FairFly: A Fair Motion Planner for Fleets of Autonomous UAVs in Urban Airspace

Connor Kurtz[1] and Houssam Abbas[1]

Abstract— We present a solution to the problem of fairly planning a fleet of Unmanned Aerial Vehicles (UAVs) that have different missions and operators, such that no one operator unfairly gets to finish its missions early at the expense of others - unless this was explicitly negotiated. When hundreds of UAVs share an urban airspace, the relevant authorities should allocate corridors to them such that they complete their missions, but no one vehicle is accidentally given an exceptionally fast path at the expense of another, which is thus forced to wait and waste energy. Our solution, FairFly, addresses the fair planning question for general autonomous systems, including UAV fleets, subject to complex missions typical of urban applications. FairFly formalizes each mission in temporal logic. An offline search finds the fairest paths that satisfy the missions and can be flown by the UAVs, leading to lighter online control load. It allows explicit negotiation between UAVs to enable imbalanced path durations if desired. We present three fairness notions, including one that reduces energy consumption. We validate our results in simulation, and demonstrate a lighter computational load and less UAV energy consumption as a result of flying fair trajectories.

I. INTRODUCTION: WHAT IS FAIR USAGE OF AIRSPACE?

The growth in Unmanned Aerial Vehicles (UAVs) research is driven by many stakeholders, from different industries and government agencies: to list a few, retailers want to use UAVs to deliver goods faster and with less energy expenditure, engineering companies use UAVs to inspect urban infrastructure like rails and solar panels in a more timely manner, city government can use UAVs for traffic analysis over a wider scale, and communications companies envision the creation of 'Cellular Networks on Demand' using UAVs as wireless hotspots at times of greater-than-usual demand or in disasters that reduce the fixed network's capacity. The website Aviation Planning lists over 400 uses of UAVs as of the end of 2018.

The fragmentation in the use cases of UAVs noted above, however, points to a serious challenge facing regulators who want to safely enable advanced urban UAV applications. Namely, UAVs sharing the same airspace have different operators who respond to different priorities. Generally, even if the deadlines for the different missions have been agreed, an operator would still prefer completing its mission earlier rather than later - finishing early can mean higher usage of the UAV, less energy wasted in holding patterns, etc. However, this can conflict with *another* operator completing *its*

[1]The authors are with the School of Electrical Engineering and Computer Science, Oregon State University, OR, USA {kurtzco,houssam.abbas}@oregonstate.edu

mission as soon as it could. Intuitively, if two missions have the same deadline of 10mins, it would be 'unfair' to allocate a 1-min motion path to the first drone, and a 9-mins motion path to the second, assuming a more balanced solution exists. Thus, there remains a need to *balance* the flight durations of all the UAVs, such that all UAVs accomplish their mission within the deadline, and no UAVs are treated unfairly.

This is different from the task of scheduling commercial airliners: in UAV Traffic Management, the number of UAVs sharing a small space is significantly larger, their missions are more complex, their dynamics are more agile, and they are more susceptible to disturbances. These characteristics require flexible yet robust controllers, which complicates the fairness question beyond a scheduling problem. Without a *transparent and explicit* fairness mechanism, smaller operators are discouraged from leveraging UAV technology, innovation can be stifled, and the economic benefits of unpiloted aerial systems are foregone.

Contributions of this work. This paper addresses the question of fair and safe motion planning for heterogeneous groups of UAVs which happen to share the same relatively small airspace.
1) We provide a computational formulation for the problem of fair allocation of airspace volume in UAV Traffic Management (UTM). This formulation allows complex missions typical of small UAV applications over urban airspace.
2) We define 3 notions of fairness, including a notion that gives priority to privileged operators. Our framework can accommodate other fairness notions.
3) We provide an algorithm for solving the fair motion planning problem and demonstrate on quadrotor simulations using controllers that have been shown to work on real-life quadrotors.
4) The FairFly framework contributes towards a concrete implementation of equitable access emphasized in the FAA UTM Concept of Operations.

The paper is organized as follows: Section II gives technical preliminaries and presents Fly-by-Logic, on top of which we build our solution. Section III presents the Fair Control problem and our solution to it, and Section IV presents experimental validation.

