Swarm Intelligence in Robotics

Dr. Alaa Khamis
Pattern Analysis and Machine Intelligence
University of Waterloo

akhamis@pami.uwaterloo.ca

Outline

• Introduction
• Body/Brain Evolution
• Traditional Robotics
• Swarm Robotics
• Swarm Intelligence (SI)
• SI in Traditional Robotics
• SI in Swarm Robotics
• Concluding Remarks
Outline

- Introduction
- Body/Brain Evolution
- Traditional Robotics
- Swarm Robotics
- Swarm Intelligence (SI)
- SI in Traditional Robotics
- SI in Swarm Robotics
- Concluding Remarks

Introduction

- PAMI Research Group

Synchromedia Lab

CIM Lab

Intelligent Multi-crane

CRS F3 and CRS T265

Pattern Analysis Lab

Computer vision

Data Mining Services Catalog

Mobile Robotics Lab

Magellan Pro

ATRV-mini

Soccer-playing Robots
Introduction

- **Swarm Intelligence in Robotics**

  The objective of this talk is to highlight the different applications of the rapidly emerging field of swarm intelligence in solving complex problems of traditional and swarm robotics.

Outline

- Introduction

- **Body/Brain Evolution**

- Traditional Robotics

- Swarm Robotics

- Swarm Intelligence (SI)

- SI in Traditional Robotics

- SI in Swarm Robotics

- Concluding Remarks
**Body/Brain Evolution**

- Brains: SI (100s) → DAI (10s) → AI (1)
- Cognitive Robotics
- Multiagent
- Distributed Robot System
- Centralized Control
- Multiple Machines (10s) → MEMS-based Multiple Machines (100s)

**Outline**

- Introduction
- Body/Brain Evolution
- **Traditional Robotics**
- Swarm Robotics
- Swarm Intelligence (SI)
- SI in Traditional Robotics
- SI in Swarm Robotics
- Concluding Remarks
Traditional Robotics

The Evolutionary Stages

- Industrial Robotics
- Service Robotics
- Personal Robotics

Evolution

- Service Robots for Professional Use
- Service Robots for Personal Use

Traditional Problems

- Environment Perception
- SLAM
- Path Planning
- Navigation
- Autonomy
- Human Interaction
- Learning
Outline

• Introduction
• Body/Brain Evolution
• Traditional Robotics
  • Swarm Robotics
• Swarm Intelligence (SI)
• SI in Traditional Robotics
• SI in Swarm Robotics
• Concluding Remarks

Swarm Robotics

• Definition

  Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment.

  ❖ Resolving complexity.
  ❖ Increasing performance.
  ❖ Simplicity in design.
  ❖ Reliable.
Swarm Robotics

• Typical Problem Domains
  - Box-pushing
  - Foraging
  - Aggregation and Segregation
  - Formation Control
  - Cooperative Mapping
  - Soccer Tournaments
  - Site Preparation
  - Sorting
  - Collective Construction

http://www.robocup.org/
http://www.fira.net/

Swarm Robotics

• Applications of Swarm Robotics
  - Autonomous inspection of complex engineered structures.
  - Distributed sensing tasks in micromachinery or the human body.
  - Killing Cancer Tumors in Human Body
  - Mining
  - Agricultural Foraging
  - Cooperative Tracking
  - Interactive Art
  - Space-based Construction
  - Rescue Operations
  - Humanitarian Demining
  - Surveillance, Reconnaissance and Intelligence.
Swarm Robotics

- Projects

The SWARM-BOT project aims to study a novel swarm robotics system.

- It is directly inspired by the collective behavior of social insects and other animal societies.
- It focuses on self-organization and self-assembling of autonomous agents.
- Its main scientific challenge lays in the development of a novel hardware and of innovative control solutions.

