Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2021-IC-005

2021-IC-005

Approximate optimal hybrid control synthesis by classification-based derivative-free optimization

Xing, Shaopeng and Wang, Jiawan and Bu, Lei and Chen, Xin and Li, Xuandong

Technical Report 2021

Approximate Optimal Hybrid Control Synthesis By Classification-based Derivative-free Optimization

Shaopeng Xing, Jiawan Wang, Lei Bu, Xin Chen, Xuandong Li
State Key Laboratory For Novel Software Technology,
Department of Computer Science and Technology,
Nanjing University
Nanjing, Jiangsu, China

ABSTRACT

Hybrid systems are widely used in safety-critical areas. Hybrid optimal control synthesis, which aims to generate an optimal sequence of control inputs for a given task, is one of the most important problems in the field. The classical Gradient-based methods are efficient but they require the system under control should be differentiable. Sampling-based methods have no such limitations, but the ability of existing ones to solve complex control missions is restricted.

In this paper, we propose a practical and efficient method to solve a general class of hybrid optimal control problems. Basically, we transform the control synthesis problem into a derivative-free optimization (DFO) problem. Then, we adapt a start-of-art classification-based DFO method to solve the optimization problems based on sampled variables efficiently. Furthermore, for complex state space, which is difficult to solve, we present a piecewise control synthesis method to make a tradeoff between optimality and efficiency by generating feasible and piecewise optimal control inputs instead. The empirical results on two complex real-world hybrid systems: a vehicle and a quadcopter drone system, demonstrate that our method outperforms existing methods significantly.

CCS CONCEPTS

• **Computer systems organization** → *Embedded software.*

KEYWORDS

hybrid systems, optimal control, derivative-free optimization

ACM Reference Format:
Shaopeng Xing, Jiawan Wang, Lei Bu, Xin Chen, Xuandong Li. 2021. Approximate Optimal Hybrid Control Synthesis By Classification-based Derivative-free Optimization. In *24th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '21), May 19–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447928.3456658

Corresponding author: Lei Bu, bulei@nju.edu.cn.

HSCC '21, May 19–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8339-4/21/05…$15.00
https://doi.org/10.1145/3447928.3456658

1 INTRODUCTION

Hybrid systems [5, 20, 25] are dynamical systems that combine continuous and discrete behavior. Hybrid automata (HA) [2, 25] is a natural modeling language for hybrid systems. Hybrid systems have been widely used in various areas such as industry, transportation, biological systems and so on. Thus, how to design a safe, efficient hybrid system [12, 19, 26] has attracted lots of attention.

Hybrid optimal control [6] is one of the most critical problems in hybrid systems. It attempts to generate a sequence of control modes of the HA model and also generate the corresponding dwell time and control inputs in each control mode to meet an optimality target. However, due to the complex behavior of hybrid systems, including continuous variables changes and discrete control modes transitions, the optimal control problem of such systems is difficult.

During the past two decades, various algorithms have been proposed to solve hybrid optimal control problems. The gradient-based optimization algorithm, firstly proposed by Xu and Antsaklis [37, 38], is the most typical method. Following this work, researchers have devoted a lot of effort to the optimization of these works to achieve better efficiency and usability [1, 3, 11, 16, 17, 22, 23, 27] .

Nevertheless, the applicability of gradient-based methods is restricted. To use such methods, both the behavior of the HA model and objective functions have to be differentiable or continuous, which is difficult to meet by general nonlinear systems. Actually, non-differentiable behavior such as discontinuous objective functions or constraints widely exists in hybrid systems. Obviously, existing gradient-based methods cannot handle such problems.