II. TECHNICAL PRELIMINARIES

Notation. The set of non-negative integers is $\mathbb{N}$. Given a set X and integer n, X^n is its n-fold Cartesian product, and $\mathbf{x}^n$ is an element of X^n, i.e., a sequence of n values from X.
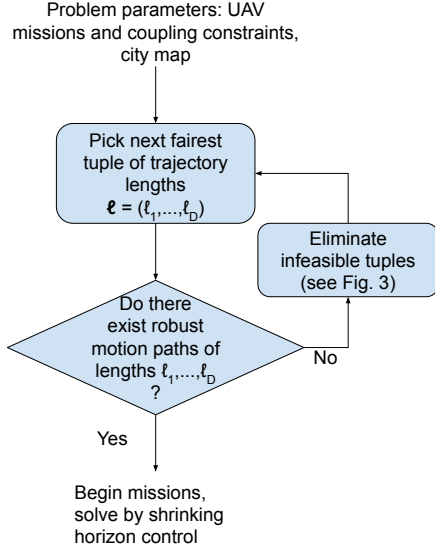
Fig. 1: The FairFly algorithm

System model. Consider a fleet of D UAVs. The nth UAV is modeled as a discrete-time dynamical system $x_n[k+1] = f_n(x_n[k], u_n[k])$, $x_n[0] \in I_0 \subset \mathbb{R}^d$ where d is the dimensionality of the system state. The control input applied to the UAV at time k is $u_n[k] \in U \subset \mathbb{R}^m$ where m is the dimensionality of the input. I_0 denotes the set of possible take-off positions, velocities, accelerations, etc. By concatenating all D states together into a *system state* $x = [x_1, x_2, \ldots, x_D]$ and all inputs into a system input $u = [u_1, \ldots, u_D]$ we get the fleet dynamical system:

$$x[k+1] = f(x[k], u[k]), \quad x[0] \in X_0 \subset \mathbb{R}^{d \cdot D} \quad (1)$$

Here, $X_0 = I_0^D$. Given an initial state $x[0]$ and an input sequence $\mathbf{u}^{H-1} = (u[0], \ldots, u[H-2])$, the corresponding *trajectory* is the sequence $\mathbf{x}^H = (x[0], \ldots, x[H-1])$ of states that satisfy (1). We will sometimes write it as $\mathbf{x}(\mathbf{u}^{H-1})$. Our method applies to nonlinear dynamical systems in general, not only UAVs.

Missions formalization. In our approach, we formalize the complex missions of Urban Air Mobility (UAM) applications as formulas in *Signal Temporal Logic* (STL) [1]. STL can be thought of as Boolean logic with added temporal operators to capture temporal behavior. It allows the succinct and unambiguous specification of a wide variety of complex system behaviors over time [2], [3], [4] and has been used extensively to formalize control objectives, e.g. [4]. Due to space limitations we refer the reader to [1] for formal semantics; we introduce STL via examples. For example, the specification "UAV1 reaches the Park within 10 time steps and avoids obstacles on the way there" is formalized as

$$\phi_1 = \Diamond_{[0,10]}(x_1 \in P) \wedge \Box_{[0,10]}(x_1 \notin Obs)$$

in which $\Diamond$ is the Eventually operator, $\Box$ is Always, and $\wedge$ is Boolean AND. Now suppose there is another UAV in the airspace with mission "UAV2 reaches the bridge within 5 steps and avoids obstacles":

$$\phi_2 = \Diamond_{[0,5]}(x_2 \in B) \wedge \Box_{[0,5]}(x_2 \notin Obs)$$

Although the UAVs are independently operated, they do share the airspace, so we must add a mutual separation formula:

$$\phi_3^c = \Box_{[0,10]}(\|x_1 - x_2\| > s)$$

in which s is a lower bound on the inter-drone separation, set by regulators for example. We call ϕ_3 a *coupling constraint*. In general, a coupling constraint is an STL formula that creates a dependency between the behaviors of 2 or more UAVs. We say the *global mission* is then

$$\phi_g = \phi_1 \wedge \phi_2 \wedge \phi_3^c$$

The specification "UAV2 stays out of Zone 1 *until* UAV3 exits it, which happens in the next 5 steps" is formalized as

$$\phi_4 = (x_2 \notin Z_1)\mathcal{U}_{[0,5]}(x_3 \notin Z_1)$$

in which $\mathcal{U}$ is the Until operator. Interfaces for visualizing formulas [5] and specifying missions [6] have been created.