Swarmbots, Marco Dorigo, 2005

Swarm Robotics

- Projects

Leaderless Distributed (LD) Flocking Algorithm
Flocking in Embedded Robotic Systems
CORO - Caltech

PAMI Lab
http://horizon.uwaterloo.ca/multirobot/
Swarm Robotics

- Team Size and Composition

- Team Reconfigurability
Swarm Robotics

• Communication Pattern

Explicit Communication
- Address or Directed Messages
- Broadcast
- Graph

Implicit Communication
- Interaction via Environment (Stigmergy)
- Interaction via Sensing

Swarm Robotics

• Communication Range and Bandwidth

Communication Range
- None
- Near
- Infinite

Communication Bandwidth
- High
- Moderate
- Low
- zero
Swarm Robotics

• **Challenging Problems**
  - Coordination
  - Algorithm Design
  - Implementation and Test
  - Analysis and Modelling

Swarm Robotics

• **Challenging Problems: Coordination**

```
Cooperation
  Planning
    Centralized
    Decentralized
    Hierarchical
    Holonic

Collaboration

Competition
  Negotiation
```
Swarm Robotics

• **Challenging Problems: Algorithm Design**

  Swarm roboticists face the problem of designing both the physical morphology and behaviours of the individual robots such that when those robots interact with each other and their environment, the desired overall collective behaviours will emerge. At present there are no principled approaches to the design of low-level behaviours for a given desired collective behaviour [1].

  “collective behavior is not simply the sum of each participant’s behavior, as others emerge at the society level” [2].

Swarm Robotics

• **Challenging Problems: Implementation and Test**

  To build and rigorously test a swarm of robots in the laboratory requires a considerable experimental infrastructure. Real-robot experiments thus typically proceed hand-in-hand with simulation and good tools are essential [1].
Swarm Robotics

• Challenging Problems: Implementation and Test


  - Open Robot Control Software (OROCOS): open-source real time control architecture for different machines.

  - Microsoft Robotics Studio: is a Windows-based environment for robot control and simulation.

  - Player/Stage/Gazebo: PSG is open source software that used and developed by an international community of researchers from over 30 universities/companies.
Swarm Robotics

- **Challenging Problems: Analysis and Modelling**

  A robotic swarm is typically a *stochastic, non-linear system* and constructing *mathematical models* for both validation and parameter *optimization* is challenging. Such models would surely be an essential part of constructing a safety argument for real-world applications [1].

Outline

- Introduction
- Body/Brain Evolution
- Traditional Robotics
- Swarm Robotics
  - **Swarm Intelligence (SI)**
- SI in Traditional Robotics
- SI in Swarm Robotics
- Concluding Remarks
## Swarm Intelligence

### Analogs

<table>
<thead>
<tr>
<th>Analogy</th>
<th>Nature</th>
<th>Unmanned Autonomous Vehicles (UAVs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ants/UAVs</td>
<td><img src="image" alt="Ants/UAVs" /></td>
<td><img src="image" alt="UAVs1" /> <img src="image" alt="UAVs2" /> <img src="image" alt="UAVs3" /></td>
</tr>
<tr>
<td>Prey/Target</td>
<td><img src="image" alt="Prey/Target" /></td>
<td><img src="image" alt="Target1" /> <img src="image" alt="Target2" /></td>
</tr>
<tr>
<td>Predators/Threats</td>
<td><img src="image" alt="Predators/Threats" /></td>
<td><img src="image" alt="Threat1" /></td>
</tr>
<tr>
<td>Environment/Battlefield</td>
<td><img src="image" alt="Environment/Battlefield" /></td>
<td><img src="image" alt="Battlefield" /></td>
</tr>
</tbody>
</table>
Swarm Intelligence

• **Definition**

Swarm intelligence (SI) refers to the phenomenon of a system of spatially distributed individuals coordinating their actions in a decentralized and self-organized manner so as to exhibit complex collective behavior.

“SI is the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge.”

Swarm intelligence solves optimization problems.

---

Swarm Intelligence

• **Key Elements**

- A large number of “simple” processing elements;
- Neighborhood communication;
- Though convergence in guaranteed, time to convergence is uncertain.
- Most of the research are experimental:

  ![Swarm Intelligence Diagram](image)
Swarm Intelligence

• SI-based Approaches

- Initialize parameters
- Initialize population
- While (end condition not satisfied) loop over all individuals
  - Find best so far
  - Find best neighbor
  - Update individual

Swarm Intelligence

• SI-based Approaches

- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)
- Stochastic Diffusion Search (SDS)
Outline

- Introduction
- Body/Brain Evolution
- Traditional Robotics
- Swarm Robotics
- Swarm Intelligence (SI)
  - SI in Traditional Robotics
- SI in Swarm Robotics
- Concluding Remarks

SI in Traditional Robotics

- ACO-based Motion Planning [4]
- PSO-based Motion Planning [5]
- Wall-following autonomous robot (WFAR) navigation [6]
- Gait Optimization [7]
- Kinematics and Dynamics of Robot Manipulators [8,9]
- Learning [10]
SI in Traditional Robotics

• Motion Planning

Path planning is a process to compute a collision-free path for a robot from a start position to a given goal position, amidst a collection of obstacles.

