Heuristic Approach for Real-Time Multi-Agent Trajectory Planning Under Uncertainty

HASEEB TAHIR1, MUJAHID N. SYED1, AND UTHMAN BAROUDI2

1 Systems Engineering Department, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran 31261, Saudi Arabia
2 Computer Engineering Department, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran 31261, Saudi Arabia
Corresponding author: Uthman Baroudi (ubaroudi@kfupm.edu.sa)

This work was supported in part by the National Plan for Science, Technology and Innovation (MAARIFAH) - King Abdulaziz City for Science and Technology through the Science and Technology Unit, King Fahd University of Petroleum and Minerals (KFUPM), Kingdom of Saudi Arabia, under Award Project 11-ELE2147-4.

ABSTRACT Motion planning of agents is one of the fundamental research problems in autonomous systems. An important aspect of motion planning is collision avoidance of the agents with other agents and obstacles that are present in the agent’s environment. Typically, the collision avoidance constraints are non-linear and non-convex. Thus, the mathematical formulation of motion planning of multiple agents, in the presence of other agents and obstacles, is NP-Hard. In this paper, a novel heuristic approach for motion planning in multi-agent dynamic environment is proposed. The approach is computationally cheap, and can be launched locally on each agent for the trajectory planning. The applicability of the proposed approach is illustrated by numerical examples considering uncertainty in the environment. Detailed discussions on the performance of the proposed approach are presented. Finally, the observations on the key characteristics of the proposed approach are summarized.

INDEX TERMS Trajectory planning, multi-agent, path planning, robotic networks, collision avoidance.

I. INTRODUCTION

Motion planning is one of the most significant attributes of autonomous systems and still continues to be an active area of research. Some early applications of motion planning were found in rocket propulsion and space exploration [1], [2]. Automation in the manufacturing industry brought forward new ways that could be explored using the motion planning techniques [3]–[8]. The recent works in motion planning are found to address diverse issues like machine learning, advanced manufacturing processes, puzzle’s solutions, computer graphics, games, navigation in self-driving cars and Autonomous Unmanned Vehicles (AUV) [7], [8].

Motion planning for single agent in the presence of multiple static obstacles can be modeled as a mathematical programming problem. The key difficulty in the mathematical model arises due to the presence of collision avoidance constraint, as illustrated in (1):

$$\|p_a - o_i\|_m \geq D$$

where, \(p_a\) and \(o_i\) are the current positions of the agent \(a\) and obstacle \(i\), respectively. The constraint enforces the norm-m distance to be greater than or equal to some fixed distance \(D\), in order to avoid the collision. This constraint is the main cause for the non-convexity. In addition, motion planning in dynamic environments results in a combination of path planning and velocity planning problems [9]. The dynamic motion planning problem where each agent is modeled as a point among multiple moving bodies with bounded velocity is an NP-hard problem [10].

The trajectory planning algorithms typically require some set of finite transformations that move an agent from its initial location to the target (or final) location. The set of all possible transformations that move an agent from initial to any other location is referred to as state space or Configuration Space (C-space), with \(C_{obs}\) as the transformations that lead to obstacles and \(C_{free}\) as otherwise [7]. In simple words, C-space is actually how the agent sees and understands its environment, and it primarily depends on how the problem is modeled. Obviously, there are many unrelated things in the environment w.r.t to trajectory planning. Therefore, the problem should be formulated in a way that it extracts the information efficiently. The model of the problem also depends on the accuracy of available sensors on board. The algorithm should find the best sequence of transformations in \(C_{free}\) that yields a feasible trajectory to the target.
The main contributions of this paper can be summarized in the following points:

- A novel mathematical formulation for the local path planning in a multi-agent dynamic & uncertain environment that avoids stalling and oscillation is proposed.
- A heuristic decomposition of the formulation is proposed, which reduces the computational complexity of the proposed approach.
- The proposed algorithm can be implemented on agents with limited computational capabilities.

The paper is structured as follows: Section II presents the comparison of motion planning algorithms in literature along with some related problems and limitations. Section III defines the problem, and Section IV provides detailed explanation of the proposed mathematical model. Later, in Section V, an algorithm is developed to search for collision free velocities for the agents. Performance of the algorithm is illustrated in Section VI.

II. LITERATURE SURVEY

The existing algorithms/methods developed for motion planning can be classified into the following categories: mathematical programming algorithms, sampling based methods, potential field and reactive methods. In the following, the main contributions of each category are presented.

A. MATHEMATICAL PROGRAMMING METHODS

Multi-agent path planning typically becomes a non-convex and non-linear mathematical optimization problem due to the presence of the collision avoidance constraints [11]. These constraints are usually referred to as the reverse convex constraints and have been well studied in the literature. Various algorithms have been suggested for finding optimal solutions specifically for convex problems with only one additional reverse convex constraint [12], [13]. However, these techniques are inappropriate for multi-agent dynamic environments where numerous such constraints are implied, and the feasible space may become complicated enough to be analyzed for global optimality. Conversely, sequential and incremental sequential convex programming heuristics may be used to linearize these constraints as in [14] and [11], but they produce sub-optimal solutions, especially in the presence of dynamic agents. Mixed Integer Linear Programming (MILP) models may also be used for non-convex optimization such as in [15], where the usual quadratic cost function is expressed as weighted norm-1 function. A similar formulation for 3D motion and collision avoidance of spacecrafts and obstacles is presented in [16]. Mixed Integer Non Linear Programming (MINLP) formulations are also used in path planning, [17]. MILP models may also be extended to Mixed Integer Quadratic Programming (MIQP) model, [18] and the control inputs for position, roll, pitch and yaw angles are optimized for each of the discrete time instants. The Mixed Integer Programming (MIP) techniques may render optimal or near optimal solutions, but are computationally very expensive due to the presence of integer variables. The problem dimensionality increases exponentially with increasing number of agents and huge computational resources may be required for optimization. Additionally, mathematical programming techniques require central computing and sensing capabilities which may not be feasible for many real world applications.

B. SAMPLING BASED METHODS

In contrast to the above, sampling based algorithms can address complicated motion planning problems. The challenges faced by the mathematical programming methods may be avoided, especially where a complicated higher dimensional motion planning is required. Two of the well-known classical sampling based algorithms are Probabilistic Road Maps (PRMs) and Rapidly exploring Random Trees (RRTs). Roadmaps define the configuration space topology by developing a network of collision free trajectories. PRMs are simply the Monte-Carlo evolution of the roadmaps. PRMs can address multi-degree complex robotic movements as it simplifies the problem by expanding the sample in only relevant portion of the configuration space [19]. PRMs have also been used with Dynamic Robot Networks (DRN) [20] in dynamic motion planning problems but heavily relies on robot network communication for environment sampling. On the other hand, incremental search methods such as dynamic programming, A*, bi-directional search etc. evolved into randomized methods like RRT. RRTs randomly explore the environment by biasing the search through random sampled points in unexplored portion of the state space and are specifically designed to handle problems with non-holonomic constraints and high degree of freedom [21], [22]. Well known extensions of RRT are as follows: single RRT planners, bi-directional planners, RRT*, RRT* Fixed Node and RRG to address multi-agent and multi degree non-linear path planning problems, [22]–[25].

A group of sampling based methods has Artificial Intelligence (AI) flavor. For example: D*, focused D*, LPA*, D* Lite etc. [26]. The computationally expensive environmental sampling in AI algorithms may be improved to some extent by using the concept of super nodes where each super node represents a group of connected sub-graphs [27]. AI methods have also been extended for 3-D spaces [28]. Moreover, evolutionary metaheuristics methods including Ant-Colony (ACO), Simulated Annealing (SA) and Genetic Algorithms (GA) have been used in the path planning methods [26], [29]–[32].

