Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 183401, 10 pages
http://dx.doi.org/10.1155/2014/183401

Research Article

A Case Study on Air Combat Decision Using Approximated Dynamic Programming

Yaofei Ma, Xiaole Ma, and Xiao Song

School of Automation Science and Electrical Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China

Correspondence should be addressed to Yaofei Ma; mayaofeibuaa@163.com and Xiao Song; songxiao@buaa.edu.cn

Received 6 June 2014; Accepted 21 August 2014; Published 25 September 2014

Academic Editor: Minrui Fei

As a continuous state space problem, air combat is difficult to be resolved by traditional dynamic programming (DP) with discretized state space. The approximated dynamic programming (ADP) approach is studied in this paper to build a high performance decision model for air combat in 1 versus 1 scenario, in which the iterative process for policy improvement is replaced by mass sampling from history trajectories and utility function approximating, leading to high efficiency on policy improvement eventually. A continuous reward function is also constructed to better guide the plane to find its way to "winner" state from any initial situation. According to our experiments, the plane is more offensive when following policy derived from ADP approach other than the baseline Min-Max policy, in which the "time to win" is reduced greatly but the cumulated probability of being killed by enemy is higher. The reason is analyzed in this paper.

1. Introduction

Unmanned aerial vehicle (UAV) plays an important role in modern battlefield. For the past decades, UAV has made significant advancement on both hardware and software and achieved mission capabilities including "simple" ones like intelligence, surveillance, and reconnaissance (ISR), and "complex" ones like electronic attack, ground targets strike, suppression or destruction of enemy air defense (SEAD/DEAD), and others. According to the development roadmap [1] proposed by UAV technical leading countries, even the mission of air combat, which has been believed to be the dominated domain of human pilots due to the dynamic and complexity on tactical decisions, is possible to be carried out by autonomy UAV in the near future.

However, the decision technologies supporting automatic air combat are far from maturity. They are still on the way for better robustness, intelligence, autonomy, team cooperation, and adaption to complex environment. The methods applied in this domain include game theory [2–4], knowledge-based decision [5–9], graphic based methods like influence diagram [10], and others. Dynamic programming (DP) [11] is one of the most powerful methods for its adaptation to dynamic environment and the capability to improve policy constantly by learning [12]. However, traditional DP approach is not suitable to resolve continuous state space problem like air combat, in which the computation complex becomes intractable because of the curse of dimensionality.

In this paper, a tactical decision framework employing approximated dynamic programming (ADP) method [13–15] is proposed for air combat mission. The trait of ADP method is that the utility function is learned from mass sampled states in problem space rather than from scratch, which lead to high efficiency in policy converging. As the result, ADP can be used as a second stage tool to improve the policy derived from other decision systems (denoted by the "first stage tool" for decision here), for example, a knowledge-based system. If we treat the combat traces produced in the first stage tool as the sampled states, then ADP algorithms can learn its utility function and policy from these states directly. Considering the optimizing capability of ADP inherited from DP approach, the learned policy can be improved constantly to achieve better decision performance. Thus, the merits of different decision system are combined together.

The content of this paper would be arranged as follows. In Section 2, the 1 versus 1 air combat problem is formulated

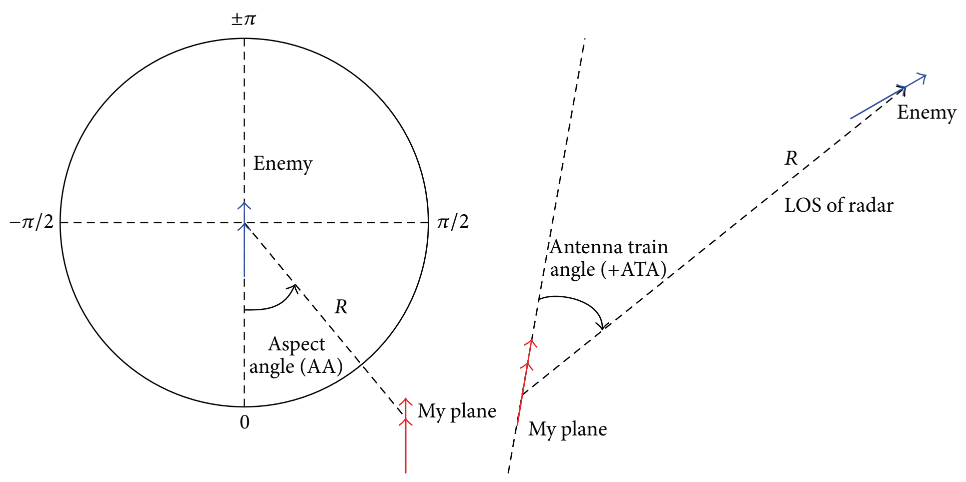

FIGURE 1: The geometrical measures of the firing position.

with DP formation. In Section 3, the ADP method is briefly reviewed. Section 4 discusses the reward function for air combat, which is designed to guide the UAVs to enter into goal states smoothly. Some features are also specified to gain an insight of engagement situation. In Section 5, the key algorithms of ADP decision framework are proposed. The followed comparative experiments (Section 6) validate the effectiveness of the proposed framework.