SI in Traditional Robotics

• ACO-based Motion Planning [4]

  ➢ Proposed Method

  ❖ Step 1: MAKLINK graph theory to establish the free space model of the mobile robot;
  ❖ Step 2: utilizing the Dijkstra algorithm to find a sub-optimal collision-free path;
  ❖ Step 3: utilizing the ACS algorithm to optimize the location of the sub-optimal path so as to generate the globally optimal path.
SI in Traditional Robotics

• ACO-based Motion Planning [4]

➤ Step 1: MAKLINK-based Free Space Model

1. The heights of the environment and obstacles can be ignored;
2. There exist some known obstacles distributed in the environment, both the environment and the obstacles have a polygonal shape;
3. In order to avoid a moving path too close to the obstacles, the boundaries of every obstacle can be expanded.

Free MAKLINK line:

1) Either its two end points are two vertices on two different grown obstacles or one point is a vertex of a grown obstacle and the other is located on a boundary of the environment;
2) Every free MAKLINK line cannot intersect any of the grown obstacles.
SI in Traditional Robotics

- ACO-based Motion Planning [4]
  - Step 1: MAKLINK-based Free Space Model [11,5]
    - Constructing MAKLINK graph
      1. Find all the lines that connect one of the corners that belong to a polygonal obstacle, with all the other obstacles' corners including the corners of the current obstacle;

- ACO-based Motion Planning [4]
  - Step 1: MAKLINK-based Free Space Model [11,5]
    - Constructing MAKLINK graph
      2. Delete the redundant free links to make every free space, of which the edges are free links, obstacle edges and boundary walls, be a convex polygon and its area be largest;
Step 1: MAKLINK-based Free Space Model [11,5]

Constructing MAKLINK graph

3. Find the midpoint of the remained free links and take them as the path nodes, labeling orderly as 1, 2, ..., n. The connections among the midpoints that belong to the same convex area compose a network.

\[ v_1, v_2, \ldots, v_l \] are the middle points of these free MAKLINK lines;

\( l \) is total number of the free MAKLINK lines on a MAKLINK graph.

\( l = 26 \)

\( v_o \) is S

\( v_{l+1} \) is T

Network graph for free motion of robot
SI in Traditional Robotics

• **ACO-based Motion Planning [4]**

  ➢ **Step 2: Dijkstra Algorithm**

  *Sub-optimal path*

  \[ S \rightarrow v_1 \rightarrow v_2 \rightarrow v_{10} \rightarrow v_{11} \rightarrow v_{12} \rightarrow v_{13} \rightarrow v_{16} \rightarrow v_{17} \rightarrow v_{24} \rightarrow T \]

  *Path length* = 507.692 m.

  This path is just a sub-optimal path because it passes only through the middle points of those free MAKLINK lines.

  ![Sub-optimal path generated by Dijkstra](image)

 SI in Traditional Robotics

• **ACO-based Motion Planning [4]**

  ➢ **Step 3: ACS Algorithm**

  Assume sub-optimal path generated by the Dijkstra algorithm is

  \[ P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_{d+1} \]

  These path nodes lie on the middle points of the relevant free MAKLINK lines. Now, we need to adjust and optimize their locations on their corresponding free MAKLINK lines.

  \[ P_i = P_{i1} + (P_{i2} - P_{i1}) \times h_i, \quad h_i \in [0,1], \quad i = 1,2,\ldots, d \]

  \[ P_i = P_{i1} \quad h_i = 0 \]

  \[ P_i = (P_{i1} + P_{i2}) / 2 \quad h_i = 0.5 \text{ midpoint} \]

  \[ P_i = P_{i2} \quad h_i = 1 \]

  ![Path Coding Method](image)
Step 3: ACS Algorithm

The objective function of the optimization problem will be:

\[ L = \sum_{i=0}^{d} \text{length } \{ P_i(h_i), P_{i+1}(h_{i+1}) \} \]

ACS algorithm is used to find the optimal parameter set

\[ \{ h_1^*, h_2^*, ..., h_d^* \} \]

such that \( L \) has the minimum value.