C. POTENTIAL FIELD METHODS

The numerical complexities faced by mathematical programming models may also be avoided by another set of methods known as Artificial Potential Field (APF) [33]. APF methods provide virtual force field functions for collision avoidance. One of the issues in APF method is the trapping of algorithm in local minima, which occurs in situations when the forces due to attractive fields and repulsive fields balance out each
other. The study of Hessian of the potential field function shows its non-convexity, particularly near obstacles [34]. Navigation functions may be constructed such that local maxima of the potential function only occurs at the boundary of the active space. But tweaking these parameters for different scenarios may be infeasible for dynamic environment and very complicated even for static environments (when performed by the agent’s on-board computing system). The second issue is the inability of motion between the closely spaced obstacles, which is caused by the repulsive fields of the obstacles that overcome the attractive fields even when the space between obstacles allows robot motion. The third issue is the oscillatory motion, while moving near obstacles and narrow passages [35]. Attempts have also been made to address the issue of oscillations for example in [36] and [37] by the grid histogram and Newtonian potential field models. However, these attempts do not address the issues related to trapping of agent in local minima. Another issue faced by APF methods is Goal Non-Reachable with Obstacles Nearby (GNRON), which is due to the non-convexity of potential function in the presence of obstacles near goal position. To ensure that the global minimum of potential function lies only at the goal position, an additional function is introduced in the potential field expression that depends on the distances of goal and obstacles from the agent’s position [38]. The situation complicates even further in the presence of multi-agent multi-obstacle environment, and this simplistic model of the motion planning problem may not achieve the desired results [39]–[41].

D. REACTIVE METHODS

Reactive methods use local information available to the agent in its nearby vicinity. Such methods are suitable for on-board path planning, where computational resources are limited. The issue of local minima faced by previous methods is solved by certain sensors based methods like in [42] and [43] that work similar to the boundary following approach found in bug algorithms. These algorithms are able to solve navigation problems with minimal information and do not require self-localization, which is computationally expensive. However, there is no systematic way to detect local minima situations without any self-localization mechanism. Some reactive methods also use the concept of collision cones [9]. A possible future collision is found by analyzing these cones and applying necessary changes to the agent’s velocity or direction to avoid collision [44]. The difficulty arises in the presence of cluttered environment and narrow passages due to the presence of many collision cones. The reactive methods have also been suggested for non-holonomic motion planning, where the algorithm switches between three different approaches i.e., global planning, local planning and controls planning [45]. However, if global information is not available, then they rely on boundary tracking with similar results as in some previously discussed methods. Some learning based techniques [46], may generate improved trajectories but require higher training times. Consensus-based formation control strategies have also been applied for multi-agent systems subjected to external disturbances [47], [48].

Reactive path planning techniques are generally classified as greedy algorithms. The idea of collision cones provide a sound mathematical foundation of the problem for further optimization. Therefore, such ideas have been extended with some mathematical programming techniques to avoid collision in multi-agent environment, for example in [49]. The problem is first modeled with non-convex collision cones. Later, hyperspaces are chosen such that only feasible velocities are left and require minimum deviation from the agents preferred velocity. Thus, the non-convex feasible space is converted into convex polytope. Each agent solves a quadratic objective function to avoid collision in multi-agent dynamic environment. The algorithm is further improved in [50] and [51] to deal with some issues related to oscillations in the agents.

Reactive methods in combination with optimization techniques may render good solution quality, but may require high computational capabilities for on-board optimization. Also, non-convexity of the collision cones may be avoided by convex approximation of the feasible space, but doing this is itself a cumbersome task. In this work, we propose a novel heuristic approach to the multi-agent path planning problem. The proposed approach incorporates ideas from mathematical programming and reactive methods.

III. PROBLEM DESCRIPTION

A. STATEMENT

$N$ agents with $K$ obstacles are present in an environment. All agents have the information of their target locations. The agents may not be able to communicate with each other, but their sensors have capability to acquire the information from their surroundings. The objective is to find collision free trajectories for the agents from their initial locations to their corresponding target locations.

B. GOAL

As evident from the above statement, the agents do not have global information of the environment. From the literature, even if the global information is available, finding global optimal paths is NP-hard [10]. Thus, our goal is to develop an algorithm, which can find collision free paths of traverse for each agent. Furthermore, the algorithm should be fast and simple enough to suit the limited on-board computational capability of the agents.

C. ASSUMPTIONS

In this work, the agents are assumed to be spherical in shape ($i^{th}$ agent has radius $r_i$) and are capable of holonomic motion. The obstacles present in the environment (may be static or dynamic) are also assumed to be spherical. An $i^{th}$ agent has the capability to recognize the objects that are present with in the vicinity of radius $s_i$. The agent can only recognize the surface edges of obstacles that are exposed to
FIGURE 1. Environment model for single agent with multiple obstacles, where \( w_j \) is the current velocity, \( L_j \) and \( R_j \) are the left and right edges of \( j^{th} \) obstacle.

it. Thus, each agent can obtain the edge information of the obstacles and other agents present in its vicinity. Furthermore, each agent is capable of acquiring the velocities \( v_i \) and \( w_j \) of every \( i^{th} \) agent and \( j^{th} \) obstacle that are present in its vicinity. The velocity of each agent is assumed to be constant during each time step. The change in agent’s velocity between the time steps is governed by kinodynamic constraints. Moreover, we assume that the information is imperfect, i.e., the information related to surface edges obtained by the agent include uncertainty. The uncertainty in the edges is captured by multiple probes.

IV. METHODOLOGY

In this section, a novel mathematical formulation is developed based on the problem description (see Section III). First, collision avoidance constraints based on collision cones theory is illustrated. The theory is then extended to handle uncertain scenarios. Finally, the kinodynamic, stalling and oscillation avoidance constraints are proposed.

A. BASIC FORMULATION

Consider a scenario where an agent is present in a multi-agent and multi obstacle dynamic environment. The agent is assumed to be able to gather boundary or edge information of other agents and obstacles that are present in some vicinity of its surroundings. A 2-D scenario of single agent and multiple obstacles is shown in Fig. (1), where the agent has obtained the velocities and edges of the obstacles present in its vicinity. The edge information of each \( j^{th} \) obstacle contains two vectors, \( R_j \) and \( L_j \), showing relative position of the right and left edges, respectively, where the agent’s center is considered as the origin of the vector space. In order to avoid collision with an obstacle, the center of the agent should always remain at some distance away from the obstacles. The set of all such minimum distances, thus, form Collision Space (CS) which depends on the dimensions of the agent and obstacles. The CS w.r.t each obstacle can be conveniently formed by taking the following Minkowski sum (See Fig. (2)):

\[
A \oplus P_j = \{ a + p_j | a \in A, p_j \in P_j \} \quad (2)
\]

where \( A \) and \( P_j \) are the sets representing the shape of the agent and the \( j^{th} \) obstacle, respectively. It is important to point out here that the agent may not be able to find the Minkowski sum due to the sensory limitations. However, the model of the problem presented here does not require the agent to calculate all the CSs for collision avoidance. Each pair of vectors, \( R_j \) and \( L_j \), for each obstacle as illustrated in Fig. 2, can be used in combination with the sum as obtained in (2) to form the Collision Cone (CC). The agent will not collide with any of the obstacles in a future time, if and only if, its current relative velocity vector w.r.t every obstacle lie outside of the CC formed by the edge vectors of that particular obstacle [9]. A smaller rectangular region from Fig. (2) has been chosen to show the formation of the CC w.r.t Obstacle-1 (see Fig. (3)). Vectors \( R_1 \) and \( L_1 \) can be used together with the agent’s radius, \( r \), to find vectors \( a_{R_1} \) and \( a_{L_1} \), respectively in the right angled triangles shown in Fig. (3).

The CC thus formed by these two vectors is convex. Similar CCs can be constructed with respect to all the obstacles detected by the agent. The agent’s relative velocity, \( c_j \), w.r.t to each of the \( j^{th} \) obstacle should exist anywhere in the space outside of all the CCs in order to avoid collision with any of the obstacles, which is shown mathematically as follows:

\[
c_j = c - w_j \notin CC \quad (3)
\]
and, if then the system in (6) can be solved as: 

where, \( c \) is the absolute velocity of the agent under consideration, and 

\[
CC = \{ x : x = \lambda R_j a_{R_j} + \lambda L_j a_{L_j} | \lambda R_j, \lambda L_j \in [0, \infty) \forall j \} \tag{4}
\]

Using (3) and (4), the following set of constraints can be written for the agent’s relative velocity w.r.t Obstacle-1:

\[
A_1 \lambda_1 \neq c_1 \tag{5}
\]

where \( A_1 = [a_{R_1}, a_{L_1}] \), \( \lambda_1 = [\lambda_{R_1}, \lambda_{L_1}] \) such that \( \lambda_1 \geq 0 \) and the column vectors \( a_{R_1}, a_{L_1} \in \mathbb{R}^n \). The above system holds true, if and only if, the following system does not have a solution:

\[
A_1 \lambda_1 = c_1, \quad \lambda_1 \geq 0 \tag{6}
\]

Obviously, when \( A_1 \) is a square & non-singular matrix, then the system in (6) can be solved as:

\[
\lambda_1 = A_1^{-1} c_1 \tag{7}
\]

and, if \( A_1^{-1} c_1 \neq 0 \), then the system in (5) can be classified as feasible. However, in practice \( A_1 \) may not be a square matrix. Possible reasons for this include the problem dimensions, uncertainty, sensor sampling rate, etc. Thus, a general case of the system in (6) may not be trivial to solve.

