A Markovian Decision Process Analysis of Experienced Agents Joining Ad-Hoc Teams

1st Roghaiyeh Heidari
Institute for Advanced Studies in Basic Sciences
Gava Zang, Zanjan, Iran
r.heidari@iasbs.ac.ir

2nd Mohsen Afsharchi
Department of Engineering
University of Zanjan
Zanjan, Iran
afsharchi@znu.ac.ir

3rd Reza Khanmohammadi
Department of Electrical, Biomedical and Mechatronics Engineering
Qazvin Branch, Islamic Azad University
Qazvin, Iran
reza926@gmail.com

***Abstract*—As the number of agents grows in a multi-agent system, it is very impractical to have a team that is pre-programmed to share a cooperation protocol. In this situation experienced agents that have been trained to cooperate with different teams come in handy. In this paper based on some UAV teams, we investigate the behavior of an experienced agent which we call ad-hoc agent. First, we train the ad-hoc agent to cooperate with different teams and in different environmental situations. This training is based on an approximated version of MDP which is very fast. Then the ad-hoc agent joins the teams and tries to cooperate with them in a Persistent Surveillance Mission(PSM). Our experiment for the ad-hoc agent starts with joining different fixed strategy teams and ends with joining to a fully cooperative team. The results of the simulation show that the performance of a team having an ad-hoc agent is even comparable to a team that has completely trained together.**

***Index Terms*—Ad-hoc Team, Multi-UAV Systems, Cooperation**

I. Introduction

In a multi-agent system, as the number of agents grows, it is expected that they will interact and cooperate with other agents that are coming from different companies, leading to the situation where these agents may not be pre-programmed to share a common coordination protocol. This problem motivates the area of ad-hoc teamwork in which an agent may potentially cooperate with a variety of teammates in order to achieve a shared goal. the concept of the ad-hoc team is about cooperation without pre-coordination [1]. Barrett et. al. have launched the concept and created an evaluation framework for ad-hoc teams in which the teammates must collaborate in the pursuit domain [2].

Unlike cooperative team modeling in an ad-hoc team setting, agents use the previous experiences to obtain the best action in each situation [3]. The problem becomes more challenging since agents can be heterogeneous and the control might not be centralized. In [4] authors proposed the PLASTIC framework which stands for Planning and Learning to Adapt Swiftly to Teammates to Improve Cooperation. Using that method, an ad-hoc agent uses past experiences to find out the best way to cooperate with new teammates. Two approaches for using past experience are proposed. The first one is estimating the model of unknown teammates, and the other one is finding the best-fitted policy of collaborating with them. Markov Decision Process as a learning framework is used to model the problem in these work [3]. We used the second approach to experiment with a UAV ad-hoc teams, which is broadly used nowadays [5]. UAVs are playing important role in a vast range of applications such as Search and Rescue, Area Surveillance and Urban Management. For instance, DIJ Company has reported that ”At least 59 lives have been saved by civilian drones in 18 incidents” [13]. Generally, UAVs are fast and easy to use when some missions are impossible for human to carry out, that is why they have been studied a lot in this area.

The main contribution of this paper is that in the ad-hoc team setting, we train an ad-hoc agent to cooperate with different teammates and in different environmental situations.We use an approximate multi-agent learning method to train the ad-hoc agent in a fast way. Then we assess the performance of different teams when this ad-hoc agent faces unknown teammates, and uses its previous experiences and learned policies to choose the best one. This assessment is also done in the different environmental situation which affects the agents' policies.

The experimental setup is Persistent Surveillance Mission (PSM) problem, which is introduced in [10]. In PSM problem, a multi-UAV team is responsible for surveying an area with minimum cost. The health and fuel level of UAVs is modeled in state space of the Markov Decision Process. The problem is solved using a modified version of the value-iteration algorithm. In the learning step, the ad-hoc agent learns the best action to collaborate with various teammates and in simulation step, this agent chooses one of the previously learned actions in cooperation with unknown teammates using a greedy method. The control is decentralized, teammates are heterogeneous and the observability of the environmental parameters are approximated in the states of the MDP [6]. In the evaluation step, the team performance is calculated with and without an ad-hoc agent. The remainder of this article is organized as follows. Section 2 provides some basic information about the ad-hoc team. Section 3 introduces the PSM problem. A detailed version of problem setting is described in Section 4. Section 5 describes the approximate MDP model of the PSM and in Section 6 the experiments are presented. Section 7 contains the results and finally in Section 8 comes our conclusion as well as the future work.

II. Ad-hoc Team

The cooperative team is a well-studied concept in multi-agent systems. When a task cannot be completed by an agent, a cooperative group of agents will be formed to do the task. Multi-agent teams are groups of agents dedicated to a particular task in modern multi-agent environments. For instance, a team of autonomous UAVs cooperates with each other to do the surveillance task with minimum cost.
In some cases, agents get involved in a pre-coordination process and learn how to cooperate with each other. In some other cases, agents do the team work without any pre-coordination. It is even possible that the teammates are unknown to each other, and they may not have any knowledge of the environment either. These teams are known as "**ad-hoc teams**" [3]. An ad-hoc team is defined generally as the following [1], [4]: "An ad-hoc team setting is one in which teammates must work together to obtain a common goal, but without any prior agreement regarding how to work together".

