Application of Dynamic Game Theory to efficient parcels collection

Michele Avella and Elena Leonelli

January 31, 2022

Abstract

In this paper we apply a game-theoretic approach with the aim of finding an efficient parcel collection strategy with multiple drones. We implemented an algorithm that simulates a Multistage Game where each player represents a drone, whose trajectory in the space is defined during the sequential moves. The decision is taken according to a game-theoretic approach, considering the positions of the remaining parcels and of the other players.

Introduction

Automating the collection of parcels in a bi-dimensional space using multiple drones is a very challenging but useful task. What makes it interesting is that it is very flexible and can be easily applied to accomplish different tasks such as delivering objects or analyzing specific points inside an area. One of the key issues for a multiple drones system is how to coordinate the behaviors of individual drone so that the overall global performance can be optimized. In order to tackle this problem we will use game theory considering each drone as a player of a game. One of the main point of the paper will be the choice of the payoff of the drones that will determine their behaviour.

The content of this paper is the following. At first, we present the scenario and the game-theory tools that allowed us to develop the model. Then, we introduce the procedure used for calculating the strategy each drone should follow. Eventually, we show an extensive set of simulations to validate our approach.

1 The scenario

We considered the following situation. The players of this game are a team of drones moving on a bidimensional grid (lattice) delimited at the edges, *i.e.* is not possible to exit the perimeter. Before the start of the game some parcels are arranged on the field, and the goal of the drones is to pick-up all these boxes.

Time is discretized in T steps. At each timestep,

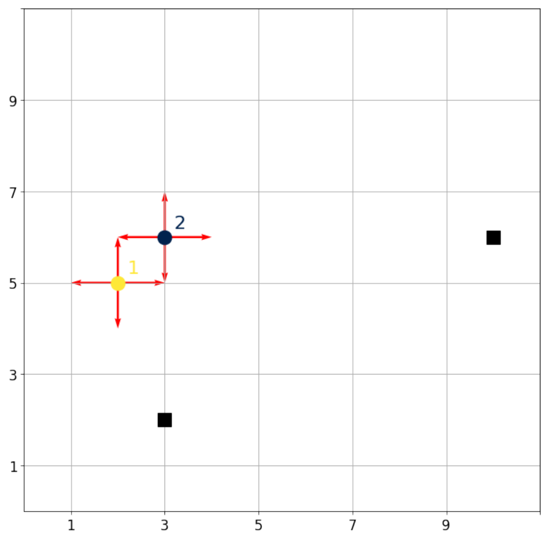


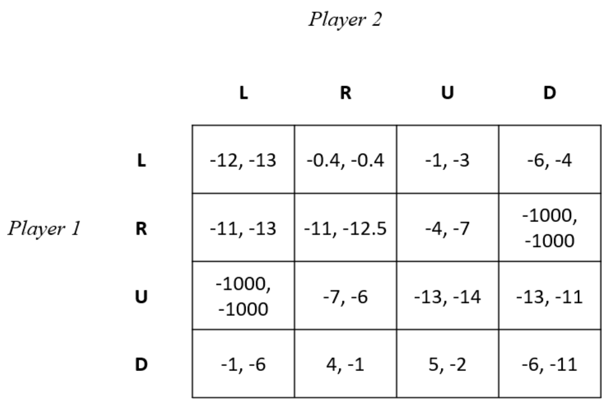
Figure 1: *The game field. The two black rectangles are two boxes, the yellow and blue points are the drones; with the red arrows are highlighted the possible actions each player can take at that moment.*

for the i-th player we define a set of actions:

$$A_i = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}$$

(*move up, right, down, left*), and the payoffs $u_i(\mathbb{X}, \mathbb{B})$ as a function of the position of the boxes and of all the other players.

Each drone knows at all times the locations of the other players and of the parcels. The game is the one with *complete information*: the set of actions A_i and the payoffs are common knowledge among all the players of the game.



