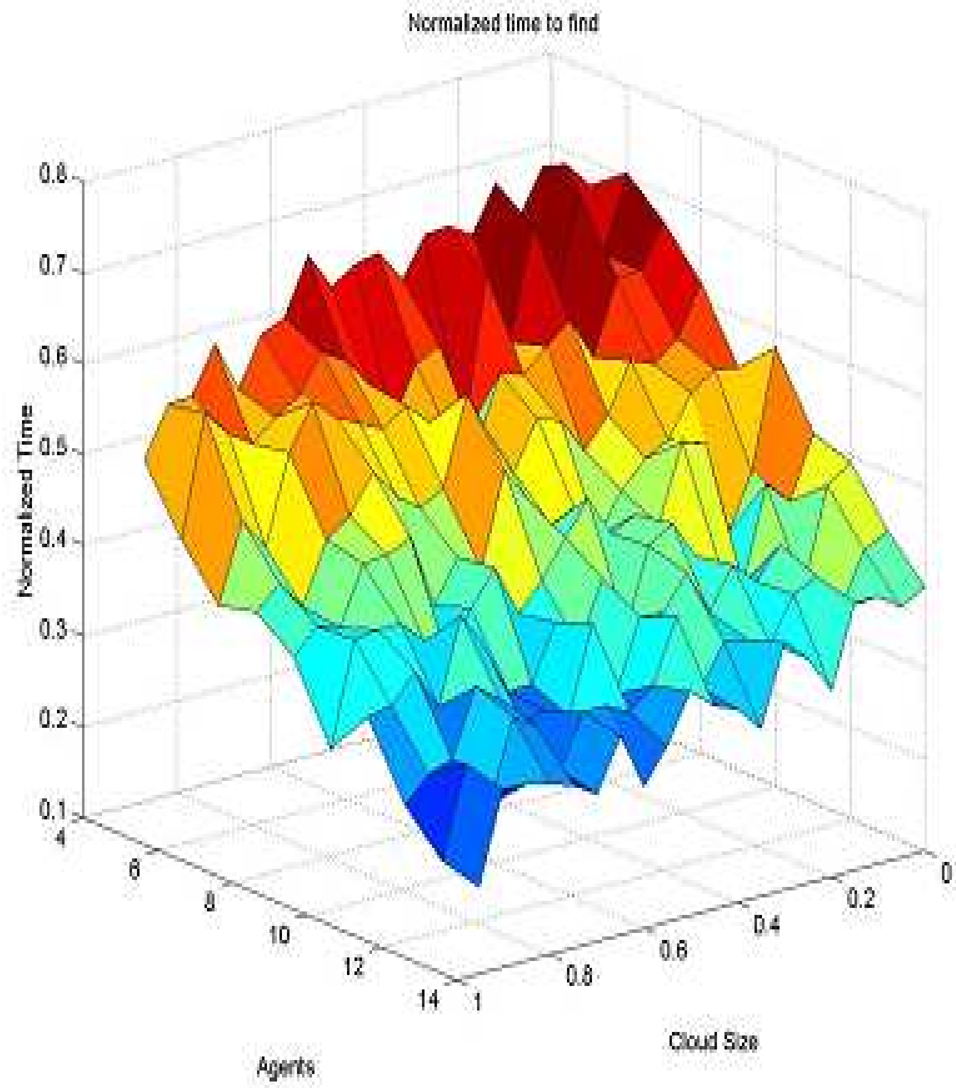
- Cloud detection only
- 20,000 simulations
  - 5 to 15 agents
  - 20 cloud sizes
- 30 minute power supply
- 40 knots constant speed
- Constant wind speed and direction

```
generateRandomPath()
loop
  if endOfPath() or rand() < .1 then
    generateRandomFlightPlan()
  end if
end loop
```



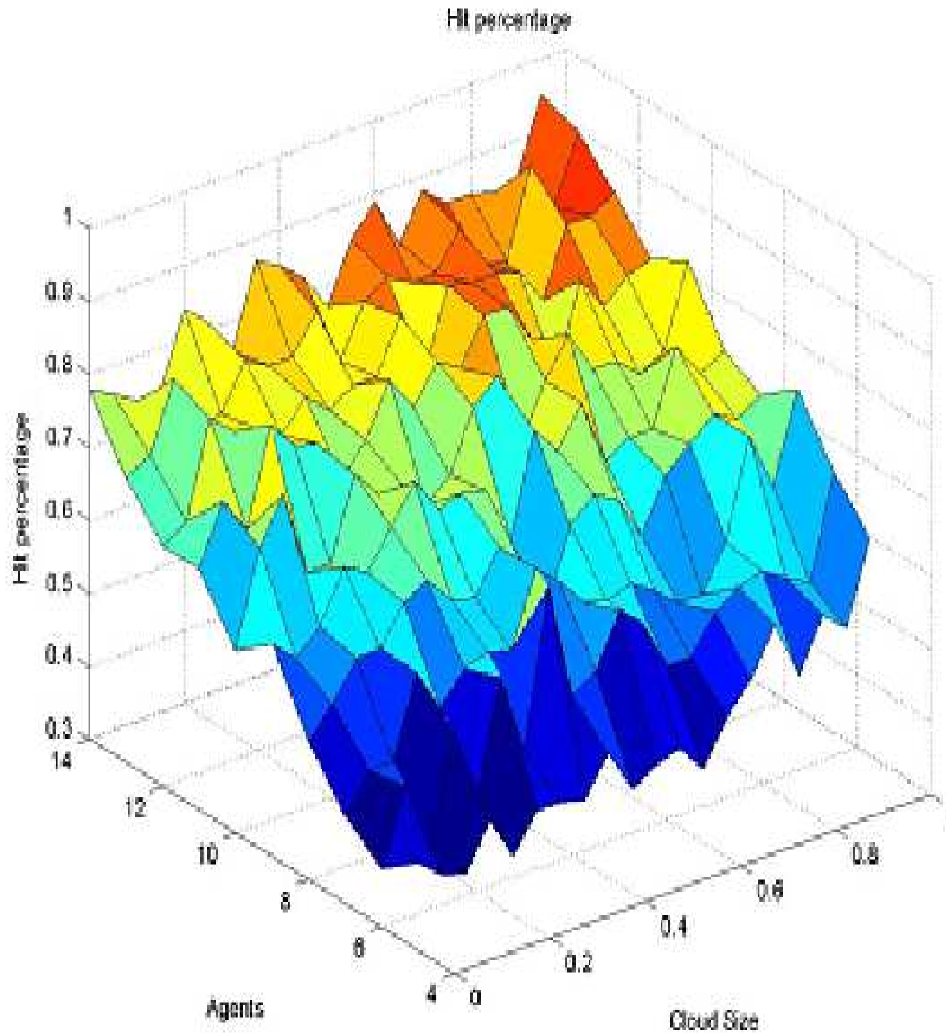
UAV Chemical Cloud Tracking

- Normalized search time
- Speed increases with swarm and cloud size
- Larger swarms are faster
- Bigger clouds are easier



UAV Chemical Cloud Tracking

- Detection rate
- Rate increases with swarm and cloud size
- Larger swarms are more accurate
- More UAVs, more area covered



Autogenerating Swarm Behaviors

Why autogenerate swarm behaviors?