2. Problem Formulation

A 1 versus 1 air combat scenario involves two opponent planes (denoted by red and blue, where the red is supposed to be "my" side). Omitting vertical movement, the kinematic equations of the plane are

$$\begin{aligned}\dot{\psi} &= \frac{f_n}{v} \\ \dot{x} &= v \cdot \cos(\psi) \\ \dot{y} &= v \cdot \sin(\psi),\end{aligned} \tag{1}$$

where v is the scalar value of velocity, which is assumed to be const during the combat. $\psi \in [-\pi, \pi]$ is yaw angle and is defined as the deviation of velocity from north (the y axis). ψ is controlled by f_n. f_n is plane's normal overload, which always points right from the gravity center of the plane and is orthogonal to velocity. In our control schema, f_n can take a value from three options once a time: $\{-3, 0, 3\}$. The plane will turn counterclockwise, turn clockwise, and keep current velocity direction, respectively, with these values.

The goal of the planes is to occupy advantage position by tactical decision and gain the fire opportunities at its rival. The state space of air combat can be described with vector

$$x = \{x_r, y_r, \psi_r, x_b, y_b, \psi_b\}, \tag{2}$$

where subscript r and b refer to red and blue, respectively. Any state s is an instance of x. With (1), the state transition in combat space can be represented as a function

$$s' = f\left(s,\ a_r = f_n^1,\ a_b = f_n^2\right) \tag{3}$$

which means the current state s will transfer into a new state s' after performing a_r and a_b.

The goal state is reached when one plane gains opportunities to fire at its opponent. The firing position is defined by three geometrical measures:

(a) $|\text{Aspect angle}| < \pi/3$. Aspect angle (AA) is a relative angle between the longitudinal symmetry axis (to the tail direction) of the target plane and the connecting line from target plane's tail to attacking plane's nose. $|\text{AA}| < \pi/3$ refers to area where the killing probability is high when attacking from rear considering most close-combat air missiles are infrared guidance;

(b) $|\text{Antenna train angle}| < \pi/6$. Antenna train angle (ATA) is the angle between attacking plane's longitudinal symmetry axis and its radar's line of sight (LOS), as Figure 1 shows. This criterion defines an area from which the target plane is difficult to escape with radar locking.

(c) Relative range (R) between two planes: this criterion makes sure that the target plane is within the attacking range of air-to-air weapon.

3. ADP Method Review

DP defines adaptive learning process and its mathematic model is Markov decision process (MDP). In DP formulation, the air combat can be described as a discrete time decision problem with five-tuples: $\{S, A, P, R, U\}$:

(1) $S = \{s\}$ is the problem space defined with state variable x; s is the instance of x;

(2) $A(s)$ is the finite action set available in state s, from which the plane selects one to execute at each decision

interval. In our problem, $A(s)$ is same for each state s and thus can be simply denoted as A;

(3) $P(s' \mid s, a)$ is the probability of transition from state s to s';

(4) $R(s)$ is the reward of state s. If s is visited multiple times during the combat, the rewards are discounted cumulated to form utility value of that state;

(5) $U(s)$ is the utility of state s. Its value is the cumulated rewards of multiple visiting. If every state is visited adequate times, the utility distribution will converge to the optimal one, by which the optimal policy is derived.

The decision process starts from an initial state s_0 and then selects action to perform. The action interacts the environment and leads to a new state, and so on. Then, the utility of the starting state is the expectation of discounted cumulated rewards on all states following the start one:

$$U(s) = E\left\{R(s_0) + \gamma \cdot R(s_1) + \gamma^2 \cdot R(s_2) + \cdots \mid s = s_0, \pi(s)\right\}, \quad (4)$$

where $\gamma \in (0,1)$ is the discounted coefficient, making sure $U(s)$ converges eventually. Policy $\pi(s) \rightarrow a$ is a mapping from state space to action space. For a fixed policy π, the utility satisfies Bellman equation

$$U(s) = \sum_{s'} P(s, \pi(s), s') \cdot \left[R(s' \mid s, \pi(s)) + \gamma \cdot U(s')\right]. \quad (5)$$

The optimal utility U^* is the value function that simultaneously maximizes the expected cumulative reward in all states $s \in S$. Bellman proved that U^* is the unique solution of (5):

$$U^*(s) = \max_a \left\{\sum_{s'} P(s, a, s') \cdot \left[R(s' \mid s, a) + \gamma \cdot U^*(s')\right]\right\}. \quad (6)$$

Actually, U^* can be obtained through iterations on Bellman equation:

$$U^{k+1}(s) = TU^k(s) = \max_a \left\{\sum_{s'} P(s, a, s') \cdot \left[R(s' \mid s, a) + \gamma \cdot U^k(s')\right]\right\}. \quad (7)$$

T is denoted by Bellman operator, representing the iterative improvement on U by traversing states throughout the space eventually. During this process, $P(s, a, s')$ would also converge to its "true" distribution. Then, the optimal policy can be derived:

$$\pi^*(s) = \operatorname*{argmax}_a \left\{\sum_{s'} P(s, a, s') \cdot \left[R(s' \mid s, a) + \gamma \cdot U^*(s')\right]\right\}. \quad (8)$$

As we can see from (4)–(8), traditional DP method needs to traverse discrete states iteratively, resulting in tabular utility function. This approach is not suitable to resolve continuous state space problem. Discretization on state space leads to two defects: (i) the unreasonable assuming that utility function is const in each discrete state cell and (ii) the curse of dimensionality.