For non-differentiable problems, sampling-based methods were proven very successful in practice. The most influential sample-based algorithms include Rapidly-exploring Random Trees (RRTs) [8, 30] and Probabilistic RoadMaps (PRMs) [28, 29]. They share the idea of incrementally extending the search graph by randomly sampling points towards the unexplored state space, but differ in how they construct a graph connecting these points. However, neither RRTs nor PRMs can guarantee the optimality of the synthesized solution. Vasile [35] presented how to generate maximally-satisfying controllers for temporal logic specifications. This technique was designed only for continuous control and cannot be applied directly into hybrid optimal control area. Farahani [18] proposed a robust model predictive control approach based on Monte Carlo simulation and rejection-sampling to solve hybrid optimal control problems. Nevertheless, its ability to generate feasible trajectories for complex control missions is restricted.

In this paper, we propose a new practical sampling-based algorithm to handle complex hybrid optimal control problems involving non-differentiable behavior. Firstly, we traverse the HA model of

the system to enumerate potential control mode sequences in a given depth bound. Then, for a fixed mode sequence, we transform the non-differentiable optimal control synthesis problem of dwell times and control inputs into a derivative-free optimization (DFO) problem. Then, the new optimization problem can be solved by adapting sampling-based derivative-free algorithms, e.g., genetic algorithms [21], randomized local search [32], Bayesian optimization methods [10], cross-entropy methods [13] and so on.

Specifically, in this paper, we adapt a classification-based derivative-free optimization framework [40] to solve this DFO problem, instead of using state-of-the-art rejection sampling methods directly. It was discussed theoretically in [39] that such framework takes polynomial time to approximate optimal solutions in solving local-Lipschitz functions, while state-of-the-art rejection sampling methods need exponential time. Based on this theoretically-grounded framework, we design a new **C**lassification-based **D**erivative-free optimization method for **H**ybrid optimal control (**CDH**) specifically, to solve such non-differentiable control problems efficiently.

Furthermore, for complex problems with large bound, the number of potential control sequences may increase exponentially. We make a tradeoff between performance and optimality by splitting a whole trajectory into multiple subsequences and generating optimal control inputs for each subsequence, respectively. Then, we concatenate these sub-control inputs to generate a feasible and piecewise optimal control input set instead.

We implement the above algorithm and apply it to handle hybrid optimal control of two complex real-world hybrid systems: a vehicle and a drone system. Our method can generate optimal feasible solutions for complex control tasks within only one minute. In summary, this paper makes the following contributions:

- We propose a new control synthesis framework that can transform hybrid optimal control problems to DFO problems and give the proof of the equivalence of the transformation.
- We adapt a **C**lassification-based **D**erivative-free optimization method for **H**ybrid optimal control (**CDH**), to solve the above optimization problems on sampled variables. In our experiments, **CDH** outperforms other DFO methods and existing sampling-based control algorithms substantially.
- We present a new method to generate piecewise optimal control for complex tasks by splitting the original task into multiple sub-tasks. The experiments show this piecewise control synthesis method can save more than 88% percent of the time usage and achieve similar control performance.

The rest of this paper is organized as five sections introducing: the definition of our hybrid optimal control problem, our detailed optimal control algorithm, the experiments, related works and the conclusion of the paper, respectively.

2 PROBLEM FORMULATION

Due to the complex behavior combining continuous changes and discrete transitions of nonlinear hybrid systems, we model such systems by *Hybrid Automata*. The following definition of hybrid automata is extended from [25].

