Should we Compete or Should we Cooperate? Applying Game Theory to Task Allocation in Drone Swarms

Presented By : Chandima Fernando

Information On the Paper

- Juan Jesus Roldan , Jaime del Cerro , Antonio Barrientos are the authors
- Centre for Automation and Robotics (UPM- CSIC), Technical University of Madrid
- Presented in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Madrid, Spain, October 1-5, 2018

Plan of the talk

- Introduction
- Related Background
- Competitive Algorithm
- Cooperative Algorithm
- Experiments and Results
- Conclusion

Introduction – Set Up

- Swarms have a objective
  - Vising a location
  - Taking pictures
  - Building a map
- Communication limitations exist
- Task : Find a local task allocation that can be merged into a suitable global task allocation

Goal of the Paper

- Evaluate Nash equilibrium based competitive strategy vs voting based cooperative strategy for task allocation in a robotic swarm

Modification to previous work

Two major improvements over previous work

- Every robot with the same number of connections
- Use of genetic algorithms for real-time agents with large fleets

Evaluation

- Completed Tasks CT

$$U_{R_i,T_j} = \begin{cases} 0, & \text{if } \exists R_k : R_k \rightarrow T_j \\ d_{max} - d_{R_i,T_j} + 1, & \text{otherwise} \end{cases}$$

- Social Utility SU

$$SU = \sum_{i=1}^{N_R} \frac{(d_{max} - d_{R_i,T_j} + 1)}{N_{R_k,T_j}}$$

Nash Equilibrium

- If each player has chosen a strategy, and no player can benefit by changing strategies while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding payoffs constitutes a Nash equilibrium – Wikipedia

Prisoner 1 \ Prisoner 2	Cooperate (with other)	Defect (betray other)
Cooperate (with other)	-1, -1	-3, 0
Defect (betray other)	0, -3	-2, -2

Competitive Algorithm

Algorithm 1 Competitive algorithm.

function COMPETITIVE(N_R, N_T)

$R = createRobots(N_R, G_R, N_C)$

$T = createTasks(N_T, G_T)$

$U = calculateUtility(R, T)$

for $r \in R$ **do**

$competitors = robotAwareness(r, R)$

$allocation = searchBestNE(competitors, U)$

$taskAllocation(r) = allocation(0)$

end for

return $taskAllocation$

end function

Competitive Algorithm

- Each agent knows the neighbor agents distances and the set of available tasks
- Agents calculate the equilibrium points for all neighbors

Competitive Algorithm

to perform two tasks. The robots are located in $R_1 : (1,3)$ and $R_2 : (3,1)$, whereas the tasks are located in $T_1 : (2,4)$ and $T_2 : (2,1)$. The maximum distance considered for this scenario is $d_{max} = \|(5,5)\| = 7.0711$. The properties of

	Robot 2 $\rightarrow$ Task 1	Robot 2 $\rightarrow$ Task 2
Robot 1 $\rightarrow$ Task 1	(0, 0)	**(5.84, 7.07)**
Robot 1 $\rightarrow$ Task 2	**(6.65, 4.91)**	(0, 0)

Cooperative Algorithm

- Pre assigned citizens and leader
- Citizens vote for robots to perform each task
- Leaders count vote and determine the task allocation

Voting Methods

- **Borda count** is a ranked voting system where the preferences of voters {1,2,3,...,N} are weighted with decremental coefficients {N, N −1, N −2, ..., 1}
- **Plurality rule** is a binary voting system where each voter assigns 1 to the preferred candidate and 0 to the rest

Voting Methods

- **Approval voting** is also a binary voting system, but each voter assigns 1 to the N preferred candidates and 0 to the rest
- **Cumulative voting** is a rated vote system where each voter has P points that can distribute among the N preferred candidates

Cooperative Algorithm

Algorithm 2 Cooperative algorithm.