ADP method mitigates these problems with two operations: (i) sampling mass states effectively from problem space, thus reducing the consumed time on space exploration; (ii) approximating state utility $\widetilde{U}$ using sampled states, with which the near-optimal policy, rather than the optimal one, is employed to determine actions. Denoting the sampled states as a set $S = \{s_1, s_2, \ldots, s_m\}$, we have

$$\widetilde{U}^{k+1} = T\widehat{U}^k, \quad (9)$$

where $\widehat{U}^k$ is the current approximation of utility; $\widetilde{U}^{k+1}$ is one Bellman iteration from $\widehat{U}^k$. Then, $\widehat{U}^{k+1}$ can be approximated based on $\widetilde{U}^{k+1}$. There are multiple options for approximation operation [16, 17]; the least squares approximation is used here:

$$\begin{aligned} \beta^{k+1} &= (S^T S)^{-1} \cdot \widetilde{U}^{k+1}, \\ \widehat{U}^{k+1} &= S \cdot \beta^{k+1}, \end{aligned} \quad (10)$$

where β is approximation coefficients vector and S is sampled states set. Normally, a set of features need to be defined to gain an insight on characteristics of the studied problem. The approximated utility function will converge more quickly and be more precise since these features come from pilots' combat experiences in real world. We have

$$\begin{aligned} \Phi(S) &= \{\phi_1(S), \ldots, \phi_c(S)\}, \\ \beta^{k+1} &= (\Phi^T \Phi)^{-1} \cdot \widetilde{U}^{k+1}, \\ \widehat{U}^{k+1} &= \Phi \cdot \beta^{k+1}. \end{aligned} \quad (11)$$

Φ is feature vector.

As a conclusion, the steps of ADP method can be briefly listed as follows:

(a) to sample states set S in problem space;

(b) to get the one iteration improvement utility $\widetilde{U}^{k+1}$ from current utility $\widehat{U}^k$; the initial value of $\widehat{U}^k$ can be set as the reward of initial states; that is, $\widehat{U}^0 = R(S_0)$, where S_0 is initial state set;

(c) to update the next value of approximated utility $\widehat{U}^{k+1}$ following (10) and (11);

(d) if the policy still needs to be improved, go back to (a).

```
ADP_Learn()
Input variables:
(1) $S$: sampled states set;
(2) $N$: the number of learning round.
(3) $\pi^{\text{mm}}(S)$: blue plane's policy derived from Min-Max approach.
Output variables:
(1) $\widehat{U}^k$: utility function approximated.
Local variables:
(2) $A_b$: action vector of blue plane derived from Min-Max policy;
(3) $A_r$: action vector of red plane derived from current utility;
(4) $\widetilde{U}^k$: one step improved utility function by Bellman iteration;
(5) $\Phi$: features vector used to compute approximation coefficients;
(6) $\beta^k$: vector of approximation coefficients.
Code:
(1) $\widehat{U}^{k=0} = R(S)$;
(2) FOR $k = 1 : N$, DO:
(3)     $A_b = \pi^{\text{mm}}(S)$;
(4)     $\{A_r, \widetilde{U}^k\} =$
(5)             $\arg\max_A \{\lambda \cdot \widehat{U}^{k-1}(f(S, A, A_b)) + R(f(S, A, A_b))\}$;
(6)     $\Phi(S) = \{\phi_1(S), \phi_2(S), \ldots\}$
(7)     $\beta^k = (\Phi^T\Phi)^{-1}\Phi^T \cdot \widetilde{U}^k$
(8)     $\widehat{U}^k = \Phi \cdot \beta^k$
(9) END
(10) RETURN $\widehat{U}^{k=N}$;
```

Algorithm 1: Utility function approximating based on sampled states.

```
Rollout_ADP_Policy()
Input variables:
(1) $s$: the current state;
(2) $N_{\text{roll}}$: the number of rollout steps;
(3) $\widehat{U}^{k=N}$: utility function approximated in ADP_Learn algorithm;
(4) $\pi^{\text{appx}}(s)$: red plane's policy derived from $\widehat{U}^{k=N}$, see (8).
Output variables:
(1) $a_{\text{best}}$: the best action respect to current state $s$.
Local variables:
(1) $U_{\text{best}}$: to cache the maximum utility responding to different actions;
(2) $s'$: the cache the next state computed by system equation.
Code:
(1) $a_{\text{best}} = NULL$;
(2) $U_{\text{best}} = NULL$;
(3) FOR $a_r = \{-3, 0, 3\}$, DO:
(4)     $s' = f(s, a_r, \pi^{\text{mm}}(s))$;
(5)     FOR $i = 1 : N_{\text{roll}}$, DO:
(6)         $s' \leftarrow f(s', \pi^{\text{appx}}(s'), \pi^{\text{mm}}(s'))$;
(7)     END
(8)     IF $\{\gamma \cdot \widehat{U}^{k=N}(s') + R(s')\} > U_{\text{best}}$, THEN:
(9)         $U_{\text{best}} = \gamma \cdot \widehat{U}^{k=N}(s') + R(s')$;
(10)        $a_{\text{best}} = a_r$;
(11)    END
(12) END
(13) RETURN $a_{\text{best}}$;
```

Algorithm 2: Rollout decision procedure using approximated utility function.

