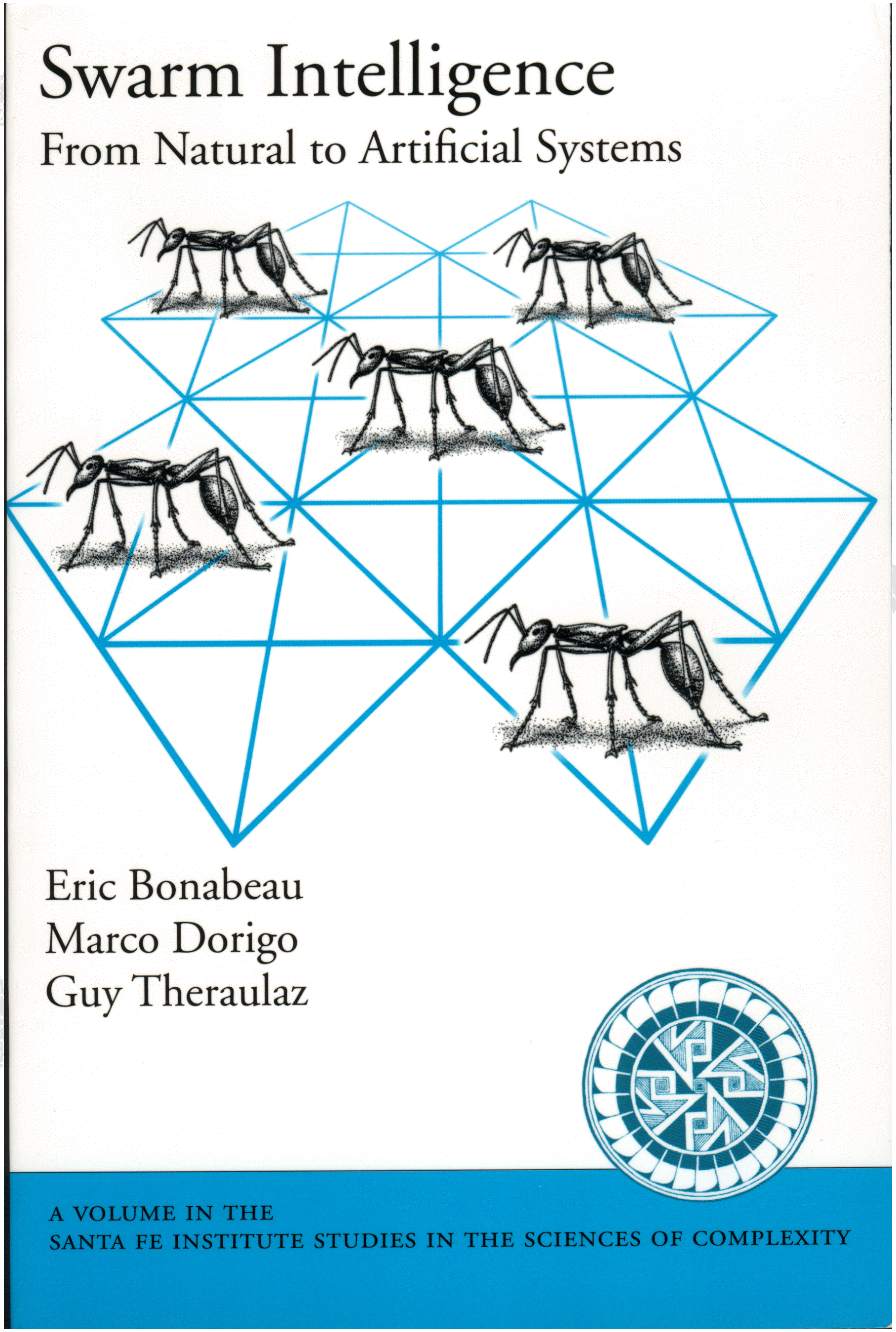
Group Robotics

- Related topics have been or will be discussed:
    - Neural networks
    - Classical conditioning
    - AHC with NNs
    - Genetic Algorithms
    - Classifier Systems
    - Fuzzy learning
    - Case-based learning
    - Memory-based learning
    - Explanation-based learning

Swarm Intelligence

Natural and artifficial



Swarm Intelligence

From Natural to Artificial Systems

Eric Bonabeau
Marco Dorigo
Guy Theraulaz

A VOLUME IN THE
SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY

Ants in the Pants!

An Overview

- **Real world insect examples**
- ***Theory* of Swarm Intelligence**
- **From Insects to Realistic A.I. Algorithms**
- **Examples of AI applications**

Real World Insect Examples

Bees

Bees

- Colony cooperation
- Regulate hive temperature
- Efficiency via Specialization:
  - division of labour in the colony
- Communication :
  - Food sources are exploited according to quality and distance from the hive

Wasps

Wasps

- Pulp foragers, water foragers & builders
- Complex nests
  - Horizontal columns
  - Protective covering
  - Central entrance hole

Termites

Termites

- Cone-shaped outer walls and ventilation ducts
- Brood chambers in central hive
- Spiral cooling vents
- Support *pillars*

