Overview of Spatial Computing and SASO

Alex Cornejo

October 31, 2008
What is this about?

- Spatial computing is a collection of devices distributed through physical space, where:
  - Moving information between devices is strongly dependent on the distance between them.
  - The functional goals of the system are defined in terms of the spatial structure.
  - SASO - Self-Adapting Self-Organizing Systems
    - Algorithms that work on large scale systems/networks and operate by local rules and have self-* properties.
  - Bio-inspired algorithms seems to be a recurrent topic.
  - We overview 3 papers which seem somewhat related to our work.
What is this about?

- Spatial computing is a collection of devices distributed through physical space, where:
  - Moving information between devices is strongly dependent on the distance between them.
What is this about?

- Spatial computing is a collection of devices distributed through physical space, where:
  - Moving information between devices is strongly dependent on the distance between them.
  - The functional goals of the system are defined in terms of the spatial structure.
What is this about?

- Spatial computing is a collection of devices distributed through physical space, where:
  - Moving information between devices is strongly dependent on the distance between them.
  - The functional goals of the system are defined in terms of the spatial structure.
- SASO - Self-Adapting Self-Organizing Systems
What is this about?

- Spatial computing is a collection of devices distributed through physical space, where:
  - Moving information between devices is strongly dependent on the distance between them.
  - The functional goals of the system are defined in terms of the spatial structure.
- SASO - Self-Adapting Self-Organizing Systems
  - Algorithms that work on large scale systems/networks and operate by local rules and have self-* properties.
What is this about?

- Spatial computing is a collection of devices distributed through physical space, where:
  - Moving information between devices is strongly dependent on the distance between them.
  - The functional goals of the system are defined in terms of the spatial structure.
- SASO - Self-Adapting Self-Organizing Systems
  - Algorithms that work on large scale systems/networks and operate by local rules and have self-* properties.
  - Bio-inspired algorithms seems to be a recurrent topic.
What is this about?

▶ Spatial computing is a collection of devices distributed through physical space, where:
  ▶ Moving information between devices is strongly dependent on the distance between them.
  ▶ The functional goals of the system are defined in terms of the spatial structure.
▶ SASO - Self-Adapting Self-Organizing Systems
  ▶ Algorithms that work on large scale systems/networks and operate by local rules and have self-* properties.
  ▶ Bio-inspired algorithms seem to be a recurrent topic.
▶ We overview 3 papers which seem somewhat related to our work.
First paper

- Bioinspired environmental coordination in spatial computing systems.
First paper

- Bioinspired environmental coordination in spatial computing systems.
- A review/position paper that summarizes key ideas of most papers in the conference.
First paper

- Bioinspired environmental coordination in spatial computing systems.
- A review/position paper that summarizes key ideas of most papers in the conference.
- The key word here is stigmergy.
First paper

- Bioinspired environmental coordination in spatial computing systems.
- A review/position paper that summarizes key ideas of most papers in the conference.
- The key word here is stigmergy.

**Definition (Stigmergy)**

A mechanism of spontaneous, indirect coordination between agents or actions, where the trace left in the environment by an action stimulates the performance of a subsequent action, by the same or a different agent.
Foraging

- Working ants lay down pheromones in their environment when foraging.
Foraging

- Working ants lay down pheromones in their environment when foraging.
- When the insects find a food source they leave chemical trails during their return.
Foraging

- Working ants lay down pheromones in their environment when foraging.
- When the insects find a food source they leave chemical trails during their return.
- When exploring space, ants are attracted to these pheromones.
Foraging

- Working ants lay down pheromones in their environment when foraging.
- When the insects find a food source they leave chemical trails during their return.
- When exploring space, ants are attracted to these pheromones.
- It seems to work pretty well for them (ants are still here after $13 \times 10^7$ years).
Is this practical?

- A first implementation appeared on ICRA 1999 using volatile chemicals and chemical sensors.
Is this practical?

- A first implementation appeared on ICRA 1999 using volatile chemicals and chemical sensors.
- The idea was tested using robots and disappearing ink at UMD in 2003.
Is this practical?

- A first implementation appeared on ICRA 1999 using volatile chemicals and chemical sensors.
- The idea was tested using robots and disappearing ink at UMD in 2003.
- A group in Japan implemented pheromones by projecting the trails on the floor of the lab and placing sensors on the robots (IROS 2004).
Collective construction

- Inspired by nest construction of termites.
Collective construction

- Inspired by nest construction of termites.

