IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 3, September 2024, pp. 2978∼2986
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i3.pp2978-2986

Multi quadrotors coverage optimization using reinforcement learning with negotiation

Glenn Bonaventura Wijaya, Tua Agustinus Tamba
Department of Electrical Engineering, Parahyangan Catholic University, Bandung, Indonesia

Article Info

Article history:

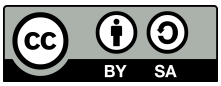
Received Aug 19, 2023
Revised Dec 20, 2023
Accepted Feb 10, 2024

Keywords:

Area coverage optimization
Game theory
Markov decision process
Multi-agent systems
Reinforcement learning

ABSTRACT

This paper proposes an optimization scheme to maximize the area coverage of multiple quadrotor unmanned aerial vehicles that are deployed to monitor an operational area/space. Each quadrotor initially performs a single agent reinforcement learning to determine target points with optimal coverage area. Whenever each quadrotor encounters the others within a predetermined negotiation region that is defined by an inter-agent distance threshold, it will activate a multi-agent reinforcement learning with action negotiation algorithm and coordinate its movement policies to maximize the total coverage area and avoids inter-agent coverage overlaps. Results of simulation evaluations are shown to illustrate the performance of the proposed learning-based coverage optimization method.



Corresponding Author:

Tua Agustinus Tamba
Department of Electrical Engineering, Parahyangan Catholic University
St. Ciumbuleuit no. 94, Bandung 40141, West Java, Indonesia
Email: ttamba@unpar.ac.id

1. INTRODUCTION

The optimization of coverage area are usually required in sensor networks deployment for environmental monitoring and surveillance. Each sensor in such networks usually has constraints in terms of power, sensing distance, communication range, and thus should be deployed in groups statically at fixed locations or dynamically following certain reference trajectories [1], [2]. It is thus important to ensure that the collective sensing of such statically/dynamically placed sensors can cover the desired operational region [3]–[5].

This paper proposes a multi-agent reinforcement learning (MRL) method to address the area coverage problem which often occurs in wireless sensor networks (WSN) applications such as smart farming or transportation networks implementations [6]–[11]. The paper considers the case when the WSN is a group of quadrotors that is deployed to monitor a particular farming land [12]. To formulate a MRL-based optimization scheme, the quadrotors are treated as agents that operate in the farming land. The quadrotors' objective are to maximize the total area coverage that is defined as the field-of-view (FoV) of a camera attached to it while simultaneously minimizes overlaps between different quadrotors' coverage [13], [14].

The proposed MRL is equipped with a negotiation scheme which is triggered whenever the involved quadrotors enter a negotiation region that is defined by a predetermined inter-agent distance threshold. Thus, each quadrotor initially executes a single agent reinforcement learning (SRL) movement algorithm to search for possible target points with optimal area coverage. When a quadrotor encounters the other within the predetermined negotiating region, the involved quadrotors switch their movement algorithms to MRL and coordinate their decisions to maximize the overall area coverage with minimum/no overlaps [15], [16]. Simulation results are shown to illustrate the performance of the proposed MRL-based coverage optimization method.

2. COVERAGE OPTIMIZATION APPROACH

2.1. Area coverage of quadrotor operation

Assume a group of n quadrotors which monitors a desired planar region. Each quadrotor has a localization system for dynamic positioning in a global frame and a control module for movement navigation. Each quadrotor also has a camera pointing downward to monitor the target area. As shown in Figure 1, the camera screens a rectangular-shaped area below the ith quadrotor ($i = 1, \ldots, n$) according to the FoV of its camera. This implies that the area size of the covered FoV will vary with respect to the quadrotor's altitude.

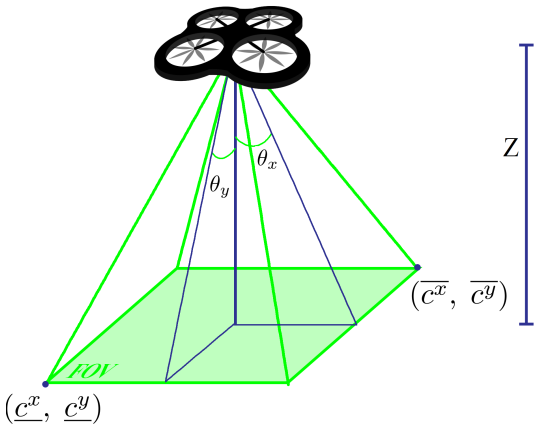


Figure 1. Illustration of single quadrotor's FoV coverage

2.1.1. Coverage area of single quadrotor

Let $O - X - Y - Z$ be a global frame as shown in Figure 1 which defines the reference frame of the operational space of n quadrotors which takes the form of a box. Let ℓ_X, ℓ_Y, and ℓ_Z be the length, width, and height of the box in the X, Y, and Z axis, respectively. Consider the partition of ℓ_X, ℓ_Y, and ℓ_Z into $N_X = \frac{\ell_X}{\triangle_X}, N_Y = \frac{\ell_Y}{\triangle_Y}$ and $N_Z = \frac{\ell_Z}{\triangle_Z}$ number of equally spaced length with size $\triangle_X, \triangle_Y$, and $\triangle_Z$ in the X, Y and Z axis direction, respectively. The operational space may then be viewed as a box with $N_X \times N_Y \times N_Z$ partitions of smaller boxes with discrete locations of the form $L = (X_{k_X}, Y_{k_Y}, Z_{k_Z})^T$ where $k_X = 1, \ldots, N_X$, $k_Y = 1, \ldots, N_Y$, and $k_Z = 1, \ldots, N_Z$. An object's position in the operational space is defined as the discrete location of the box partition where the object resides.