The difference between a normal and an ad-hoc team is shown in part (a) and (b) of Fig.(1) respectively. In an ad-hoc team, there is no pre-coordination and the dynamic of the environment is unknown to the agents. Acting optimally in an ad-hoc team is a very complicated task.
In this article, similar to the work in [9], we want to train an agent to do the best action facing unknown teammates. We called this member, "ad-hoc agent" or "ad-hoc UAV" in our context. The ad-hoc agent has to estimate teammate behavior and select the best action accordingly. The other members operate autonomously and the control is decentralized.

III. PSM Problem

The objective of agents in a persistent surveillance mission (PSM), is to continuously survey a pre-specified the region of interest and to closely track any objects of interest discovered there [6]. In the problem, the whole region is divided into three distinct locations: surveillance area, communication relay, and the base. Some targets are moving in the surveillance area and the mission is to search for targets in the tasking area while continuously tracking those that have already been detected. There must be an agent in the communication area which is responsible for connecting the base and the surveillance area. In addition, each of the available agents in the mission may have problems due to the sensor/actuator failure or the shortage of fuel. In these cases, the agent in the communication relay can act as an alternative to the agents in the surveillance area. In addition, the agent whose sensor is damaged still can do the communication task. Also, it is assumed that if the agent gets to the base, it will be repaired and refueled. The absence of UAV in the communication and surveillance area or the lack of the required number of UAVs in the surveillance region will impose some costs on the mission. The objective is to minimize the cumulative cost of the mission. In Fig. (2), an overview of the PSM problem is shown.

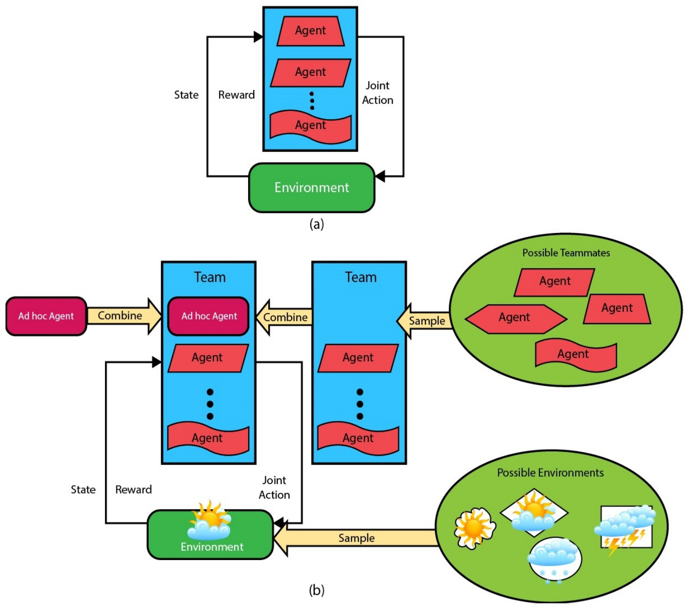

Fig. 1. (a) A normal team. (b) An ad-hoc team [3].

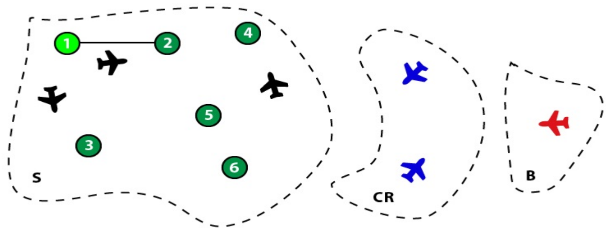

Fig. 2. Persistent Surveillance Mission;"**S**" is the Surveillance area, "**CR**" is the Communication Relay and "**B**" is the base.

IV. Problem Description

Assume that we have a UAV team doing PSM and we want to add another ad-hoc UAV to the team without pre-coordination. This is a full version of an ad-hoc team problem and illustrates how an ad-hoc UAV tries to match itself with unknown teammates and in an unknown environment. The problem setting is shown in Fig. (3). The fully cooperative behavior that an ad-hoc agent can do at each state while facing with completely unknown teammates and environments is a very challenging goal.

