Learning to Trick Cost-Based Planners into Cooperative Behavior

Carrie Rebhuhn, Ryan Skeele, Jen Jen Chung, Geoffrey A. Hollinger and Kagan Tumer

***Abstract*— In this paper we consider the problem of routing autonomously guided robots by manipulating the cost space to induce safe trajectories in the work space. Specifically, we examine the domain of UAV traffic management in urban airspaces. Each robot does not explicitly coordinate with other vehicles in the airspace. Instead, the robots execute their own individual internal cost-based planner to travel between locations. Given this structure, our goal is to develop a high-level UAV traffic management (UTM) system that can dynamically adapt the cost space to reduce the number of conflict incidents in the airspace without knowing the internal planners of each robot. We propose a decentralized and distributed system of high-level traffic controllers that each learn appropriate costing strategies via a neuro-evolutionary algorithm. The policies learned by our algorithm demonstrated a 16.4% reduction in the total number of conflict incidents experienced in the airspace while maintaining throughput performance.**

I. Introduction

The use of unmanned aerial vehicles (UAVs) in commercial applications seems inevitable, and recent proposals to update legislation regarding UAV operation in the US airspace [1] have accelerated this discussion. Many operators seem poised to take advantage of any relaxation in the current legislation to introduce UAVs to the urban airspace, with the goal of having these aerial robots fulfill emergency aid and package delivery roles among other tasks.

Allowing commercial use of UAVs up to 500ft may lead to dense UAV traffic in the urban airspace, which will need to be managed to reduce conflict incidents arising from traffic congestion. A UAV traffic management (UTM) system will need to adapt to changing demands on the airspace and dynamically apply travel costs to influence traffic flow. Furthermore, with any number of UAV operators designing custom hardware and software, it is desirable to design a system that is capable of managing traffic density without needing to know or regulate the onboard planning algorithms of each UAV in the airspace.

Existing work in air traffic control has focused on the flow of airplanes through fixed geographical points and have typically used multiagent system methods to model the distributed system of traffic controllers [2]. A multiagent reinforcement learning framework was used in [3] to reduce congestion by setting the required separation between aircraft, ordering ground delays or ordering reroutes. A hierarchical system of management was considered in [4], with airlines negotiating for airspace to minimize chances of congestion at the higher strategic level, and at the lower tactical level, air traffic controllers could reroute aircraft by introducing no-go zones called "shocks".

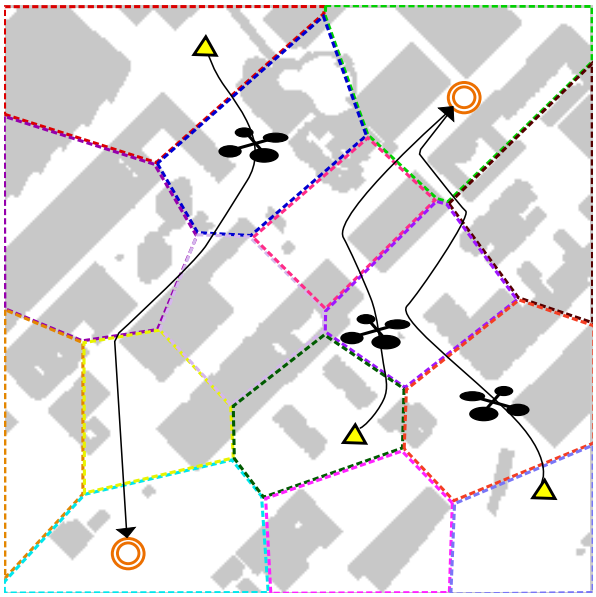


Fig. 1: The UAV traffic management problem. The airspace is divided into discrete sectors with a single UTM agent controlling the cost of travel through each sector. UAVs traveling from start (triangle) to goal (circle) locations plan and execute paths to reduce their individual travel cost without explicitly coordinating with other UAVs in the airspace. This can lead to potentially dangerous conflict incidents with significant knock-on effects. The goal of the sector agents is to learn appropriate costing strategies to alleviate bottlenecks in the traffic and minimize congestion in the airspace.

While some aspects of these approaches can be applied to the UTM problem, many new issues arise in this new domain that are not addressed by existing traffic management strategies. Most notably, the national airspace is a large and relatively obstacle-free environment, whereas this is not always the case in an urban setting with UAVs operating in close proximity to one another. If the flow of UAVs is not well-managed, congestion in cluttered environments may exceed a UAV's ability to plan around other UAVs' trajectories, potentially escalating to larger knock-on effects throughout the system. However, it is impractical to model the behaviors of all the actors in the system. This is further complicated by the diverse range of planning software that may be used across the platforms present in the airspace, especially if they are competing for throughput and do not explicitly coordinate with foreign platforms to avoid conflict

This work was supported by NASA grant NNX14AI10G.

