Swarm Intelligence
Outline

- Overview of swarm intelligence

- Paper presentation
  - Dom Antanasio
Overview

- Swarm intelligence (SI) is a type of artificial intelligence based on the collective behavior of decentralized, self-organized systems.
- SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment.
- There is no centralized control.
- Interactions between agents lead to the emergence of global intelligent behavior.
Nature flocks consist of two balanced, opposing behaviors:
- Desire to stay close to flock
- Desire to avoid collisions within flock

Why desire to stay close to flock?
- Protection from predators
- Improving survival
- Profit from effective search of food
- Advantages for social and mating activities
Properties of Swarming

- Made up of **discrete agents**, yet overall motion seems fluid

- **Simple** in concept, yet visually complex

- **Randomly** arrayed, yet highly synchronized

- Seems intentional, with centralized control, yet evidence suggests group motion is only due to **aggregate** result of individual agents
Computational Complexity of Flocking?

- In natural systems:
  - No indication that flocking is bounded
  - Flocks don’t become “overloaded” or “full” as new agents join
    - E.g., Herring migration: schools are as long as 17 miles and contain millions of fish
  - Individual natural agent (e.g., bird) doesn’t seem to pay attention to each flockmate
  - In birds, seem to be 3 categories of awareness:
    - Itself
    - 2-3 closest neighbors
    - Rest of flock
From Wikipedia and other sources

EXAMPLE ALGORITHMS
Altruism Algorithm

- Joint research by robotics professor and biologist in Switzerland
- Application of Hamilton’s rule of kin selection developed in 1964
- It is shown that the theory predicts the relationship between evolution of altruism and relatedness of individuals in a species
- Used to improve control system of flying robots so that they fly in swarm formation more successfully
- http://genevalunch.com/blog/2011/05/04/altruism-helps-swarming-robots-fly-better-study-shows/
Ant Colony Optimization (ACO)

- Contains a class of optimization algorithms modeled on the actions of an ant colony
- Useful for finding paths
Particle Systems

- **Particle systems:**
  - Collections of large numbers of individual particles, each having its own behavior
  - Used to model fire, smoke, clouds, spray/foam of ocean waves, etc.
  - Particles are created, age, and die off

- However, need to also add:
  - Geometric objects with local coordinate system, since robots aren’t points
  - Geometric/kinematic models of motion of a body
Paper Reading


Robots Used in Study

- "Nerd Herd" collection of 20 robots:
  - 12" long
  - 4 wheels
  - bump sensors around body
  - Two pronged grippers
    - Contact switches on tip of each gripper finger
    - 6 infrared sensors (2 forward, 2 inside, 2 down)
- Radio system for:
  - Localization (based on triangulation with data from two fixed base stations)
  - Communication
  - Data collection
  - “Kin” recognition

The Nerd Herd: Mataric, MIT, 1994
Basic Idea

- Work of Mataric, 1994
- General idea:
  - Use “local” control laws to generate desired “global” behavior
- Fundamental principle:
  - Define basis behaviors as general building blocks for synthesizing group behavior

The Nerd Herd: Mataric, MIT, 1994
Basic Behaviors

Set of behaviors:
- **Safe-wandering**: ability of group to move about while avoiding collisions
- **Following**: ability of an agent to move behind another agent
- **Aggregation**: ability of a group to gather so as to maintain some maximum inter-agent distance
- **Dispersion**: ability of a group to spread out so as to establish and maintain some minimum inter-agent distance
- **Homing**: ability to find a particular region or location
Objective: Combine Basis Behaviors

Idea:

- Use basis behaviors in a variety of combinations to enable creation of more complex group behaviors, such as:
  - Flocking
  - Foraging
  - Surrounding
  - Herding
Safe-Wandering Algorithm

• **Avoid-Kin:**
  - Whenever an agent is within d_avoid
    • If the nearest agent is on the left
      - Turn right
      - Otherwise, turn left

• **Avoid-Everything-Else**
  - Whenever an obstacle is within d_avoid
    • If obstacle is on right only, turn left
    • If obstacle is on left only, turn right
    • After 3 consecutive identical turns, backup and turn
    • If an obstacle is on both sides, stop and wait.
    • If an obstacle persists on both sides, turn randomly and back up

• **Move-Around:**
  - Otherwise move forward by d_forward, turn randomly
Following Algorithm

Follow:
- Whenever an agent is within d_follow
  - If an agent is on the right only, turn right
  - If an agent is on the left only, turn left

Ant osmotropotaxis is based on the differential in pheromone intensity perceived by the left and right antennae.

