EXPLORATION OF COLLECTIVE PERCEPTION IN THE PRESENCE OF LYING AGENTS

January 3, 2019

Shubham Jain - Yasmina Benkhoui - Sanket Gujar
Swarm Intelligence - Spring 2018

REFERENCES

[1] Robust distributed decision-making in robot swarms: Exploiting a third truth state M Crosscombe, J Lawry, S Hauert, M Homer IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

[2] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas and V. Kumar, Resilient Flocking for Mobile Robot Teams, IEEE Robotics and Automation Letters, Vol. 2, No. 2, pp. 1039-1046, April 2017.

How can we achieve consensus in swarm in the presence of lying individuals?

Environment

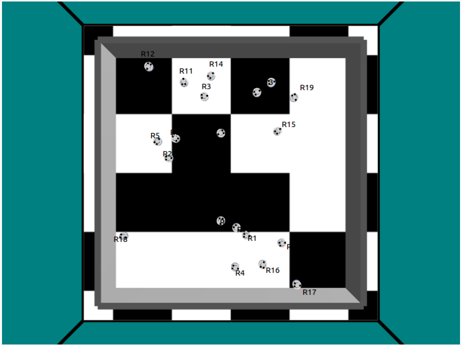


Figure: Environment

Each robot has ground sensor so it can read the color of the tile beneath it. The robot also knows its position in the grid with the help of position sensor.

Terminology

Templates - Patterns for which robot maintains a belief value

Belief vector - Has dimensions equal to number of templates and always sums up to 1

Messages - Robots broadcast their belief vectors to neighbours

Sensor update - Robot compares sensor value with template, adds noise of 0.2 and makes update to belief

Neighbour update - Update using neighbour's beliefs (Depends on algorithm)

First case study: Basic pattern recognition

The problem is calculating the probability of a certain pattern knowing the position of the robot, their observation and the received messages at a certain time t

$$p(y^l | z_{1:t}, x_{1:t}, m_{1:k}^{l:t})$$

y^l	l th pattern
x_t	Robot position at time t
z_t	Robot's observation at time t
m_k^t	k^{th} message received at time t

First case study: Basic pattern recognition

Algorithm:

We start with all the robots having equal beliefs about all the patterns.

We update the beliefs as following:

$$\mathit{CurrentBelief} = \mathit{OldBelief} \times \mathit{MsgBrodcastNeigh}$$

We normalize after each update

Algorithm 1 Decentralized Pattern Detection

for `no of iterations` **do**
    for `every robot` **do**
        $neighbelief_t \leftarrow$ **receive()**
        for `all patterns` **do**
            $belief_t^l \leftarrow (\prod_i^k neigbelief_t^{l,i}) belief_{t-1}^l$
            $belief_t^l \leftarrow p(z_t | y^l, x_t) \overline{belief_t^l}$
        $belief_t \leftarrow \mathbf{normalize}(belief_t^l)$
        $\mathbf{broadcast}(belief_t)$