Definition 1. *A **Hybrid Automaton** H is consisted of:*

- ***Variables.*** *A finite set $X = \{x_1, ..., x_n\}$ of real-numbered variables. $\dot{X}$=$\{\dot{x_1}, ..., \dot{x_n}\}$ are first derivatives during continuous changes and X'=$\{x'_1, ..., x'_n\}$ represents values at the conclusion of changes. A finite set $U = \{u_1, ..., u_{n'}\}$ of external control inputs bounded to the input space $\mathbb{U}$.*
- ***Control graph.*** *A finite directed multigraph (V, E). The vertices in V are called control modes. The edges in E are called control switches.*
- ***Initial, invariant, flow conditions*** *Three vertex labeling functions $init$, inv, and $flow$ that assign to each control mode $v \in V$ three predicates. Each initial condition $init(v)$ is a predicate whose free variables are from X. Each invariant condition $inv(v)$ is a predicate whose free variables are from X. Each flow condition $flow(v)$ is a predicate whose free variables are from $X \cup \dot{X} \cup U$.*
- ***Jump conditions.*** *An edge labeling function $jump$ that assigns to each control switch $e \in E$ a predicate. Each jump condition $jump(e)$ is a predicate whose free variables are from $X \cup \dot{X}$.*
- ***Events.*** *A finite set $\sum$ of events, and an edge labeling function $event: E \to \sum$ that assigns to each control switch an event.*

Definition 2. ***Semantic.*** *Given a hybrid automaton H, the state of H is a triple $(v, \mathbf{x}, \mathbf{u}) \in (V \times \mathbb{R}^n \times \mathbb{U}^{n'})$, which means H is in the control mode v and the values of X and U are $\mathbf{x}$ and $\mathbf{u}$ respectively. H can perform two kinds of transitions between two states.*

- ***Continuous transition****: If H enters mode v, and stays in v by time τ with control input $\mathbf{u}$, we have $(v, \mathbf{x}, \mathbf{u}) \xrightarrow{\tau} (v, \mathbf{x'}, \mathbf{u})$, and there exists a differentiable function f that satisfies*

$$C(v, \mathbf{x}, \tau, \mathbf{u}) = \begin{cases} f(0) = \mathbf{x},\ f(\tau) = \mathbf{x}' \\ flow(v)[X, \dot{X}, U := f(t), \dot{f}(t), \mathbf{u}] \\ \forall t \in [0, \tau], inv(v)[X = f(t)] \end{cases}$$

- ***Discrete transition****: H can also perform discrete jumps between control modes. For each event $\sigma \in \sum$, we have $(v, \mathbf{x}, \mathbf{u}) \xrightarrow{\sigma} (v', \mathbf{x}', \mathbf{u}')$ iff there is a control switch $e \in E$ that satisfies*

$$D(v, v', \mathbf{x}, \mathbf{x}') = \begin{cases} Source(e) = v,\ Target(e) = v' \\ jump(e)[X, X' := \mathbf{x}, \mathbf{x}'] \\ event(e) = \sigma \end{cases}$$

We now define the control trajectory for a hybrid automaton H. Firstly, we define a **control tuple** $(v, \tau, \mathbf{u}) \in (V \times R^+ \times \mathbb{U}^{n'})$, which means H stays in the control mode v by time τ with the control input $\mathbf{u}$. Specifically, $c = (\tau, \mathbf{u})$ is called the **control configuration** of the control mode v that defines the dwell time and control input.

Definition 3. *The **control trajectory** is defined as a sequence of the tuples above*

$$\xi = (v_0, c_0), \ldots, (v_{N-1}, c_{N-1})$$

where $N = |\xi|$ is the length of the control trajectory.

For the convenience of introducing our algorithm in the next section, we use $\xi = (\mathbf{v}, \mathbf{c})$ to denote ξ where $\mathbf{v} = v_0, v_1, ..., v_{N-1}$ is the **control mode sequence** of ξ and $\mathbf{c} = c_0, c_1, ..., c_{N-1}$ is the corresponding **control configuration**.

Definition 4. *For a control trajectory ξ of length N, if*

$$\Psi(\xi) = \left(\wedge_{i=0}^{N-1} C(v_i, \mathbf{x}_i, \tau_i, \mathbf{u})\right) \wedge \left(\wedge_{i=1}^{N-1} D(v_{i-1}, v_i, \mathbf{x}_{i-1}, \mathbf{x}_i)\right)$$

is true, we say ξ is a **feasible control trajectory**.