Let $p_i = (x_i, y_i, z_i)^T$ be the discrete position of the ith quadrotor's center-of-mass in the global frame. Let θ_x and θ_y be angles in the X and Y axes, respectively, of the global frame which form the camera's FoV of each quadrotor. As shown for $z_i = 0$ in Figure 1, let $(\underline{c_i^x}, \underline{c_i^y})$ and $(\overline{c_i^x}, \overline{c_i^y})$ respectively, be the minimum and maximum planar $X - Y$ coordinate of the corner points of the ith quadrotor's FoV. Using the geometry in Figure 1, it can be shown that (1) holds:

$$\begin{aligned} \underline{c_i^x} &= x_i - (z_i \tan(\theta_x)), \quad \overline{c_i^x} = x_i + (z_i \tan(\theta_x)) \\ \underline{c_i^y} &= y_i - (z_i \tan(\theta_y)), \quad \overline{c_i^y} = y_i + (z_i \tan(\theta_y)) \end{aligned} \tag{1}$$

The theoretical coverage area λ_i of the ith quadrotor may simply be defined as: $\lambda_i = \left|\overline{c_i^x} - \underline{c_i^x}\right| \left|\overline{c_i^y} - \underline{c_i^y}\right|$.

It should be noted, however, that the coverage area of each quadrotor must be defined over the considered box-shaped operational space. In this regard, whenever the boundary points (1) of the coverage are outside the operational space, then the considered coverage area should be defined only up to the limit of the operational space. As such, the practical coverage area of each quadrotor can be rewritten as in (2):

$$\lambda_i = \left|\overline{c_i^{*,x}} - \underline{c_i^{*,x}}\right| \left|\overline{c_i^{*,y}} - \underline{c_i^{*,y}}\right| \tag{2}$$

where $\underline{c_i^{*,x}} = \max\left(\underline{c_i^x}, X_{\min}\right)$, $\overline{c_i^{*,x}} = \min\left(\overline{c_i^x}, X_{\max}\right)$, $\underline{c_i^{*,y}} = \max\left(\underline{c_i^y}, Y_{\min}\right)$, and $\overline{c_i^{*,y}} = \min\left(\overline{c_i^y}, Y_{\max}\right)$, with $(X_{\min}, Y_{\min})$ and $(X_{\max}, Y_{\max})$ denote, respectively, the minimum and maximum planar area coordinates. Based on (2), the total coverage area Λ of n operating quadrotors becomes (3):

$$\Lambda = \cup_{i=1}^{m} \lambda_i = \cup_{i=1}^{m} \left(\left| \overline{c_i^{*,x}} - \underline{c_i^{*,x}} \right| \left| \overline{c_i^{*,y}} - \underline{c_i^{*,y}} \right| \right) \tag{3}$$

Figure 2 illustrates the coverage area when the operational space is $E = (5, 5, 2)^T$. When the quadrotor is at cell $p = (1, 2, 2)^T$, the actual coverage area is computed as $\lambda = 6$ cells which will otherwise be incorrectly computed as $\lambda = 9$ cells by a direct use of (1). Meanwhile, when the quadrotor is at cell $p = (3, 4, 2)^T$, $\lambda = 6$ cells is obtained which is the same as that computed using (1).

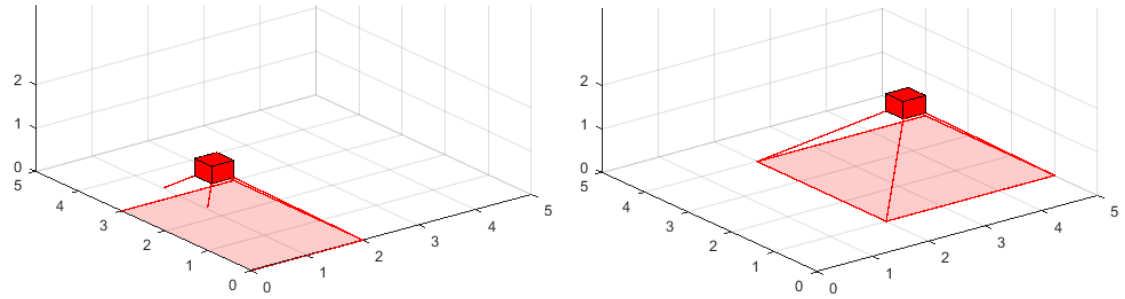


Figure 2. Quadrotor coverage when at $p = (1, 2, 2)^T$ with $\lambda = 6$ (left) and at $p = (3, 4, 2)^T$ with $\lambda = 9$ (right)

2.1.2. Coverage area of multiple quadrotors

In practice, the area coverage (3) should excludes possible coverage overlaps between neighboring quadrotors. This can be achieved using three main steps, namely: i) detection of possible coverage overlaps between several agents, ii) computation of overlapping coverage areas (when exists), and iii) determination of actual coverage area which excludes the overlaps. Each of these steps are elaborated as follows.