Definition 2.1: Given D UAVs with their respective missions $\phi_n, 1 \le n \le D$ and N coupling constraints $\phi_n^c, 1 \le n \le N$, the *global fleet mission* is

$$\phi_g = \phi_1 \wedge \ldots \wedge \phi_D \wedge \phi_1^c \wedge \ldots \wedge \phi_N^c \quad (2)$$

Formula horizon. All missions/formulas we work with have a finite *horizon*, i.e., they can be satisfied by a finite-length trajectory $\mathbf{x}^H$. The horizon $H \in \mathbb{N}$ can be calculated from ϕ directly [7], and will be denoted $hrz(\phi)$. For instance, ϕ_1 above has a horizon of 10+1=11 (since we count from 0), and ϕ_2's horizon is 6. Two important remarks are in order:
R1) The horizon H is an *upper bound* on the length of a satisfying trajectory: if all trajectories of length H violate the formula, then there are no satisfying trajectories of any length. A shorter satisfying trajectory might exist. For instance, ϕ_2 has horizon 6, but a length-2 trajectory in which UAV2 reaches its goal at $k = 1$ does satisfy the formula, and indeed is preferable because it's more efficient.
R2) The horizon of the global formula ϕ_g is greater than any one missions's horizon. This can be easily deduced from the definition of $hrz(\cdot)$ function in [7].

Traditional control problem. We must first present the 'traditional' control problem and how Fly-by-Logic [6] solves it, before defining the fair version. Let ϕ_g be the global mission (2) of the UAV fleet with horizon H. The problem is to compute D sequences of inputs of equal length, $\mathbf{u}_1^{H-1}, \ldots, \mathbf{u}_D^{H-1}$, one per UAV, such that the resulting trajectory $\mathbf{x}(\mathbf{u}^{H-1})$ satisfies ϕ_g. Fly-by-Logic finds these input sequences in a centralized fashion by maximizing the *robustness* function ρ_{ϕ_g} over all possible input sequences:

$$\max_{\mathbf{u}^{H-1}} \rho_\phi(\mathbf{x}(\mathbf{u}^{H-1})) \quad (3)$$

where $\mathbf{u}^{H-1} = (\mathbf{u}_1^{H-1}, \ldots, \mathbf{u}_D^{H-1})$. It is shown in [8] that a positive maximum implies that the corresponding trajectory $\mathbf{x}(\mathbf{u}^{H-1})$ satisfies ϕ_g. See [6] for details about robustness and the maximization problem.

We immediately note that Fly-by-Logic only searches over length-H trajectories (equivalently, length-$(H-1)$ input sequences), even though their own missions may have shorter horizons by remark R2. And as noted above in R1, this also means that UAVs are potentially forced to fly longer, and solve larger control problems online, than they strictly need to.

III. FAIR CONTROL PROBLEM AND SOLUTION

A. Problem definition

Consider D trajectories $\mathbf{x}^{\ell_1}, \ldots, \mathbf{x}^{\ell_D}$, one per UAV, such that collectively they satisfy the global mission ϕ_g. We understand fairness of the trajectories $\{\mathbf{x}^{\ell_n}\}_{n=1}^D$ as being a notion defined only on the lengths tuple $\langle \ell_1, \ldots, \ell_D \rangle := \vec{\ell}$: this is about how early or late each UAV completes its mission, not about how they fly to do so. (Trajectory length can also serve as proxy for energy consumption). For instance, consider a 3-UAV fleet with $hrz(\phi_g)=10$, and two possible length tuples: $\vec{\ell} = \langle 8, 8, 8 \rangle$ and $\vec{\ell}' = \langle 2, 9, 10 \rangle$. We want to say that $\vec{\ell}'$ is less fair than $\vec{\ell}$, because it lets UAV1 finish early and forces UAVs 2 and 3 to finish at the limit of what's possible. We can perform this reasoning purely by looking at the lengths. Thus, fairness will be a function f that maps a length tuple $\vec{\ell}$ to a real number such that a larger f-value implies greater fairness.

It thus emerges that we need a way to determine *which* length tuples to consider and rank by fairness. For instance, we can now see that Fly-by-Logic only considers $\langle hrz(\phi_g), \ldots, hrz(\phi_g) \rangle$. But as noted in R1, shorter satisfying trajectories may be possible. Thus, we need to build a set $PL(\phi)$ of *promising lengths*.