The main objective of hybrid optimal problems is to find an optimal feasible trajectory to fulfill specific control targets. Beside

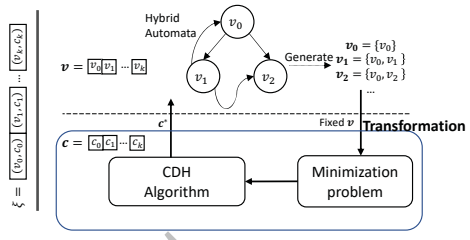


Figure 1: Our Hybrid Optimal Control Synthesis Framework.

this, it is very often to see that the cost function is taken into consideration in the control synthesis. So, we define our **objective function** as

$$J(\xi) = \phi(\mathbf{x}^{(\xi)}) + \sum_{i=0}^{|\xi|-1} L_i(v_i, \mathbf{x}, \mathbf{u})$$

where $\mathbf{x}^{(\xi)}$ is the value of X after H finishes the control trajectory ξ, ϕ is the control target function, while L_i is an arbitrary function related to v_i that can be a cost function or any other function for some specific control goal. Finally, we define our **Hybrid Optimal Control Problem** based on hybrid automata.

Definition 5. *A hybrid optimal control problem $\mathcal{P}$ of a HA model is to calculate an optimal feasible trajectory of the HA model that minimize $J(\xi)$ $\quad s.t.\ \Psi(\xi)$ is true*

It's worth noting that the problem defined in Def.5 is a more general case compared to the conventional one given in [22]. It makes **no restriction** on the objective function, constraints or control input spaces. Next, we will introduce our newly proposed derivative-free algorithm to solve the above problem.

3 HYBRID OPTIMAL CONTROL SYNTHESIS FRAMEWORK

3.1 Overall Description

To solve the problem in Def.5, we propose a new non-differentiable hybrid optimal control synthesis framework in Fig.1. Firstly, we traverse the hybrid automata model of the system as a finite directed graph and generate many potential control mode sequences in a given depth bound using the depth-first search algorithm. Then, for a fixed control mode sequence $\mathbf{v}$, we synthesize the optimal dwell time and control inputs for each mode by transforming the optimal control synthesis problem into a minimization problem that can be solved by our proposed **C**lassification-based **D**erivative-free optimization method for **H**ybrid optimal control (**CDH**).

The algorithm is presented in Alg.1. With a hybrid automaton H, an objective function J, the initial control mode v_0 and the maximum search bound $\mathcal{D}$ as inputs, the algorithm returns an optimal trajectory ξ^* in the bound $\mathcal{D}$ by Def.3. Firstly, the algorithm initializes the optimal trajectory ξ^*, the optimal value $value^*$, the mode sequence $\mathbf{v}$ and the current mode v (line 2). Then, it iteratively generates mode sequences $\mathbf{v}$ by depth-first search starting from v_0 with the given maximum search depth $\mathcal{D}$ (line 3-16). For each candidate mode sequence $\mathbf{v}$, we encode the constraints along this path into penalty function p (line 4) and construct a new function $f_{\mathbf{v}}$ as the objective function of the minimization problem by combining

Algorithm 1 The Overall Bounded Algorithm for Hybrid Optimal Control Problem

Input:
H: Hybrid Automaton; J: Objective Function; v_0: Start mode; $\mathcal{D}$: The maximum search depth;

1: **function** SYNTHESIS(H,J,v_0,$\mathcal{D}$)
2: Initialize $\xi^* = null$,$value^* = +\infty$, $\mathbf{v} = v_0$, $v = v_0$
3: **while** true **do**
4: Encode constraints of $\mathbf{v}$ into penalty functions p
5: Construct a new objective function $f_{\mathbf{v}}$ by combining J and p
6: $\mathbf{c}, value = \mathbf{CDH}(\mathbf{v}, f_{\mathbf{v}})$
7: **if** $value \leq 0$ **and** $value < value^*$
8: $\xi^* = \mathbf{v}, \mathbf{c}$
9: $value^* = value$
10: **if** v does not have unvisited successor **or** $|\mathbf{v}| \geq \mathcal{D}$ **or** $value > 0$
11: **if** $v == v_0$ **then break**
12: **else** backtrack
13: **else**
14: $v' = getNextSuccessor(v)$
15: $\mathbf{v} = \mathbf{v}, v'$
16: $v = v'$
17: **return** ξ^*
18: **end function**

p and J (line 5). The details of transformations will be introduced in sect.3.2. Then, we provide the new minimization problem to the bottom-layer **CDH** algorithm (line 6), which will be explained in detail in sect.3.3 for optimal control configurations synthesis.