- Complex functional architecture including features like gardens, nurseries, royal chambers, and extensive systems for temperature control and atmospheric regulation, but with no central control.
How to use this?

- Some groups focused on specifying sets of local rules and interactions and observing the characteristics of the resulting structures.

- Others (D. Rus @CSAIL) specified a particular target structure and tried to find a set of actions and rules that are guaranteed to procure such structure.

- Information can be stored on the building blocks by using RFID tags.
How to use this?

- Some groups focused on specifying sets of local rules and interactions and observing the characteristics of the resulting structures.
- Others (D. Rus @CSAIL) specified a particular target structure and tried to find a set of actions and rules that are guaranteed to procure such structure.
How to use this?

- Some groups focused on specifying sets of local rules and interactions and observing the characteristics of the resulting structures.
- Others (D. Rus @CSAIL) specified a particular target structure and tried to find a set of actions and rules that are guaranteed to procure such structure.
- Information can be stored on the building blocks by using RFID tags.
The problem of simultaneous localization and mapping requires one or more robots to build a map of an unknown environment.
The problem of simultaneous localization and mapping requires one or more robots to build a map of an unknown environment.

One of the classic SLAM problems is to close the loop (have you already visited this or is just similar?).
The problem of simultaneous localization and mapping requires one or more robots to build a map of an unknown environment.

One of the classic SLAM problems is to close the loop (have you already visited this or is just similar?).

Very common problem in man made interiors since corridors and doors tend to be in square grid (furthermore indoors there is no GPS).
Solution

- As before, storing information in the environment is a fast and reliable way to solve this problem (by dropping an RFID tag for example)
Moving on to...
Second paper

- Local desynchronization in wireless networks.
Second paper

- Local desynchronization in wireless networks.
- Defines desynchronization and proves its existence on several classes of graphs.

Proposes bio-inspired local desynchronization algorithm (how do fireflies synchronize their flashes?)

Presents some simulation results, not too many though...
Second paper

- Local desynchronization in wireless networks.
- Defines desynchronization and proves its existence on several classes of graphs.
- Proposes bio-inspired local desynchronization algorithm (how do fireflies synchronize their flashes?)
Local desynchronization in wireless networks.
Defines desynchronization and proves its existence on several classes of graphs.
Proposes bio-inspired local desynchronization algorithm (how do fireflies synchronize their flashes?)
Presents some simulation results, not too many though...
Desynchronousization

- To find an optimal scheduling of agents is NP-hard (minimum number of colors), and more so if the graph changes you might need to recompute the scheduling.
Desynchrononization

- To find an optimal scheduling of agents is NP-hard (minimum number of colors), and more so if the graph changes you might need to recompute the scheduling.
- An alternative solution to coloring is to “desynchronize” the nodes.
Desynchronization

- To find an optimal scheduling of agents is NP-hard (minimum number of colors), and more so if the graph changes you might need to recompute the scheduling.
- An alternative solution to coloring is to “desynchronize” the nodes.
- Assign each node a phase, such that conflicting nodes are as far apart as possible.
Notation

- The distance between phase $\phi_i$ and $\phi_j$ is defined as $\Delta_{i,j} = \phi_j - \phi_i$. 
Notation

- The distance between phase $\phi_i$ and $\phi_j$ is defined as $\Delta_{i,j} = \phi_j - \phi_i$.
- The next and previous phase neighbors of node $i$ are

  \[
  n(i) = \arg \max_{j \in N(i)} \Delta_{i,j}
  \]

  \[
  p(i) = \arg \min_{j \in N(i)} \Delta_{i,j}
  \]
Notation

- The distance between phase $\phi_i$ and $\phi_j$ is defined as $\Delta_{i,j} = \phi_j - \phi_i$.
- The next and previous phase neighbors of node $i$ are

  $$n(i) = \arg \max_{j \in N(i)} \Delta_{i,j}$$
  $$p(i) = \arg \min_{j \in N(i)} \Delta_{i,j}$$

- A configuration is desynchronized if for every $i$ it holds that

  $$\Delta_{i,p(i)} = \Delta_{n(i),i}$$
Desynchronised graphs

- For a complete graph, the only desynchronisation corresponds to all nodes spread evenly across the phase ring \( \Delta_p(i) = \Delta_n(i) = \frac{1}{n} \)
Desynchrononized graphs

- Two-colorable graphs have a two-phase desynchronization
  \[ \Delta_p(i) = \Delta_n(i) = \frac{1}{2} \]
Cycles of length \( n \) have a number of desynchronizations equal to the number of divisors of \( n \) greater than 1.
Desyncrhonized graphs

- Any graph that contains a Hamiltonian cycle has a desyncrhonization.
Achieving desynchronization

- Suppose each device is equipped with an oscillator with a constant frequency $\omega$. 