- Trial-and-error gets tedious
- Complexity can quickly increase
- Emergence not always obvious
- Move away from low-level swarm programming

Autogenerating Swarm Behaviors

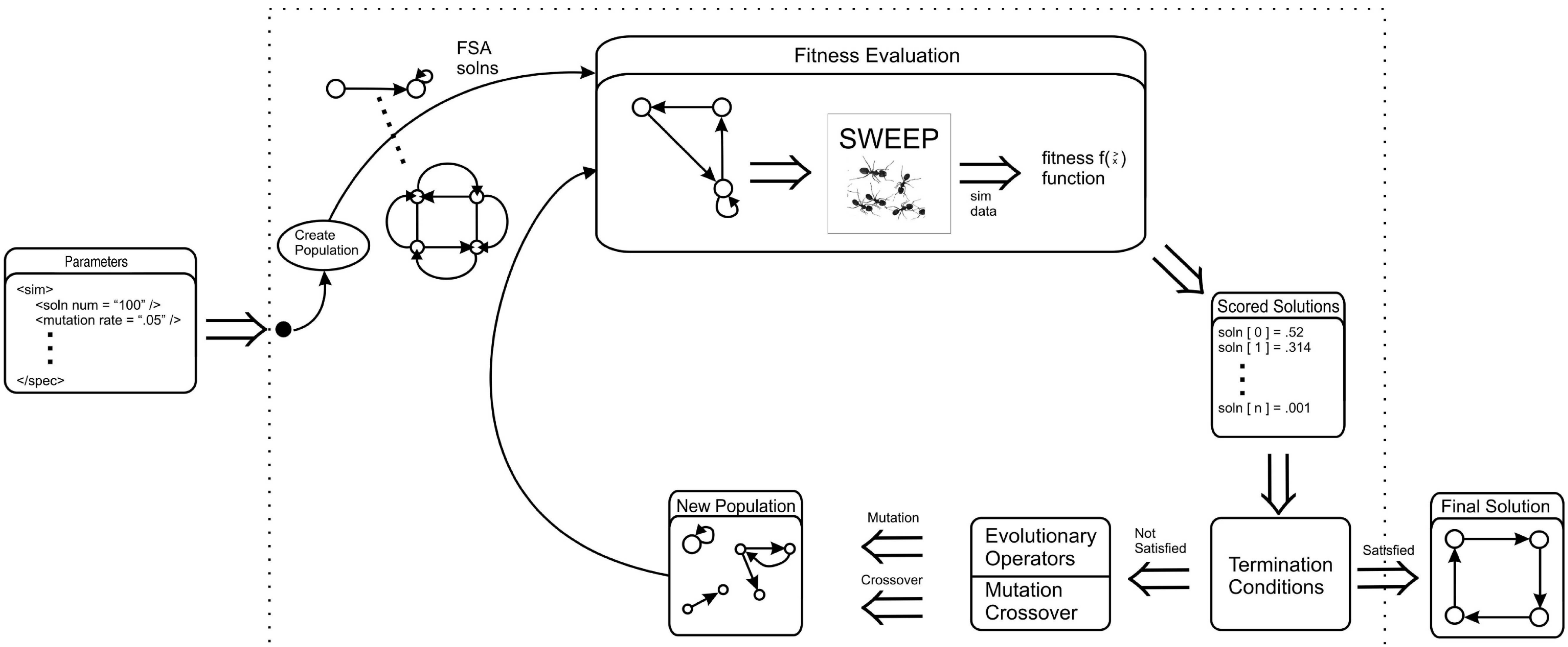
Why autogenerate swarm behaviors?

- Trial-and-error gets tedious
- Complexity can quickly increase
- Emergence not always obvious
- Move away from low-level swarm programming

What is needed?

- Specify high-level goals
- Define lower-level behaviors, sensors, . . .
- Use simulation to evaluate performance

ECS Overview



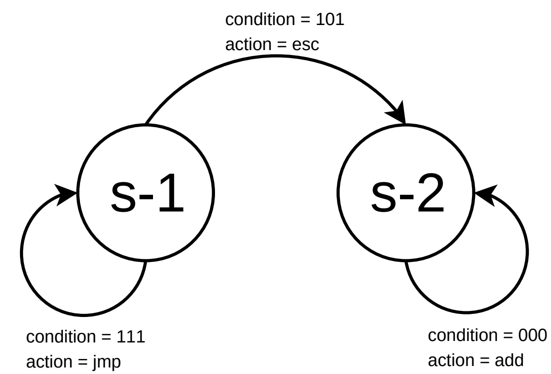
ECS - System Parameters

Parameters	
Objective	Dispersion
Max. Generations	500
Population Size	32
Number of Sims	2
Mutations	
Change-Sensor-Value	top 6 + 2 random
Add-Transition	top 6 + 2 random
Actions	*Sensors*
move-random	too-many neighbors
move-none	chemical-present
Simulation	
Number of Agents	100
Environment	50×50 grid

ECS - Solution Representation

SWEEP XML state machine

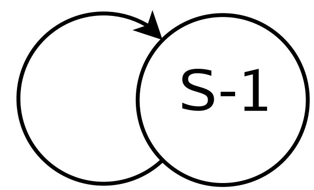
- Simple but expressive
- Graph-based
- Robust to random modification




```
<states>
  <state name="A">
    <transition nextState="s-1">
      <sensor name="holding-object" value="true"/> ...
      <action name="jmp"/>
    </transition>
  </state>
  ⋮
```