According to the Farkas’s lemma [52], if the system in (6) does not have a solution, then the following system must have a solution (and vice versa):

\[
A_1^T x_1 \leq 0 \tag{8a}
\]

\[
c_1^T x_1 > 0 \tag{8b}
\]

where vector \( x_1 \) represents the unknown variables in System (8). The feasible space created by the above system of inequalities for Obstacle-1 is shown in Fig. (4). Also, it can be seen that the new relative velocity vector of the agent, \( c_1 \), will generate a collision free trajectory with respect to Obstacle-1, if and only if, the system in (8) has a solution. The set of constraints in (8) are checked for the feasibility of agent’s new velocity with respect to one obstacle. Similar CCs can also be formed for the other obstacles as shown in Fig. (5).

![FIGURE 4. Shaded area shows the set of hyperspaces creating feasible space formed by the two collision cone vectors of Obstacle-1 and vector \( c \).](image)

\[\text{FIGURE 5.} \text{ Obstacles moving with velocity,} \ w_j, \text{ and CCs formed by their respective vectors.}\]

**B. FORMULATION OF DATA UNCERTAINTY**

In practical scenarios, the edge vectors \( a_{R_j}, a_{L_j} \) for neighbor objects (agents and/or obstacles) \( j = 1, \ldots, N \) might not be estimated with certainty. The uncertainty in the edge vectors arise due to the sensor capabilities, precision & accuracy, background noise, and speed of the agent. This uncertainty in the data needs additional consideration. Different approaches to handle uncertainty in optimization models are presented in [53]–[56]. In order to incorporate uncertainty in the edge vectors the following constraints instead of (8a) can be used.

\[
P(v_j^T x_j \leq 0) \geq \alpha_j \forall r, j \tag{9}
\]

where \( \xi_j \) is a random vector corresponding to the \( v_j \)th edge vector of neighbor obstacle \( j \), \( \alpha_j \) represents the minimum probability with which the constraint should be satisfied, and \( v \in \{R, L\} \). If the density function of \( \xi_j \) is known, then the above equation can be simplified further. Typically, the density function corresponding to the edge vector is unknown in real scenarios. A pragmatic approach to handle the uncertainty in the edge vectors is to sample the edge vectors multiple times (if possible, using multiple sensors). The samples can be obtained by repetitive sampling from one and/or multiple sensors. In this work, uncertainty in edge vectors is handled via robust optimization technique. The idea is to use the sample realizations of the edge vectors and avoid the worst-case scenario. In (8a), for every \( g^{th} \) sample realization of the \( v_j \)th edge vector obtained from \( j^{th} \) neighbor obstacle, an edge constraint of the form \( a_{R_j}^T x_j \leq 0 \) is added. Thus, matrix \( A_j \) w.r.t obstacle \( j \) will have more than \( n \) columns in \( n \)-dimensions.

**C. KINODYNAMIC CONSTRAINTS**

The agent’s new velocity vector \( c \) should be chosen such that it satisfies the agent’s kinodynamic constraints. The agent’s current acceleration, \( g_{tr} \), and its current velocity, \( v_{tr} \), determines its future velocity, \( c \), in the next iteration after \( \tau \).
time step. Both $\mathbf{g}_a$ and $\mathbf{v}_a$ are bounded and the kinodynamic constraints on acceleration and velocity can be written as:

$$v_{\min} \leq ||\mathbf{e}||_2 \leq v_{\max}$$  \hspace{1cm} (10)  

$$a_{\min} \leq ||\mathbf{g}_a||_2 \leq a_{\max}$$  \hspace{1cm} (11)  

$$\mathbf{c} = \tau \mathbf{g}_a + \mathbf{v}_a$$  \hspace{1cm} (12)  

where, $a_{\min}$ ($v_{\min}$) is the lower bound, and $a_{\max}$ ($v_{\max}$) is the upper bound on agent’s current acceleration (future velocity) after $\tau$ time step.

D. STALLING CONSTRAINTS

The phenomenon of stalling can be explained with an example shown in Fig. (6). In order to move as close as possible to the target from position $O$, the best possibilities for the agent are to move along any of the two edge vectors, $\mathbf{a}_R$ and $\mathbf{a}_L$. Let us suppose that the agent selects $\mathbf{a}_R$ as the direction of motion in current time step. Let, $O$ and $O'$ show the agent’s initial location and the location it moved to in one time-step respectively. Now, the following conditions must hold:

$$\mathbf{a}_R^T \mathbf{t}_a < \mathbf{a}_R^T \mathbf{t}_{a'}$$  \hspace{1cm} (13)  

$$\mathbf{a}_L^T \mathbf{t}_a > \mathbf{a}_L^T \mathbf{t}_{a'}$$  \hspace{1cm} (14)  

where $\mathbf{t}_a$ and $\mathbf{t}_{a'}$ are vectors from the agents locations $O$ and $O'$ to the target, respectively. As per the above conditions, now the best possibility for the agent is to move along $\mathbf{a}'_L$. This will guide the agent to change its direction abruptly in the very next iteration. This abrupt change will lead the agent towards initial location $O$. Thus, the agent is going to move back and forth between $O$ and $O'$. This phenomenon is known as stalling. Stalling may be avoided by forcing the agent not to change the direction of its velocity very rapidly as compared to its previous velocity. Constraining the agent’s velocity in this manner also helps to reduce mechanical jerks on the system and produce smooth trajectories. Thus, the following constraint is placed on agent’s velocity to avoid stalling and any abrupt velocity changes:

$$\mathbf{c}^T \mathbf{v}_a \geq 0$$  \hspace{1cm} (15)  

Although the above constraint restricts abrupt changes in the velocity of an agent, the jerks in the agent’s path may not be completely eliminated. In order to avoid jerks, following constraint can be added to the model:

$$||\mathbf{g}_a^{\text{new}} - \mathbf{g}_a^{\text{old}}||_2 \leq \psi$$  \hspace{1cm} (16)  

where $\mathbf{g}_a^{\text{new}}$ and $\mathbf{g}_a^{\text{old}}$ are the current and the previous acceleration of agent $a$ respectively, and $\psi$ is a positive constants that limits the amount of jerk. Restriction in (16) is a convex constraint. However, adding such constraint will transform the formulation to a nonlinear formulation. Thus, the above constraint is ignored in the current study.

E. OSCILLATION AVOIDANCE CONSTRAINTS

Oscillations are typically caused by two or more interacting agents. Agents may get stuck in oscillation forever [50]. Thus, avoidance of oscillation is one of the key concepts in multi-agent scenarios. Let $OC$ be the set of velocities of interacting agents that will end up in oscillations. The goal of the interacting agents is to avoid such velocities.