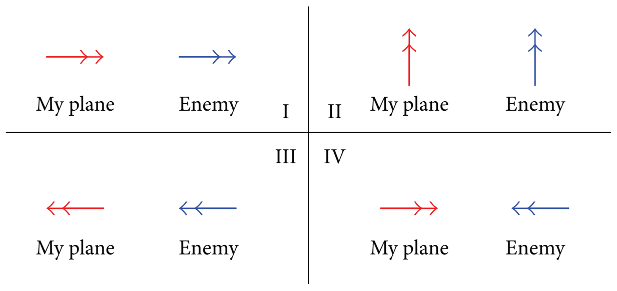

Figure 2: Four basic initial situations in air combat.

4. Reward Function and Combat Features

Before giving ADP algorithm, the reward function $R(s)$ needs to be discussed firstly since it is a necessary part in ADP steps. As (4) shows, the utility is actually the discounted cumulated $R(s)$ along state trace. Thus, a properly defined $R(s)$ can better guide plane approach to goal state from any starting state.

The computation of $R(s)$ is domain related. As for our scenario, the attacking plane in its goal state gets reward +1, and the target plane in the same state gets −1 as punishment. The reward in other states is 0. Intuitively, with these discrete rewards, the planes will spend more time on space exploration to find trace from starting state to goal state. To better guide the plane, a reward function is defined as

$$R' = \left[\frac{(1-|\mathrm{AA}|/\pi)+(1-|\mathrm{ATA}|/\pi)}{2}\right]e^{-((|R|-R_D)/(\pi\cdot k))}, \quad (12)$$

where R_D is the expected attacking range of weapons, R is the relative distance between planes, and k is an coefficient adjusting the influence of R in total reward.

With (12), the plane occupying firing position ($\mathrm{AA}=0$, $\mathrm{ATA}=0$, and $R=R_D$) gets reward $R'=1$; the plane under attack ($\mathrm{AA}=\pi$, $\mathrm{ATA}=\pi$, and $R=R_D$) gets reward $R'=0$. In other states, the reward will increase continuously and monotonically from the worst state to the most advantage state. To emphasize the punishment in bad state (the punishment will guide planes to avoid these states), a simple linear transformation is applied to R' to get the final reward function:

$$R(s) = 2\cdot R' - 1. \quad (13)$$

To construct the utility function, some geometric features [18] are specially defined to describe combat situation, as Table 1 shows. These feature are optional for utility approximation in ADP steps because we can use sampled states instead, as (10) shows. However, they are more straightforward to capture the “true” utility of states, and that is why human pilots also use them to judge their situations in real-world combat. In other words, these well-defined features are more representative to approximate utility function.

5. Method

In the combat scenario, the red plane is marked as “my” side, and the blue one is marked as enemy. To describe

Table 1: Features evaluating combat situation.

Features (Φ)	Description
R	Relative distance between red and blue planes.
AA	Aspect angle. Reference is Figure 1.
$\dot{\mathrm{AA}}$	The changing ratio of AA.
$\lvert\mathrm{AA}\rvert$	Absolute value of AA.
ATA	Antenna Train angle.
$\lvert\mathrm{ATA}\rvert$	Absolute value of ATA.
$\dot{\mathrm{ATA}}$	The changing ratio of ATA.
HCA	The error on yaw angle of both planes.
$\lvert HCA\rvert$	Absolute value of HCA.

ADP approach, a reference decision algorithm, the Min-Max search algorithm [19] is employed here. The Min-Max algorithm looks into future for n steps, using domain knowledge to determine the acting consequent before giving final decision.

ADP approach involves two algorithms: (i) the learning algorithm (*ADP_Learn()*), in which the utility function is approximated, and (ii) the decision algorithm (*Rollout_ADP_Policy()*), in which the final action is determined based on ADP policy derived from learned utility. The *ADP_Learn()* algorithm is displayed in Algorithm 1.

In *ADP_Learn()*, the utility function is approximated with sampled states, which is expected to be sampled from frequently visited space, to fully capture the changes on utility values in these areas. An option is to use the trajectories produced in real-world combat or other authoritative decision tools for air combat, since the trajectories themselves indicate the high probability of being visited in combat. In this paper, a scenario is built to get combat trajectories, where two rival planes all take Min-Max policies, and their trajectories are recorded as S_0.

The initial value of $\widehat{U}^k$ is assigned as the reward of S_0 (line 1 in **Code** section) and then is improved N rounds. In each round, firstly, the blue plane's action A_b is determined by Min-Max policy (line 3). Secondly, the red plane's action A_r is selected by applying one step Bellman operator. The changed utility $\widetilde{U}^k$ is also recorded (lines 4-5) for further use. Thirdly, the feature vector is updated (line 6), with which the least squares approximation is performed to approximate $\widehat{U}^k$ according to $\widetilde{U}^k$ (lines 7-8).

The approximated utility function $\widehat{U}^{k=N}$ returned by *ADP_Learn()* already can be used to give decisions, as (8) shows. However, a rollout procedure is employed here to further improve the quality of final decisions, as Algorithm 2 shows.