Results of the first case study

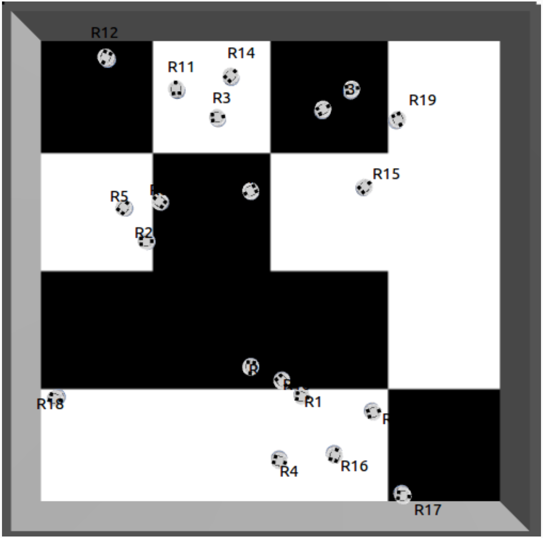


Figure: Initial state

Results of the first case study

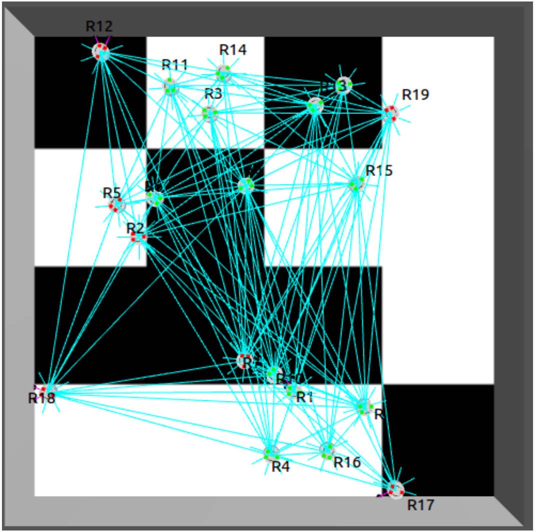


Figure: After two time-steps

Results of the first case study

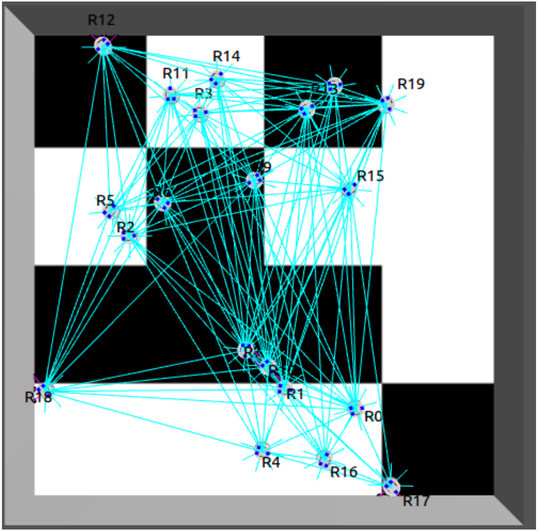


Figure: After six time-steps

First case study: Basic pattern recognition

Result

The algorithm achieves consensus in 3 to 4 steps: all the robots agree on the right pattern.

Lying Individual

But what will happen if we introduce a lying individuals to the swarm?

Lying Individual

Lying Individual

A lying individual is a robot broadcasting a false belief because it may have a bug or because it may be deliberately designed by a malicious hacker.

For our case - Lying individual continuously transmits belief of 0 for right pattern

