# Moving Drones for Wireless Coverage in a Three-Dimensional Grid Analyzed via Game Theory

Elena Camuffo, Luca Gorghetto, and Leonardo Badia
Department of Information Engineering, University of Padova, Italy
email: {elena.camuffo, luca.gorghetto}@studenti.unipd.it, leonardo.badia@unipd.it

***Abstract*—Drones offer opportunities for networking and control solutions, but also challenges when it comes to their coordination. In this paper, we apply a game theoretic model to multi-agent drone scenarios, to regulate their movement so as to ensure timely positioning but avoiding collisions. Simulations are led through the Nash-Q learning algorithm in order to prove the theoretical analysis and confirm their predicted trajectory dynamics. The results of this work can be exploited as a tool to provide insights for multi-agent control.**

***Index Terms*—Game theory; Cooperative systems; Intelligent robots; Aerial base stations.**

## I. Introduction

Unmanned Aerial Vehicles (UAVs), also called drones, are expected to play a significant role in future technological scenarios, including wireless communications, logistic transportation and delivery, and industrial production [1]. Their interaction becomes complex when multiple UAVs exist in the same area, and coordination strategies are required to harmonize their operations.

The architectures of network comprising drones have been studies at all the layers of the protocol stack [2]. Open issues include energy efficiency, routing (in the form of gateway selection) and mobility of the nodes, which can hinder the overall network stability. A management of drones with intense communication exchanges would nullify the advantages to use drones in the first place, i.e., the adoption of multiple simple systems instead of a unique complex architecture [3].

Therefore, approaches related to the field of game theory have started to gain attention [4]–[6]. The interaction among different agents can be captured as resulting in a Nash equilibrium (NE). This can apply to a multi-agent system where a number of non-adversarial drones (but not necessarily explicitly cooperative) move in the same environment while being limited in both mutual communications and available resources. These kinds of multiagent environments can be modeled as stochastic games, i.e., a theoretical model representing multi-state multi-agent environments having Markov property and a stochastic inter-state transition rule.

However, moving drones are often unaware of the reward functions or state transition probabilities. Hence, a learning problem arises, which can be addressed through model-free reinforcement learning (RL) [7], where agents directly learn about their optimal policy without knowing the reward function or the state transitions. In the last years since a wide range of algorithms have been developed for solving these games [8]–[12]. So far, the standard setting which has been used as a reference for the learning of these algorithms is the one of two-dimensional grid games, where a number of agents are placed on square cells and are allowed to move in 4 directions: *Up, Down, Left* and *Right*. However, this structure lacks in fundamental aspects when dealing with the dynamics in trajectory control, also not considering that aerial drones are also moving in a three-dimensional environment.

In this paper, we examine and simulate an original game-theoretical model, where a new topological scenario is considered. We consider two drones in a three-dimensional grid environment, where cells are considered as voxels. This specific choice can be encountered in real-world situations when multiple drones are located in the same environment.

## II. Related Work

In the past few years, the use of game theory has grown extensively in the robotic research community. Reference [4] proposed game theory for the high-level planning of multiple robot coordination. Relevant applications include a multi-robot search for targets [5] or the shared exploration of structured workspaces like building floors [6]. In parallel, the interest in the implementation of game theoretical techniques in conjunction with RL increased significantly. Single-Agent RL has seen wide application in robotics, such as in robotic arm control [13], and also in multiagents domains such as robotic soccer [1]. However, the latter scenario, as argued in [7], poses many theoretical problems since the environment can no longer be considered as stationary.

The actual extension of RL techniques from single-agent to multi-agent environments includes two main classes of learning algorithms, called *adaptive learning algorithms* and *equilibrium learning algorithms*. The main difference is that in the latter case agents are calculating an equilibrium solution assuming that their opponents are rational, and their convergence is limited to a number of cases where these equilibria are identifiable. The adaptive learning agents, on the contrary, make no assumptions about rationality or learning capabilities of other agents. These learning algorithms are proven to converge in self-play (i.e., when learning against agents that are using the same learning algorithm) to an equilibrium solution in a wide variety of repeated matrix games. Among adaptive algorithms, the ones which have been exploited more frequently are Infinitesimal Gradient Ascent (IGA) [8], Policy Hill-Climbing (PHC) [9] and Adaptive Play Q-learning (APQ) [10]. Regarding equilibrium learning algorithms instead, it is possible to mention popular techniques such as Minimax Q-learning [11], Friend-or-foe Q-learning