Carrie Rebhuhn, Jen Jen Chung and Kagan Tumer are with the Autonomous Agents and Distributed Intelligence Lab, School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, OR, 97330, USA rebhuhnc@onid.oregonstate.edu, {jenjen.chung, kagan.tumer}@oregonstate.edu

Ryan Skeele and Geoffrey A. Hollinger are with the Robotic Decision Making Lab, School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, OR, 97330, USA skeeler@onid.oregonstate.edu, geoff.hollinger@oregonstate.edu

situations.

This motivates the use of a high-level UTM learning framework that can map the salient features of the UAV airspace to control actions that can mitigate congestion, such as increasing the cost of flying through particular areas. In this paper, we propose a UTM strategy employing a decentralized system of neuro-evolutionary learning agents that manage the flow of UAV traffic through their designated sectors of the airspace. Each sector agent learns a policy that assigns the cost of travel through the sector according to the current number of UAVs in the airspace. Policy evaluation is performed by assessing the total number of UAV conflicts in the airspace during a single learning epoch.

Results from testing on a simulated urban environment show that given only local information regarding the number of UAVs in their sector and the intended heading of each UAV, and training on the total conflict count, the team of sector agents was able to learn appropriate costing strategies to reduce the number of conflicts experienced in the entire airspace. Comparison to a uniform costing strategy showed on average a 16.4% reduction of conflicts in the airspace after 100 learning epochs.

II. Related Work

The UTM problem draws from two areas of research. The first is multiagent learning for air traffic routing. In this area there has been successful research demonstrating the improved performance of machine learning techniques for complex system management over traditional air traffic management methods [3]. The second research area is in robotic routing and scheduling where both global and distributed approaches have been applied to routing robot teams.

A. Multiagent Learning for Air Traffic Routing

Recent work in multiagent systems has focused on developing effective routing for commercial air traffic across the national airspace. In air traffic, delays and congestion can cascade throughout the system, and are difficult to mitigate and control. Using reinforcement learning agents to manage air traffic through geographical fixes, Agogino and Tumer [3] were able to reduce airspace congestion by over 90% when compared to current traffic control methods. Similar demonstrations were performed using evolutionary algorithms to train the learning agents [5], [6].

In this work, we apply a similar network of routing agents to control the flow of UAV traffic. However, we consider the UAV domain where platforms are not restricted to fly through particular fixes in the environment. Furthermore, our routing algorithm is based on the assumption that we have no direct control over the path planning aspect of the UAVs.

B. Robotic Routing and Scheduling

Previous research has explored congestion as a centralized controller scheduling and routing problem. A comprehensive survey of centralized scheduling methods for automated vehicles is given in [7]. Using centralized methods, every robot in the system is told when it can travel, and where it can travel. In order to compute a solution, these methods require full state information and are often slow and computationally complex. Dynamic routing methods [8], [9] manage the time-window of each robot through time expanded graphs without needing full state information [10]. However these methods become computationally expensive when applied to a fast-changing system.

Other methods for traffic routing have focused on vehicle negotiation to resolve conflicts in congested areas. Large numbers of aircraft can use these schemes to resolve local conflicts resulting in improved system performance [11]. Some research has developed peer to peer collision avoidance protocols. More specifically, a software called AgentFly uses an agent-based distributed negotiation approach to the air traffic routing problem [12]. This technique works well in domains with standardized communication and vehicle abilities. However, UAVs are very diverse in both hardware and software. The method that we propose in this paper allows for these characteristics of the domain and still leaves open the possibility of future progression towards standardization.

Unlike previous research we use an incentivized routing technique instead of a central controller or vehicle negotiation scheme. Our proposed UTM system does not assign paths to each robot nor does it require the robots to act in a cooperative manner. Our algorithm assumes each UAV is using a cost-based planner and so manipulates the cost of travel through the airspace to incentivize the robots to avoid congested areas.

III. Problem Formulation

Consider an airspace containing multiple UAVs traveling between various start and goal locations. Each UAV plans its trajectory through the airspace according to a cost-based path planner aimed at minimizing travel cost. Furthermore, it is assumed that apart from low-level collision avoidance procedures, each UAV acts independently and does not coordinate with any other UAV in the airspace. Our goal in this work is to monitor and reduce congestion in the airspace by employing high-level UTM agents that manage the cost of traveling through sectors of the airspace.