Result of introducing a lying individual

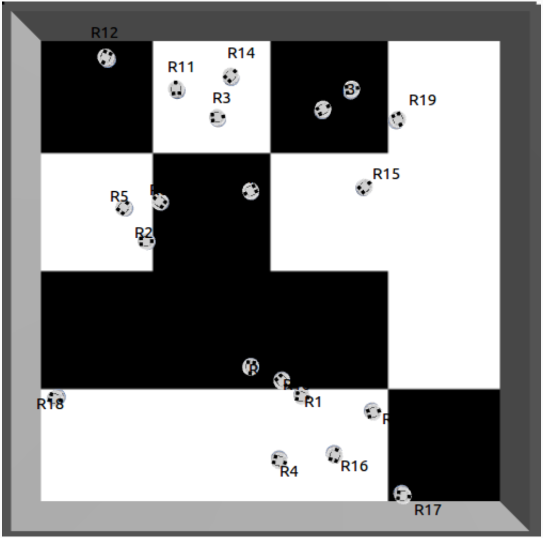


Figure: Initial State

Result of introducing a lying individual

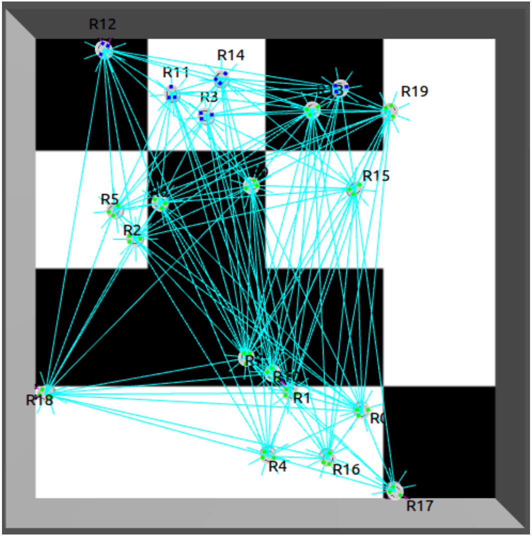


Figure: 3 steps after the introduction of a lying individual

Result of introducing a lying individual

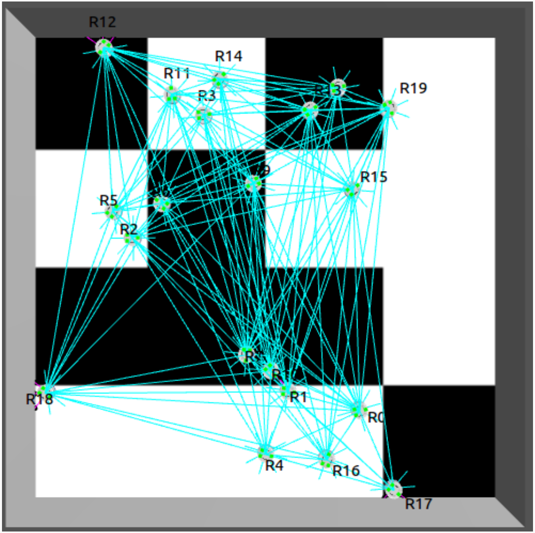


Figure: 4 steps after the introduction of a lying individual

2nd case study: Using Average Beliefs

Algorithm:

We introduce a lying individual and we update the belief by using the average of neighbours beliefs:

$$\mathit{CurrentBelief} = \frac{\mathit{OldBelief} \times \sum \mathit{NeighBelief}}{\mathit{NumNeighb}}$$

Algorithm 2 Decentralized Pattern Detection(Avg. Belief)

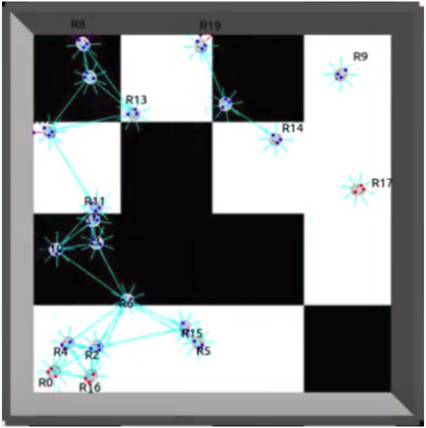
for `no of iterations` **do**
    for `every robot` **do**
        $neighbelief_t \leftarrow$ **receive()**
        for `all patterns` **do**
            $neighavg \leftarrow ((\sum_i^k neigbelief_t^{l,i})/neigh)$
            $\overline{belief_t^l} \leftarrow (neighavg) belief_{t-1}^l$
            $belief_t^l \leftarrow p(z_t|y^l, x_t)\overline{belief_t^l}$
        $belief_t \leftarrow$ **normalize**$(belief_t^l)$
        broadcast$(belief_t)$

Result of using average beliefs- Low range

Result: Depending on the range we specify, the robots converge on a pattern or they do not

Range = 1

Average with low range

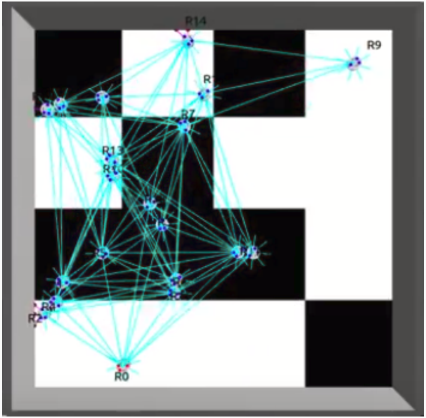


Result of using average beliefs- Medium range

Result: Depending on the range we specify, the robots converge on a pattern or they do not

Range = 2

Average with medium range

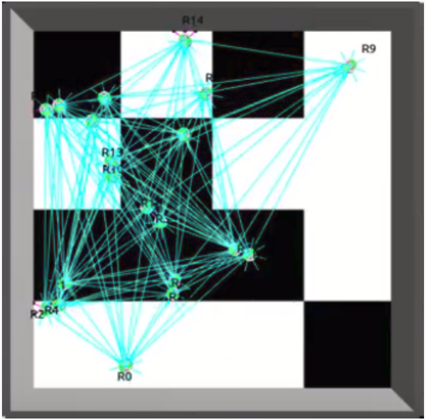


Result of using average beliefs- High range

Result: Depending on the range we specify, the robots converge on a pattern or they do not

