Collision-free Encoding for Chance-constrained, Non-convex Path Planning

Marcio da Silva Arantes, Claudio F. M. Toledo, Brian Charles Williams, Masahiro Ono

***Abstract*—The path planning methods based on non-convex constrained optimization, such as mixed-integer linear programming (MILP), have found various important applications, ranging from unmanned aerial vehicles (UAVs) [1] and autonomous underwater vehicles (AUVs) [2] to space vehicles [3]. Moreover, their stochastic extensions have enabled risk-aware path planning, which explicitly limits the probability of failure to a user-specified bound. However, a major challenge of those path planning methods is constraint violation between discrete time steps. In the existing approach, a path is represented by a sequence of waypoints and the safety constraints (e.g., obstacle avoidance) are imposed on waypoints. Therefore, the trajectory between waypoints could violate the safety constraints. A naive continuous-time extension results in unrealistic computation cost. We propose a novel approach to ensure constraint satisfaction between waypoints without employing a continuous-time formulation. The key idea is to enforce that the same inequality constraint is satisfied on any two adjacent time steps, under assumptions of polygonal obstacles and straight line trajectory between waypoints. The resulting problem encoding is MILP, which can be solved efficiently by commercial solvers. Thus, we also introduce novel extensions to risk-allocation path planners with improved scalability for real-world scenarios and run-time performance. While the proposed encoding approach is general, the particular emphasis of this paper is placed on the chance-constrained, non-convex path-planning problem (CNPP). We provide extensive simulation results on CNPP to demonstrate the path safety and scalability of our encoding and related path planners.**

***Index Terms*—Autonomous agents, chance constraints, optimization under uncertainty, probabilistic planning, mixed-integer linear programming.**

I. Introduction

Real-world path planning problems often have complex, non-convex state constraints for the sake of safety. Take, for example, the airspace around Tokyo's two airports shown in Fig. 1, which constitute the third busiest city airport system in the world. Commercial airplanes take off from and land on the airports must keep off two airspaces reserved for nearby military airbases, shown in green and blue in the figure. The airspaces have a non-convex, three-dimensional shape, where the ceiling height varies discontinuously like stairs. Various other safety constraints must also be respected, such as spacial separation between airplanes and the scheduling constraints of runways. Failure to satisfy these constraints could result in critical consequences.

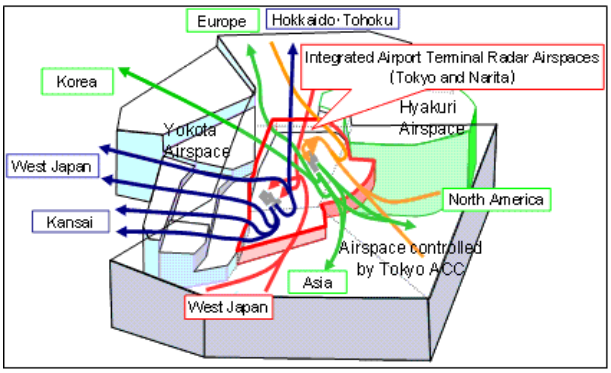


Fig. 1. Airspace around Tokyo, Japan. 3-D constraints. Source: Japanese Ministry of Land, Infrastructure, Transport and Tourism website.

An efficient path planning approach for dynamic systems to handle nonconvex state constraints is to formulate the problem as model predictive control (MPC) [4]. It plans discrete-time control and state sequences concurrently through constrained optimization. The optimization problem to solve is mixed integer linear programming (MILP) when the dynamics is linear, obstacles are represented by combinations of polytopes, and no uncertainty is present. If the system is subject to stochastic uncertainty, a viable approach is path planning with chance-constraints [5], which cap the probability of constraint violations.

An outstanding problem of this approach is "jumps" between time steps. Since the approach only concerns the constraint satisfaction at discrete points of time, the resulting path could cut through obstacles as shown in Fig. 2-Left. In fact, such undesirable "jumps" are commonly observed both in deterministic and stochastic settings because "jumps" often help to reduce typical cost functions, such as the path length and the control effort. This issue could be mitigated, but not fully addressed, by increasing sampling frequency (i.e., reduced time intervals between time steps) at a cost of increased computation time. Another existing approach is to set safety margin around the obstacles [6], but it introduces conservatism. More importantly, this approach works only when the distance between neighbor states is upper-bounded.

The contribution of this paper is two-fold. First, we propose a simple yet powerful approach to fully suppress the "jumps"

Marcio da Silva Arantes and Claudio F. M. Toledo are with University of Sao Paulo, Institute of Mathematics and Computer Science (e-mail: marcio@icmc.usp.br and claudio@icmc.usp.br).

Brian Charle Williams is with Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory (e-mail: williams@csail.mit.edu).

Masahiro Ono is with California Institute of Technology, Jet Propulsion Laboratory (e-mail: masahiro.ono@jpl.nasa.gov).

Manuscript received YYY XX, 20XX; revised YYY XX, 20XX.

This research was supported by Fundacao de Amparo a Pesquisa do Estado de Sao Paulo (FAPESP), projects 2014/12297-0, 2014/11331-0 and 2013/07375-0, and by the Office of Naval Research Grant N00014-15-IP-00052. The research described in this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

without incurring significantly increased computation time or conservatism. The main idea is to impose constraints that require every pair of adjacent states to be on the same "side" of an obstacle, as shown in Fig. 2-Right. The original set of obstacle avoidance constraints are replaced by the new set of constraints as the latter is an sufficient condition of the former.

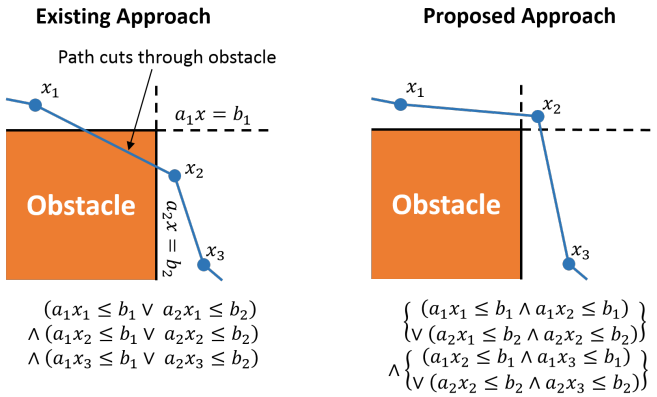


Fig. 2. Left: Conventional MPC-based path planning requires that states are outside of polygonal obstacles at each time step. As a result, path between time steps could cut through obstacles. Right: Instead, the proposed approach requires that each state pair at adjacent time steps are on the same side of the obstacles. The resulting path is guaranteed to be outside of the obstacles even between time steps. In the figure, the two sides of the obstacle are part of hyperplanes represented by $a_ix = b_i$. A state x is outside of the obstacle if $a_1x \le b_1 \vee a_2x \le b_2$.

The second contribution of this paper is to extend this approach to chance-constrained, non-convex path planning (CNPP) improving scalability and run-time performance for real-world scenarios. We specifically build upon the risk-allocation approach, where the allocation of risk among time steps and constraints are optimized through iterative optimization in order to minimize the conservatism in the handling of chance constraints [7]. From the linear encoding proposed, extensions to risk-allocation path planners are improved as well as a novel risk-allocation path planner is introduced.

The primary limitation of our approach is the assumptions of polygonal obstacles and straight-line paths between waypoints, which are standard assumptions employed by most MPC-based path planning methods. These assumptions are justified by the fact that, in many practical applications, if not all, vehicle paths and keep-out/keep-in zones consist of simple geometries for operational convenience. For example, typical no-fly zones in airspace, such as the ones in Fig 1, are polygonal. Keep-in zones for Mars rovers' on-board path planning are also polygonal, as shown in Fig. 3-Left. In various applications, ranging from aircraft to Mars rovers, paths are often composed of straight lines and sharp turns, or turns in place in case of Mars rovers (Fig. 3-Right). In these domains, the contribution of this paper has a practical importance because it provides a safety guarantee along a continuous path, with the comparable computational complexity as the discrete-time path optimization, in both deterministic *and* stochastic settings.

The remainder of this paper is organized as follows: Section 2 reviews the main pieces of prior work that are related to this

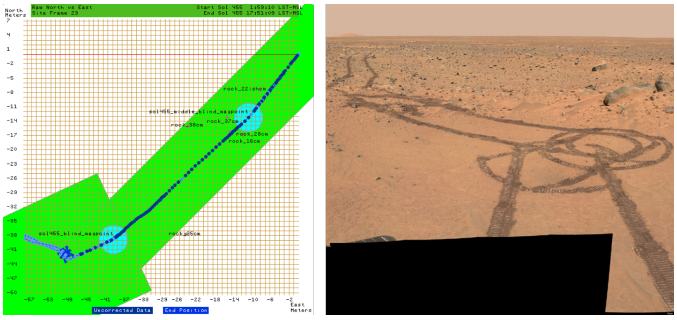


Fig. 3. Left: Polygonal keep-in zones for the autonomous path planning of Mars Rover Curiosity on Sol 455. Right: While Mars rovers have capabilities to drive on constant-curvature arcs, actual paths often consist of straight lines and turns in place, as shown in this figure.

study. We formally define CNPP in Section 3, and then present the proposed constraint encoding in Section 4. The solution methods are described in Section 5 and the simulation results are reported in Section 6. Finally, the conclusions of this work follows in Section 7.

II. Related Work

Research on autonomous vehicles has become increasingly relevant in recent years to real world problems [8], [9], [10]. There are several papers modeling path planning problems for autonomous vehicles, where features like continuous and discrete actions, convex and non-convex scenarios, and chance-constraints are approached.

A sequence of discrete actions and optimal continuous control values are generated by a system called Kongming in [11], aiming to accomplish missions with an autonomous vehicle. The authors propose a method with two innovations. First, the strengths of a planning graph [12] for encoding discrete actions is combined with flow tubes [13] for encoding continuous actions. Second, a formal language specifying temporally extended goals [14] over continuous and discrete actions is introduced. The author in [15] describes a system for autonomous vehicles that is able to make decisions without an human operator. The system is evaluated simulating a space exploration scenario in Mars. The author in [16] presents $tBurton$, a domain-independent temporal planner for complex network systems. $tBurton$ handles a set of problem features and it is able to verify whether a plan is temporally consistent.