We assume that the ad-hoc UAV had previous experiences with other teammates and different environments. But at the simulation time, it doesn't know which UAV is cooperating with or how the environmental condition is now. The ad-hoc agent has to choose one of the previously learned policies at each simulation step.
We formulate the mission with 3 or 4 UAVs. Each UAV has 3 types of health status, "healthy", "sensor failure" and "actuator failure". In our setting, we have one healthy UAV that must be in the communication relay and if this condition is disregarded, the mission will fail due to the fact that UAVs will not be able to transmit any information of surveillance area to the

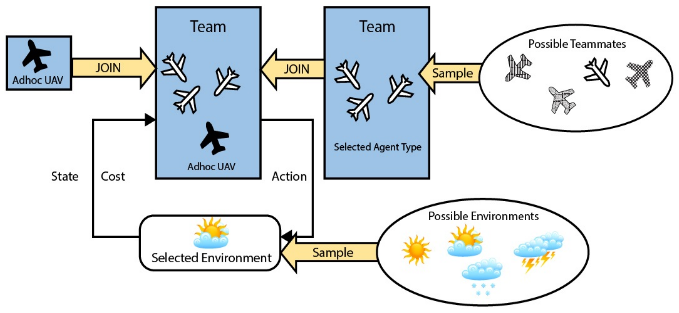

Fig. 3. An ad-hoc UAV agent in an unknown team and unknown environment.

base. In this circumstances, Fail_cost will be imposed on the system. Also in the 4-UAV team, n_{dmnd} is 3 that means 3 healthy UAVs are needed to monitor the surveillance area. In general, n_{dmnd} refers to the number of healthy UAVs, which are needed to monitor the surveillance area. On the 3-UAV team, this number is equal to 2. If the number of healthy UAVs in the surveillance area gets less than this value, Gap_cost will be considered in the system.

Each UAV may face with sensor or actuator failure with 0.1 or .05 probabilities respectively. The fuel consumption unit (i.e. traveling from one area to another) is 1 or 2 with a probability of 0.5.

Because in the mission UAVs must avoid crashing, we set the Crash_cost to a very high value which is 1000 times worse than Gap_cost. This value is set to make sure that none of the UAVs finishes their fuel in the middle of the mission. Failing in the mission imposes a larger cost than the gap cost. When a UAV team fails, it means that there is no way to send any information from the surveillance area to the base. So, we assume Fail_cost 20 times worse than the gap cost.
The goal is doing the surveillance job at minimum cost. The cost function will be described in Section (V-D).

V. Approximated MDP Model

Based on the model presented in [6] we use an approximated state representative in our multi agent Markov Decision Processes (MMDP) to coordinate our ad-hoc team work. The decentralized approach helps the cooperating agents to deal with uncertainties in fuel consumption and health-related issues. In this article, we model the problem from the ad-hoc UAV point of view.
A Markov Decision Process (MDP) includes the states set, the actions set, a reward function and a transition function [12]. More formally, in an MDP, all the following parameters should be identified.

- S: is a finite set of states
- A: is a finite set of actions
- $T(s,a,s')$: is the probability of transition from state s to state s' with action a
- $C(s)$: immediate reward/cost receiving in the state s

The core problem of an MDP is to find a "policy" function which specifies the best action that the agent will choose when the agent is in the state s. Our modeling of PSM as an MDP is as the following.

A. States

The states of the whole team are combination of the state of the ad-hoc UAVs(S_a) and the status of teammates(S_{team}). So, for the set of states we have:

$$S = S_a \times S_{team} \tag{1}$$

We show the state of a UAV in the PSM problem with 3 parameters: location, health, and fuel, which is represented as a tuple $< l, h, f >$. Each of these parameters comes from the following sets.

$$l \in L = \{base, communication\_relay, surveillance\} \tag{2}$$

$$h \in H = \{healthy, sensor\_failure, actuator\_failure\} \tag{3}$$

$$f \in F = \{0, 1, \ldots, 8\} \tag{4}$$

All the possible status for S_a is represented in the Table (I). The *loc*, *health* and *fuel* columns are the corresponding numerical values for the location of the UAV and its health and fuel. For instance $l = 1$ means base and $h = 2$ means sensor failure. Since in the base, the fuel amount is 8 and the agent is in a healthy condition, and the fuel consumption from one region to another is at least one unit, there are overall 46 status.

TABLE I
All the possible statuses for one UAV

Status	loc	health	fuel	Status	loc	health	fuel
0	1	1	8	23	2	3	6
1	2	1	0	24	2	3	7
2	2	1	1	25	3	1	0
3	2	1	2	26	3	1	1
4	2	1	3	27	3	1	2
5	2	1	4	28	3	1	3
6	2	1	5	29	3	1	4
7	2	1	6	30	3	1	5
8	2	1	7	31	3	1	6
9	2	2	0	32	3	2	0
10	2	2	1	33	3	2	1
11	2	2	2	34	3	2	2
12	2	2	3	35	3	2	3
13	2	2	4	36	3	2	4
14	2	2	5	37	3	2	5
15	2	2	6	38	3	2	6
16	2	2	7	39	3	3	0
17	2	3	0	40	3	3	1
18	2	3	1	41	3	3	2
19	2	3	2	42	3	3	3
20	2	3	3	43	3	3	4
21	2	3	4	44	3	3	5
22	2	3	5	45	3	3	6

Having n UAVs the state space of the problem grows exponentially to 46^n. If we have enough time, the training phase can run on exact (non-approximate) MDP or MMDP model and find the best policy before starting off the mission. Of course, when the team is large (i.e. more than 3 UAVs), it is difficult to solve the exact MDP model which has more than 4 million states which enforces us to use an approximated model [6].