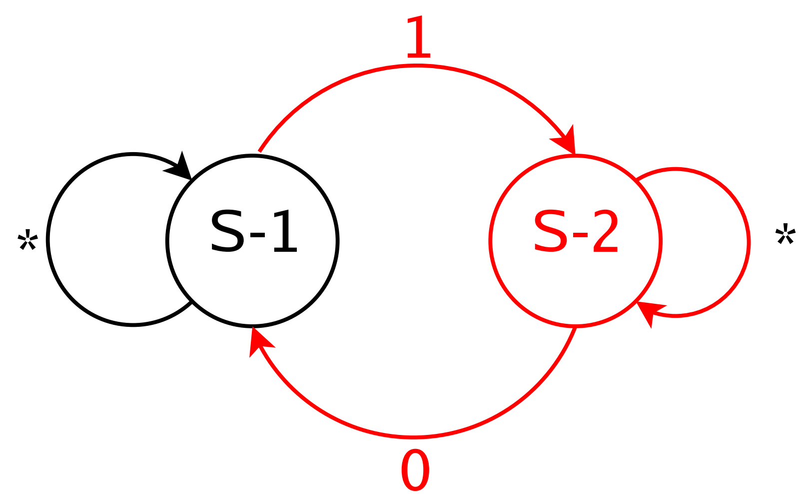
ECS - Mutations

- AddState
- AddTransition
- ChangeNextState
- InvertSensor
- ChangeAction



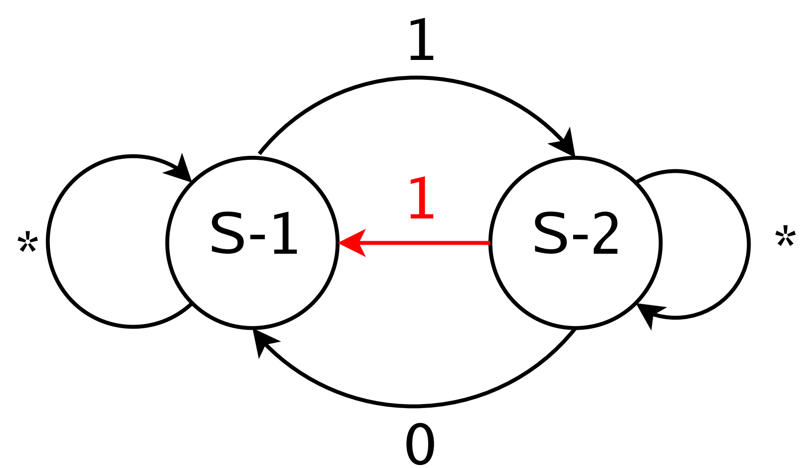
ECS - Mutations

- AddState
- AddTransition
- ChangeNextState
- InvertSensor
- ChangeAction



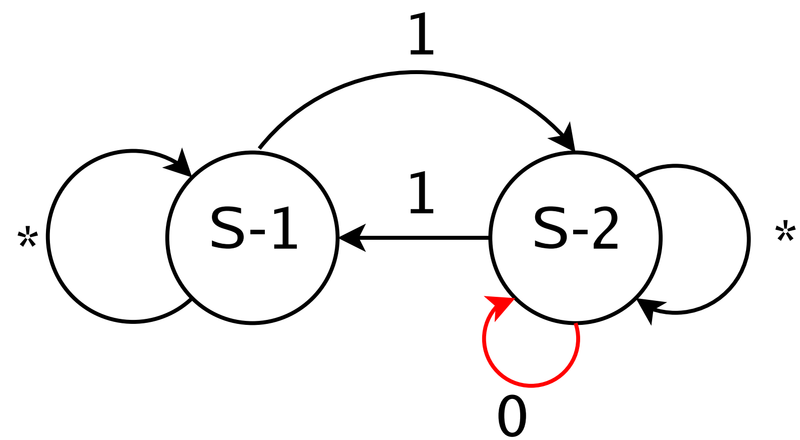
ECS - Mutations

- AddState
- AddTransition
- ChangeNextState
- InvertSensor
- ChangeAction



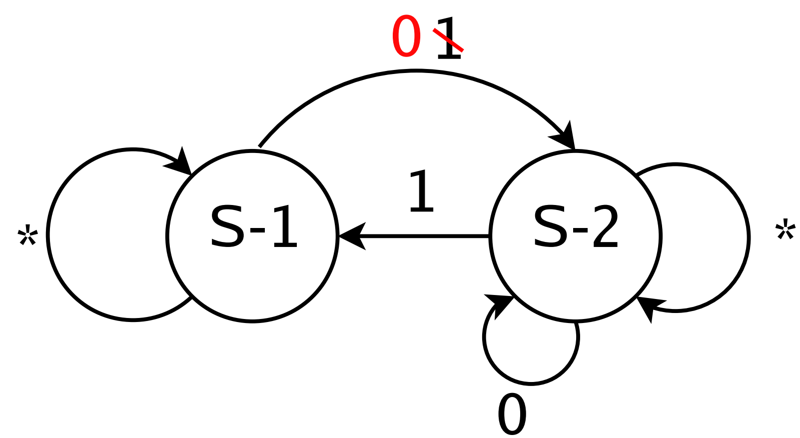
ECS - Mutations

- AddState
- AddTransition
- ChangeNextState
- InvertSensor
- ChangeAction



ECS - Mutations

- AddState
- AddTransition
- ChangeNextState
- InvertSensor
- ChangeAction



ECS - Fitness Evaluation

- Differentiate good and bad solutions, imposes order
- Solutions simulated in SWEEP
  - One state machine solution $\rightarrow$ single agent program
  - Homogeneous swarm
- Multiple runs, remove biases
- Calculate fitness from raw SWEEP output
- Error proportional to fitness, normalized
- Example

sweep(s_1)	= 7	*sweep*(s_2)	= 12
error(s_1)	= 20-7 = 13	*error*(s_2)	= 20-12 = 8
fitness(s_1)	= 13/20 = 0.65	*fitness*(s_2)	= 12/20 = 0.40

$\boxed{s_2 \text{ more fit}}$

Evolving Swarm Algorithms

Four scenarios examined:

- Agent Dispersion
- Object Collection
- Object Destruction
- Object Manipulation

  Simultaneous object collection and destruction

Object Manipulation

Two types of objects

- $C \rightarrow$ objects to be collected
- $D \rightarrow$ objects to be destroyed

Swarm Goal

- Collect all C objects
- Destroy all D objects

Approach

1. Collection
2. Destruction
3. Collection and Destruction

Object Manipulation

Behavior	**Scenarios**		
	Collection	Destruction	Manipulation
move-up	x	x	x
move-down	x	x	x
move-left	x	x	x
move-right	x	x	x
move-random	x	x	x
pick-up	x		x
put-down	x		x
move-to-goal	x		x
broadcast_C	x		x
move-to-object_C	x		x
first-attack		x	x
second-attack		x	x
broadcast_D		x	x
move-to-object_D		x	x

Object Manipulation

Sensor	Scenarios		
	Collection	Destruction	Manipulation
near-object_C	x		x
on-object_C	x		x
holding-object_C	x		x
on-goal	x		x
near-object_D		x	x
on-object_D(untouched)		x	x
on-object_D(damaged)		x	x

Object Manipulation

Fitness Metric	Description
c_1	number of objects picked up but not put in the goal
c_2	number of objects not collected
d_1	number of objects in the partially destroyed state
d_2	number of objects in the untouched state
t	number of time steps

Object Manipulation

The challenge

Collection and destruction metrics
are independent but equally weighted

- Imposing an order skews evolution
- “Experts” are evolved
- Solution
  - Construct new composite metrics
  - Eliminate sequential dependencies
  - Rank solutions using radix-based sorting

Object Manipulation

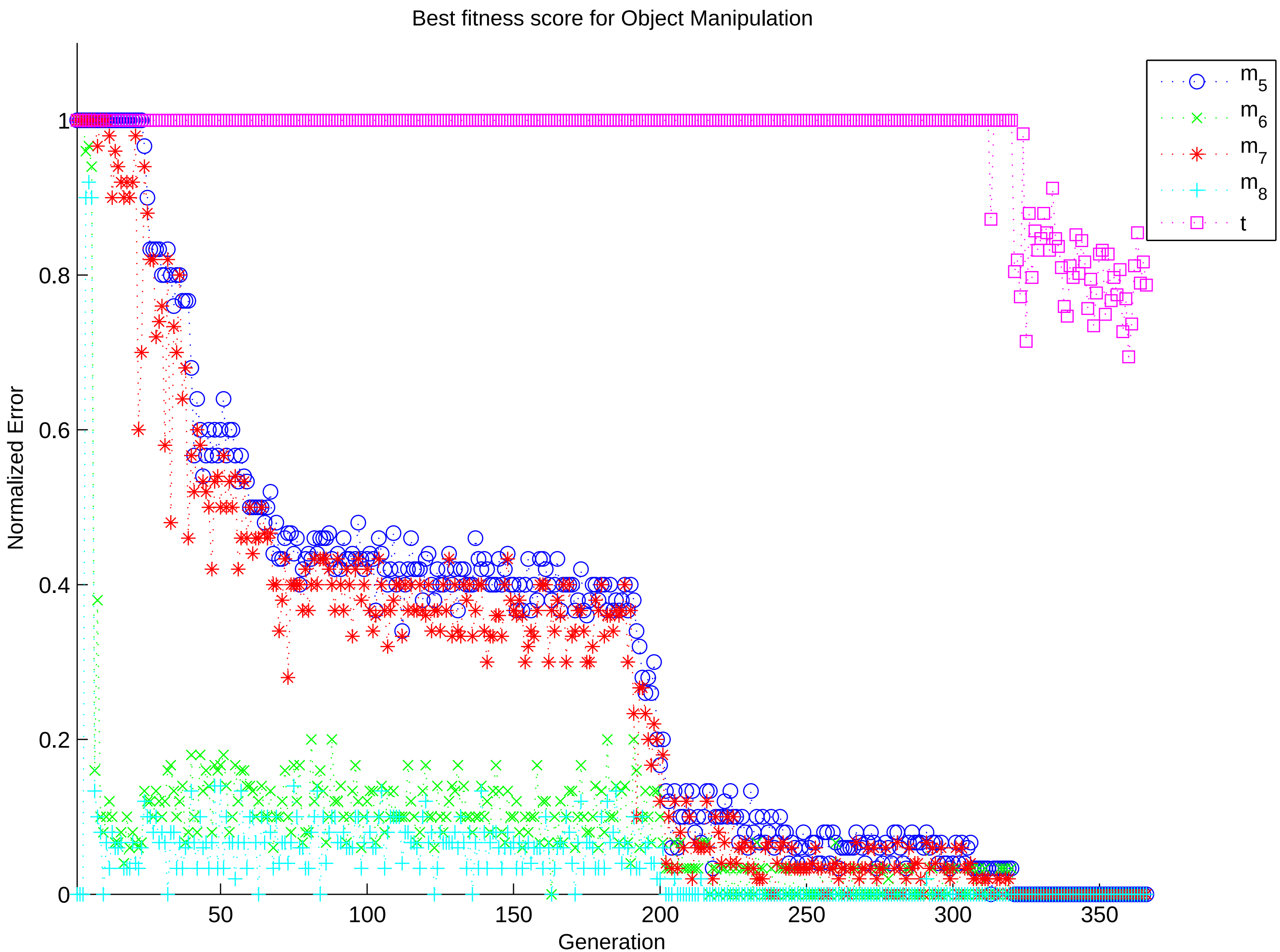
Composite Metric	Composition	Description
m_1	$\sim c_1 \wedge \sim d_1$	flag, fully performing both
m_2	$\sim c_2 \wedge \sim d_2$	flag, partially performing both
m_3	$\sim c_1 \vee \sim d_1$	flag, fully performing either
m_4	$\sim c_2 \vee \sim d_2$	flag, partially performing either
m_5	$\max(c_1, d_1)$	select the weakest
m_6	$\max(c_2, d_2)$	select the weakest
m_7	$\min(c_1, d_1)$	select the strongest
m_8	$\min(c_2, d_2)$	select the strongest
m_9	t	number of timesteps