Assuming the red plane is making decision in *Rollout_ADP_Policy()*. It will not follow ADP policy directly. On the contrary, it tries each possible action (line 1 in Code section). For each possible action, the red plane's future state

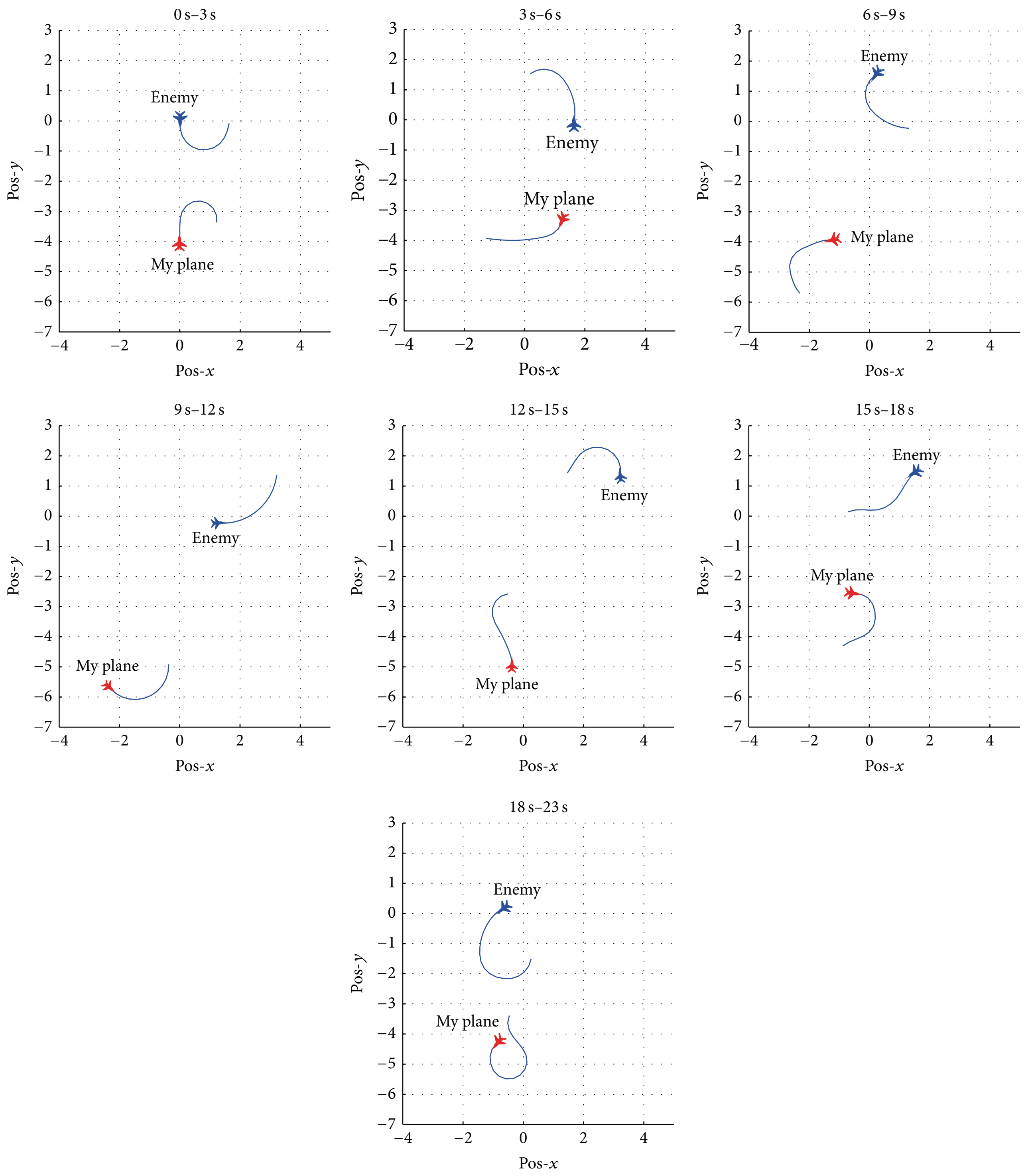

Figure 3: Baseline experiment: combat start from *Setup 4*. Both red and blue follow Min-Max policies. The plane icons appear at the start position in each subfigure; the x, y positions (denoted as Pos-x and Pos-y) are scaled down for better representation. The red plane wins at last and TTW = 23 s.

(s') is rolled out for N_{roll} steps (lines 5–7). The red plane follows ADP policy during this process and the blue one follows Min-Max policy. The sum of reward and utility of s' is compared with the historical best value U_{best}: if the former is bigger than U_{best}, then update U_{best} and record the corresponding best action a_{best} (lines 8–11). Having tried all possible actions, the red plane gets the best action a_{best} in state s.

Table 2: Four initial setups of air combat.

Initial situation	Description	x_b	y_b	ψ_b	x_r	y_r	ψ_r
Setup 1	Offensive	0	−2.5	0	0	0	0
Setup 2	Neutral	2.75	0	$-\pi/10$	0	0	0
Setup 3	Defensive	0	0	0	0	−4	0
Setup 4	Confronting	0	−4	0	0	0	π

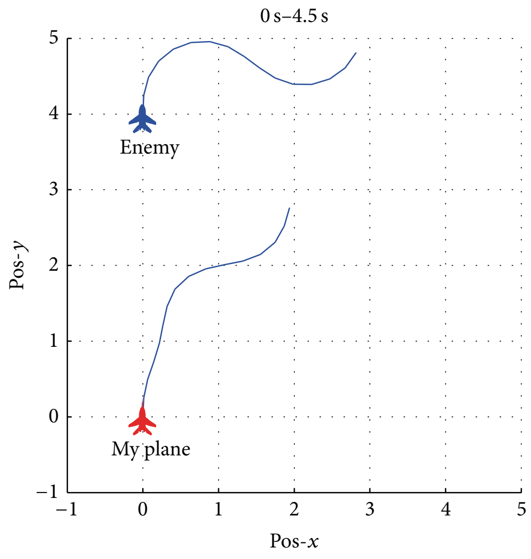

Figure 4: Engagement process starts from *Setup 1* with red plane following ADP policy $\pi_r^{k=40}$ and blue plane following π_b^{mm}. TTW = 4.5 s.