As it is stated earlier, the status of teammates is important in PSM problem because every UAV has to know whether a healthy UAV is in the communication relay or not and how many healthy UAVs are in the surveillance area at the moment. Incorporating this exact information makes the state space of our ad-hoc agents very large. But all the information is needed for the ad-hoc agent to choose the next action is just two numbers. The first one is a flag (c), that shows the existence of a healthy agent in the communication relay. If this number is 1, it means at least one healthy UAV exists there and when it is zero, it means that there is no agent in the communication relay. The second number, n_s, is the number of healthy UAVs in the surveillance area. This parameter can be any integer value between 0 and $n-1$, where n is the number of UAVs in the team. It means that the number of UAVs in the surveillance area may be zero in the worst case. We use $S_{team} =< c, n_s >$ to represent the status of the team. As a result, the status of the whole team in our model can be represented as (5).

$$s =< l, h, f, c, n_s > \tag{5}$$

For example, consider a PSM mission with 4 agents (i.e. $n = 4$). In this case, S_{team} for 3 teammates can be shown as the following set (6).

$$\begin{aligned} S_{team} = \{ &< 0, 0 >, < 0, 1 >, < 0, 2 >, < 0, 3 >, \\ &< 1, 0 >, < 1, 1 >, < 1, 2 > \} \end{aligned} \tag{6}$$

To illustrate more, when the state is $< 3, 1, 4, 1, 2 >$, the ad-hoc UAV is in the surveillance area. It is healthy and has 4 units of fuel. One healthy UAV is in the communication relay and 2 healthy UAVs are in the surveillance area at this moment. Obviously, the state space is the Cartesian product of S_a and S_{team}. So, in this example with 4 UAVs, we have $46 * 7 = 322$ states. As it is seen the number of the state space has been significantly dropped compared to the exact model (without approximation), which is $46^4 = 4477456$.

B. Actions

The space of possible actions for each UAV is $\{1, 0, -1\}$ where number 1 means "Move toward Surveillance", number 0 means "Stay", and number -1 means "Move toward Base". So for the action set we have:

$$A = \{-1, 0, 1\} \tag{7}$$

C. Transition Function

We divided the state space into 2 sets, S_a that shows the ad-hoc agent and S_{team} which is representing the teammates approximated state. So, the description of the transition function is divided into 2 parts. The first part is about the transition probability of the ad-hoc agent, which is represented by P_a and the second part is about the transition probability of teammates, which is represented by P_{team}.

We calculate P_a based on the probability of sensor failure (i.e. PS) or actuator failure (i.e. PA) and the fuel consumption (i.e. PF) which may be one or two gallons in each step. The next "location" of the UAV is determined based on the selected action. For example, having the following values.

$$PA = 0.05, PS = 0.1, PF = 0.5 \tag{8}$$

then the probability of moving from state $s_a =< 2, 1, 7 >$ to $s_a =< 3, 3, 6 >$ by moving forward equals to 0.025. The second part of the transition function is calculated by sampling. So, we simulated all the possible states for these teammates and calculated the P_{team} by using the rules that had been coded. The probability of the transition function is the multiplication of P_a by P_{team} as shown in (9).

$$T(s, a, s') = P_a \times P_{team} \tag{9}$$

D. Cost Function

The cost function is a fundamental element of our problem as it quantifies the quality of each state. As the control is distributed, the ad-hoc UAV has to calculate the cost of each state individually. The only known parameters from teammates situations is $S_{team} =< c, n_s >$.

The costs in PSM problem are categorized into 3 different types [6]. For the UAV that finishes its fuel in the middle of its mission, we consider the crash cost (i.e. Crash_cost), in case there is no UAV in the communication relay or in the surveillance area, we consider the fail cost (i.e. Fail_cost), and for the case that the number of the required UAVs in the surveillance area is less than the expected limit, we consider the gap cost (i.e. Gap_cost) according to the shortage of the number of the required UAVs.

For example, for a mission with 3 UAVs, one UAV is considered performing the communication task and two UAVs are considered performing the surveillance mission. When there is no UAV in the communication relay or the surveillance areas, the fail cost is imposed to the whole set. In case there is one UAV in the communication relay and there is one UAV in the surveillance area, the gap cost will be applied. Accordingly, the cost of each state can be calculated. For instance, for state $< 1, 1, 8, 0, 1 >$ we impose Fail_cost because no healthy agent is in communication relay to do the communication task.