The detection of coverage overlaps between two agents can be done by evaluating the set intersection of their coverage areas which can be defined by their boundary points as in (1). Thus, the coverage overlaps between quadrotor i and its neighbor $-i$ is concluded to exist if the set relationship in (4) is false/violated.

$$\left\{ \left[\underline{c_i^x}, \overline{c_i^x} \right] \cap \left[\underline{c_{-i}^x}, \overline{c_{-i}^x} \right] \right\} \wedge \left\{ \left[\underline{c_i^y}, \overline{c_i^y} \right] \cap \left[\underline{c_{-i}^y}, \overline{c_{-i}^y} \right] \right\} = \emptyset \tag{4}$$

When (4) indicates that overlaps exist, the size of the overlapping area is illustrated in Figure 3 and can be computed based on the information of the corner points of each quadrotor's coverage area. As depicted in Figure 3(a), two temporary corner points $(\nu_{\text{min}}^x, \nu_{\text{min}}^y)$ and $(\nu_{\text{max}}^x, \nu_{\text{max}}^y)$ which define the area of coverage overlap between quadrotor i and its neighboring quadrotor $-i$ can be defined as (5):

$$\nu_{\text{min}}^x = \max\left(\underline{c_i^{*,x}}, \underline{c_{-i}^{*,x}}\right),\ \nu_{\text{max}}^x = \min\left(\overline{c_i^{*,x}}, \overline{c_{-i}^{*,x}}\right),\ \nu_{\text{min}}^y = \max\left(\underline{c_i^{*,y}}, \underline{c_{-i}^{*,y}}\right),\ \nu_{\text{max}}^y = \min\left(\overline{c_i^{*,y}}, \overline{c^{*,y}_{-i}}\right) \tag{5}$$

By (5), the coverage overlaps $\rho_{i,-i}$ between quadrotor i and its neighbour quadrotor $-i$ can be computed as in (6):

$$\rho_{i,-i} = \left| \nu_{\text{max}}^x - \nu_{\text{min}}^x \right| \left| \nu_{\text{max}}^y - \nu_{\text{min}}^y \right| \tag{6}$$

Once the corner points which determine the coverage overlaps are obtained, the actual coverage of neighboring quadrotors excluding the overlapping area can be computed as in (7):

$$\Lambda = \cup_{i=1}^{m} \lambda_i \setminus \cup_{i=1}^{m} \rho_{i,-i} \tag{7}$$

The coverage overlap computation is illustrated in Figures 3(b)-(c). Figure 3(b) shows two quadrotors at positions $p_1 = (2, 2, 2)^T$ and $p_2 = (4, 3, 2)^T$ for which an overlap of size 2 unit cells are detected and the actual coverage equals 16 cells. Figure 3(c) instead considers quadrotors at positions $p_1 = (2, 2, 2)^T$ and $p_2 = (5, 2, 2)^T$ for which overlaps are not detected and so the actual total coverage area equals 18 unit cells.

