Game Theory-based Parameter-Tuning for Path Planning of UAVs

Diksha Moolchandani*, Geesara Prathap§, Ilya Afanasyev§, Anshul Kumar*, Manuel Mazzara§, and Smruti R. Sarangi*†

*Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
New Delhi, India
*{diksha.moolchandani, anshul, srsarangi}@cse.iitd.ac.in

§Department of Computer Science
Innopolis University
Innopolis, Republic of Tatarstan, Russia
§{g.mudiyanselage, i.afanasyev, m.mazzara}@innopolis.ru

***Abstract*—Modern UAVs are incredibly complex systems with numerous tunable knobs such as the battery capacity, camera settings, sampling rate, constraints on the route, etc. The area of theoretical exploration of the optimization problems that arise in such settings is dominated by traditional approaches that use regular nonlinear optimization often enhanced with AI-based techniques such as genetic algorithms. These techniques are sadly rather slow, have convergence issues, and are typically not suitable for use at runtime. In this paper, we leverage recent and promising research results that propose to convert the optimization problem into a *game* and then find the set of equilibrium strategies of different players. The strategies can then be mapped to the optimal values of the tunable parameters. With simulation studies in virtual worlds, we show that our solutions are 5-21% better than those produced by traditional methods, and our approach is 10-100 times faster.**

I. Introduction

With the growing demand of unmanned aerial vehicles (UAVs) for a multitude of present-day applications, there has been an upsurge in the global UAV market. The UAV market that was worth \$2.67 billion in 2016 is expected to reach \$10.28 billion by 2022 with a CAGR (Compound Annual Growth Rate) of 25.2% [1]. According to the U.S. Federal Aviation Administration (FAA), the number of UAVs registered as of 2019 was 1.1 million and this number is expected to exceed 4 million by 2021 [2].

Unfortunately, the software and computing aspects of a UAV have not been given adequate importance in the literature. Boroujerdian et al. [3] have shown that with the wrong choice of the algorithm or its parameters, it is possible to compute routes that take four times longer to traverse. This is a wastage of time as well as battery power. In general, in UAVs, path planning is regarded as the slowest operation [4]. The algorithms for path planning are well established, however their behavior is governed by a large number of hyper-parameters that need to be set based on runtime conditions. Hence, tuning a path planning algorithm is disproportionately important in UAV design. The state of the art uses a combination of traditional optimization and AI enhanced algorithms to find the right set of input parameters. We argue in this paper that these are very slow approaches and cannot be carried out at runtime without incurring significant penalties.

We instead propose a very different approach that actually uses game theory. Even though an optimization problem of this kind is not a traditional game with payoffs and players; however, we can use some of the insights from latest research to very quickly solve an optimization problem by first converting it into a game [5]. We thus propose a novel approach to actually model a traditional optimization problem that involves finding the hyper-parameters for a UAV path planning algorithm by actually finding the equilibrium strategies of different players in an equivalent game [6], [7]. Our game-theoretic approach is 10-100 times faster than the optimization based approach, and our solutions are 5-21% better than the latter approach.

The novel contributions of this paper are as follows.

1) We perform simulations and model the path length and hover time of a UAV as a function of the hyper-parameters using a multi-layered perceptron model.
2) We propose a game-theoretic approach to formulate and solve the resultant optimization problem that minimizes the overall time taken by the UAV to reach its destination. We define novel payoff functions to incorporate the selfish and altruistic objectives of the players.
3) We derive interesting insights from the observed Nash equilibria.
4) We show that our game-theoretic approach is 10-100X faster and provides solutions that are 5-21% better than the best optimization based approaches for three different virtual worlds.

II. Background

A. Navigation in UAVs

The problem of navigation in UAVs has been targeted by both classical geometric methods and end-to-end learning-based methods. However, for the case of UAVs, the classical methods are still the most popular choice [3], [8] because of their simplicity and deterministic approach. These methods follow three basic steps: ❶ **perception:** building a 3D view of the surroundings, and extracting the information in the form of an obstacle/occupancy map, ❷ **planning:** using the information of the obstacles from the perception step to create a collision-free path, and ❸ **control:** sending the control commands to the UAV to follow the planned path. This is referred to as the Perception, Planning, and Control (PPC) paradigm.

We focus on the path planning step because it is the most time consuming step in the entire pipeline (roughly 65% [4]). Yang et al. [9] suggested that among all the path planning algorithms with bounded time complexity, the sampling-based algorithms are self-sufficient in determining an optimal path. We choose the most popular sampling based algorithm, RRT*, for this work, which is known to provide near-optimal solutions and allows re-planning.