		Player 2			
		L	R	U	D
Player 1	L	-12, -13	-0.4, -0.4	-1, -3	-6, -4
	R	-11, -13	-11, -12.5	-4, -7	-1000, -1000
	U	-1000, -1000	-7, -6	-13, -14	-13, -11
	D	-1, -6	4, -1	5, -2	-6, -11

Figure 2: *Normal form of the stage game depicted in Figure 1, with (L,R,U,D)=(left,right,up,down). The NE is (D,R).*

The game is a **multistage game** of complete but imperfect information: at each stage of the game the players move simultaneously, each choosing an action $a_i \in A_i$ that maximizes his payoff (they are *rational players*).

In Figure 1 is depicted a representation of one stage of a 2-players game with 2 parcels on the grid. In Figure 2 is presented the normal form of an example of stage game.

Now we want to find a strategy that solves the multistage game, *i.e.* a subgame-perfect equilibrium of the game. We use the following:

Preposition [4] *Consider a multistage game with T stages, and let s^{t*} be a Nash equilibrium strategy profile for the tth stage-game. There exists a subgame-perfect equilibrium in the multistage game in which the equilibrium path coincides with the path generated by $s^{1*}, s^{2*}, ..., s^{T*}$.*

Thus, our problem boils down to finding a Nash equilibrium strategy for each stage game, and playing it. At the end, we will end up with a subgame-perfect equilibrium in the multistage game.

In order to find the Nash Equilibrium of each stage game we define the best response.

Definition [4] *The strategy $s_i \in S_i$ is player i's best response to his opponent's strategies $s_{-i} \in \S_{-i}$ if:*

$$u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i}) \quad \forall s_i' \in S_i$$

Then, we use the following result:

Preposition[4] *If s^* is a strict dominant strategy equilibrium then $s*_i$ is a best response to s^*_{-i} $\forall i \in N$.*

In this way, to construct the Nash Equilibrium of a stage game is sufficient to find a joint strategy where all the player are playing their best response.

2 Implementation

In the following, we use this notation:

- $\vec{b}_i$ is the position of the box i;
- $\mathbb{B} = \left\{\vec{b}_i\right\}_{i=1}^{N_B}$;
- $\vec{x}_i$ is the position of the drone i;
- $\mathbb{X} = \{\vec{x}_i\}_{i=1}^{N_D}$.

At each step, we define the payoff of each drone $u_i(\mathbb{X}, \mathbb{B})$ as follows:

$$u_i(\mathbb{X}, \mathbb{B}) = \left\langle \frac{100}{1 + |\vec{x}_i - \vec{b}|^2} \right\rangle_{\vec{b} \in \mathbb{B}} - \left\langle \frac{\gamma \cdot 50}{1 + |\vec{x}_i - \vec{y}|^2} \right\rangle_{\vec{y} \in \mathbb{X}} \tag{1}$$

where the first therm is an average over all the boxes positions, and the second is an average over all the drones positions except the one we are considering.

The first term drives the drones to the boxes preferring areas with larger number of boxes; in this way each drone can collect more boxes per number of steps.

The second term can be seen as a repulsive term because it keeps the drones away from each other; in this way, in particular with a larger number of drones, the drones cover a wither area of the lattice avoiding to go to the same boxes. With the second term we can reduce, on average over different realizations of the game, the number of steps needed to take all the boxes. Later in the paper we will show the difference between the number of step needed with and without the second term.

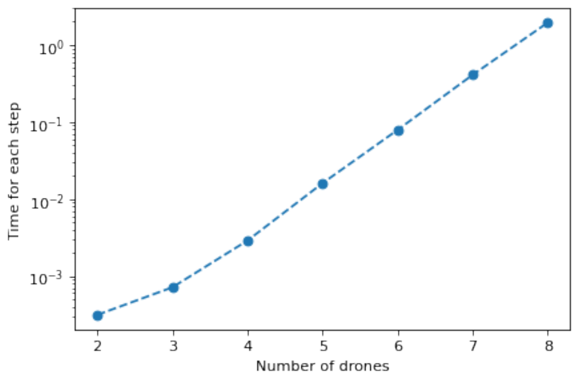


Figure 3: *Time needed for the computation of each step as a function of the number of drones.*

