Team-Triggered Coordination of Robotic Networks for Optimal Deployment

Cameron Nowzari[1], Jorge Cortés[2], and George J. Pappas[1]

Electrical and Systems Engineering[1]
University of Pennsylvania

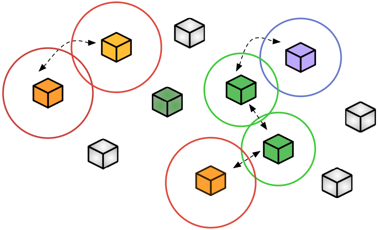
Mechanical and Aerospace Engineering[2]
University of California, San Diego

American Control Conference
Chicago, Illinois
July 3, 2015

-Coordination of robotic networks-

Each individual

- **senses** immediate environment
- **communicates** with others
- **processes** information gathered
- **takes action** in response



Multiple agents provide

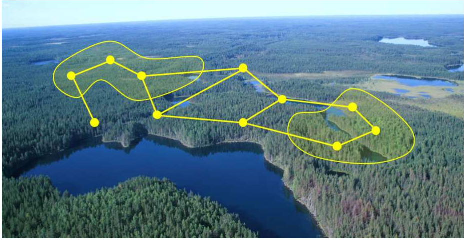
- inherent robustness
- adaptive behavior
- enable tasks beyond individuals' capabilities

-Optimal deployment- of robotic sensor networks

Objective: optimal task allocation and space partitioning
optimal placement and tuning of sensors

Why?

- servicing
- resource allocation
- environmental monitoring
- data collection
- force protection
- surveillance
- search and rescue



Agents take actions at some fixed period T

Agents take actions at some fixed period T

Eve, where are you?!

Agents take actions at some fixed period T

Eve, where are you?!

I'm here!

Real-time implementations of -optimal deployment-

Time-triggered (periodic) coordination

Agents take actions at some fixed period T

Now where are you?

Agents take actions at some fixed period T

Now where are you?

Still here!

Real-time implementations of -optimal deployment-

Time-triggered (periodic) coordination

Agents take actions at some fixed period T

How about now?

Real-time implementations of -optimal deployment-

Time-triggered (periodic) coordination

Agents take actions at some fixed period T

How about now?

Still here...

Agents take actions at some fixed period T

And now??

Agents take actions at some fixed period T

And now??

Leave me alone.

Agents take actions at some fixed period T

And now??

Leave me alone.

Simple, but Wasteful

Agents decide when to take actions based on available information

Agents decide when to take actions based on available information

Eve, where are you?!

Real-time implementations of -optimal deployment-

Self-triggered coordination

Agents decide when to take actions based on available information

Eve, where are you?!

I'm here!

Real-time implementations of -optimal deployment-

Self-triggered coordination

Agents decide when to take actions based on available information

(She probably hasn't moved too far...)

Real-time implementations of -optimal deployment-

Self-triggered coordination

Agents decide when to take actions based on available information

(She probably hasn't moved too far...)

Agents decide when to take actions based on available information

(I'll ask her again in a few seconds)

Agents decide when to take actions based on available information

(I'll ask her again in a few seconds)

Agents decide when to take actions based on available information

Now where are you?

Real-time implementations of -optimal deployment-

Self-triggered coordination

Agents decide when to take actions based on available information

Now where are you?

Still here!

Real-time implementations of -optimal deployment-

Self-triggered coordination

Agents decide when to take actions based on available information

Now where are you?

Still here!

Actions taken only when necessary, but potentially still conservative!

Cooperative agents share more information with each other

Cooperative agents share more information with each other

Eve, where are you?!

Cooperative agents share more information with each other

Eve, where are you?!

I'm here!

Cooperative agents share more information with each other

But I'm going this way!

Real-time implementations of -optimal deployment-
-Team-triggered coordination-

Cooperative agents share more information with each other

Cooperative agents share more information with each other

Real-time implementations of -optimal deployment-

-Team-triggered coordination-

Cooperative agents share more information with each other

Cooperative agents share more information with each other

Higher quality information allows for less communication!

-Team-triggered- control

Objective: Combine best properties of event- and self-triggered strategies into a unified, implementable approach

How?

-Team-triggered- control

Objective: Combine best properties of event- and self-triggered strategies into a unified, implementable approach

How?