Definition 3.1: i) Let ϕ be a one-UAV formula. Then the *promising lengths set $PL(\phi)$* of ϕ is the set of integer lengths ℓ such that there exists a sequence $\mathbf{x}^\ell \in X^\ell$ that satisfies ϕ.

ii) Let ϕ_g be a D-UAVs formula. Then the *promising lengths set $PL(\phi_g)$* of ϕ_g is the set of D-tuples $\vec{\ell} = \langle \ell_1, \ldots, \ell_D \rangle$ such that there exist D sequences $\{\mathbf{x}_n^{\ell_n}\}_{n=1}^D$, $\mathbf{x}_n^{\ell_n} \in X^{\ell_n}$, which collectively satisfy ϕ_g.

Note that whether the sequence $\mathbf{x}^\ell$ can be flown by the UAV - i.e., whether it's a system trajectory or not - remains to be determined.

Proposition 3.1: It is possible to compute an over-approximation of $PL(\phi)$ using a recursion on the structure of ϕ. ∎

Thus, because computing PL only requires knowledge of the formula, it can be computed *offline* by the central planner. All proofs in this paper are omitted in the interest of space.

We now define the *Fair Temporal Logic Control Problem*:

Problem 1: Consider a global formula ϕ_g over D UAVs, a fairness function f, and an initial state $x_0 \in X_0$. The fair

control problem is

$$\max_{\vec{\ell} \in PL(\phi_g)} f(\vec{\ell}) \tag{4a}$$

$$\text{s.t.} \quad \max_{\mathbf{u}^{\vec{\ell}}} \rho_{\phi_g}(\mathbf{x}(\mathbf{u}^{\vec{\ell}})) \geq 0 \tag{4b}$$

$$\text{s.t.} \quad x(0) = x_0$$

$\mathbf{u}^{\vec{\ell}}$ is short for the set of input sequences $\{\mathbf{u}_n^{\ell_n}\}_{n=1}^D$. This is a bi-level optimization: the constraints require solving an *inner optimization* (4b) dependent on the primary decision variable $\vec{\ell}$. The bi-level optimization is only necessary to be solved in an offline phase. We will say that $\vec{\ell}$ is *feasible* if the corresponding inner optimization has positive solution. Thus by solving (4) we seek the fairest feasible tuple of trajectory lengths. We show how to solve (4) in Section III-C.

Remark 3.1: The dimension of the inner optimization, and therefore the time to solve it, is proportional to $\sum_n \ell_n$. For Fly-by-Logic, $\ell_n = hrz(\phi_g)$ for all n, so the dimension is proportional to $D \cdot hrz(\phi_g)$.

B. Fairness functions

How to measure fairness? We recognize that there is no 'best' notion of fairness, and the results from any choice should be interpreted in light of the application. We present three fairness function candidates; our framework can accommodate other application-appropriate functions.

Intuitively, every UAV has a range of promising lengths, which is determined by its own mission but also by the coupling constraints that tie it to other UAVs. It is preferable for a UAV to be as close as possible to the lower end of this range, but this might force other UAVs towards the upper ends of their ranges. *Thus, for fairness, all UAV trajectory lengths will lie roughly around the same point in their respective ranges.* This intuition is formalized as follows. Given a D-tuple $\vec{\ell} \in PL(\phi_g)$, we write $\vec{\ell}(n)$ for the nth element of $\vec{\ell}$. Define

$$\underline{\ell_n} = \min\{\vec{\ell}(n) \mid \vec{\ell} \in PL(\phi_g)\}, \quad \overline{\ell_n} = \max\{\vec{\ell}(n) \mid \vec{\ell} \in PL(\phi_g)\}$$

$$\alpha_n = (\ell_n - \underline{\ell_n})/(\overline{\ell_n} - \underline{\ell_n})$$

So $\{\underline{\ell_n}, \ldots, \overline{\ell_n}\}$ is the range of lengths that $\vec{\ell}(n)$ can take in $PL(\phi_g)$, and α_n measures the fraction of that range at which a given ℓ_n lies. Note that α_n is a monotone function of ℓ_n. Given $\vec{\ell}$ and the corresponding tuple of fractions $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_n \rangle$, the first fairness function is simply the negative of the variance of the α_n's:

$$f_1(\vec{\ell}) = -\text{var}(\vec{\alpha}) \tag{5}$$

(we use negative variance because we want to maximize fairness).

f_1 is a negative function with maximum 0, achieved when all α_n's are equal, i.e. when all UAVs trajectory lengths are exactly at the same fraction in their length ranges $\{\underline{\ell_n}, \overline{\ell_n}\}$.