The **RRT*** algorithm builds a path from the source to the destination in the form of a tree. To grow the tree, the algorithm first *samples* the environment (*dimension:* $x \times y \times z$) and chooses a random point (p_{rand}). Subsequently, the node A in the tree that is the closest to p_{rand} attempts to create a new node in the direction of p_{rand}. We assume a step size (resolution) in the algorithm that restricts the maximum distance (from A) at which the new node can be placed.

† Smruti R. Sarangi is currently the Usha Hasteer Chair Associate Professor in the department of Computer Science and Engineering, and holds a joint appointment with the department of Electrical Engineering.

Thus, a new node B is placed at step-size units away from A in the direction of p_{rand}. Node B is added to the vertex set of the tree if the direct path from A to B is free of obstacles. After B is added to the vertex set, it needs to connect to some vertices via edges to become a part of the tree. Instead of directly creating an edge from A to B, we create edges using the notion of a cost function. Each vertex in the tree has an associated cost that quantifies the cost of reaching that vertex from the start (root) node. To connect B, all the nodes within a radius r are checked. If any node C from this neighborhood has a path to B that is of lower cost as compared to the cost of the path from A to B, then an edge is created between B and C in place of the edge between A and B. Needless to say, the edge between C and B should be free of obstacles. The minimum distance between an edge and an obstacle should be at least equal to the *obstacle avoidance distance* – this avoids collisions even if there is a slight amount of nondeterminism in the UAV's position.

The number of samples, step-size, obstacle avoidance distance, and the dimensions of the environment form the hyperparameters of the RRT* algorithm, which determine the behavior of the algorithm. In this work, we perform experiments to identify this behavior and develop a game-theoretic framework to predict this behavior at runtime.

B. Game Theory Preliminaries

In a game-theoretic system, there are multiple selfish yet rational players. Each player has a strategy, which it plays to maximize its chances of winning the game. The notion of winning the game is captured by the payoff or utility that a player derives by playing a certain strategy. Thus, for a combination of strategies across the players, each competing player obtains a payoff.

There is no notion of optimality here, because fundamentally the players are at odds with each other. Hence, we define the notion of a *Nash equilibrium* instead, where no player can increase its payoff by unilaterally changing its strategy (the rest of the strategies remaining the same). The notion of a *Nash equilibrium* is very powerful in describing the results of games, and it is often possible to derive profound insights about the inherent trade-offs and feasible solutions. A Nash equilibrium is said to be *stable* if a small change in the strategy for any player makes it strictly worse off. The strategies should be independent, implying that the players can independently choose their strategies regardless of the strategy of other players.

III. Experimental Setup

A. Overview

In this work, we formulate an optimization problem where our aim is to find a relationship between the parameters of RRT* such that the total time to reach the destination is minimized. Due to the complex relationship between the parameters (see Table I) and the uncertain nature of sampling-based algorithms such as RRT*, we need to collect a huge amount of data for multiple environments to formulate a master equation for the optimization problem. This process is extremely time-consuming if done in a real outdoor setting, thus we perform exhaustive simulations in an open-source robotics simulator, Gazebo, for multiple configurations and virtual worlds.

Since the convergence of the formulated optimization problem takes a lot of time to solve, we map it to a game-theoretic framework where the independent parameters are the players and the dependent parameters are used to formulate the payoffs of the players. We, in effect, perform sensitivity analyses of the dependent parameters with respect to the independent parameters. Once the payoffs of the players are formulated using the sensitivity results, we use Gambit-v15.1.1 [10] along with its Python API to calculate the Nash equilibria of the game.

B. Setup for Sensitivity Analyses

TABLE I
Tunable Parameters

Parameter	**Description**	**Range**
Dimension size (dim)	Dimension of the search space for the UAV (physical 3D dimensions)	$5 \times 5 \times 5 - 40 \times 40 \times 40$ m^3
#samples (sam)	Number of random samples for RRT*	$100 - 5000$
Step size (res) of RRT*	Minimum step length that can be taken in the direction of the chosen random sample	$0.01 - 10$ m
Obstacle avoidance distance (obs_av)	Minimum distance that should be maintained between the nearest obstacle and the calculated path	$0.1 - 0.5$ m

These values are obtained as feasible ranges from the experiments. Feasible values are those that do not degrade the accuracy significantly.