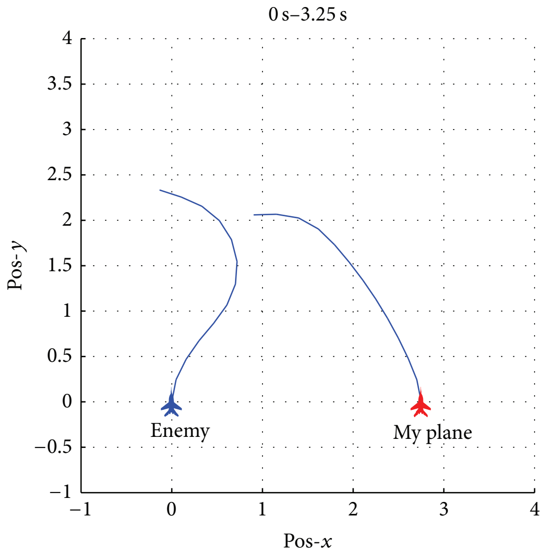

Figure 5: Engagement process starts from *Setup 2* with red plane following ADP policy $\pi_r^{k=40}$ and blue plane following π_b^{mm}. TTW = 3.25 s.

6. Simulation and Analysis

The initial state of 1 versus 1 air combat can be classified into 4 basic situations (from red plane's perspective): offensive, neutral, defensive, and confronting, as Figure 2 shows.

In our experiments, all four initial situations are configured (Table 2) to compare the performance of ADP policy and Min-Max policy.

The experiments are arranged as follows. Firstly, a baseline experiment is conducted in which both red and blue plane would take Min-Max policy (denoted as π^{mm}). The decision performance of π^{mm} is treated as the baseline to compare ADP policy.

Secondly, the learned ADP policy is applied with same initial situations. ADP policy is denoted as $\pi^{k=N}$, where N is the learning rounds. For example, $\pi^{k=40}$ means this policy is approximated after 40 rounds. The decision performance is measured with 2 metrics: (a) the average time to win (TTW); the winning states have been defined in Section 2; the attacking plane needs to hold that state for at least 10 seconds to win; (b) the accumulated probability of being killed (APK); this indicates the total risks of one plane during the combat, in which the probability of being killed by enemy would be cumulated. A good policy would result in both small TTW and APK.

To speed up the experiment, a bigger f_n is assigned to red plane which means it can change direction more quickly. This measurement would avoid long time standoff when both planes follow the same policy. This performance advantage will not influence policy comparison since both set experiments use the same configured planes.

The baseline experiments are conducted firstly. Only the result of *Setup 4* (confront) is displayed here considering the paper space limitation, as Figure 3 shows.

Figures 4, 5, 6, and 7 show the result of each initial setup where red plane follows ADP policy and blue plane follows Min-Max policy. Comparing Figures 7 and 3, we can see the performance of red plane is improved greatly by taking ADP policy, in which the TTW is reduced from 23 s to 10.5 s.

The comparison on decision performance is displayed in Table 3. As we can see, the TTW of $\pi_r^{k=40}$ is reduced in all setups compared to π_r^{mm}, especially in *Setup 3*. This means $\pi_r^{k=40}$ can guide the red plane to get rid of the chasing quickly and find its way to occupy the firing position. On the other hand, the APK is slightly higher with ADP policy.

These results show that a plane is more offensive when following ADP policy. The plane is likely to occupy firing position risk at the risk of being killed. This phenomenon can be explained from the working mechanism of two decision