Non-convex scenarios are approached in [6], where no-fly-zones are modelled as obstacles and path planning between waypoints is discussed aiming obstacle avoidance. The main idea is to expand obstacle sizes during the estimation and trajectory phases of the proposed approach. Blackmore and Ono [17] define a convex model for path planning. The autonomous vehicle can keep a safety distance from boundaries through the convex region.

Path planning with chance-constraints is incorporated by [18], making possible to plan a path within an acceptable level of risk for non-convex scenarios. Uncertainty about lower and upper bound limits for time mission are addressed by [19]. The path planning problem for autonomous underwater vehicle is studied in [20] by describing it as a stochastic

single-vehicle routing problem. A chance-constrained solution method is proposed to solve it, based on the previous existence of roadmap.

The present paper introduces a mixed-integer linear programming (MILP) model for autonomous vehicle path planning. The proposed encoding approach is general, but particular emphasis is placed on the chance-constrained, non-convex path-planning problem (CNPP) introduced in [18]. Therefore, the MILP model will be described taking into account the dynamic system from an Unmanned Aerial Vehicle (UAV). There are several works that address UAV path planning, where vehicle behavior is represented as linear dynamic systems [18], [11], [3], [5]. These formulations are typically based on mixed integer linear programming (MILP) [21], [1], [18] or mixed integer quadratic programming (PQIM) [3], [5].

A MILP is proposed in [21] for path planning aiming to minimize fuel consumption in autonomous vehicles. As already mentioned, the authors in [18] study a path planning problem within non-convex region and with risk allocation to avoid obstacles. The authors encoded this problem as a stochastic non-convex and non-linear mathematical programming model. A heuristic that combines upper and lower bounds solutions, based on linear mathematical programming techniques, is also introduced.

Most of the approaches cited are based on a discrete-time formulation. A crucial problem is that this formulation can produce vehicle paths that exceed the risk bounds specified by the problem formulation, since it is not considered the behavior of the vehicle between discrete time points. For example, using the formulation described in [18], [3], [5] excessive risk can occur when the vehicle position at two successive waypoints is feasible, but the line between the two positions intersect an obstacle.

In addition, these prior methods can be computationally prohibitive when the scenario includes a large number of obstacles and a long time horizon. The authors in [3] and [5] describe solution methods applied to real-world environments using UAVs, autonomous underwater vehicles and spacecraft. Scalability is a problem for such solution methods as reported in [5]. The run time growths exponentially following the increase of the number of obstacles, leading the authors to left the problem of deal with large number of obstacles as future work.

III. Problem Statement

Let's suppose we have an UAV that starts from one airport and reaches another airport located behind some mountains as shown by Figure 4. It is not desired that such aircraft fly over populated areas, regular airports or storm regions, these are specified as no-fly zones (NFZ). The aircraft can fly, for example, over forests and mountains, but uncertainties related to the environment and the UAVs dynamics can lead to disturbances during the flight. A path planner that operates under this situation must contend with a certain level of risk that the UAV goes through a NFZ or collides with a mountain. The task of the planner is to find a path for the UAV that terminates on the destination, while optimizing a measure such as fuel consumption or distance, without violating a maximum level of allowed risk. In our treatment here, NFZs are viewed as obstacles with unlimited height (h[∞]), and avoidance of these regions is treated as a hard constraint. Mountains and forests are also considered obstacles, but with limited height, as illustrated by Figure 4, hence it is possible for the UAV to fly over these obstacles. Given this treatment, path planning is performed over a non-convex region, represented by obstacles, along with a maximum acceptable risk of going through such obstacles. We call this a Chance-constraint Non-convex Path-planning Problem (CNPP) as first described in [18].

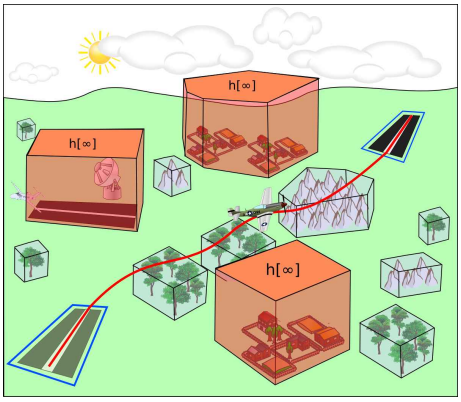


Fig. 4. Path planning through non-convex regions.

The inputs of the CNPP include initial position ($\hat{x}_0$), goal position ($\hat{x}_{goal}$), a stochastic plant model (SPM), time horizon (T), time interval (Δt), external uncertainties w_t and level of risk (Δ). The decision variables are system states (x_t) and controls (u_t) applied at each time step (t) to change these states in the plant (physical system) following the SPM. Thus, x_t defines the state of the plant at time step t, while u_t is the action or control applied over x_t that changes the current state. The probabilistic state transitions in the SPM are approximated by a discrete and linear recursive equation over x_t and u_t, where external uncertainties are added by w_t. A convex objective function g($\cdot$) is assumed over states x_t and controls u_t. The output of the CNPP is the optimal sequence of controls $S = (u_0^*, ..., u_T^*)$ applied over the system that is able to reach a goal position, without violating the level of risk, minimizing some measure defined by g($\cdot$).

IV. Problem Encoding Improvements

This section introduces the improvements proposed to encode the CNPP defined on section III, which are described over the previous encoding introduced in [18].

A. Control plant and chance-constraints

First, the time horizon T is divided into time steps $t = 0, ..., T$, using a fixed time interval Δt between consecutive times steps. These time steps guide the vehicle from its initial state to its final state (goal position), while avoiding obstacles $j = 1, ..., J$. We assume that the initial position follows a Gaussian distribution with mean $\hat{x}_0$ and covariance matrix Σ_{x_0}, so $x_0 \sim \mathcal{N}(\hat{x}_0, \Sigma_{x_0})$. Moreover, uncertainties in the

plant model are related to the initial position and external factors w_t, which act over time in the system dynamic with $w_t \sim \mathcal{N}(0, \Sigma_{w_t})$.

Thus, the authors in [18] defined exactly the distribution of the future states following constraints (1) as stochastic plant model SPM. The mean μ_t of the current state x_t is linearly defined from control inputs $u_0, ..., u_{t-1}$, while the covariance at x_t (Σ_t) is not related to previous control inputs by equation (2). The covariance is determined from the initial state covariance (Σ_{x_0}) and noise covariances (Σ_{w_t}), which are previously known. The mean of the final state of the vehicle has to reach the goal state following constraint (3).

$$\mu_t = A^t \hat{x}_0 + \sum_{s=0}^{t-1} A^{t-s-1} B u_s \qquad \forall(t) \tag{1}$$

$$\Sigma_t = A^t \Sigma_{x_0} (A^T)^t + \sum_{s=0}^{t-1} A^s \Sigma_{w_t} (A^T)^s \qquad \forall(t) \tag{2}$$

$$\mu_T = \hat{x}_{goal} \tag{3}$$

The chance-constraint in Equation (4) imposes a constraint on acceptable probability of flying over a NFZ and hitting an obstacle. The obstacles are defined by using disjunctive sets G_j of linear constraints, where $\bigwedge$ and $\bigvee$ are the logical *AND* and *OR* operators. Figure 5 depicts an obstacle avoidance, where the vehicle's position must be outside of at least one surface.

$$Pr(\bigwedge_{t=0}^{T} \bigwedge_{j=1}^{J} \bigvee_{i \in G_j} a_i^T x_t \geq b_i) \geq 1 - \Delta \tag{4}$$

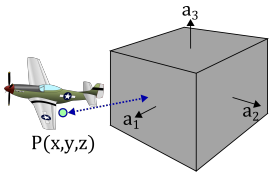


Fig. 5. Vehicle's position must be outside of at least one surface

This constraint guarantees that the level of risk incurred, when planning a path, will not violate the allowed risk bound Δ in the CNPP. However, for a Gaussian probability distribution function $Pr(.)$, constraint (4) becomes non-linear. In general, the problem involves a multivariate Gaussian random variable x_t for the vehicle state at time step t, with variable mean μ_t and known covariance matrix Σ_t. In this case, the chance constraint can be reformulated to equations (5)-(7).

$$Pr(a_i^T x_t - b_i < 0) \leq \delta_{jt} \quad \Leftrightarrow \quad a_i^T \mu_t - b_i \geq c_{i,t}(\delta_{jt}) \tag{5}$$

$$c_{i,t}(\delta_{jt}) = erf^{-1}(1 - 2\delta_{jt}) \sqrt{2 a_i^T \Sigma_t a_i} \tag{6}$$

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{7}$$

Equation (5) reformulates the chance-constraint as a inequality over mean μ_t and a safety margin $c_{i,t}(\delta_{jt})$ that is a function of risk allocation δ_{jt}. At this point two aspects must be considered. First, to guarantee obstacle avoidance, the vehicle must be on the "outside" of at least one side of the obstacle ($\bigvee_{i \in G_j} a_i^T x_t \geq b_i$). Second, the safety margin is determined by a non-linear function, given by equations (6) and (7). Note that erf^{-1} is non-convex. To restrict erf^{-1} to a convex region, we limit the risk-bound in equation (6) to a probability of failure of less than 50% ($\Delta \leq 0.5$).

The obstacle avoidance can be encoded by using binary variables (Z_{jti}) to specify the obstacle side for which the vehicle maintains a positive distance. This is achieved by constraints (8) and (9) where $\bar{M}$ is a positive big number. Constraints (8) engage the safety margin for the selected obstacle side ($Z_{jti} = 1$), while constraints (9) guarantee that this will happen for at least one side of each obstacle. Constraint (10) will satisfy the global chance constraint, where the sum of each risk δ_{jt} allocated to obstacle j at time step t can not exceed the allowed risk Δ.

$$c_{i,t}(\delta_{jt}) + b_i - a_i^T \mu_t \leq \bar{M}(1 - Z_{jti}) \qquad \forall(j, t, i \in G_j) \tag{8}$$

$$\sum_{i \in G_j} Z_{jti} \geq 1 \qquad \forall(j, t) \tag{9}$$

$$\sum_j \sum_t \delta_{jt} \leq \Delta \tag{10}$$

The authors in [18] did not deal with non-linearity when solving CNPP from their full encode; instead, relaxation approaches were proposed. Also, constraints (8) and (9) can lead to vehicle trajectories that exceed risk bounds specified in the problem formulation. The next sections describes improvements to solve such issues.