Now all else being equal, a solution where every $\alpha_n = 1/2$, say, is preferable to a solution where every $\alpha_n = 1$, since

a smaller α means a shorter and more efficient trajectory. Moreover, if a (3-UAV) solution with $\vec{\alpha} = \langle 3/4, 3/4, 1/2 \rangle$ is feasible, forcing a solution with $\vec{\alpha} = \langle 3/4, 3/4, 1 \rangle$ seems unfair - after all, a shorter trajectory for the 3rd UAV is not forcing longer trajectories for the others. The second fairness function captures this by adding a regularizer to f_1 which encourages small lengths, thus balancing between balance and efficiency:

$$f_2(\vec{\ell}; w) = -w\text{var}(\vec{\alpha}) - (1-w)\sum_{n=1}^{D} \alpha_n^2 \qquad (6)$$

Here, $w \in (0,1]$ is a weighting factor.

The last fairness function allows for explicit negotiation between operators for preferential treatment: this is a legitimate use-case, in which some operators pay to choose trajectory lengths that suit them, unconstrained by fairness considerations. Given a weight tuple $\vec{v} \in (0,\infty)^D$, define

$$f_2^{imb}(\vec{\ell}; w, \vec{v}) = -w\text{var}(\vec{\alpha}) - (1-w)\sum_{n=1}^{D} v_n \alpha_n^2 \qquad (7)$$

As v_n grows larger, α_n gets valued more in the fairness weighting, thus favoring UAV$_n$.

C. Solving the Fair Control Problem

The control problem is solved in an offline and online phases. Offline, we find one solution to (4), which is a promising length tuple $\vec{\ell}^* = \langle \ell_1^*, \dots, \ell_n^* \rangle$ and satisfying trajectory $\mathbf{x}^*$ in which UAV$_n$ has a trajectory of length ℓ_n^*. Online, i.e. after take-off, a classical shrinking horizon procedure is implemented to continuously update the trajectory $\mathbf{x}^*$ based on the latest state estimate. See [6] for details. Because the online phase is a special case of the offline phase, we focus on the latter.

The offline phase could be solved as follows. See Fig. 1. The set $PL(\phi_g)$ is finite: start with a fairest promising length tuple (one which maximizes $f(\vec{\ell})$) and check whether it is feasible - i.e., whether the corresponding inner maximization has positive maximum. If yes, we are done. Else, we pick the next fairest promising tuple $\vec{\ell}$ and repeat, until we find the fairest promising length whose inner optimization has positive robustness using Fly-by-Logic.

This brute force approach is only possible for small numbers of UAVs and small horizons since the size of PL grows as $O(H^D)$. For larger D or H it is not even possible to store PL in memory. Therefore, in our implementation of FairFly, we don't build PL in memory, we use instead an implicit representation. The search is made more efficient by the following proposition.

Proposition 3.2: (a) If $\vec{\ell}$ is infeasible, then every smaller $\vec{\ell}$ in lexicographic order is also infeasible.
(b) If $\vec{\ell}$ is feasible, then the optimizer of (4) has fairness at least $f(\vec{\ell})$. ∎
This proposition allows us to reduce the search space of (4a) with every iteration. See Fig. 2: Prop. 3.2(a) says that every infeasible promising tuple allows us to eliminate from
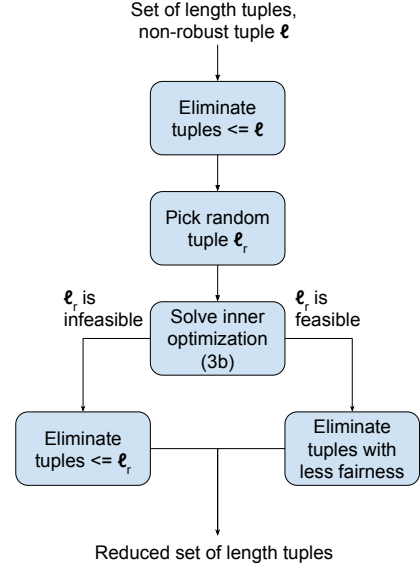

Fig. 2: Sound reduction of search space for outer optimization (4a). This is step "Eliminate infeasible tuples" in Fig. 1.

consideration all smaller tuples in one go. Therefore we store every infeasible tuple we encounter to check whether future tuples are smaller than it; if yes, we skip them without wasting time solving the inner optimization for them, which is the real computational bottleneck.