Object Manipulation

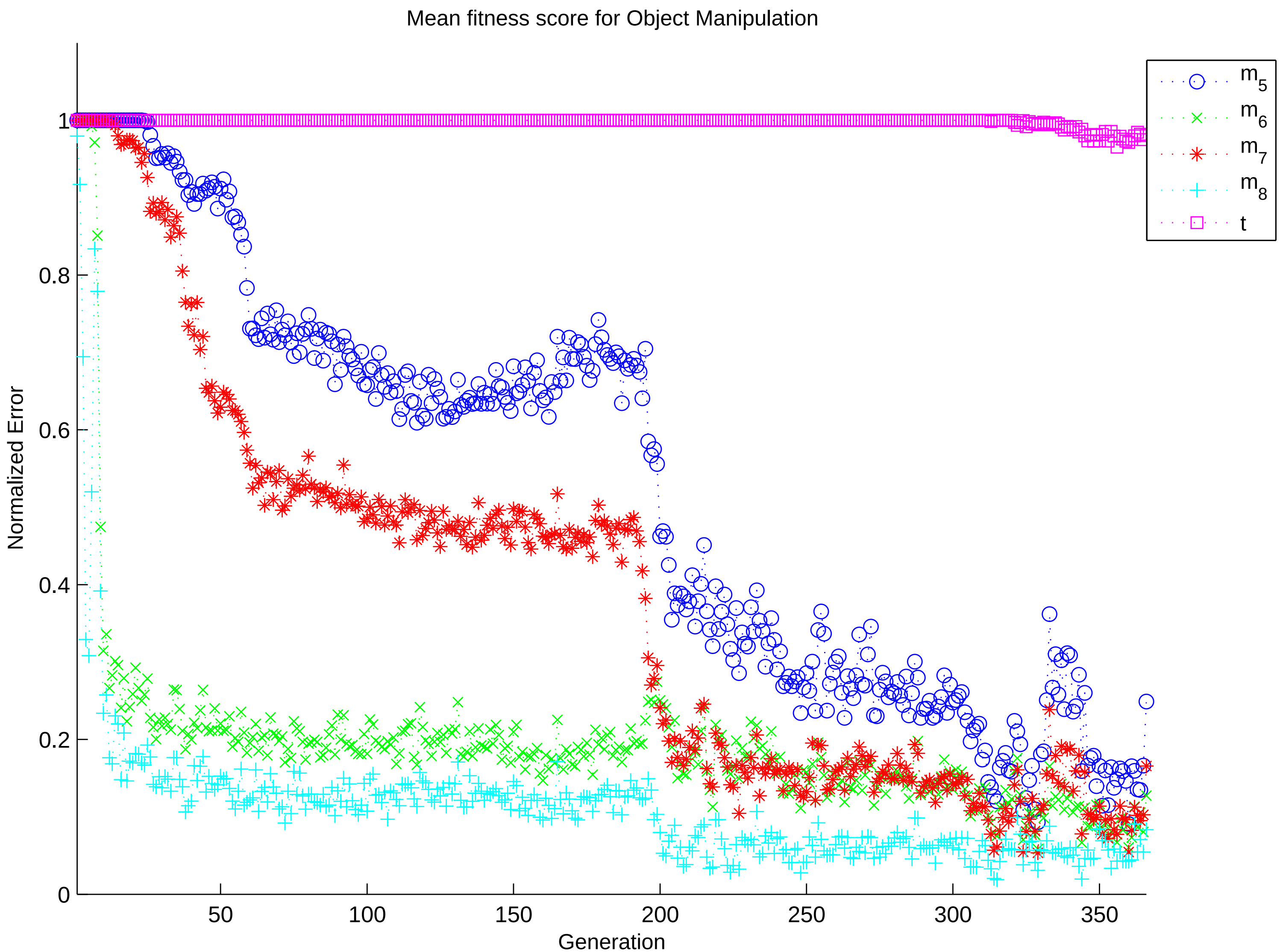
Parameters

- Population: 32
- Mutations: all top 6 + 2 random
- SWEEP: 100 agents, 50×50 grid
- C objects = 50
- D objects = 30
- Broadcast range = 25
- Sensing range = 5