B. Improvements over chance constraints

As mentioned in section I an issue arises in the soundness of this formulation due to the discrete time formulation. It is possible to have a path, chosen, for example, to save fuel, that incurs a total risk greater than the maximum allowed risk Δ. Figure 6(a) shows another, more extreme situation, while Figure 6(b) illustrates a possible solution that restores the risk assessment to a conservative bound.

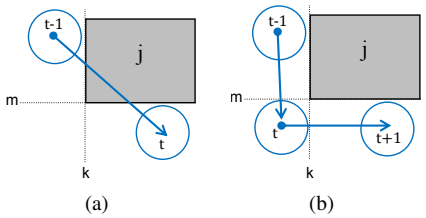


Fig. 6. (a) The vehicle could go trough the obstacle (b) Vehicle can not go trough it.

The previous encoding from constraints (8) and (9) in section IV-A under estimates risk, where a constraint is enabled at one time step and becomes disabled at the next time step. For example, as shown by Figure 6(a), suppose that at time step $t-1$, constraint related to side k is enabled and constraint m is disabled for obstacle j with $Z_{j,t-1,k} = 1$ and $Z_{j,t-1,m} = 0$. Furthermore, assume that at the next time step t, they switch values $Z_{j,t,k} = 0$ and $Z_{j,t,m} = 1$, . Constraints (8) and (9) are not violated, as presented by expressions (11), thus,

no constraint prevents the vehicle from moving through the obstacle.

$$\left\{\begin{array}{ccc} t-1 & : & \sum_{i\in G_j} Z_{j,t-1,i} = Z_{j,t-1,k} = 1 \geq 1 \\ & & c_{k,t-1}(\delta_{jt-1}) + b_{jk} - a_{jk}^T \mu_{t-1} \leq 0 \\ t & : & \sum_{i\in G_j} Z_{jti} = Z_{jtm} = 1 \geq 1 \\ & & c_{m,t}(\delta_{jt}) + b_{jm} - a_{jm}^T \mu_t \leq 0 \\ & & \textbf{feasible} \end{array}\right\} \quad (11)$$

To avoid such a situation, if a constraint k is enabled at time step $t-1$ ($Z_{j,t-1,k} = 1$), we introduce an additional condition that requires constraint k to continue to hold up through time t ($Z_{j,t,k} = 1$). This imposes the constraint that the vehicle must remain outside of constraint k between time steps $t-1$ and t, thus leading the vehicle along the corresponding edge of the obstacle (Figure 6(b)), instead of jumping from one side directly to another side (Figure 6(a)). With the added requirement that constraint k persist between the successive time points, the probability of failure estimate is guaranteed to be conservative across continuous time. We use linear constraints to encode such obstacle avoidance, as stated next.

Lemma. $\forall j, \forall t > 0$
$\bigvee_{i\in G_j} (Z_{j,t,i} \wedge Z_{j,t-1,i}) = 1 \Leftrightarrow$ *Constraints (12)-(15) hold.*

$$\sum_{i\in G_j} p_{j,t,i} \geq 1 \quad (12)$$

$$p_{j,t,i} \geq Z_{j,t,i} + Z_{j,t-1,i} - 1 \quad (13)$$

$$p_{j,t,i} \leq Z_{j,t,i} \quad (14)$$

$$p_{j,t,i} \leq Z_{j,t-1,i} \quad (15)$$

Proof. If $\exists i \in G_j$ such that $(Z_{j,t,i} \wedge Z_{j,t-1,i}) = 1$

$$\Rightarrow \sum_{i\in G_j} (Z_{j,t,i} \wedge Z_{j,t-1,i}) \geq 1 \Rightarrow \sum_{i\in G_j} p_{j,t,i} \geq 1$$

where $p_{j,t,i} = Z_{j,t,i} \wedge Z_{j,t-1,i}$ with $p_{j,t,i} \geq 0$. Thus, we can have:

$$p_{j,t,i} = Z_{j,t,i} \wedge Z_{j,t-1,i} \Leftrightarrow$$

$$\left\{\begin{array}{rl} p_{j,t,i} \geq Z_{j,t,i} + Z_{j,t-1,i} - 1 & \forall i \in G_j, \forall j, (t>0) \\ p_{j,t,i} \leq Z_{j,t,i} & \forall i \in G_j, \forall j, (t>0) \\ p_{j,t,i} \leq Z_{j,t-,i} & \forall i \in G_j, \forall j, (t>0) \end{array}\right\}$$

□

Returning to our problematic example, shown in Figure 6(a), the new encoding (12)-(15) will be infeasible as shown by expression (16).

$$\left\{\begin{array}{c} p_{j,t,k} = (Z_{j,t,k} \wedge Z_{j,t-1,k}) = (0 \wedge 1) = 0 \\ \bigvee \\ p_{j,t,m} = (Z_{j,t,m} \wedge Z_{j,t-1,m}) = (1 \wedge 0) = 0 \\ \Rightarrow \\ \sum_{i\in G_j} p_{j,t,i} = p_{j,t,k} + p_{j,t,m} = 0 + 0 = \mathbf{0} \geq \mathbf{1} \\ \textbf{infeasible} \end{array}\right\} \quad (16)$$

The infeasibility can be resolved by activating the constraints violated leading to the situation shown in Figure 6(b). The situation satisfies now the constraints of the new encoding, as shown by expressions (17) and (18) for $t-1$, t and $t+1$.

$$\left\{\begin{array}{c} p_{j,t,k} = (Z_{j,t,k} \wedge Z_{j,t-1,k}) = (1 \wedge 1) = 1 \\ \bigvee \\ p_{j,t,m} = (Z_{j,t,m} \wedge Z_{j,t-1,m}) = (1 \wedge 0) = 0 \\ \Rightarrow \\ \sum_{i\in G_j} p_{j,t,i} = p_{j,t,k} + p_{j,t,m} = \\ 1 + 0 = 1 \geq 1 \\ \textbf{feasible} \end{array}\right\} \quad (17)$$

$$\left\{\begin{array}{c} p_{j,t+1,k} = (Z_{j,t+1,k} \wedge Z_{j,t,k}) = (0 \wedge 1) = 0 \\ \bigvee \\ p_{j,t+1,m} = (Z_{j,t+1,m} \wedge Z_{j,t,m}) = (1 \wedge 1) = 1 \\ \Rightarrow \\ \sum_{i\in G_j} p_{j,t+1,i} = p_{j,t+1,k} + p_{j,t+1,m} = \\ 0 + 1 = 1 \geq 1 \\ \textbf{feasible} \end{array}\right\} \quad (18)$$

C. Non-linear risk allocation

The CNPP encoding described thus far introduces a non-linearity through constraints (5)-(7), which relates risk to the size of the obstacle safety margins. The authors in [18] address this non-linearity by applying the YALMIP branch-and-bound solver to the full disjunctive convex program, and by applying the SNOPT solver to the convex subproblems. To improve efficiency, this paper proposes instead to use a piece-wise linear function to approximate the $erf^{-1}(x)$. We note that the domain and image of $erf^{-1}(x)$ are positive for $\Delta \leq 0.5$ and $0 < \delta_{jt} \leq \Delta$ from equation (6) and constraint (10). Further, in the range $1 - 2\Delta \leq x < 1$, as $x \to 1$, the error function $erf^{-1} \to \infty$. In addition, $\Delta > 0$ for $x = 1 - 2\delta_{jt}$. Furthermore, the function $erf^{-1}(x)$ increases monotonically, hence it can be bounded by a piece-wise linear curve, specified through a linear program formulation. Figure 7 illustrates an approximation of $erf^{-1}(x)$ using $N = 8$ linear segments, for $\Delta = 0.1$.

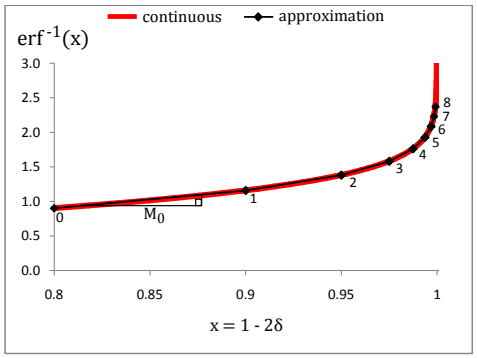


Fig. 7. Piecewise linear approximation for erf^{-1}, where $\delta \leq \Delta = 0.1$ and $N = 8$.