[12] and Nash Q-learning [7]. In particular, the latter was the one that was exploited in our simulations. All these techniques were empirically tested by their respective authors on the different test benches. However, although these algorithms were tested on a number of repeated matrix games and on some examples of stochastic games [7], a number of questions is remaining whether these algorithms are well extensible to the general form of stochastic games. Our study employs more advanced underlying models than those already existing in the literature, providing thus a first proof of their effectiveness in a wide variety of possible new settings.

## III. Background

We consider a scenario where multiple moving UAVs are controlled by independent agents. This means the motion control of our drones has the necessary intelligence to perceive the characteristics of the environment and take independent actions. An *agent* can move in the environment and its position at time $t$ is denoted with $x_t$. The temporal sequence of locations, or *path*, is given as $X_T = \{x_0, x_1, x_2, \ldots, x_T\}$, where $T \leq \infty$ denotes the terminal time, at which the game ends. The initial location $x_0$ often serves as a point of reference for the estimation algorithm. We define $u_t$ the odometry that characterized the motion between time $t-1$ and time $t$, obtained from the feedback control. Sequence $U_T = \{u_0, u_1, u_2, \ldots, u_T\}$ characterizes the relative motion of the agent, given by the sequence of its actions. Let $m$ denote the map of the environment, which is supposed to be static, i.e., time-invariant. The agent measurements establish information between features in $m$ and the agent location $x_t$. If we assume, without loss of generality, that the agent takes exactly one measurement at each point in time, the sequence of measurements is given as $Z_T = \{z_1, z_2, z_3, \ldots, z_T\}$.

*Definition 1:* A *localization problem* [14] seeks to obtain the path or current position of the agent $x_{0:T}$ given the agent controls $u_{1:T}$ and observations $z_{1:T}$.

Localization is needed in order to perform motion planning, i.e., the ability for an agent to compute its own collision-free path motion towards certain goal. Motion planning is performed knowing geometry and kinematics, initial and goal positions, and the geometry of the environment assuming static obstacles. These definitions show that our scenario fits the framework of stochastic games, explained below.

Stochastic games model multi-agent systems with discrete-time and non-cooperative nature, meaning that players pursue their individual goals and cannot form an enforceable agreement on their joint actions. In a stochastic game, agents choose actions simultaneously. The state space and action space are assumed to be discrete. Given state $s$, agents independently choose actions $a_1, \cdots, a_n$, and receive rewards $r^i(s, a_1, \cdots, a_n)$, $i = 1 \cdots n$. The state then transits to the next state $s'$ based on fixed transition probabilities, satisfying the constraint:

$$\sum_{s' \in S} p\left(s' \mid s, a^1, \ldots, a^n\right) = 1 \tag{1}$$

*General-sum stochastic games* do not impose constraints on the rewards to the agents. As special cases, *zero-sum stochastic games* are instances where agents' rewards are always negatively related. In a *discounted stochastic game*, the objective of each player is to maximize the discounted sum of rewards, with discount factor $\beta \in [0, 1)$. A strategy $\boldsymbol{\pi}$ is defined as a plan for playing a game. Here $\boldsymbol{\pi} = (\pi_0, \ldots, \pi_t, \ldots)$ is defined over the entire course of the game, where $\pi_t$ is called the decision rule at time $t$. A decision rule is a function $\pi_t : \mathbf{H}_t \to \Delta(A)$, where $\mathbf{H}_t$ is the space of possible histories at time $t$, with each $H_t \in \mathbf{H}_t, H_t = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$, and $\Delta(A)$ is the space of probability distributions over the agent's actions. $\boldsymbol{\pi}$ is called a stationary strategy if $\pi_t = \bar{\pi}$ for all $t$, that is, the decision rule is independent of time. $\boldsymbol{\pi}$ is called a behavioral strategy if its decision rule may depend on the history of the game play, $\pi_t = f_t(H_t)$. If we let $\boldsymbol{\pi}^i$ be the strategy of player $i$, then for any given initial state $s$, player $i$ tries to maximize:

$$v^i\left(s, \boldsymbol{\pi}^1, \ldots, \boldsymbol{\pi}^n\right) = \sum_{t=0}^{\infty} \beta^t E\left(r_t^1 \mid \boldsymbol{\pi}^1, \ldots, \boldsymbol{\pi}^n, s_0 = s\right) \tag{2}$$

