Agent-Based Decentralized Coordination for Sensor Networks Using the Max-Sum Algorithm

A. Farinelli | A. Rogers | N. R. Jennings

Ryan Erdmann | Michael Hollman | Scott Johnson

Citation

A. Farinelli, A. Rogers, and N. R. Jennings (2014). Agent-based Decentralised Coordination for Sensor Networks Using the MaxSum Algorithm, *Autonomous Agents and Multiagent Systems*, **28**(3):337-380.

Introduction

Michael Hollman

The Premise

- Use network of sensors to observe large environment
- Examples:
  - Animal tracking
  - Environmental phenomena monitoring
  - This paper: automobile traffic monitoring
- Devices' characteristics and environment impose some constraints

Limitations

- Sensor location/distribution not guaranteed
- Sensor and communication reliability not guaranteed
- Primitive nature of sensor devices
- Devices cannot be active 100% of the time
  - They use energy faster than they can harvest it
  - Overcome this with *duty cycles* (sense/sleep schedules)

The Problem from 10,000 Feet

How can a network of sensor-laden devices optimally schedule their own duty cycles to maximize the system’s effectiveness at observing their environment?

Modelling the Problem

- Represent each sensor as an *autonomous agent*
  - Learn interactions with neighbors
  - Coordinate sense/sleep schedules
- One model: Markov decision problem variations (Dec-MDP, POMDP)
  - Collaborate with peers to learn a joint action policy
  - Methods don't scale when agents have many neighbors

Modelling the Problem

- Constraint Networks
  - Nodes represent agents
  - Edges represent constraints that arise between the agents

- Hard Constraints → Constraint Satisfaction Problem (DCSP)
- Soft Constraints → Constraint Optimization Problem (DCOP)

Distributed Constraint Optimization Problems

- Finding an exact solution to DCOP is an NP-hard problem
  - Algorithm grows exponentially
  - Require a holistic understanding of the environment and/or network
  - Requires preprocessing, so the system is not flexible to changes
  - Too computationally complex for extremely low-powered devices
- Approximate algorithms often converge to poor-quality solutions

- Thus, there is a need for an efficient, decentralized coordination algorithm

The Max-Sum Algorithm

- A generalized distributive law (GDL) framework algorithm
  - Used in information theory, AI, statistical physics
- Can be exploited to find good approximate solutions to DCOPs
- Was tweaked to be used by communicative agents to converge to *neighborhood maximums*

The Max-Sum Approach To Coordination

Ryan Erdmann

The Max-Sum Algorithm

- Max-Sum decomposes complex calculations by “factorizing” it
- Uses a combination of the best features of both the optimal and stochastic algorithms

- Built on previous work
  - Factoring agent interactions (action selection) into a coordination graph
  - Max-Sum has been previously used for agent coordination, operating on the coordination graph

Contribution to the State of the Art

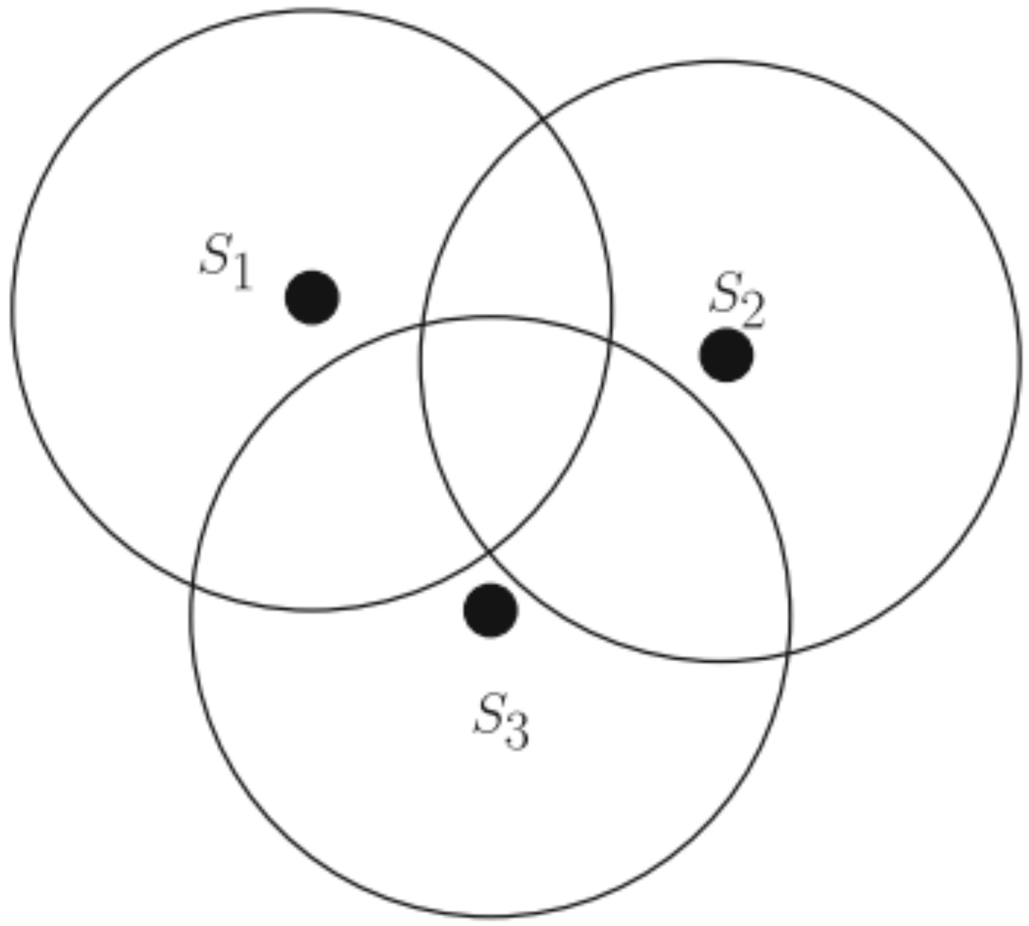
- Using the *factor graph* to
  - Model agent interactions
  - Provide an operational framework for the max-sum algorithm that defines the message passing protocol
- Provides a distributed, near-optimal solution to DCOP