Basically, **CDH** returns the best control configuration $\mathbf{c}$ found for the current control sequence $\mathbf{v}$ and its evaluation value $value$. Specifically, we split the range of the return value to distinguish infeasible and feasible paths. If the control sequence is feasible, its returned $value$ will be non-positive, and if the returned $value$ is smaller than the current minimum value, we will update ξ^* and $value^*$ accordingly (line 8-9) and continue the search. Otherwise, if the returned $value$ is positive, it means the derivative-free engine fails to find a control configuration to make the current control sequence feasible. Thus, we backtrack to the previous mode (line 12).

Since we have set a maximum search depth $\mathcal{D}$, the number of candidate mode sequences is finite. As a result, the search will finally stop (line 11) and the optimal trajectory ξ^* is returned (line 17) when all the sequences have been investigated. The next section will introduce how we synthesize the optimal control configuration for each fixed mode sequence.

3.2 From Optimal Control Synthesis To Minimization

We now introduce how we synthesize optimal control configurations. When $\mathbf{v}$ is fixed, the hybrid optimal control problem in Def.5 becomes the following $\mathbf{v}$-fixed hybrid optimal control problem.

Definition 6. *A* $\mathbf{v}$*-fixed hybrid optimal control problem* $\mathcal{P}_{\mathbf{v}}$ *is to find the optimal control configuration* $\mathbf{c}$ *along control mode sequence* $\mathbf{c}$ *that*

$$\begin{aligned} &minimize\ J_{\mathbf{v}}(\mathbf{c}) = J(\mathbf{v},\mathbf{c}) = \phi(\mathbf{x}^{(\mathbf{v},\mathbf{c})}) + \sum_{i=0}^{|\mathbf{v}|-1} L_i(v_i,\mathbf{x},\mathbf{u}) \\ &s.t.\ \Psi(\mathbf{v},\mathbf{c})\ is\ true \end{aligned}$$

To solve the new $\mathbf{v}$-fixed hybrid optimal control problem, we transform it into a minimization problem that can be solved by derivative-free optimization (DFO) methods. We first give the definition of a minimization problem.

Definition 7. *Let X denote a solution space that is a compact subset of $\mathbb{R}^n$, and $f : X \to \mathbb{R}$ is a function of $\mathbf{x}$. The **minimization problem** of f is to find a solution $\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) \in X\ s.t.\ \forall \mathbf{x} \in X : f(\mathbf{x}^*) \leq f(\mathbf{x})$.*

According to the above definition, we need to construct a function f to combine the evaluation of the trajectory on both the feasibility requirement and the original objective function. To do so, we design som specific **penalty functions** to encode the path constraints along $\mathbf{v}$, defined in Def.2 and Def.4, into penalty functions to report a positive **penalty value** if some path constraints are not satisfied.