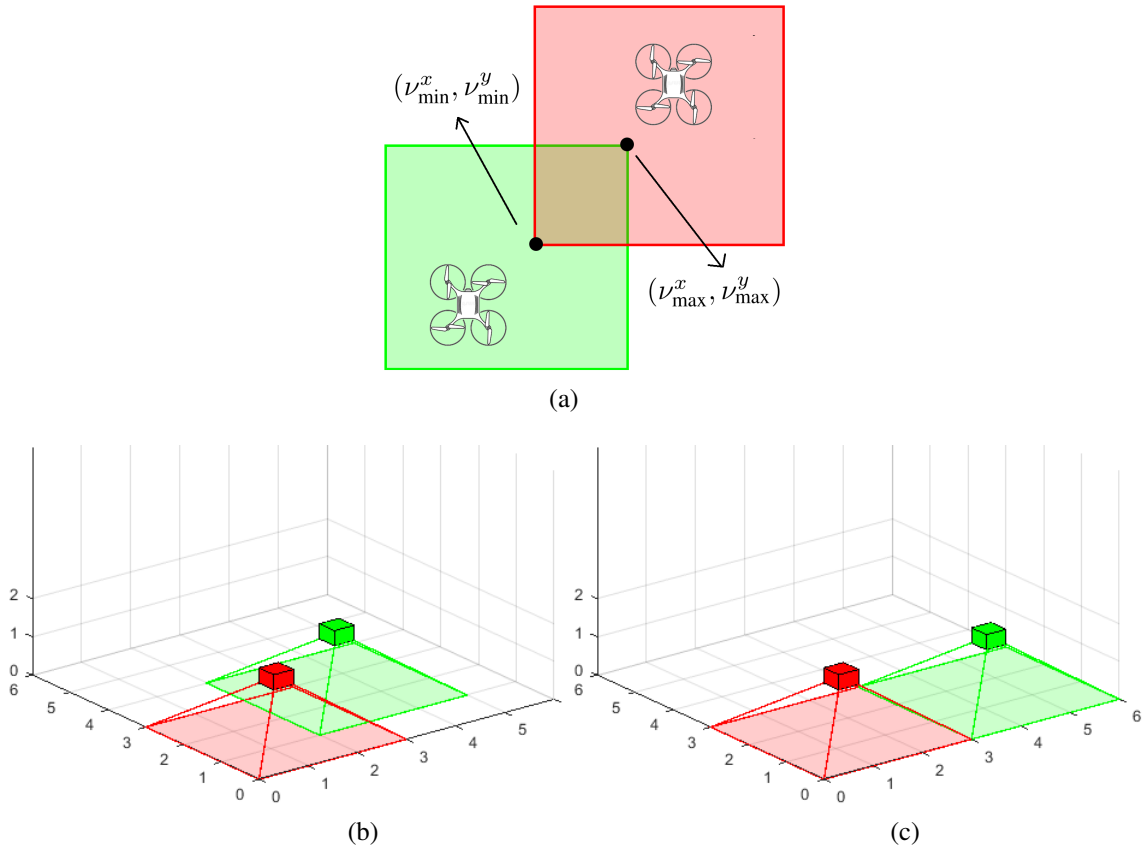


Figure 3. MRL coverage illustration: (a) corner points computation, (b) $\lambda = 2$ when $p_1 = (2, 2, 2)^T$, $p_2 = (4, 3, 2)^T$, (c) $\lambda = 19$ when $p_1 = (2, 2, 2)^T$, $p_2 = (5, 2, 2)^T$

2.2. Learning procedures

2.2.1. Single agent reinforcement learning for single quadrotor's coverage optimization

Markov decision process (MDP) can models the coverage area optimization of a quadrotor [17]. Formally, an MDP is defined as a tuple $\mathcal{D} = \langle P, p^1, U, W, F \rangle$ which consists of a state space $P = \{p^1, p^2, \cdots, p^{n_p}\}$ with initial state p^1, an action space $\mathcal{U} = \{u^1, u^2, \cdots, u^{m_u}\}$, a reward function $W : P \times U \rightarrow W$, and a transition probability function $F : P \times U \times P \rightarrow [0, 1]$. In this regard, a quadrotor is viewed as an agent with state that is defined by its location such that its operational area defines its environment [10], [18]–[20]. The agent's state evolves in the environment over a finite discrete time steps $t = 1, 2, \ldots, T$ from an initial state $p_{t=1} = p^1$. In particular, it observes its state at every time step $t = \{1, \cdots, T\}$, $p_t = p \in P$ and decides an action $u_t = u \in U$ according to a policy function $\omega : P \times U \rightarrow [0, 1]$. For each chosen action u_t at time t, the agent gets a reward $w_t \in W(p, u)$ from the set of expected rewards $W(p, u)$ that is defined as (8):

$$W(p, u) \doteq \mathbb{E}\left\{w_{t+1} | (p_t, p_{t-1}, \cdots, p_1), (u_t, u_{t-1}, \cdots, u_1)\right\} = \mathbb{E}\{w_{t+1} | p_t = p, u_t = u\} \quad (8)$$

Once an action u_t is chosen, the agent's state is transitioned from $p_t = p$ at time t to $p_{t+1} = p'$ at time $t + 1$ where $(p, p') \in P$ with a transition probability function $F(p, u, p')$ of the form:

$$F(p, u, p') \doteq \mathbb{P}\{p_{t+1} = p' | (p_t, p_{t-1}, \cdots, p_1), (u, u_{t-1}, \cdots, u_1)\} = \mathbb{P}\{p_{t+1} = p' | p_t = p, u_t = u\} \quad (9)$$

After the agent transitions to state $p_{t+1} = p'$ at time $(t + 1)$, the evolution sequence is repeated. The MDP analysis seeks for an optimal policy ω^* which maximizes reward aggregate $\Omega^*(p)$ as in (10):

$$\Omega^*(p) := \Omega_{\omega^*}(p) = \max_{\omega} \mathbb{E}\left\{\Sigma_{k=0}^{K} \gamma^k w_{t+k} \,|\, p_t = p\right\} \quad (10)$$

where $\Omega^*(p)$ is the maximum reward of choosing ω^*, $\mathbb{E}_\omega$ is the expectation operator for a policy ω, w_{t+k} is the k steps (of length $K > 0$) reward in the future, and $\gamma^k \in [0, 1]$ is a discount factor which measures

future rewards' potential [8], [10], [19], [20]. Since computing (10) is challenging, approximation methods are often used. Reinforcement learning (RL) is one of such methods whereby an optimal policy is sought using an iterative/learning process which approximates the immediate optimal Q-value $V^*(p,u)$ of the form [8], [21].

$$V^*(p,u) = w_t(p,u) + \gamma^k \Sigma_{p'} \max_{u'} V^*(p',u')\, W(p,u,p') \tag{11}$$

$w_t(p,u)$ in (11) is the immediate reward of the agent at time t under action u_t, $(p',u') = (p_{t+k}, u_{t+k})$ is the joint state-action after k steps, and $W(p,u,p')$ is the transition probability from p to p' under action $u_t = u$. The optimal $V^*(p,u)$ in (11) is iteratively computed using Q-learning (12) with learning rate $\alpha \in [0,1]$ [8], [10], [20].

$$V(p,u) \leftarrow V(p,u)(1-\alpha) + \alpha\{w(p,u) + \gamma^k \max_{u'} V^*(p',u')\} \tag{12}$$

If iteration (12) examines each and every joint state-action pairs often enough with appropriate learning rates, the resulting estimate of the Q-value is known to converge to the optimal one [8], [10], [20].