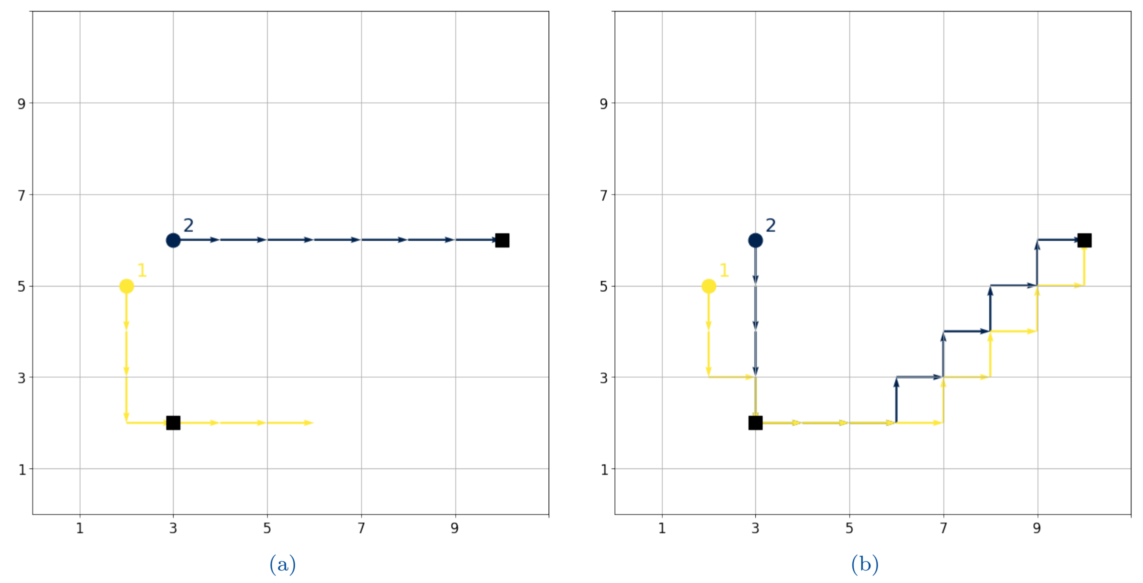


Figure 4: *A 2-players game. The two black rectangles are two boxes, the yellow and blue points are the drones (respectively player 1 and player 2); the arrows represent the move in each time step. On the right, the game with $\gamma = 0$. On the left, the game with $\gamma = 1$.*

In order to avoid crashes between drones or with the borders of the lattice if two drones are in the same position or if a drone is outside the lattice the payoff is $u_i = -1000$.

Nash equilibrium
At each step, we compute the Nash Equilibrium $\vec{s}$ of the stage game and we move all the drones; after that we remove the boxes that have been reached by a drone. We repeat this procedure until there are no more boxes. If there are more than one NE, we choose the *best* as the one that maximizes the mean of all payoffs. The details of the algorithm are reported in the pseudo-code 1.

In 2 is shown the pseudo-code used for finding the best response strategy.

Note that algorithm 1 is computational demanding since for each step game we compute the payoff $N_D \cdot 4^{N_D - 1}$ times; for this reason we could test out algorithm with at most 8 drones. In Figure 3 we computed the time needed to compute each step with $L = 20, N_B = 10$. As we can see this is exponential, according to our estimation.

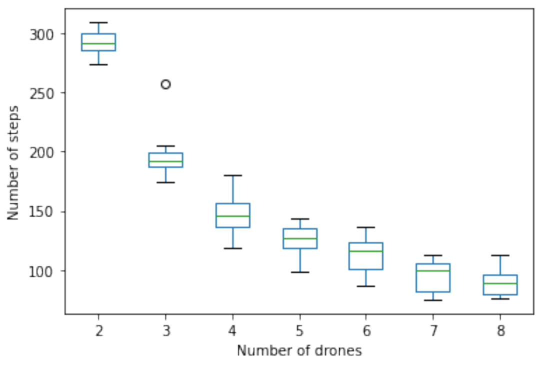
3 Simulation Results

In order to have a better understanding of the behaviour of the simulation, we started with a system of $N_B = 2$ boxes and $N_D = 2$ drones, moving on a square lattice of size $L = 10$ (see Figure 4). For the first simulation we set $\gamma = 1$: thus we expected to see the presence of the repulsive term between the two drones, and his role in each player strategy's decision. This is clearly visible in Figure 4(a): the starting positions of the players (randomly selected) are close enough to influence the choice of the first move. In fact in the first stage-game, for player 1 the payoff associated to move "right" is lower than the one of the strategy "down", regardless of the action of player 2 (see Figure 2).