If sufficient robot density, safe_wandering + follow yield more complex behaviors:
  - e.g., osmotropotaxic behavior of ants: unidirectional lanes
Dispersion Algorithm

Dispersion:
- Whenever one or more agents are within d_disperse
  - Move away from Centroid_disperse

![Diagram with grid and agent paths](image-url)
Aggregation Algorithm

**Aggregate:**

- Whenever nearest agent is outside $d_{aggregate}$
  - Turn toward the local centroid_aggregate, go.
- Otherwise, stop.
Homing Algorithm

**Home:**
- Whenever at home
  - Stop
- Otherwise, turn toward home, go.

Greedy local pursuit

Homing became increasingly inefficient as the group size grew.
Combining Behaviors

Two types of coordination

- **Complementary**
  - Outputs are expected concurrently
  - Direct combination (vector summation)

- **Contradictory**
  - Outputs are mutually exclusive and can only executed one at a time
  - Temporal combination (a sequence)
Generating Flocking Through Behavior Combinations

• Flock:
  - Sum weighted outputs from Safe-Wander, Disperse, Aggregate, and Home

Safe-wandering, dispersion and aggregation produce robust flocking; homing gives the flock a goal location and direction to move in.

In general, flocking should allow agents to move around obstacles.
What Robot Capabilities are Needed to Implement Flocking?

- **Capabilities**
  - Ability to sense nearby robots
    - Position, direction, and speed of travel
  - Ability to sense nearby obstacles
  - Ability to distinguish between robots and obstacles

- **How to obtain?**
  - Vision recognition
  - Beacons on robots detected locally
  - Global positioning + radio broadcast of global robot positions
Flocking Behavior
Foraging Example

- Foraging:
  - Robot moves away from home base looking for attractor objects
  - When detect attractor object, move toward it, pick it up, and return it to home base
  - Repeat until all attractors collected at home base

- High-level behaviors required?
  - **Wander**: move through world in search of an attractor
  - **Acquire**: move toward attractor
  - **Retrieve**: return the attractor to home base
Finite State Acceptor for Foraging
Mataric’s Approach to Foraging

- Foraging:
  - Whenever crowded? Disperse.
  - Whenever at-home?
    - If have-puck? drop-puck
    - Otherwise disperse
  - Whenever sense-puck?
    - If not have-puck? pickup-puck
  - Whenever behind-kin? Follow.
What Mobile Robot Capabilities are Needed to Implement Foraging?

- **Capabilities:**
  - Ability to sense nearby robots’ positions
  - Ability to sense nearby obstacles
  - Ability to distinguish between robots and obstacles
  - Ability to sense and grasp pucks
  - Ability to find home
  - Ability to search for pucks

- **How to obtain?**
  - Vision recognition
  - Beacons on robots detected locally
  - Global positioning + radio broadcast of global robot positions
Foraging Behavior
STUDENT PRESENTATION
More Recent “Swarm” Robotics

- James McLurkin, MIT and iRobot
- Developed libraries of swarm behaviors:
  - avoidManyRobots
  - disperseFromSource
  - disperseFromLeaves
  - disperseUniformly
  - computeAverageBearing
  - followTheLeader
  - navigateGradient
  - clusterIntoGroups

- There are about 40 of the behaviors
McLurkin’s Robot Swarms

- Approach to generating behaviors is similar to Mataric’s, in principle

- Primary differences:
  - Algorithms more tuned to the SwarmBot
  - More exhaustively tested
  - Parameters explored
  - More kinds of behaviors
  - etc.

- The key challenge of swarm intelligence:
  - Find out the local interactions between nearby robots to produce large-scale group behaviors from the entire swarm
  - Often inspired from biological foundations, such as ants, bees, termites, etc.
Some Movies

Glumping
Video Shown 4X Speed

Matching Orientation

Navigation

Flowing
Video Shown 4X Speed
SwarmBot

- Charger Contacts
- User Interface Switches
- Expansion Port
- JTAG Port
- 230kbps Serial ports (x2)
- ISIS Infra-Red Tranceivers (x4)
- 1.1 Watt Audio System
- Bump Skirt/Sensors (x8)
- Behavior LEDs (x3)
- Light Sensors (x4)
- SwarmCam Emitters
- Camera
- Hard Power Switch
- 40 mhz ARM Processor
- 648 KB RAM
- 3 MB Flash
- 200 kgate FPGA
- 0.51 watt Drive Motors (x2)
ISIS Infrared Communication System

- Each robot has four ISIS transceivers, one in each corner

- Nearby robots can communicate and determine the bearing, orientation and range of their neighbors

- $R_{\text{safe}}$ is the maximum distance that provides reliable positioning
Gradient Communication

- A gradient-based multi-hop messaging protocol provides long-range communication.
- A source robot creates a gradient msg that is relayed throughout the network in a breadth-first fashion.
- Robot processes messages synchronously and relays the one with the lowest hop count.
More On Communication

- Robots periodically (250ns) transmit their state info (UID, task, any gradient messages they are relaying)
Direct Dispersion Algorithm

Goal: spread robots throughout an enclosed space quickly and uniformly, while keeping each robot connected to the network

Accomplished by both:

- **disperseUniformly**
  - Spreads robots evenly, using boundary conditions to limit dispersion

- **frontierGuidedDispersion**
  - Directs robots towards unexplored areas
Uniform Dispersion

- Disperse robots uniformly in an environment

- Boundaries – walls and max dispersion distance between robots

- Move robots based on vector sums of c closed neighbors
  - C = 2 works the best in practice
Frontier Guided Dispersion

- **Goal**
  - Guide robots to unexplored areas

- **Concerns**
  - Cannot disconnect
  - Self-stabilize for robot charging
  - Termination when completely dispersed

- **Idea**
  - Use robots on the frontiers to explore
  - Leads swarm
How to Determine Frontier

- **Three positions:**
  - **Wall** – Detects obstacles
  - **Frontier** – no walls no neighbors over some large angle
  - **Interior** - remainder
Swarm Motion - DisperseFromLeaves

- Frontier robots source a gradient msg
- Frontier tree guides swarm towards frontier
- Problem:
  - may pull robots away from explored area
  - Frontier reappears to pull back, thus oscilation
- Method: robots move away from children in frontier tree and make sure they are in communication
- Leaves become anchors
Example

These robots are frontier robots and guide the swarm into unexplored areas.

These robots can detect nearby walls.

These robots are in the interior of the swarm.

There are more robots in explored space in this direction.
Putting it Together

- Directed dispersion
  1. If there is a frontier, source gradient msg
  2. If there is a gradient msg, disperse from leaves
  3. Else disperse uniformly
  4. Terminates once there is no unexplored areas (no more frontier detected)
Experimental Results

- 56 robots were used
- Times to reach three goals were recorded
- Five algorithms were compared:
  - idealGasMotion
    - Robots move in straight lines but turn when they collide with each other or with a wall
    - Results: disconnected network, interference, no termination condition, dispersion is not uniform
- **disperseFromSource**
  - A robot near the base station sources a “disperse” gradient
  - Uniform, complete coverage only occurs for known environment and proper distance value
  - Very efficient

- **avoidClosestNeighbor**
  - Robots move away from their closest neighbor at constant velocity
  - Similar to disperseUniformly, but robots oscillate between closest neighbors
- **disperseUniformly**
  - Avoiding two closest neighbors
  - Motion is smoother
  - Uniform dispersion and maintains connectivity

- **directedDispersion**
  - Effectively push frontiers to boundaries
  - Robots rarely head in wrong direction
Table 1. Dispersion Efficiency vs. Location
Other Experiments

- A swarm of 108 robots dispersed into 3000 ft$^2$ of indoor space in about 25 mins, located an object of interest, and led human to it.
- Multiple room configurations were tested.
Third Paper

- Research work done @ Harvard
  - http://www.eecs.harvard.edu/ssr/

- Programmable self-assembly in thousand-robot swarm
  - https://www.youtube.com/user/ssrlab0
Key Idea

- Self-assembly enables nature to build complex forms, through interaction of vast numbers of limited and unreliable individuals

- Creating this ability in engineered systems
  - Large groups of autonomous robots
  - Collective algorithms
Fig. 1 Kilobot swarm robot. (A) A Kilobot robot, shown alongside a U.S. penny for scale.

Michael Rubenstein et al. Science 2014;345:795-799

Published by AAAS
Fig. 2 Collective self-assembly algorithm. Top left: A user-specified shape is given to robots in the form of a picture.

The desired shape is given to all robots in the form of a binary bitmap. Four pre-localized seed robots (green) define the origin and orientation of the coordinate system.

The desired shape is aligned with the coordinate system and scaled by the input parameter ‘s’.

Edge-following
A robot (red) moves by maintaining a fixed distance ‘d’ to the center of the closest stationary robot (green).

Gradient formation
Each robot sets its gradient value to 1 + the minimum value of all neighbors closer than distance ‘g’. The source robot (green) maintains a gradient value of 0.

Localization
A robot (blue) determines its position in the coordinate system by communicating with already localized robots (green).

\[ (x_4, y_4) = \min_{x_i, y_i} \left( \sum_{i=1}^{3} (d_{(i,4)} - \alpha_i) \right) \]

where \( \alpha_i = \sqrt{(x_i - x_4)^2 + (y_i - y_4)^2} \)

Self-assembly algorithm

Starting position of the initial group (blue robots), and seed robots (green). Gradient value displayed on robots.

First edge-following robot enters desired shape, as determined by its location in the coordinate system.

Second robot stops and joins the assembly when next to a stationary robot with the same gradient value.

Completed shape after all robots have joined the assembly, with numbers showing the order in which robots joined.

Michael Rubenstein et al. Science 2014;345:795-799
Fig. 3 Self-assembly experiments using up to 1024 physical robots. (A, C, and E) Desired shape provided to robots as part of their program.
Summary
Announcement

- Project 2 assigned!