2.2.2. Multi-agent reinforcement learning for multi quadrotors' coverage optimization

The Markov Game (MG) model generalizes the SRL to the multi-agent case [11], [20], [22], [23].

Definition 1 *An MG of $n \geq 2$ agents is a tuple $\mathcal{G} = \langle n, \{P_i\}_{i=1}^n, \{p_i^1\}_{i=1}^n, \{U_i\}_{i=1}^n, \{W_i\}_{i=1}^n, \{F_i\}_{i=1}^n \rangle$ where $\{P_i\}_{i=1}^n$ are the states of all agents with initials $\{p_i^1\}_{i=1}^n$, $\{U_i\}_{i=1}^n$ are the actions of all agents, $W_i : P_i \times U_i \rightarrow W$ defines agent i's reward, and $F_i : P_i \times U_i \times P_i \rightarrow [0,1]$ is agent i's transition probability function.*

In an MG $\mathcal{G}$, the action choice and transition function of each agent depend on all agents' state-action pairs. In particular, if $\omega_i : P_i \times U_i \rightarrow [0,1]$ for $i = 1, \cdots, n$ is the ith agent's policy, then $\omega_{\mathcal{G}} = (\omega_1, \cdots, \omega_n)$ defines the joint policy of $\mathcal{G}$. The Q-learning of the ith agent in $\mathcal{G}$ is can be formulated as in (13) [10], [24]:

$$V_i^{\omega_{\mathcal{G}}}(\hat{s},\hat{a}) = \mathbb{E}_{\omega_{\mathcal{G}}}\left\{\Sigma_{k=0}^K \gamma^k w_{t+k}^i | \hat{p}_t = \hat{p}, \hat{u}_t = \hat{u}\right\} \tag{13}$$

In (13), $\hat{p}_t$ and $\hat{u}$, respectively, are the spaces of joint states and actions of agent i at time t, and w_{t+k}^i is the ith agent's future reward after k steps. Note that the one step computational complexity of each agent's Q-value in (13) quickly increases as it must be evaluated over an expanded space of joint states and actions of all agents. In this paper, the computation of the ith agent's joint policy (13) is done using the iteration in (14).

$$V_i(\hat{p},\hat{u}) \leftarrow V_i(1-\alpha)(\hat{p},\hat{u}) + \alpha\{w_i(\hat{p},\hat{u}) + \gamma\Phi_i(\hat{p'})\} \tag{14}$$

In (14), $\Phi_i(\hat{p'})$ is the ith agent's expected equilibrium at joint state $\hat{p'} = \hat{p}_{t+k}$ after k steps. Such a joint state can be found using mixed strategy equilibrium computation method in multiplayer game.

3. METHOD

3.1. Single agent reinforcement learning design

In the proposed SRL-based coverage area optimization, the environment of the quadrotor is defined as a three dimensional gridded box such that the partitioned cube-shaped cells of the box define the set of states of the agent. Specifically, each agent can change its state by moving to the right ($x+$), to the left ($x-$), forward ($y+$), backward ($y-$), up ($z+$), and down ($z-$). The set of actions U is thus defined to be $\{U\} = \{x+, x-, y+, y-, z+, z-, \text{idle}\}$ in which an idle state refers to the case where the agent decides to stay at its current state. The set $\mathcal{W} = \{W_i\}_{i=1}^6$ of agent's rewards is defined as follows: i) $W = +100$, if the coverage area equals the *threshold* and no change on movement direction takes place; ii) $W = +99$, if the coverage area equals the *threshold* but there are changes on movement direction; iii) $W = 0$, if an agent is at state P_k and choose certain action; iv) $W = -1$, if an agent moves with no direction change; v) $W = -2$, if an agent moves with direction change; and (vi) $W = -10$, if an agent leaves/moves outside environment.

The transition probability function is defined based on either/combination of the greedy and exploration algorithms developed in [25]. When the greedy algorithm is chosen, the agent selects the best possible action at each step of the learning iteration. If the explore algorithm is selected instead, the agent decides its action randomly and not based on the best possible one. The explore algorithm will thus result in a choice of equiprobable actions. The agent's choice to either use the greedy or explore algorithms is determined by the learning parameter ϵ whereby $\epsilon_G = 0.98$ is set for a greedy agent while $\epsilon_{NG} = 0.2$ is set for non-greedy one.

3.2. Multi-agent reinforcement learning design

Note for the MRL implementation that the reward computation depends on both an agent's V value as well as those of other agents, and will be used to decide whether coordination among agents are needed to take place [26]. Such a coordinated decision also depends on the so-called safe inter-agent distances which are used to ensure that each agent will not receive negative rewards. Such distances consist of minimum distance $\zeta_{min} = 2$ and maximum distance $\zeta_{max} = 4Z + 2\sqrt{2}$ values. In this regard, all agents will be assigned with a logical *flag* marker to indicate if the inter-agent distance between every two agents remains safe or not. Given the status of the flag marker, the agents' rewards are then defined as one of the following possibilities: i) $W(p,u) = [-15, -15]$ when agents are off the environment, ii) $W(p,u) = [(-1.5 \times \rho),\ (-1.5 \times \rho)]$ when coverage overlap exists and exceeds the threshold, and iii) $W(p,u) = [w_1(p_1,u_1),\ w_2(p_2,u_2)]$ when action-state relations do not affect agents. Note that if an agent is in safe distance situation, then the agent will receive reward according to the rules set forth for the case of single agent reinforcement learning case.