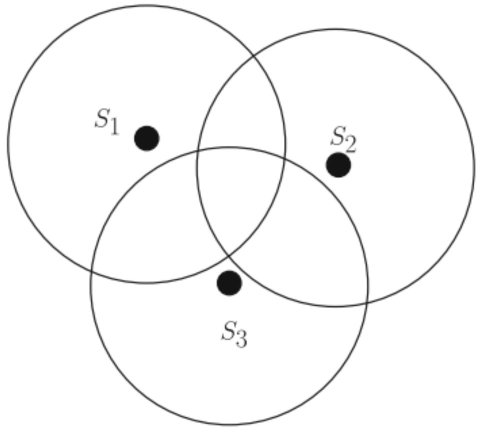
DCOP Formalization

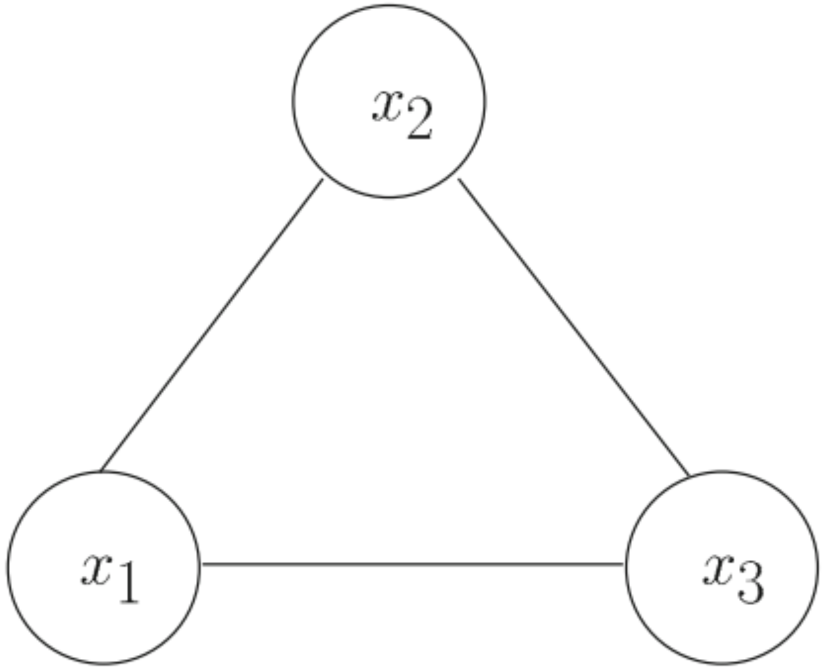
- $DCOP = \langle A, X, D, F \rangle$
  - $A = \{a_1, a_2, \ldots a_m\}$ set of agents
  - $X = \{x_1, x_2, \ldots x_s\}$ set of variables
  - $D = \{D_1, D_2, \ldots D_s\}$ set of domains
  - $F = \{F_1, F_2, \ldots F_n\}$ set of constraints
- Goal: $\operatorname{argmax}_x \sum_i F_i(\boldsymbol{x_i})$
- Recall that this is an NP-hard problem

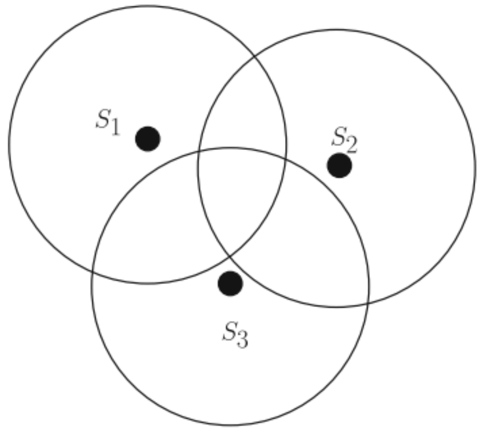
Factor Graph Representation

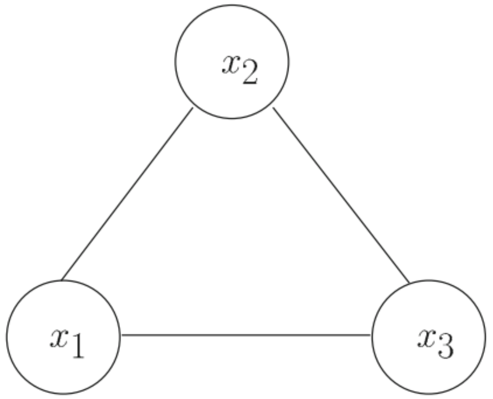
- Bipartite graph comprised of
  - Variable nodes
  - Function nodes
- Function choice depends on system decomposition
- Explicitly models a *k*-ary relationship between agents