In a stochastic game $\Gamma$, a NE is a tuple of $n$ strategies $\left(\boldsymbol{\pi}_*^1, \ldots, \boldsymbol{\pi}_*^n\right)$ such that for all $s \in S$ and $i = 1, \ldots, n$,

$$v^i\left(s, \boldsymbol{\pi}_*^1, \ldots, \boldsymbol{\pi}_*^n\right) \geq v^i\left(s, \boldsymbol{\pi}_*^1, \ldots, \boldsymbol{\pi}_*^{i-1}, \boldsymbol{\pi}^i, \boldsymbol{\pi}_*^{i+1}, \ldots, \boldsymbol{\pi}_*^n\right)$$

for all $\boldsymbol{\pi}^i \in \boldsymbol{\Pi}^i$, where $\boldsymbol{\Pi}^i$ is the set of strategies available to agent $i$. The meaning of a NE is that of a joint strategy where each agent plays a best response to the others. In general, the strategies that constitute a NE can be behavioral strategies or stationary strategies. It is proven in [15] that every $n$-player discounted stochastic game possesses at least one NE in stationary strategies. In this paper, we limit our study to stationary strategies, thus, if a state is visited multiple times, the players' choices would be the same each time. We define the Nash Q-value as the expected sum of discounted rewards when all agents follow a specified NE, that is,

$$\begin{aligned} &Q_*^i\left(s, a^1, \ldots, a^n\right) = r^i\left(s, a^1, \ldots, a^n\right) + \\ &+\beta \textstyle\sum_{s' \in S} p\left(s' \mid s, a^1, \ldots, a^n\right) v^i\left(s', \boldsymbol{\pi}_*^1, \ldots, \boldsymbol{\pi}_*^n\right) \end{aligned} \tag{3}$$

where $\left(\boldsymbol{\pi}_*^1, \ldots, \boldsymbol{\pi}_*^n\right)$ is the joint Nash equilibrium strategy, $r^i\left(s, a^1, \ldots, a^n\right)$ is agent i's one-period reward in state $s$ and under joint action $\left(a^1, \ldots, a^n\right)$, $v^i\left(s', \boldsymbol{\pi}_*^1, \ldots, \boldsymbol{\pi}_*^n\right)$ is agent $i$'s total discounted reward over infinite periods starting from state $s'$ given that agents follow the equilibrium strategies. Also, we distinguish between NE for a stage game (one-period game), and for the stochastic game (many periods). An $n$ player stage game is defined as $\left(M^1, \ldots, M^n\right)$, where for $k = 1, \ldots, n, M^k$ is agent k's payoff function over the space of joint actions, $M^k = \left\{r^k\left(a^1, \ldots, a^n\right) \mid a^1 \in A^1, \ldots, a^n \in A^n\right\}$, and $r^k$ is the reward for agent $k$. If now we let $\sigma^{-k}$ be the product of strategies of all agents other than $k, \sigma^{-k} \equiv \sigma^1 \cdots \sigma^{k-1} \cdot \sigma^{k+1} \cdots \sigma^n$, a joint strategy $\left(\sigma^1, \ldots, \sigma^n\right)$ constitutes a Nash equilibrium for the stage game $\left(M^1, \ldots, M^n\right)$ if, for $k = 1, \ldots, n$:

$$\sigma^k \sigma^{-k} M^k \geq \hat{\sigma}^k \sigma^{-k} M^k \quad \text{for all } \sigma^k \in \hat{\sigma}\left(A^k\right)$$

In the Nash Q-Algorithm, at each time $t$, the $i$-th agent observes the current state and takes its action. After that, it observes its own reward, the actions taken by all other agents