When two or more agents know their rewards, each agent will broadcast its Q-value to the other to allow all agents to compute their joint actions and Q values [27]. Such knowledge of each agent's state and action as well as the joint states and actions, then negotiation among interacting agent can be performed to decide the best action for each agent. This negotiation can be conducted in reference to the utility value $\Upsilon_i(p,u)$ of each agent which in the case of MRL is equivalent to the joint Q value of all agents, i.e. $V_i^J(p,u) = \Upsilon(p,u)$. Based on such values, the negotiation can be done accordingly to possibly achieve a Nash equilibrium among all actions of all agents [11]. In particular, even if such a Nash equilibrium does not exist, one may also resort to find more relaxed equilibrium values such as *non-strict* EDSP or *meta-equilibrium* [10], [11].

4. RESULTS AND DISCUSSION

4.1. Single agent reinforcement learning simulation

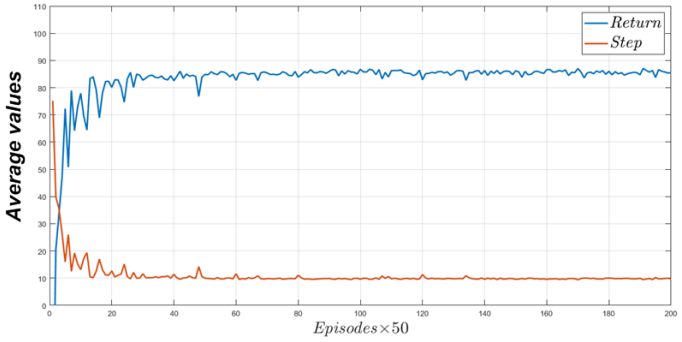
The SRL simulation was conducted on a box-shaped box with a dimension of $[16 \times 16 \times 4]$ such that the maximum height of the quadrotor is four (4) cells with an achievable optimal coverage area of $\lambda = 49$ cells. The agent is initialized at cell $(1,1,1)$ and the learning parameters were set to $\alpha = 0.9$ and $\gamma = 0.9$. The decision making scheme is chosen to be the $\epsilon-$greedy policy with $\epsilon = 0.98$. The simulation was conducted for 10.000 episodes with a maximum of 150 steps at each episode. The simulation evaluates agents' movements and averaged rewards at every 50 episodes. The results of the SRL simulation are summarized in Figure 4.

Figure 4(a) plots the trend of the agent's return reward which shows that the agent successfully achieve an optimal coverage within the defined episodes and steps per episode. It can also be seen that learning process achieves an optimal value after 2000 episodes. Particularly at episode 1600, the agent obtained a return of 85 and achieves the optimal coverage of $\lambda = 49$. In the following episode, the agent requires only few steps to achieve optimal decision thereby showing the success of the proposed learning algorithm. Figure 4(b) also shows that the average number of agent exiting the environment or changing its movement are decreasing and then reach steady state values once the optimal coverage and return are achieved. Figure 5 further shows the profiles of the returned rewards for different SRL simulation settings. Figure 5(a) shows the consistent profile of the returned rewards when simulations were repeated five times, whereas Figure 5(b) depicts converging trend of the returned reward profiles to the optimal one when parameter ϵ is increased. These results demonstrate the robustness and reasonable choice of parameter values of the proposed SRL learning scheme.

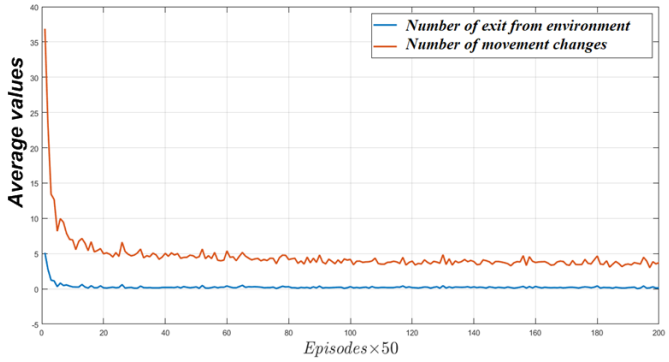
4.2. Multi-agent reinforcement learning simulation

Simulation of the proposed MRL approach was conducted for two quadrotors operational case. The operational environment size is similar to the SRL simulation case. The used learning parameters are $\alpha = 0.9$ and $\gamma = 0.9$, with $\epsilon_1 = 0.98$ and $\epsilon_2 = 0.5$ for agents 1 and 2, respectively. The simulation was conducted for 10,000 episodes with a maximum of 500 steps for each episode. The initial state of both agents are chosen randomly in such a way that their position distance is between 2 to 10 cells.