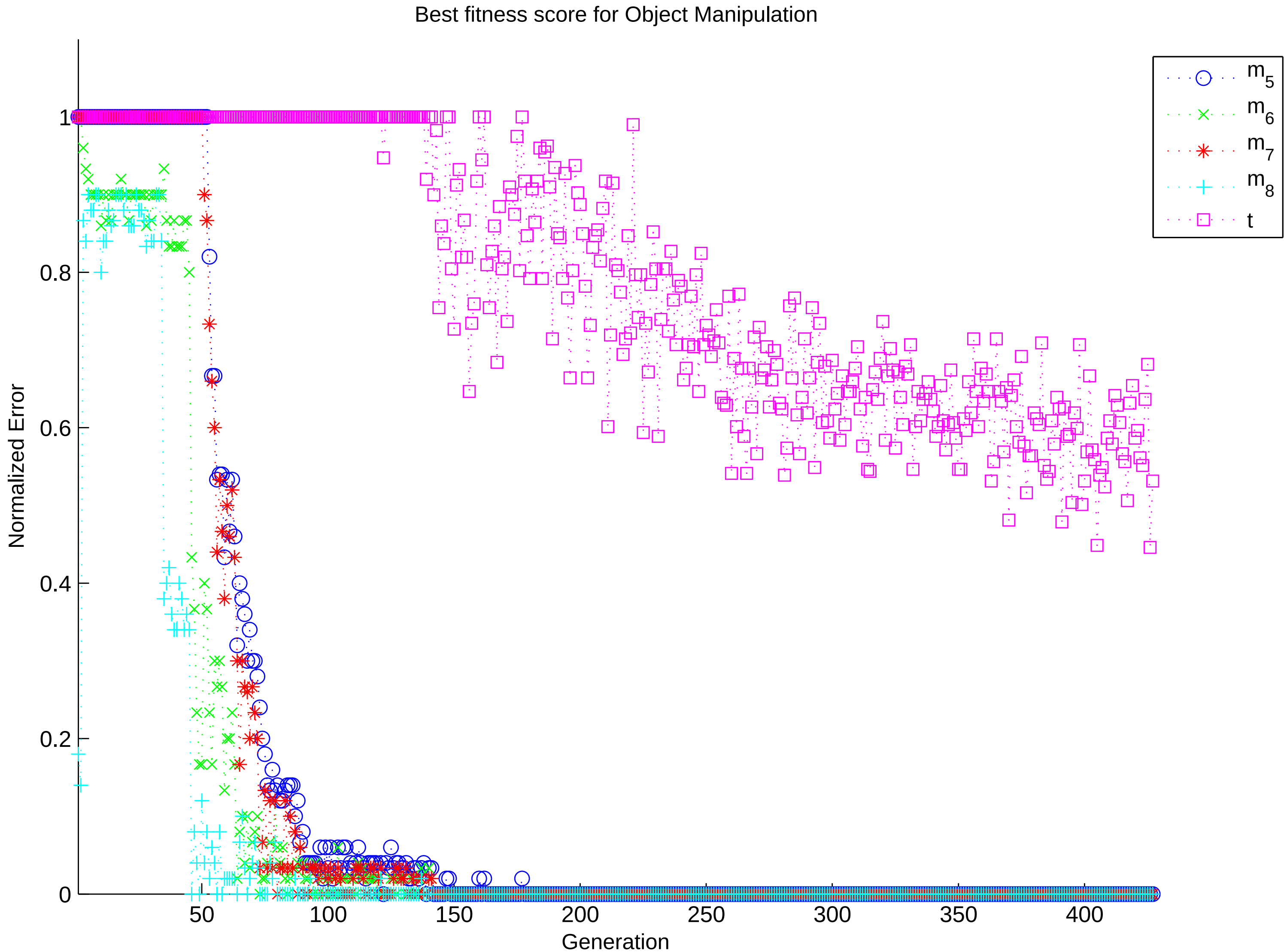
Object Manipulation



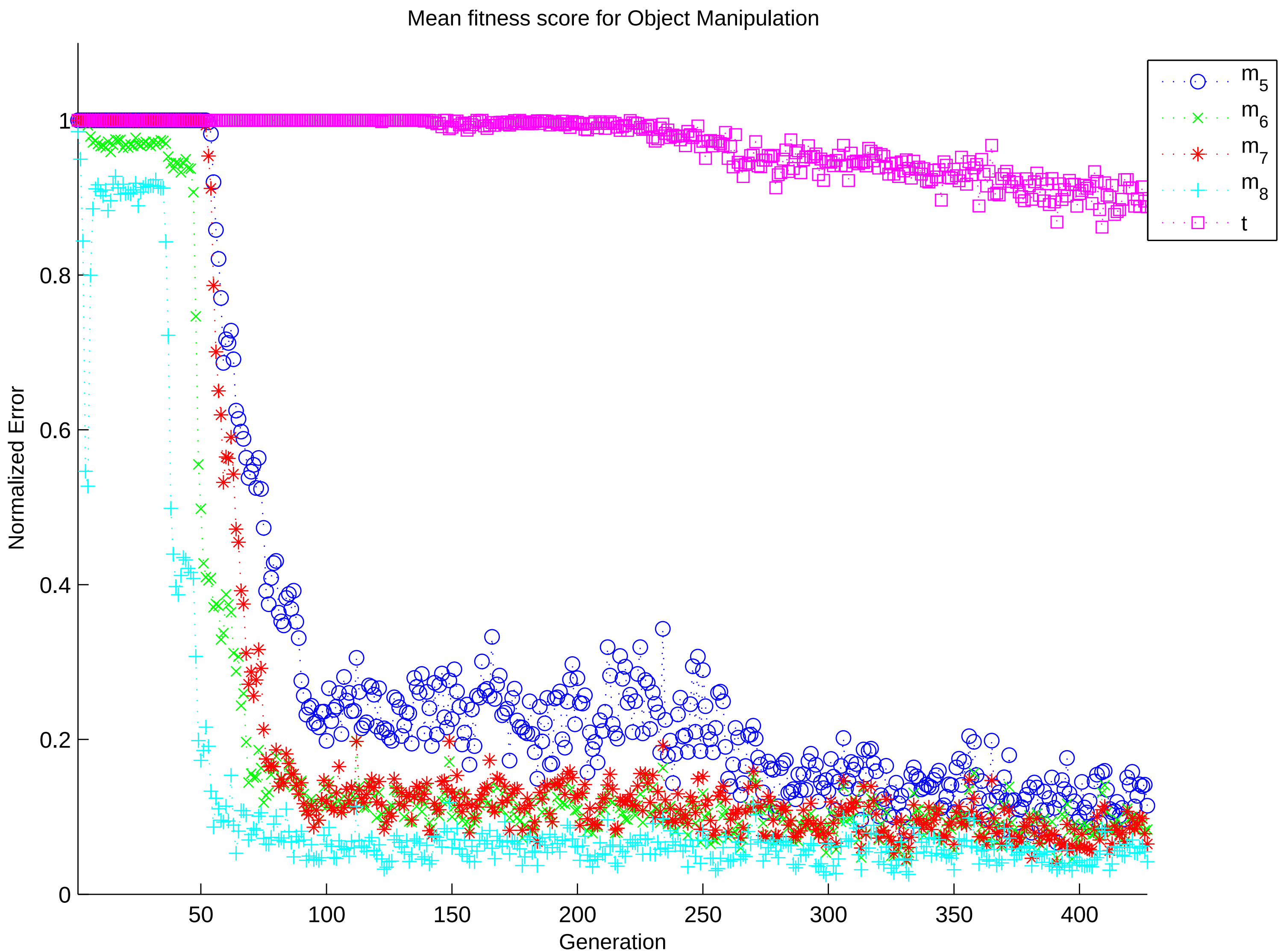
Object Manipulation



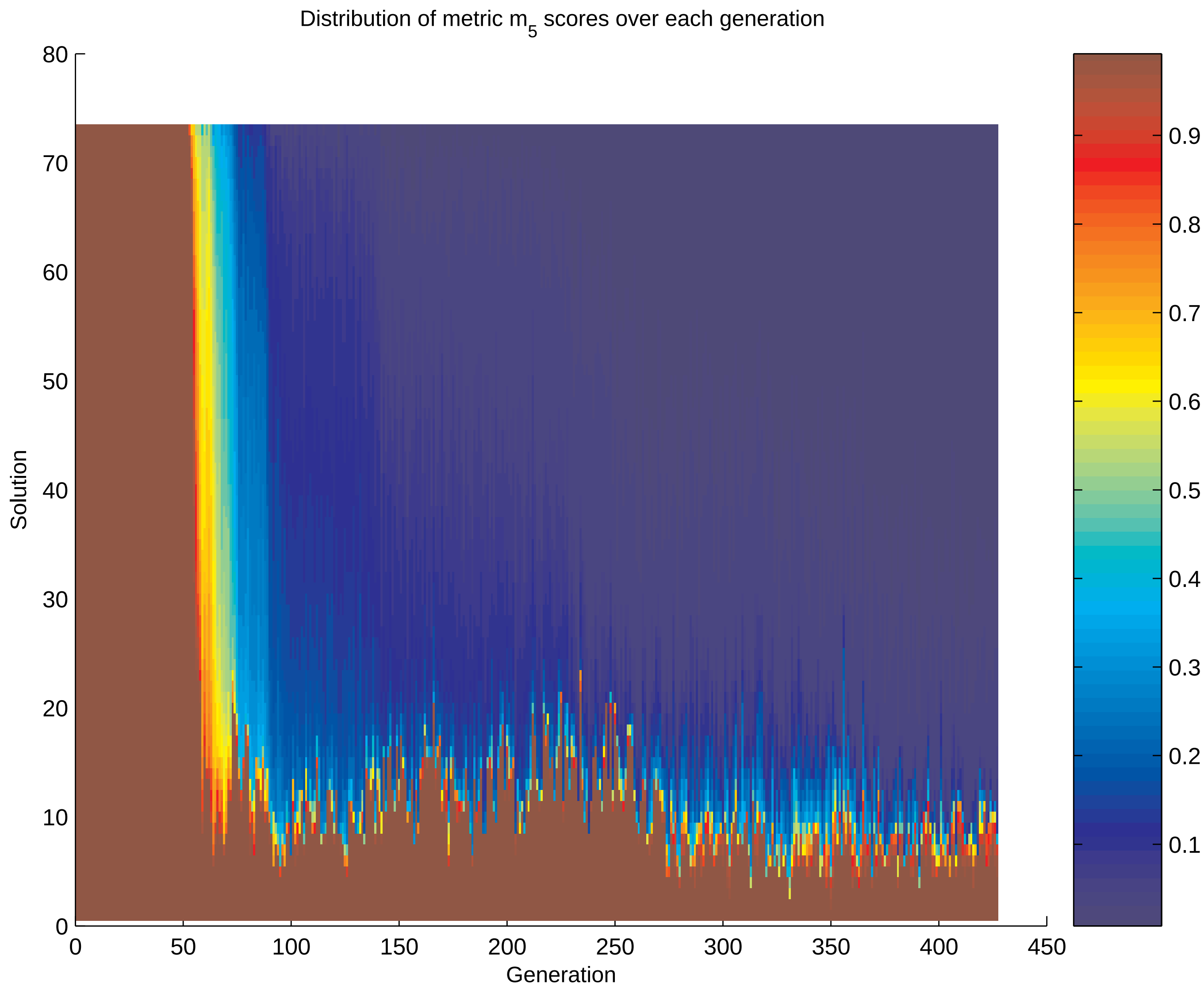
Object Manipulation



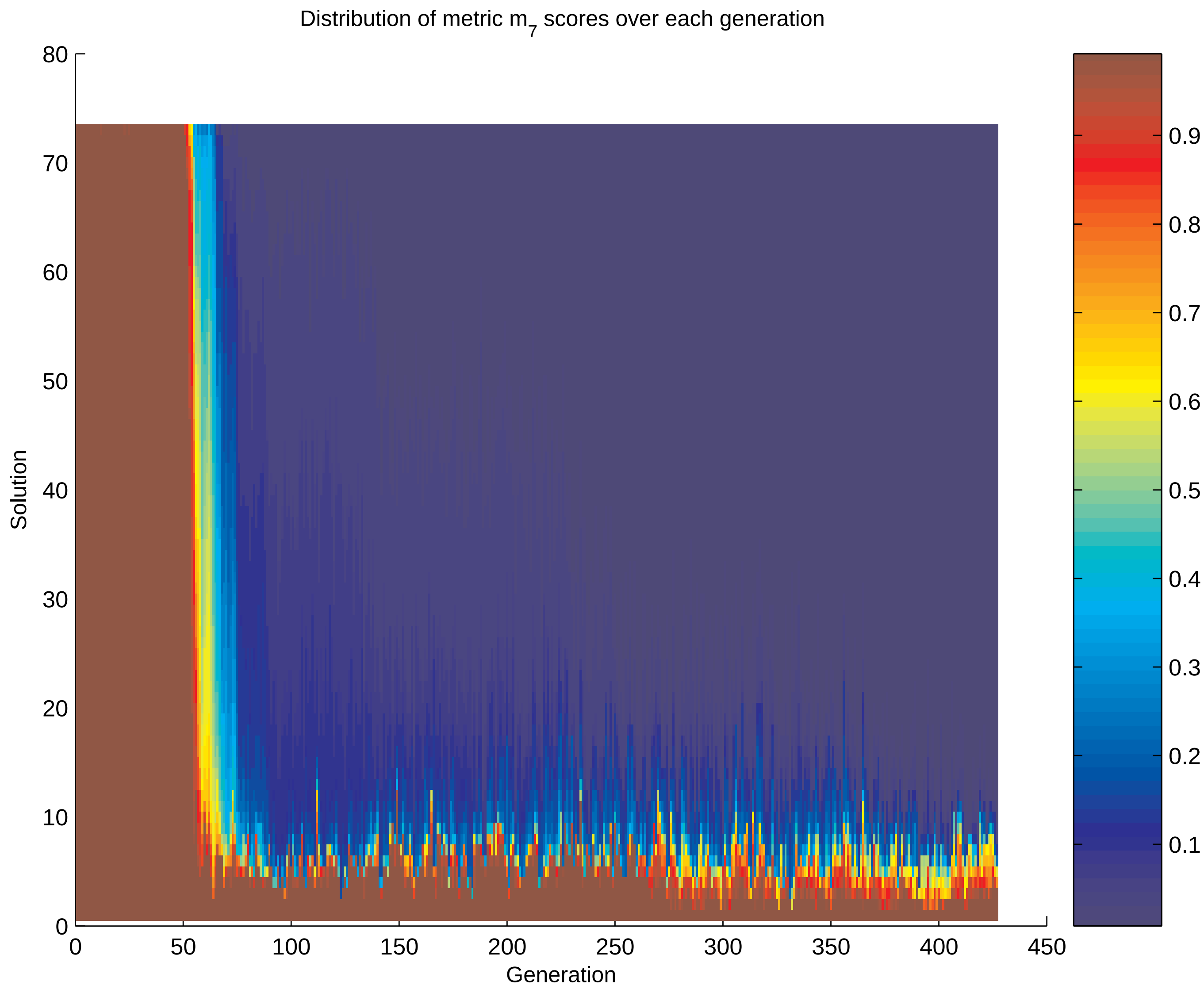
Object Manipulation



Object Manipulation



Object Manipulation



Conclusions

- Re-designed and Implemented SWEEP
  - Demonstrated the capabilities of SWEEP
  - Better suited for larger/more complex problems
  - Successfully used in applications outside this work
- Designed and implemented ECS
- Established the feasibility of evolving state machines for swarm algorithms
- Successfully generated swarm algorithms for a number of different scenarios
- Demonstrated the use of composite metrics and radix-based ranking to address multi-objective problems

Future Work

- More efficient SWEEP core
- Build a standard SWEEP component library
- Use aspect-oriented programming for probing
- Explore other solution encodings
- Attempt more difficult problems