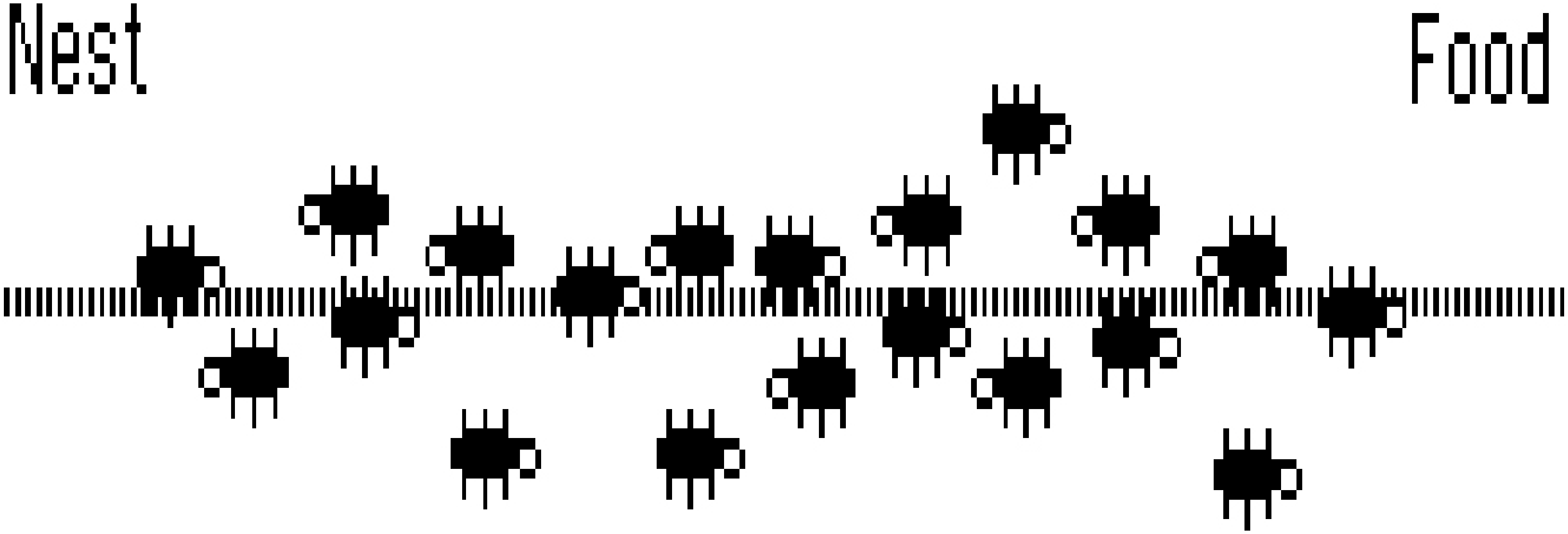
Ants

Ants

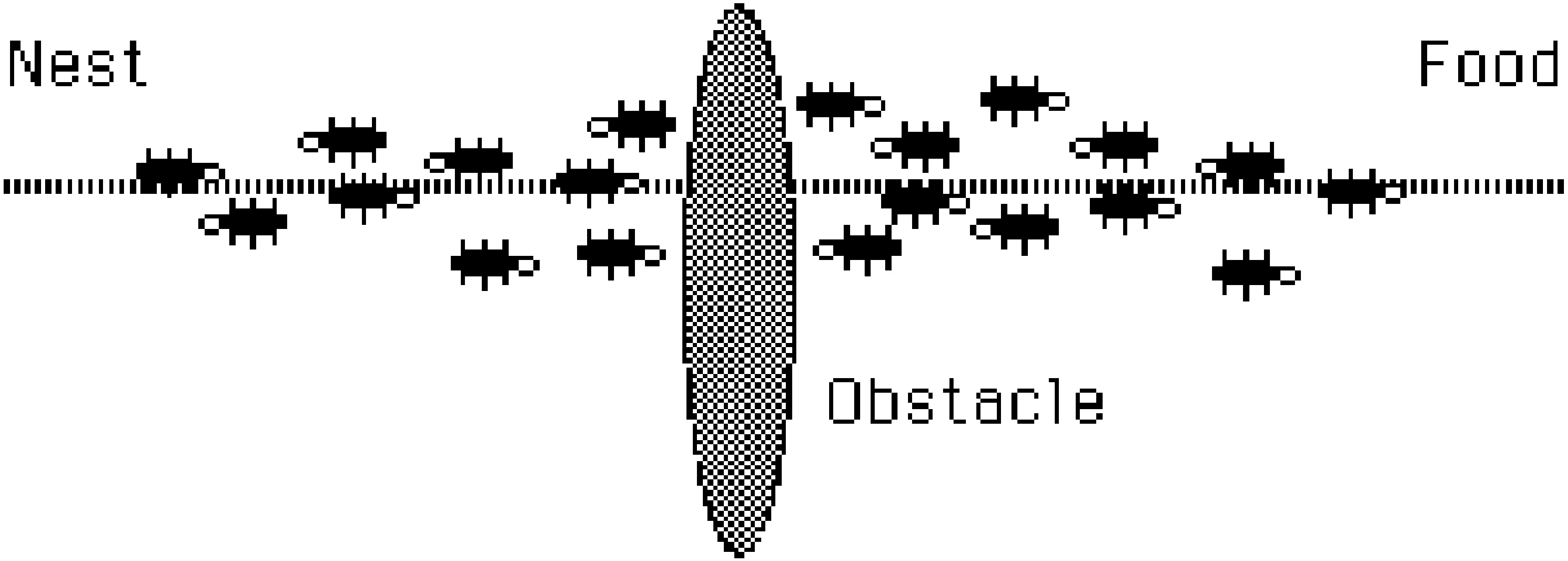
- **Organizing highways to and from their foraging sites by leaving *pheromone trails***
- **Form *chains* from their own bodies to create a bridge to pull and hold leafs together with silk**
- ***Division of labour* between major and minor ants**