The linear approximation defines values $\bar{y}_n$ in terms of $\bar{x}_n$ through equations (19) and (20), while the slope of each linear segment M_n between points n and $n+1$ is defined by equation (21).

$$\bar{x}_n = 1 - 2\left(\frac{\Delta}{2^n}\right) \quad n = 0, ..., N \quad (19)$$

$$\bar{y}_n = erf^{-1}(\bar{x}_n) \quad n = 0, ..., N \quad (20)$$

$$M_n = \frac{\bar{y}_{n+1} - \bar{y}_n}{\bar{x}_{n+1} - \bar{x}_n} \quad n = 0, ..., N-1 \quad (21)$$

For $erf^{-1}(x)$ with $x = 1-2\delta_{jt}$, the domain $1-2\Delta \leq x < 1$ is divided into N intervals. A variable λ_{njt} is introduced for each interval $n = 1, ..., N$ and $x = 1 - 2\delta_{jt}$ is defined by Equation (22), where $\bar{x}_0$ specifies the minimum x value. The approximated value for $erf^{-1}(x)$ is reached by constraints (23) and the bounds for the λ_{njt} variables are given by constraints (24).

$$1 - 2\delta_{jt} = \bar{x}_0 + \sum_{n=0}^{N-1} \lambda_{njt} \qquad \forall j, t \quad (22)$$

$$erf^{-1}(x) := \bar{y}_0 + \sum_{n=0}^{N-1} M_n \lambda_{njt} \qquad \forall j, t \quad (23)$$

$$0 \leq \lambda_{njt} \leq [\bar{x}_{n+1} - \bar{x}_n] \qquad \forall j, t, n < N \quad (24)$$

$$\lambda_{n+1,jt} > 0 \Rightarrow \lambda_{njt} = [\bar{x}_{n+1} - \bar{x}_n] \qquad \forall j, t, n < N \quad (25)$$

We have $\lambda_{njt} = \bar{x}_{n+1} - \bar{x}_n$ for $n = 0, 1, ..., k-1$; $0 < \lambda_{kjt} < [x_{k+1} - x_k]$ for $n = k$ and $\lambda_{njt} = 0$ for $n = k+1, ..., N-1$. This will always happen once we have a monotonically increasing function inside a minimization problem, so expression (25) holds. Finally, we re-state previous constraints (8) and (10) as follow:

$$c_{i,t}(\delta_{jt}) = erf^{-1}(1 - 2\delta_{jt})\sqrt{2a_i^T \Sigma_t a_i}$$

$$\Rightarrow c_{i,t}(\delta_{jt}) = (\bar{y}_0 + \sum_{n=0}^{N-1} M_n \lambda_{njt})\gamma_{it} \quad (26)$$

with $\gamma_{it} = \sqrt{2a_i^T \Sigma_t a_i}$.

$$c_{i,t}(\delta_{jt}) + b_i - a_i^T \mu_t \leq \bar{M}(1 - Z_{jti})$$

$$\Rightarrow (\bar{y}_0 + \sum_{n=0}^{N-1} M_n \lambda_{njt})\gamma_{it} + b_i - a_i^T \mu_t \leq \bar{M}(1 - Z_{jti}) \quad (27)$$

$$\sum_j \sum_t \delta_{jt} \leq \Delta \Rightarrow \sum_j \sum_t (1 - \bar{x}_0 - \sum_{n=0}^{N-1} \lambda_{njt}) \leq 2 \cdot \Delta \quad (28)$$

D. Objective functions

In this work, the objective function $g(\cdot)$ is a real-valued function of the controls u_t applied throughout the path. Two candidate objective functions are described here; their performance is evaluated in section VI. The first function is based on an approximation to the norm-2 by a 32 sided polygon as proposed in [18]. Specifically, let's assume a two dimension vector $u_t := [u_t^x, u_t^y]^T$ and a continuous variable r_t. The objective function is then approximated by equation (29), where constraints (30) relate r_t to the approximated values.

$$g(\cdot) = \sum_t r_t \quad (29)$$

$$r_t \geq cos\left(\frac{2\pi n}{32}\right) u_t^x + sin\left(\frac{2\pi n}{32}\right) u_t^y \quad \forall (t, n = 0, ..., 31) \quad (30)$$

The second objective function, which is introduced by this paper, proposes a piece-wise linear approximation for the quadratic objective function described in [5]. For $u_t = [u_t^x \quad u_t^y]^T$, we have $< u_t^T \cdot u_t >= (u_t^x)^2 + (u_t^y)^2$; this requires computing two quadratic terms, $(u_t^x)^2$ and $(u_t^y)^2$. The objective function is applied to a positive domain, resulting in a monotonically increasing function. This function is approximated by dividing the domain into $N = 32$ equal sub-intervals ($\frac{u_{max}}{N}$), and by defining variables β_{tn}^d for each one of these sub-interval, according to equations (31) to (33). In addition, we introduce variable $h_t^d \geq 0$ with $h_t^d = \left|u_t^d\right|$ for $d \in \{x, y\}$.

$$h_t^d = \left|u_t^d\right| \quad \Leftrightarrow \quad \begin{matrix} h_t^d \geq +u_t^d \\ h_t^d \geq -u_t^d \end{matrix} \qquad \forall (t, d \in \{x, y\}) \quad (31)$$

$$h_t^d = \sum_n \beta_{tn}^d \qquad \forall (t, d \in \{x, y\}) \quad (32)$$

$$0 \leq \beta_{tn}^d \leq \frac{u_{max}}{N} \qquad \forall (t, n, d \in \{x, y\}) \quad (33)$$

The use of modulus is justified in this case since we have a quadratic function for each variable, where it will be enough to approximate only the positive side of its domain. Our second objective function is summarized by equation (34) with parameter M_{tn}^d being the angular coefficient for the quadratic function, similar to one presented in section IV-C. The constraints (35) have to be satisfied and they always occur once each term in the objective function (35) is monotonically increasing.

$$g(\cdot) = \sum_t \sum_{d \in \{x,y\}} \sum_n M_{tn}^d \beta_{tn}^d \quad (34)$$

$$\beta_{t,n+1}^d > 0 \Rightarrow \beta_{tn}^d = \frac{u_{max}}{N} \quad \forall (t, n < N, d \in \{x, y\}) \quad (35)$$

E. CNPP encoding

We conclude our encoding discussion by summarizing the two MILP models proposed for CNNP. During the remainder of this paper we use $CNPP^{LE}$ to refer to our Linear Encoding from the original formulation, proposed by Blackmore et al [18], and $CNPP^{LEN}$ to refer to this Linear Encoding added by the new constraints introduced in section IV-B.

$CNPP^{LE}$: **Min** $g(\cdot)$ **s.t.**{ **(1),(3),(9),(27),(28)**}

$CNPP^{LEN}$: **Min** $g(\cdot)$ **s.t.**{ **(1),(3),(12)-(15),(27),(28)**}

The objective function is quadratic in the CNPP, but the proposed encodings are still MILP, since two linear approximation are applied as described in section IV-D.

V. Methods

The proposed linear encoding $CNPP^{LE}$ and $CNPP^{LEN}$ allow us to solve instances of the CNPP using exact methods as branch & cut algorithm from CPLEX solver. However, it is indeed possible to apply other relaxation approaches over those encoding. First, this is done over the customized solution approach (CSA), described in [18], which computes upper and lower bound risks from the original non-linear encoding of CNPP. Next, a new method is proposed by the present paper, named hybrid incremental solution approach (HISA), that improves an initial solution returned from Dijkstra algorithm solving MILP sub-models.

A. Customized Solution Approach (CSA)

CSA combines fixed risks with risk allocation approaches as described in [18] and [3]. The fixed risk approaches aim to lower and upper bound the linear encoding. The lower bound variant is named fixed risk relaxation (FRR) and replaces constraints (27) by (36).

$$c_{i,t}(\Delta) + b_{ji} - a_{ji}^T \mu_t \leq \bar{M}(1 - Z_{jti}) \quad \forall (j,t,i \in G_j) \quad (36)$$

This is achieved by assigning to δ_{jt} all of the risk, Δ, specified in the global chance constraint, that is, $\delta_{jt} = \Delta$. Thus the non-linear term $c_{i,t}(\delta_{jt})$ is replaced by constant $c_{i,t}(\Delta)$. Of course, the sum over δ_{jt} violates constraint (10), which is removed in the FRR. While the FRR solution may be infeasible, it lower bounds the cost of the solution and identifies when no solution exists. Therefore, CSA uses FRR as a starting point to a correct and locally optimal method, named the risk allocation approach (RAA). Intuitively, RAA locally adjusts the FRR solution to a solution of the original CNPP. Specifically, the solution to the FRR offers a candidate assignment to the binary variables of the CNPP. This assignment corresponds to deciding which side of each obstacle the vehicle traverses. CSA fixes this binary variable assignment within the original CNPP encoding, while leaving the obstacle risk allocation as decision variables δ_{jt}. RAA then solves the resulting non-linear convex program, to find, if one exists, the optimal risk allocation and trajectory that respects this binary variable assignment.

In this paper, the piece-wise linear approximation introduced allows RAA to solve a linear convex program. To formulate the reduced CNPP, RAA returns constraints (27) only for those obstacles where a risk must be allocated. These obstacles are taken from Z_{ijt} values in the final solution of FRR, as defined by constraints (37).

$$Z_{jti} = 1 \Rightarrow (\bar{y}_0 + \sum_{n=0}^{N-1} M_n \lambda_{njt})\gamma_{jit} + b_{ji} - a_{ji}^T \mu_t \leq 0 \quad (37)$$

The process of computing a lower-bound solution using FRR may not result in a feasible solution. In this event CSA employs a second strategy, in which it computes an upper-bound solution, using a method called fixed risk tightening (FRT). This method upper bounds cost by distributing global risk Δ uniformly across all obstacles. Specifically, FRT replaces constraints (27) by (38):

$$c_{i,t}(\frac{\Delta}{T.J}) + b_{ji} - a_{ji}^T \mu_t \leq \bar{M}(1 - Z_{jti}) \quad \forall (j,t,i \in G_j) \quad (38)$$

FRT preserves feasibility since the risk allocated $\delta_{jt} = \frac{\Delta}{T.J}$ does not violate constraint (10). If FRT returns a feasible solution, CSA will improve it by using RAA. Similar to FRR, this is done by fixing the CNPP binary variables to those from FRT solution, while leaving obstacle risk allocation as decision variables δ_{jt} for RAA. The described processes of finding initial relaxed solutions using FRR or FRT, followed by a local improvement step using RAA, are summarized in Algorithm 1. Also the whole CSA procedure combining FRR, FRT and RAA resolution is illustrated by Figure 8.

Algorithm 1: FRR_FRT_RAA(Model,relaxType)

```
if (relaxType == FRR) then
    Model ← Replace constraints (27) by (36) //Solving FRR;
else
    Model ← Replace constraints (27) by (38) //Solving FRT;
Model ← Remove constraints (28) and variables λ_nijt;
Solve(Model);
if isFeasible(Model.status) then
    //Solving RAA;
    if (relaxType == FRR) then
        Model ← Remove constraints (36);
    else
        Model ← Remove constraints (38);
    Model ← Add constraints (37) from values of Model.Z_ijt;
    Model ← Add constraints (28) and variables λ_nijt;
    Solve(Model);
return Model;
```

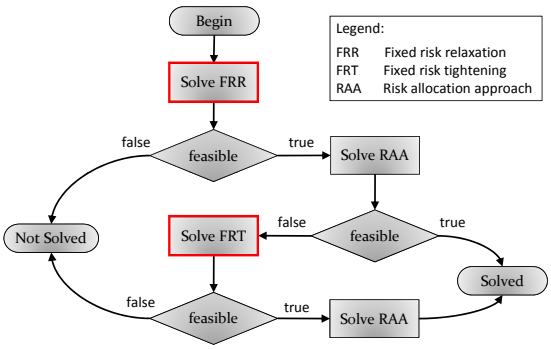


Fig. 8. Overall idea about how CSA combines FRR, FRT and RAA

B. Hybrid incremental solution approach (HISA)

In this section we propose HISA that generates a visibility graph [6], based on relaxed risks, and increases the risk level step by step. The graph is built based on relaxed risk values ($\delta_{jt} = \frac{\Delta}{2^n}$) such that all feasible paths are kept in the visibility graph (G) (see Figure 9), where the dots are nodes ($v \in G$) near vertexes of obstacles.

