2015 IEEE 54th Annual Conference on Decision and Control (CDC)
December 15-18, 2015. Osaka, Japan

Continous Time Autonomous Air Traffic Control for Non-Towered Airports

Zouhair Mahboubi and Mykel J. Kochenderfer

Abstract— **The majority of reported near mid-air collisions that involve a general aviation aircraft (GA) occur in the vicinity of non-towered airports. In this paper, we propose a concept for air traffic collision prevention for GA aircraft, which can be modeled as a continuous-time Markov decision process (CTMDP). The problem is equivalent to controlling independent homogeneous multi-agents under communication or resource constraints. We outline a methodology that leverages the structure of the problem to efficiently solve it despite the large state space. Finally, we present simulation results for aircraft operating under the optimized policies in the traffic pattern.**

I. INTRODUCTION

The Aircraft Owners and Pilots Association (AOPA) and the Federal Aviation Administration (FAA) produce a yearly aviation safety report. In 2000, they individually highlighted that most mid-air collisions involving general aviation (GA) aircraft occur in the vicinity of airports [1], [2]. The majority of these collisions occur in the traffic pattern of non-towered airports. Similar trends have been observed in Australia, Canada, and France [3].

Large aircraft benefit from the use of the Traffic Alert and Collision Avoidance System (TCAS), but it is prohibitively expensive. Additionally, the system was not designed for use by smaller aircraft [4]. Recent work has presented modifications to TCAS for use in GA aircraft [5], [6], but the research has remained focused on on-board systems [7]. Solutions that require new equipment are unlikely to be adopted by the GA community [8]. Recent research recognizes the need for a low cost solution. A ground-based surveillance concept for small airports was investigated by Campbell and demonstrated promising performance [9]. The goal is to deploy the sensor at towered airports and high density non-towered airports.

This paper proposes a concept for air traffic collision prevention in the vicinity of airports with a focus on GA aircraft. Drawing inspiration from prior work of others [10], we envision an autonomous ATC system (auto-ATC) at a non-towered airport that is advisory in nature. It would use observations from a ground-based surveillance system and issue alerts over the common traffic advisory frequency. The pilot would still be responsible for maintaining separation, but the system would be able to provide advice to participating aircraft by issuing high-level commands.

In previous work [11], we showed that the auto-ATC can be modeled using the decision making under uncertainty framework as a Markov decision process (MDP). However, when we evaluated the policies in a 3D simulation, they did not perform as well as we expected. We hypothesized that the poor performance could be explained by the transitions not occurring synchronously as modeled by the MDP. In this paper, we tackle this shortcoming by reformulating the problem as a continuous-time Markov decision process (CTMDP). Furthermore, we propose a methodology which leverages the problem structure to efficiently solve it despite the large state space. This methodology is general and can be applied to problems involving the control of independent homogeneous multi-agents under communication or resource constraints

The outline of the paper is as follows: Section II introduces the notation for CTMDPs and formulates the collision prevention in the traffic problem as one, Section III shows how we can take advantage of the problem's structure to achieve computational efficiency, Section IV presents simulation results, and Section V concludes with a summary and suggestions for future work.

II. PROBLEM FORMULATION

MDPs model sequential problems where decisions need to be made under uncertainty. They have been successfully used in the context of aircraft collision avoidance [6], [12]. Our initial formulation of the auto-ATC used an MDP formulation [11]. However, when evaluated in a 3D simulation, we realized that transitions do not all occur synchronously as modeled by the MDP, and the actions need to be taken at potentially non-uniform time steps. To address this, we use continuous-time Bayesian networks (CTBN) to represent the dynamics [13], and pose the problem as a CTMDP [14]. In this section, we explain how the auto-ATC system can be formulated as a CTMDP.

A. Continuous-Time MDP

CTBNs are a probabilistic framework used to model continuous time dynamics of structured stochastic processes. When the process dynamics can be influenced through actions, the resulting model is a CTMDP, which is defined by a state space $\mathcal{S}$, action space $\mathcal{A}$, intensity matrices Q_a (one for each action), a lump sum reward r and a reward rate function ρ [14].

State Space: The state variables are tuples $s = (s^{(1)}, \ldots, s^{(K)})$, where $s^{(i)} \in \{\ell_1, \ldots, \ell_n\}$ is the location of the ith aircraft in the traffic pattern.