Grid graph on which there are \( dx11 \) nodes in total.

\( n_{ij} \) is node \( j \) on line \( h_i \).

The path of an ant depart from the starting point \( S \) is

\[ S \rightarrow n_{1j} \rightarrow n_{2j} \rightarrow ... \]
\[ \rightarrow n_{dj} \rightarrow T \]

Generating of nodes and moving paths
SI in Traditional Robotics

- ACO-based Motion Planning [4]

  ➢ Step 3: ACS Algorithm

  ◦ Pheromone Concentration

  Assume that at initial time $t = 0$

  All the nodes have the same pheromone concentration $\tau_0$: $\tau_{ij}(0) = \tau_0$ ($i = 1, 2, ..., d$; $j = 0, 1, 2, ..., 10$)

  ◦ Transition Rule

  Assume the number of ants is $m$.

  In moving process, for an ant $k$, when it locates on line $h_{i-1}$, it will choose a node $j$ from the eleven nodes of the next line $h_i$ to move to according to the following rule:

  $$ j = \begin{cases} 
  \arg \max_{u \in A} \{ [\tau_{iu}(t)][\eta_{iu}]^\beta \}, & \text{if } q \leq q_o \\
  J, & \text{if } q > q_o 
  \end{cases} $$
SI in Traditional Robotics

• ACO-based Motion Planning [4]

➤ Step 3: ACS Algorithm

◊ Transition Rule

\[
j = \begin{cases} 
  \arg \max_{u \in A} \{[\tau_{iu}(t)][\eta_{iu}]^\beta\}, & \text{if } q \leq q_o \\
  J, & \text{if } q > q_o 
\end{cases}
\]

where

A represents the set: \{0, 1, 2, \ldots, 10\};

\(\tau_{iu}(t)\) is the pheromone concentration of node \(n_{iu}\);

\(\eta_{iu}\) represents the visibility of node \(n_{iu}\) and is computed by the following equation:

\[
\eta_{ij} = \frac{1.1 - |y_{ij} - y_{ij}^*|}{1.1}
\]

\(y_{ij}\) is the y-coordinate of node \(n_{ij}\) and \(y_{ij}^*\) are values that are corresponding to the path nodes on the optimal robot path generated by the ACS algorithm in the previous iteration.
SI in Traditional Robotics

• ACO-based Motion Planning [4]

➢ Step 3: ACS Algorithm

◊ Transition Rule

\[ j = \begin{cases} 
\arg \max_{u \in A} \left\{ \frac{\tau_{iu}(t) \cdot \eta_{iu}^\beta}{J}, \right. \\
\left. \frac{1}{J}, \right. \\
\end{cases} \]

if \( q \leq q_o \)

if \( q > q_o \)

\( \beta \) is an adjustable parameter which controls the relative importance of visibility \( \eta_{iu} \) versus pheromone concentration \( \tau_{iu}(t) \);

\( q \) is a random variable uniformly distributed over \([0, 1]\); \( q_o \) is a tunable parameter \((0 \leq q_o \leq 1)\);
SI in Traditional Robotics

- ACO-based Motion Planning [4]

  - **Step 3: ACS Algorithm**

    - **Pheromone Update: After Each Iteration**
      \[
      \tau_{ij}(t) \leftarrow (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta \tau_{ij}(t)
      \]
      \[0 < \rho < 1\]
      \[\Delta \tau_{ij}(t) = \frac{1}{L^+}\]
      where \(L^+\) is the length of robot path corresponding to the best ant tour.

  - ACO-based Motion Planning [4]

  - **Step 3: ACS Algorithm**

    - **Pheromone Update: After passing through a node \(n_{ij}\)**
      \[
      \tau_{ij}(t) \leftarrow (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_o
      \]
      When a node is visited several times by ants, repeatedly applying the local updating rule will make the pheromone level of this node diminish.