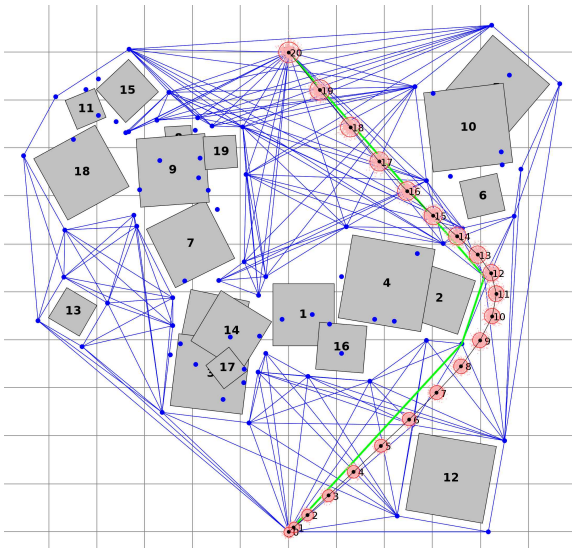


Fig. 9. Graph of the paths and the route of the HISA

Figure 9 also highlights the reference path, achieved by applying Dijkstra algorithm, and the full path solution, after risk allocation.The reference path allows us to define almost all binary variables from the proposed CNPP encoding, becoming easier to solve the risk allocation problem. HISA does not guarantee solution bounds or even optimal solution as CSA, but returns good solutions within a reduced computational time as reported in the next section. Algorithm 2 describes HISA.

Algorithm 2: HISA

```
1  model ← CNPP^LEN;
2  n ← 0;
3  repeat
4      G ← create-graph(Δ/2^n);
5      path ← Dijkstra(G);
6      submodel ← reference-path(model, path, n);
7      Execute Algorithm1(submodel,FRR) with Δ/2^n;
8      n ← n + 1;
9  until isFeasible(submodel.status) or n ≥ maxIter;
10 if isFeasible(submodel.status) then
11     return model;
12 else
13     return HISA fail;
```

A visibility graph G is first built at line 4, where nodes are defined from obstacle vertexes plus the start ($\hat{x}_0$) and goal ($\hat{x}_{goal}$) nodes. The nodes are positioned from obstacle vertexes by considering relaxed risks and they are obtained solving the equation system (39). This system gives the coordinates of nodes v for obstacle j with hyperplanes $i \in G_j$ and $k \in G_j$ at the n^{th} iteration of HISA algorithm.

$$\begin{aligned} a_i^T \cdot v &= b_i + c_{i,0}(\tfrac{\Delta}{2^n}) \\ a_k^T \cdot v &= b_k + c_{k,0}(\tfrac{\Delta}{2^n}) \end{aligned} \tag{39}$$

All nodes v with collision risk greater than $\frac{\Delta}{2^n}$ are removed because they are infeasible (see nodes without connection in Figure 9). Thus edges are built only among nodes v that satisfy the collision risk $\frac{\Delta}{2^n}$ at each iteration. Next, Dijkstra algorithm is applied at line 5 over G to find the shortest path from the start vertex $\hat{x}_0$ to the goal $\hat{x}_{goal}$. A $submodel$ is created in line 6 by fixing the reference $path$ returned by Dijkstra algorithm. Similar to the path returned by FRR, the reference path allows us to prune the search space by fixing several Z_{ijt} values in the $submodel$. However, other Z_{ijt} variables can be optimized when applying Algorithm 1 with $relaxType = FRR$ at line 7. This reduces the number of binary varibles optimized by FRR and becomes possible to reach a feasible full path from FRR using the reference path provided by Dijkstra algorithm. To reach feasibility for CNPP, the allocated risk made by FRR, $\frac{\Delta}{2^n}$, is reduced at each iteration. Of course, if no feasible solution is found after $maxIter$, HISA will fail. This is the same possibility of fail reported for CSA in [18].

Algorithm 3 describes the procedure fix-path. The fix-path algorithm builds a $submodel \subseteq model$ to follow a reference $path$ in G. It begins by calculating $step$ as the average distance traveled between two consecutive time steps t in line 1. In lines 2-12, it is defined the set S_{jt} with all hyperplanes $i \in G_j$ that will be in $submodel$, for each obstacle j and time step t. First, the edges $(v \rightarrow w)$ in the $path$ are traveled considering the lower-bound (lb) and upper-bound (ub) time to go through

Algorithm 3: fix-path($model$, $path$, n)

```
1  step ← cost(path)/T;
2  S_jt ← ∅            ∀(j, t);
3  for (v, w) ∈ edge(path) do
4      lb ← ⌊v.cost/step⌋;
5      ub ← ⌊w.cost/step⌋;
6      for j ∈ obstacles do
7          for i ∈ G_j do
8              for t ∈ [lb, ub] do
9                  λ ← (t − lb)/(ub − lb);
10                 p ← v ∗ (1 − λ) + w ∗ λ;
11                 if (a_ji^T · p ≥ b_ji + c_{i,t}(Δ/2^n)) then
12                     S_jt ← S_jt ∪ {i};
13 submodel ← model
14 for t ∈ [0, T] do
15     for j ∈ obstacles do
16         remove from submodel at period t all i ∈ G_j | i ∉ S_jt;
17 return submodel;
```

these edges (lines 4-5). Next, for each obstacle j, it is added in S_{jt} the hyperplanes $i \in G_j$ enabled at time steps $t \in [lb, ub]$ from v to w. The parameter Δt of CNPP allows us to define the amount of time steps t available in $[lb, ub]$. Finally, the $submodel$ is built by removing all hyperplanes that are disable in S_{jt} (lines 14-16).

VI. Results

The computational tests in this section run on an Intel i7 computer with 3.4 GHz and 32 GB RAM. The $CNPP^{LEN}$ and $CNPP^{LE}$ encoding as well as heuristics CSA and HISA are developed in Java using the mathematical programming software IBM ILOG CPLEX 12.6 Callable Library. All methods execute within 1 hour of time limit, and for the CSA this total time is divide equally between steps FRR/RAA (limited to 1800 sec) and FRT/RAA (limited to 1800 sec).

The experiments are conducted over 2D and 3D maps. The state vector x_t has the positions p and velocities v for each dimension, while the control vector u_t has accelerations defined by expressions (40) and (41) for 2D and 3D maps, respectively. Expressions (42)-(43) present plant control and covariance matrices for 2D map. The 3D control parameters extends the 2D plant control.

$$x_t := [p_x \ p_y \ v_x \ v_y]^T, \quad u_t := [a_x \ a_y]^T \tag{40}$$

$$x_t := [p_x \ p_y \ p_z \ v_x \ v_y \ v_z]^T, \quad u_t := [a_x \ a_y \ a_z]^T \tag{41}$$

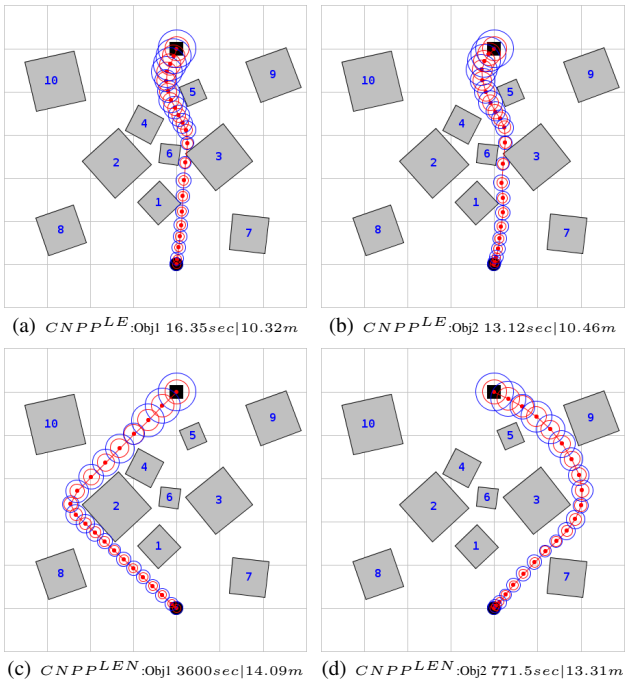
$$A = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} \Delta t^2/2 & 0 \\ 0 & \Delta t^2/2 \\ \Delta t & 0 \\ 0 & \Delta t \end{pmatrix} \tag{42}$$

$$\Sigma_{x_0} = \begin{pmatrix} \sigma_{x_0}^2 & 0 & 0 & 0 \\ 0 & \sigma_{x_0}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \Sigma_{w_t} = \begin{pmatrix} \sigma_{w_t}^2 & 0 & 0 & 0 \\ 0 & \sigma_{w_t}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{43}$$

The majority of tests are set with $\sigma_{x_0} = 0.05$ and $\sigma_{w_t} = 0.05 \cdot \sqrt{3/T}$ where T is the number of time steps for a time horizon of 20 seconds and $\Delta_t = 20/T$. The controls u_t and velocities v_t are given by $|u_t| \leq u_{max} = 1 \ m/s^2$ and $|v_t| \leq v_{max} = 3 \ m/s$. These parameter values are based on those described in [5], [18]; if any value changes for some experiment, the new value is reported.