Figure 6 shows the simulation results of the proposed MRL scheme. As can be seen in Figure 6(a), both agents coordinate with each other in that whenever the first agent reach chooses a policy with high return value then the other agent chooses a policy with minimum return value, and vice versa. Figure 6(b) further shows that such a coordination in computing the next policy results in optimal agents placement which ensure the minimum/absence of overlapping coverage regions. These results thus illustrate the effectiveness of the proposed MRL scheme for optimal area coverage optimization of multiple quadrotors' operation.

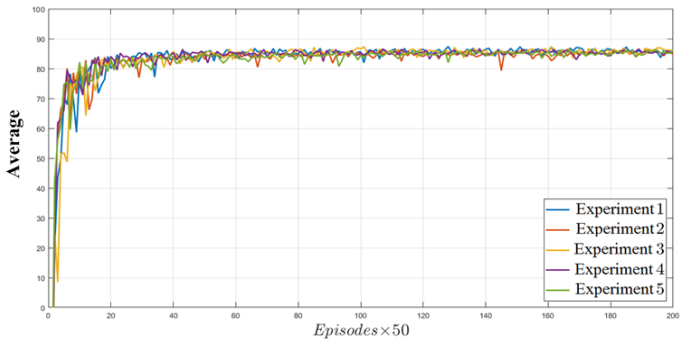


(a)

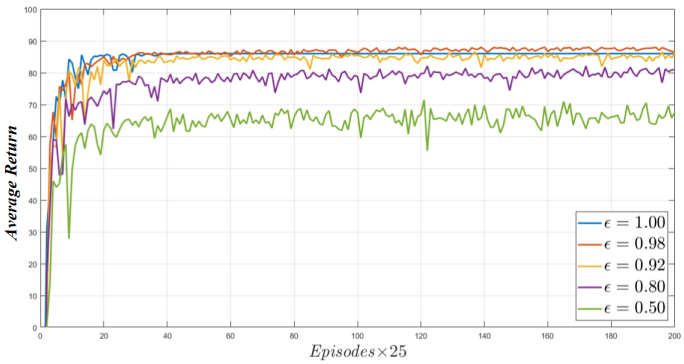


(b)

Figure 4. SRL simulation results: (a) average reward and (b) numbers of exit/change of move directions

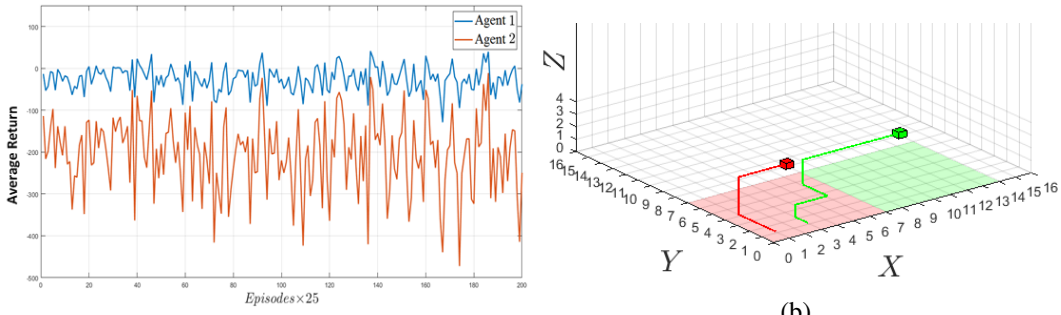


(a)



(b)

Figure 5. Average rewards of SRL for (a) 5 simulation experiments and (b) different ϵ values



(a) (b)

Figure 6. MRL simulation results: (a) average reward and (b) optimal placement

5. CONCLUSION

This paper has presented SRL and MRL schemes to maximize the area coverage of multiple quadrotors' operation. The proposed MRL scheme is equipped with a negotiation scheme which is triggered whenever the involved quadrotors enter a negotiation region that is defined by a predetermined inter-agent distance threshold. The presented simulation results for both single and multi quadrotors learning showed the effectiveness of the proposed method. Future works will be directed toward examining the extension of the methods developed in this paper to the case in which the dynamics of multi-quadrotor operations are taken into consideration.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support for this research from the Directorate General for Higher Education, Research, and Technology of the Ministry of Education, Culture, Research, and Technology (Kemdikbudristek) of the Republic of Indonesia under the Regular Fundamental Research grant year 2023 with contract number: 040/SP2H/RT-MONO/LL4/2023; III/LPPM/2023-07/118-PE, and from the internal research grant from the Institute for Research and Community Service (LPPM) at Parahyangan Catholic University.

REFERENCES

[1] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned aerial vehicles in agriculture: a review of perspective of platform, control, and applications," *IEEE Access*, vol. 7, pp. 105100–105115, 2019, doi: 10.1109/ACCESS.2019.2932119.

[2] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: a survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018, doi: 10.1002/net.21818.

[3] C. Ju, J. Kim, J. Seol, and H. I. Son, "A review on multirobot systems in agriculture," *Computers and Electronics in Agriculture*, vol. 202, 2022, doi: 10.1016/j.compag.2022.107336.

[4] D. P. Kumar, T. Amgoth, and C. S. R. Annavarapu, "Machine learning algorithms for wireless sensor networks: a survey," *Information Fusion*, vol. 49, pp. 1–25, 2019, doi: 10.1016/j.inffus.2018.09.013.