The Algorithm (1) demonstrates the way we impose the cost on each state. Here n_{serv} and n_{comm} are local variables to show the number of healthy UAVs in the surveillance and communication relay area respectively and n_{dmnd} is a fixed value in the problem to specify the number of needed healthy UAVs in the surveillance area.

By specifying all the parameters of the MMDP we can solve the problem. When the MMDP is solved, we will achieve one policy $\pi : S \rightarrow A$, which specifies what action must be done in each state. We use value-iteration to find the best policy for each state in our experiments.

VI. EXPERIMENTS

The main focus of our experiments is on the training of an ad-hoc UAV before starting the mission and allowing this agent

Algorithm 1 CalculateCost

```
1: function CalculateCost($s < l, h, f, c, n_s >$) returns $cost$ a real value
2: local variables: $n_{serv}$, $n_{comm}$
3: $n_{serv} \leftarrow n_s$
4: $n_{comm} \leftarrow c$
5: $cost \leftarrow 0$
6: if ($l = 3, h = 1, f > 0$) then
7:     $n_{serv} = n_s + 1$
8: else if ($l = 2, h < 3, f > 0$) then
9:     $n_{comm} = 1$
10: else if ($f = 0$) then
11:     $cost \leftarrow Crash\_cost$
12: end if
13: if ($n_{serv} = 0$ or $n_{comm} = 0$) then
14:     $cost \leftarrow Fail\_cost + cost$
15: else if ($n_{dmnd} > n_{serv}$ and $n_{comm} = 1$ and $n_{serv} > 0$) then
16:     $cost \leftarrow (n_{dmnd} - n_{serv}) \times Gap\_cost + cost$
17: end if
18: return $cost$
19: end function
```

to join a team with different settings. The ad-hoc settings are categorized into two parts. Unknown teammates and unknown environments. In the first section of our experiments, we show how an ad-hoc UAV cooperates with unknown teammates. In the second part, the behaviour of an ad-hoc UAV is evaluated in some unknown environments.

A. Unknown Teammates

Three types of fixed strategy UAVs are created and we calculated and saved the transition function of these teams as 3 separate matrices (i.e. P_{team}) (See Section VI-B). The ad-hoc UAV learned the best policy in each team by solving the approximated MDP using the value-iteration algorithm. So, this agent has 3 policies and is enabled to choose one of them at each simulation step. The brief abstraction of our scenario is illustrated in Fig. (4).

The black UAV is the ad-hoc one and the others are fixed strategy UAVs. First, the ad-hoc UAV learns how to cooperate with three different teammates. The output of the training phase, which is shown in the top of Fig. (4), is 3 policy functions. These functions have the best policy for each state. Then, at each time step, the ad-hoc UAV decides to choose one of the previously learned policies.

The online decision algorithm which we call "**Policy-Selection**" is shown in Algorithm (2) which is a greedy algorithm. The ad-hoc UAV will choose the policy that minimizes the immediate cost. It uses the `CalculatePolicyCost` function at line 5. This function aggregates the cost of each state "*state*" which is reachable from the current state "*s*" by following the policy "*p*". The reachable states are the output of the function "*Create_Next*". Accordingly, each state is reachable by a probability which is calculated in *Trans* function. Then, the probability of the transition to state "*state*" is multiplied by the cost of the state *state*. Finally, all these values are summed up for each policy in variable *total_cost*. This value is returned to "*tempCost*".

Algorithm 2 Policy-Selection

```
1: function Policy-Selection( $s$: is a state, $Policy$ is an array of 3 learned policies ) returns $policy$
2: $minCost \leftarrow MAX\_VALUE$
3: $tempCost \leftarrow 0$
4: for each $p$ in $Policy$ do
5:     $tempCost = CalculatePolicyCost(s, p)$
6:     if $minCost > tempCost$ then
7:         $minCost \leftarrow tempCost$
8:         $policy \leftarrow p$
9:     end if
10: end for
11: return $policy$
```

Algorithm 3 CalculatePolicyCost

```
1: function CalculatePolicyCost($s$,$policy$ ) returns $total\_cost$
2: $total\_cost \leftarrow 0$
3: $next\_states \leftarrow Create\_Next(s, policy)$
4: for each $state$ in $next\_states$ do
5:     $total\_cost = Trans(s, policy, state) \times CalculateCost(state) + total\_cost$
6: end for
7: return $total\_cost$
```

Now we describe the team formation method.

B. The Teams Formation

To compare the efficiency of an ad-hoc agent in a team we created 6 different teams. The first 3 teams are not intelligent and use a fixed strategy.

- Random; Each UAV chooses the next action randomly.
- Risky; A "Risky" UAV in the team chooses the "move toward base" action in 8% of times, "stay" action in 25% of times and "moving toward surveillance" in 67% of times.
- Conservative; A "Conservative" UAV chooses the "move toward base" action in 50% times, "stay" action in 33% of times and "moving toward surveillance" in 7% of times.