A. Objective functions and Encoding Issues

The first experiment illustrates the performance of $CNPP^{LE}$ and $CNPP^{LEN}$ when guided by the two objective functions presented on section IV-D. The 32-sided polygon approximation for the norm-2 is called Obj1 and the piece-wise approximation proposed in this paper for the quadratic function is called Obj2. The tests are executed over a hard map similar to one proposed in [18] with maximum risk $\Delta = 0.001$, time horizon $T = 20$ time steps and number of obstacles $J = 10$. The branch & cut algorithm from solver CPLEX is applied to solve the linear encodings $CNPP^{LE}$ and $CNPP^{LEN}$. Figures10(a)-10(d) show the results achieved.



(a) $CNPP^{LE}$:Obj1 $16.35sec|10.32m$ (b) $CNPP^{LE}$:Obj2 $13.12sec|10.46m$

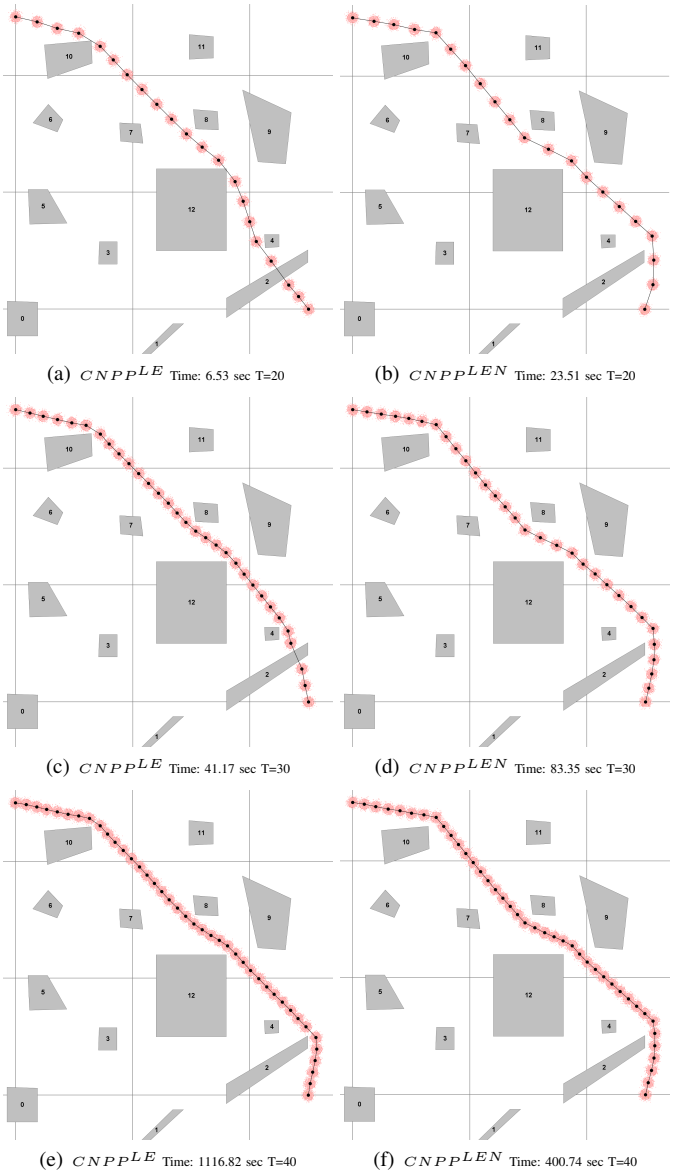
(c) $CNPP^{LEN}$:Obj1 $3600sec|14.09m$ (d) $CNPP^{LEN}$:Obj2 $771.5sec|13.31m$

Fig. 10. $CNPP^{LE}$ and $CNPP^{LEN}$ results guided by Obj1 and Obj2.

$CNPP^{LE}$ is faster and returns shorter paths than $CNPP^{LEN}$, once these paths take a shortage through the obstacles 3 and 6 (Figures 10(a) and 10(b)) with risks incurred but not computed. The new constraints in $CNPP^{LEN}$ become its encoding more time consuming with longer paths returned, but no-computed risks are now considered. In this case, paths between obstacles 3 and 6 are avoided, so the risk bound is properly satisfied (Figures 10(c) and 10(d)).

Both models show a better computational time when guided by Obj2. The performance is even better for $CNPP^{LEN}$, where Obj2 reduces considerably the computational time against Obj1. We do not claim that one objective function is always better than other. It is only possible to say that the approximation applied over Obj2, as coded in this papers and evaluated over the hard map illustrated, leads $CNPP^{LEN}$ to be faster than the approximation coded for Obj1. Since the other maps that will be evaluated are generated based on this hard map, the next experiments will be conducted using only Obj2. Also the piece-wise approximation employed in Obj2 can be extended easier to 3D maps than the 32-side polygon for Obj1.

The next experiment emphasizes how the new constraints in $CNPP^{LEN}$ can avoid to overcome obstacles or incur in no-computed risks. We consider a mobile robot that has to avoid obstacles when going from one origin to a destination point. The parameter values and the 2D environment are taken from Rover robot simulations, where the plant model consider $x_t := [p_x \ p_y]^T$ and $u_t := [v_x \ v_y]^T$, it is set $\Delta = 0.01$, $T = 20, 30, 40$, $\Delta t = 20$, $\sigma_{x_0} = 0.1$, $\sigma^x_{w_t} = 0.3$, $\sigma^y_{w_t} = 0.03$ and $v_{max} = 0.5$ m/s. Figures 11(a)-11(f) show the optimal paths returned by $CNPP^{LE}$ and $CNPP^{LEN}$ along with the execution time, when the number of time steps increases.



(a) $CNPP^{LE}$ Time: 6.53 sec T=20 (b) $CNPP^{LEN}$ Time: 23.51 sec T=20

(c) $CNPP^{LE}$ Time: 41.17 sec T=30 (d) $CNPP^{LEN}$ Time: 83.35 sec T=30

(e) $CNPP^{LE}$ Time: 1116.82 sec T=40 (f) $CNPP^{LEN}$ Time: 400.74 sec T=40

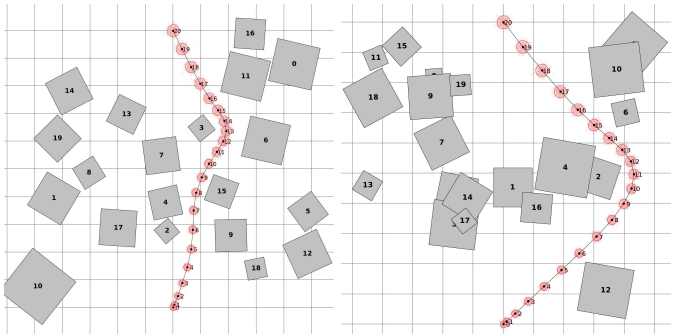
Fig. 11. Solutions and execution time for Rover tests T=20,30,40

$CNPP^{LE}$ overcomes obstacles with $T = 20$ and 30 time steps, so the paths are infeasible even though $CNPP^{LE}$ returns them as optimal solutions. The only way to make this encoding to return a path planning, without overcome obstacles in this scenario, is setting $T = 40$. On the other hand,

$CNPP^{LEN}$ finds optimal solutions that do not overcome obstacles for all time steps. $CNPP^{LEN}$ spends more computational time, but improves risk bounds through the paths when the number of time steps increases. Since paths have to be always returned without overcome obstacles and within the maximum level of risk Δ, $CNPP^{LEN}$ is the only encoding able to do this for different times steps in this example. Also, for the only time horizon in which $CNPP^{LE}$ returns a valid solution, $CNPP^{LEN}$ is faster than it to find the same optimal solution (Figures 11(e) and 11(f)).

B. Scaling number of obstacles and time steps

The next computational test evaluates $CNPP^{LEN}$, CSA and HISA for path planning when the number of obstacles and time steps increase. In this experiment, we are comparing an exact solution, solving the full linear encoding $CNPP^{LEN}$, against heuristic solutions returned by CSA and HISA. Two groups of maps are generated: 2D and 3D maps. For each of these groups, two another groups are defined: regular and non-regular maps. The 2D maps have no heights related with the obstacles, so they are always non-fly zones (NFZ). The 3D maps have NFZ as well as obstacles with heights, so the UAV flies over these obstacles. The so-called non-regular maps have obstacles that can be stacked one over another generating irregular shapes regions. These shapes may represent scenarios where we have, e.g, irregular no-fly zones for 2D maps or mountains with different heights at a same area for 3D maps. The regular maps will spread obstacles through the available area, so it can be harder to define a path with several non-fly zones on the map. Figure 12 shows maps for each group.



(a) 2D regular map (b) 2D non-regular map

(c) 3D regular map (d) 3D non-regular map

Fig. 12. Examples for 2D and 3D regular and non-regular maps

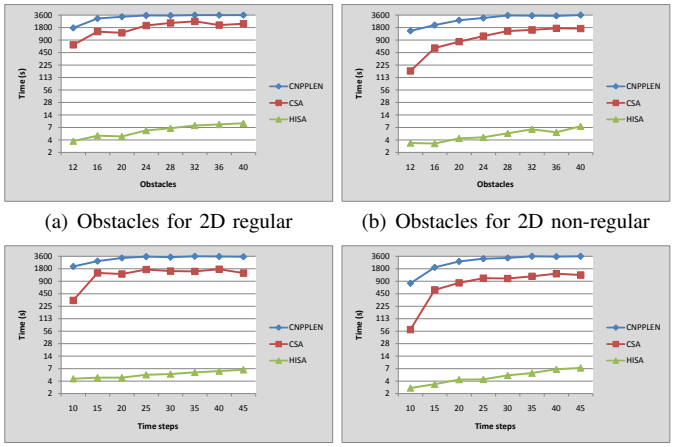
A map generator was developed and two large set were randomly created. In the first set of maps, the map generator increased the number of obstacles by $J = 12, 16, 20, 24, 28, 32, 36, 40$ with $T = 20$ and $\Delta = 0.001$, and 50 maps were created for each J value. Thus, a total of 1,600 maps will be evaluated for 2D and 3D regular and non-regular maps. In the second set of maps, it was generated 50 maps increasing the time horizon (number of time steps) by $T = 10, 15, 20, 25, 30, 35, 40, 45$ with $J = 20$ and $\Delta = 0.001$. A total of 1,600 maps is also defined for 2D and 3D regular and non-regular maps. Thus, there is a total of 3,000 different maps because we have an overlap for $J = 20$ and $T = 20$.