Swarm Intelligence in Theory

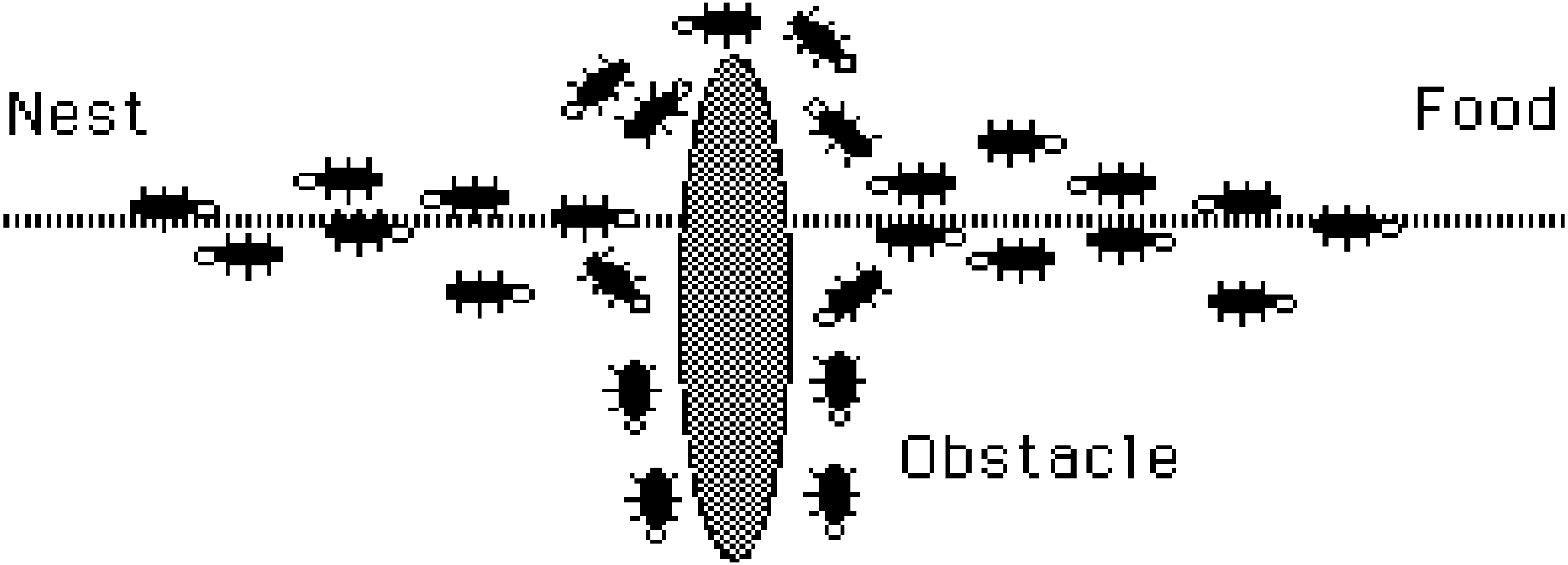
An In-depth Look at Real Ant Behaviour



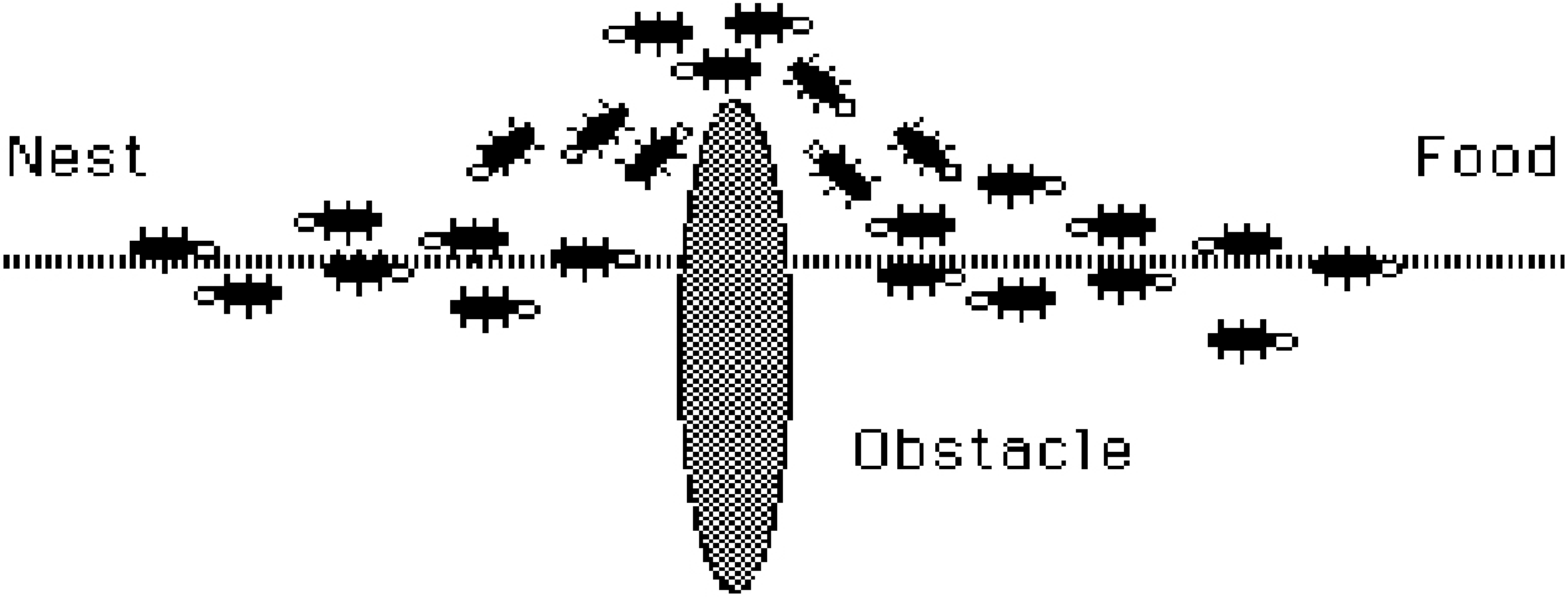
Interrupt The Flow



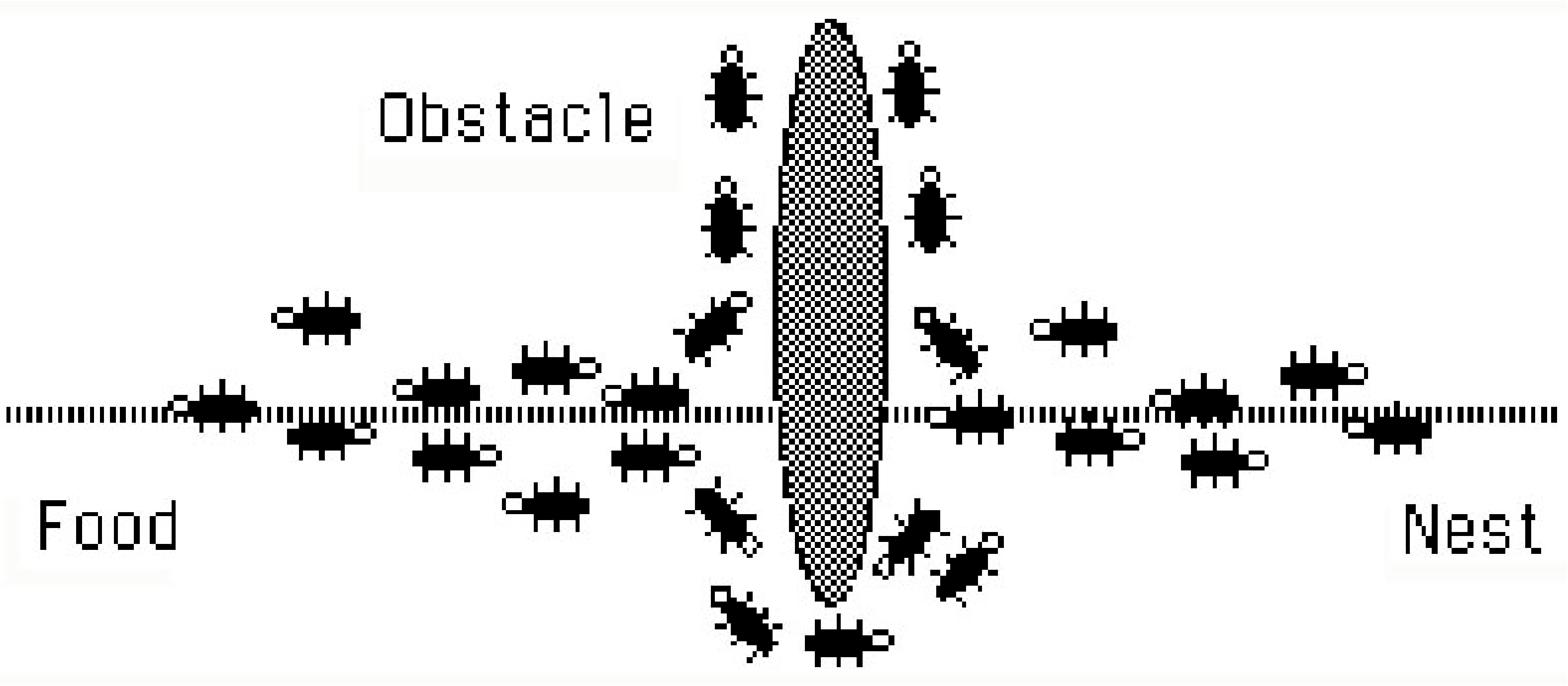
The Path Thickens!



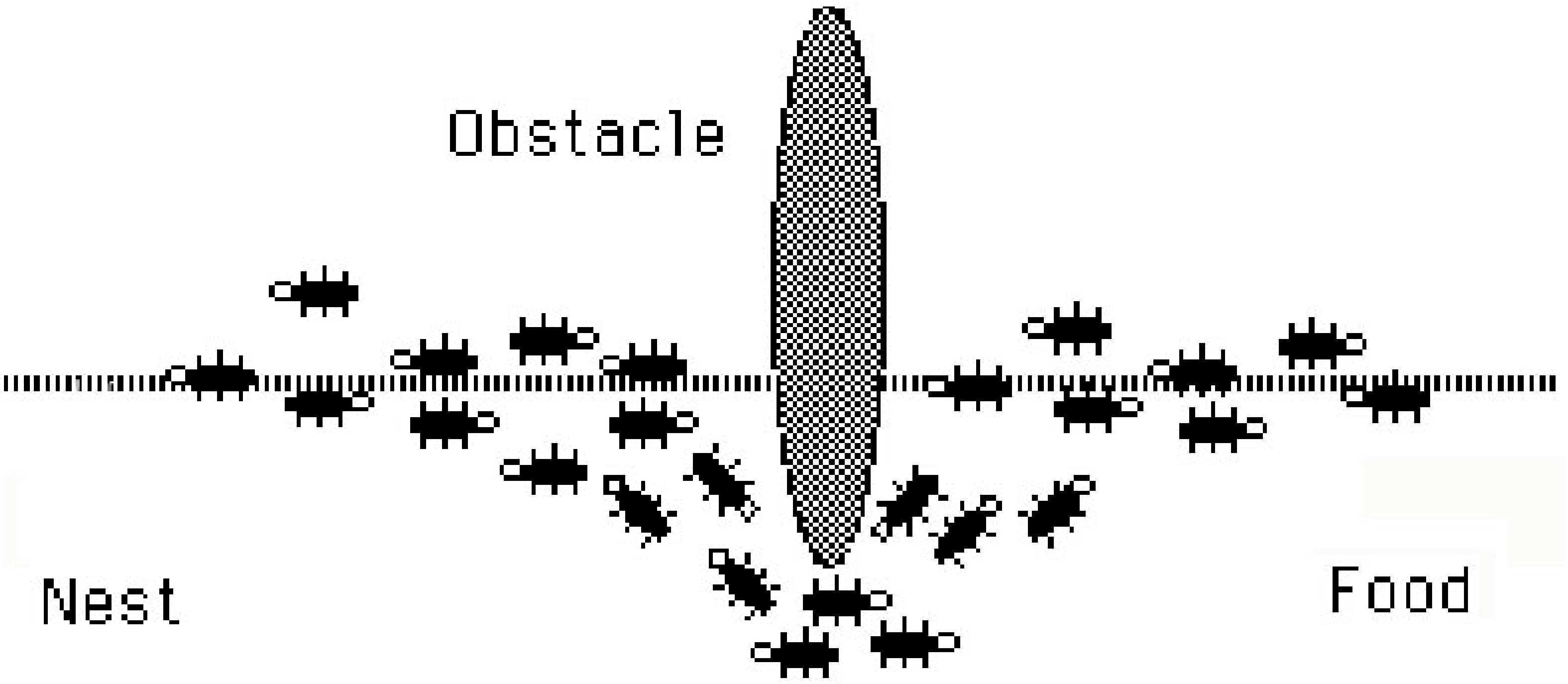
The New Shortest Path



Adapting to Environment Changes



Adapting to Environment Changes



Ant Pheromone and Food Foraging Demo

Social Insects

- Problem solving benefits include:
  - Flexible
  - Robust
  - Decentralized
  - Self-Organized

Summary of Insects

- The complexity and sophistication of Self-Organization is carried out with no clear leader
- What we learn about *social insects* can be applied to the field of Intelligent System Design
- The modeling of social insects by means of Self-Organization can help design artificial distributed problem solving devices.
- This is also known as Swarm Intelligent Systems.

Terminology

- Various rather interchangeable terms are used in this area:
  - Group behavior / robotics
  - Collective behavior / robotics
  - Cooperative behavior / robotics
  - Swarm robotics
  - Multi-robot systems
- Some terms imply larger sizes and/or more or less deliberative approaches;
  - for now the differences can be ignored

Benefits of Group Solutions

- Using multiple robots to solve certain tasks can provide great benefits, which include:
  - Improved system performance (usually in terms of speed of completion)
  - Improved task enabling
  - Distributed sensing
  - Distributed action at a distance
  - Fault tolerance through redundancy

Negatives of Group Solutions

- The benefits come with a price:
  - Interference between robots
  - Communication cost and robustness
  - Uncertainty regarding other robots' intentions
  - Overall *system cost*

Types of Collective Systems

- **Merely Coexisting:** multiple robots coexist in a shared environment, but do not even recognize each other, merely as obstacles
    - + no need for coordination
    - - increased group size results in uncontrolled interference
- **Loosely Coupled:** multiple robots share an environment and sense each other and may interact, but ...
  - … do not depend on one another; members of the group can be removed without significant effect
    - + robust
    - - difficult to coordinate for precise tasks
- **Tightly Coupled:** multiple robots cooperate on a precise task, usually by using communication, turn-taking, and other means of tight coordination
    - - depend on each other

Example Domains

- Mere coexistence
  - foraging
- Loosely coupled
  - foraging
  - collection
  - distributed mapping
- Tightly coupled
  - formations
  - moving objects

Competitive Domains

- Besides cooperation there is also ***competition***
- **Game scenarios** are a good challenge for developing group robotics
  - robot soccer, the grand AI challenge
- **Real world scenarios** have competitive elements:
  - robots are always competing for space,
  - interference.