- Autogenerate composite metrics from high-level goals
- Methods of detecting / measuring emergent behaviors

Acknowledgments

Advisor(s):

- Dr. Michael Branicky
- Dr. Dan Palmer
- Dr. Ravi Vaidyanathan

Committee:

- Dr. Randall Beer
- Dr. Roger Quinn

Also:

- Orbital Research, Inc.

Questions

Backup Slides

Swarm Intelligence

Definitions

- Beni, Hackwood, and Wang

  Unintelligent agents with limited processing capabilities, but possessing behaviors that collectively are intelligent

Swarm Intelligence

Definitions

- Bonabeau, Dorigo, and Theraluz

  Any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies

Swarm Intelligence

Definitions

- Clough

  A collection of simple autonomous agents that depend on local sensing and reactive behaviors to emerge global behaviors

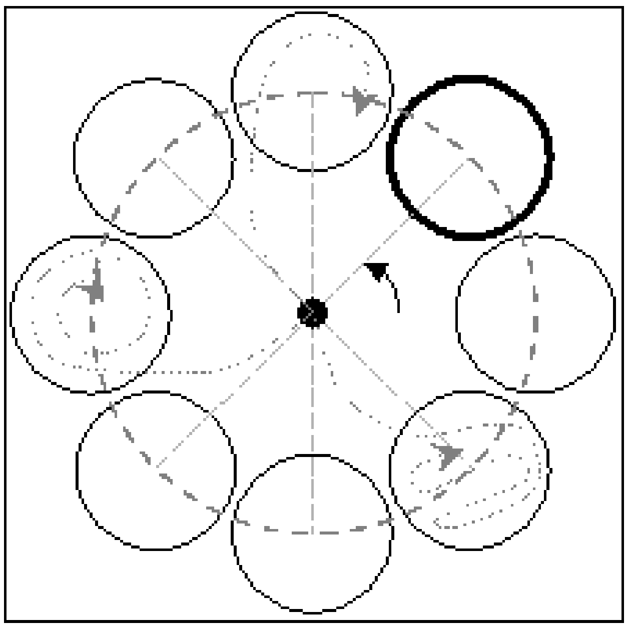
Swarm Intelligence

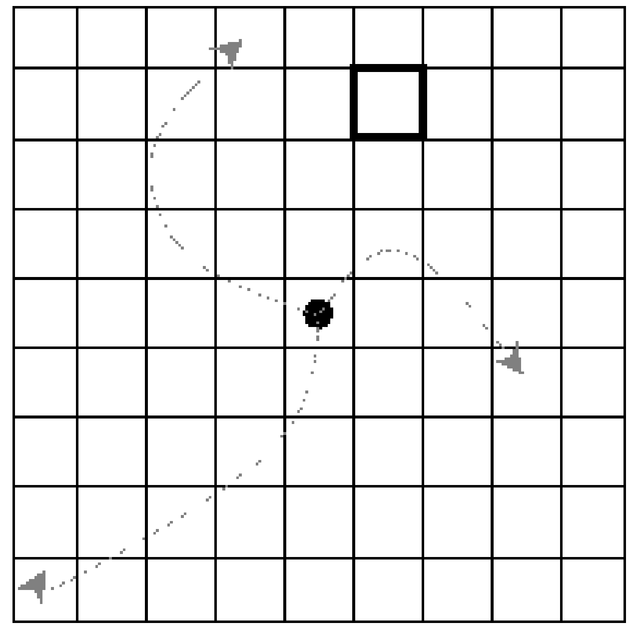
Benefits of swarm intelligence

- Robust
- Distributed
- Parallel
- Simple agents
- Scalability
- Effort Magnification