1) Results for 2D maps: Table I has the number of feasible solutions returned by $CNPP^{LEN}$, CSA and HISA for 2D regular and non-regular maps when increasing J and T. The *same maps* row has the amount of maps where all methods find solutions. HISA returns solutions for maps in all groups, except by one regular and another non-regular map with $T = 10$. CSA shows competitive results, while $CNPP^{LEN}$ has problems to return solutions for J and T increasing. The number of obstacles and time steps have a reasonable impact over the amount of decision variables in $CNPP^{LEN}$ encoding, which becomes harder to apply an enumeration algorithm over it.

TABLE I
NUMBER OF FEASIBLE SOLUTIONS FOR 2D MAPS INCREASING J AND T

Obstacles (J)	12	16	20	24	28	32	36	40
Regular								
$CNPP^{LEN}$	48	50	48	45	47	34	29	18
CSA	50	50	50	50	50	48	48	47
$HISA$	50	50	50	50	50	50	50	50
same maps	48	50	48	45	47	33	27	17
Non-regular								
$CNPP^{LEN}$	49	50	50	47	47	44	44	36
CSA	50	50	50	50	50	50	50	50
$HISA$	50	50	50	50	50	50	50	50
same maps	49	50	50	47	47	44	44	36
Time Steps (T)	10	15	20	25	30	35	40	45
Regular								
$CNPP^{LEN}$	49	48	48	47	45	37	33	29
CSA	49	49	50	50	50	49	50	47
$HISA$	49	50	50	50	50	50	50	50
same maps	47	47	48	47	45	37	33	28
Non-regular								
$CNPP^{LEN}$	49	50	50	47	45	45	44	44
CSA	50	50	50	50	50	50	50	50
$HISA$	49	50	50	50	50	50	50	50
same maps	48	50	50	47	45	45	44	44

Figures 13(a)-13(d) present the average computational time to return solutions, but taking into account only *same maps* to allow a fair comparison. HISA is the fastest approach spending on average less than 9 sec to define a path for regular and 7.5 sec for non-regular maps (Figures 13(a) and 13(b)). On the other hand, $CNPP^{LEN}$ takes more than 1,700 sec for regular and 1,500 sec for non-regular maps. CSA has a performance better than $CNPP^{LEN}$, but worse than HISA. In Figures 13(c) and 13(d), $CNPP^{LEN}$ also takes longer to return a solution when the number of time steps increases. HISA is again the fastest method, spending less than 6.6 sec on average for all regular maps and less than 7.3 sec for all non-regular maps.

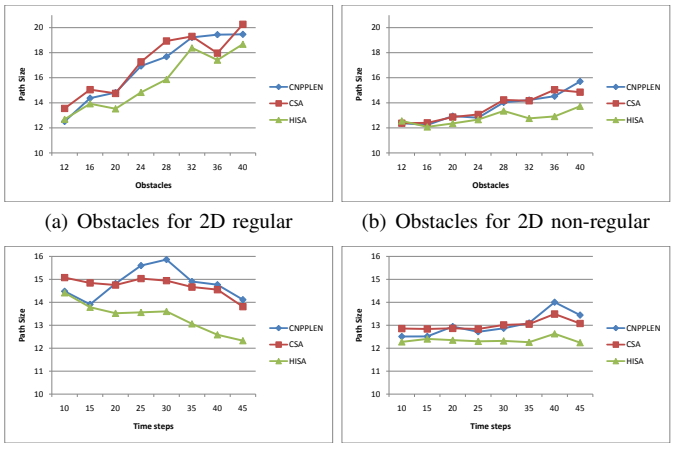


(a) Obstacles for 2D regular (b) Obstacles for 2D non-regular

(c) Time steps for 2D regular (d) Time steps for 2D non-regular

Fig. 13. Average computational time and path size when increasing J and T for 2D maps

Figures 14(a)-14(d) show average path size values, based on the final trajectories returned for *same maps*. The UAV always starts from (0,0) and goes to (0,-10), thus, the path sizes have values between 10 and 20 units. The path sizes are larger for regular than non-regular maps when increasing J, since their obstacles are more spread through the map. If the number of time steps increases, it is possible to find better path sizes for regular maps. This makes sense since more time steps can lead to a better path planning. HISA returned the shortest paths on average for most of the results, but all approaches reach similar path sizes.



(a) Obstacles for 2D regular (b) Obstacles for 2D non-regular

(c) Time steps for 2D regular (d) Time steps for 2D non-regular

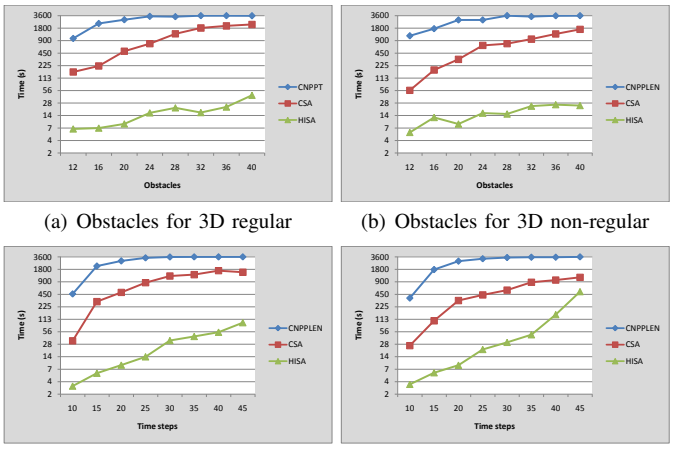
Fig. 14. Average path size when increasing J and T for 2D maps

To summarize, CSA is able to return feasible solutions when the number of obstacles and time steps increase, but its drawback is the computational time. $CNPP^{LEN}$ has problems to find feasible solutions when J or T increase within 1h of time limit. HISA seems to be the best approach since it solves the majority of maps, it is computationally fast and reaches competitive path size values.

2) Results for 3D maps: Table II shows the number of feasible solutions returned for regular and non-regular 3D maps increasing J and T. The number of *same maps* is also presented. HISA and CSA return feasible solutions for all groups, except CSA for one non-regular map with $J = 36$. $CNPP^{LEN}$ encoding does not find 19 solutions for both regular and non-regular maps increasing J, and it fails for 11 regular and 12 non-regular maps increasing T. Figures 15(a)-15(d) show the average computational time for *same maps*. HISA is again the fastest method for regular and non-regular maps, while $CNPP^{LEN}$ takes more than 1,800 sec for the majority of regular and non-regular maps. CSA perfoms better than $CNPP^{LEN}$, but significantly worse than HISA.

TABLE II
NUMBER OF FEASIBLE SOLUTIONS FOR 3D MAPS INCREASING J AND T

Obstacles (J)	12	16	20	24	28	32	36	40
Regular								
$CNPP^{LEN}$	49	49	48	50	45	45	50	45
CSA	50	50	50	50	50	50	50	50
$HISA$	50	50	50	50	50	50	50	50
same maps	49	49	48	50	45	45	50	45
Non-regular								
$CNPP^{LEN}$	50	50	48	47	46	46	50	44
CSA	50	50	50	50	50	50	49	50
$HISA$	50	50	50	50	50	50	50	50
same maps	50	50	48	47	46	46	49	44
Time Steps (T)	10	15	20	25	30	35	40	45
Regular								
$CNPP^{LEN}$	49	47	48	48	49	49	49	50
CSA	50	50	50	50	50	50	50	50
$HISA$	50	50	50	50	50	50	50	50
same maps	49	47	48	48	49	49	49	50
Non-regular								
$CNPP^{LEN}$	50	48	48	50	50	50	46	46
CSA	50	50	50	50	50	50	50	50
$HISA$	50	50	50	50	50	50	50	50
same maps	50	48	48	50	50	50	46	46



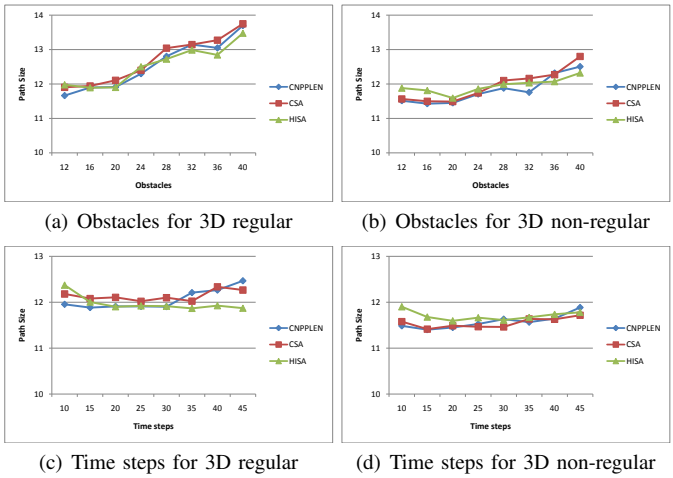
(a) Obstacles for 3D regular (b) Obstacles for 3D non-regular

(c) Time steps for 3D regular (d) Time steps for 3D non-regular

Fig. 15. Average computational time and path size increasing J in 3D maps

HISA seems to have more problems solving 3D maps than 2D maps, where its average execution time is less than 8 sec. This behavior is expected once HISA emphasizes optimization over NFZ, while obstacles with heights are dealt with by solving the related MILP submodel. CSA spends on average half of the time spent by $CNPP^{LEN}$, which is an expected behaviour from a heuristic based on relaxations from $CNPP^{LEN}$ encoding. Figure 16(a) and 16(d) show the path size values, where there is no great difference among these

methods. In this case, HISA returns on average path sizes closer to CSA and $CNPP^{LEN}$ when increasing J and T.



(a) Obstacles for 3D regular (b) Obstacles for 3D non-regular

(c) Time steps for 3D regular (d) Time steps for 3D non-regular

Fig. 16. Average computational path size increasing J and T for 3D maps

The results for 3D maps show that all methods are able to find more feasible solutions, which can be explained by the possibility to fly over obstacles. HISA is again the best approach, returning more solutions within small computational time and with similar path size compared to the others. CSA is competitive against HISA in terms of feasible solutions, but it is also time-consuming solving 3D maps.