Range = 3

Average with high range



3rd Case study: Similarity Subsequence Reduced

Assumption - Number of lying individuals (n) is known

Inspiration - Resilient Flocking [2] presented in class

For each robot, we calculate the cosine similarity of its belief vector with its neighbour

$$cos\theta_i = \frac{\sum a \times b_i}{\|a\| \times \|b_i\|}$$

Remove n neighbors having lowest similarity from neighbor's list before performing update

Improvement - Need to disregard only n neighbours instead of 2n neighbours as in [2]

Video

Similarity Subsequence Reduced - Results

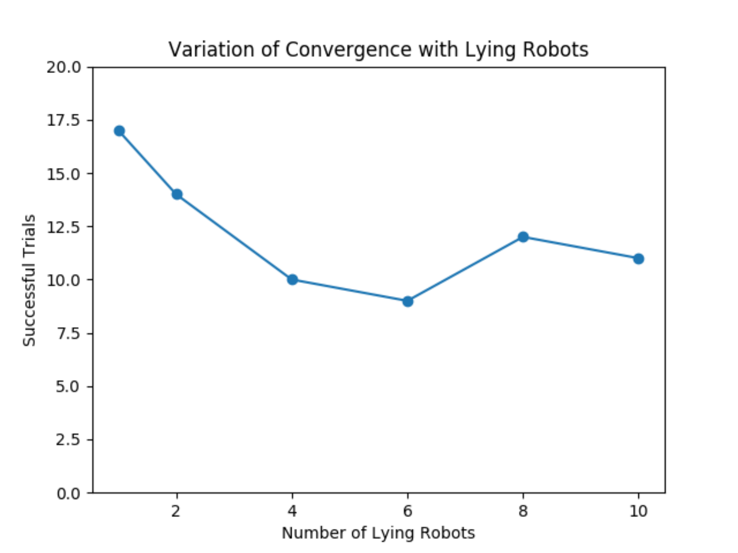


Figure: Varying number of liars with communication range fixed at 2

Similarity Subsequence Reduced - Results

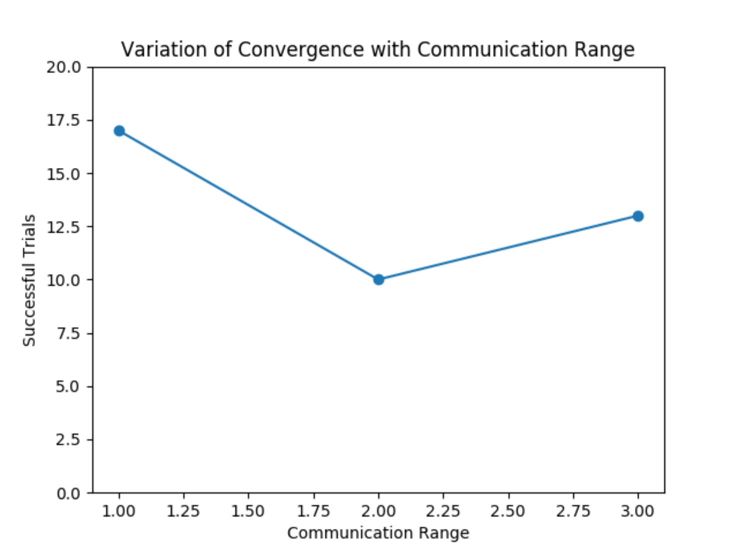


Figure: Varying communication range with number of liars fixed at 4

Conclusion

3 different algorithms tried out for decentralized decision making

- Basic Algorithm - Does not work at all with presence of lying robots
- Using averaged beliefs - Fails for high communication range
- Similarity Subsequence reduced - More robust to changes in communication range

Performed experiments with 20 trials each to explore effect of following on SSR -

- Number of lying robots
- Communication Range

We concluded that our Similarity Subsequence Reduced algorithm is more robust than others and converges to the right pattern most often.

Future Work

More experiments to vary parameters like - Robot density, number of templates and difficulty of templates

Explore effect of robot position initialization in the pattern.

We can try to not communicate during the initial time steps, letting the robots first build their belief and then start communicating.

Thank You

Questions?