Prop. 3.2(b) allows us to eliminate tuples that are less fair than feasible tuples we encounter. Therefore, we occasionally randomly sample the set PL as shown in Fig. 2: if $\vec{\ell}_r$ is feasible, we store it and compare future tuples to it. If they are less fair, we skip them. Note that as soon as an elimination takes place the remaining search space becomes non-convex (as a compact subset of $\mathbb{R}^D$). Therefore, the step to pick the next fairest tuple in Fig. 1 only yields local optima.

IV. Experiments

We implemented our solution, called FairFly, on top of Fly-by-Logic, a toolbox for motion planning and control of quadrotor fleets [9], [6]. This is implemented by expanding Fly-by-Logic to solve using distinct horizons for each UAV instead of a global horizon, and then implementing the outer optimization. We compare the solutions provided by default Fly-by-Logic (without fairness considerations) and FairFly. All simulations were run with an Intel CPU at 2.60 GHz on a single core.

A. The effects of fairness

We report the results of quadrotor fleet simulations for Reach-Avoid missions. The Reach-Avoid formula for D
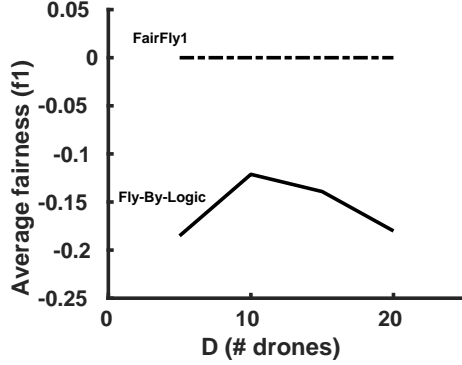
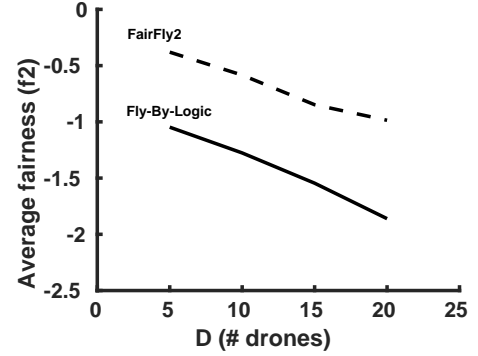Fig. 3: Average fairness of solutions found by Fly-by-Logic (solid) and FairFly (dash dotted) using f_1.

Fig. 4: Average fairness of solutions found by Fly-by-Logic (solid) and FairFly (dashed) using f_2.

quadrotors is

$$\phi = \phi_G \wedge \phi_O \wedge \phi_M$$

$$\phi_G = \bigwedge_{n=1}^{D} \diamondsuit_{[0,H_n]}(x_n \in G_n)$$

$$\phi_O = \bigwedge_{n=1}^{D} \square_{[0,H_n]}(x_n \notin O) \qquad (8)$$

$$\phi_M = \bigwedge_{n,m=1,n\neq m}^{D} \square_{[0,H_{n,m}]}(\|x_n - x_m\| > d)$$

in which H_n is the horizon of UAV$_n$'s mission, G_n is the goal of UAV$_n$, O is the set of obstacles that all UAVs are avoiding, d is the minimum inter-drone distance, and $H_{n,m} = \min(H_n, H_m)$ (UAV$_n$ and UAV$_m$ need to avoid each other only for as long as both are flying). $\bigwedge_{n=1}^{D} \phi_n$ represents the conjunction of every ϕ_n for $1 \leq n \leq D$

We ran experiments with $D = 5, 10, 15$ and 20 quadrotors. For each value of D, we ran three algorithms to solve (4): default Fly-by-Logic and FairFly using both f_1 and f_2. For each algorithm, and to have meaningful results, we solve the fair control problem (4) 20 times, each time starting from a different initial state x_0. The same map is used for each iteration of the experiment. The map is left simple with a single obstacle that the UAVs must avoid, and the goals of each UAV are held constant.