We first consider the constraint in the form: $\mathbb{C} : \psi(\mathbf{x}) \sim 0$, where $\sim \in \{>, <, \leq, \geq, ==\}$, and $\psi(\mathbf{x})$ denotes a function on $\mathbf{x}$. Next, we define an indicator function $\mathbb{I}[\mathbb{C}]$ which returns value 1 when $\mathbb{C}$ is true, and 0, otherwise. Then, for the constraint $\mathbb{C}$ where $\sim \in \{\leq, \geq, ==\}$, we define its penalty function as $p(\mathbb{C}) = \mathbb{I}[\neg(\psi(\mathbf{x}) \sim 0)] \cdot |\psi(\mathbf{x})|$. For the special case where $\sim \in \{<, >\}$, and $\psi(\mathbf{x})$ happens to be equal to 0, we define $p(\mathbb{C}) = \mathbb{I}[\neg(\psi(\mathbf{x}) \sim 0)] \cdot |\psi(\mathbf{x})| + \mathbb{I}[\psi(\mathbf{x}) == 0]$. Combination constraints connected with $\vee$ or $\wedge$ are also considered and the penalty values are computed recursively

$$\begin{aligned} p(\mathbb{C}_1 \vee \mathbb{C}_2) &= \min(p(\mathbb{C}_1), p(\mathbb{C}_2)) \\ p(\mathbb{C}_1 \wedge \mathbb{C}_2) &= p(\mathbb{C}_1) + p(\mathbb{C}_2) \end{aligned}$$

In addition, as we can see from Def.2, constraints from $C(v)$ involving continuous dynamics are in the form: $\widetilde{\mathbb{C}} : \forall t \in [0, \tau], \mathbb{C}$. To compute the penalties of such constraints, we propose a numerical ODE method. For the dwell time τ, we use δ_s as the time step and compute the values of $\mathbf{x}$ with different units of δ, up to τ/δ_s. Then, we compute its penalty at each time step and finally sum all the penalty values up $p(\widetilde{\mathbb{C}}) = \sum_{i=0}^{\tau/\delta_s} p(\mathbb{C})$.

To distinguish between feasible and infeasible solutions, our function f of the minimization problem conducts a twofold evaluation of a control configuration $\mathbf{c}$.

- If $\mathbf{c}$ cannot satisfy all the constraints, $f(\mathbf{c})$ is the sum of penalties of all the constraints, which is always positive.
- On the other hand, if all the constraints are satisfied, the penalty value of $\mathbf{c}$ is 0. Then, we compute the original objective function value $J_{\mathbf{v}}(\mathbf{c})$ and shift it to the non-positive region to make the value smaller than the infeasible cases. Without loss of generality, we can assume the objective function J is upper-bounded by real value $\mathcal{K}$. So we define a **mapping function** $\mathcal{M}(\mathbf{c}) = J_{\mathbf{v}}(\mathbf{c}) - \mathcal{K} = J(\mathbf{v},\mathbf{c}) - \mathcal{K}$ to map the original value to a non-positive one.

Finally, we give the following minimization problem:

Definition 8. *Let X denote the control configuration solution space which is a compact subset of $\mathbb{R}^n$, and $f_{\mathbf{v}} : X \to \mathbb{R}$ is a minimization objective function of $\mathbf{c}$. We define $f_{\mathbf{v}}$ as*

$$f_{\mathbf{v}}(\mathbf{c}) = \mathbb{I}[\Psi(\mathbf{v},\mathbf{c})]\mathcal{M}(\mathbf{c}) + \mathbb{I}[\neg\Psi(\mathbf{v},\mathbf{c})]p(\Psi(\mathbf{v},\mathbf{c}))$$

where $\mathbb{I}[\cdot]$ is the indicator function that returns 1 when $\cdot$ is true and 0, otherwise. $\mathcal{M}$ and p are the mapping function and the penalty function respectively, $\Psi(\mathbf{v},\mathbf{c})$ is the constraint defined in Def.4.

Then we have the following lemma about the transformation:

Lemma 1. *Given a $\mathbf{v}$-fixed hybrid optimal control problem $\mathcal{P}_{\mathbf{v}}$ by Def.6 and its corresponding minimization objective function $f_{\mathbf{v}}$ by Def.8, we have $\mathbf{c}^* = \arg\min_{\mathbf{c}} f_{\mathbf{v}}(\mathbf{c})$ is the optimal solution of $\mathcal{P}_{\mathbf{v}}$.*

Proof. To prove Lemma.1, we just need to prove that, for the $\mathbf{v}$-fixed hybrid optimal control problem $\mathcal{P}_{\mathbf{v}}$ given in Def.6, we have $\arg\min_{\mathbf{c}} f_{\mathbf{v}}(\mathbf{c}) = \arg\min_{\mathbf{c}} J_{\mathbf{v}}(\mathbf{c})$.