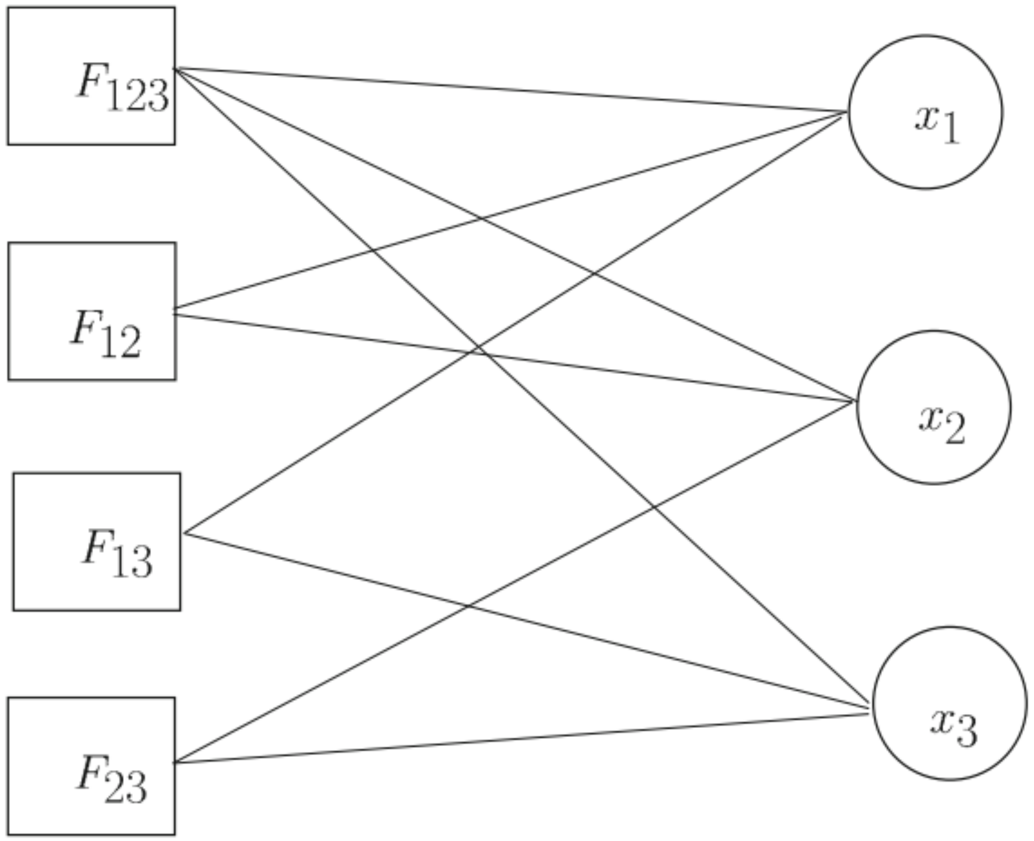

S_1
S_2
S_3


s_1
s_2
s_3


x_2
x_1
x_3

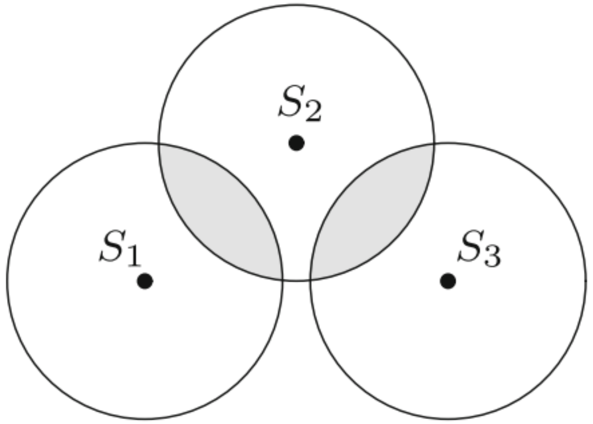

s_1
s_2
s_3


x_2
x_1
x_3


F_{123}
F_{12}
F_{13}
F_{23}
x_1
x_2
x_3

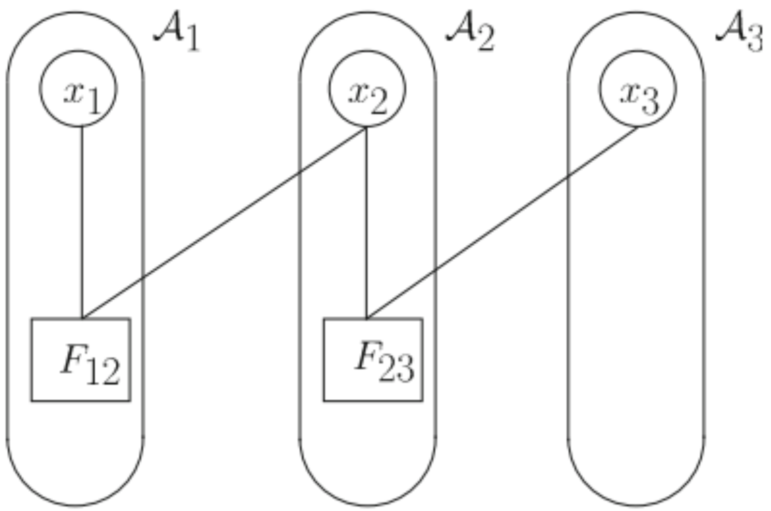
Factor Graph Decomposition

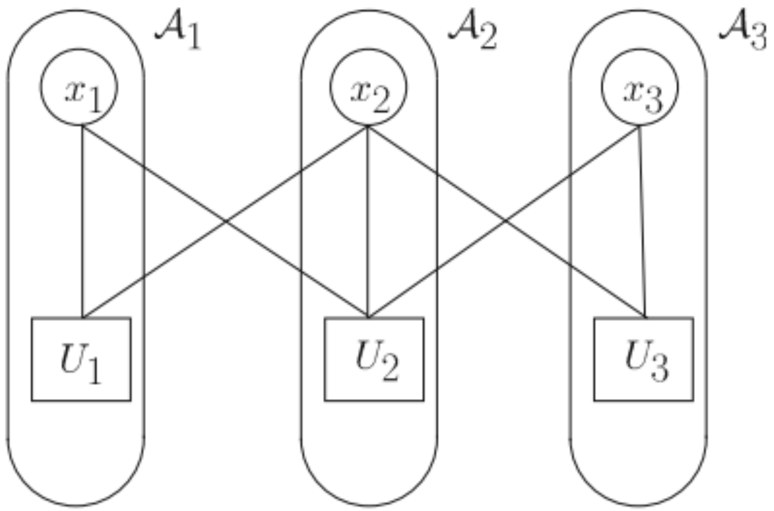
- Interaction-based
  - Functions represent interactions of neighboring agents
  - One function node per constraint
  - Agents must negotiate for ownership of shared functions
- Utility-based
  - Each agent's function represents their own utility, and the sum of all agents' functions is equal to the objective function
  - Agents can negotiate continuously - dynamic
  - Creates cycles, which impacts optimality of max-sum



Interaction-Based

Utility-Based





The Max-Sum Algorithm

- The algorithm iteratively passes messages back and forth between connected variables and functions
- **These messages are functions that estimate the total utility of the network for a possible variable assignment**
- Each agent makes local decisions to maximize the sum over all agents' utilities.