We assume every oscillator $i$ fires and resets its state to 0 when $\theta_i(t) = 1$.

Idea: Adjust your phase to be in the middle of your neighbor's phases.
Achieving desynchronization

- Suppose each device is equipped with an oscillator with a constant frequency $\omega$.
- The oscillator state of node $i$ at time $t$ is denoted by $\theta_i(t)$, where $\theta_i(t) = \omega t + \phi_i(t) \mod 1$. 

We assume the every oscillator $i$ fires and resets its state to 0 when $\theta_i(t) = 1$.

Idea: Adjust your phase to be in the middle of your neighbor's phases.
Achieving desynchrononization

- Suppose each device is equipped with an oscillator with a constant frequency \( \omega \).
- The oscillator state of node \( i \) at time \( t \) is denoted by \( \theta_i(t) \), where \( \theta_i(t) = \omega t + \phi_i(t) \mod 1 \).
- We assume the every oscillator \( i \) fires and resets its state to 0 when \( \theta_i(t) = 1 \).
Achieving desynchronization

- Suppose each device is equipped with an oscillator with a constant frequency $\omega$.
- The oscillator state of node $i$ at time $t$ is denoted by $\theta_i(t)$, where $\theta_i(t) = \omega t + \phi_i(t) \mod 1$.
- We assume the every oscillator $i$ fires and resets its state to 0 when $\theta_i(t) = 1$.
- Idea: Adjust your phase to be in the middle of your neighbor’s phases.
Achieving desynchronization

- When $i$ fires
  
  \[
  \text{justFired} = \text{True}
  \]
Achieving desynchronization

- When $i$ fires
  
  $\text{justFired} = \text{True}$

- When $j \in N(i)$ fires
  
  $\text{if justFired}$
  
  $\text{justFired} = \text{False}$
  
  $\text{next} = \Delta_{i,j}$
  
  $\phi_i = \phi_i + \alpha(\text{prev} - \text{next})/2$

  $\text{else}$

  $\text{prev} = \Delta_{i,j}$
Does it converge?

- There is a convergence proof for 1-hop graphs (complete graphs).
Does it converge?

- There is a convergence proof for 1-hop graphs (complete graphs).
- For other graphs it is not clear that a desynchronized state even exist.
Does it converge?

- There is a convergence proof for 1-hop graphs (complete graphs).
- For other graphs it is not clear that a desynchonized state even exist.
- But so far the simulations look promising...
Does it converge?
Does it converge?
Moving on to...
Third paper

- Distance sensitive design of WSNs
Third paper

- Distance sensitive design of WSNs
- Not sure if to classify as theory or systems paper.
Third paper

- Distance sensitive design of WSNs
- Not sure if to classify as theory or systems paper.
- Only one theorem, which is stated without proof.
Third paper

- Distance sensitive design of WSNs
- Not sure if to classify as theory or systems paper.
- Only one theorem, which is stated without proof.
- No pseudo-code, no algorithm description, no simulations...
Distance sensitive design of WSNs

- Large-scale wireless sensor networks (i.e. > 1000 nodes) are rare at best.
Distance sensitive design of WSNs

- Large-scale wireless sensor networks (i.e. > 1000 nodes) are rare at best.
- With a channel with 99.5% reliability, a 100-hop graph drops more than $\frac{1}{3}$ of the packets.
Distance sensitive design of WSNs

- Large-scale wireless sensor networks (i.e. > 1000 nodes) are rare at best.
- With a channel with 99.5% reliability, a 100-hop graph drops more than $\frac{1}{3}$ of the packets.
- Distance sensitive guarantees seem the best approach to designing reliable services in such networks.
Pursuer-evader tracking application

- Just a fancy way of saying mobile tracking (i.e. **Move** and **Find** operations) with velocities.

Let \( V_e \) and \( V_p \) be the velocities of the evader and the pursuer respectively, we only care about their ratio \( \alpha = \frac{V_p}{V_e} \) (\( \alpha > 1 \)).