F. OVERALL FORMULATION

The mathematical formulation obtained by assimilating system (8) for all obstacles, along with uncertainty in edges, the kinodynamic, stalling and oscillatory constraints is presented in Formulation (17). The problem can be formulated as a maximization problem, where the objective is to find the best possible agent’s velocity vector $\mathbf{c}$ in the direction of its target.

$$\max. : \mathbf{t}_a^T \mathbf{c}$$  \hspace{1cm} (17a)  

$$\mathrm{s.t.} : \mathbf{A}_j^T \mathbf{x}_j \leq 0 \ \forall j$$  \hspace{1cm} (17b)  

$$\mathbf{c}_j^T \mathbf{x}_j \geq \varepsilon \ \forall j$$  \hspace{1cm} (17c)  

$$\mathbf{c}_j = \mathbf{c} - \mathbf{w}_j \ \forall j$$  \hspace{1cm} (17d)  

$$\mathbf{c} = \tau \mathbf{g}_a + \mathbf{v}_a$$  \hspace{1cm} (17e)  

$$v_{\min} \leq ||\mathbf{e}||_2 \leq v_{\max}$$  \hspace{1cm} (17f)  

$$a_{\min} \leq ||\mathbf{g}_a||_2 \leq a_{\max}$$  \hspace{1cm} (17g)  

$$\mathbf{c}^T \mathbf{v}_a \geq 0$$  \hspace{1cm} (17h)  

$$\mathbf{c} \notin OC$$  \hspace{1cm} (17i)  

In Formulation (17), $\mathbf{t}_a$ is the target vector w.r.t the agent’s current position, and $\varepsilon$ is a small positive scalar. Constraints (17b) & (17c) ensure that all $\mathbf{c}_j$’s are outside their respective collision cones. Constraint (17d) captures the relative velocities, and Constraints (17e)-(17g) enforce kinodynamic constraints on the agent’s velocity. Constraints (17h) & (17i) avoid stalling and oscillations. Notice that Constraint (17c) has bi-linear terms which makes the above formulation non-linear and non-convex. In addition to that, Constraints (17f) & (17g) are nonlinear. Furthermore, the set $OC$ in Constraint (17i) is dynamic and it may not be realized beforehand in practical scenarios. Thus, solving the overall formulation is computationally very expensive, and solving it at each time step may not be practical. Nevertheless, the constraints of the overall formulation provide a mechanism to
check feasibility of agent’s velocity vector $c$. In this work, a heuristic approach to decompose the overall mathematical formulation is proposed and presented in Section V.

V. PROPOSED APPROACH

This section presents the proposed approach to handle Formulation (17). Our approach is to decompose the formulation, and provide a heuristic solution method. The proposed solution method will be computationally cheap (requires basic mathematical operations), and can be implemented on agent’s on board processing mechanism.

A. DECOMPOSITION OF THE OVERALL FORMULATION

The overall formulation contains three major parts: Constraints (17b)-(17d) are related to feasibility & uncertainty, Constraints (17e)-(17g) are related to kinodynamics, Constraint (17h) is related to stalling, and Constraint (17i) is related to oscillations. We propose to handle Constraints (17b) & (17c) by decomposing $A_j$ into square scenarios. Section V-B provides the details of the square decomposition. Kinodynamics & stalling related constraints are handled by discretizing the set of the agent’s possible velocity. Section V-D provides the details of the discretization mechanism. Constraints (17a), (17d)-(17g), and (17h) are handled through the discretization mechanism. The oscillatory constraint is handled by a scaling mechanism. Section V-C provides the details of the scaling mechanism. Using the square decomposition, scaling mechanism and discretization mechanism, the overall formulation is heuristically solved.

B. SQUARE DECOMPOSITION

The non-square nature of $A_j$ that arises due to uncertainty in the data can be decomposed into a square case. Let $f_j$ be the vector originating from the center of agent and terminating in the center of object $j$. Due to uncertainty, $f_j$ is unknown. However, it can be estimated as:

$$\hat{f}_j = \frac{\sum_{r \in K_R} a_{rj}^+}{2|K_R|} + \frac{\sum_{l \in K_L} a_{lj}^+}{2|K_L|}$$

where $\hat{f}_j$ is the estimated value of $f_j$, $K_R$ and $K_L$ are set of edge vectors related to right and left edges of the obstacle. Now, define $\text{a}_{Rj}^+$ and $\text{a}_{Lj}^+$ as:

$$\text{a}_{Rj}^+ = \arg\min_{a_{Rj}^+} \hat{f}_j, \quad \forall v \in \{R, L\}$$

Thus, (17b) & (17c) can be decomposed into worst-case scenario, where the scenario contains square $A_j^* = [\text{a}_{Rj}^+, \text{a}_{Lj}^+]$ matrix. Furthermore, the columns of $A_j^*$ are arranged such that the right edge vector is placed in the first column, and the left edge vector is placed in the second column. Now from the right hand rule it can be stated that the determinant is positive. In other words, such placement ensures that $\text{det}(A_j^*) > 0$. Thus, for a given $c$ vector, the existence of a feasible solution to (17b) & (17c) is true, if the following conditions hold:

$$(c_{ij}a_{Rij}^+ - c_{ij}a_{Lij}^+) < 0 \quad \text{or} \quad (c_{ij}a_{Rij}^+ - c_{ij}a_{Lij}^+) < 0$$

where $c = [c_{ij}, c_{ij}]^T$, and $A_j^* = [a_{Rij}^+, a_{Lij}^+]$. A $c$ vector satisfying the above conditions will be robust to uncertainty. Thus, instead of handling the system of nonlinear equations, the conditions given in (20) are evaluated to check the feasibility of the given vector $c$ w.r.t one obstacle. Therefore, the worse case complexity of the proposed square decomposition will be $O(NK)$, where $N$ is the total number of neighbor objects in the vicinity of the agent, and $K$ is the maximum number of copies of any edge vector received by the agent. After the square decomposition, every obstacle is referred by two edge vectors ($\text{a}_{Rj}^+, \text{a}_{Lj}^+$).

C. SCALING MECHANISM

A technique similar to [50] has been proposed to avoid the oscillations. However, instead of shifting the original VO cones for all other agents as done in [50], the magnitudes of the left edge vectors are scaled by a factor of $\alpha$, where $0 < \alpha < 1$, and the vectors are archived in a set called Edge Vector Stack (EVS). The vectors in EVS are sorted in descending order of their dot product with the relative target vector of the agent, $v$. Thus, such scaling increases the priority of selecting the right edge vectors from EVS as a potential candidate for agent’s new velocity. This idea of avoiding oscillations requires minimum computational effort. Additionally, in contrast to HRVO, the above approach does not require sensing capabilities in the agents to distinguish between an agent and an obstacle. Thus, the proposed scaling mechanism is a computationally cheap trick to ensure that (17i) is feasible. The worse case complexity of the proposed scaling mechanism will be $O(N)$, where $N$ is the total number of neighbor objects in the vicinity of the agent.

D. DISCRETIZATION MECHANISM

A finite set of the possible values of the agent’s new velocity vector $c$, can be obtained by discretizing the set of the agent’s kinodynamic constraints. For static environments, it can be easily seen that if there exists a collision free path, the $c$ vector in the direction of the target or in the direction of one of the vectors in EVS must be feasible. For dynamic environments, we need additional options for velocity vector. Thus, a uniform random sample of the Possible Velocity Space (PVS) is generated. The vectors in PVS are sorted in descending order of their dot product with the relative target vector of the agent, $v$.

E. PROPOSED ALGORITHM

Every agent based on its vicinity radius builds the possible pool of velocity vectors. The possible pool of velocity vectors are archived into ordered set $\Delta = v \cup EVS \cup PVS$ and tested for feasibility. Now, the first vector from $\Delta$ is tested for feasibility w.r.t (20) & (17h). If the first vector is
infeasible, then it is neglected, and the search continues to the next vector. If no vector from $\Delta$ is feasible, which happens due to the presence of obstacles all around the agent, then the agent reduces its vicinity radius. With the new vicinity radius, the agent rebuilds and rechecks each vector in $\Delta$. If the vicinity radius falls below a specified threshold and no vector in $\Delta$ is feasible, then a zero velocity is assigned to the agent. If a feasible vector is obtained, then (17e)-(17g) are incorporated as follows.

Let $\gamma$ be the ratio of current vicinity radius and the maximum vicinity radius. Let the values of maximum speed and normal speed be derived from $v_{\text{min}}, v_{\text{max}}, \alpha_{\text{min}}$ and $\alpha_{\text{max}}$. Now, the agents velocity is set as per the following rules: If no obstacles are present in the vicinity of an agent, then the agent moves with $\gamma$ times the maximum speed along $t_{0}$. If there are obstacles in the vicinity of an agent, then the agent moves with a speed of 80% to 100% of $\gamma$ times normal speed along the feasible vector identified from $\Delta$. Let $S$ be the maximum number of times the vicinity radius of an agent is updated during the search in a given time step. Let $P$ be the cardinality of $PVS$. The worse case complexity of Algorithm (1) is $O(SN^{2} + SNK + SNP + SP \log(P))$. For practical cases, typically $K$, $S$ and $P$ are positive constants, i.e., they do not change with number of objects in the vicinity. On the other hand, $N$ do change with number of other objects in the vicinity. For the certainty case, the complexity of the proposed algorithm will be: $O(SN^{2} + SNP + SP \log(P))$. Therefore, the approach presented in Algorithm (1) is polynomial w.r.t number of obstacles within vicinity ($\approx O(N^{2})$, since $S$, $P$ and $K$ can be fixed based on the computational capacity of agent’s processor). The algorithm is designed to effectively deal with dynamic and uncertain situations without global information. The proposed algorithm is practical in scenarios where acquiring global information is difficult and expensive, and may require huge on-board computational resources. For the cases where global information is available, it may be incorporated in the proposed algorithm by updating the target vector information.