The Max-Sum Algorithm

- Message: From Variable i to Function j

$$q_{i\to j}(x_i) = \alpha_{ij} + \sum_{k\in\mathcal{M}_i\backslash j} r_{k\to i}(x_i)$$

- Message: From Function j to Variable i

$$r_{j\to i}(x_i) = \max_{\mathbf{x}_j\backslash i}\left[U_j(\mathbf{x}_j) + \sum_{k\in\mathcal{N}_j\backslash i} q_{k\to j}(x_k)\right]$$

- Local Decision to maximize:

argmax┬𝑥↓𝑖
𝑧↓𝑖(𝑥↓𝑖)

$$z_i(x_i) = \sum_{j\in\mathcal{M}_i} r_{j\to i}(x_i)$$

Message Update Scheduling

- There is no need for a strict ordering or synchronization of messages
- Agents update their variable assignment when they receive new messages.
- Note: the variable assignment does not impact message computation
- So, agents are asynchronous & continuously updating

Algorithm Termination

- Three possible outcomes:
  - Variable assignment and messages converge
  - Assignment converges, but messages do not
  - Neither assignment nor messages converge
- Termination Rules:
  - Continue to propagate messages until they converge
  - Propagate messages for a fixed number of iterations

Max-Sum Solution Quality

- When cycles exist, it's impossible to guarantee convergence or quality of the resulting solution
- Empirical evidence suggests that max-sum will converge and provide near-optimal variable assignments
- Quality is dependent on the structure of the factor graph; future work may look at transforming areas of the graph with many cycles

Coordination Overhead

- For each iteration, the agent broadcasts messages to all connected variables and functions
- Alternative DCOP methods optimize who to broadcast to

- Benefits:
  - It's possible to add or remove agents to the system
  - The multiagent system is resilient to message transmission failures

The Wide Area Surveillance Problem

Michael Hollman

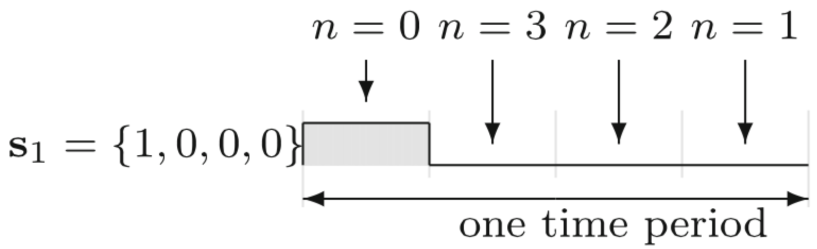
Problem Formalization

- *M* sensors
- Each sensor has a set of possible schedules x_i
- Agent interaction produces a set of functions that depend on a subset of sensors $\mathbf{x}_i$
- We want to find $\mathbf{x}^*$, the subset of sensors that maximizes utility (social welfare)

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_{i=1}^{M} U_i(\mathbf{x}_i)$$

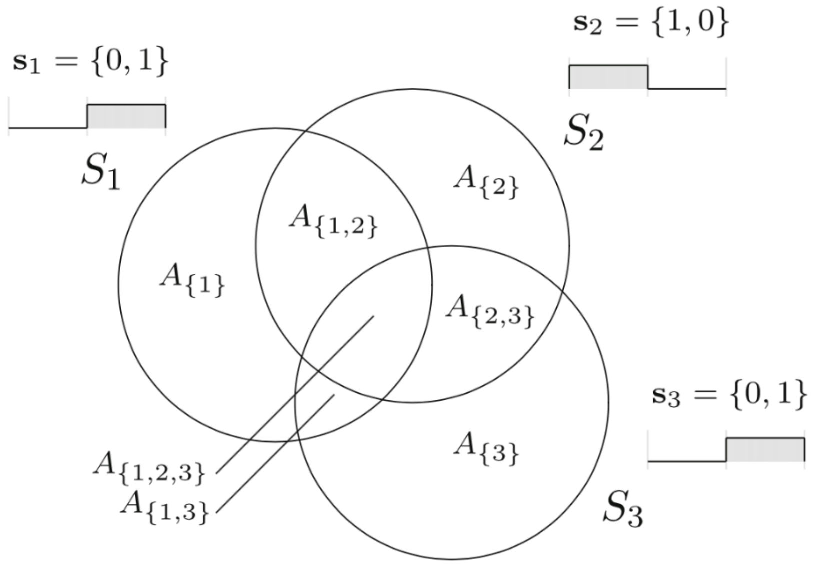
Problem Formalization

- Sensors and event distribution are both assumed to be Poisson
  - λ_s - sensors expected in a unit area
  - λ_d - event duration
  - $\lambda_d e^{-\lambda_d t}$ – probability of an event lasting time t
- Sensors follow a schedule defined by the set s_i
  - L – length of s_i (same for all sensors)
  - All schedules are discrete and “tick” together



Coordination

- System goal is to maximize the probability that any given event is detected by a sensor
- Smells like a combinatorial optimization problem
  - Similar to graph coloring
- Key difference: multiple (>2) sensors can communicate together



Theoretical Analysis

1. Continuously powered sensors
   - Represents absolute upper bound.
2. Synchronized sensors
   - All sensors sense in the same one time slot; no adaptation.
3. Randomly coordinated sensors
   - Each sensors senses in one random time slot; no coordination.
4. Optimally coordinated sensors
   - Each sensor senses in one coordinated time slot

For each case, calculate the probability of event detection (PED)

Theoretical Analysis

1. Continuously powered sensors

$$PED_{\text{continuous}} = 1 - \mathrm{e}^{-\lambda_s \pi r^2}$$

2. Synchronized sensors

$$PED_{\text{synchronised}} = \left(1 - \mathrm{e}^{-\lambda_s \pi r^2}\right) \sum_{n=0}^{L-1} \begin{cases} 1/L & n = 0 \\ \frac{\mathrm{e}^{-\lambda_d n/L}}{\lambda_d} \left(\mathrm{e}^{\lambda_d/L} - 1\right) & n \geq 1 \end{cases}$$