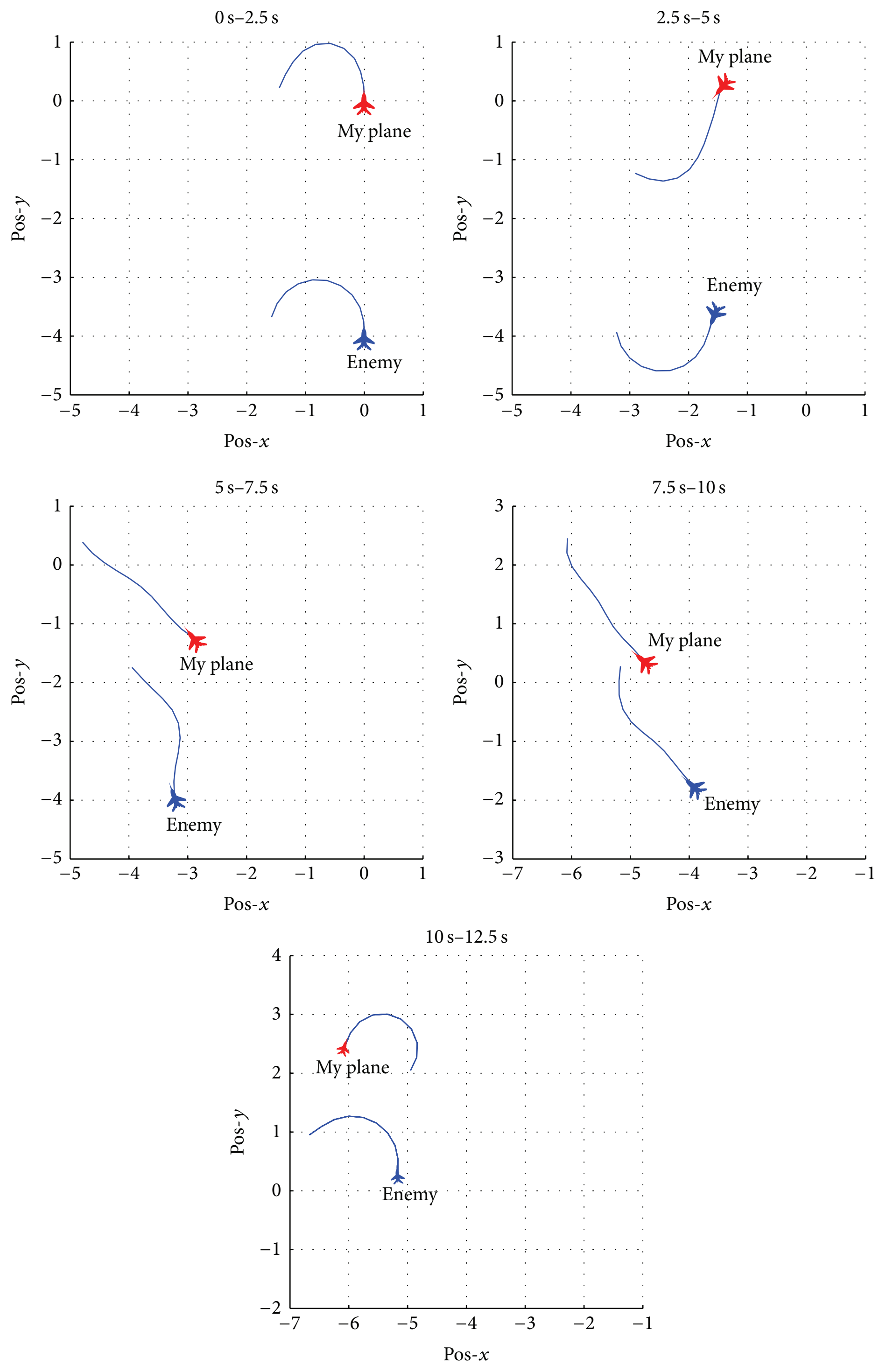

Figure 6: Engagement process starts from *Setup 3* with red plane following $\pi_r^{k=40}$ and blue plane following π_b^{mm}. TTW = 12.5 s.

approaches. In Min-Max algorithm, the decision is finally made considering each possible reaction from the opponent. This leads to a conservative style in decision making. By contrast, the ADP approach uses utility to guide the plane to make best profit by acting properly and avoid punishment at the same time. This somehow makes the plane abandon conservative choice for high reward in the future. This result also proved that ADP approach is effective and of high performance to resolve air combat problem

7. Conclusion

This paper studied the 1 versus 1 air combat decision problem and employed ADP approach to resolve it quickly and