In this formulation, there are a set of $n = 33$ possible locations, e.g., Taxi ($\ell_1 =$ T) and Runway ($\ell_2 =$ R). We

Z. Mahboubi and M.J. Kochenderfer are with the Aeronautics and Astronautics department, Stanford University, Stanford, CA 94305, USA.

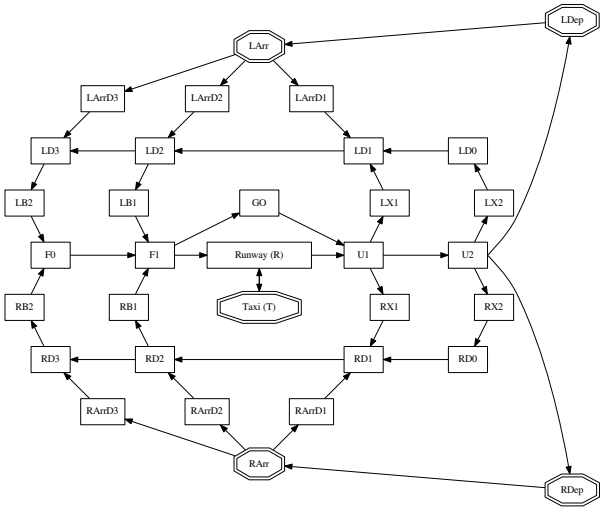


Fig. 1: Aircraft states in the pattern.

use numerical aliases for the locations to represent the state as a vector $s \in \mathbb{Z}^K$ when it is convenient, e.g., $\ell_1 = \mathrm{T} = 1$. The size of the state space is $|\mathcal{S}| = n^K$ and the set of neighboring locations of ℓ_i is denoted $\mathcal{N}(\ell_i)$. Figure 1 shows a representation of the various locations and their neighbors in the pattern.

Action Space: An action $a = (a_1, a_2) \in \mathcal{A}$ involves commanding aircraft $a_1 \in \{1, \ldots, K\}$ to transition from its location $s^{(a_1)}$ to its $a_2 \in \{1, \ldots, |\mathcal{N}(s^{(a_1)})|\}$ neighbor. If no aircraft is to be addressed, we use $a = (0, 0)$. The valid set of actions from s depends on the state and is denoted $A(s)$.

Rewards: Executing action a in state s results in a lump sum reward $r(s, a)$, and a reward accumulated during the time spent in state s at a rate $\rho(s, a)$. Future rewards are discounted exponentially in time with rate ζ, i.e., unit reward at t time in the future is worth $e^{-\zeta t}$.

We make the following assumptions:

- Only lump sum rewards, i.e., $\rho(s, a) = 0$.
- Additive rewards, i.e., $r(s, a) = r(s) + r(a)$.
- Two aircraft occupying the same state result in a cost C_c, unless they are in one of the states considered safe.
- There is a cost C_a for issuing an advisory.

The reward function can be expressed by defining $\tilde{s} = (s^{(i)} \in s \mid s^{(i)} \notin \{\mathrm{T}, \mathrm{LDep}, \mathrm{RDep}, \mathrm{LArr}, \mathrm{Rarr}\})$ as:

$$r(s, a) = -C_c(|\tilde{s}| - |\text{unique}(\tilde{s})|) - \begin{cases} 0 & \text{if } a = (0, 0) \\ C_a & \text{otherwise} \end{cases}, \tag{1}$$

where we use the list comprehension notation $(x \in X \mid F(x))$ to mean the list of all elements in X that satisfy the logical expression $F(x)$. The notation $|y|$ denotes the number of elements in the list y.

Intensity Matrices: In the MDP formulation, the function $T(s' \mid s, a)$ provides the probability of transitioning to state s' from state s given that action a is taken.

If we make the assumption that transitions are independent across aircraft we can write:

$$T(s' \mid s, a) = \prod_{i=1}^{K} T\left(s'^{(i)} \mid s^{(i)}, \begin{cases} a_2 & \text{if } a_1 = i \\ 0 & \text{otherwise} \end{cases}\right), \tag{2}$$

where $T(s'^{(i)} \mid s^{(i)}, a)$ is the transition function for a single aircraft. It is defined as follow:

- If the aircraft is not being addressed by an action, all of the neighboring locations have equal probability $1/|\mathcal{N}(s^{(i)})|$.
- If the aircraft is being addressed, the commanded neighbor has probability α (a cooperation factor), while other neighboring locations have probability $(1 - \alpha)/(|\mathcal{N}(s^{(i)})| - 1)$.