VI. PERFORMANCE EVALUATION

In this section, experiments and results on the simulated scenarios are presented to illustrate the performance of the proposed algorithm. The static and dynamic exploratory experiments represent typical trajectory planning scenarios, and the experiments show the usability of the proposed algorithm. Through the uncertainty based simulated scenarios a rigorous evaluation of the proposed algorithm is conducted. A discussion on the performance of the proposed algorithm is provided at the end of each subsection.

A. PRELIMINARY STATIC EXPLORATORY SCENARIOS

1) SPIRAL SCENARIO

Setup: The agent is surrounded by the gray colored obstacles, enclosing the agent in a spiral maze, as shown in the Fig. (7). The objective of the agent is to have a collision free trajectory in order to move out of the maze and reach to its target location. Results: The algorithm was successful in finding a trajectory out of the spiral maze. The path traveled by agent is shown in red colored circles. From this experiment, it can be seen that the vicinity radius step of the proposed algorithm (Line 18 of Alg. 1) is very crucial. From the trial runs, it has been observed that for high vicinity radius at the beginning, the agent do not find feasible velocities. However, when the vicinity radius gets smaller during the execution of the algorithm, the agent identifies the feasible velocity and escapes the maze.

2) U-SHAPED SCENARIO WITH MOVING OBJECTS

Setup: The experiment involves trajectory planning in the presence of static (gray unnumbered circles) as well as dynamic obstacles (gray numbered circles). The static obstacles are positions in a U-Shaped structure around the agent (red color circle). Additionally, there are 2 moving obstacles having non-linear velocity profiles. Initially, Obstacle-1 is outside while Obstacle-2 and the agent are inside the U-shaped structure. The y-component of the velocity of Obstacle-1 is constant and is in negative y direction.

| Algorithm 1 Proposed Algorithm |
|--------------------------------|
| **Input:** |
| $v_{a} \leftarrow$ Agent’s current velocity; |
| $s_{a} \leftarrow$ Agent’s vicinity radius; |
| $p_{a} \leftarrow$ Agent’s current location; |
| $T_{a} \leftarrow$ Agent’s target location; |
| while $||T_{a} − p_{a}|| \geq \epsilon$ do |
| | **Data:** Obtain $a_{t_{0}}^{*}, a_{t_{0}}^{*}$ and $w_{f}$ for all $N$ obstacles present in the agent’s vicinity. |
| | **Estimate:** $a_{t_{0}}^{*}, a_{t_{0}}^{*}$ for all $N$; |
| | **Set:** $t_{0} = T_{a} − p_{a}, c = 0$; |
| | **Build:** $EVS \cup PVS$; |
| | **Update:** $EVS \leftarrow$ SORT($EVS$); |
| | $PVS \leftarrow$ SORT($PVS$); |
| | **Set:** $\Delta \leftarrow t_{0} \cup EVS \cup PVS$; |
| | for $i = 1$ to $|\Delta|$ do |
| | | **Check-1:** $c$ is feasible w.r.t Eq. (20); |
| | | **Check-2:** $c$ is feasible w.r.t. Eq. (17h); |
| | | if $\text{Check-1} \land \text{Check-2}$ are TRUE then |
| | | | **Set:** Set $c$ as per velocity rules; |
| | | | break; |
| | | if $(c = 0) \land (s_{a} \geq \xi)$ then |
| | | | $s_{a} \leftarrow$ Decrease$(s_{a})$; |
| | | | go-to Step-3; |
| | | else if $(c = 0) \land (s_{a} < \xi)$ then |
| | | | break; |
| | | $v_{a} \leftarrow c$; |
| | | $p_{a} \leftarrow \tau v_{a} + p_{a}$; |
while its x-component accelerates constantly in the positive x direction. Obstacle-2 has a sinusoidal velocity profile which moves it out of the U-shaped structure. The objective of the agent is to find a collision free trajectory in this dynamic environment and move itself out of the U-shaped structure.

**Results:** Local planning methods are typically greedy in nature and may face phenomenon of stalling as explained in Section IV-D. However, the inclusion of scaling mechanism (see Section V-C) enables the agent to handle scenarios similar to the one shown in Fig (7b). The instances of the moving obstacles and the agent are printed with the index number of each iteration to show their motion w.r.t time.

The above experiments show that the proposed algorithm works as expected in the simple scenarios containing one agent. In the following sections we design simulated scenarios to rigorously evaluate the proposed algorithm.

### B. MULTI-AGENT EXPLORATORY SCENARIOS

#### 1) MULTI-AGENTS WITH RANDOMLY MOVING OBSTACLES

The experiment is depicted in Fig. (8) & (9), where 4 agents are initially positioned at the corner of the square and their targets are located at the diagonally opposite corners. While heading for their target positions, agents have to avoid collisions with randomly positioned, randomly sized and randomly moving obstacles. **Setup:** The agents have radius of 30 cm. The agents have a maximum and minimum speed of 180 and 0 cm s\(^{-1}\) respectively. The normal speed of the agents is 140 cm s\(^{-1}\). For all the agents, the maximum vicinity radius is 600 cm, and the minimum vicinity radius is 400 cm. An agent starts the search for a feasible velocity with maximum vicinity radius. In the event of no feasible solution, the agent reduces its vicinity radius by an amount of 50 cm. The agent repeats the process until the agent finds a feasible velocity or the vicinity radius reaches to its minimum value (the agent’s velocity will be set as zero units when no feasible solution is obtained at the minimum vicinity radius). The cardinality of PVS is 20, and the cardinality of EVS is twice the number of obstacles & other agents present in the vicinity radius. The obstacles have radius uniformly between 30 and 50 cm. The obstacles have an average speed of 140 cm s\(^{-1}\), and their speed varies uniformly ±20% of their average speed. The number of obstacles is varied from 10 to 100. Two groups of experiments are conducted. In the first group, the square size is selected such that the Floor Space Factor (FSF) is 5 (mimicking the density in a typical office), and in the second experiment the FSF is 30 (mimicking the density in a typical warehouse).

**Results:** The results of this experiment are displayed in Table 1, where each row presents a summary of 30 replications. Column with header O corresponds to the number of obstacles. Columns with headers AT5 and AT30 display the Average Time per Agent (ATA) in seconds to reach the destination for FSF 5 and FSF 30 density case respectively. Column with header T5 displays a 6 letter sequence, the first three letters corresponds to ATA. A value of ‘Y’ indicates that there was a significant difference in the ATA values from the mean of the 30 values, and a value of ‘N’ indicates otherwise. Significance was measured by p-value test with 5% threshold. Similarly, the second (and the fifth) letter corresponds to number of Average Collisions per Agent (ACA), and the third (and the sixth) letter corresponds to the Total Time Steps (TTS) of the simulation.

**Discussion:** From Table 1, column T5, it can be stated that the ATA, ACATS and TTS values for a given number of obstacles are consistent, and do not vary significantly. In addition to that, from ACATS values in a row of C5 and C30, it can be seen
A color and line style coding similar to Fig. (10a) is used. per time step w.r.t the number of obstacles in the scenario. Furthermore, we can see that dense scenarios (FSF 5) take higher computational times than agents in their vicinity increases. Fig. (11) shows snapshots of the 30 replications for FSF 5 case. Similarly, the last three corresponds to FSF 30 case.

The graph shows that a small vicinity radius leads to high number of collisions. Where as higher vicinity radius leads to low number of collisions. Furthermore, we can see that dense scenarios (FSF 5) have higher collisions than the sparse scenarios (FSF 30).