      This has the effect of making the visited nodes less and less attractive to the ants, which indirectly favors the exploration of not yet visited nodes.
SI in Traditional Robotics

• ACO-based Motion Planning [4]

➢ Results

<table>
<thead>
<tr>
<th></th>
<th>ACS Algorithm</th>
<th>Real-coded GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average CPU time per iteration (sec.)</td>
<td>0.00059</td>
<td>0.00067</td>
</tr>
<tr>
<td>Average number of iterations needed for convergence</td>
<td>175</td>
<td>912</td>
</tr>
<tr>
<td>Average CPU time needed for obtaining optimal solution (sec.)</td>
<td>0.1033</td>
<td>0.6110</td>
</tr>
</tbody>
</table>

Sub-optimal path from Dijkstra algorithm is

\[ P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_{d+1} \]

These path nodes lie on the middle points of the relevant free MAKLINK lines. Location adjustment:

\[ P_i = P_{i1} + (P_{i2} - P_{i1}) \times h_i, \quad h_i \in [0,1], \quad i = 1, 2, \ldots, d \]

\[ P_i = P_{i1} \quad h_i = 0 \]
\[ P_i = (P_{i1} + P_{i2}) / 2 \quad h_i = 0.5 \quad \text{midpoint} \]
\[ P_i = P_{i2} \quad h_i = 1 \]
SI in Traditional Robotics

- **PSO-based Motion Planning [5]**

  Each particle $X_i$ is constructed as:
  
  $$X_i = (h_1, h_2, ..., h_d)$$

  The fitness value of each particle is:
  
  $$f(X_i) = \sum_{i=0}^{d} \text{length } \{ P_i(h_i), P_{i+1}(h_{i+1}) \}$$

  The smaller the fitness value is, the better the solution is.

---

SI in Traditional Robotics

- **PSO-based Motion Planning [5]**

  1. Initialize particles at random, and set pBest = $X_i$;

  2. Calculate each particle’s fitness value according to equation

  $$f(X_i) = \sum_{i=0}^{d} \text{length } \{ P_i(h_i), P_{i+1}(h_{i+1}) \}$$

  and label the particle with the minimum fitness value as gBest;
SI in Traditional Robotics

• PSO-based Motion Planning [5]

3. For $k_1 = 1$ to $k_{\text{max}}$ do {

4. For each particle $X_i$ do {

5. Update $v_i$ and $x_i$ according to the following equations:

\[
v_i(k + 1) = \omega \times v_i(k) + c_1 \times \text{rand}() \times [p\text{Best}_i(k) - x_i(k)] + \\
c_2 \times \text{rand}() \times [g\text{Best}(k) - x_i(k)]
\]

\[
x_i(k + 1) = x_i(k) + v_i(k + 1), \quad 1 \leq i \leq n,
\]

6. Calculate the fitness according to equation

\[
f(X_i) = \sum_{i=0}^{d} \text{length} \{P_i(h_i), P_{i+1}(h_{i+1})\}
\]

7. Update $g\text{Best}$ and $p\text{Best}_i$ ;

8. If $||v|| \leq \epsilon$, terminate ;}

PSO-based Motion Planning [5]
Outline

• Introduction
• Body/Brain Evolution
• Traditional Robotics
• Swarm Robotics
• Swarm Intelligence (SI)
• SI in Traditional Robotics
  • SI in Swarm Robotics
• Concluding Remarks

SI in Swarm Robotics

• Multirobot Control [12]
• Collective Robotic Search (CRS) [13]
• Odor-source Localization [14,15]
• Mobile Sensor Deployment [16]
• Learning [17]
• Communication Relay [18]
• Group Behavior [19]
SI in Swarm Robotics

• Multirobot Control [12]

Unintelligent carts are commonly found in large airports. Travelers pick up carts at designated points and leave them in arbitrary places. It is a considerable task to re-collect them.

It is, therefore, desirable that intelligent carts (intelligent robots) draw themselves together automatically.

Objectives

Using mobile software agents to locate robots scattered in a field, e.g. an airport, and make them autonomously determine their moving behaviors by using an ACO-based clustering algorithm.
SI in Swarm Robotics

• Multirobot Control [12]

➢ Assumptions

- Each robot has a function for collision avoidance;

- Each robot has a function to sense RFID embedded in the floor carpet to detect its precise coordinate in the field.