The CTMDP formulation differs from a discrete time MDP in its use of intensity matrices to represent the transition probabilities. In the case of a single aircraft in the pattern, the intensity matrix would be

$$Q_a = \begin{bmatrix} -q_a(\ell_1) & q_a(\ell_1, \ell_2) & \cdots & q_a(\ell_1, \ell_n) \\ q_a(\ell_2, \ell_1) & -q_a(\ell_2) & \cdots & q_a(\ell_2, \ell_n) \\ \vdots & \vdots & \ddots & \vdots \\ q_a(\ell_n, \ell_1) & \cdots & \cdots & -q_a(\ell_n) \end{bmatrix} \tag{3}$$

where $q_a(\ell_i, \ell_j)$ is the intensity of the aircraft transitioning from location ℓ_i to ℓ_j when action a is taken. Before transitioning, the process remains in ℓ_i for a period of time distributed exponentially with parameter $q_a(\ell_i) = 1/\bar{t}_i$, where $\bar{t}_i$ is the mean sojourn time. The probability that the process transitions to ℓ_j is given by $q_a(\ell_i, \ell_j)/q_a(\ell_i)$. Intensity matrices can be related to MDP transition functions using the sojourn time matrix M_a and the Markov transition matrix $T_a(\ell_i, \ell_j) = T(\ell_j \mid \ell_i, a)$ as follows [13]:

$$M_a = \text{diag}(\frac{1}{\bar{t}_1}, \ldots, \frac{1}{\bar{t}_n}), \tag{4}$$

$$Q_a = M_a(T_a - I). \tag{5}$$

In order to describe the joint intensity matrices for K aircraft in the pattern, we use Kronecker algebra [15]. In particular, under the assumption of independent transition events, the joint intensity matrix $\mathcal{Q}_a$ is:

$$\mathcal{Q}_a = Q_a^{(1)} \oplus \cdots \oplus Q_a^{(K)}, \tag{6}$$

where $\oplus$ is the Kronecker sum operator [16], and $Q_a^{(i)}$ denotes the intensity matrix for the ith aircraft. The matrix $\mathcal{Q}_a$ has dimensions (n^K, n^K) but is sparse.

B. Extension to Generalized Semi-MDP

CTMDPs allow continuous time to be computationally tractable by assuming exponentially distributed sojourn times. Specifically, the tractability is due to the memorylessness of exponential distributions. Unfortunately, such distributions also restrict the coefficient of variation to $cv = \frac{\sigma}{\mu} = 1$. This limitation can be addressed by using a generalized semi-MDP (GSMDP) formulation which handles any non-negative distribution G for the sojourn time [17].

Despite its name, the GSMDP formulation leads to a non-Markovian representation. This shortcoming is typically addressed by approximating the distribution G using phase-type distributions, which are effectively a network of exponential distributions [17]. However, while this approximation allows us to better represent the sojourn time in each state, it comes at the cost of increasing the size of the state space. If it takes c phases to represent each state, the state space size is $(cn)^K$. For example, with just $c = 4$ phases for each of the $K = 4$ aircraft in the $n = 33$ pattern, the size is on the order of 100 million.

In many systems that are modeled as a GSMDP, the phase states are considered fictitious. But since there is a connection between the time an aircraft has been in a state and the distance it has traveled, we can think of the phases as further decompositions for each state. Therefore, unlike other applications, phases in the auto-ATC system might be observable. For example, the north-east measurement for an aircraft can be used to infer whether it is in the early or later phase of the cross-wind leg depending on its distance from the runway centerline.