Theoretical Analysis

3. Randomly coordinated sensors

Algorithm 5 $PED_{\text{random}}(\lambda_s, \lambda_d, r, L)$

```
1: $value \leftarrow 0$
2: for $m = 1$ to $\infty$ do
3:     $P(m) = (\lambda_s \pi r^2)^m e^{-\lambda_s \pi r^2} / m!$
4:     for $\mathbf{s} \in \mathcal{S}$ do
5:         $n \leftarrow \sum_{k=0}^{L-1} s_k$
6:         if $n \leq m$ then
7:             $P(\mathbf{s}) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} (n-k)^m / L^m$
8:             $value \leftarrow value + P(m) \times P(\mathbf{s}) \times P(\text{detection}|\lambda_d, \mathbf{s})$
9:         end if
10:    end for
11: end for
12: return $value$
```

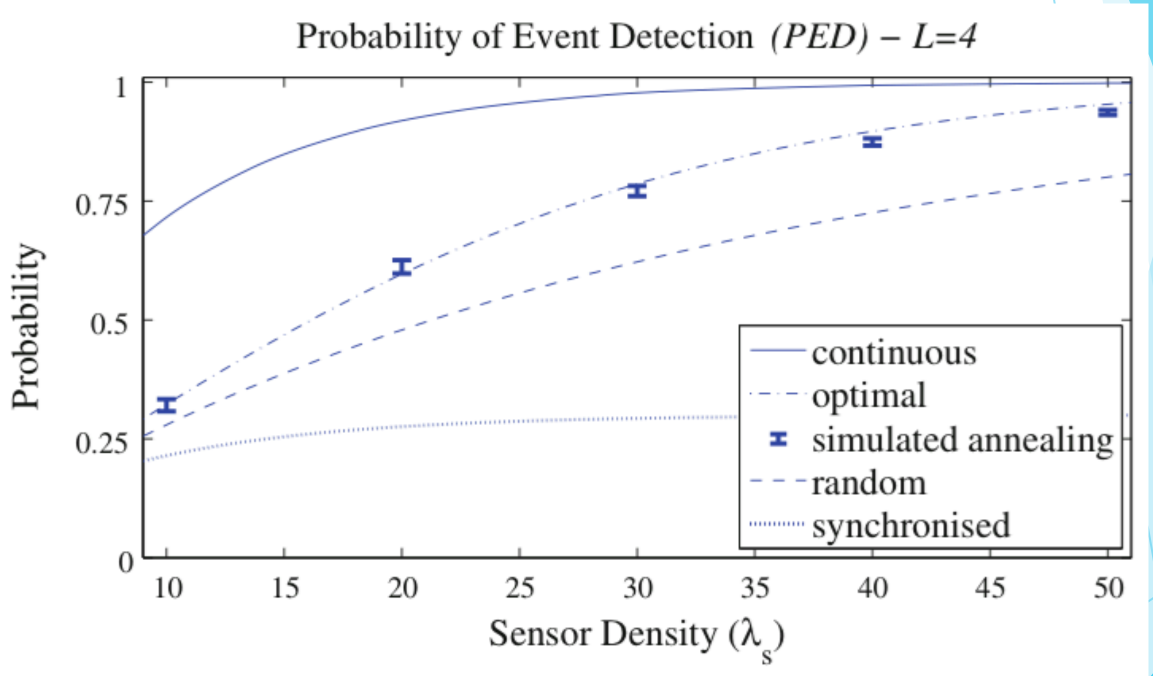
4. Optimally coordinated sensors

Algorithm 6 $PED_{\text{optimal}}(\lambda_s, \lambda_d, r, L)$

```
1: $value \leftarrow 0$
2: for $m = 1$ to $\infty$ do
3:     $P(m) = (\lambda_s \pi r^2)^m e^{-\lambda_s \pi r^2} / m!$
4:     if $m < L$ then
5:         $value \leftarrow value + P(m) \times P(\text{detection}|\lambda_d, \mathbf{s}^*_{m,L})$
6:     else
7:         $value \leftarrow value + P(m)$
8:     end if
9: end for
10: return $value$
```

Enough of that…

- Bottom line is that optimally coordinated sensors are… optimal!
- E.g.: when $L = 4$, $\lambda_s > 35$
  - Optimal schedule will detect 50% of what the random schedule missed
  - Optimal schedule can perform as well as random schedule with 40% fewer sensors



Decentralized Coordination for the Wide Area Surveillance Problem

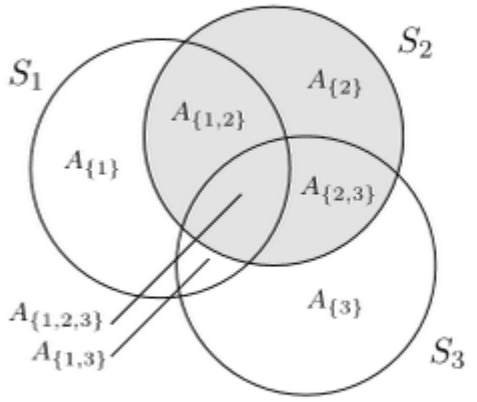
Scott Johnson

Application

- The authors tested the theoretical findings using the RoboCup simulation environment.
- Energy-constrained sensors are placed randomly throughout the environment to detect vehicles.
- Sensors have no knowledge of their location or of the road layout.
- The radius of each sensor's field is independently assigned and comes from a uniform distribution between 0.05 and 0.15 times the maximum dimension of the environment.

Utility

- Must transition from the theoretical global goal of maximizing event detection to individual utility functions.
- Basic premise behind a practical utility function:
  - Sum over all subsections of the sensors range the product of the area of the region and the probability of detecting an event in that region given the schedule divided by the total number of sensors overlapping in that region.