A schematic of the problem setup is shown in Fig. 1. The airspace is divided into sectors that are each controlled by a single UTM agent. UAVs flying from start to goal locations in the world must travel through a series of sectors and are subject to the cost of travel in each sector. The sector agents act as a decentralized system of traffic controllers that each individually learn to apply the appropriate costs in their respective sectors to minimize overall congestion in the entire airspace. The obstacle map shown in Fig. 1 is a section of San Francisco's South of Market area, with data gathered from a building height map [13].

Given the potentially large number of UAVs in each sector, it is impractical to specifically account for the individual states and policies of all UAVs. This motivates the use of a learning algorithm for developing costing strategies that map a reduced sector state space to the travel costs applied in the sector. In this work, we formulate the control

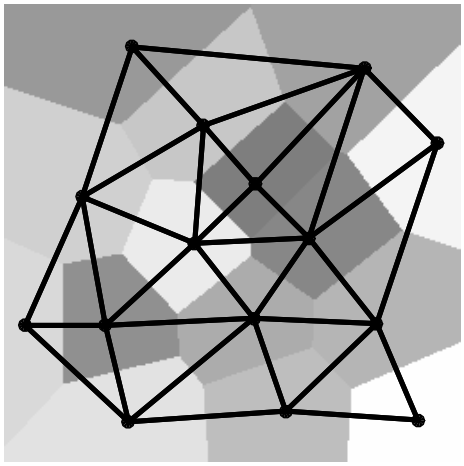


Fig. 2: Graph G_h of sector agent connections used in the experiments. Agent policies assign edge costs based on the current sector state s, and the discretized cardinal direction followed on the edge.

policy of each sector agent as a single-layer neural network (NN) where the weights of the NNs are learned using an evolutionary algorithm. The following section describes the neuro-evolutionary algorithm used to develop the control policies for each sector agent.

A trivial solution to this problem would seem to be to move to different altitudes in order to avoid congested areas. However if this space is sufficiently congested, a UAV could encounter areas of congestion above or below it as well, particularly if other UAVs are taking the same congestion-avoidance maneuvers. Our approach to incentivizing travel through an airspace could be extended for use with 3-dimensional path-planners and multiple layers of sector agents, but for now we restrict the UAV planner actions and essentially look only at one altitude cross-section of UAV traffic.

Our approach presents a reasonable formulation of the UAV traffic management problem because it decouples the traffic management approach from the technology on the UAV. Competing commercial entities may not use the same planners or UAVs, but cost-based planners are commonplace in robotics. Furthermore, because airspace safety is essential to the operation of aerial vehicles, it is reasonable to assume that an airspace authority (for our work, sector agents) will have control over the movement of traffic. We present this control in an incentive structure, similar to imposing tolls.

IV. Neuro-Evolutionary Learning

Neural networks are function approximators that can be used to model continuous state-action control policies to arbitrary accuracy while only requiring a coarse approximation of the system state [14]. This is suitable for the UTM problem considered in this paper since it allows a simplification of the system state to the discretized UAV headings while requiring the sector agents to output actions drawn from the space of continuous travel costs. For each NN controller, the four-dimensional input state is defined as the number of UAVs in the sector discretized into the four

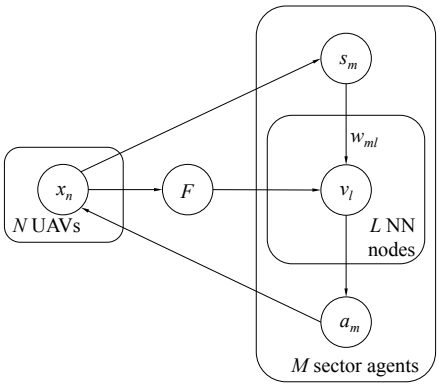


Fig. 3: Interaction between sector agents and UAVs in the airspace. The state, s, of each sector agent includes the number of UAVs in its sector discretized into four cardinal travel directions. The actions, a, output by each agent are the cost to travel in each direction in the sector. The total number of conflicts, F, in the airspace is tracked for each training epoch and represents the fitness of the NN controllers, which each have L nodes in their hidden layer. The evolutionary algorithm selects the NNs with the fittest set of weights, w, at the end of each epoch.

cardinal heading directions, that is,

$$\mathbf{s} = [n_N, n_E, n_S, n_W]\,. \tag{1}$$

The output action is also a four-dimensional vector specifying the cost of travel in each of the cardinal directions,

$$\mathbf{a} = [c_N, c_E, c_S, c_W]\,. \tag{2}$$

These costs are applied to each of the directed edges connecting the sector agent graph (shown in Fig. 2) by again discretizing the bearing of each edge to the nearest cardinal direction. The edge costs are broadcast to UAVs in the airspace that then recompute their flight trajectories to reduce their individual cost of travel. This abstraction into four cardinal directions allows greater generality, and does not impose the graph structure on the learning process.