III. SOLUTION METHODOLOGY

Similar to an MDP, the goal in a CTMDP is to select actions $a = \pi(s)$ which maximize the future expected rewards function $U^\pi(s)$. For a CTMDP with future rewards exponentially discounted with rate ζ, $U^\pi(s)$ takes the form:

$$U^\pi(s) = r(s, \pi(s)) + \frac{\rho(s,a) + \sum_{s' \neq s} q_{\pi(s)}(s, s') U^\pi(s')}{\zeta + q_{\pi(s)}(s)} \tag{7}$$

There are different methodologies for finding the optimal policy π^* which yields the optimal value function U^*. Unfortunately, problems with phase-type distributions can become computationally intractable due to the exponential size of the state space [18]. Several solutions have been proposed to handle large state CTMDPs. Kan and Shelton use an approximate linear program approach, where the structure of the problem is taken into account in order to make it scalable [14]. Younes transforms the CTMDP into an MDP using the process of uniformization, and uses value iteration in conjunction with decision diagrams, where the fact that some of the states are phases is used to improve the algorithm [19]. In this paper, we show how we can take advantage of the structure of the auto-ATC problem to efficiently compute solutions despite the large state space. We use Gauss-Seidel value iteration [20] directly on the value function defined for the CTMDP, i.e., without using uniformization. The computational efficiency comes from leveraging the independence of the events and the homogeneity of the agents, both of which contribute to sparse intensity matrices.

The remainder of this section outlines the methodology. We note that it can be applied to problems with central control of multiple independent homogeneous agents, under communication or resource constraints which arise from the fact that only one agent can be addressed or controlled at the same time.

A. State Reordering

Given that the problem is compromised of K homogeneous agents, the value function is invariant under permutations. Therefore, if P is a permutation matrix, and using the numerical alias representation of s, $U(Ps) = U(s)$. Instead of computing the value function for all n^K permutations with repetition, we only need to compute $\binom{n+K-1}{K}$ combinations with repetition. These combinations are a compact state representation S_c. For large values of n, $|\mathcal{S}_c|$ is still exponential in K. However, $\frac{|\mathcal{S}|}{|\mathcal{S}_c|} \approx K!$ which yields a reduction in both memory and computation complexity that makes realistic problem sizes solvable in a practical amount of time.

B. Action Reordering

Given that the value function is invariant under permutations, and that only one agent can be controlled at each time, the state action value function $\mathtt{Q}(s, a)$ for action $a = (a_1, a_2)$ can always be written in such a way that $a_1 \leq 1$ (note the use of a different font for the $\mathtt{Q}$-value to avoid confusion with the intensity matrices Q_a). In other words, if $a_1 > 1$, we can always find a permutation such that $P_{1,a_1} = 1$, and $\mathtt{Q}(s, a) = \mathtt{Q}(Ps, (1, a_2))$. While we still have to evaluate all actions including those with $a_1 > 1$, the reordering has the effect of reducing the number of intensity matrices we need to compute. Specifically, we only need to be able to compute the joint intensity matrices $\mathcal{Q}_{0,0}$ and $\mathcal{Q}_{1,a_2}$ for $a_2 \in [1, \ldots, \max_i(\mathcal{N}(\ell_i))]$. Therefore, $\mathcal{Q}_a$ takes the form:

$$\mathcal{Q}_a = Q_{a_2} \oplus \underbrace{Q_0 \cdots \oplus Q_0}_{K-1} \tag{8}$$

C. Leveraging the Sparse Joint Intensity Matrices

The $\mathcal{Q}_a$ matrices are sparse due to the nature of the Kronecker sum and product. Additionally, the individual Q_a matrices for each aircraft are also sparse due to low connectivity of the graph representing the aircraft in the pattern. Despite the sparsity, computing and storing all combinations of $\mathcal{Q}_a$ requires a large amount of memory. Instead of storing the matrices, we explain in this subsection how to efficiently compute the relevant row at each iteration.

Using matrix analysis, the following can be proven:

$$C = \underbrace{Q_0 \oplus \cdots \oplus Q_0}_{K-1} \tag{9}$$

$$= \sum_{u=1}^{K-1} P_{(n^u, n^{K-1-u})} \left(I_{n^{K-2}} \otimes Q_0\right) P^{\intercal}_{(n^u, n^{K-1-u})}, \tag{10}$$

where $P_{(m,p)}$ is known as the perfect shuffle permutation [16]. In value iteration, calculating $\mathtt{Q}(s, a)$ involves computing the dot product of $U(s)$ and the $i = \text{sub2ind}((n, \ldots, n), s^{(1)}, \ldots, s^{(K)})$ row of $\mathcal{Q}_a$. The sub2ind function returns the equivalent linear index which corresponds to a given set of multi dimensional indices (we assume a row major implementation). Its inverse is the ind2sub function.