In these teams, all the UAVs behavior is the same. Some conditions are hard coded into UAVs to avoid the crash during the mission. For instance, the UAV which has just less than 3 units of fuel, and is located in the communication relay forced to move toward the base because it has to avoid crashing. To make it clear we need these teams just for comparison and no ad-hoc agent will get involved in them as you will see in the next subsection.

We also have three intelligent teams in which they include at least one experienced ad-hoc agent.

- PoliSel1; This team has just one intelligent ad-hoc UAV with learned policies before starting the mission. The other teammates are from Random, Risky or Conservative teams.
- PoliSel2; This team is trained to work cooperatively and one ad-hoc UAV which is trained independently will join them.
- MMDP; This is a fully cooperative intelligent team in which agents learned to cooperate with each other in the mission.

As it is seen we have two "**PoliSel**" teams. On the first team, PoliSel1, the other teammates have variant behaviors selected from previously mentioned teams. In other words, at each simulation step, 3 other UAVs choose one of the 3 behaviors to follow. On the second team, PoliSel2, the other teammates are trained and know the best action in each state where agents trained to cooperate with other agents. In these two teams, the ad-hoc agent has to adapt itself in each step. The UAV chooses one action which minimizes the cost at the moment Using Algorithm (2). The name "PoliSel" is used for these teams because of the "Policy-Selection" algorithm is used at each simulation step.

The last team, which is known as "**MMDP**", is a fully intelligent team. It stands for **M**ulti-Agent "**MDP**" which is used in [6]. It means that all the UAVs learned the best policy to cooperate with each other. We need this team to show the difference between the team with just one intelligent ad-hoc agent with the case when all teammates are treating intelligently and doing the best action.

This team is the main goal in all multi-agent systems. The MMDP model guarantees the full cooperative behavior which leads to the minimum cost or the best reward for the team.

C. Unknown Environments

For unknown environments, the teammates policy in each team is fixed. Assume that we want to improve one team's performance by just adding some more experience to one agent. The training phase of this experiment is very fast and took less than a second. Three different weather conditions are simulated by manipulating PS, PA and Fail_cost. Then the ad-hoc agent is trained to do the best action in different weather conditions. The environment parameters are shown in Table (II).

PoliSel1 is formed by random UAVs with one ad-hoc UAV. The ad-hoc UAV is trained to act in three different environments (see Table (II)). The UAV keeps the best actions for each state in 3 different arrays called Policy1, Policy2, Policy3 respectively. The ad-hoc agent decides online and

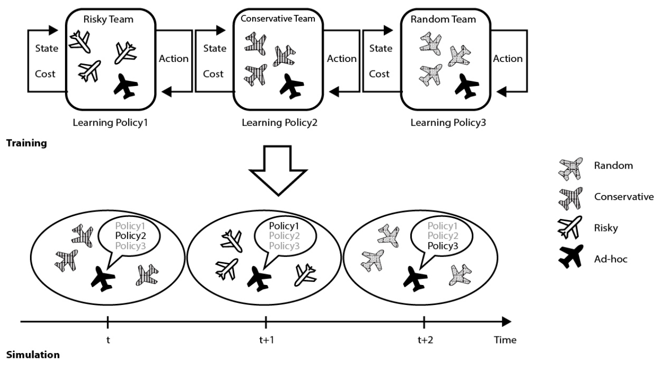

Fig. 4. The schema of the experiment. The top of the image shows the training step; the bottom illustrates the simulation time with policy selection.

TABLE II
PARAMETERS TO MAKE DIFFERENT ENVIRONMENTS

Environment Name	PA	PS	Fail_Cost
Environmet1	0.1	0.1	20
Environmet2	0.25	0.4	15
Environmet3	0.05	0.05	10

select the policy that should follow. The algorithm for training and policy selection are the same as Unknown Teammates which is presented in section (VI-A). The other teams are formed like the previous experiment. So, PoliSel2 includes one ad-hoc UAV and the other agents are trained as a team. In the same way, MMDP is a fully cooperative team in which all the agents are trained together.

The performances of teams are compared to a fully random UAV team and a fully cooperative UAV team which is trained under the following parameters.

$$PS = 0.1, PA = 0.05, Fail\_Cost = 20 \qquad (10)$$

D. Evaluation

We conduct an experiment to evaluate the performance of UAV teams in the PSM problem based on the cost function which is defined earlier. We calculate the cumulative cost of the mission in 1000 steps of the simulation and the goal of our experiments is to minimize this value. As we said before, the UAVs are avoiding the crash while doing their mission. So, we just use two costs for the measure: Fail_cost and Gap_cost. We count the number of gaps and fails occurred during the mission. To make a unified cost function, we use (11) to combine fail and gap cost where the coefficient μ is the balancing factor. Using this equation, we have exactly one value to compare teams.

$$Ev(team) = \mu \times count(fail) + count(gap) \qquad (11)$$

We compare two groups of teams which are described previously where our focus is on the performance of "**PoliSel1**" and "**PoliSel2**". We set μ to 20 in our evaluations which means the fail event imposes 20 times worse cost than the gap.