- Agents make **promises** to neighbors about their future states
- Agents warn each other when **promises** need to be broken

Outline

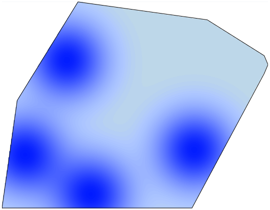
1. Motivation
2. Problem Formulation
   - aggregate objective optimization
   - Voronoi partition
3. Triggered Deployment Algorithms
   - self-triggered deployment algorithm
   - team-triggered deployment algorithm
   - simulations
4. Conclusions

Expected-value multicenter function

Objective: Given sensors/nodes/robots/sites $(p_1, \ldots, p_n)$ moving in environment S achieve **optimal coverage**

$\phi : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ density

agent performance decreases with distance



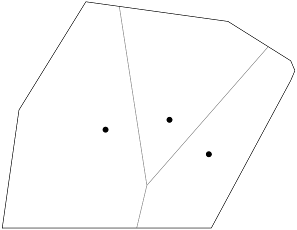
$$\text{minimize} \quad \mathcal{H}(p_1, \ldots, p_n) = E_\phi \left[\min_{i \in \{1,\ldots,n\}} \|q - p_i\|^2 \right]$$

Voronoi partitions

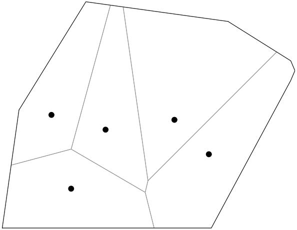
Let $(p_1, \ldots, p_n) \in S^n$ denote the positions of n points

The **Voronoi partition** $\mathcal{V}(P) = \{V_1, \ldots, V_n\}$ generated by $(p_1, \ldots, p_n)$

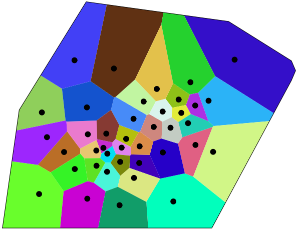
$$\begin{aligned} V_i &= \{q \in S | \ \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\} \\ &= S \cap_j \mathcal{HP}(p_i, p_j) \qquad \text{where } \mathcal{HP}(p_i, p_j) \text{ is half plane } (p_i, p_j) \end{aligned}$$



3 generators



5 generators



50 generators

Optimal configurations of $\mathcal{H}$

Alternative expression in terms of Voronoi partition,

$$\mathcal{H}(p_1, \ldots, p_n) = \sum_{i=1}^{n} \int_{V_i} \|q - p_i\|_2^2 \phi(q) dq$$

$\mathcal{H}$ as a function of agent positions and partition,

$$\begin{aligned}\mathcal{H}(p_1, \ldots, p_n, W_1, \ldots, W_n) &= \sum_{i=1}^{n} \int_{W_i} f(\|q - p_i\|_2) \phi(q) dq \\ &\leq \sum_{i=1}^{n} \int_{V_i} f(\|q - p_i\|_2) \phi(q) dq\end{aligned}$$

For fixed positions, **Voronoi partition is optimal**

For fixed partition, **centroid configurations are optimal**

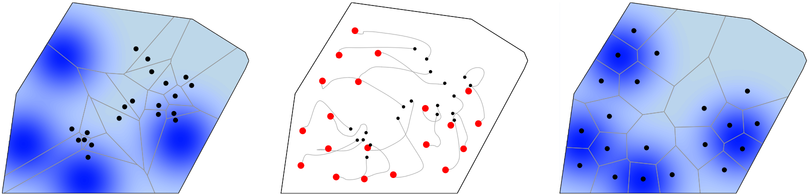
Centroid algorithm

[Cortes, Martinez, Karatas, Bullo '04]

At each round, agents synchronously execute:

- transmit position and receive neighbors' positions;
- compute centroid of own cell determined according to some notion of partition of the environment

Between communication rounds, each robot moves toward centroid



initial configuration   gradient descent   final configuration

Properties: provably correct, adaptive, distributed over Voronoi graph

Outline

1. Motivation
2. Problem Formulation
   - aggregate objective optimization
   - Voronoi partition
3. Triggered Deployment Algorithms
   - self-triggered deployment algorithm
   - team-triggered deployment algorithm
   - simulations
4. Conclusions

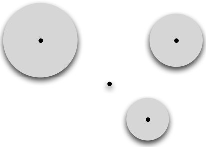
Trading computation for communication/sensing

Balance cost of up-to-date information with limited resources

what can agents do with outdated information about each other?

Agents have **uncertainty regions** on other agents

- how up-to-date information must be to positively contribute to task
- when information must be updated

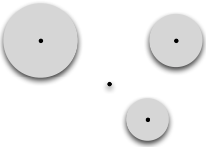


Trading computation for communication/sensing

Balance cost of up-to-date information with limited resources

what can agents do with outdated information about each other?