Interference

- Robots can interfere with each other at different levels
  - physical interference
  - competition for physical resources, like space
- task interference:
  - competition for task resources, like objects
  - competition for winning resources, like goals, pieces, etc.

Control Approaches

- How can we control a group of robots?
- Two basic options exist:
  - centralized control
  - distributed control
- These are two ends of the control spectrum.
- There are numerous compromises:
  - hierarchical control

Types of Control

Centralized Control

- A single, centralized controller takes the information about all of the robots as input, and outputs the actions for all of them.
- There are many problems:
  - requires a lot of information
  - requires global communication
  - it is slow to plan for many agents (global state space is huge)
  - depends on the centralized controller
- Centralized control
  - creates a bottleneck
  - scales very badly with increased group sizes
  - is very slow
  - is not robust
- But there is one major advantage: the approach allows us to compute optimal solutions (at least in theory) at the group level

Distributed Control

- Each robot uses **its own controller** to decide what to do.
  - **There are many advantages :**
    - no information needs to be gathered
    - communication can be minimized or avoided (no bottle-neck)
    - robots or sub-groups can fail
    - group size can change dynamically
    - scales well with increased group size
    - individuals can adapt and improve

Distributed Control

- Each robot uses its own controller to decide what to do.
  - **There are many advantages :**
    - no information needs to be gathered
    - communication can be minimized or avoided (no bottle-neck)
    - robots or sub-groups can fail
    - group size can change dynamically
    - scales well with increased group size
    - individuals can adapt and improve
  - But there is a key disadvantage
  - Distributed control requires that the desired group-level collective behavior be produced in a decentralized, non-planned fashion from the interactions of the individuals
  - Designing individual/local behaviors for each robot that result in the desired group/global behavior is a VERY hard problem

Deliberative Group Control

- There are only 4 types of control arch’s
- Those are suitable for different types of group behaviors (how?)
- Deliberative systems are well suited for the centralized approach
  - the single controller (on or off a robot) performs the standard SPA loop:
    - gathers the sensory data,
    - uses it all to make a plan for all robots,
    - sends the plan to each robot,
    - and each executes it

Hybrid Group Control

- Hybrid systems are also well suited for the centralized approach, but can be used in a distributed fashion as well;
  - the centralized controller (on or off a robot) performs the SPA loop,
    - individual robots monitor their sensors, and
    - update the planner with any changes, so that a new plan can be generated when needed
  - each robot can run its own hybrid controller,
    - but it needs info on all others to plan;
    - synchronizing the plans is hard

Reactive Group Control

- Reactive systems are well suited for implementing the distributed approach;
  - each robot executes its own controller, and can communicate and cooperate with others as needed
  - the group-level behavior emerges from the interaction of the individuals

Behavior-Based Group Control

- Behavior-based systems are well suited for implementing the distributed approach;
  - each robot behaves according to its own, local behavior-based controller
  - each robot can also learn over time and display adaptive behavior
  - as a result, the group-level behavior can also be improved and optimized

Hierarchies in Groups

- Hierarchical approaches can be implemented with any of the controllers
  - Fixed hierarchies can be generated by a planner within a deliberative or hybrid system
  - Dynamic, changing, adaptive hierarchies can be formed by behavior-based systems
  - Reactive distributed multi-robot systems can also form hierarchies, either by pre-programming, or dynamically (e.g., based on size, color, ID number, etc.).

Challenges

- Controlling groups of robots is even more difficult than controlling one robot, because:
  - the environment is inherently dynamic
  - there are more interactions to consider
  - there is more uncertainty in the system

- **Group Behavior Approaches**
  - Ethological
  - Organizational behavior
  - Computational models
  - Distributed AI
  - Motion planning
  - Artificial life
- **Prototypical Group Tasks**
  - Foraging
  - Consuming
  - Grazing/coverage
  - Formations/flocking
  - Object transport

Foraging and Ethological Models

Why Foraging?

- Foraging is a prototype for a large variety of real-world applications of group robotics:
  - locating and disabling/marking land mines
  - distributed mapping of the area
  - collectively distributing objects (markers, cables, seeds, etc.)
  - collective reconnaissance
  - collective surveillance
  - and many more...

Ethological Models

- **Simple social behavior types**
  - antagonistic
  - reciprocal
  - sympathetic induction
- **Mating behaviors**
  - persuasion/appeasement
  - orientation/approach
- **Family/group behaviors**
  - flocking/herding (defense-related)
  - congregation
  - Infectious: alarm/sleep/eating
- **Fighting behaviors**
  - reproductive
  - mutual hostility
    - pecking order

Ethological Models (cont)

- **Characteristics**
    - Reliability
    - Organization
    - Communication
    - Spatial distribution
    - Congregation
    - Performance
- **Example Taxonomy**
    - Team size
    - Communication range
    - Communication topology
    - Communication bandwidth
    - Team reconfigurability
    - Team unit processing ability
    - Team composition

- **Example:** CEBOT
    - The original example of reconfigurable teams
    - Cellular Robot (CEBOT); Japan
- **Example**: Nerd Herd
    - **Nerd Herd:** a collection of 20 coodinated small wheeled robots (Mataric 1994, MIT/Brandeis/USC) (video)
    - **Basis behaviors:** homing, aggregation, dispersion, following, safe wandering
    - Organized in **Subsumption** style
    - **Complex aggregate behaviors:** flocking, surrounding, herding, docking
    - Complex behaviors result from **combinations or sequences** of basis set

- **Example:** Alliance
    - L. Parker MIT/ORNL
    - Heterogeneous teams
    - Adds layer to subsumption for switching behavioral sets
    - Uses impatience and acquiescence for team coordination
    - Tasks include box-pushing, janitorial service, hazardous waste clean-up, bounding overwatch
- **Example:** Stagnation
    - R. Kube and Zhang - U of Alberta
    - Stagnation occurs when cooperation is poor
    - Arbitrates between multiple strategies to recover when detected

Box Pushing Task

- Arbitrary object geometry
- Arbitrary numbers of robots
- Arbitrary initial configuration
- Homogeneous or heterogeneous teams
- Different approaches to communication
  - no explicit communication
  - minimal communication
  - global communication (broadcast)

Types of Pushing Tasks

- Homogeneous:
  - collection of wheeled robots
  - a pair of 6-legged robots
- Heterogeneous:
  - wheeled and legged
  - different types of sensors
- Applications
  - removing barriers
  - help in disaster scenarios
  - moving wounded

- **Implementations**
  - Examples:
    - MIT (Parker, Mataric video),
    - Cornell (Donald et al video),
    - Alberta (Kube)
- **Communication**
  - Provides synchronization of action
  - Information exchange
  - Negotiations
  - Communication not essential for cooperation
  - Louder not necessarily better

Old lecture notes, for reference

- Everything we have covered so far has dealt with the control of a single robot.
- Today we will *scale up* to the problem of *multi-robot* control.
  - i.e. the challenge of generating
    - coherent,
    - robust, and
    - reliable behavior

with more than one robot co-existing in the same environment.

Old lecture notes, for reference

- There are several different forms of *multi-robot* systems.
- In some, the robots merely exist in the same environment, but do not even detect each other as robots, but merely obstacles.
- This is the simplest form, and is the least efficient: the more robots there are, the less effective the system is, since the robots must avoid collisions with each other.

- For example, we might have a group of foraging robots, whose job is to look over a field for scattered objects, pick them up, and *bring them back to some deposit point.*
- At the same time, the robots avoid other robots or any other obstacles.
- You can see how the more robots are introduced, the more potential *interference* there is between them.
- In this approach, the robots do not help each other or even recognize each other, and there is a sensitive relationship between:
    - their task (including the size of the space and the number of objects they are foraging for),
    - physical size,
    - sensor range,
    - behavior, and
    - number,

for getting the task done efficiently

Loosely coupled

- In more sophisticated *multi-robot* systems, multiple (i.e., two or more) robots co-exist in the same environment.
- They are aware of each other.
- They are *loosely coupled* in that they *do not depend on each other* for completing the task.
- This means they can react to each other in more interesting ways than just avoidance.
- But *they do not directly help* each other.

- For example, take the same foraging robot scenario.
- Now, instead of treating each other as obstacles, the robots can actually react to each other in more interesting ways.
- Such as:
  - following a robot that has an object, in hopes that it would lead toward more objects.
- Or avoiding a crowd of robots in the assumption that the objects in that area will have already been picked up.
- Or flocking with other robots that are heading to the deposit point.

Sophisticated multi-robot systems

- In even more sophisticated multi-robot systems, multiple robots ***actively cooperate*** with each other.
- If the robots *depend on each other*, their organization can be said to be *tightly coupled*.
  - For example, consider two or more robots that need to move a large object to some location.
  - (like ***ants*** having to move food or another, larger, dead ant)
- If the object is too heavy for one robot, cooperation is necessary.
- Furthermore, *coordination* is necessary, since it is not simply enough to have all robots randomly pushing, they must be sufficiently coordinated to make *joint progress*.