Instead provide information with latency, error and rate which is proportional to the distance.
Pursuer-evader tracking application

- Just a fancy way of saying mobile tracking (i.e. MOVE and FIND operations) with velocities.
- Let $V_e$ and $V_p$ be the velocities of the evader and the pursuer respectively, we only care about their ratio $\alpha = V_p/V_e$ ($\alpha > 1$).
Pursuer-evader tracking application

- Just a fancy way of saying mobile tracking (i.e. MOVE and FIND operations) with velocities.
- Let $V_e$ and $V_p$ be the velocities of the evader and the pursuer respectively, we only care about their ratio $\alpha = V_p/V_e$ ($\alpha > 1$).
- With perfect information this problem is trivial.
Pursuer-evader tracking application

- Just a fancy way of saying mobile tracking (i.e. Move and Find operations) with velocities.
- Let $V_e$ and $V_p$ be the velocities of the evader and the pursuer respectively, we only care about their ratio $\alpha = V_p/V_e$ ($\alpha > 1$).
- With perfect information this problem is trivial.
- Instead provide information with latency, error and rate which is proportional to the distance.
What assumptions to use?

- In the pursuer-evader application it is not clear if providing information with a distance sensitive latency, error and rate will allow the pursuer to catch the evader.
What assumptions to use?

- In the pursuer-evader application it is not clear if providing information with a distance sensitive latency, error and rate will allow the pursuer to catch the evader.

- Let $k$ be a constant such that $k > \frac{\alpha+1}{\alpha-1}$, let $z(t)$ denote the error in the location of the evader at time $t$, let $d(t)$ denote the distance between the pursuer and the evader at time $t$, let $\delta(t)$ denote the staleness of the state provided to the pursuer at time $t$, and let $I(t)$ be the maximum interval after $t$ at which the location is provided to the pursuer.
What assumptions to use?

Theorem (\( i + - i \))

The evader will be caught if there exists a time \( T_0 \) such that following conditions hold for all \( t > T_0 \).

1. \( z(t) \leq \frac{d(t)}{k} \)
2. \( \delta(t) \leq \frac{d(t)}{v_e} \left( 1 - \frac{\alpha + k + 1}{\alpha k} \right) \)
3. \( l(t) \leq \frac{d(t)}{v_p} \left( \frac{k + 1}{k} \right) \)
How to achieve this?

- Use hierarchical clustering service.

- Spread info across all level $k$ neighbors ($\leq 8$) to avoid the "boundary problem".

- Use pipelining to achieve distance sensitive latency.
How to achieve this?

- Use hierarchical clustering service.

- Spread info across all level $k$ neighbors ($\leq 8$) to avoid the “boundary problem”.
How to achieve this?

▶ Use hierarchical clustering service.

▶ Spread info across all level $k$ neighbors ($\leq 8$) to avoid the "boundary problem".

▶ Use pipelining to achieve distance sensitive latency.
1. A unique node is clusterhead at each level.
Clustering properties

1. A unique node is clusterhead at each level.
2. All nodes within distance $\frac{3^k - 1}{2}$ from a level $k$ clusterhead belong to that cluster.
Clustering properties

1. A unique node is clusterhead at each level.
2. All nodes within distance $\frac{3^k-1}{2}$ from a level $k$ clusterhead belong to that cluster.
3. The maximum distance of a node from its level $k$ clusterhead is $z^k \times \frac{3^k-1}{2}$
Clustering properties

1. A unique node is clusterhead at each level.
2. All nodes within distance $\frac{3^k - 1}{2}$ from a level $k$ clusterhead belong to that cluster.
3. The maximum distance of a node from its level $k$ clusterhead is $z^k \times \frac{3^k - 1}{2}$
4. There is a path from each clusterhead to all nodes in the cluster using only nodes from that cluster,
Clustering properties

1. A unique node is clusterhead at each level.
2. All nodes within distance $\frac{3^k - 1}{2}$ from a level $k$ clusterhead belong to that cluster.
3. The maximum distance of a node from its level $k$ clusterhead is $z^k \times \frac{3^k - 1}{2}$
4. There is a path from each clusterhead to all nodes in the cluster using only nodes from that cluster,
5. At all levels $k > 0$ there is at least one and at most 8 neighboring level $k$ clusters (and paths between the clusterheads)