As the UAVs execute their planned trajectories, conflicts occur when two or more UAVs pass within some distance, $d_{conflict}$, of one another. The number of conflicts detected in the entire airspace over a single learning epoch is then used to calculate the team fitness evaluation. A graphical model of the system is shown in Fig. 3.

The weights of the NN controllers of each sector agent are trained using a cooperative coevolutionary algorithm (CCEA), as given in Algorithm 1. As an extension of standard evolutionary algorithms, CCEAs have been shown to perform well in cooperative multiagent domains [15]. This allows multiple populations to evolve in parallel such that each population develops a policy for a single sector agent in the UTM system. In the following experiments, there are M coevolving populations of NN controllers, one for each sector agent and each with a population size of k. For mutation of the NNs, 50% of the network weights are chosen at random and values drawn from a normal distribution (zero mean and unit variance) are added to the weights.

The neuro-evolutionary algorithm used in the following experiments is given in Algorithm 1. The

Algorithm 1 Neuro-Evolution for Sector Agents

1: Initialize population of k neural networks for each $agent$
2: **while** $epoch < total\_epochs$ **do**
3: Mutate populations to produce k more children
4: **for** $i = 1 \rightarrow 2k$ **do**
5: Select the ith NN from each population
6: Assign NNs to sector $agents$
7: $team\_fitness$ = SIMULATE($agents$)
8: ASSIGNFITNESS($team\_fitness$)
9: Select fittest k policies and repeat from line 2

SIMULATE($agents$) function on line 7 represents the execution of the policies of the current set of sector agents in the UTM domain and is described in further detail in Algorithm 2. ASSIGNFITNESS($agents$) allocates the fitness calculated from the detected conflicts to each of the agents that participated in the learning epoch.

V. SIMULATION SETUP

We tested our UTM learning algorithm on a simulated urban airspace derived from a 256×256 unit cell map of San Francisco, which is shown in Fig. 1. Simulation timesteps were discretized into 1s time intervals with each learning epoch defined as an evaluation of each random team in the population over 100 simulated timesteps.

A. Sector Agent Parameters

The tested airspace was divided into 15 Voronoi partitions with each sector controlled by a single agent in the UTM system. Voronoi partitioning was hand-selected to place sector agents in locations where congestion was expected due to the obstacle layout of the region. The airspace sectors are shown in Fig. 1 by the dashed lines.

At the start of learning, each sector agent is initialized with a population of $k = 10$ random NN control policies. As described in (1), UAV headings are discretized to the nearest cardinal direction such that at each timestep and in each sector, the total north-, south-, east- and west-bound UAV traffic is tallied to form the input state vector of the NN policy. The output action of each agent, that is, the costs to travel in each direction as shown in (2), were restricted to lie within $c = [0, 1]$.

B. UAV Parameters

Each UAV generates an initial path to the goal by querying the initial cost to travel along all edges of the sector agent graph and performing a high-level A* search to determine the lowest-cost sequence of sectors to travel through to reach their goal location. Given the sector sequence, each UAV then performs a low-level A* search across a discretized unit cell representation of the free space to plan a trajectory from start to goal. The set of initial paths planned by 27 UAVs are shown in Fig. 4.

At each timestep during a learning epoch, sector agents update sector traversal costs based on the current sector state causing each UAV to update its path according to the new cost information. Furthermore, at each timestep and at each start location, or *fix*, a new UAV is generated with probability $p_{gen} = 5\%$ with a randomly-selected goal *fix* and executes the planning procedure outlined above. This value was selected to produce about 100 UAVs in the airspace in a single run, demonstrating the effectiveness of the algorithm in a high-traffic situation. Once a UAV reaches its goal location, it is removed from the airspace.

In the following experiments, each UAV traveled at a constant speed of 1 cell per timestep and had a conflict radius $d_{conflict} = 2$ unit cells. The complete simulation algorithm is provided in Algorithm 2.