Firstly, the mapping function $\mathcal{M}(\mathbf{c}) = J_{\mathbf{v}}(\mathbf{c}) - \mathcal{K} \leq 0$ is an affine transformation of $J_{\mathbf{v}}(\mathbf{c})$. So we have $\arg\min_{\mathbf{c}} \mathcal{M}(\mathbf{c}) = \arg\min_{\mathbf{c}} J_{\mathbf{v}}(\mathbf{c})$. On the other side, according to the definition of penalty functions p, the minimization of $f_{\mathbf{v}}(\mathbf{c})$ can only be reached when the sequence is feasible. In this case $f_{\mathbf{v}}(\mathbf{c}) = \mathcal{M}(\mathbf{c})$, according to Def.8. Therefore, we have $\arg\min_{\mathbf{c}} f_{\mathbf{v}}(\mathbf{c}) = \arg\min_{\mathbf{c}} \mathcal{M}(\mathbf{c}) = \arg\min_{\mathbf{c}} J_{\mathbf{v}}(\mathbf{c})$. □

3.3 Classification-based Derivative-free Optimization Method for Hybrid Optimal Control (CDH)

To solve complex hybrid optimal control problems, we can take advantage of existing mature sampling-based derivative-free methods, including genetic algorithms [21], randomized local search [32], Bayesian optimization methods [10], cross-entropy methods [13] and so on. For example, we have implemented cross-entropy method and genetic algorithms to solve the problem defined in Def.8, c.f. sect.4.3. Specifically, Yu [40] proposed a classification-based derivative-free optimization (DFO) framework, with good performance and sufficient theoretical analysis. In this section, by adapting this framework, we propose a new **C**lassification-based Derivative-free optimization algorithm for **H**ybrid optimal control domain (**CDH**).

In most cases, DFO algorithms works in a **sample-evaluate-learn-style** way. They firstly sample a large set of candidate solutions and then evaluate these samples by the objective function. After that, the samples and their evaluation results are learned to refine the search space to sample better solutions in the next iteration. As we have introduced the objective function that will be used in the **evaluate** stage in Def.8, we will now introduce how we **sample** and **learn** the candidate solutions in our proposed **CDH**.

3.3.1 Sample a Control Configuration Sequence. We first introduce how we sample a solution. As we need to find the optimal configuration sequence $\mathbf{c}^*$ for the fixed $\mathbf{v}$, therefore, $\mathbf{c}$ is the sample here. The detailed sampling procedure is depicted in Alg.2.

Taking the fixed control mode sequence $\mathbf{v}$ and the sampling model $h_{\mathbf{v}}$ as inputs, the sampling function returns a sample of the corresponding control configuration sequence $\mathbf{c} = c_0, c_1, ..., c_{|\mathbf{v}|-1}$ for $\mathbf{v}$, where $c_i = (\tau_i, \mathbf{u}_i)$ is composed of the dwell time and control inputs in control mode v_i. Here, $h_{\mathbf{v}}$ maintains the solution space of control configuration sequences for $\mathbf{v}$, which is a multidimensional region. For a specific control mode v in the sequence $\mathbf{v}$, we sample its control configuration $\mathbf{c}[v]$ from $h_{\mathbf{v}}[v]$. Similarly, for the dwell time τ of $\mathbf{c}[v]$, denoted as $\mathbf{c}[v][\tau]$, we sample it from $h_{\mathbf{v}}[v][\tau]$ that is the solution space interval for the dwell time τ.