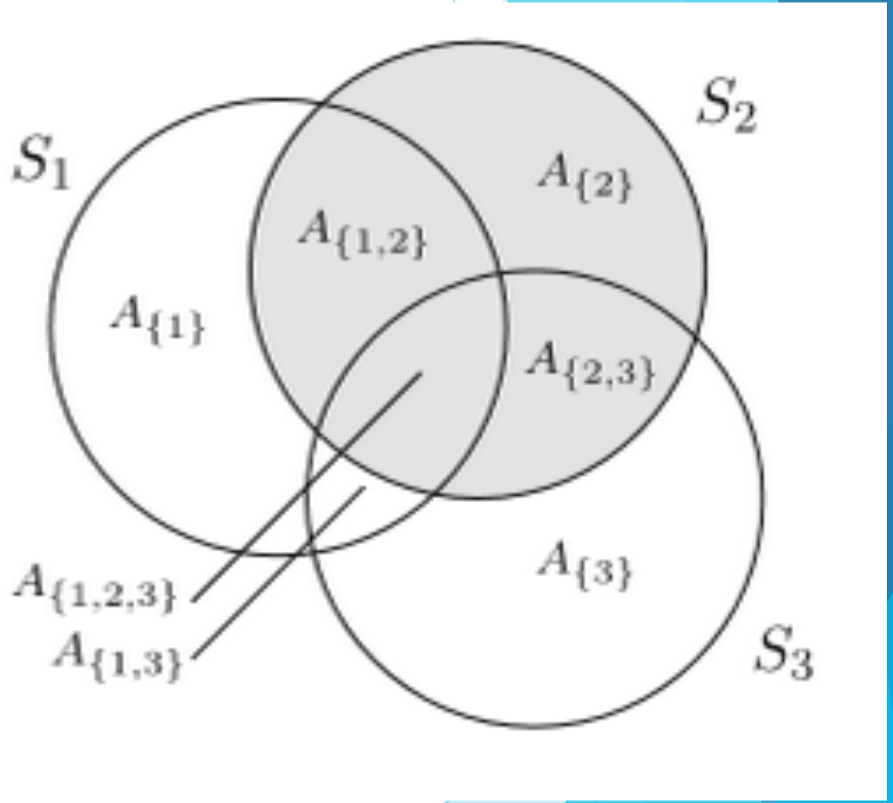


Utility Revisited

- Previous utility formula is still not general enough for practical application:
  - Sensors may not know the exact location of the other sensors.
  - Sensors may not know where the overlap exists.
  - Some regions may have a higher probability of an event occurring.
  - Sensing regions may be irregular.
- Solution: additional calibration phase.

Calibration Phase

- Occurs before any coordination is done.
- All sensors follow the same sense/sleep schedule and record the number of vehicles it sees and the number of vehicles it and some subset of its neighbors see.
  - Uses times of appearance and disappearance of cars to determine when neighboring sensors have sensed the same vehicle.

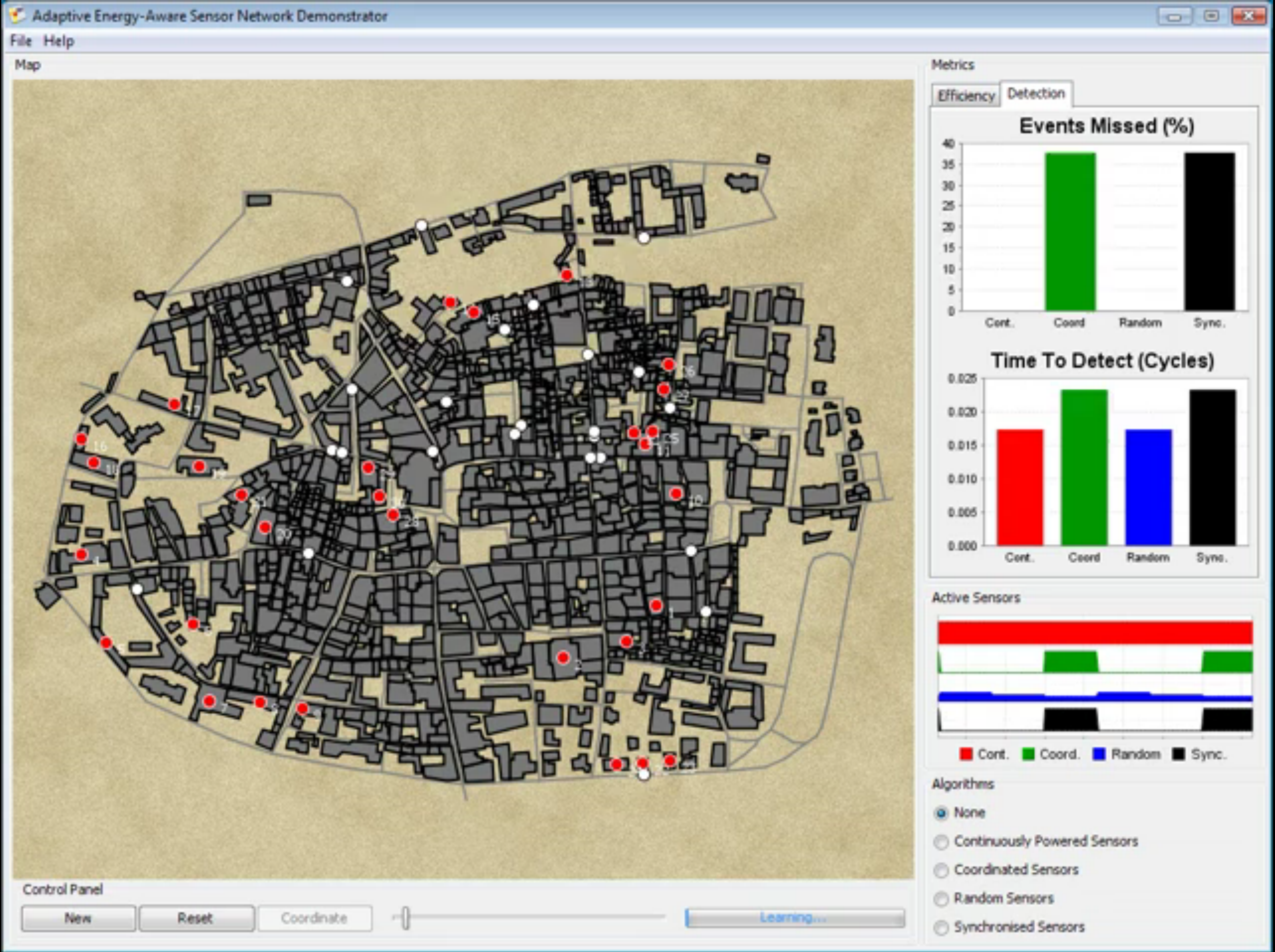


Coordination Phase

- Uses the results from the calibration phase to run the max-sum algorithm.
- The previous utility function is revised so that instead of using area, the utility function substitutes in the number of vehicles that crossed the section in the calibration phase.