Agents have **uncertainty regions** on other agents



- how up-to-date information must be to positively contribute to task
- when information must be updated

Each agent i stores $\mathcal{D}^i = ((p_1^i, r_1^i), \ldots, (p_n^i, r_n^i))$,

- p_j^i : last known location of agent j
- r_j^i : maximum distance traveled by agent j since last info
- $p_i^i = p_i$ and $r_i^i = 0$

Agents move at max speed $v_{\max}$

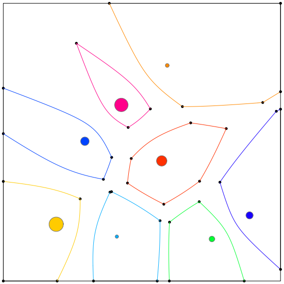
Guaranteed Voronoi diagram

Guaranteed Voronoi diagram $g\mathcal{V}(D_1, \dots, D_n) = \{gV_1, \dots, gV_n\}$ of S generated by $D_1, \dots, D_n \subset S$,

$$gV_i = \{q \in S \mid \max_{x \in D_i} \|q - x\|_2 \leq \min_{y \in D_j} \|q - y\|_2 \text{ for all } j \neq i\}$$

gV_i contains points guaranteed to be closer to any point in D_i than to any other point in D_j, $j \neq i$

In general, for $p_i \in D_i$, $gV_i \subset V_i$



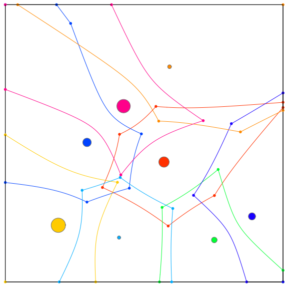
Dual guaranteed Voronoi diagram

Dual guaranteed Voronoi diagram $\mathrm{dg}\mathcal{V}(D_1, \dots, D_n) = \{\mathrm{dg}V_1, \dots, \mathrm{dg}V_n\}$ of S generated by $D_1, \dots, D_n \subset S$,

$$\mathrm{dg}V_i = \{q \in S \mid \min_{x \in D_i} \|q - x\|_2 \leq \max_{y \in D_j} \|q - y\|_2 \text{ for all } j \neq i\}$$

Points outside $\mathrm{dg}V_i$ are guaranteed to be closer to any point of D_j than to any point of D_i

In general, for $p_i \in D_i$, $V_i \subset \mathrm{dg}V_i$



When is motion good?

[Nowzari, Cortés '12]

With outdated info, agent i cannot calculate C_{V_i}

Proposition

Let $L \subset V \subset U$. Then, for any density function ϕ,

$$\|C_V - C_L\|_2 \leq \texttt{bound}(L, U) = 2\,\mathrm{cr}(U)\Big(1 - \frac{\mathrm{mass}(L)}{\mathrm{mass}(U)}\Big)$$

When is motion good?

[Nowzari, Cortés '12]

With outdated info, agent i cannot calculate C_{V_i}

Proposition

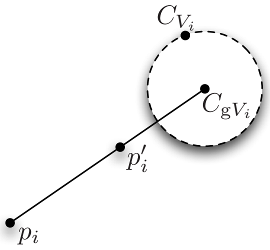
Let $L \subset V \subset U$. Then, for any density function ϕ,

$$\|C_V - C_L\|_2 \leq \mathit{bound}(L, U) = 2\,\mathrm{cr}(U)\Big(1 - \frac{\mathrm{mass}(L)}{\mathrm{mass}(U)}\Big)$$

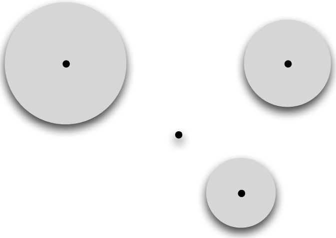
Agent i moves from p_i to p_i' making sure that

$$\begin{aligned}\|p_i' - C_{\mathrm{g}V_i}\|_2 &\geq \mathtt{bound}_i = \mathtt{bound}(\mathrm{g}V_i, \mathrm{dg}V_i) \\ &\geq \|C_{V_i} - C_{\mathrm{g}V_i}\|_2\end{aligned}$$

move towards $C_{\mathrm{g}V_i}$ as much as possible in one time step until it is within distance $\mathtt{bound}_i$ of it. As time elapses without new info, $\mathtt{bound}$ grows



Triggered coordination algorithms



Reachable sets

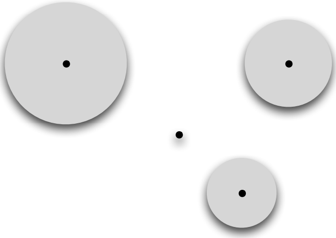
`self-triggered centroid algorithm` combines

- motion law
- self-triggered update policy (requesting information)

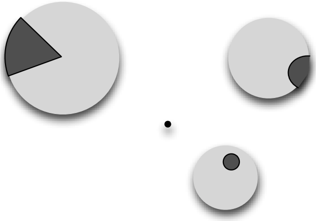
Proposition

Set of Centroidal Voronoi Configurations is globally asymptotically stable

Triggered coordination algorithms



Reachable sets



Promise sets

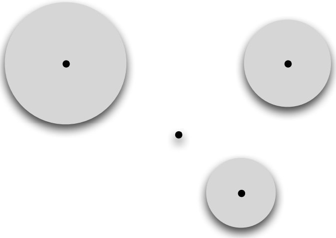
`self-triggered centroid algorithm` combines