We use an NVIDIA Xavier board (state-of-the-art board for autonomous machines such as UAVs) for the sensitivity analyses. It consists of 8 ARMv8.2 cores having a frequency of 2.26 GHz, main memory of 16 GB, 8 MB L2 cache, 4 MB L3 cache, and a 512-core NVIDIA Volta GPU with 64 Tensor cores. We emulate the path planning algorithm on the NVIDIA Xavier board to get the sensitivity results. The algorithm runs within the Robot Operating System (ROS). ROS allows running multiple concurrent processes, also called nodes. These nodes pass data between each other using non-blocking FIFO queues [3]. ROS uses the publish-subscribe model where some nodes publish messages while other nodes receive the messages by subscribing to the publishing nodes. The messages can belong to some specific categories, also called *ROS topics*. For the sensitivity analyses, we kept the simulation environment the same. We primarily relied on prerecorded messages for simulation that were stored in a *ROS bag* file during the outdoor flight of the UAV. *ROS bag* files help us realize a deterministic simulation by replaying the prerecorded messages.

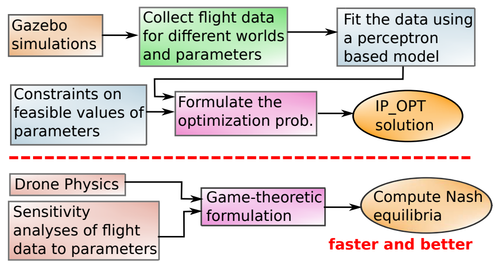
Gazebo simulations
Collect flight data for different worlds and parameters
Fit the data using a perceptron based model
Constraints on feasible values of parameters
Formulate the optimization prob.
IP_OPT solution
Drone Physics
Sensitivity analyses of flight data to parameters
Game-theoretic formulation
Compute Nash equilibria
faster and better

Fig. 4. Overview of our approach

C. Gazebo and Rviz for UAV Simulation

To collect the data for formulating the optimization problem, we simulate (as explained above) the virtual world and the UAV in Gazebo-9. Gazebo takes in a *world* file that specifies the environment along with a map of the obstacles, and the UAV's specifications. After the planning step, the control commands are sent to the Gazebo engine to make the UAV move in the desired direction. We use Rviz-v1.13.13 for visualizing the surroundings in the form of an occupancy map. Rviz is a ROS graphical interface that allows us to visualize the position, orientation of the UAV, and the locations of obstacles. For creating the virtual Gazebo worlds, we have used the standard tools that have been used in most prior work [11], [12].

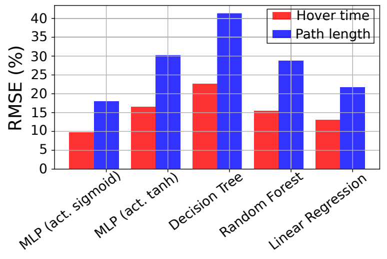

Fig. 1. RMSE comparison of different techniques for World 1

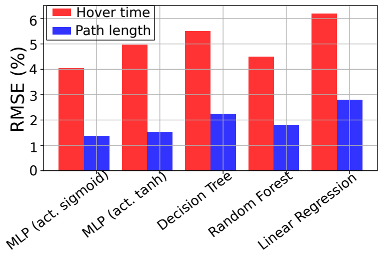

Fig. 2. RMSE comparison of different techniques for World 2

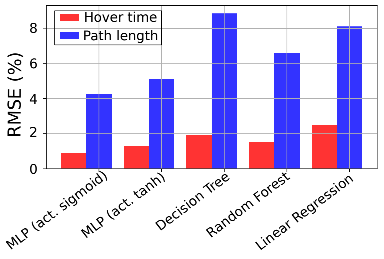

Fig. 3. RMSE comparison of different techniques for World 3

D. IPOPT, and AMPL

In order to solve the nonlinear optimization problem formulated using the data collected from Gazebo, we used a nonlinear solver called *IPOPT-v3.12.13* [13]. It uses an algebraic modeling language, *AMPL*, to model the objective function and the constraints of the optimization problem. Figure 4 shows an overview of the formulation of the optimization problem and the game-theoretic approach. These are discussed in detail in Sections IV and V.

IV. Formulation of the Optimization Problem

As explained in Section III-A, we need to perform exhaustive simulations in Gazebo to collect multiple data points corresponding to different configurations and different virtual worlds. Subsequently, we use ML based techniques to fit a curve on these data points. The equation for the curve is then used to formulate the constraints of the optimization problem. The objective of the formulation is to minimize the total time taken to reach the destination (see Section III-A). The total time is defined as the sum of the hover time and the flight time. Here, hover time (HT) is the time elapsed before a decision is made by the planner. The UAV hovers at the current position during this time and is not doing any useful work. The flight time (FT) is defined as the time taken to reach the *destination position*. It is roughly proportional to the distance travelled, also called the path length (PL). Thus, in order to minimize the time taken to reach the destination, we need to minimize either HT or FT or both. Let us discuss these steps in detail.