Since $\mathcal{Q}_a$ is the result of a Kronecker sum, the row in question can be computed as follows:

$$(\mathcal{Q}_a)_i = (Q_{a_2} \oplus C)_i \tag{11}$$
$$= (Q_{a_2} \otimes I_{n^{K-1}} + I_n \otimes C)_i \tag{12}$$
$$= (Q_{a_2})_l \otimes e_m^\intercal + e_l^\intercal \otimes C_m, \tag{13}$$

where $(m, l) = \text{ind2sub}((n^{K-1}, n), i)$ and e_l, e_m are Cartesian coordinates versors of lengths n and n^{K-1} respectively. Computing the m row of C deserves some discussion: although it is of length n^{K-1}, it is the result of summing and shuffling different rows of the same matrix Q_0. Using a compressed row storage representation of Q_0, C_m can be computed efficiently because Kronecker multiplication with identity matrices and regular matrix multiplication with the shuffling matrices only involve changing column indices.

D. Reducing Number of Value Iterations

The above concepts allow us to efficiently carry out each iteration of Gauss-Seidel. However, we might require a lot of iterations before the value function converges. One possibility to speed-up convergence is to warm start by computing $U(s)$ using a policy which is computed using a coarser number of phases. The value function with the larger number of phases can then be computed using Gauss-Seidel.

Another option is to use state prioritization methods [21]. These methods do not necessarily result in an improvement due to the cyclic nature of the states. However, by introducing terminal states in the problem (in the auto-ATC system, we set states with collisions as terminal), these cycles can be broken which yields fewer number of iterations to converge.

E. Modified Gauss-Seidel Algorithm

Putting all of the above concepts together, a modified Gauss-Seidel value iteration algorithm which leverages the problem structure is given in Alg.1. The key take-away of the algorithm is the outer loop iteration over only the compact representation $\mathcal{S}_c$, and the fast computation of $\mathtt{Q}(s, a)$ by properly indexing into the appropriate Q matrices.

Algorithm 1 Modified Gauss-Seidel Value Iteration

function GAUSSSEIDELVALUEITERATION
  $U(s) \leftarrow 0$ for all $s \in \mathcal{S}$
  repeat
    for all $s \in \mathcal{S}_c$ **do**
      $U(s) \leftarrow \max_{a \in A(s)} \mathtt{Q}(s, a, U)$
      for all $s_p \in \text{permutations}(s)$ **do**
        $U(s_p) \leftarrow U(s)$
  until convergence
  return U
function $\mathtt{Q}(s, a, U)$
  $s \leftarrow \text{permute}(s, 1, a_1)$
  $i \leftarrow \text{sub2ind}((n, \ldots, n), s^{(1)}, \ldots, s^{(K)})$
  $(m, l) \leftarrow \text{ind2sub}((n^{K-1}, n), i)$
  $C_m \leftarrow (Q_0 \oplus \cdots \oplus Q_0)_m$
  $\mathcal{Q}_i \leftarrow (Q_{a_2})_l \otimes e_m^\intercal + e_l^\intercal \otimes C_m$
  return $r(s, a) + (\rho(s, a) + \mathcal{Q}_i \cdot U)/(\zeta - \mathcal{Q}_i(i))$

IV. EXPERIMENTAL RESULTS

To illustrate that our approach leads to computationally tractable solutions despite the large state space, we report the solution times in Table I for $n = 33$, $K = 4$, and different numbers of phases c. We used a discount rate of $\zeta = 0.5$ and allowed the value function to converge within 0.1% of the maximum absolute reward. Smaller values of ζ will require longer to converge. The usual MDP discount factor can be computed from ζ as $\frac{q}{q+\zeta}$ with $q = \max(-\text{diag}(\mathcal{Q}))$. A constant ζ yields larger values of γ for more phases, but this is a desired behavior because the higher number of state variables requires a larger γ to maintain similarity. These results were obtained using a single thread on a 1.9 GHz Intel i7 laptop. We expect that solving MDPs takes polynomial time in the size of the state space $|\mathcal{S}|$ [22]; however, our methodology yields linear runtime in $|\mathcal{S}|$ for our problem structure, allowing us to solve large problems with over 100 million states in about 30 minutes on a laptop. Note that $|\mathcal{S}|$ is still exponential in the number of aircraft K. In this paper, we assumed four aircraft in the pattern which is reasonable. Indeed, the pattern becomes congested with five or more aircraft even at towered GA airports.