function COOPERATIVE(N_R, N_T)
  $R = createRobots(N_R, G_R, N_C)$
  $T = createTasks(N_T, G_T)$
  $U = calculateUtility(R, T)$
  $[L, C] = classifyRobots(R)$
  for $l \in L$ **do**
    $A_l = robotAwareness(r, R)$
    $votes(l) = Vote(A_l)$
    $V = getVoters(l, C)$
    for $v \in V$ **do**
      $A_c = robotAwareness(r, R)$
      $votes(c) = Vote(A_c)$
    end for
    $result = Count(votes, M_C)$
    $allocation(V) = searchBestNE(V, result)$
    $taskAllocation(V) = allocation(V)$
  end for
  return $taskAllocation$
end function

Use of Genetic Algorithms

- When the number of tasks and the number of robots are large
- Search space for task allocation explodes
- Use genetic algorithms to find the Nash equilibrium values

Experiment 1 – Tasks per leader

- Performance of cooperative algorithm depending on the tasks per leader
- 500 Simulations performed
- Tasks are {21, 42, 63, 84, 100}
- Every leader coordinates 10 citizens

Experiment 1 - Results

F_T	Completed Tasks	Social Utility
21	67.82%	827.11
42	63.55%	733.16
63	58.30%	627.55
84	52.90%	537.09
100	48.95%	471.76

Experiment 2 - Best Electoral System

- Performance of cooperative algorithm depending on the electoral system
- 800 simulations were performed
- All 4 methods were evaluated

Experiment 2 - Results

Electoral method	Completed Tasks	Social Utility
Borda count	67.75%	825.60
Plurality rule	67.30%	826.89
Approval voting	68.96%	847.03
Cumulative voting	67.92%	829.69

Experiment 3 – Size of the group

- Performance of algorithms depending on the size of scenario.
- 1000 Simulations conducted
- NR = NT = {20, 40, 60, 80, 100, 120, 140, 160, 180, 200}

Results

$N_R = N_T$	Competitive	Cooperative
20	CT=66.45% SU=175.04	CT=78.80% SU=155.98
40	CT=64.63% SU=346.71	CT=72.70% SU=324.21
60	CT=63.48% SU=514.98	CT=70.27% SU=486.94
80	CT=63.59% SU=690.38	CT=68.64% SU=650.11
100	CT=63.52% SU=864.71	CT=67.74% SU=822.86
120	CT=63.01% SU=1,032.0	CT=67.05% SU=1,002.4
140	CT=63.19% SU=1,208.0	CT=67.19% SU=1,186.6
160	CT=63.04% SU=1,381.1	CT=66.69% SU=1,365.1
180	CT=63.31% SU=1,559.1	CT=66.44% SU=1,541.6
200	CT=62.74% SU=1.823.8	CT=66.17% SU=1,721.7

Experiment 4 – Number of connections

- Performance of cooperative algorithm depending on its parameters
- For 1000 simulations
- Number of connections = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}