We compared: a) the average robustness of the three solutions, b) the average fairness of the three solutions, c) the average time is takes to solve the offline phase, and d) the average time to complete the first iteration of the online control phase (subsequent iterations take less time). All averages are over the 20 initial states.

Fairness Results. See Figs. 3 and 4. Using either fairness notion, the solutions obtained by FairFly are more fair than ones that Fly-by-Logic outputs for any number of UAVs. With f_1, the fairest solution is when all α's are equal. The maximizer that gets chosen by FairFly is when all α's are equal to 1, or equivalently $\vec{\ell} = \langle H_1, H_2, ...H_D \rangle$. Note that if this length tuple is not feasible, then no other tuple can be, so only a maximum fairness solution needs to be checked.
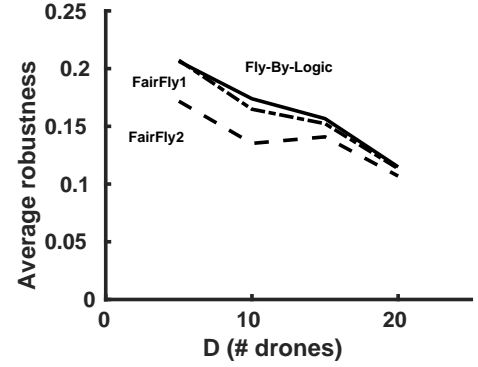

Fig. 5: Average robustness of solutions found by Fly-by-Logic (solid) and FairFly with f_1 (dash dotted), and f_2 (dashed)

With f_2, the fairness value declines as more UAVs are added. This is expected: with more drones the airspace is more densely occupied, requiring more compromises to the trajectories of the quadrotors, thus increasing the impact of the efficiency factor in f_2.

Effects on robustness. Fig. 5 shows the robustness of the optimal trajectories returned by Fly-by-Logic and FairFly using both f_1 and f_2. Using f_1 is slightly less robust on average than Fly-by-Logic, and f_2 is less robust than f_1. It is noted that the reduction in robustness is not dramatic, and more importantly, robust trajectories still exist while increasing fairness.

Offline computational overhead. Now we examine the computational overhead of finding fair trajectories. See Fig. 6. FairFly with f_1 is actually quicker offline than Fly-by-Logic. The explanation is this: as noted above, an f_1-fairest trajectory tuple is equal to the individual horizon of each UAV, i.e. $\vec{\ell} = \langle H_1, \ldots, H_D \rangle$. If this tuple is not feasible, than no other tuple can be feasible, so only this one length tuple needs to be checked for feasibility. By Remark 3.1, the size of the inner optimization for FairFly is smaller than that of Fly-by-Logic, so it is quicker to solve.

FairFly calculated with f_2 was slightly slower than Fly-by-Logic. This is while acknowledging the fact that FairFly was not necessarily producing the globally f_2-fairest result because it is using a non-convex optimization to find the next
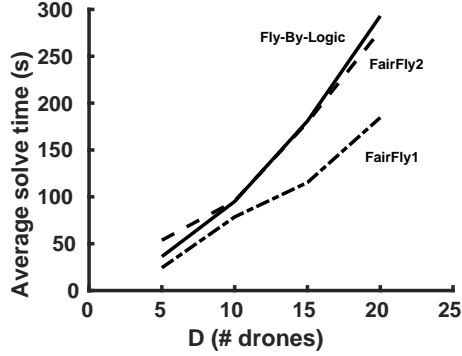
Fig. 6: Time to solve the offline optimization for Fly-by-Logic (solid) and FairFly with f_1 (dash dotted), and f_2 (dashed), averaged over choice of initial states.
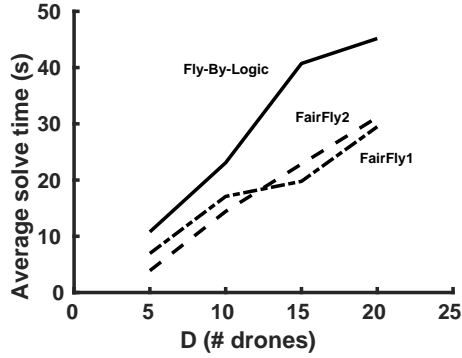

Fig. 7: Time to solve first iteration of the online control problem in shrinking horizon fashion for Fly-by-Logic (solid) and FairFly with f_1 (dash dotted), and f_2 (dashed), averaged over choice of initial states.

fairest length tuple.