A. Collection of Data Points

The idea is to collect the flight data (HT and PL) for multiple virtual worlds from Gazebo for varying configurations of the tunable parameters (shown in Table I) of the RRT* based path planning algorithm. We performed simulations for all possible configurations of the independent parameters in their feasible ranges. We collected 300 data points for each virtual world. A data point is an n-tuple of the parameter configuration (res, sam, dim, obs_av), input environment (obs_den), HT, and PL. The input environment is captured in terms of the obstacle density (obs_den) of the virtual world. For each simulation, the start and the end points were kept the same so that the effect of changing configurations can be accurately captured.

B. Curve Fitting

The configuration parameters and the obstacle density together form the feature vector of these datapoints. To get the exact dependence of the flight data (HT and PL) on the feature vector, we performed curve fitting. The curve fitting problem takes in the collected data points as its input and provides a trained predictor model as the output. The equation for this model is the fitted curve that provides a relationship of the flight data with the configuration parameters and the input environment. Here the idea is to consider 80% of the collected data points and fit the curve using these points. We use the remaining 20% of the data points as the test points. We use the root mean square error (RMSE) metric to quantify the error of prediction.

All the data points are normalized using min-max scaling. Since two values (HT and PL) need to be predicted for each feature vector, this is a multi-output regression problem. Figures 1, 2, and 3 show the comparison of five different learning techniques for the three different virtual worlds. We achieve a lower root mean square error (RMSE) for the curve derived from the multi-layer perceptron (MLP) algorithm with a Sigmoid-based activation function and 2 hidden layers, each having 10 neurons. MLP performs better than all the competing techniques because the relationship is complex and it is difficult to predict using the simple linear regression or decision tree based models. It can be observed that the RMSE (%) is higher in the first virtual world as compared to the other two virtual worlds because its average HT is lower as compared to the other two worlds. This is mainly due to a lower obstacle density in the first world.

C. Formulation of the Optimization Problem

$$\begin{aligned} minimize \quad Totaltime &= HT + FT \\ &= HT + \zeta * PL \end{aligned} \tag{1}$$

$$\begin{aligned} \text{s.t.} \quad h1[i] &= \sum_{j=1}^{5} w_1^\intercal[i,j] * v[j] + b1[i], \forall i \in [1,10] \\ h1o[i] &= 1/(1+e^{-h1[i]}), \forall i \in [1,10] \\ h2[i] &= \sum_{j=1}^{10} w_2^\intercal[i,j] * h1o[j] + b2[i], \forall i \in [1,10] \\ h2o[i] &= 1/(1+e^{-h2[i]}), \forall i \in [1,10] \\ h3[i] &= \sum_{j=1}^{10} w_3^\intercal[i,j] * h2o[j] + b3[i], \forall i \in [1,2] \end{aligned} \tag{2}$$

Our objective is to minimize the total time taken by a UAV to reach the destination (see Equation 1). Here the total time is written as a sum of HT and FT, where FT is expressed in terms of PL. The curve derived from the MLP formulation provides the equations for the constraints as shown in Equation 2. Here $h1, h2, h3, h1o, h2o$ are the neurons in the hidden layers of the MLP. We use two hidden layers and correspondingly three weight matrices w_1, w_2, w_3 and three bias vectors $b1, b2, b3$ from the input layer to $h1$, $h1o$ to $h2$, and $h2o$ to $h3$, respectively, where $h3$ is passed to the Sigmoid layer to give the outputs HT and PL. The input to the MLP is the feature vector v that captures the configuration parameters and the input environment. Here, $v = \langle res, sam, dim, obs\_av, obs\_den \rangle$.

The process for deriving the curve from the parameters of the MLP is as follows. In Equation 2, the first constraint for the first hidden