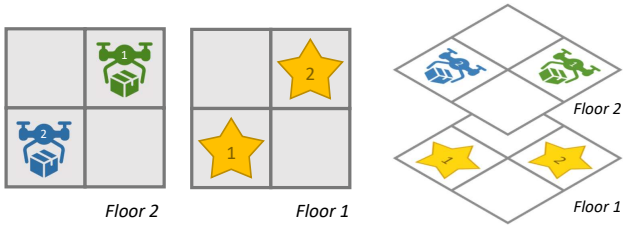


Figure 1. Scenario

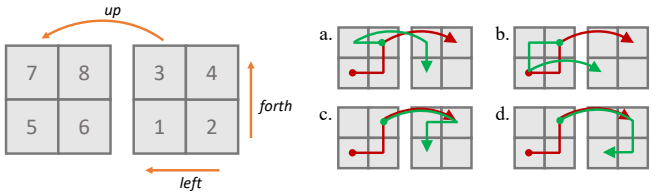


Figure 2. Notation and examples of equilibrium paths

and their rewards, and the new state $s'$. It then calculates a NE $\pi^1(s') \cdots \pi^n(s')$ for the stage game $\left(Q_t^1(s'), \ldots, Q_t^n(s')\right)$, and updates its Q-values according to:

$$\begin{aligned} Q_{t+1}^i\left(s, a^1, \ldots, a^n\right) &= (1-\alpha_t)\, Q_t^i\left(s, a^1, \ldots, a^n\right) \\ &\quad +\alpha_t\left[r_t^i + \beta \text{NASH}\left(Q_t^i(s')\right)\right] \end{aligned}$$

$$\text{where:} \quad \text{NASH}\left(Q_t^i(s')\right) = \left(\pi^1(s') \cdots \pi^n(s')\right) \cdot Q_t^i(s') \quad (4)$$

## IV. Scenario

We consider a game played by two drones in a three-dimensional environment, which traverse a tight area to safely deliver their loads. They need to avoid the other drone staying safely with no collisions, but they need also to overcome the issue in the lowest amount of time possible, therefore following the shortest path to achieve their goals.

We assume a 3D grid, where each voxel represents a position that an agent can occupy. Each drone can move only one voxel at a time, in six possible directions: *Up, Down, Left, Right, Forth, Back*. The two agents are placed in the opposite corners of the upper floor and try to reach their goal on the opposite corner in the lower floor. If they attempt to move into the same cell (excluding a goal cell), they are bounced back to their previous cells. The game ends as either drone reaches its goal. When the scenario is deterministic, the two shortest paths that do not interfere with each other constitute a Nash equilibrium, since each is a best response to the other. The objective of each drone is to reach its goal with the minimum number of steps without colliding. The drones do not know the goal at the beginning of the learning period. Furthermore, the drones choose their actions simultaneously, after observing the previous actions of both, the current state (current position of both) and their rewards.

Fig. 1 shows this game using three different representations. The action space of drone $i$, $i = 1, 2$, is $A^i = \{$*Up, Down, Left, Right, Forth, Back*$\}$; the state space is $S = \{(1,2), (1,3), \ldots, (8,7)\}$, where a state $s = (l^1, l^2)$ represents the two agents' joint location. Drone $i$'s location is represented by a position index, as shown in Fig. 2. State transitions are deterministic and the rewards that each drone can receive are:

- 100 if it reaches the goal position.
- -1 if it collides with the other drone.
- 0 otherwise.

Let the initial state be $s_0 = (8,5)$, as in Fig. 1, and the discount factor $\beta = 0.99$. The value of the game can be computed for both players. Since the game is symmetric, we can restrict the analysis to drone 1 only. The value of the game for drone 1 is defined, see (2), as its accumulated reward when both agents follow their NE strategies,

$$v^1(s_0) = 0 + 0.99 \cdot 0 + 0.99^2 \cdot 0 + 0.99^3 \cdot 100 \ = 97.0$$

Different strategies can yield the same value, and Fig. 2 shows that different strategies can reach similar NEs. Based on the values of each state, it is possible to derive the Nash Q-values for drone 1 in state $s_0$ using (3),

$$\begin{aligned} Q_*^1(s_0, \textit{Left, Forth}) &= -1 + 0.99 \cdot v((8,5)) = 95.1 \\ Q_*^1(s_0, \textit{Down, Forth}) &= 0 + 0.99 \cdot v((6,7)) = 97.0 \end{aligned}$$

The whole set of Nash equilibria can be found, and it can be shown that there are seven of them, all with payoffs $(97.0, 97.0)$.