[5] P. R. -Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of UAV applications for precision agriculture," *Computer Networks*, vol. 172, 2020, doi: 10.1016/j.comnet.2020.107148.

[6] P. H. -Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019, doi: 10.1007/s10458-019-09421-1.

[7] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning & its application in different fields," *IEEE Access*, vol. 8, pp. 209320–209344, 2020, doi: 10.1109/ACCESS.2020.3038605.

[8] A. Nowé, P. Vrancx, and Y. M. D. Hauwere, "Game theory and multi-agent reinforcement learning," *Adaptation, Learning, and Optimization*, vol. 12, pp. 441–470, 2012, doi: 10.1007/978-3-642-27645-3_14.

[9] M. V. Otterlo and M. Wiering, "Reinforcement learning and markov decision processes," *Adaptation, Learning, and Optimization*, vol. 12, pp. 3–42, 2012.

[10] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the 11th International Conference on Machine Learning, ICML 1994*, 1994, pp. 157–163. doi: 10.1016/B978-1-55860-335-6.50027-1.

[11] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: a selective overview of theories and algorithms," *Studies in Systems, Decision and Control*, vol. 325, pp. 321–384, 2021, doi: 10.1007/978-3-030-60990-0_12.

[12] A. Din, M. Y. Ismail, B. Shah, M. Babar, F. Ali, and S. U. Baig, "A deep reinforcement learning-based multi-agent area coverage control for smart agriculture," *Computers and Electrical Engineering*, vol. 101, 2022, doi: 10.1016/j.compeleceng.2022.108089.

[13] H. X. Pham, H. M. La, D. F. -Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field cover," *Arxiv-Computer Science*, vol. 1, pp. 1–7, 2018.

[14] Y. Mekonnen, S. Namuduri, L. Burton, A. Sarwat, and S. Bhansali, "Review—machine learning techniques in wireless sensor network based precision agriculture," *Journal of The Electrochemical Society*, vol. 167, no. 3, 2020, doi: 10.1149/2.0222003JES.

[15] C. F. Hayes et al., "A practical guide to multi-objective reinforcement learning and planning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 1, 2022, doi: 10.1007/s10458-022-09552-y.
[16] L. Canese et al., "Multi-agent reinforcement learning: a review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, 2021, doi: 10.3390/app11114948.
[17] M. Natarajan and A. Kolobov, "Planning with markov decision processes: an AI perspective," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 17, pp. 1–203, 2012, doi: 10.2200/S00426ED1V01Y201206AIM017.
[18] W. Du and S. Ding, "A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications," *Artificial Intelligence Review*, vol. 54, no. 5, pp. 3215–3238, 2021, doi: 10.1007/s10462-020-09938-y.
[19] A. K. Sadhu and A. Konar, "An efficient computing of correlated equilibrium for cooperative q-learning-based multi-robot planning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–16, 2019, doi: 10.1109/TSMC.2018.2865488.
[20] W. Mao and T. Başar, "Provably efficient reinforcement learning in decentralized general-sum markov games," *Dynamic Games and Applications*, vol. 13, no. 1, pp. 165–186, 2023, doi: 10.1007/s13235-021-00420-0.
[21] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust reinforcement learning: a review of foundations and recent advances," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 276–315, 2022, doi: 10.3390/make4010013.
[22] K. L. -Brown and Y. Shoham, "Essentials of game theory: a concise multidisciplinary introduction," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 2, no. 1, pp. 1–88, 2008, doi: 10.2200/s00108ed1v01y200802aim003.
[23] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team Markov games," *Advances in Neural Information Processing Systems*, vol. 15, 2002, doi: 10.5555/2968618.2968817.
[24] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *Journal of Artificial Intelligence Research*, vol. 75, pp. 1401–1476, 2022, doi: 10.1613/JAIR.1.13673.
[25] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. Cambridge, Massachusetts: MIT Press, 2018.
[26] A. Zehfroosh and H. G. Tanner, "PAC reinforcement learning algorithm for general-sum markov games," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2821–2831, 2023, doi: 10.1109/TAC.2022.3219340.
[27] A. Grover, M. A. -Shedivat, J. K. Gupta, B. Yura, and H. Edwards, "Learning policy representations in multiagent systems," *35th International Conference on Machine Learning, ICML 2018*, vol. 4, pp. 2887–2897, 2018.

BIOGRAPHIES OF AUTHORS

Glenn Bonaventura Wijaya received his B.Eng. in Electrical Engineering from Parahyangan Catholic University, Indonesia in 2022. He is currently working as a distribution engineer at Riway International company. His research includes various applications of machine learning methods in multi-agent systems analysis and control applications. He can be contacted at email: 2017630019@student.unpar.ac.id.

Tua Agustinus Tamba is an assistant professor in the Department of Electrical Engineering at Parahyangan Catholic University, Indonesia. He received both his MSEE and Ph.D. in Electrical Engineering from University of Notre Dame (USA) in 2016, MSc in Mechanical Engineering from Pusan National University (Republic of Korea) in 2009, and BEng in Engineering Physics from Institut Teknologi Bandung (Indonesia) in 2006. His research interests include dynamical systems, control theory, and optimization with applications in mechatronics, robotics, automation systems, and systems biology. He can be contacted at email: ttamba@unpar.ac.id.