Competitive Robot Systems

- Since two robots cannot be in the same place at the same time, there is always some potential interference between robots.
- But besides the fundamental spatial interference that is unavoidable in physical robots, there are other kinds of ***interference*** that appear in multi-robot systems as well.
- A more sophisticated kind of interference has to do with the robot's goals:
  - one robot can ***undo*** the work of another, if their goals are conflicting.
- It turns out that it is quite a difficult problem to come up with a multi-robot (or even non-robotic multi-agent) system that *minimizes interference*

Competitive Robot Systems

- Finally, multi-robot systems can be ***competitive***.
- It is easy to imagine how two or more robots may compete in some kind of a game scenario
  - (such as robot soccer or a contest like the one you will have at the end of the semester).
- It is more interesting to realize that in any multi-robot situation, there is an element of competition:
  - in any such situation, the robots are competing for at least one *common or shared resource*, i.e, physical space.

How might we control a group of robots?

- We can consider two ends of the control spectrum, and then think of what falls in between:
  - **1)** A single, centralized, controller can be used, which takes the information about all of the robots as input, and outputs the actions for all of them.
  - There are *many problems* with this approach because:
    - a) it requires a great deal of information to be gathered
    - b) it requires the information to be communicated to and from the robots and the centralized controller (i.e., the controller is a *bottle-neck)*
    - c) it can be very slow to plan for so many agents, because the global state space is exponential

  - The key potential advantage of the *centralized approach* is that it allows us to use search to generate optimal solutions for the group as a whole, assuming we have enough time and information for that computation.
- 2) Each robot can **use its own controller** to decide what to do.
  - There are many advantages of this approach over #1 above:
    - a) no information needs to be gathered between robots
    - b) communication can be minimized or avoided (no bottle-neck)
    - c) the environment can change and each agent can adapt, because it is not a part of a global plan
    - d) the group size can change dynamically (i.e., robots can fail or new ones can be added) without the need to re-plan

  - The key disadvantage of the ***distributed* approach** is the difficult challenge of designing individual/local behaviors for each robot which will result in the desired group/global behavior.
- Between the two ends of the spectrum, one can employ hierarchies between the robots.
- In these hierarchies;
  - the dominant individuals may make decisions (and use planning), while the others do not,
  - where the task is divided between the individuals in unequal ways (note that the division can be done at compile-time or at run-time),
  - etc.

- Given the above, consider what control architectures lend themselves to the multi-robot control problem:
  - **1)** *Deliberative* systems are well suited for the centralized approach;
    - in them, the centralized controller (on a robot or in some other location) performs the standard SPA loop:
      - it gathers the sensory data,
      - uses it to form a plan for each robot,
      - sends the plan to each robot, and executes it.
  - **2)** Hybrid systems are also well suited for the centralized approach;
    - in them, the centralized controller (on a robot or in some other location) also performs the SPA loop,
    - but the individual robots monitor their sensors and effectors, and update the planner with any changes,

      so that a new plan can be generated when needed.

- **3)** Reactive systems are well suited for implementing the distributed approach,
  - in which each robot executes its own controller, and can communicate and cooperate with others *as needed*.
- **4)** Behavior-based systems are well suited for implementing the distributed approach, as are reactive systems,
  - but they also enable the individuals to learn over time and display adaptive behavior at the local and global level.

- Once again, we have considered the *ends of the control spectrum* (centralized and distributed).
- Hierarchical approaches can also be implemented with any of the above controllers:
  - fixed hierarchies can be generated by a planner within a deliberative or hybrid system,
  - while dynamic, changing, adaptive hierarchies can be formed by behavior-based

- Reactive distributed multi-robot systems can also form hierarchies,
  - either by pre-programming, or dynamically at run time,
  - by simply reacting to each other's sensed or communicated properties

    (such as size, color, ID number, etc.).

- Given the various control alternatives, it may seem quite simple to control a multi-robot system.
- However, it is just the opposite:
  - as more robots are introduced, the control problem becomes more difficult, because:
    - 1) the environment is dynamic
    - 2) there are more interactions to consider
    - 3) there is more uncertainty in the system

Dynamic environment:

- Multi-robot systems are, by definition, situated in dynamic environments.
- As we have seen, changes in the environment make the robot's world more challenging, because it is less predictable.
  - The more novel interactions the robot has, the more challenging its world is.
- (Now imagine if the robots in the group can actually adapt their behavior over time, i.e., learn.
  - This makes the environment even more dynamic.
  - However, it makes the system more interesting and potentially more robust.
  - We will talk about learning next time.)

Interaction:

- As you have seen so far, interactions between the robot and the world are complex, but can be used to generate interesting behavior.
- The same is even more true for multi-robot systems, where all kinds of interactions can happen between robots (symmetric movements, reciprocation, competition, cooperation, etc.).
- But those must be carefully characterized, well understood, and only the desirable ones must remain in a well-designed controller.
- As you might imagine, multi-robot systems can generate a great deal of emergent behavior, which can be used to the designer's advantage or disadvantage.

Uncertainty

Uncertainty:

- As you have seen so far, behavior in the physical world is fraught with ***uncertainty.***
- This is due to:
  - intrinsic sensor and effector noise,
  - locality/partial observability, etc.
- This uncertainty makes reliable behavior difficult to achieve on a single robot, and it grows with the size of the group, since each robot itself has its own level of uncertainty, and each interaction between two or more robots produces uncertainty as well.
- Thus, it is known to be theoretically impossible to produce ***totally predictable*** group behavior in multi-robot systems.
- In fact, it is a lost cause to attempt to prove or guarantee where each robot will be and what it will do after the system is running.

Uncertainty:

- However, this does not mean that multi-robot system behavior is random.
- Far from it, we can program our robots so that it is possible to characterize, even prove, *properties or behaviors of the group*, i.e., ***ensemble-level properties*** or ***collective behavior***, rather than the behavior of individuals.
- This is a powerful method for describing and verifying multi-robot systems.

Problems Regarding Swarm Intelligent Systems

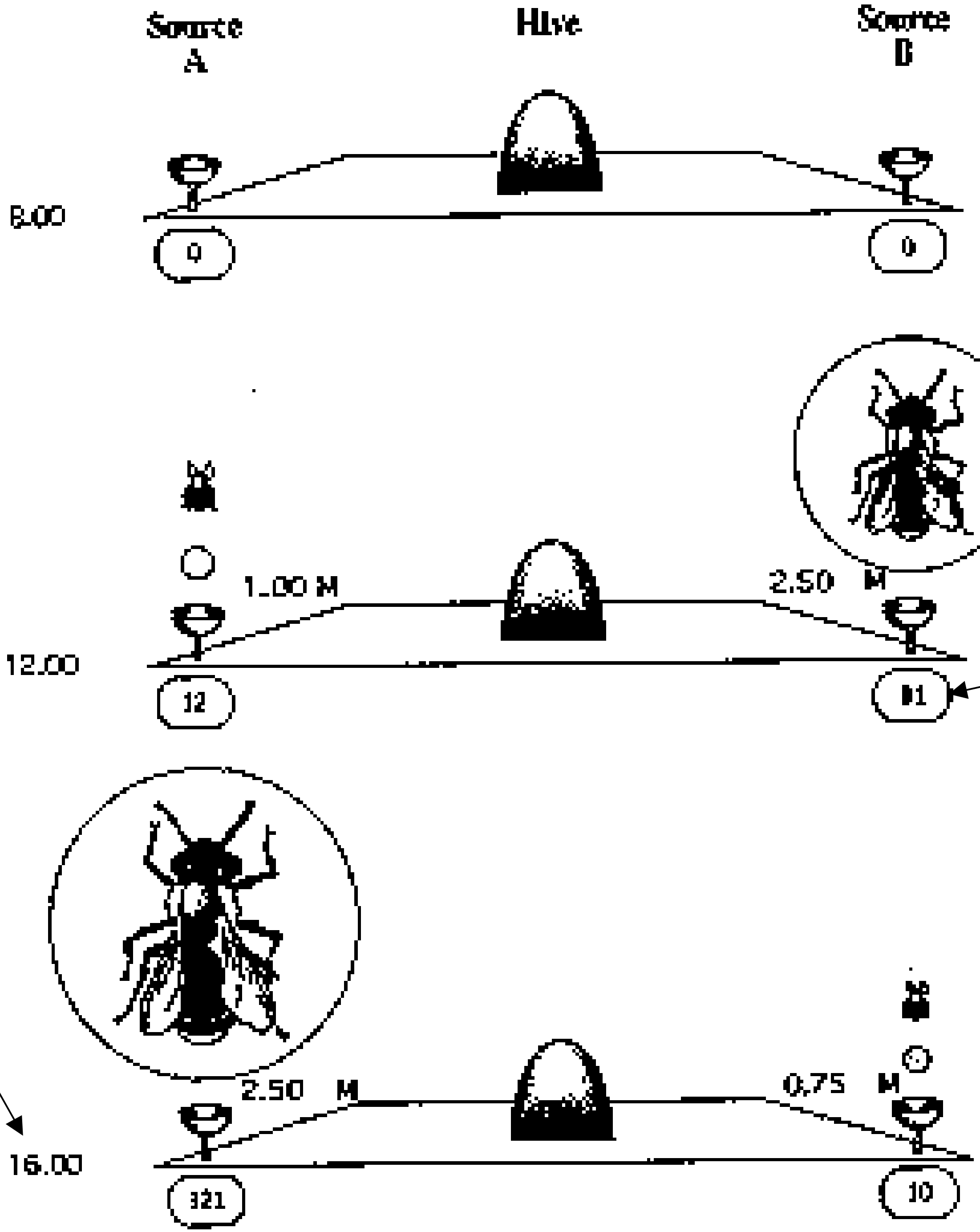
- Swarm Intelligent Systems are hard to 'program' since the problems are usually difficult to define
  - Solutions are emergent in the systems
  - Solutions result from behaviors and interactions among and between individual agents