Assume now that the drones know their goal positions. The drones always know also their own and the other's location and the moves are deterministic; consequently, the game can be modeled as a stage game with complete and perfect information.

At the beginning, the drones are placed on the same floor. This means that in the first stage they are led both to choose the move *Down*. Then (*Down*, *Down*) is the dominant strategy. This will also be the case if the number of floors is higher, since only in the last stage, where the two are in the same floor of their objective, they choose the strategy that brings them to their goal.

The game is a *collaborative game* where the collaborative strategy at stage 1 is to move *Down* until the lowest floor is reached. That strategy is also the stage NE. At the second stage, the game becomes no longer collaborative and no NE in pure strategy can be found. The game becomes a *discoordinaton* game, whose only NE is achieved using mixed strategies, playing each move half of the times.

The second stage leads to two different results: if the discoordination strategy is achieved, the drones go to different cells; in this case, the game ends in the following stage with payoff $100\beta$ for both. Instead, if they attempt to go to the same cell, the stage game is repeated until discoordination is achieved; in the latter case, the payoff is $-1 + \sum_{t=1}^{T-1} \beta^t \cdot (-1) + 100\beta^T$ for both. This means that the best strategy that the two drones can adopt is to reduce the problem to a coin flip, and postpone the inevitable fight to the last round, which is discounted and has therefore a lower impact on the final payoff.

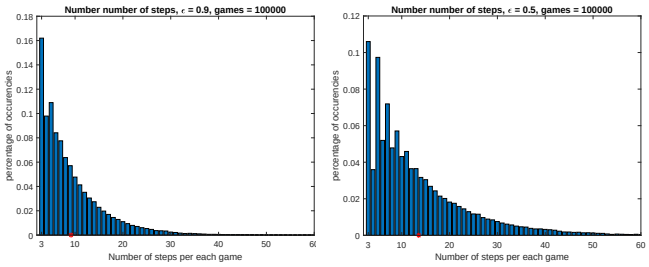


Figure 3. Path length with $\epsilon = 0.9$ (left), $\epsilon = 0.5$ (right).

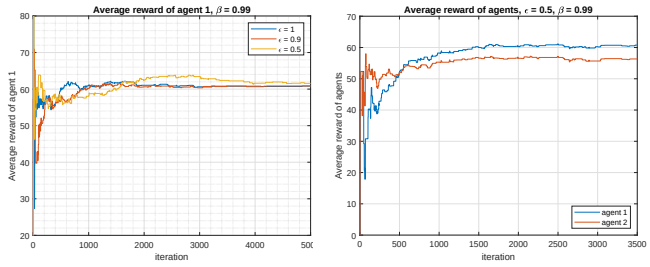


Figure 4. Average value of the Q-function vs number of iterations, for different values of $\epsilon$ (left) or different users (right).

## V. Results

We developed a version of Nash-Q learning algorithm originally proposed by [16]. The function takes advantage of the Lemke Howson algorithm [17] to find the NEs.

The algorithm uses a multiplayer $\epsilon$-greedy exploration strategy, then the strategies to adopt can be *explore*, *exploit*, or *explore and exploit*. In the implemented version, the value of parameter $\epsilon$, controls the probability of choosing the *exploit* strategy. The average length of the path to reach the goal is measured with different $\epsilon$ and the results are given in Fig. 3. The more the *exploit* strategy is used, the shorter the average path length, since moving the drone at random in the space, keeps it into the play for a long time and implies a waste. In addition, the number of steps per path is distributed according to a geometric distribution, which generally models the waiting time of an event, in this case the reaching of the goal. The average length is quite high on average because the drones do not know where their goals are, but choosing always the best move of the situation the average path length shortens. This result is also influenced by the discount factor.