Algorithm 2 SIMULATE($agents$)

while $t < T$ **do**
  for each $f \in fix$ **do**
    if p_{gen} **then**
      generate a UAV
  for each $m \in agents$ **do**
    $m.\mathbf{s} \leftarrow [n_N, n_E, n_S, n_W]$
    $m.\mathbf{a} \leftarrow$ GETEDGECOSTS($m.\mathbf{s}$)
  for each $n \in UAV$ **do**
    $n.path_{high}$ = SECTORPATH($G_h, \mathbf{m.a}$)
    $n.path_{low}$ = TRAVERSALPATH($n.path_{high}, G_{obs}$)
    $n.x$ = EXECUTEPATH($n.path_{low}$)
  $conflicts$ += UAVs within a distance $d_{conflict}$ of others
  if any $UAVs$ have reached goal **then**
    Remove respective $UAVs$ from airspace
F = $f(conflicts)$
return F

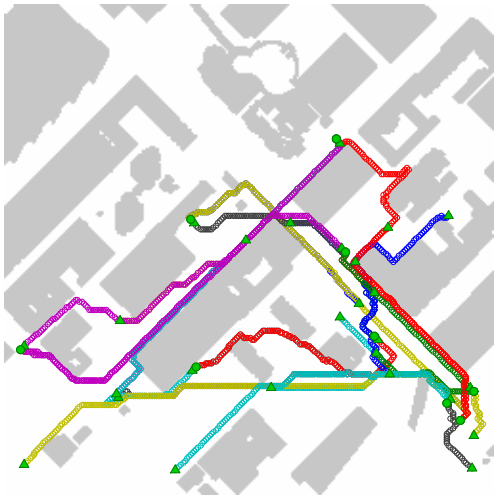
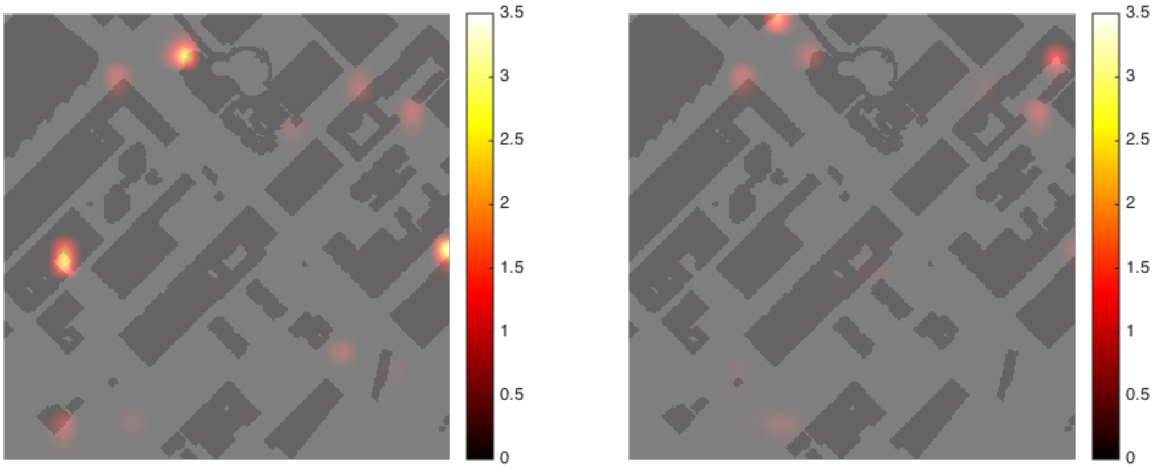


Fig. 4: Initial planned paths for 27 UAVs in the airspace travelling from start (solid green triangle) to goal (solid green circle) locations. It can be seen that high levels of congestion may be expected in some thoroughfare regions of the airspace, such as the bottom right area where there are a group of close fixes.