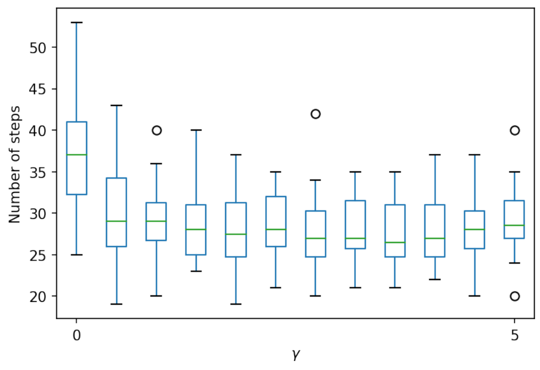
Then, we made another simulation of the same scenario, with same starting positions but without the repulsive term ($\gamma = 0$). The result is consistent with our predictions: as shown in Figure 4(b), both drones start by moving in the direction of the nearest box, regardingless their mutual positions, ad they do not see each other. As expected, the number of steps needed to conclude the game is higher than in the same scenario with $\gamma = 1$.

Scaling in N_D
One of the purposes of our project is to investigate how our scenario scales with the number of drones. We performed many simulations, varying the number of drones in the range $[2, 8]$; the upper bound is due to computational limitations, however the results obtained with 4/5 drones are equally satisfactory.

We expect that, in the same conditions (number of boxes, γ, lattice size), adding more drones has a positive effect on the speed in concluding the game. Trivially, more players means that they can point more boxes at the same time. Moreover, an impor-



(a)



(b)

Figure 5: *In (a), boxplot showing how the number of steps needed to collect all the boxes varies with the number of drones. In (b), boxplot representing the number of steps needed to collect all the boxes as a function of* γ*. The box extends from the Q1 to Q3 quartile values of the sample, with a green line at the median. The whiskers extend from the edges of box, at maximum up to* $1.5 \cdot (Q3 - Q1)$*, ending at the farthest data point within that interval.*

tant role will be the one of the repulsive term: with a non-zero γ, the players will spread over the lattice, covering a larger area as drones increase and favoring the boxes collection.

In Figure 5(a) are shown the results obtained for the number of steps to the end of the game as a function of the number of drones. For this set of simulations we used a square lattice of size $L = 70$ with $N_B = 40$ boxes, $N_D \in [2, 8]$ and $\gamma = 1$. For each value of N_D we perform 10 simulations, varying the random seed that determines the starting positions of the boxes and of the players.

As we can see, with the same number of boxes and lattice size, the number of steps is a decreasing function of the number of drones. In particular, it is an exponential decrease: the difference in number of steps between 2 and 3 drones ($\sim$ 100 steps) is higher than the one existing between 3 and 4 drones ($\sim$ 50 steps).

Parameter γ

In this part we compute the number of steps needed to collect all the boxes as a function of γ. More big is γ and more repulsion there is between the drones. So with $\gamma = 0$ the drones are independent and do not interact between each other, except avoiding to collide. On the contrary with $\gamma > 0$ we have repulsion and so the drones are more uniformly distributed on the lattice. In order to make this comparison we did several simulation on a square lattice with $L = 20$, $N_D = 6$ and $N_B = 30$. In order to compute N_s for each γ we did a mean over 20 different simulations with different random initial positions. We can see the result in Figure 5(b).

As we can see with $\gamma = 0$ we have the worst results. This is because it can happen that two or more drones start to go for the same box and the result is that they start chasing each other wasting efficiency (see Figure 6). With larger γ we have better results because we avoid the situation described above. Of course if we use a too large γ the number of steps starts increasing because the drones are more interested in staying away from each other then collecting boxes; in the worst case the algorithm can not converge.