Online computational gains. We now study the online run time performance of FairFly, as this will directly impact online control stability of the UAVs. The offline phase yields a fairest feasible length tuple $\vec{\ell}^*$ and a corresponding most robust trajectory $\mathbf{x}^*$. In the first iteration of the online phase the first control action from $\mathbf{x}^*$ is applied. At the second iteration, the inner maximization is re-solved with one less time-step (since we already did one step). More generally, at the kth iteration, the inner maximization is re-solved with k fewer time-steps (since we already did k steps). Therefore, the horizons that the optimization must solve for shrink every time step.

For our analysis we simply looked at the run time for the first iteration of the online phase. See Fig. 7. The FairFly variations greatly distinguish themselves from Fly-by-Logic here. The run time of FairFly is about 50% faster than Fly-by-Logic with f_2 being fastest on average. This is expected; f_2 is solving for the shortest overall trajectories due to the length regularizer term (see Fig. 6). Because the length of trajectories is shorter in f_2, the overall search space is reduced for the inner optimization.

B. Comparing the fairness functions

Now we will look at the length trajectories that each fairness function generates, in order to conceptualize what each function considers as "fairest".

To compare the fairness functions f_1 and f_2, we ran a $D = 5$ Reach-Avoid experiment with individual horizons $H_1 = 10, H_2 = 8, H_3 = 5, H_4 = 6, H_5 = 7$ (see (8)). With f_1, the fairest length tuple is $\vec{\ell} = \langle 10, 8, 5, 6, 7 \rangle$. With $f_2(\cdot; w = 0.75)$ the fairest length tuple is $\vec{\ell} = \langle 9, 7, 5, 6, 6 \rangle$. It is noted that with f_2 every UAV finishes at least as fast as it would have with f_1, and some can finish quicker. With f_1, every UAV travels a mission that is the longest allowed by their horizon. No UAV is allowed to complete their mission any quicker, but neither are any of the UAVs required to continue flying until the rest finish their mission.

We then ran a case where UAVs negotiate an imbalanced solution to compare our third notion of fairness, as shown in (7). We ran the same $D = 5$ experiment as above with f_2^{imb} with $v_1 = v_2 = 10$, and $v_3 = v_4 = v_5 = 1$. This gives advantage to the first two quadrotors at the expense of the last three. The fairest length vector is $\vec{\ell} = \langle 7, 6, 5, 6, 7 \rangle$, compared to the solution from f_2, namely $\langle 9, 7, 5, 6, 6 \rangle$. As expected, the first two UAVs received shorter trajectories, while the last three had the same or longer trajectories.

V. Conclusions

FairFly was shown to produce both fairer and more efficient trajectories for UAVs with a slight impact to trajectory robustness. Future work will focus on faster offline optimization and hardware implementations.

References

[1] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in Proceedings of the International Conference on Formal Modeling and Analysis of Timed Systems, 2010.

[2] H. Abbas and G. Fainekos, "Computing descent direction of MTL robustness for non-linear systems," in American Control Conference, 2013.

[3] M. Hekmatnejad and G. Fainekos, "Optimal multi-valued ltl planning for systems with access right levels," in 2018 Annual American Control Conference (ACC), 2018, pp. 2363–2370.

[4] L. Lindemann, J. Nowak, L. Schnbchler, M. Guo, J. Tumova, and D. V. Dimarogonas, "Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks," IEEE Transactions on Control Systems Technology, pp. 1–8, 2019.

[5] B. Hoxha, H. Bach, H. Abbas, A. Dokhanchi, and Y. Kobayashi, "Towards formal specification visualization for testing and monitoring of cyber-physical systems," in In Int. Workshop on Design and Implementation of Formal Tools and Systems, 2014.

[6] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: Control of multi-drone fleets with temporal logic objectives," in Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems. IEEE Press, 2018.

[7] A. Dokhanchi, B. Hoxha, and G. Fainekos, "Online monitoring for temporal logic robustness," in Proc. of Runtime Verification, 2014.

[8] G. Fainekos and G. Pappas, "Robustness of temporal logic specifications for continuous-time signals," Theoretical Computer Science, 2009.

[9] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in Control Technology and Applications (CCTA), 2017 IEEE Conference on. IEEE, 2017.