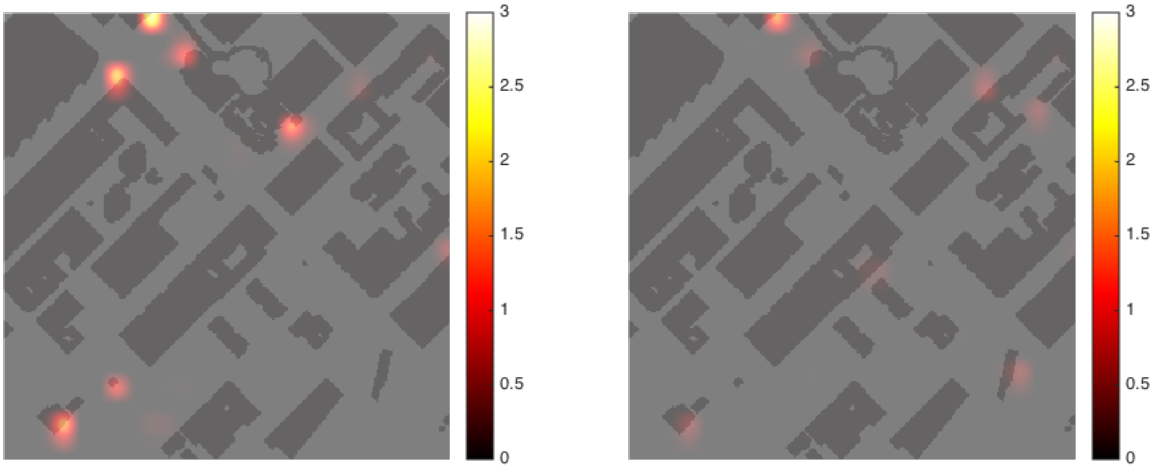
C. Hierarchical Path Planner

We used two levels of path planners for the simulation, a high-level sector planner and a low-level traversal planner.



(a) (b)

Fig. 5: Change in congestion observed over one run of the *linear* conflict evaluation trials. The overlaid heat map shows the number of conflicts experienced in an area for (a) the best evaluation trial for agents with random initialized sector travel costs and (b) the best evaluation trial for agents with evolved sector travel costs after 100 epochs. The overall number of conflict instances has reduced with some high congestion regions removed completely.



(a) (b)

Fig. 6: Change in congestion observed over one run of the *quadratic* conflict evaluation trials. The overlaid heat map shows the number of conflicts experienced in an area for (a) the best evaluation trial for agents with random initialized sector travel costs and (b) the best evaluation trial for agents with evolved sector travel costs after 100 epochs. As with the *linear* trials, the overall number of conflict instances has been reduced.

Each of these took in a graph of costs and performed an A* search to find the minimum-cost path across the graph from a given start point to a given goal point. Though both planners used A* to traverse a graph, the graph construction was different for each of the planners.

The role of the *high-level* planner was to find the lowest-cost sequence of sectors to visit according to the costs assigned by the sector agents. The A* search was performed over the spatial connection graph, G_h, generated by identifying the neighboring Voronoi sectors shown in Fig. 2. This graph did not consider obstacles across the map, and it was assumed that a clear path existed from points in one sector to the edge of another.

The *low-level* planner represented the connectivity of the unit cell graph with the obstacle map, G_{obs}. An 8-connected graph was constructed with unit traversal costs in any direction. The connectivity of the graph was further restricted by the high-level path. The low-level path was restricted to only those sectors present in the high-level plan. Furthermore, sectors could only be traversed in a particular order and the connectivity of the graph reflected this fact. For example, if a high-level plan specified a path visiting the sequence of sectors $\{1, 5\}$, there would be connections between cells along the border of sector 1 and 5, but only in the direction of 5. Backward traversal into sectors in the high-level map was not permitted.

Path failure occurred when it was impossible to satisfy the high-level plan using a low-level path. This could occur when the path to the specified sector was blocked by an object. In the case that no path was available that satisfied the high-level planner, the UAV did not move in the airspace. Multiple occasions for replanning were available, due to the fact that high-level edge costs changed each time a UAV changed sector membership, so the UAV may have a chance to generate a different high-level plan if a particular one was infeasible.

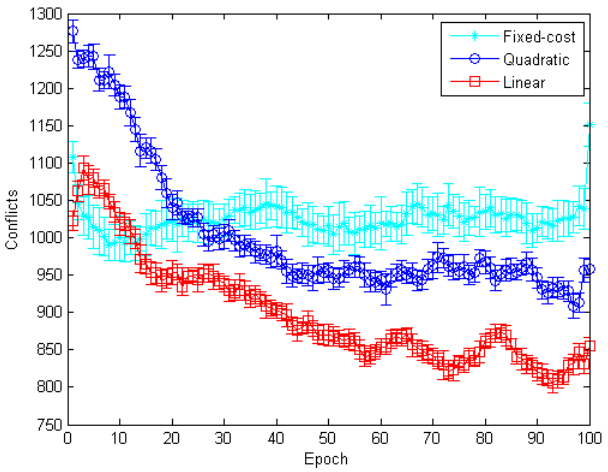


Fig. 7: Comparison of conflict occurrence over epochs using fixed costs versus evolved costs with two different fitness functions.