E. Implementation

Our learning program is coded in the Visual Studio 2015 environment in C++ with a system with Intel Core i5-3320 Processor and 8GB RAM .
Learning ad-hoc UAV is equal to solving the approximate MDP which is described in Section (V). The learning process is very fast and it takes less than one second. Training the MMDP team by solving the exact MDP for just 4 UAVs with value-iteration in the PSM mission takes about 3 weeks [6]. Using an optimized version of value iteration, we decreased this time to 2 hours for 40 iterations. This optimized version is shown in Algorithm (4). Considering the fact that the reachable states from the current state by doing an action is limited in the PSM problem, the value-iteration algorithm's runtime can be optimized. As it is shown in Algorithm (4), line 8 is responsible for finding an action which minimizes the cost. Here the second part of the summation (i.e. (1)) can be solved in a linear time because of the sparseness of the reachable states in the PSM problem.

Algorithm 4 PSM-VALUE-ITERATION

```
1:  function PSM-Value-Iteration($Trans, Cost$) returns a $Policy\ \pi$
2:  inputs: $Trans$, a transition model
        $Cost$, a cost function on states
3:  local variables: $Value$, value function, initially identical to $Cost$
4:  $Valuep$, temprory value function, initially identical to $Cost$
5:  repeat
6:      $Value \leftarrow Valuep$
7:      for each state $i$ do
8:          $Valuep[i] \leftarrow min_a \underbrace{\sum_j Trans^a_{ij} \times Value[i]}_{1} + Cost[i]$
9:          $\pi[i] = a$ , $a$ is the action which minimize Value[i]
10:     end for
11: until Close-Enough($Value$,$Valuep$)
12: return $\pi$
13: end function
```

VII. Results

In the following we bring the results of our experiments and compare the performance of different teams in different environmental situations.

A. Unknown Teammates

Table (III) and Table (IV) show the fail and gap costs and the corresponding Ev for 3 and 4 UAVs respectively. Overall, what stands out from the results is that more intelligence in teammates leads to less fail event count in the mission and that is because trained UAVs are aware of effect of fail against the gap. Another interesting point is that the MMDP is the best team (as it is expected) and PoliSel1 is remarkably worse than MMDP but, the performance of the PoliSel2 and MMDP teams are very close. This shows that in PoliSel2 the ad-hoc agent can adapt itself to the team. On the other hand, in PoliSel1 just one ad-hoc UAV is the intelligent member and the other teammates are fixed strategy agents.
Looking at the details, it is obvious that the performance of Conservative team is about 3 times worse than MMDP. It is significant that just one ad-hoc UAV made non-intelligent teams better. For instance, PoliSel1 had 50% less fail count than the Conservative team, 45% less than the Random team and approximately 24% less than the Risky team in the team of 4 UAVs.

To sum up, by looking at Fig. (5), you can see the evaluation function for PoliSel1 is clearly better than non-intelligent teams. It means that by spending a little time for training just one agent in an ad-hoc team setting, the performance gets at least 17% better in the team of 4 UAVs and 11% better in the team of 3 UAVs. There is less than 10% difference between Ev in MMDP comparing with PoliSel2 teams.

TABLE III
The results of experiment for 3 UAVs

Team Name	count(gap)	count(fail)	Ev(Team)
Conservative	102	893	17962
Random	110	880	17710
Risky	200	757	15340
PoliSel1	296	667	13636
PoliSel2	606	346	7526
MMDP	693	306	6813

TABLE IV
The results of experiment for 4 UAVs

Team Name	count(gap)	count(fail)	Ev(Team)
Conservative	299	842	17139
Random	356	812	16596
Risky	513	695	14413
PoliSel1	756	560	11956
PoliSel2	1374	218	5734
MMDP	1432	200	5432

B. Unknown Environments

The performance of teams is depicted in Table (V) for a team of 3 UAVs and in Table (VI) for a team of 4 UAVs. In general, 4 teams are compared and as it is expected the instability of the weather condition makes team performance worse than what's presented in Section (VII-A). Obviously, the more intelligence in teams, the better performance we get. In contrast with Section (VII-A), the significant point here is better performance of PoliSel2 compared to MMDP. That is because PoliSel2 has one UAV which is familiar with 3 types of environments and can adapt the policy based on variant weather conditions.