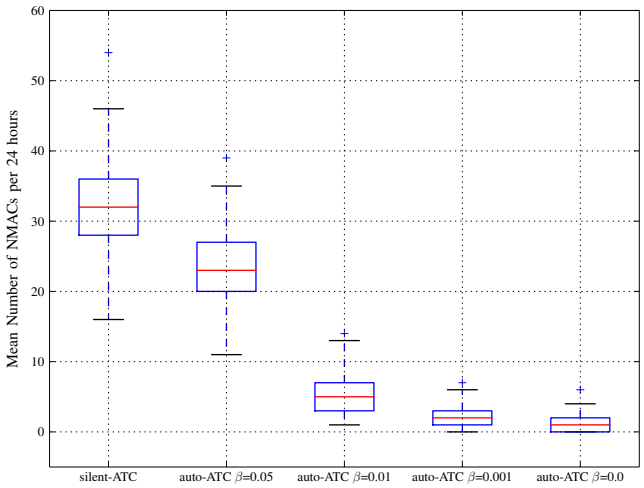
TABLE I: Runtimes for different state space sizes

c	$\approx \lvert\mathcal{S}\rvert$	Iterations	$\approx$ Runtime
1	$1 \cdot 10^6$	25	7 sec
2	$14 \cdot 10^6$	31	2 min
3	$63 \cdot 10^6$	47	13 min
4	$187 \cdot 10^6$	56	30 min
6	$896 \cdot 10^6$	75	2.5 hr

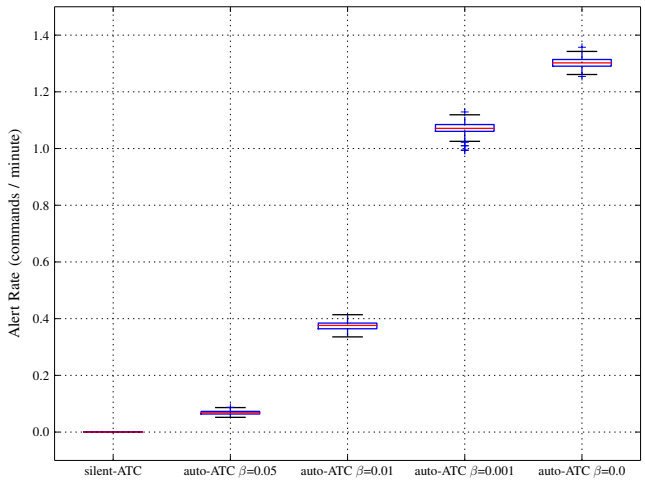
We now present results of the optimized policies evaluated in a 3D simulation of aircraft in the pattern. We only provide a short summary of the assumptions made in the simulation; further detail is included in an earlier paper [11]. Each aircraft is parametrized using north-east-down coordinates ($x = [x_N, x_E, x_D]^\top$), heading (ψ), and airspeed (V). We assume the pilot perfectly regulates the flight-path angle γ_f and roll ϕ while maintaining coordinated flight. The resulting equations of motion are integrated using the Euler method. In order to make the aircraft fly around the pattern, we implemented two successive loop controllers:

- The aircraft flies towards a way-point by commanding a desired altitude h_d and bearing ψ_d.
- The h_d and ψ_d are tracked by commanding γ_f and ϕ.

We define near-mid air collision (NMAC) events as any two aircraft coming within 150 m($\approx$ 500 ft) horizontally or 30 m($\approx$ 100 ft) vertically of each other. We keep track of the number of NMAC events, and reset the aircraft locations each time one occurs. Simulations are initialized with the aircraft at random airspeeds and locations in the pattern. We run 100 batches, each simulating 24 hours of flight. We note that the mean-time to NMAC events in the simulations is not necessarily an accurate representation of reality, where we expect a higher level of baseline safety due to the



(a) Flight hours before NMAC event.



(b) Alert rates.

Fig. 2: 3D simulation results for six phases.