VI. Results

We performed three sets of simulation experiments, the first tested the proposed UTM learning algorithm trained using the total number of conflicts in the airspace (*linear* conflict evaluation). The second set of trials tested the same algorithm, however in this instance, policies were evaluated according to the sum of squared collocated conflicts in the airspace (*quadratic* conflict evaluation). As a baseline, we performed a set of simulations using uniform costs for travel between the sectors. This meant that the UAVs took the path that traveled through the smallest number of sectors. Each set of experiments contained 20 runs of 100 learning epochs. The algorithm was implemented in C++ with the high-level and low-level planners using the A* search algorithm provided by the Boost Graph Library. Planners re-planned whenever the cost map generated by the agents changed.

Figure 5a shows the congestion observed in the first learning epoch of one *linear* trial. Sector agents initially assign travel costs randomly, so there are areas of high congestion that occur. Figure 5b shows the performance at the end of evolution. Comparing Fig. 5a and Fig. 5b we can see a reduction in the number of conflict instances.

In a realistic UAV routing scenario, the number of close calls doesn't necessarily matter as much as the severity of the congestion in the system. For example, if two UAVs get within a dangerous distance of one another, a backup safety mechanism may be able to divert one and avoid a collision. However, if more UAVs get within dangerous distance of one another, the backup safeties must then deal with the cascading effects of the collision-avoidance maneuver. This could lead to a failure of the lower-level collision-avoidance systems if the conflict density is too high. To address this fact, we also tested a *quadratic* fitness evaluation function that summed the squared number of conflicts for *each cell*. This attempts to decrease the likelihood that there will be large numbers of conflicts that the UAV's low-level planner cannot handle. By comparing Fig. 6a and Fig. 6b we see a larger reduction in the *severity* of congestion across the map, that is, there are fewer conflicts in each hotspot.

We also quantitatively compare the reduction of conflicts using evolution with the *linear* and *quadratic* fitness evaluations to the conflicts arising in the fixed-cost map. We see in Fig. 7 that the number of conflicts using the fixed-cost map is higher than the conflicts using the learned costs after evolution has been performed for 100 epochs. The number of conflicts in the baseline comparison algorithm remains high, with an overall average of 1023 conflicts, while the neuro-evolutionary system is able to reduce the number of conflicts seen in the system. Note that the evolution process does not monotonically decrease the occurrence of conflicts due to the fact that there is randomness in the population generation as well as in the goal selection of the UAVs. However evolution still significantly decreases the occurrence of conflicts in the system, particularly compared with the fixed-cost method. Comparing to the fixed-cost method, we see an average of 856 conflicts at the end of learning with the *linear* fitness evaluation, which represents a 16.4% reduction in total conflicts after 100 epochs using the linear fitness evaluation with evolution. Learning with the *quadratic* fitness evaluation does not outperform the *linear* fitness evaluation in terms of total *number* of conflicts, however this is to be expected since it is not what the fitness function uses to determine survival. Nonetheless we see an average performance of 950 conflicts after 100 epochs, giving us a 7.1% improvement over the fixed-cost method.

The number of UAVs in the system was not held constant during the simulation. Because of the probabilistic generation process, there could be a variable number of UAVs present in the system, but generally this number was close to 100. If the generation rate is lowered, it is possible to achieve zero conflicts in the system without learning intervention, with the tradeoff of severely reduced throughput. If the generation rate is increased, it is also possible to saturate the system with UAVs such that a learning policy cannot effectively reduce the number of conflicts. However, these extremes would not demonstrate realistic scenarios, and would not show the improvements that the algorithm offers. We leave further exploration of the scalability of this policy to future work.

VII. Conclusion

The results presented in this paper show that a distributed UTM system can learn appropriate costs to apply to UAVs traveling in the airspace to incentivize implicit cooperation between the UAV planners. Using our method, we achieved an 16.4% reduction in conflicts compared to the baseline method with uniform sector costs. In terms of the simulated values, this represents a reduction from about 10.2 conflicts to about 8.6 conflicts per timestep over the entire simulation. We also showed a reduction in severity of conflict occurrences using a quadratic fitness function The agents were able to take in directional sector congestion information and appropriately weight the cost of travel to promote safety in the system. It is also worth noting here that agents do not require a model of the available airspace or knowledge of the

obstacles in the sector, the number of conflicts experienced in the sector was sufficient information by which to evaluate the performance of the current policy.