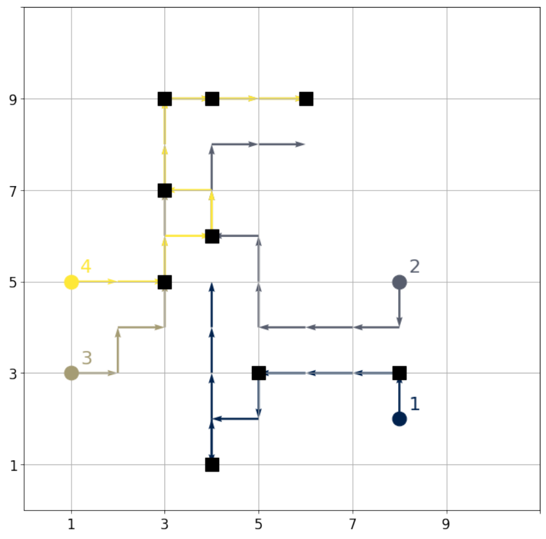


Figure 6: *Simulation with* $N_D = 4$ *drones and* $N_B = 9$ *boxes, lattice size* $L = 10$ *and* $\gamma = 0$*. Here we can see that player 3 and player 4 chase each other, lowering the efficiency of parcel collection.*

4 Conclusions

In this paper we successfully implement a multi drone parcels collection system using Game Theory. We define our game where the players are drones that can move in a two dimensional grid. We chose a proper payoff in order to collect all the parcels in the less possible game steps. We implement an algorithm to move all the drones simultaneously using Game Theory. We test our system in a simulation in order to see the performances and how they changes varying the parameters of the system such as the number of drones and the repulsion between them.

Lot of things can be done to improve or extend this work. First of all, it could be interesting to add a realistic map to this scenario, with obstacles that simulates real-world maps. Also adding different kinds of players, with different payoffs, and also adding a schedule priority on parcels pick-up.

Of course, the most appealing improvement is to implement a faster algorithm to compute Nash Equilibria. In this way it could be possible to extend the scenario, increasing the number of players and boxes, the map size, and in general the complexity.

References

[1] F. Amigoni, N. Gatti, and A. Ippedico. A game-theoretic approach to determining efficient patrolling strategies for mobile robots. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 500–503, 2008.

[2] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '09, page 557–564, USA, 2009. IEEE Computer Society.

[3] S. N. Givigi and H. M. Schwartz. A game theoretic approach to swarm robotics. *Appl. Bionics Biomechanics*, 3(3):131–142, jul 2006.

[4] S. Tadelis. *Game theory: an introduction.* Princeton University Press, 2013.

Appendix: algorithms

Algorithm 1 NE stage game

```
while $N_B > 0$ do
    for each drone $i$ do
        $L_i \leftarrow \{\ \}$                              ▷ Empty set
        for each $s_j \in \{S_{-i}\}$ do
            Compute best response $s_i^*|s_j$ (see 2)
            Add joint strategy $\vec{s}_j$ to $L_i$
        end for
    end for
    $R \leftarrow \bigcap_{i=1}^{N_D} L_i$
    Chose an element $\vec{s} \in R$                ▷ $\vec{s}$ is a NE
    Move all drones using joint strategy $\vec{s}$
    if $\mathbb{X} \cap \mathbb{B} \neq \emptyset$ then
        $\mathbb{B} \leftarrow \mathbb{B} \setminus (\mathbb{X} \cap \mathbb{B})$          ▷ Remove the boxes
    end if
end while
```

Notation

- $\{S_{-i}\}$ is the set of all the joint strategies of all the drones except i.
- $s_i^*|s_j$ is the best response of drone i given the joint strategy s_j of the other drones.
- $\vec{s}_j$ is the joint strategy where i plays $s_i^*|s_j$ and the others play s_j.

Algorithm 2 Best response

```
Require: $s_j$
for $a \in A$ do                          ▷ for each action of $i$
    Move all drones FIP except $i$ with $s_j$
    Move $i$ FIP with $a$
    Compute $u_{i,a}$
end for
return $\arg\max_a(u_{i,a})$
```

Notation: FIR is From Initial Position.