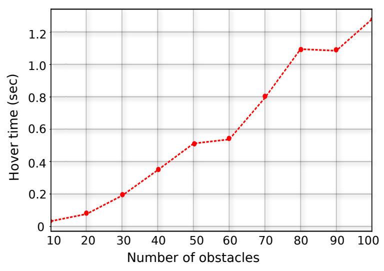

Fig. 5. No. of obstacles v/s hover time

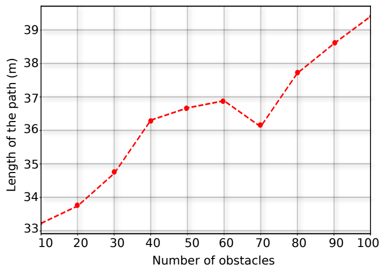

Fig. 6. No. of obstacles v/s path length

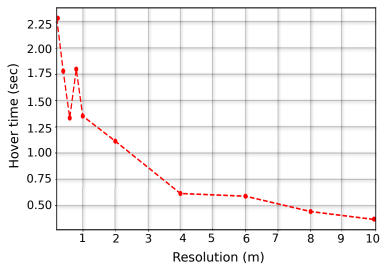

Fig. 7. Resolution of the path v/s hover time

layer $h1$ is the sum of the product of the weights (w_1) learned from curve fitting and the feature vector (v). A bias term $b1$ is also added for each neuron ($h1[i]$) of the hidden layer ($h1$). The second equation for $h1o$ introduces a nonlinearity in the equation using the Sigmoid activation function. The equations for the second hidden layer ($h2$) are obtained similarly.

$$\begin{gathered}
1/(1+e^{-h3[1]}) - \xi_h \leq HT \leq 1/(1+e^{-h3[1]}) + \xi_h \\
1/(1+e^{-h3[2]}) - \xi_p \leq PL \leq 1/(1+e^{-h3[2]}) + \xi_p \\
1 \leq HT \leq FT/2 \\
dist(start, dest.) \leq PL \leq 2 * dist(start, dest.) \\
0.01 \leq v[1] \leq 10 \\
100 \leq v[2] \leq 5000 \\
dist(start, dest.) \leq v[3] \leq 5 * dist(start, dest.) \\
0.1 \leq v[4] \leq 0.5
\end{gathered} \tag{3}$$

The output of the second hidden layer is multiplied by its corresponding weights and passed through the Sigmoid layer to generate the constraints on HT and PL. The equations for HT and PL capture the final regression output along with the error margins: ξ_h and ξ_p (see Equation 3). These two equations are expanded using the Taylor Series expansion of e^{-y}, which is $1 - \frac{y}{1!} + \frac{y^2}{2!} - \frac{y^3}{3!}...$; hence, the formulated optimization problem can have a polynomial form (see Equations 1, 2, and 3). The next two constraints on HT and PL ensure a good quality of the solution. Equation 3 shows the constraints on the feasible regions of all the configuration parameters: res ($v[1]$), sam ($v[2]$), dim ($v[3]$), and obs_av ($v[4]$). These are obtained by the experimental observations. There is no constraint on the obstacle density ($v[5]$) because it is an environmental parameter and depends on the navigation scene. $dist(start, dest.)$ is the Euclidean distance between the start and the end points.

V. Game Theory

We show in Section VI that solving the formulated optimization problem using a nonlinear solver takes time and sometimes even does not converge to a solution. Thus, we propose to develop games where the tunable parameters of the RRT* path planning algorithm are the players. We divide the parameters into dependent and independent parameters. The independent parameters (shown in Table I) are the players while the dependent parameters are used in conjunction with the independent parameters to formulate the payoffs of these players. This is because the dependent parameters are able to accurately model the physical quantities that govern the motion and hence the behavior of the UAV.

Our approach is to make the payoff of the players a function of two objectives: altruistic and selfish. The **altruistic objective** of the players is to minimize the hover time and the path length, which amounts to minimizing the wasted energy as well. It is negative in nature. We use hover energy as a proxy for hover time and energy consumed to travel a sub-optimal path as a proxy for the path length in our payoff equations. The **selfish objective** of the players is proportional to their individual parameters. Thus, the payoffs of the players is a combination of maximization of the selfish objective and minimization of the wasted energy, given the map of the environment.

A. Sensitivity Analyses

We performed the sensitivity analyses of the UAV flight (HT and PL) with respect to the parameters of the path planning algorithm on the NVIDIA Xavier board. Figure 5 shows the relationship of the worst-case hover time with the number of obstacles present in the search space. The worst-case hover time is when the UAV has to do re-planning at every obstacle. With an increase in the number of obstacles, the congestion in the search space increases, thereby leading to a higher hover time.

Figure 6 shows the relationship of the length of the path with the number of obstacles, while the source and destination for all the experimental points are kept the same. As the number of obstacles increases, the number of free spaces to form a collision-free path reduces. Thus, the length of the path increases because the planner has to take many detours.

If the resolution (step-size) of the path increases, the UAV takes larger steps in the direction of the destination, thereby reducing the number of collision checks and replanning steps. Hence, the hover time reduces. Figure 7 shows the relationship of the resolution with the hover time of the UAV.