Possible Solutions to Create Swarm Intelligence Systems

- Create a catalog of the collective behaviours (Yawn!)
- Model how social insects collectively perform tasks
  - Use this model as a basis upon which artificial variations can be developed
  - Model parameters can be tuned :
    - within a biologically relevant range
    - or by adding non-biological factors to the model

Four Ingredients of Self Organization

- Positive Feedback
- Negative Feedback
- Amplification of Fluctuations - randomness
- Reliance on multiple interactions


Source A
Hive
Source B
8.00
0
0
12.00
1.00 M
2.50 M
12
81
16.00
2.50 M
0.75 M
321
10

total

time

Properties of Self-Organization

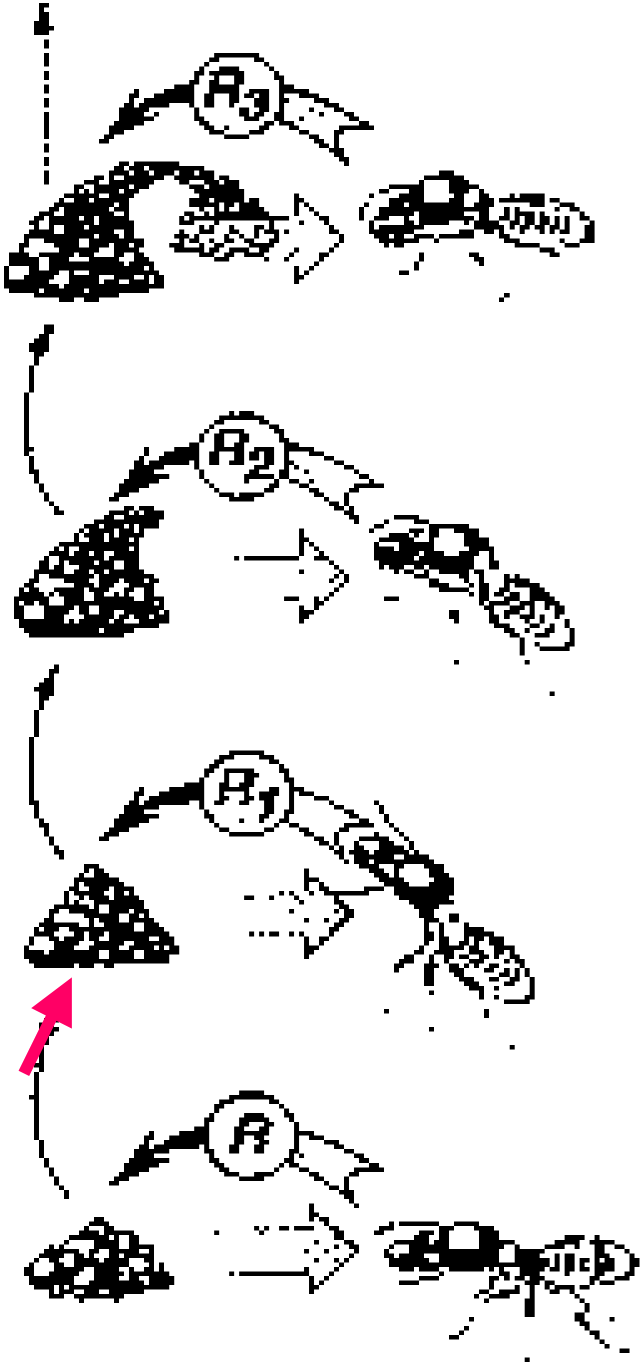
- Creation of structures
  - Nest, foraging trails, or social organization
- Changes resulting from the existence of multiple paths of development
  - Non-coordinated & coordinated phases
- Possible coexistence of multiple stable states
  - Two equal food sources

Types of *Interactions* For Social Insects

- Direct Interactions
  - Food/liquid exchange, visual contact, chemical contact (pheromones)
- Indirect Interactions (Stigmergy)
  - Individual behavior modifies the environment, which in turn modifies the behavior of other individuals

Stigmergy Example

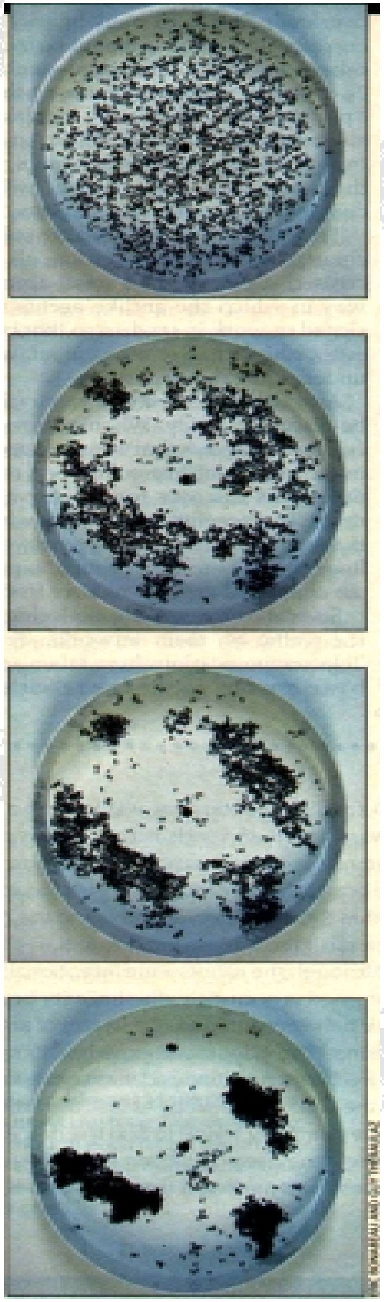
- **Pillar construction in termites**



Stigmergy

in

Action



Ants ≡ Agents

- Stigmergy can be operational
  - Coordination by indirect interaction is more appealing than direct communication
  - Stigmergy reduces (or eliminates) communications between agents

From Insects to Realistic A.I. Algorithms

From Ants to Algorithms

- Swarm intelligence information allows us to address modeling via:
  - Problem solving
  - Algorithms
  - Real world applications

Modeling

- Observe Phenomenon
- Create a biologically motivated model
- Explore model without constraints

Modeling...

- Creates a simplified picture of reality
- Observable relevant quantities become variables of the model
- Other (hidden) variables build connections

A Good Model has...