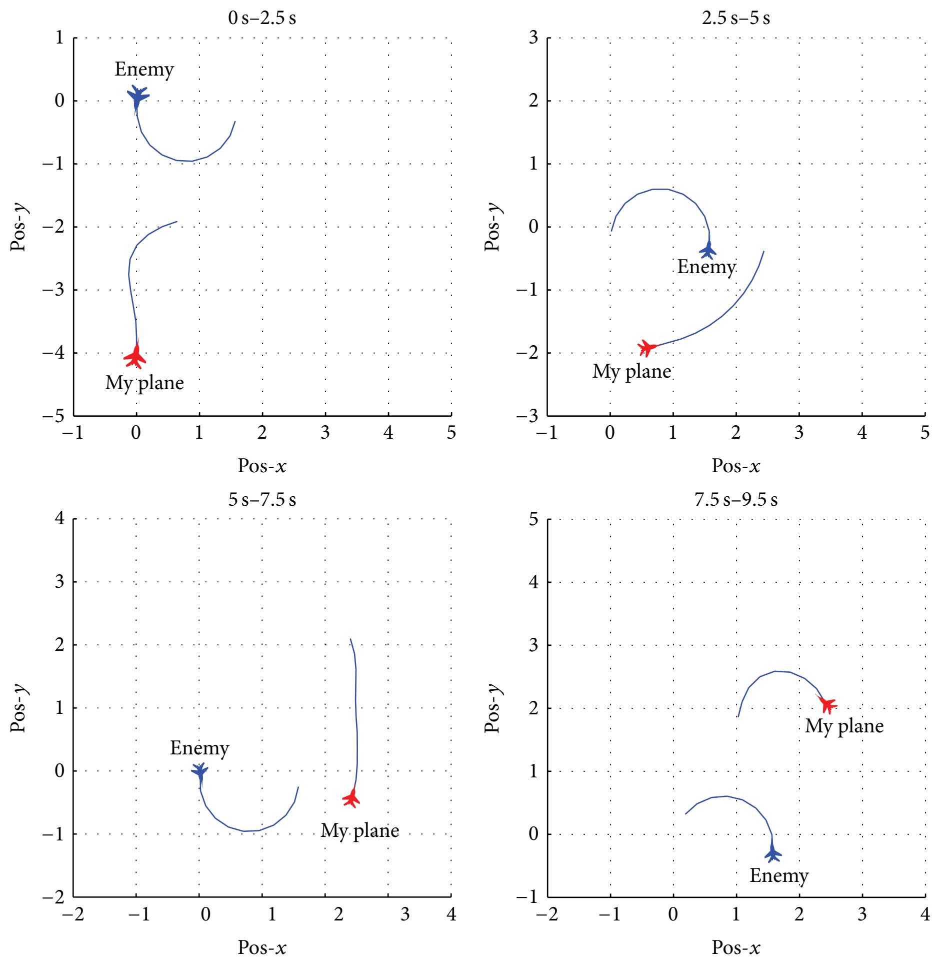

FIGURE 7: An engagement starts from *Setup 4* with red plane following ADP policy $\pi_r^{k=40}$ and blue plane following π_b^{mm}. TTW = 9.5 s.

TABLE 3: Comparison on performance of different policies.

Initial setup	TTW (s)				APK			
	1	2	3	4	1	2	3	4
π_r^{mm}	134	39	72	23	0.108	0	0	0
$\pi_r^{k=40}$	4.5	3.25	12.5	9.5	0.139	0	0.0179	0.047

effectively. The ADP approach involves two operations: (i) learning utility function from mass sampled states rather than from scratch; (ii) making final decision by evaluating the future incoming of any possible action. Comparative experiments show that the policy initially produced by Min-Max algorithm is improved greatly after ADP process.

In the future work, we plan to build a hybrid-decision framework for UAV in which the decision functionality is divided into two parts. The first part is responsible for making initial acting policy for specific task, which is a knowledge-based decision module, and can handle large scale, complex task environment. The second part is the ADP module proposed in this paper. This ADP module gets state samples from the first part, with which the utility function is approximated and the acting policy is improved. This process combines the advantages of different decision frameworks together.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research is supported by the Armament Research Foundation (9140A04010112HK01041).

References

[1] Defense Technical Information Center, “Unmanned aircraft systems roadmap 2013–2038 [R/OL],” 2013, https://info.publicintelligence.net/DoD-UnmannedRoadmap-2013.pdf.

[2] R. Isaacs, “Games of pursuit,” 1951.

[3] L. A. Petrosjan, *Differential Games of Pursuit*, vol. 2, World Scientific, 1993.

[4] F. Austin, G. Carbone, M. Falco, and H. Hinz, "Automated maneuvering decisions for air-to-air combat," AIAA Paper 87-2393, 1987.

[5] H. Burgin George and L. B. Sidor, "Rule-based air combat simulation," Tech. Rep. TITAN-TLJ-H-1501, Titan Systems Inc, La Jolla, Calif, USA, 1988.

[6] S. B. Banks and C. S. Lizza, "Pilot's Associate: a cooperative, knowledge-based system application," *IEEE Expert*, vol. 6, no. 3, pp. 18–29, 1991.

[7] K. Schwamb, F. Vincent, and K. D. Keirsey, "Working with ModSAF: interfaces for programs and users," in *Proceedings of the 4th Conference on Computer Generated Forces and Behavioral Representation*, 1994.

[8] B. McEnany, "CCTT SAF functional analysis," in *Proceedings of the 4th Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, 1994.

[9] A. J. Courtemanche and R. L. Wittman Jr., "OneSAF: a product line approach for a next-generation CGF," in *Proceedings of the 11th Computer Generated Forces Conference*, IEEE Computer Society Press, Orlando, Fla, USA, 2002.

[10] K. Virtanen, J. Karelahti, and T. Raivio, "Modeling air combat by a moving horizon influence diagram game," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 5, pp. 1080–1091, 2006.

[11] R. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, pp. 716–719, 1952.

[12] A. Coates, P. Abbeel, and A. Y. Ng, "Apprenticeship learning for helicopter control," *Communications of the ACM*, vol. 52, no. 7, pp. 97–105, 2009.

[13] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1107–1149, 2004.

[14] M. Maggioni and S. Mahadevan, "Fast direct policy evaluation using multiscale analysis of Markov diffusion processes," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 601–608, ACM, June 2006.

[15] M. Geist and P. Olivier, "A brief survey of parametric value function approximation," Rapport interne, Supélec, 2010.

[16] B. Li and J. Si, "Approximate robust policy iteration using multilayer perceptron neural networks for discounted infinite-horizon markov decision processes with uncertain correlated transition matrices," *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1270–1280, 2010.

[17] Y. Ma, G. Gong, and X. Peng, "Cognition behavior model for air combat based on reinforcement learning," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 36, no. 4, pp. 379–383, 2010.

[18] S. Baron, D. L. Kelinman, and S. Serben, "A study of the Markov game approach to tactical maneuvering problems," NASA CR-1979, NASA, 1972.

[19] F. Austin, G. Carbone, M. Falco, H. Hinz, and M. Lewis, "Game theory for automated maneuvering during air-to-air combat," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 6, pp. 1143–1149, 1990.