From Fig. (10a) and Fig. (10b), it can be stated that selecting vicinity radius range is a crucial aspect of the proposed algorithm. Based on our trial experiments, a maximum radius of three times the max speed, and a minimum radius of 2 times the max speed are pragmatic choices.

### Table 1. Multi-agents with Randomly Moving Obstacles. O := the number of obstacles; AT5 and AT30 := the Average Time per Agent (ATA) in seconds to reach the destination for FSF 5 and FSF 30 density case respectively; C5 and C30 := the Average Collisions per Agent per Time Step (ACATS) for FSF 5 and FSF 30 density case respectively. TS5 and TS30 := the average number of total time steps for FSF 5 and FSF 30 density case respectively. Sig := First three letters indicate whether there was a significant difference (using p- value test) in the ATA, ACATS and TTS values from their corresponding mean values over the 30 replications for FSF 5 case. Similarly, the last three corresponds to FSF 30 case.

| O | AT5 | C5 | AT30 | C30 | Sig | TS5 | TS30 |
|---|-----|----|------|-----|-----|-----|------|
| 10 | 0.0442 | 0.0034 | 0.0222 | 0.0013 | NNNNNN | 26.43 | 28.57 |
| 20 | 0.0448 | 0.0075 | 0.0268 | 0.0009 | NNNNNN | 33.33 | 38.63 |
| 30 | 0.0590 | 0.0049 | 0.0258 | 0.0008 | NNNNNN | 41.73 | 43.10 |
| 40 | 0.0496 | 0.0061 | 0.0313 | 0.0008 | NNNNNN | 41.83 | 50.30 |
| 50 | 0.0477 | 0.0085 | 0.0269 | 0.0011 | NNNNNN | 41.70 | 53.17 |
| 60 | 0.0551 | 0.0067 | 0.0280 | 0.0007 | NNNNNN | 51.77 | 56.67 |
| 70 | 0.0634 | 0.0074 | 0.0324 | 0.0011 | NNNNNN | 56.70 | 65.23 |
| 80 | 0.0600 | 0.0069 | 0.0332 | 0.0015 | NNNNNN | 52.93 | 68.43 |
| 90 | 0.0634 | 0.0078 | 0.0348 | 0.0009 | NNNNNN | 56.23 | 70.37 |
| 100 | 0.0774 | 0.0073 | 0.0382 | 0.0012 | NNNNNN | 65.93 | 75.10 |

O:= the number of obstacles; AT5 and AT30 := the Average Time per Agent (ATA) in seconds to reach the destination for FSF 5 and FSF 30 density case respectively; C5 and C30 := the Average Collisions per Agent per Time Step (ACATS) for FSF 5 and FSF 30 density case respectively. TS5 and TS30 := the average number of total time steps for FSF 5 and FSF 30 density case respectively. Sig := First three letters indicate whether there was a significant difference (using p-value test) in the ATA, ACATS and TTS values from their corresponding mean values over the 30 replications for FSF 5 case. Similarly, the last three corresponds to FSF 30 case.

that a low density in general reduces the number of collisions. Furthermore, when the number of obstacles increases, then the ATA increases. There is approximately 75% increase in AT5 and 72% increase in AT30 as the number of obstacles increase from 10 to 100. Furthermore, there is approximately 150% increase in TS5 and 163% increase in TS30 as the number of obstacles increase from 10 to 100.

Fig. (10a) shows the trend in computational time per agent per time step w.r.t the number of obstacles present in the scenario. The curves in green color correspond to those scenarios where the initial vicinity radius is very high (all obstacles were visible to the agents) and the minimum radius is 400 cm. The curves in red color correspond to those scenarios where the initial vicinity radius is 600 cm and minimum radius is 400 cm. The curves in blue color correspond to those scenarios where the initial vicinity radius is 200 cm and minimum radius is 100 cm. The solid lines correspond to FSF 30 case, and the dashed lines correspond to FSF 5 case. The graph shows that as the number of agents in the vicinity increases, the computation time per agent also increases. This is due to the increase in the amount of data to be processed to obtain the collision free trajectories. However, if the agents vicinity is fixed to nearby agents, then the computational time converges to a specific value and it is independent of total number of agents in the entire environment. Furthermore, we can see that dense scenarios (FSF 5) take higher computational times than the sparse scenarios (FSF 30).

Fig. (10b) shows the trend in number of collisions per agent per time step w.r.t the number of obstacles in the scenario. A color and line style coding similar to Fig. (10a) is used.
agents resolved the dense configuration, and from this stage the density gradually decreases. Thus, this dynamic density scenario shows that ATA times increase by approximately 523% as we move from 10 agents in the scenario to 100 agents in the scenario. Furthermore, the number of time steps per agent increases by approximately 717% as we move from 10 agents in the scenario to 100 agents in the scenario. Thus, scenarios with dynamic densities containing bottlenecks are highly volatile to the number of agents. Such dynamic densities often arise in disaster management, specifically during
emergency evacuation. For example, during the emergency evacuation of a stadium, bottlenecks can be seen at the exit points/doors. Based on the application, agents might face similar scenario that require them to exit from bottlenecks. To sum, the proposed approach also works in applications that involves dynamic densities.

### C. MULTI-AGENT & UNCERTAINTY SCENARIOS

1) MULTI-AGENTS WITH UNCERTAINTY & MOVING OBSTACLES

This experiment is similar to Experiment VI-B.1 except that the agents have uncertain information about the environment. Specifically, the agents do not have crisp edge information of other agents and/or obstacles in the vicinity. Thus, the agents use multiple sensors, or multiple probes to get samples of the edges.

**Setup:** The agents and obstacles have similar characteristics as described in Experiment VI-B.1. The number of agents in this experiment are set to 4. The number of obstacles varies from 10 to 100. In addition to that, the number of samples per edge that an agent will evaluate varies from 2 to 10. In this experiment, the random edge samples are perturbations of the actual edges. The perturbations are normally distributed with $\mu = 0$ and $\sigma = 1/3$. Two groups of experiments are conducted. In the first group, the square size is selected such that the Floor Space Factor (FSF) is 5 (mimicking the density in a typical office), and in the second experiment the FSF is 30 (mimicking the density in a typical warehouse).

**Results:** The results of this experiment are displayed in Table 3, where each row presents a summary of 30 replications. Column with header $(O,K)$ corresponds to the number of obstacles and number of samples per edge. Columns with headers $AT5$ and $AT30$ display ATA to reach the destination for FSF 5 and FSF 30 density case respectively. Columns with headers $C5$ and $C30$ display ACATS for FSF 5 and FSF 30 density case respectively. Columns with headers $TS5$ and $TS30$ display ATSS for FSF 5 and FSF 30 density case respectively. Column with header $ComSig$ displays a 6 letter sequence, the first three letters corresponds to ATA 5 case, and the next three corresponds to ATA 30 case. The first (and the fourth) letter corresponds to ATA.

A value of ‘Y’ indicates there was a pairwise significant difference in the ATA values of the certainty case compared to the uncertainty case, and a value of ‘N’ indicates otherwise. Significance was measured by $p$-value test with 5% threshold. Similarly, the second (and the fifth) letter corresponds to number of Average Collisions per Agent (ACA), and the third (and the sixth) letter corresponds to the Total Time Steps (TTS) of the simulation.

**Discussion:** Most of the cases show that the difference in performance between ACATS (or ATSS) for certainty and uncertainty scenarios is statistically insignificant. However, it can be seen that there is a consistent significant difference in the ATA values, when then number of obstacles are high. Table 3 also indicates that there is a change in the ATA due to uncertainty. The maximum and minimum increase in ATA for a given number of obstacles is approximately 21% and 7% respectively. However, the maximum and minimum increase in ATA for a given number of sampled edges is approximately 210% and 147% respectively. Thus, the number of obstacles have high effect on ATA than compared to number of sampled edges.