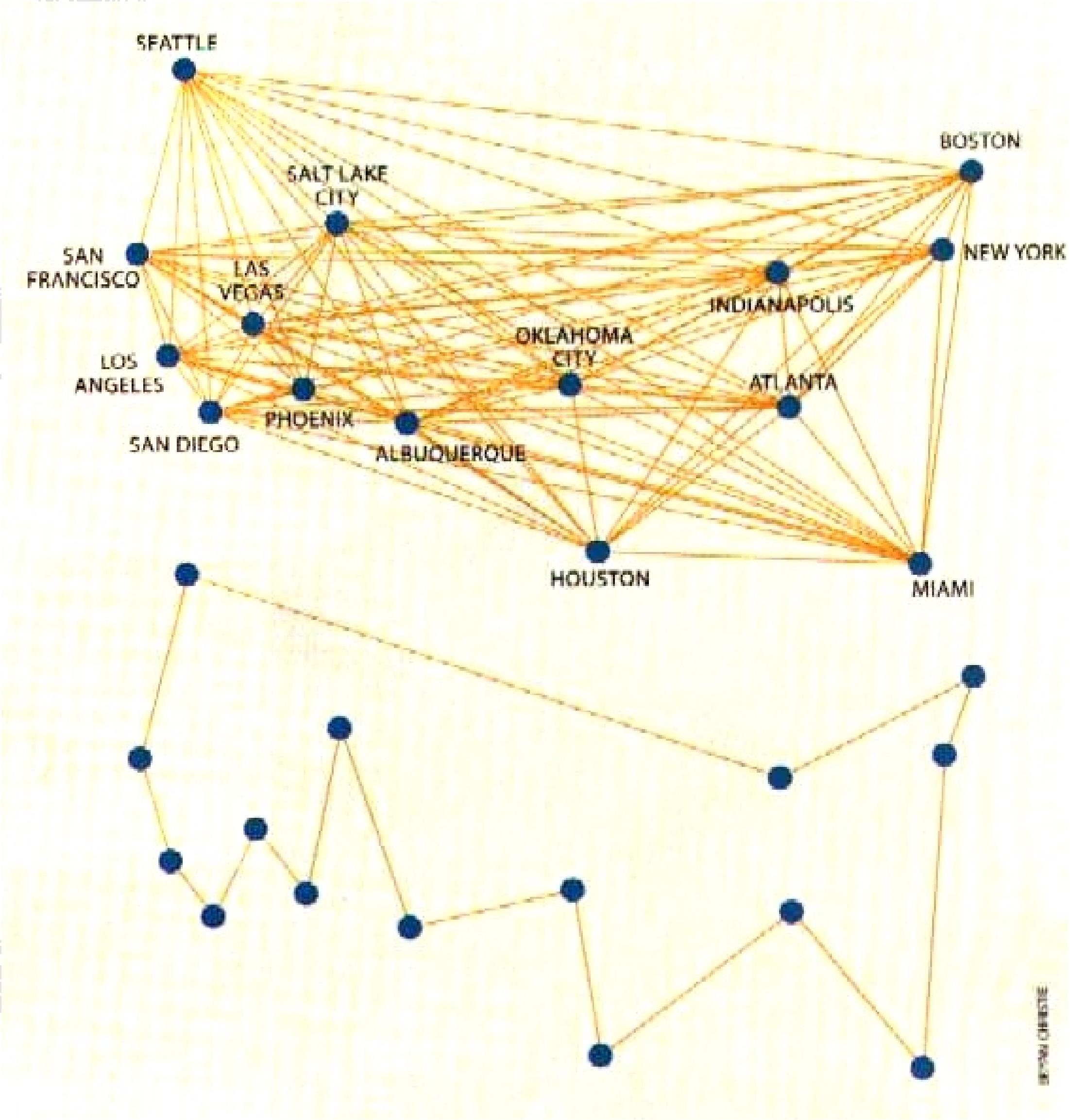
- Parsimony (simplicity)
- Coherence
- Refutability
- Parameter values correspond to values of their natural counterparts

Travelling Salesperson Problem

```
Initialize
    Loop /* at this level each loop is called an iteration */
    Each ant is positioned on a starting node
        Loop /* at this level each loop is called a step */
        Each ant applies a state transition rule to incrementally
        build a solution and a local pheromone updating rule
    Until all ants have built a complete solution
A global pheromone updating rule is applied
Until End_condition
```

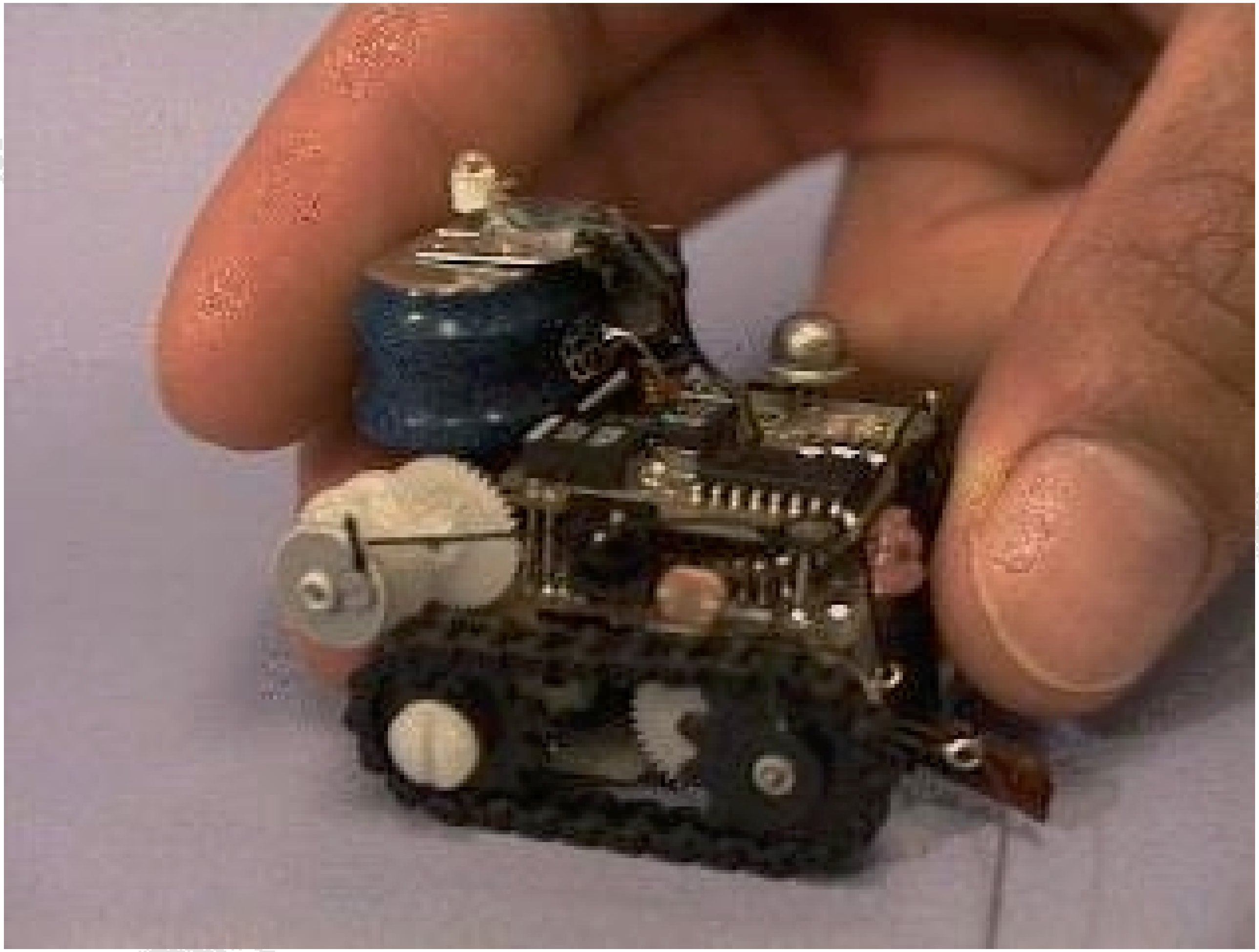
M. Dorigo, L. M. Gambardella : ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.16-TEC97.US.pdf
Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem

Traveling Sales Ants



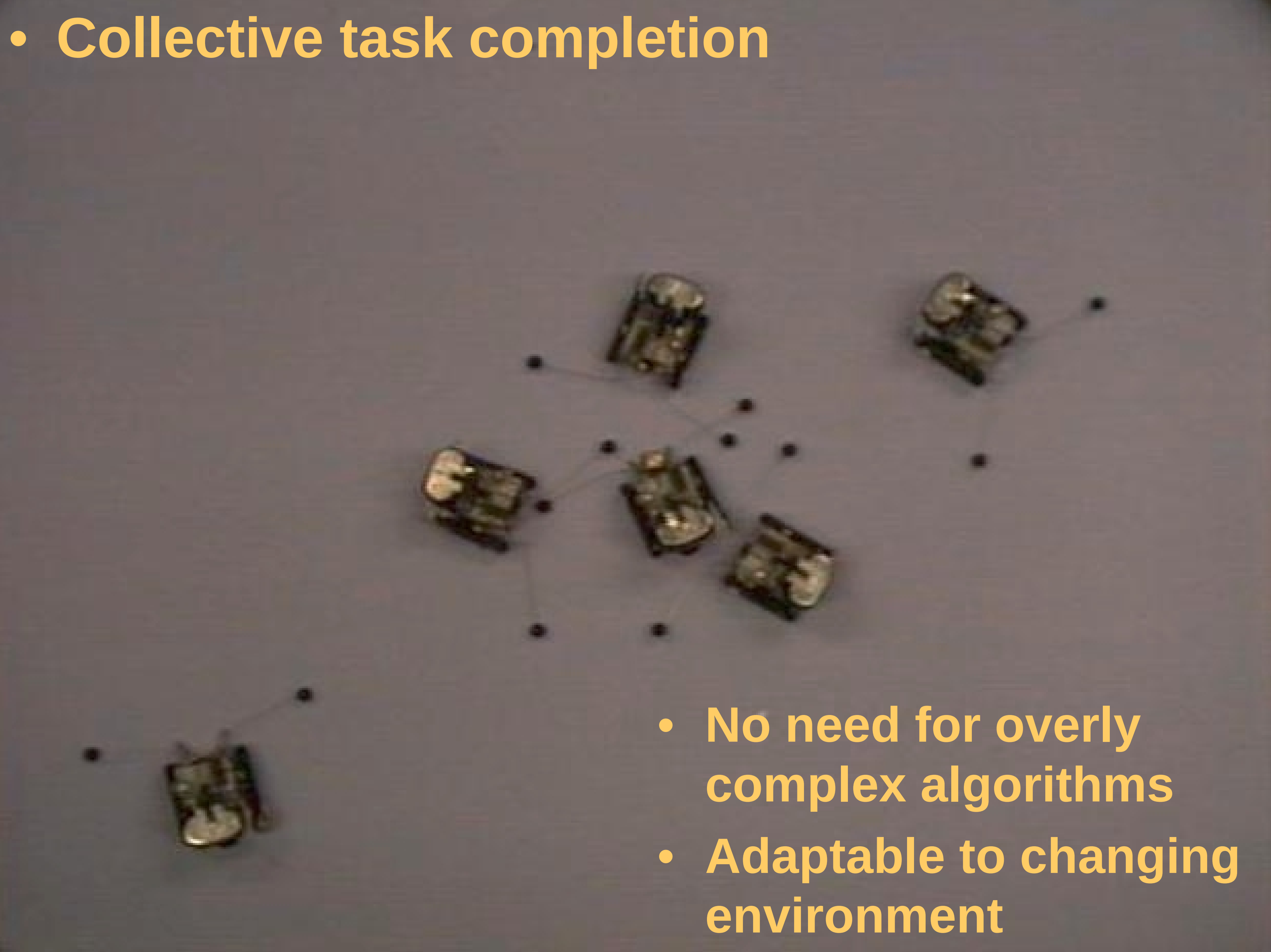
Welcome to the Real World

Robots



These techniques have been applied to groups of small robots

- **Collective task completion**



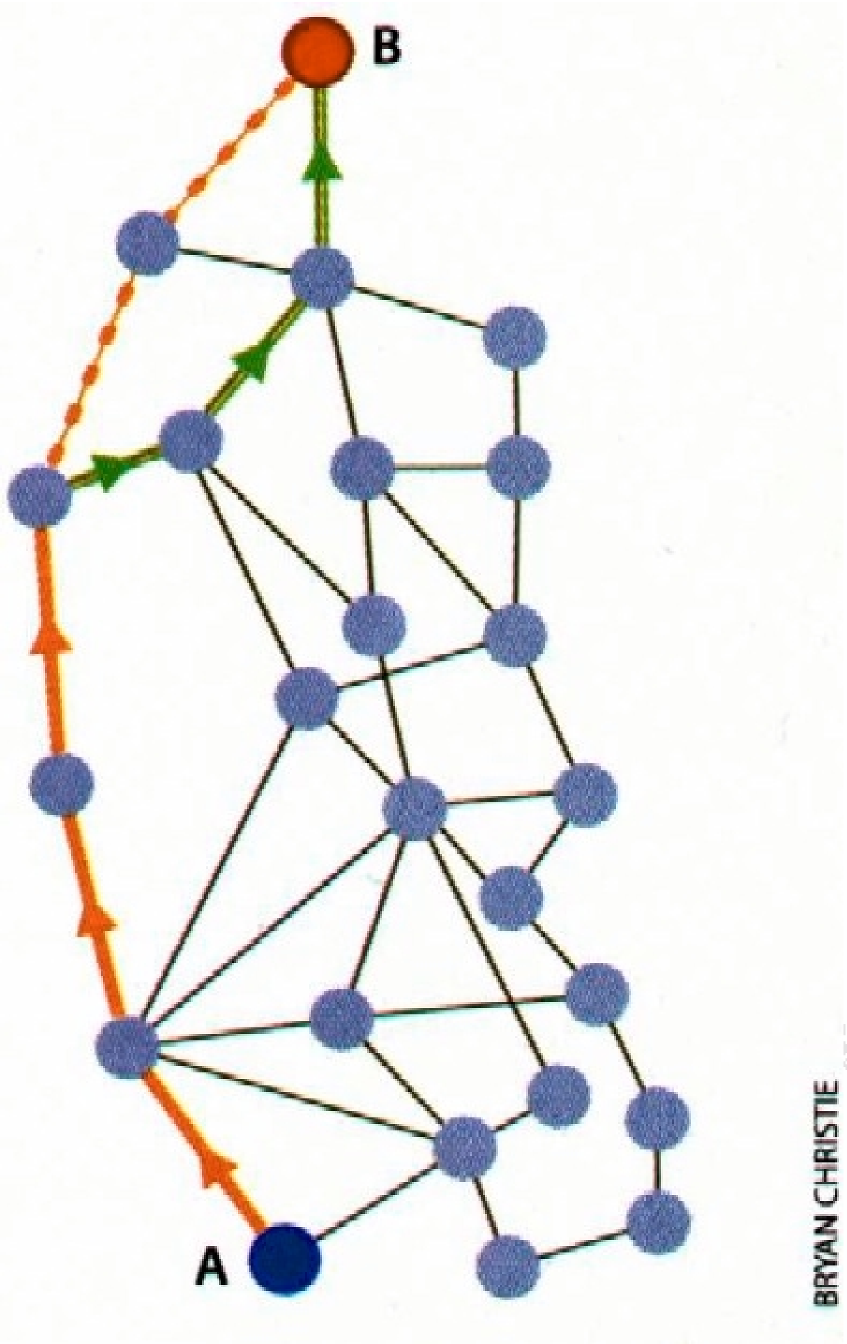
- **No need for overly complex algorithms**
- **Adaptable to changing environment**

Robot Feeding Demo

Communication Networks

- Routing *packets to destination* in shortest time
- Similar to Shortest Route
- Statistics kept from prior routing (learning from experience)

- **Shortest Route**
- **Congestion**
- **Adaptability**
- **Flexibility**



Antifying Website Searching

- Digital-Information Pheromones (DIPs)
- Ant World Server
- Transform the web into a gigANTic neural net

Closing Arguments

- **Still very theoretical**
- **No clear boundaries**
- **Details about inner workings of insect swarms**
- **The future…???**

Dumb parts, properly connected into a swarm, yield smart results.

Kevin Kelly

The Future?

Miniaturization

Telecommunications

Pipe Inspection

Cleaning Ship Hulls

Medical

Self-Assembling Robots

Satellite Maintenance

Engine Maintenance

Job Scheduling

Pest Eradication

Combinatorial Optimization

Data Clustering

Interacting Chips in Mundane Objects

Vehicle Routing

Optimal Resource Allocation

Distributed Mail Systems

References

Ant Algorithms for Discrete Optimization Artificial Life
M. Dorigo, G. Di Caro & L. M. Gambardella (1999).
addr:http://iridia.ulb.ac.be/~mdorigo/

Swarm Intelligence, From Natural to Artificial Systems
M. Dorigo, E. Bonabeau, G. Theraulaz

The Yellowjackets of the Northwestern United States, Matthew Kweskin
addr:http://www.evergreen.edu/user/serv_res/research/arthropod/TESCBiota/Vespidae/Kweskin97/main.htm

Entomology & Plant Pathology, Dr. Michael R. Williams
addr:http://www.msstate.edu/Entomology/GLOWORM/GLOW1PAGE.html

Urban Entomology Program, Dr. Timothy G. Myles
addr:http://www.utoronto.ca/forest/termite/termite.htm

References Page 2

Gakken’s Photo Encyclopedia: Ants, Gakushu Kenkyusha
addr:http://ant.edb.miyakyo-u.ac.jp/INTRODUCTION/Gakken79E/Intro.html

The Ants: A Community of Microrobots at the MIT Artificial Intelligence Lab
addr: http://www.ai.mit.edu/projects/ants/

Scientific American March 2000 - *Swarm Smarts*
Pages: 73-79

Pink Panther Image Archive
addr:http://www.high-tech.com/panther/source/graphics.html

C. Ronald Kube, PhD
Collective Robotic Intelligence Project (CRIP).
addr: www.cs.ualberta.ca/~kube

Sources

Maja Mataric

Corey Fehr

Merle Good

Shawn Keown

Gordon Fedoriw