UAV Chemical Cloud Tracking

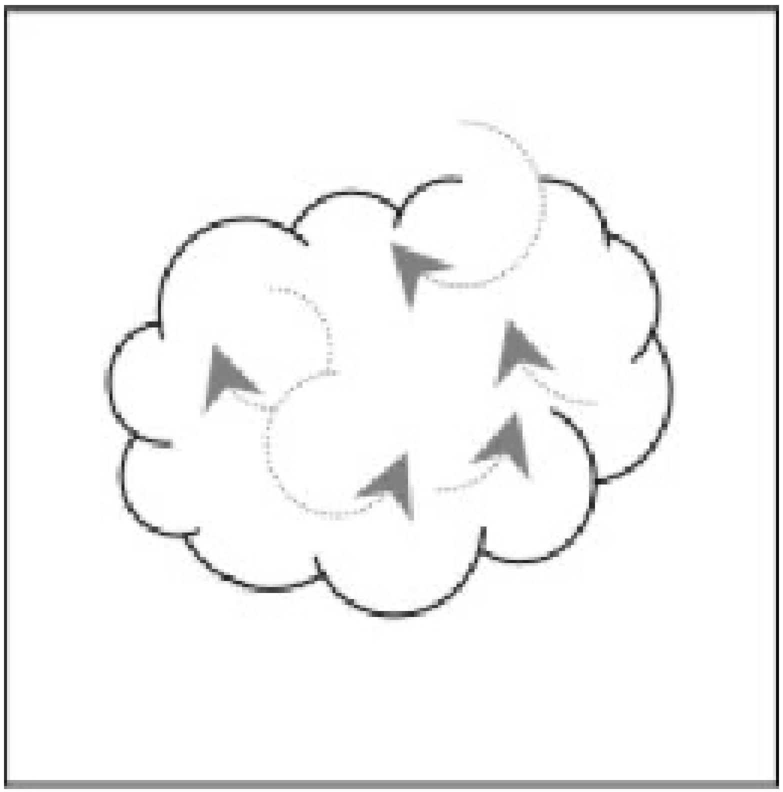
Searching

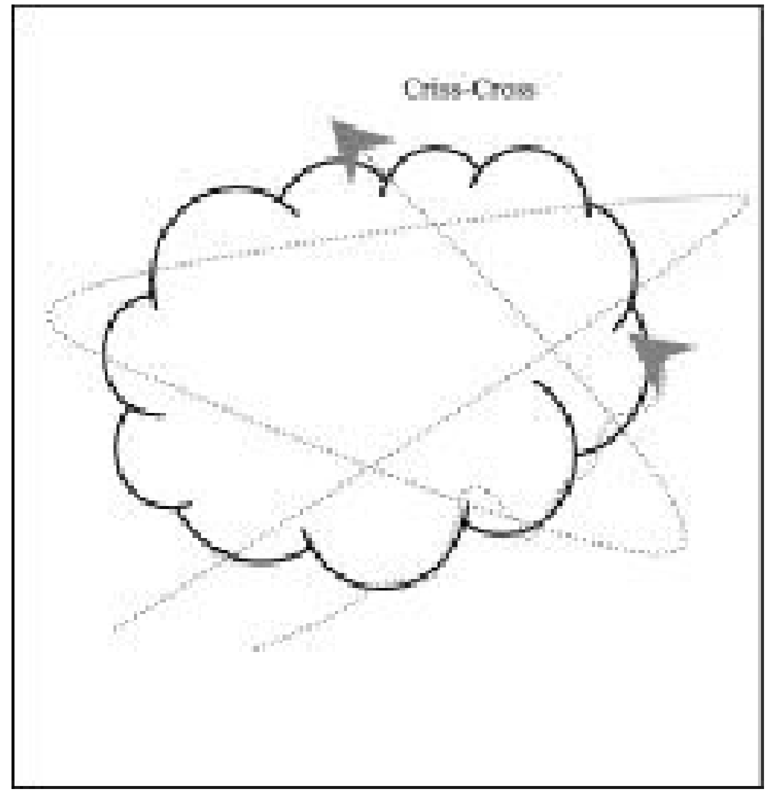




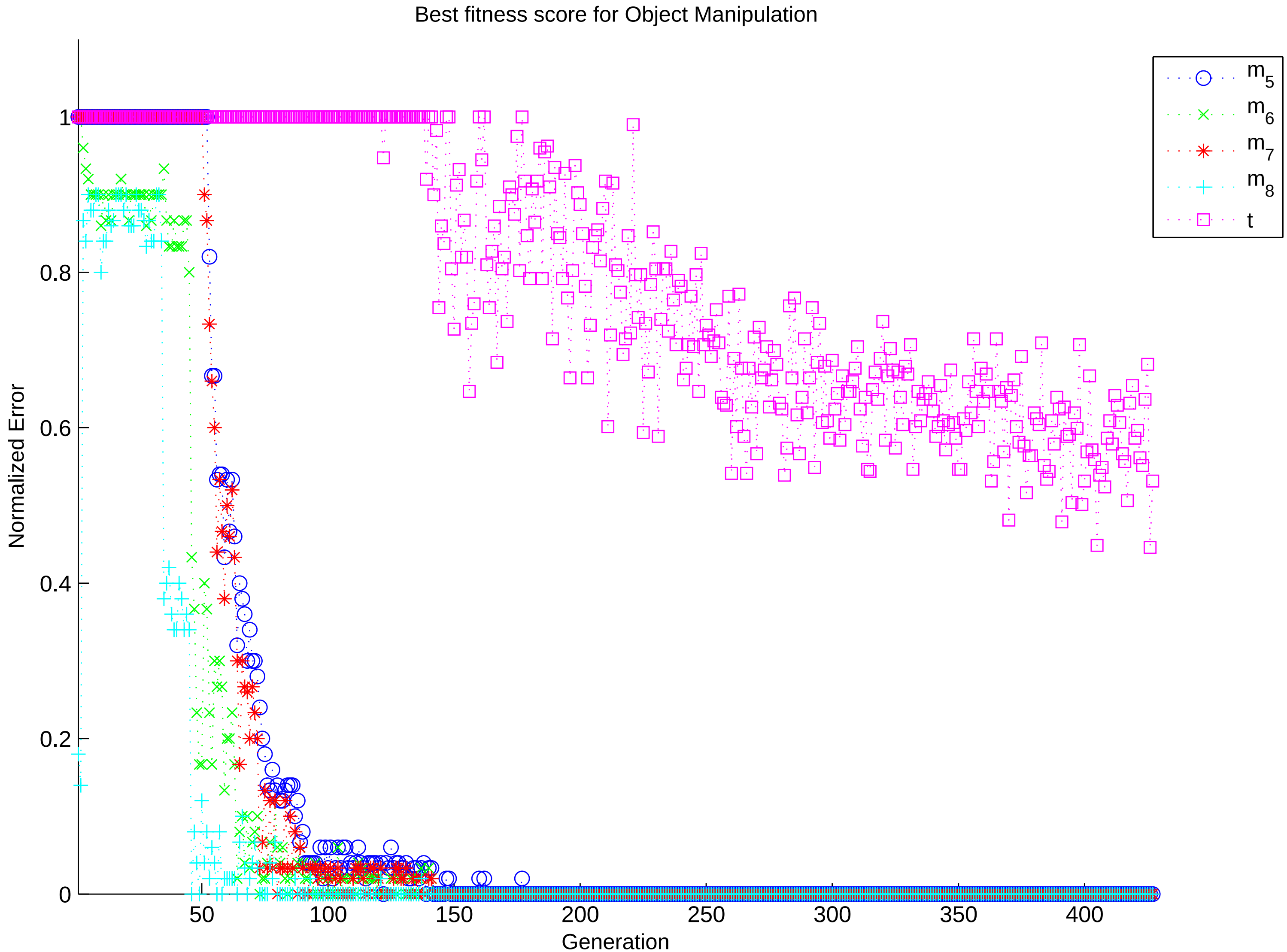
UAV Chemical Cloud Tracking

Mapping

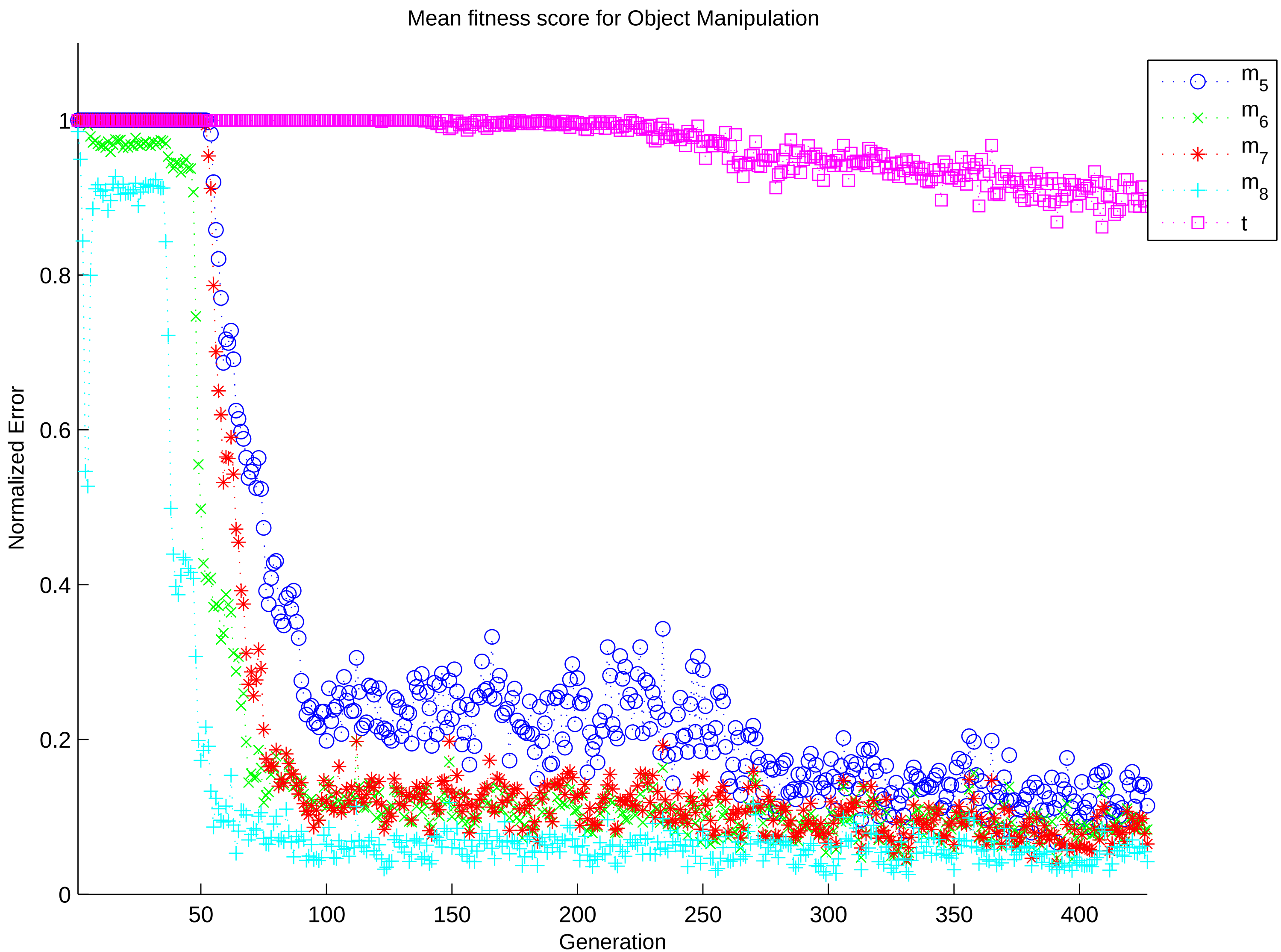




Object Manipulation

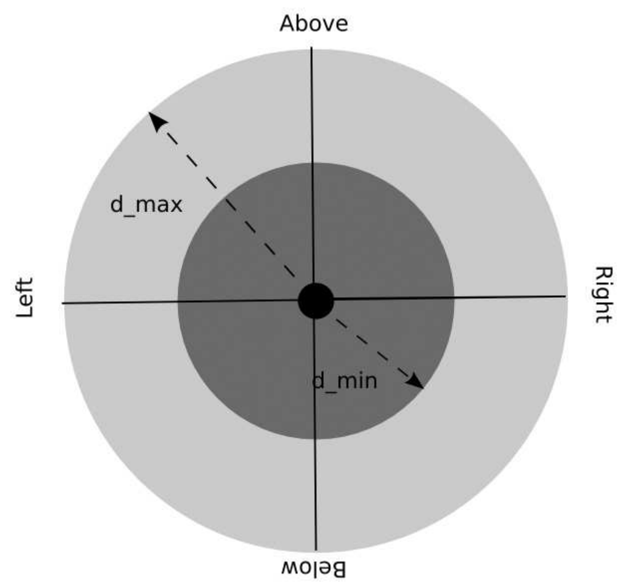


Object Manipulation



Dispersion

- Swarm Goal: achieve a density level
- Agent Goal:
  - neighbor at least d_{min} units away
  - neighbor not more than d_{max} units away



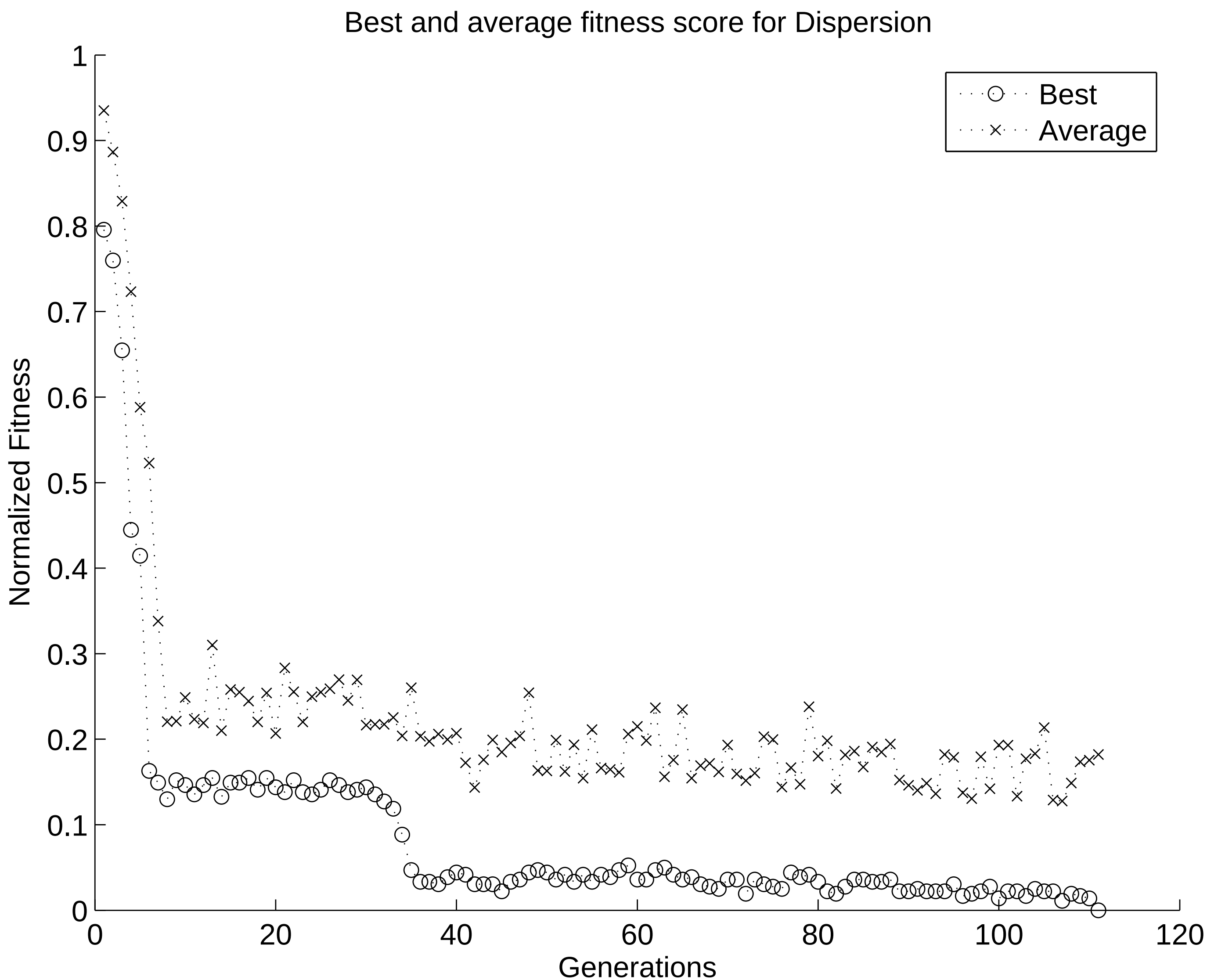
Dispersion

- Fitness: Number of agents violating dispersion criteria

Actions	**Sensors**
move-up	neighbor-above
move-down	neighbor-below
move-left	neighbor-left
move-right	neighbor-right

- Population: 32
- Mutations: all top 6 + 2 random
- SWEEP: 100 agents, 50×50 grid
- $d_{min} = 2$, $d_{max} = 4, range = 6$

Dispersion



Best and average fitness score for Dispersion
Best
Average
Normalized Fitness
Generations
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
20
40
60
80
100
120