A team of mobile robots work under control of mobile agents

RFID under a carpet tile

SI in Swarm Robotics

• Multirobot Control [12]

➢ Quasi Optimal Robots Collection

Step 1:

One mobile agent issued from the host computer visits scattered robots one by one and collects the positions of them.
• Multirobot Control [12]

> Quasi Optimal Robots Collection

**Step 2:**

Another agent called the simulation agent, performs ACC algorithm and produces the quasi optimal gathering positions for the mobile robots. The simulation agent is a static agent that resides in the host computer.

**Step 3:**

A number of mobile agents are issued from the host computer. One mobile agent migrates to a designated mobile robot, and drives the robot to the assigned quasi optimal position that is calculated in step 2.
SI in Swarm Robotics

- **Multirobot Control [12]**
  - ACO-based Clustering
    - More objects are clustered in a place where strong pheromone sensed.
    - When the artificial ant finds a cluster with certain number of objects, it tends to avoid picking up an object from the cluster. This number can be updated later.
    - If the artificial ant cannot find any cluster with certain strength of pheromone, it just continues a random walk.

\[
f(p) = 1 - \frac{(p + l) \cdot k}{100}
\]

\(p\): density of pheromone = number of adjacent objects. Thus, when an object is completely surrounded by other objects, \(p\) is the max. value = 9
SI in Swarm Robotics

- Multirobot Control [12]
  ➢ ACO-based Clustering

\[ f(p) = 1 - \frac{(p + l) \times k}{100} \]

k: constant value

Authors selected \( k = 13 \) in order to prevent any object surrounded by other eight objects be picked up.

\[ f(p) = 1 - \frac{(8 + 0) \times 13}{100} = 1 - 1.04 = -0.04 \approx 0 \quad (\text{never pick it up}). \]

SI in Swarm Robotics

- Multirobot Control [12]
  ➢ ACO-based Clustering

\[ f(p) = 1 - \frac{(p + l) \times k}{100} \]

l: constant value to make an object be locked.

l = zero (not locked).

If \( c = \# \) of clusters & \( o = \# \) of objects

\[ c < \frac{2}{3} o \quad \text{and} \quad p > 3 \Rightarrow l = 6 \]
\[ c < \frac{1}{3} o \quad \text{and} \quad p > 7 \Rightarrow l = 3 \]
\[ p > 9 \Rightarrow l = 1 \]

Any objects meet these conditions are locked. This lock process prevents artificial ants remove objects from growing clusters.
SI in Swarm Robotics

- **Multirobot Control [12]**

  - **ACO-based Clustering**

  In “pheromone walk” state, an artificial ant tends probabilistically move toward a place it senses the strongest pheromone. The probability that the artificial ant takes a certain direction is \( \frac{n}{10} \), where \( n \) is the strength of the sensed pheromone of that direction.

An artificial ant carrying an object determines whether to put the carrying object or to continue to carry. This decision is made based on the formula:

\[
f(p) = \frac{p \times k}{100}
\]

The more it senses strong pheromone, the more it tends to put the carrying object.

The probability \( f(p) = 1 \) (must put it down).
SI in Swarm Robotics

• Multirobot Control [12]
  ➢ ACO-based Clustering

*Termination Condition:*

The number of resulted clusters is less than ten, and all the clusters have more or equal to three objects.

SI in Swarm Robotics

• Multirobot Control [12]
  ➢ Results

Airport Field (Objects: 400, Ant Agent: 100)

<table>
<thead>
<tr>
<th>Airport field</th>
<th>Ant Colony Clustering</th>
<th>Specified Position Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost</td>
<td>Ave</td>
</tr>
<tr>
<td>1</td>
<td>4822</td>
<td>12.05</td>
</tr>
<tr>
<td>2</td>
<td>3173</td>
<td>7.93</td>
</tr>
<tr>
<td>3</td>
<td>3648</td>
<td>9.12</td>
</tr>
<tr>
<td>4</td>
<td>3803</td>
<td>9.51</td>
</tr>
<tr>
<td>5</td>
<td>4330</td>
<td>10.82</td>
</tr>
</tbody>
</table>

*Clust* represents the number of clusters, and *Ave* represents the average distance of all the objects moved.
SI in Swarm Robotics

• Collective Robotic Search (CRS) [13]

A group of unmanned mobile robots are searching for a specified target in a high risk environment.