B. Game Setup

We developed a game with five players: number of samples (*sam*), obstacle avoidance distance (*obs_av*), dimension of the search space (*dim*), resolution of the path (*res*), and the obstacle density (*obs_den*). Here, the first four players are the tunable parameters of the path planning algorithm and obstacle density is the input to capture the nature of the world through which the navigation is to be done. A higher obstacle density tries to reduce the payoff of the players by increasing the hover time and the path length as observed in Figures 5 and 6.

The complexity of the RRT* algorithm is directly proportional to the number of samples. Thus, the decision time and hence the hover time increases with an increase in the number of samples. The *sam* player would want to increase the number of samples to get a better decision, however it wants to minimize the wasted energy. In this case, the wasted energy is equal to the hovering energy, which is equal to the product of the hovering power and the hover time [14] (see Equation 4). The hovering power is constant [15] for a UAV with a given mass m, propeller radius r and the number of propellers n (see Equation 4). The payoff of the *sam* player is shown in Equation 5.

$$E_{hover} = P_{hover} * HT, \quad P_{hover} = \sqrt{\frac{(m*g)^3}{2*\rho*n*\pi*r^2}} \tag{4}$$

$$\begin{aligned}Payoff\_sam &= \alpha * sam - E_{hover} - \theta * obs\_den \\ &= \alpha * sam - \beta * HT - \theta * obs\_den\end{aligned} \quad (5)$$

The *obs_av* player wants that the UAV should fly at a distance from the obstacles. Thus, it would want to increase this distance while this would lead to an increase in the time for path planning and increased path length. Moreover, there will be a nonlinear relationship with the path length owing to the uncertainty in the sampling-based path planning algorithms. Hence, the payoff is captured in Equation 7, where E_{pl} is the energy spent in covering the path. It is equal to E_v if the UAV covers the entire path d at a constant velocity v (see Equation 6).

$$E_v = \int_{t_1=0}^{t_2=d/v} P(v)dt = P(v) * d/v \quad (6)$$

$$\begin{aligned}Payoff\_obs =&\alpha' * obs\_av - \gamma * PL^{\lambda} - \\ &E_{pl} - E_{hover} - \theta * obs\_den \\ =&\alpha' * obs\_av - \gamma * PL^{\lambda} - \\ &\kappa * PL - \beta * HT - \theta * obs\_den\end{aligned} \quad (7)$$

The relationship of the hover time and resolution of the path planning step is accurately captured in Figure 7. The hover time is a hyperbolic function of the resolution of the path. Thus, the *res* player would want to decrease the resolution, still observe all the obstacles and form a collision-free path, however the hovering energy would increase. Hence, the payoff can be captured using Equation 8.

$$\begin{aligned}Payoff\_res &= \alpha''/res - E_{hover} - \theta * obs\_den \\ &= \alpha''/res - \beta * HT - \theta * obs\_den\end{aligned} \quad (8)$$

As the dimension increases, the sample density reduces. This reduces the number of samples to choose from for the next nearest node. Thus, the time spent in planning and hence hovering increases. Due to the dispersed samples, the path length also increases. The relationship of the dimension with the path length is nonlinear, however the exact relationship is hard to deduce. The *dim* player would want to do the planning for a larger dimension, however it needs to minimize the wasted energy as a result of increased hovering and increase in the path length (E_{pl}). Equation 9 captures the payoff for the *dim* player.

$$\begin{aligned}Payoff\_dim =&\alpha''' * dim - \gamma' * PL^{\lambda'} - \\ &E_{pl} - E_{hover} - \theta * obs\_den \\ =&\alpha''' * dim - \gamma' * PL^{\lambda'} - \\ &\kappa * PL - \beta * HT - \theta * obs\_den\end{aligned} \quad (9)$$

All the constants – the αs, γs, and θ – are the hyper-parameters of the game. κ and β are constants derived from $P(v)$ and P_{hover}. We are using the constant λ in the equations involving PL because the exact function is not known. We will however derive insights for different values of λ.

C. Insights

We varied the strategies of the players in the feasible ranges obtained from real measurements (in prior work) and observed that:

❶ If all the αs are smaller than βs, γs, κs, and θ and $\lambda = 2$, we get a pure non-trivial NE (Nash Equilibrium). The *res* and *sam* players play their trivial strategy, while the *obs_av* and *dim* players play their non-trivial strategy. This is mainly because the altruistic objective dominates over the selfish objective and hence the players with a nonlinear dependence on PL in their altruistic objective need to tone down their strategies at the NE.

❷ Upon decreasing the value of λ to 0.5, we observe that the players with a nonlinear dependence on PL start moving towards more intuitive strategies as compared to the completely non-intuitive strategies in the previous case. This is because the strength of the non-linear dependence is drastically reduced from $\lambda = 2$ to $\lambda = 0.5$.