2) MULTI-AGENTS WITH UNCERTAINTY ON A CIRCLE

This experiment is similar to Experiment VI-B.2 except that the agents have uncertain information about the environment. The uncertainty is emulated as described in

### TABLE 2. Performance of multi-agents on a circle. $A :=$ the number of agents on the circle; $AT :=$ the average time per agent (ATA) to reach the destination; $C :=$ the average collisions per agent per time step (ACATS).

| $A$ | $AT$ | $C$ | Sig | TS  |
|-----|------|-----|-----|-----|
| 10  | 0.03 | 0.01 | NN  | 11  |
| 20  | 0.02 | 0.00 | NN  | 22  |
| 30  | 0.03 | 0.00 | NN  | 33  |
| 40  | 0.06 | 0.02 | NN  | 44  |
| 50  | 0.10 | 0.01 | NN  | 55  |
| 60  | 0.12 | 0.00 | NN  | 66  |
| 70  | 0.16 | 0.01 | NN  | 77  |
| 80  | 0.19 | 0.01 | NN  | 88  |
| 90  | 0.21 | 0.01 | NN  | 99  |
| 100 | 0.22 | 0.01 | NN  | 110 |

$A :=$ the number of agents on the circle; $AT :=$ the Average Time per Agent (ATA) to reach the destination; $C :=$ the Average Collisions per Agent per Time Step (ACATS).

### TABLE 3. Performance of multi-agents with uncertainty & obstacles. $(O,K) :=$ the number of obstacles and number of samples per edge. $ComSig :=$ first three letters indicate whether there was a pairwise significant difference (using $p$-value test) in the ATA, ACATS and TTS values of certainty and uncertainty case over the 30 replications for FSF 5 case. Similarly, the last three correspond to FSF 30 case.

| $(O,K)$ | $AT5$ | $C5$ | $AT30$ | $C30$ | $ComSig$ |
|---------|-------|-----|--------|-------|---------|
| (20,2)  | 0.037 | 0.006 | 0.021 | 0.008 | NNYNNN |
| (20,4)  | 0.037 | 0.006 | 0.021 | 0.008 | NNYNNN |
| (20,6)  | 0.039 | 0.007 | 0.023 | 0.012 | NNYNNN |
| (20,8)  | 0.041 | 0.008 | 0.025 | 0.014 | NYNNNN |
| (20,10) | 0.033 | 0.006 | 0.025 | 0.012 | NNNYYN |
| (40,2)  | 0.064 | 0.007 | 0.030 | 0.016 | YNNNNN |
| (40,4)  | 0.059 | 0.007 | 0.035 | 0.002 | NNYYNN |
| (40,6)  | 0.056 | 0.007 | 0.041 | 0.005 | NNYYNN |
| (40,8)  | 0.056 | 0.007 | 0.039 | 0.007 | NNYYNN |
| (40,10) | 0.054 | 0.006 | 0.040 | 0.005 | NNYYNN |
| (60,2)  | 0.069 | 0.007 | 0.040 | 0.012 | NNNYNN |
| (60,4)  | 0.066 | 0.008 | 0.038 | 0.001 | NNNYNN |
| (60,6)  | 0.070 | 0.007 | 0.038 | 0.001 | NNNYNN |
| (60,8)  | 0.071 | 0.007 | 0.039 | 0.013 | NNYNNN |
| (60,10) | 0.068 | 0.006 | 0.038 | 0.001 | NNYNNN |
| (80,2)  | 0.077 | 0.008 | 0.045 | 0.008 | YNNNNN |
| (80,4)  | 0.083 | 0.008 | 0.045 | 0.001 | NNYNNN |
| (80,6)  | 0.086 | 0.003 | 0.048 | 0.004 | NNYNNN |
| (80,8)  | 0.086 | 0.003 | 0.048 | 0.001 | NNYYNN |
| (80,10) | 0.083 | 0.006 | 0.049 | 0.002 | NNYYNN |
| (100,2) | 0.097 | 0.009 | 0.045 | 0.008 | NNYYNN |
| (100,4) | 0.092 | 0.008 | 0.048 | 0.009 | NNNYNN |
| (100,6) | 0.101 | 0.007 | 0.049 | 0.014 | NNNYNN |
| (100,8) | 0.096 | 0.009 | 0.050 | 0.009 | NNNYNN |
| (100,10)| 0.102 | 0.010 | 0.051 | 0.001 | NNYYNN |

$(O,K) :=$ the number of obstacles and number of samples per edge. $ComSig :=$ First three letters indicate whether there was a pairwise significant difference (using $p$-value test) in the ATA, ACATS and TTS values of certainty and uncertainty case over the 30 replications for FSF 5 case. Similarly, the last three correspond to FSF 30 case.
Experiment VI-C.1. Setup: The agents and obstacles have similar characteristics as described in Experiment VI-B.1. In addition to that, the number of samples per edge that an agent will evaluate varies from 2 to 10. In this experiment, the random edge samples are perturbations of the actual edges. The perturbations are normally distributed with $\mu = 0$ and $\sigma = 1/3$. Results: The results of this experiment are displayed in Table 4, where each row presents a summary of 30 replications. Column with header $(A,K)$ corresponds to the number of agents on the circle and the number of samples per edge. Column with header AT displays ATA to reach the destination. Column with header C displays ACATS. Column with header ComSig displays a 3 letter sequence. The first letter corresponds to ATA. A value of ‘Y’ indicates there was a pairwise significant difference in the ATA values of the certainty case compared to the uncertainty case, and a value of ‘N’ indicates otherwise. Significance was measured by $p$-value test with 5% threshold. Similarly, the second letter corresponds to number of Average Collisions per Agent (ACA), and the third letter corresponds to the Total Time Steps (TTS) of the simulation. Discussion: Most of the cases show that the difference in performance between ACATS (or ATSS) for certainty and uncertainty scenarios is statistically insignificant. However, it can be seen that there is a consistent significant difference in the ATA values, when the number of agents is high. This result can be attributed to uncertainty with increase in number of agents. The maximum and minimum increase in ATA for a given constant number of agents is approximately 17% and 4%, respectively. However, the maximum and minimum increase in ATA for a given number of sampled edges is approximately 420% and 340%, respectively. Thus, the number of agents have high effect on ATA compared to the number of sampled edges.

TABLE 4. Performance of Multi-agents with Uncertainty on a Circle. $(A,K)$ := the number of agents on the circle and the number of samples per edge.

| $(A,K)$ | AT   | C    | ComSig |
|--------|------|------|--------|
| (20,2) | 0.0209 | 0.0057 | YNY    |
| (20,4) | 0.0226 | 0.0056 | NNN    |
| (20,6) | 0.0237 | 0.0063 | NNN    |
| (20,8) | 0.0247 | 0.0058 | NNN    |
| (20,10)| 0.0229 | 0.0063 | NNN    |
| (40,2) | 0.0515 | 0.0024 | YNN    |
| (40,4) | 0.0495 | 0.0026 | YNN    |
| (40,6) | 0.0497 | 0.0025 | YNN    |
| (40,8) | 0.0525 | 0.0023 | YNN    |
| (40,10)| 0.0493 | 0.0028 | YNN    |
| (60,2) | 0.0732 | 0.0018 | YYN    |
| (60,4) | 0.0739 | 0.0018 | YYN    |
| (60,6) | 0.0758 | 0.0017 | YYN    |
| (60,8) | 0.0737 | 0.0019 | YYN    |
| (60,10)| 0.0737 | 0.0019 | YYN    |
| (80,2) | 0.0942 | 0.0015 | YNN    |
| (80,4) | 0.0963 | 0.0014 | YNN    |
| (80,6) | 0.0979 | 0.0014 | YNN    |
| (80,8) | 0.098  | 0.0014 | YNN    |
| (80,10)| 0.1073 | 0.0015 | YYN    |
| (100,2)| 0.1181 | 0.0011 | YNN    |
| (100,4)| 0.1211 | 0.0011 | YNN    |
| (100,6)| 0.1175 | 0.0012 | YNN    |
| (100,8)| 0.1179 | 0.0011 | YNN    |
| (100,10)| 0.1193| 0.0011 | YNN    |

VII. CONCLUSION
The mathematical programming approaches available in the literature for motion planning are computationally very expensive. Secondly, these methods typically require global information, which may not be easily available in many real life applications. Some reactive and sampling based methods try to reduce the above complexities, but they typically face issues related to finding efficient trajectories, and/or getting stuck at local minima. A novel heuristic approach has been presented in this paper, which not only finds collision free trajectories in multi-agent dynamic scenarios, but also handles uncertainty in collected data samples. All this is achieved with low computational effort. Both position as well as velocity information of the surroundings are used in the proposed approach. Such combination may produce smooth trajectories for small step sizes. This combination also performs effectively at high level of sensor noise. The proposed algorithm has been examined for a wide range of scenarios including typical office, warehouse and dynamic density scenarios. From the experimental analysis, it can be concluded that the effect of number of sampled edges is very low on average processing time per agent compared to the effect of number of obstacles and/or agents present in the vicinity. Moreover, selecting appropriate range for agent’s vicinity radius is crucial in simultaneously reducing processing time and number of collisions. Based on the above observations, it can be stated that the proposed algorithm is suitable for real-time multi-agent trajectory planning under uncertainty.