In detail, when other teammates are intelligent, having an ad-hoc experienced agent makes the team performance better

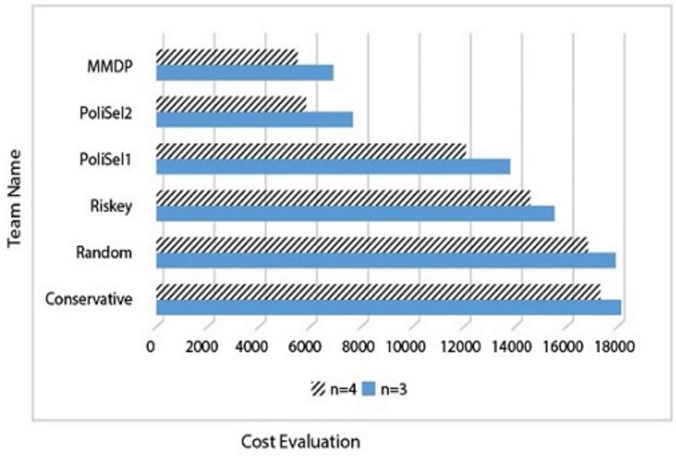

Fig. 5. The evaluation function comparison of different teams of 3 and 4 UAVs in the unknown teammates experiment.

even if the experience gained from an approximate method. Remember that training an ad-hoc UAV is very fast [6]. Even though the training time for the whole MMDP team is longer than PoliSel2, evaluation function output for PoliSel2 and MMDP are very close. The interesting point is that PoliSel2 is marginally better.

Some other concrete evidence in the results shows that PoliSel1 with just one approximately trained UAV makes the team better, at least 10% than a Random team in 3 UAVs team and 17% better in 4 UAVs team.

TABLE V
THE RESULTS OF EXPERIMENT FOR 3 UAVS

Team Name	count(gap)	count(fail)	Ev(Team)
Random	117	878	17677
PoliSel1	204	787	15944
PoliSel2	421	547	11361
MMDP	450	549	11430

TABLE VI
THE RESULTS OF EXPERIMENT FOR 4 UAVS

Team Name	count(gap)	count(fail)	Ev(Team)
Random	252	865	17552
PoliSel1	572	696	14492
PoliSel2	988	445	9888
MMDP	911	455	10011

VIII. FUTURE WORK AND CONCLUSION

In this paper, we utilized a fast and approximate way of training ad-hoc agent in the PSM problem to investigate the behavior of an experienced agent in different teams. We used our policy selection algorithm to select the best policy to cooperate with unknown teammates or environments based on the experiences of the ad-hoc agent. The cost is reduced significantly when the ad-hoc agent joins the team of fixed startegy. The results revealed that not only the team performance is much better than fixed strategy teams but also comparable to the fully cooperative teams. We will do some investigation on the online learning in future work. Also, the learning was only for a team with 3 and 4 agents and we would like to see the performance of our idea in larger teams.

REFERENCES

[1] Stone, Peter, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. ”Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination.” In AAAI. 2010.
[2] Barrett, Samuel, Peter Stone, and Sarit Kraus. ”Empirical evaluation of ad hoc teamwork in the pursuit domain.” In The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 567-574. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
[3] Barrett, Samuel, Avi Rosenfeld, Sarit Kraus, and Peter Stone. ”Making friends on the fly: Cooperating with new teammates.” Artificial Intelligence 242 (2017): 132-171.
[4] Barrett, Samuel, and Peter Stone. ”Cooperating with Unknown Teammates in Complex Domains: A Robot Soccer Case Study of Ad Hoc Teamwork.” In AAAI, pp. 2010-2016. 2015.
[5] Meng, Wei, Zhirong He, Rong Su, Pradeep K. Yadav, Rodney Teo, and Lihua Xie. ”Decentralized multi-UAV flight autonomy for moving convoys search and track.” IEEE Transactions on Control Systems Technology 25, no. 4 (2017): 1480-1487.
[6] Redding, Joshua David. ”Approximate multi-agent planning in dynamic and uncertain environments.” PhD diss., Massachusetts Institute of Technology, 2011.
[7] Chakraborty, Doran, and Peter Stone. ”Cooperating with a markovian ad hoc teammate.” In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, pp. 1085-1092. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
[8] Hausknecht, Matthew, Prannoy Mupparaju, Sandeep Subramanian, Shivaram Kalyanakrishnan, and Peter Stone. ”Half field offense: An environment for multiagent learning and ad hoc teamwork.” In AAMAS Adaptive Learning Agents (ALA) workhop. 2016.
[9] Genter, Katie, Tim Laue, and Peter Stone. ”Three years of the RoboCup standard platform league drop-in player competition.” Autonomous Agents and Multi-Agent Systems 31, no. 4 (2017): 790-820.
[10] Bethke, Brett, Jonathan How, and John Vian. ”Multi-UAV Persistent Surveillance with Communication Constraints and Health Mangement.” In AIAA Guidance, Navigation, and Control Conference, p. 5654. 2009.
[11] Nisan, Noam, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, eds. Algorithmic game theory. Cambridge university press, 2007.
[12] Vidal, J. M. ”Fundamentals of multiagent systems with Net-Logo Examples: March 2009.” (2010)..
[13] ODriscoll, Dylan. ”UAVs in Humanitarian Relief and Wider Development Contexts.” (2017).