3) FlightGear simulation: Simulations are executed to illustrate the behaviour of the aircraft following a path returned by HISA. We choose FlightGear (FG) simulator, since it is an open flight simulator with many resources available. The UAV is simulated using the flight dynamic of Cessna 172, but the simulation is not conducted by a human pilot at any moment. An autopilot is coded to control automatically the aircraft flight. Figure 17 shows the main screen of the framework developed to integrates FG, autopilot and execution of HISA. HISA sends a route to autopilot that executes the path from one region to another.

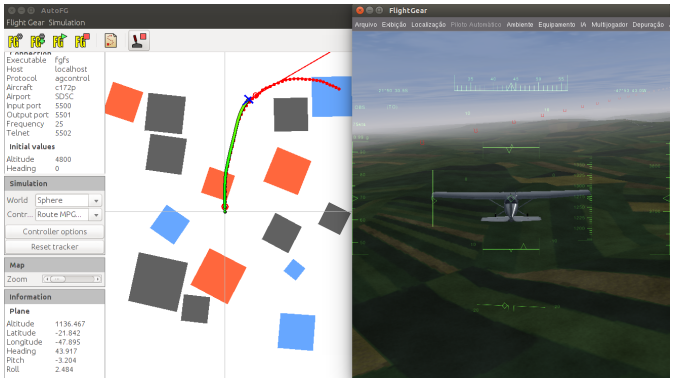


Fig. 17. Integration Flightgear, HISA and autopilot.

We previously define a map that includes an airport location from which the aircraft should take off and land. There are four regions (A, B, C and D), where the aircraft must fly through and six NFZs (gray polygons) as shown by Figure 18. The aircraft must go from the airport to region A and fly next from $A \rightarrow B \rightarrow C \rightarrow D$ when, finally, it goes back to the airport.

It is performed a total of 100 simulations taking into account default wind settings of FG (random wind velocities). The uncertainties are set as $\sigma_{x_0} = 30m$ and $\sigma_{w_t} = 10m$, based on previous test with Cessna 172, and total risk $\Delta = 0.001$. All 100 paths followed by the aircraft are shown in Figure 18, where it is possible to see that obstacles avoidance is satisfied within the allowed risk. The safety margin from obstacles can be better evaluated from Figure 19. HISA defines a path with minimum expected distance from obstacle of 250.1 meters (red straight line). Figure 19 shows the minimum distance achieved by the aircraft from an obstacle during each simulation. In the worst case, the aircraft becomes 188.9 meters close to the obstacle during simulation 91^{st}. These results indicate that HISA's paths can be conservative enough to avoid collision, keeping reasonable safety margins even under external or internal disturbances. There is a video available at web[1] with one of the simulations executed.

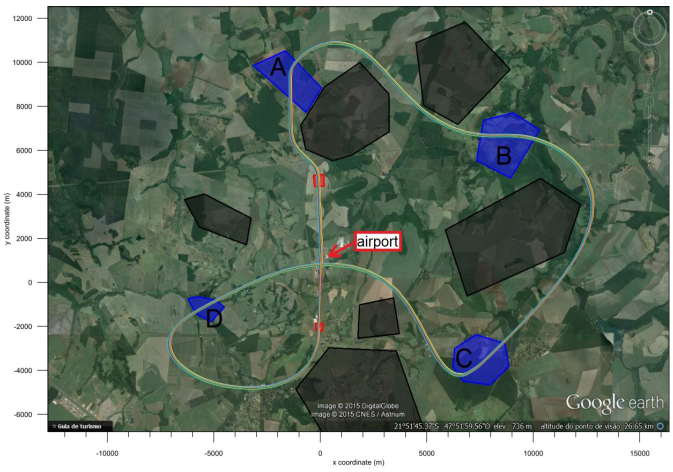


Fig. 18. FlightGear simulator with 100 routes from HISA.

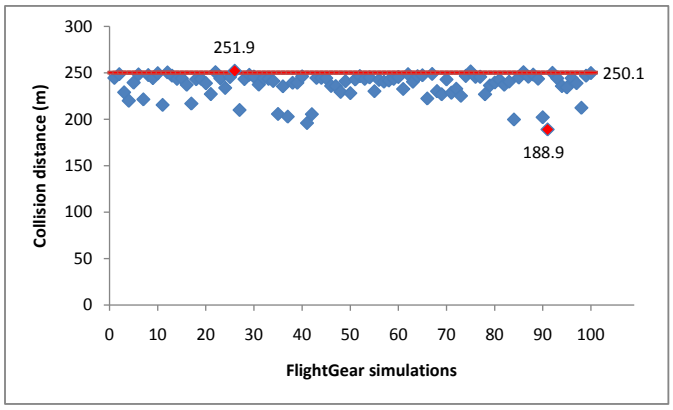


Fig. 19. Minimum collision distance considering all obstacles.

[1]https://youtu.be/i8flhA4yf90

VII. Conclusion

This paper proposes a mixed integer linear programing (MILP) encoding, named as $CNPP^{LEN}$, for the Chance-constrained Non convex Path-planning Problem (CNPP). In this problem, the unmanned aerial vehicle (UAV) path planning problem is addressed for non-convex regions with bounded risk. The proposed encoding deals with the problem of excessive risk during the path planning, which can lead to paths where an obstacle is overcome or many risks incurred are not computed. $CNPP^{LEN}$ addresses these issues without the need of adjust parameters, for example, increasing conveniently the number of time steps within the time horizon as it was done for $CNPP^{LE}$. The new constraints introduced in the $CNPP^{LE}$ encoding solve this problem, once became infeasible to set successive time steps, whose line between them intersect an obstacle.

The new encoding introduced on this paper includes constraints to handle obstacle avoidance and risk bounds, along with piece-wise linear approximations for a quadratic objective function and for the inverse of error function. Moreover, a relevant contribution achieved is to reduce the CNPP from a mixed integer programming (MIP) problem to a mixed integer linear programming problem (MILP).

The MILP encoding developed allows us to advance in terms of how to deal with bounded risks as well as parameter scalability. In terms of risks, the paths can not go through obstacles and a better approximation is reached for the total risk incurred following the paths. In terms of scalability, a new benchmark for number of obstacles and time steps is established from a set of 3,000 maps generated for 2D and 3D scenarios, with solutions reported for scenarios including 12 up to 40 obstacles, and 10 up to 45 time steps. These maps can be used by other researchers to validate their methods in problems related to CNPP.

The exact solution of the $CNPP^{LEN}$ encoding and the heuristic CSA present different computational time performance. The exact solution of the full encoding is time-consuming, taking usually more than 1800 sec and it does not return solutions for several maps within the time limit. CSA is able to find solutions for the majority of maps under the new encoding, but it is also time-consuming. The proposed HISA heuristic finds solution within a reasonable computational time. Thus, the computational results for several maps as well as the simulation example provided indicate that HISA can work properly when a path planning is needed in a short time.

Improvement over the linear encoding and the HISA approach are being developed as a future work, aiming to reach a method faster enough to be embedded in a UAV system for real-time path planning subject to the constraints defined by CNPP.

Acknowledgments

This research was supported by Fundacao de Amparo a Pesquisa do Estado de Sao Paulo (FAPESP), projects 2014/12297-0, 2014/11331-0 and 2013/07375-0, MISTI-Brazil Seed Funds with support from the Itau Fund for research on sustainability in Latin America, and by the Office of Naval Research Grant N00014-15-IP-00052. The research described in this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," *Proceedings of American Control Conference*, 2002.

[2] E. Fernández-González, E. Karpas, and B. C. Williams, "Mixed discrete-continuous heuristic generative planning based on flow tubes," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, pp. 1565–1572.

[3] M. Ono, "Robust, goal-directed plan execution with bounded risk," Ph.D. dissertation, Massachusetts Institute of Technology, 2012.

[4] A. Richards and J. P. How, "Implementation of robust decentralized model predictive control," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.

[5] M. Ono, B. C. Williams, and L. Blackmore, "Probabilistic planning for continuous dynamic systems under bounded risk," *J. Artif. Int. Res.*, vol. 46, no. 1, pp. 511–577, Jan. 2013.

[6] Y. Kuwata, "Real-time trajectory design for unmanned aerial vehicles using receding horizon control," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[7] M. Ono and B. C. Williams, "An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure," in *In Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI*, 2008.

[8] Y. Saleem, M. H. Rehmani, and S. Zeadally, "Integration of cognitive radio technology with unmanned aerial vehicles: Issues, opportunities, and future research challenges," *Journal of Network and Computer Applications*, vol. 50, pp. 15 – 31, 2015.

[9] X. Yang, T. Wang, J. Liang, G. Yao, and M. Liu, "Survey on the novel hybrid aquaticaerial amphibious aircraft: Aquatic unmanned aerial vehicle (aquauav)," *Progress in Aerospace Sciences*, vol. 74, pp. 131 – 151, 2015.

[10] L. Zhu, X. Cheng, and F.-G. Yuan, "A 3d collision avoidance strategy for uav with physical constraints," *Measurement*, vol. 77, pp. 40 – 49, 2016.

[11] H. X. Li, "Kongming: A generative planner for hybrid systems with temporally extended goals," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.

[12] A. Blum and M. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, 1997.

[13] A. Hofmann, "Robust execution of bipedal walking tasks from biomechanical principles," Ph.D. dissertation, Massachusetts Institute of Technology, 2006.

[14] R. Mattmuller and J. Rintanen, "Planning for temporally extended goals as propositional satisfiability ability," *International Joint Conference on Artificial Intelligence*, 2007.

[15] L. A. M. Bush, "Decision uncertainty minimization and autonomous information gathering," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[16] D. C.-P. Wang, "A factored planner for the temporal coordination of autonomous systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[17] L. Blackmore and M. Ono, "Convex chance constrained predictive control without sampling," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2009.

[18] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[19] A. J. Wang, "Risk allocation for temporal risk assessment," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[20] C. Fang, "Mission-level planning with constraints on risk," Ph.D. dissertation, Massachusetts Institute of Technology, 2014.

[21] E. F. T. Schouwenaars, B. D. Moor and H. J., "Mixed integer programming for multi-vehicle path planning," *Proceedings of the European Control Conference*, 2001.