Operational Phase

- The sensors follow the sense/sleep schedule that results from the coordination phase.
- The results from this stage form the basis of comparison for the various methods.



Algorithms

[✓ None]
Continuously Powered Sensors
Coordinated Sensors
Random Sensors
Synchronised Sensors

Experimentation

- Four different approaches were tested in order to compare the effectiveness of each approach:
  - Randomly Coordinated Sensors
  - Distributed Stochastic Algorithm (DSA) Coordinated Sensors
  - Max-sum Coordinated Sensors
  - Simulated Annealing Coordinated Sensors
- Each method was also compared with an upper bound scenario of having the sensors be continuously on.

Randomly Coordinated Sensors

- Each sensor randomly selects its sense/sleep schedule
- This method does not involve any coordination

DSA Coordinated Sensors

- Each node repeats the following until a termination condition is met:
  - In each iteration the node has a probability P (in this case 0.6) of executing the following optimization.
  - Find the variable value that results in the sensors maximum utility.
  - If this value is different than the previous computed value, send the new value out to all neighbors.
  - At the end of each iteration receive the values sent by other nodes.

Simulated Annealing Coordinated Sensors

- A centralized algorithm to compute an optimal or almost optimal global solution.
- Not practical because it assumes full knowledge of the sensor network.
- Used as a benchmark for judging the other methods.

Neighbor Reduction

- In order to reduce the computational complexity, the authors put a cap on the maximum number of neighbors the coordination algorithm would consider.
- Each neighbor would rank its neighbors based on the amount of overlap it had in the calibration phase.
- For the coordination algorithm, each sensor only considered the n neighbors with the most overlap.
- In the experiments, the authors used 4 as the maximum number of neighbors for both the max-sum and DSA algorithms.

Results: Calibration

- The first experiment tested the effect of varying the number of vehicles paths that occurred during the calibration phase.
- The more paths in the calibration phase, the better the results.
- For all coordination mechanisms, the added benefit from extra paths was similar.
- The authors decided to use 1000 paths in the calibration phase of the rest of the experiments.

Results: Length of Sensing Schedule

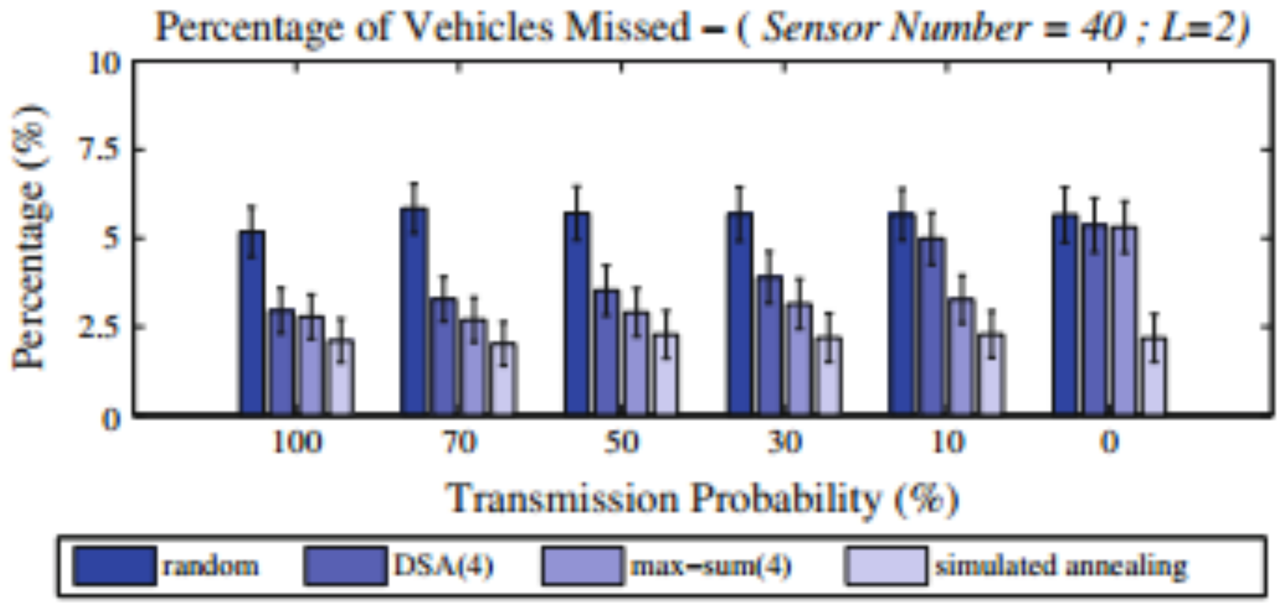
- The authors varied the length of the sensing schedule (L) while keeping the average number of sensors in each schedule unit constant.
- The more periods/sensors, the better the performance of the network.
- Coordination has a higher impact on the system as the length of the sensing schedule increases

Results: DSA vs. Max-Sum

- In the experiments, Max-Sum generally performed slightly better than DSA coordination, but the two methods were generally comparable.
- However, as the sensing schedule length (L) and the number of sensors (N) went up, max-sum became significantly better than DSA.
  - In some experiments, max-sum beat DSA by 35%.
- Although max-sum provided slightly better performance, it also requires more computational overhead.