The results for the Q-function are reported in Fig. 4. The behavior is always convergent, but the speed of convergence depends on the method chosen. If players always follow an *exploit* strategy, convergence is faster with respect to the case in which they play $exploit$ half of the times and $explore$ the other times. The average values, computed for both players, are similar to the grid-game-1 of [7]. The convergence to a NE is not always guaranteed by the Lemke Howson algorithm, used by the Nash-Q learning during the game, and repeated tries are required to lead all the configurations to convergence. Nevertheless, in the final Q-matrix obtained, the NEs are computed neglecting the moves that cannot be performed in that state. The result is that, when the convergence is reached, it gives always one of the seven NEs.

## VI. Conclusions

We investigated a game of autonomous drones moving in a three dimensional scenario, where we developed an implementation of the Nash Q-algorithm. The results obtained are promising as the NEs derived in the theoretical analysis coincide with the ones given in the simulation. This verifies the prediction capability of NE, as well as the functioning of the Nash-Q technique in this extended scenario. Moreover, the algorithm provides a means to evaluate the average path length metric for different values of $\epsilon$ and prove the convergence of the Q function.

Further investigations can consider different kinds of movement, still under a stochastic game approach. Also, the systems parameters may be changed for what concerns the transition probabilities or the discount factor. Finally, our approach can be tested with different types of RL algorithms to compare both the learning speed and the final performance.

## Acknowledgment

Part of this work was supported by MIUR (Italian Minister for Education) under the initiative "Departments of Excellence" (Law 232/2016).

## References

[1] P. Stone and R. S. Sutton, "Scaling reinforcement learning toward RoboCup soccer," *Proc. ICML*, pp. 537-544, 2001.
[2] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones, *IEEE Veh. Tech. Mag.*,vol. 12, no. 3, pp. 73-82, 2017.
[3] G. Quer, F. Librino, L. Canzian, L. Badia, and M. Zorzi, "Inter-network cooperation exploiting game theory and Bayesian networks," *IEEE Trans. Commun.*, vol. 61, no. 10, pp.4310-4321, 2013.
[4] S. M. LaValle and S. Hutchinson, "Game theory as a unifying structure for a variety of robot tasks," *Proc. IEEE Int. Symp. Intell. Control*, pp. 429-434, 1993.
[5] Y. Meng, "Multi-robot searching using game-theory based approach," *Int. J. Adv. Robot Syst.*, vol. 5, no. 4, pp. 341-350, 2008.
[6] K. Skrzypczyk, "Game theory based task planning in multi robot systems," *Int. J. Simulat.*, vol. 6, no. 6, pp. 50-60, 2005.
[7] J. Hu, M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4 pp. 1039-1069, Nov. 2003.
[8] S. Singh, M. Kearns, and Y. Mansour, "Nash convergence of gradient dynamics in general-sum games," *Proc. UAI 94*, pp. 541-548.
[9] M. Bowling, and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215-250, 2002.
[10] A. Burkov and B. Chaib-draa, "Adaptive play Q-learning with initial heuristic approximation," *Proc. IEEE ICRA*, pp. 1749-1754, 2007.
[11] M. Littman, "Markov games as a framework for multi-agent reinforcement learning," *Proc. ICML*, pp. 157-63, 1994.
[12] —, "'Friend-or-foe Q-learning in general-sum games," *Proc. ICML*, vol. 1, pp. 322-328, 2001.
[13] A. Franceschetti, E. Tosello, N. Castaman, and S. Ghidoni, "Robotic arm control and task training through deep reinforcement learning," *arXiv preprint*:2005.02632, 2020.
[14] C. Stachniss, J. J. Leonard, and S. Thrun, *Simultaneous Localization and Mapping*, chapter 46, *Handbook of Robotics*, Springer, 2016.
[15] A. M. Fink, "Equilibrium in a stochastic $n$-person game," *J. Sci. Hiroshima Univ. Ser. A-I Math.*, vol. 28, no. 1, pp. 89-93. 1964.
[16] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm," *Proc. ICML*, pp. 242-250, 1998.
[17] C. E. Lemke and J. T. Howson, Jr., "Equilibrium points of bimatrix games," *J. Soc. Ind. App. Math.*, vol. 12, no. 2, pp. 413-423, 1964.