❸ As the density of the obstacles increases, the players start moving towards their trivial strategy.

VI. Results

A. Similar Results from Related Work

The time taken to solve the optimization problem for parameter tuning typically takes between 30 min to 2 hours [16], [17]. Burger et al. [17] explored 100 parameter combinations in 30 min while Cano et al. [16] explored 24 configurations in 2 hours. In comparison, our game-theoretic approach explores 256 configurations for a 4-player game in merely $0.01sec$ on an 8-core 3.4 GHz desktop machine using optimized Nash solvers provided by Gambit v15.1.1.

B. Comparison of Optimization-based and Game Theory-based Approaches

We compare the hover time and the path length obtained using the optimization based approach and the game-theoretic approach for three different virtual worlds. Figures 8, 9, and 10 show the occupancy map of the three worlds used in the experiments. Table II shows the comparison of the hover time and the path length obtained using the two approaches.

TABLE II
Comparison of optimization-based approach and the game-theoretic framework

Environment	**Optimization**		**Game Theory**	
	HT (sec)	PL (m)	HT (sec)	PL (m)
World 1	0.72	6.3	0.56	5.8
World 2	0.4	5.7	0.29	5.24
World 3	0.5	8.3	0.4	7.9

We observe from Table II that our game-theoretic approach provides solutions that are 5-21% better than the best optimization based approach. In terms of the time taken to calculate the optimal solution, it takes $0.01s$ to compute the Nash Equilibrium in Gambit. On the contrary, the IPOPT solver does not converge to a solution in finite time. The numbers reported in Table II correspond to the results obtained by simplifying the search space in IPOPT. For the cases where IPOPT converges with the actual or reduced search space, it still takes $0.1 - 1s$ to reach the solution, which is $10 - 100$X slower than the time taken by Gambit. We also observe that the values of the hover time and the path length for World 3 are more than that for World 2. Both the worlds (10m $\times$ 10m $\times$ 10m) correspond to a forest, however the tree density is half in World 2 (0.1 $trees/\text{m}^2$) as compared to World 3 (0.2 $trees/\text{m}^2$) as shown in Figures 9 and 10. A higher obstacle density leads to higher congestion and hence more hover time, which is expected.

C. Comparison with the Random Configurations

In this section, we show the results for World 1 (see Figure 8) by considering two to three parameters as the players while the other parameters are assigned random values within their feasible range. Table III shows a comparison of the planning time for the random and the best parameter configuration of these players. The best configuration is the one that is provided by the game theory

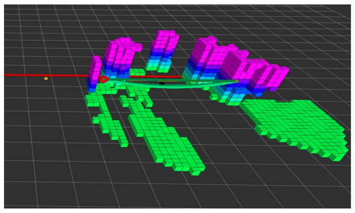
Fig. 8. 3D Occupancy map along with path planning in Rviz corresponding to World 1

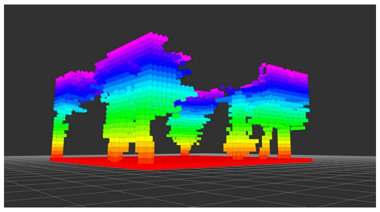
Fig. 9. 3D Occupancy map in Rviz for a forest with tree density 0.1 corresponding to World 2

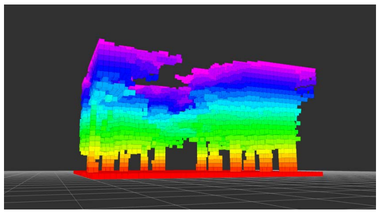
Fig. 10. 3D Occupancy map in Rviz for a forest with tree density 0.2 corresponding to World 3

based approach. We observe a 15-57% improvement in the planning time.

Scalability: It has been reported that the delays become very large while calculating the Nash Equilibrium in Gambit as the number of players or the number of strategies increases [10]. Nevertheless, this is not a problem for the path planning algorithms because the number of parameters in these algorithms and their ranges (strategies) are relatively small and finite [12], [16]. Thus, our solution is scalable for a wide range of path planning algorithms for practical settings.

TABLE III
COMPARISON OF PLANNING TIME (SEC) FOR RANDOM AND BEST CONFIGURATIONS

Players	Random	Best	Improvement(%)
sam, res	4.9	2.08	57.55
sam, dim	6.15	4.71	23.4
res, dim	4.98	2.64	46.9
res, obs_av	2.3	1.2	47.8
sam, dim, res	1.9	1.6	15.78
start: (0, 0, 2); dest.: (5.5, -2.0, 1.0)			

VII. Related Work

There are multiple proposals that target the problem of parameter tuning for path planning algorithms by formulating an optimization problem that aims to optimize a cost metric such as the hover time, planning time, or path length.