- motion law
- self-triggered update policy (requesting information)

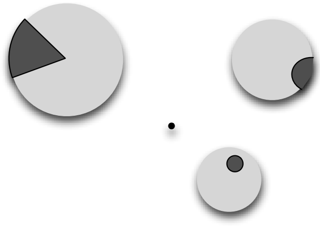
Proposition

Set of Centroidal Voronoi Configurations is globally asymptotically stable

Triggered coordination algorithms



Reachable sets



Promise sets

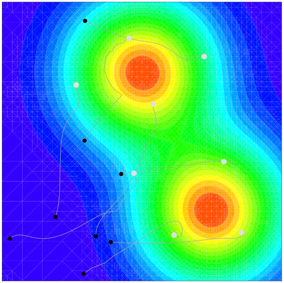
`team-triggered centroid algorithm` combines

- motion law
- self-triggered update policy (requesting information)
- event-triggered update policy (broken promises)

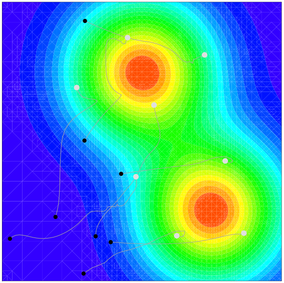
Proposition

Set of Centroidal Voronoi Configurations is globally asymptotically stable

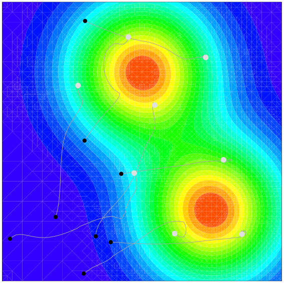
Simulations



Periodic



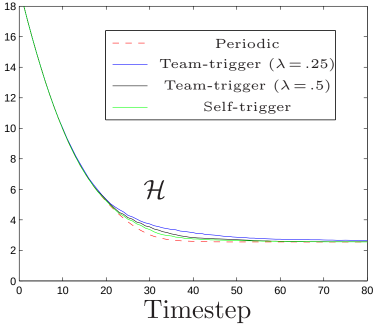
Self-triggered

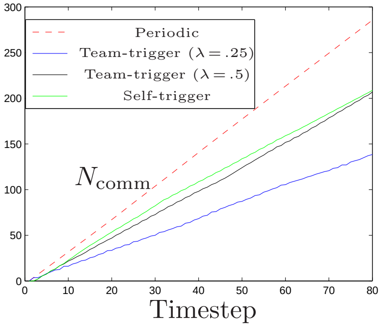


Team-triggered

$dBmW$ power units:

$$\mathcal{P}_i = 10 \log_{10} \left[\sum_{i,j \text{ comm}}^{n} 10^{0.1 + \|p_i - p_j\|_2} \right]$$





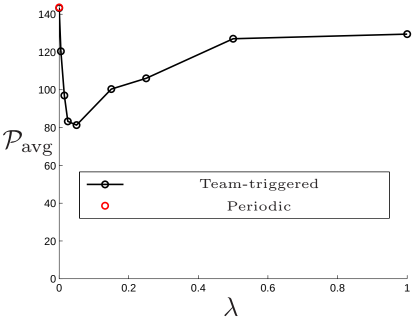
Simulation

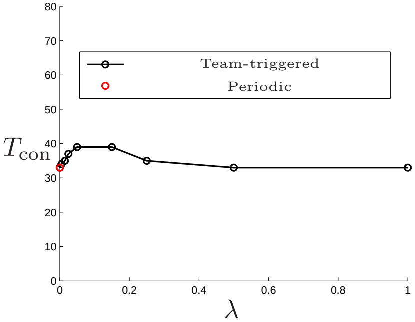
Effect of varying promise sizes

λ captures the **tightness** of promises

$\lambda = 0$ corresponds to exact trajectories for promises

$\lambda = 1$ corresponds to no promises (recovers self-triggered case)





Conclusions

Team-triggered deployment of robotic networks for optimal deployment

- `team-triggered centroid algorithm`
- correct, adaptive, distributed, asynchronous
- same **convergence guarantees** as synchronous algorithm with perfect information at all times
- **reduced** communication efforts throughout the network

Conclusions

Team-triggered deployment of robotic networks for optimal deployment

- `team-triggered centroid algorithm`
- correct, adaptive, distributed, asynchronous
- same **convergence guarantees** as synchronous algorithm with perfect information at all times
- **reduced** communication efforts throughout the network

Things I skipped:

- how agents update information
- guaranteeing **no Zeno behavior**
- maximum times without communication

Thank you!