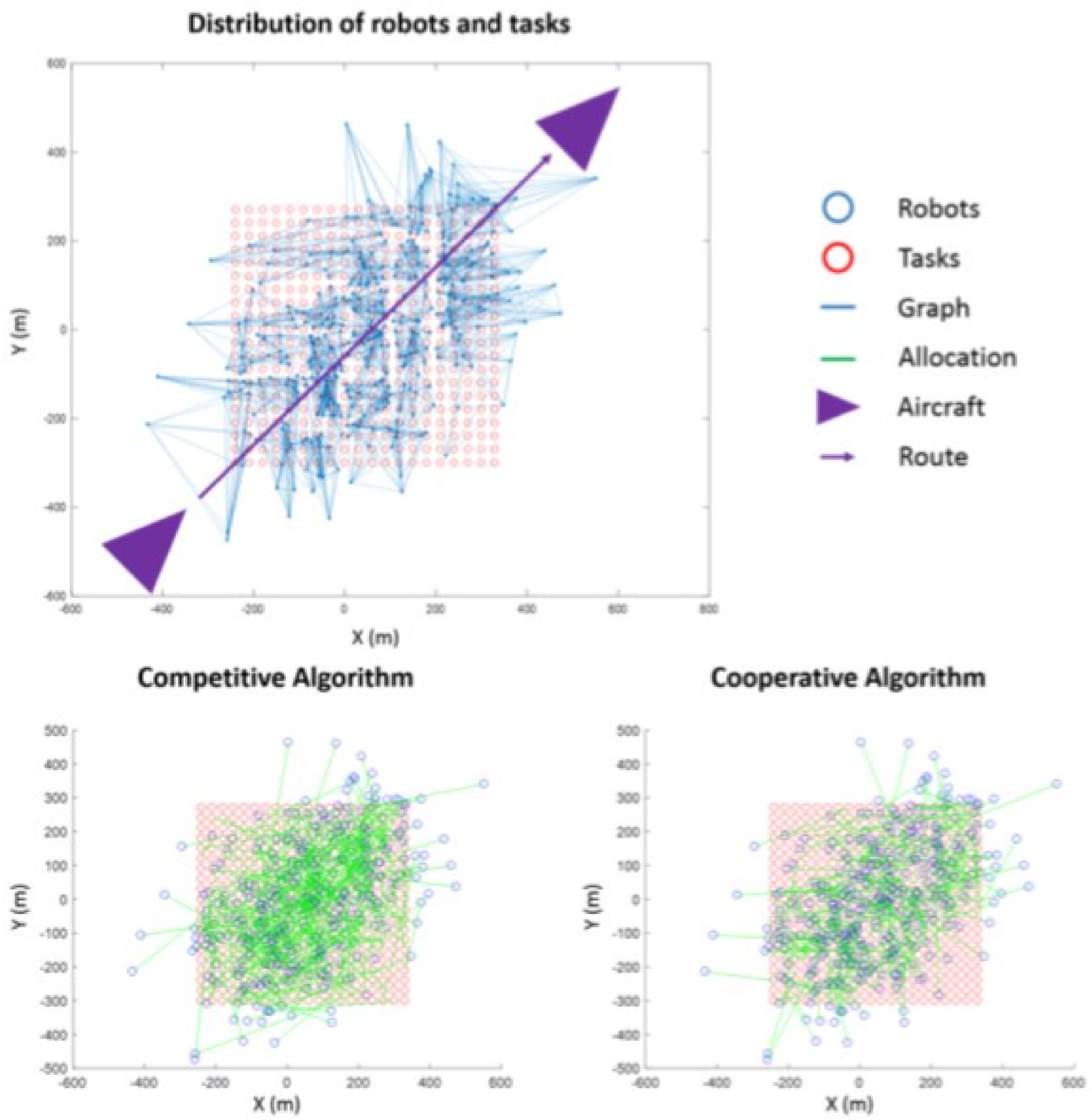
Results

N_C	Competitive	Cooperative
2	CT=61.77% SU=890.35	CT=61.54% SU=818.39
4	CT=62.67% SU=890.37	CT=62.49% SU=812.56
6	CT=62.70% SU=876.52	CT=64.47% SU=820.32
8	CT=63.20% SU=871.79	CT=66.92% SU=831.81
10	CT=62.97% SU=857.65	CT=67.25% SU=819.17
12	CT=63.23% SU=851.23	CT=69.06% SU=823.21
14	CT=62.62% SU=833.17	CT=70.39% SU=826.52
16	CT=63.44% SU=838.30	CT=71.61% SU=829.45
18	CT=62.98% SU=823.87	CT=70.38% SU=810.94
20	CT=62.80% SU=815.77	CT=72.44% SU=822.03