This structure gives a method by which neural networks may be trained for managing conflict occurrences through an obstacle-filled area. Because we introduced random traffic throughout the run, the policies generalize to a variety of different traffic scenarios. The computation time in the evolved policy is therefore a combination of the time to calculate the output of a neural network. The UAVs then independently calculate their trajectories through the space with this information.

The ability to manipulate high-level planners allows us to reduce occurrences of potentially dangerous congestion in the system. UAVs in the real world can handle encounters with other UAVs using low-level collision avoidance procedures, but by reducing the congestion in the airspace at a high level we can permit safer travel by avoiding many of these conflicts before they occur.

This work forms an important step toward the formulation and exploration of the UTM problem. Handling the challenges associated with free flight in an obstacle-filled environment will become critical as UAVs become more prolific in the airspace. Shortest-path algorithms for UAV routing may lead to unsafe UAV density. Intelligent high-level coordination must occur in conjunction with low-level collision-avoidance in order to ensure safety in the airspace.

There are many possible avenues for future work. The first involves research into the optimal placement of sectors in the space. In this work the placement was performed manually, placing sector centers in places that were likely to be intersections, similar to traffic lights. Further optimization could be explored in order to better shape the topology of the desired control points to accommodate the presence of obstacles in the map. The bounds of the usefulness of this neural network approach to routing must be explored as well. We used a probabilistic generation of UAVs, but we did not explore a variety of traffic conditions. Experiments with varying this parameter could lead to insight into the system requirements when using this traffic routing approach.

A* is also not the only or necessarily the most common solution to path planning in an obstacle-filled environment. Different graph-based planning algorithms may be explored at the high and lower levels, using the same evolutionary infrastructure for setting the graph weights. Additionally, the routing algorithm currently does not account for possible heterogeneity in the system. Further improvements could be made to this algorithm if heterogeneity in UAV type or priority could be accommodated.

REFERENCES

[1] FAA, "Operation and certification of small unmanned aircraft systems," February 2014, FAA-2015-0150.

[2] H. Emami and F. Derakhshan, "An overview on conflict detection and resolution methods in air traffic management using multi agent systems," in *16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*. IEEE, 2012, pp. 293–298.

[3] A. K. Agogino and K. Tumer, "A multiagent approach to managing air traffic flow," *Autonomous Agents and Multi-Agent Systems*, vol. 24, no. 1, pp. 1–25, 2012.

[4] C. Bongiorno, G. Gurtner, F. Lillo, L. VAlori, M. Ducci, B. Monechi, and S. Pozzi, "An agent based model of air traffic management," in *Proceedings of the Third SESAR Innovation days. Stockholm (Sweden), Novermber*, 2013.

[5] L. Yliniemi, A. K. Agogino, and K. Tumer, "Evolutionary agent-based simulation of the introduction of new technologies in air traffic management," in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 1215–1222.

[6] W. J. Curran, A. Agogino, and K. Tumer, "Addressing hard constraints in the air traffic problem through partitioning and difference rewards," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1281–1282.

[7] L. Qiu, W.-J. Hsu, S.-Y. Huang, and H. Wang, "Scheduling and routing algorithms for AGVs: a survey," *International Journal of Production Research*, vol. 40, no. 3, pp. 745–760, 2002.

[8] F. Taghaboni-Dutta and J. M. A. Tanchoco, "Comparison of dynamic routeing techniques for automated guided vehicle system," *International Journal of Production Research*, vol. 33, no. 10, pp. 2653–2669, 1995.

[9] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[10] E. Gawrilow, E. Köhler, R. H. Möhring, and B. Stenzel, "Dynamic routing of automated guided vehicles in real-time," in *Mathematics - Key Technology for the Future*, H.-J. Krebs and W. Jäger, Eds. Springer, 2008, ch. 5, pp. 165–177.

[11] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *Automatic Control,IEEE Transactions on*, vol. 43, no. 4, pp. 509–521, 1998.

[12] M. Pechoucek and D. Sislak, "Agent-based approach to free-flight planning, control, and simulation," *Intelligent Systems, IEEE*, vol. 24, no. 1, pp. 14–17, 2009.

[13] D. Burgett, "Sf building heights," https://api.tiles.mapbox.com/v3/dnomadb.ArcToMapbox/page.html, 2014.

[14] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[15] S. G. Ficici, O. Melnik, and J. B. Pollack, "A game-theoretic and dynamical-systems analysis of selection methods in coevolution," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 6, pp. 580–602, 2005.