Luo et al. [18] and Dunlap et al. [19] studied the effect of tuning the parameters that determine the path length. They show that the relationship of the path length with these parameters has a very complicated form. Cano et al. [16] formulated the optimization problem as a cost minimization problem. The aim was to find a parameter combination that provides a valid trajectory using the path planning algorithm and minimizes the planning time. Since the time required for parameter exploration and tuning is large, they employed four intelligent search space exploration techniques based on random sampling, random forest, Bayesian Optimization, and AUC Bandit. Similarly, Burger et al. [17] solved the optimization problem using the SMAC [20] tool. The tool internally uses random forests to explore the parameter space. Due to the time consuming nature of these techniques, they set a cut-off time for the exploration. In contrast, our game-theoretic formulation provides theoretical guarantees about the solution and is much more time-efficient.

VIII. Conclusion

This paper proposes a very novel solution to the hyper-parameter estimation problem in the path planning routines used in UAVs. It takes a diametrically different approach as compared to conventional work; it proposes to convert a regular nonlinear optimization problem to a game, and quickly find the set of equilibrium strategies for the game. This gave us a 10-100 X speedup without compromising on the quality of the solution. Our work has broad implications beyond the scope of the current problem; it has potential applications in a wide number of optimization problems that arise in cyber physical systems.

References

[1] A. M. Research. Global commercial drone market. https://www.globenewswire.com/news-release/2019/08/28/1907826/0/en/Global-Commercial-Drone-Market-to-Garner-10-28-Billion-by-2022-at-25-2-CAGR-Says-Allied-Market-Research.html.

[2] FAA. Fact sheet faa forecastfiscals years 2016-37. https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=21514.

[3] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. Reddi, "Mavbench: Micro aerial vehicle benchmarking," in *MICRO*. IEEE, 2018, pp. 894–907.

[4] G. Kulathunga, D. Devitt, R. Fedorenko, S. Savin, and A. Klimchik, "Path planning followed by kinodynamic smoothing for multirotor aerial vehicles (mavs)," *arXiv preprint arXiv:2008.12950*, 2020.

[5] S. R. Chowdhury, "A game theoretic approach to robust optimization," Ph.D. dissertation, Indian Institute of Science Bangalore, 2015.

[6] D. Cheng and Z. Liu, "Optimization via game theoretic control," *National Science Review*, 2020.

[7] R. Meng, Y. Ye, and N.-g. Xie, "Multi-objective optimization design methods based on game theory," in *WCICA*. IEEE, 2010, pp. 2220–2227.

[8] B. Boroujerdian, H. Genc, S. Krishnan, B. P. Duisterhof, B. Plancher, K. Mansoorshahi, M. Almeida, W. Cui, A. Faust, and V. J. Reddi, "The role of compute in autonomous aerial vehicles," *arXiv preprint arXiv:1906.10513*, 2019.

[9] L. Yang, J. Qi, J. Xiao, and X. Yong, "A literature review of uav 3d path planning," in *WCICA*. IEEE, 2014, pp. 2376–2381.

[10] McKelvey, R. D., McLennan, A. M., and T. L. Turocy. Gambit: Software tools for game theory, version 15.1.1. http://www.gambit-project.org.

[11] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *IROS*. IEEE, 2016, pp. 5332–5339.

[12] B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, and E. Magid, "Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model," in *ICRA*. IEEE, 2020, pp. 7226–7232.

[13] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[14] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in *ICARSC*. IEEE, 2015, pp. 111–117.

[15] J. Yao and N. Ansari, "Qos-aware power control in internet of drones for data collection service," *TVT*, vol. 68, no. 7, pp. 6649–6656, 2019.

[16] J. Cano, Y. Yang, B. Bodin, V. Nagarajan, and M. O'Boyle, "Automatic parameter tuning of motion planning algorithms," in *IROS*. IEEE, 2018, pp. 8103–8109.

[17] R. Burger, M. Bharatheesha, M. van Eert, and R. Babuška, "Automated tuning and configuration of path planning algorithms," in *ICRA*. IEEE, 2017, pp. 4371–4376.

[18] J. Luo and K. Hauser, "An empirical study of optimal motion planning," in *IROS*. IEEE, 2014, pp. 1761–1768.

[19] D. D. Dunlap, C. V. Caldwell, E. G. Collins *et al.*, "Motion planning for mobile robots via sampling-based model predictive optimization," *Recent advances in mobile robotics*, vol. 1, 2011.

[20] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *LION*. Springer, 2011, pp. 507–523.