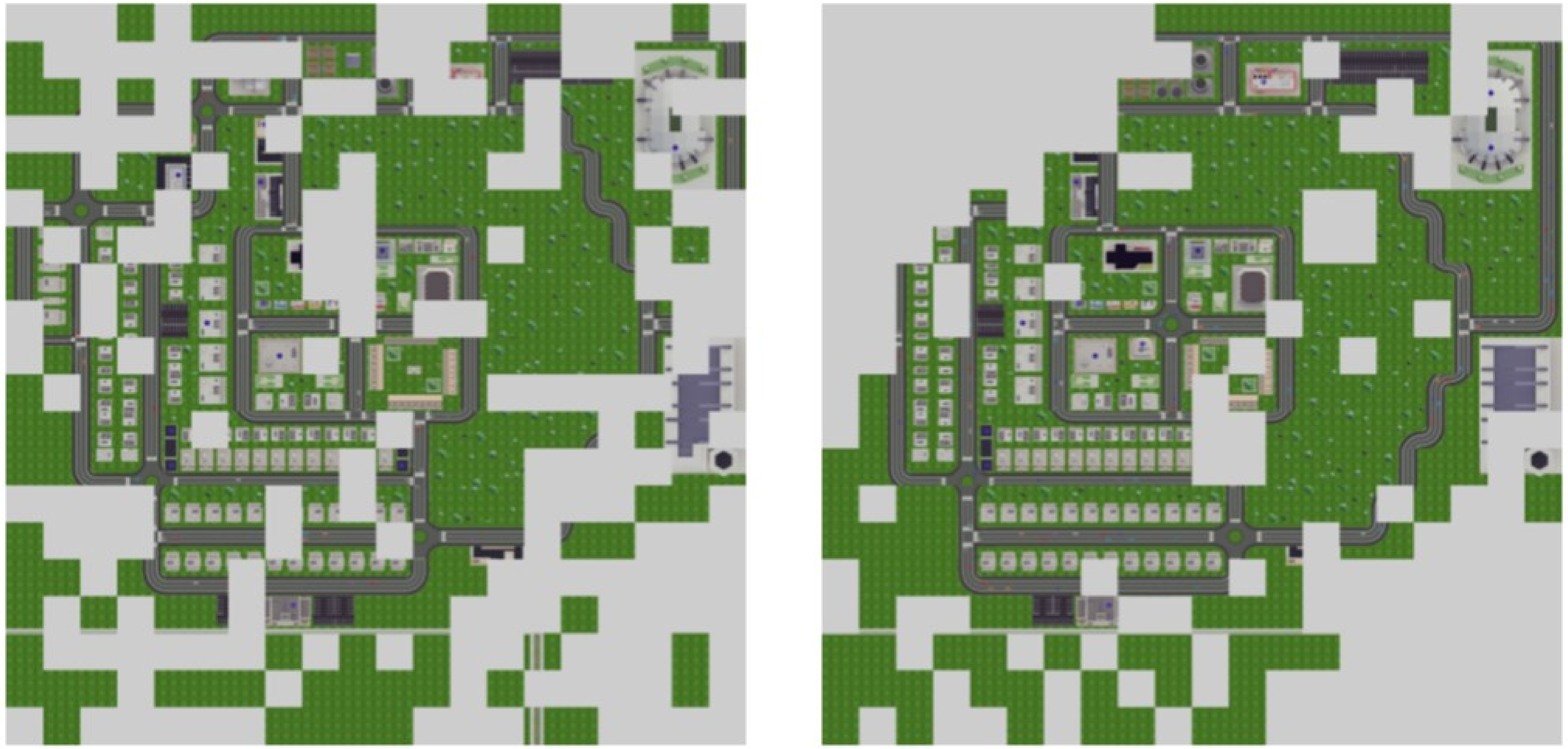
Application for swarms

- 400 Robots and 400 tasks
- 10 neighbor connectivity
- A virtual environment in a game engine
- Mission : Map a part of the virtual environment

Application for swarms



Application for swarms



(a) (b)

Conclusions – In the paper

- Key conclusion : cooperation is better than competition
- In an environment with less communication resources, cooperation will get most number of tasks completed.

Conclusions – Own

- The evaluation metrics are not enough to come at a conclusion.
  - Specially with mixed results
- The equations, symbols are not explained.
- The algorithms contain undefined functions.
- This can be expanded for any multi-robotic platform – not just drones.

Conclusions – Own

- Can a robot calculate the Nash equilibrium as the tasks are inter related ?
- The complexities are not defined !
- Therefore, we can assure “Dear robots, you should cooperate!”
- Surprised as this got accepted to IROS