Results: Message Reliability

- Measured the affect of a non-perfect messaging system between the sensors.
- The experiments showed that as the messages became less reliable, the performance of the DSA coordination went down significantly while the performance of the max-sum coordination stayed relatively constant.



Conclusions and Insights

Ryan Erdmann

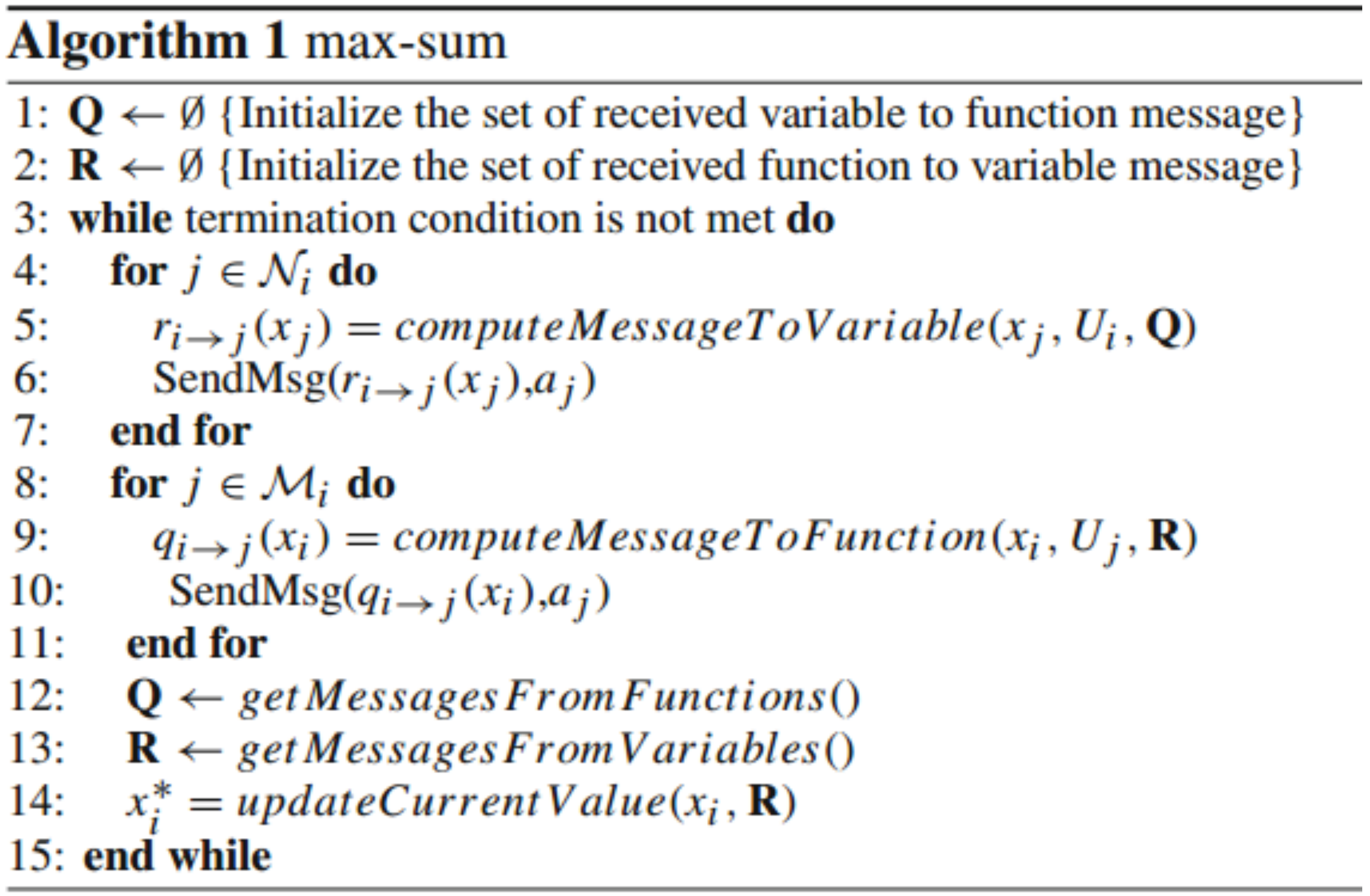
Key Conclusions

- The max-sum algorithm can be applied to the factor graph to find near-optimal solutions to the DCOP problem
- Coordination can yield significant gains in system-wide performance
- Max-Sum has comparable performance to DSA, but is superior when sensor overlap is large or transmission probability decreases

Key Insights

- There is a tradeoff between the performance of the algorithm and the coordination overhead.
  - Future work can be done to transform the factor graph and use intelligent clustering to overcome this tradeoff
- The current algorithm requires 3 distinct phases, which prevents the multiagent system from being truly dynamic
  - Relax the calibration phase by using collaborative and reinforcement learning protocols to train the system online
  - A Bayesian Reinforcement Learning approach would remove all 3 phases, and sensors would truly self organize and adapt

Questions?



Algorithm 1 max-sum

```
1: $\mathbf{Q} \leftarrow \emptyset$ {Initialize the set of received variable to function message}
2: $\mathbf{R} \leftarrow \emptyset$ {Initialize the set of received function to variable message}
3: while termination condition is not met do
4:    for $j \in \mathcal{N}_i$ do
5:       $r_{i\rightarrow j}(x_j) = computeMessageToVariable(x_j, U_i, \mathbf{Q})$
6:       SendMsg($r_{i\rightarrow j}(x_j)$,$a_j$)
7:    end for
8:    for $j \in \mathcal{M}_i$ do
9:       $q_{i\rightarrow j}(x_i) = computeMessageToFunction(x_i, U_j, \mathbf{R})$
10:      SendMsg($q_{i\rightarrow j}(x_i)$,$a_j$)
11:   end for
12:   $\mathbf{Q} \leftarrow getMessagesFromFunctions()$
13:   $\mathbf{R} \leftarrow getMessagesFromVariables()$
14:   $x_i^* = updateCurrentValue(x_i, \mathbf{R})$
15: end while
```