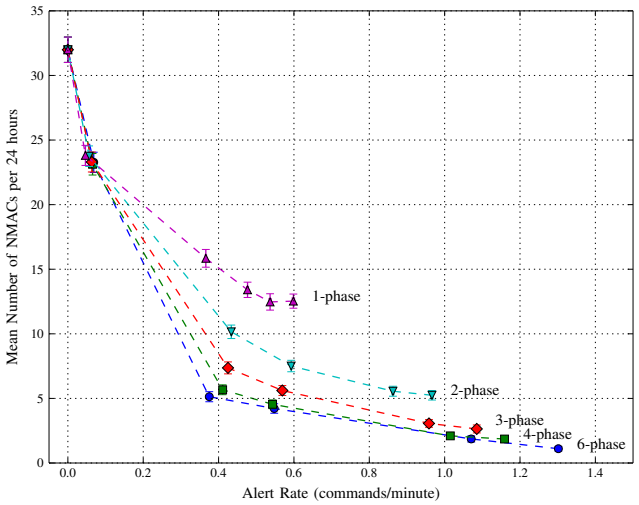


Fig. 3: Pareto fronts for varying number of phases.

pilots' ability to see and avoid other aircraft. Nonetheless, the simulation results are useful to evaluate the performance of the policies relative to each other.

We used Erlang-type distributions for the states and varied the number of phases from one to six (except for the runway, departure and arrival states which always use just one phase since they have a larger coefficient of variance). We estimated the parameters for the distribution from the time the aircraft spend in each state in the 3D simulation. The results of the policy extracted from using the six stage phase-type distribution is shown in Fig. 2 for different ratios (β) of alert cost to collision cost. The reward function of the CTMDP is multi-objective, and when we decrease β we expect higher mean times between NMAC events at the cost of larger alert rates. This behavior is preserved when the optimized policies are evaluated in the 3D simulation. This result is in contrast to our previous MDP formulation, where policies that were expected to achieve higher mean times between NMAC did worse when evaluated in the 3D simulations [11]. Therefore, our hypothesis that a CTMDP would better model the problem is confirmed.

We expected that increasing the number of phases in the Erlang distributions would lead to better system models and therefore improved performance in the 3D simulations. To verify this, Fig. 3 shows the Pareto fronts for different numbers of phases. The policies with more phases dominate those with fewer phases, achieving higher mean times between NMAC events using fewer alerts. The one-phase approximation performs rather poorly due to the exponential distribution being a poor approximation of the sojourn times. While there is a significant improvement in performance when we increase the phases up to four, there is a diminishing return beyond that. We might be able to improve the results by using different phase-type distributions (mixtures of Erlang and Coxian) and moment matching techniques [19].

We also experimented with initializing the aircraft in the steady-state distributions associated with the closed-loop CTMDPs. Computing these distributions can be done using Jacobi iteration [23], which can be efficient by leveraging Kronecker algebra. We did not find a significant difference in the results, and therefore we chose to omit the details.

The source code used to generate the experimental results is available at `https://github.com/sisl/autoATC`.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we showed how an autonomous ATC system for aircraft operating in the vicinity of a non-towered airport can be modeled as a CTMDP. We identified that such a system is equivalent to centrally controlling homogeneous multi-agents under communication or resource constraints which arise from the fact that only one agent can be addressed or controlled at each time step. We showed that the structure of this class of systems can be leveraged to efficiently solve for optimal policies despite the large exponential size of the

state space. In a 3D simulation of the aircraft in the pattern, we demonstrated that the extracted policies reduce the risk of near mid-air collision. As for future work, we believe the following items are worth investigating:

Additional Actions: The allowable actions in our formulation only control the probability of transition to the next states. One possibility is to allow for actions, such as S-turns, which change the sojourn time in a state. We can think of such actions as commanding the aircraft to decrease or increase its groundspeed.

Model Uncertainty: A major assumption in this work is that the aircraft states and phases are exactly known by the system. In practice, we will need to estimate these states from the ground sensors. The problem can then be reformulated as a partially observable Markov decision process (POMDP). When combined with the CTMDP formulation, this leads to a PO-CTMDP, or a continuous-time hidden MDP (CTHMDP). Additionally, the parameters for the 3D simulation (airspeeds, turning radius, controller gains, etc.) were chosen using engineering judgment. These parameters, along with transition probabilities and observation model for the POMDP, could be derived from recorded data.