REFERENCES
[1] J. V. Breakwell, “The optimization of trajectories,” J. Soc. Ind. Appl. Math., vol. 7, no. 2, pp. 215–247, 1959.
[2] H. M. Robbins, “Optimality of intermediate-thrust arcs of rocket trajectories,” AIAA J., vol. 3, no. 6, pp. 1094–1098, June 1965.
[3] K. Shin and N. Mckay, “A dynamic programming approach to trajectory planning of robotic manipulators,” IEEE Trans. Autom. Control., vol. 31, no. 6, pp. 491–500, Jun. 1986.
[4] M. K. Jouaneh, Z. Wang, and D. A. Dornfeld, “Trajectory planning for coordinated motion of a robot and a positioning table. i. Path specification,” IEEE Trans. Robot. Autom., vol. 6, no. 6, pp. 735–745, Dec. 1990.
[5] V. Rajan, “Minimum time trajectory planning,” in Proc. IEEE Int. Conf. Robot. Autom., vol. 2, Mar. 1985, pp. 759–764.
[6] J. Angeles, A. Rojas, and C. Lopez-Cajun, “Trajectory planning in robotic continuous-path applications,” IEEE J. Robot. Autom., vol. RA-4, no. 4, pp. 380–385, Aug. 1988.
[7] S. M. LaValle, Planning Algorithms. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[8] D. A. Schoenwald, “AUVs: In space, air, water, and on the ground,” IEEE Control Syst. Mag., vol. 20, no. 6, pp. 15–18, Dec. 2000.
[9] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” Int. J. Robot. Res., vol. 17, no. 7, pp. 760–772, Jul. 1998.
[10] J. Canny and J. Reif, “New lower bound techniques for robot motion planning problems,” in Proc. 28th Annu. Symp. Found. Comput. Sci. (SFCS), Oct. 1987, pp. 49–60.
[11] F. Augugliaro, A. P. Schoellig, and R. D’Andrea, “Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2012, pp. 1917–1922.
I. Châari, A. Koubâa, S. Trigui, H. Bennaceur, A. Ammar, and E. Rimon and D. E. Koditschek, “Exact robot navigation using artificial potential functions,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2015, pp. 5954–5961.

T. Schouwenaars, B. De Moor, E. Feron, and J. How, “Mixed integer programming for multi-vehicle path planning,” in Proc. Eur. Control Conf. (ECC), Sep. 2009, pp. 2603–2608.

A. Richards, T. Schouwenaars, J. P. How, and E. Feron, “Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming,” J. Guid., Control, Dyn., vol. 25, no. 4, pp. 755–764, Jul. 2002.

J. Peng and S. Akella, “Coordinating multiple robots with kinodynamic constraints along specified paths,” Int. J. Robot. Res., vol. 24, no. 4, pp. 295–310, 2005.

D. Mellinger, A. Kushleyev, and V. Kumar, “Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams,” in Proc. IEEE Int. Conf. Robot. Autom., May 2012, pp. 477–483.

D. Hsi, J.-C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” Int. J. Comput. Geom. Appl., vol. 09, no. 04n05, pp. 495–512, Aug. 1999.

C. M. Clark, “Probabilistic road map sampling strategies for multi-robot motion planning,” Robot. Auto., vol. 53, nos. 3–4, pp. 244–264, Dec. 2005.

S. M. Lavalle, “Rapidly-exploring random trees: A new tool for path planning,” Tech. Rep., 1998.

S. M. LaValle and J. J. Kuffner, Jr., “Rapidly-exploring random trees: Progress and prospects,” Tech. Rep., 2000.

P. Cheng, Z. Shen, and S. La Valle, “Rt-based trajectory design for autonomous automobiles and spacecraft,” Arch. Control Sci., vol. 11, nos. 3–4, pp. 167–194, 2001.

O. Adiyatov and H. A. Varol, “Rapidly-exploring random tree based memory efficient motion planning,” in Proc. IEEE Int. Conf. Mechatronics Autom., Aug. 2013, pp. 354–359.

R. Kala, “Rapidly exploring random graphs: Motion planning of multiple mobile robots,” Adv. Robot., vol. 27, no. 14, pp. 1113–1122, Oct. 2013.

O. Rouissi, R. Benalounis, D. Devizier, A. Aritha, N. Belanger, and P. Feyzue, “Path planning: A 2013 survey,” in Proc. Ind. Eng. Syst. Manage. (ISEM), Oct. 2013, pp. 1–8.

P. Svetska and M. H. Overmars, “Coordinated path planning for multiple robots,” Robot. Autom. Syst., vol. 23, no. 3, pp. 125–152, Apr. 1998.

J. Carsten, D. Ferguson, and A. Stentz, “3D field D: Improved path planning and replanning in three dimensions,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2006, pp. 3381–3386.

R. Kala, “Multi-robot path planning using co-evolutionary genetic programming,” Expert Syst. With Appl., vol. 39, no. 3, pp. 3817–3831, Feb. 2012.

M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur, and A. Ammar, “Global path planning for mobile robots in large-scale grid environments using genetic algorithms,” in Proc. IEEE Conf. Individual Collective Behaviors Robot. (ICBR), Dec. 2013, pp. 1–8.

J. Châari, A. Koubâa, H. Bennaceur, A. Ammar, S. Trigui, M. Tounsi, E. Shakhshuki, and H. Youssf, “On the adequacy of tabu search for global robot path planning problem in grid environments,” Procedia Comput. Sci., vol. 32, pp. 604–613, Jun. 2014.

I. Châari, A. Koubâa, S. Trigui, H. Bennaceur, A. Ammar, and K. Al-Shafran, “SmartPATH: An efficient hybrid ACO–GA algorithm for solving the global path planning problem of mobile robots,” Int. J. Adv. Robotic Syst., vol. 11, no. 7, p. 94, Jul. 2014.

O. Khaiti, “Real-time obstacle avoidance for manipulators and mobile robots,” in Autonomous Robot Vehicles. Springer, 1986, pp. 390–404.

E. Rimon and D. E. Koditschek, “Exact robot navigation using artificial potential functions,” IEEE Trans. Robot. Autom., vol. 8, no. 5, pp. 501–518, Oct. 1992.

Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in Proc. IEEE Int. Conf. Robot. Autom., Dec. 2002, pp. 1398–1404.

J. Borenstein and Y. Koren, “Real-time obstacle avoidance for fast mobile robots,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 19, no. 5, pp. 1179–1187, Sept./Oct. 1989.

J.-H. Chuang and N. Ahuja, “An analytically tractable potential field model of free space and its application in obstacle avoidance,” IEEE Trans. Syst., Man, Cybern. B (Cybern.), vol. 28, no. 5, pp. 729–736, Oct. 1998.

HASEEB TAHIR received the M.S. degree in industrial and systems engineering from the King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. His research interests include applied optimization in decision making, operations research, and lean management.

H. Tahir et al.: Heuristic Approach for Real-Time Multi-Agent Trajectory Planning Under Uncertainty.

VOLUME 8, 2020 3825
MUJAHID N. SYED received the Ph.D. degree in operations research from the Industrial and Systems Engineering Department, University of Florida, in 2013. Since 2013, he has been an Assistant Professor with the Department of Systems Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

UTHMAN BAROUDI received the B.Sc. and M.S. degrees from the King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 1988 and 1990, respectively, and the Ph.D. degree from Concordia University, Montreal, Canada, in 2000, all in electrical engineering. In 2000, he joined Nortel Networks, Ottawa, Canada, to work in Research and Development for the next generation wireless networks. He is currently an Associate Professor with the Department of Computer Engineering, KFUPM. His research interests cover in the areas of cloud robotics networks, network design for the IoT, wireless adhoc, as well as sensor and actuator networks. He has over 100 publications in refereed journal and conference proceedings, and holds 28 US patents.