- A number of robots/particles are randomly dropped into a specified area and flown through the search space with one new position calculated for each particle per iteration;

- The coordinates of the target are known and the robots use a fitness function, in this case the Euclidean distance of the individual robots relative to the target, to analyze the status of their current position;

- Obstacle’s boundary coordinates are known.
SI in Swarm Robotics

• Collective Robotic Search (CRS) [13]

➢ PSO-CRS Algorithm

Step 1:

A population of robots is initialized in the search environment containing a target and an obstacle, with random positions, velocities, personal best positions (p_{id}), and global best position (p_{gd}).

\[
\text{fitness} = \sqrt{(T_x - P_x)^2 + (T_y - P_y)^2}
\]
SI in Swarm Robotics

• Collective Robotic Search (CRS) [13]

  ➢ PSO-CRS Algorithm

  Step 3:
  The robot’s fitness is compared with its previous best fitness (pBest\textsubscript{id}) for every iteration to determine the next possible coordinate position for each robot in the search environment. The next possible velocity and position of each robot are determined according to the following equations:

  \[ v_{id}(k + 1) = \omega \times v_{id}(k) + c_1 \times rand() \times [pBest_{id}(k) - x_{id}(k)] + c_2 \times rand() \times [gBest_{d}(k) - x_{id}(k)] \]

  \[ x_{id}(k + 1) = x_{id}(k) + v_{id}(k + 1) \]

  If the next possible position \( x_{id}(k+1) \) resides within the obstacle space, the obstacle avoidance mechanism is employed,

  otherwise the robot moves to this new position.
SI in Swarm Robotics

• Collective Robotic Search (CRS) [13]

  ➢ PSO-CRS Algorithm

  **Obstacle avoidance mechanism:**

  $$(\text{horiz}, \text{vert}, \text{next}) = \text{arc}(x_{\text{start}}, y_{\text{start}}, x_{\text{center}}, y_{\text{center}}, \text{dir}, \text{points})$$

  dir: CW – CCW
  points=4

  The new position of the particle is set at the second point in the arc.

---

SI in Swarm Robotics

• Collective Robotic Search (CRS) [13]

  ➢ PSO-CRS Algorithm

  **Step 5:**

  The pBest$_{id}$ with the best fitness for the entire swarm is determined and the global best coordinate location, gBest$_{d}$, is updated with this pBest$_{id}$.

  **Step 6:**

  Until convergence is reached, repeat steps.
SI in Swarm Robotics

• Collective Robotic Search (CRS) [13]

➤ Simulation Results

Search space: 20 by 20 units.

\[ V_{\text{max}} = 0.5 \text{ units.} \]

# of robots = 10

# of targets = 1

# of obstacles = 1

Robots’ pathways to the target location.
Two robots collide into the obstacle.

Robots’ pathways to the target location with obstacle avoidance mechanism.
SI in Swarm Robotics

- Collective Robotic Search (CRS) [13]

Simulation Results

Average Number of Iterations taken by the Swarm with standard deviations over 20 trails

<table>
<thead>
<tr>
<th>Obstacle Type</th>
<th>PSO Parameters 1- c1=0.5, c2=2, w=0.6</th>
<th>PSO Parameters 2- c1=2, c2=2, w=0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>Average±std 89.43±9.86</td>
<td>105.6±10.68</td>
</tr>
<tr>
<td></td>
<td>Maximum     104</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Minimum     66</td>
<td>85</td>
</tr>
<tr>
<td>Circle</td>
<td>Average±std 76.75±8.75</td>
<td>106.1±11.61</td>
</tr>
<tr>
<td></td>
<td>Maximum     95</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Minimum     64</td>
<td>78</td>
</tr>
</tbody>
</table>

Outline

- Introduction
- Body/Brain Evolution
- Traditional Robotics
- Swarm Robotics
- Swarm Intelligence (SI)
- SI in Traditional Robotics
- SI in Swarm Robotics
- Concluding Remarks
Concluding Remarks

- Swarm intelligence techniques such as Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) have shown to be an effective global optimization algorithms in traditional and swarm robotics.

- ACO and PSO have been successfully applied in solving problems such as motion planning, gait optimization, group behavior, control, learning, odor-source localization, collective robotic search and mobile sensor deployment, to mention a few.

References


References