Practical Implications: A real-life implementation needs a way to identify the aircraft for communication purposes. This can be done using using the aircraft call-sign inferred through speech recognition combined with a VHF direction finder. Finally, a practical implementation needs to handle special cases such as varying number of aircraft in the pattern, aircraft overflying the runway above the traffic altitude, and change of runway direction due to shifting wind.

REFERENCES

[1] AOPA Air Safety Foundation, "Nall Report: Accident Trends and Factors for 1999," *AOPA Air Safety Foundation*, 2000.

[2] R. C. Matthews, "Characteristics of US Midairs," *FAA Aviation News*, vol. 40, pp. 1–3, 2001.

[3] Australian Transport Safety Bureau, "Review of Midair Collisions Involving General Aviation Aircraft in Australia between 1961 and 2003," *Research Report B2004/0114*, 2004.

[4] F. Kunzi and R. J. Hansman, "Mid-Air Collision Risk And Areas Of High Benefit For Traffic Alerting," in *AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, 2011.

[5] C. E. Lin and Y.-Y. Wu, "TCAS Solution for Low Altitude Flights," in *Integrated Communications Navigation and Surveillance Conference (ICNS)*, 2010.

[6] T. B. Billingsley, M. J. Kochenderfer, and J. P. Chryssanthacopoulos, "Collision Avoidance for General Aviation," *IEEE Aerospace and Electronic Systems Magazine*, vol. 27, no. 7, pp. 4–12, 2012.

[7] D. Diefes and R. Deering, "Concept Design for a Low Cost Cockpit Display/Collision Avoidance System for General Aviation Aircraft," in *IEEE Position, Location and Navigation Symposium (PLANS)*, 1996.

[8] AOPA, EAA, *et al.*, *Open Letter to Federal Aviation Administration*, 2015. [Online]. Available: `download.aopa.org/advocacy/150122-Huerta-ADS-B-FINAL.pdf`.

[9] S. D. Campbell, "Small Airport Surveillance Sensor," in *MIT Lincoln Laboratory ATC Workshop*, 2014.

[10] T. Nikoleris, H. Erzberger, R. A. Paielli, and Y.-C. Chu, "Autonomous System for Air Traffic Control in Terminal Airspace," in *AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, 2014.

[11] Z. Mahboubi and M. J. Kochenderfer, "Autonomous Air Traffic Control for Non-Towered Airports," in *USA/Europe Air Traffic Management Research and Development Seminar*, 2015.

[12] J. E. Holland, M. J. Kochenderfer, and W. A. Olson, "Optimizing the Next Generation Collision Avoidance System for Safe, Suitable, and Acceptable Operational Performance," *Air Traffic Control Quarterly*, vol. 21, no. 3, pp. 275–297, 2013.

[13] U. Nodelman, C. R. Shelton, and D. Koller, "Continuous time Bayesian Networks," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[14] K. F. Kan and C. R. Shelton, "Solving Structured Continuous-Time Markov Decision Processes," in *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.

[15] C. R. Shelton and G. Ciardo, "Tutorial on Structured Continuous-Time Markov Processes," *Journal of Artificial Intelligence Research*, vol. 51, pp. 725–778, 2014.

[16] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. New York, NY: Cambridge University Press, 1991.

[17] R. G. Simmons and H. L. Younes, "Solving Generalized Semi-Markov Decision Processes using Continuous Phase-type Distributions," in *Conference on Artificial Intelligence (AAAI)*, 2004.

[18] J. V. T. de Sousa Messias, "Decision-Making under Uncertainty for Real Robot Teams," PhD thesis, Instituto Superior Técnico, 2014.

[19] H. L. Younes, "Planning and Execution with Phase Transitions," in *Conference on Artificial Intelligence (AAAI)*, 2005.

[20] M. J. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*. Cambridge, MA: MIT Press, 2015.

[21] D. Wingate and K. D. Seppi, "Prioritization Methods for Accelerating MDP Solvers," *Journal of Artificial Intelligence Research*, vol. 6, pp. 851–881, 2005.

[22] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the Complexity of Solving Markov Decision Problems," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1995.

[23] D. P. Oleary, *Iterative Methods for Finding the Stationary Vector for Markov Chains*. New York: Springer, 1993.