Algorithm 2 Sampling and Learning

Input:
$\mathbf{v}$: The fixed control mode sequence; $h_{\mathbf{v}}$: The sampling model; S: Set of samples; $f_{\mathbf{v}}$: The objective function for v;

1: **function** SAMPLING($\mathbf{v}$,$h_{\mathbf{v}}$)
2: $\mathbf{c}$ = [][] // initialize $\mathbf{c}$
3: **for** each $v \in \mathbf{v}$ **do**
4: $\mathbf{c}[v][\tau]$ = *Uniform_Sampling*($h_{\mathbf{v}}[v][\tau]$)
5: **for** each $u \in U$ **do**
6: $\mathbf{c}[v][u]$ = *Uniform_Sampling*($h_{\mathbf{v}}[v][u]$)
7: **return** $\mathbf{c}$
8: **end function**
9:
10: **function** LEARNING($\mathbf{v}$,$h_{\mathbf{v}}$,S,$f_{\mathbf{v}}$)
11: Pos = *update*(Pos,S,$f_{\mathbf{v}}$) // choose m_{pos} best samples according to evaluation results
12: $Neg = S \setminus Pos$
13: **while** $Neg \neq \emptyset$ **do**
14: $\mathbf{c}_{pos}$ = randomly choose from Pos
15: v = randomly choose a mode from $\mathbf{v}$
16: I = randomly choose a dimension // τ or any $u \in U$
17: **for** each $\mathbf{c} \in Neg$ **do**
18: **if** $\mathbf{c_{pos}}[v][I] \geq \mathbf{c}[v][I]$ // large value is better
19: $temp$ =random($\mathbf{c}[v][I], \mathbf{c}_{pos}[v][I]$)
20: **if** $temp$ >GetLowerBound($h_{\mathbf{v}}[v][I]$)
21: SetLowerBound($h_{\mathbf{v}}[v][I], temp$)
22: $Neg = Neg - \{c\}$
23: **else** // small value is better
24: $temp$ =random($\mathbf{c}_{pos}[v][I], \mathbf{c}[v][I]$)
25: **if** $temp$ <GetUpperBound($h_{\mathbf{v}}[v][I]$))
26: SetUpperBound($h_{\mathbf{v}}[v][I]$), $temp$)
27: $Neg = Neg - \{c\}$
28: **return** $h_{\mathbf{v}}$
29: **end function**

The sampling procedure starts with initializing the control configuration sequence $\mathbf{c}$ as an empty list. After that, we iterate to sample the control configuration $\mathbf{c}[v]$ for each control mode v in the mode sequence $\mathbf{v}$ (line 3-6). We first uniformly sample the dwell time $\mathbf{c}[v][\tau]$ of v from $h_{\mathbf{v}}[v][\tau]$ (line 4) and then sample values for each control input $u \in U$ (line 5-6). When the loop stops, we can get a sampled control configuration sequence $\mathbf{c}$.

3.3.2 Learn From Evaluated Samples. After sampling a large set of candidate solutions and evaluating them by computing their objective function values, it comes to the learning stage. The detailed algorithm is also presented in Alg.2.

With the fixed control mode sequence $\mathbf{v}$, the sampling model $h_{\mathbf{v}}$, the set of samples S and the objective function $f_{\mathbf{v}}$ as inputs, the learning function returns a new refined sampling model. As the derivative-free algorithm adapted here is based on classification, we first label each sample by positive or negative according to their objective function values and update the set Pos which consists of the latest m_{pos} best samples (line 11). Other samples are stored in the set Neg (line 12). Then, it iterates to refine the space solutions of $h_{\mathbf{v}}$ based on existing positive samples until the new sample model cannot generate any negative samples in Neg anymore.

In each iteration, we first randomly choose a positive sample from Pos as the baseline solution to learn from (line 14). Meanwhile, we randomly select a mode v from $\mathbf{v}$ (line 15) and a dimension I (line 16) to change the solution space of the control configuration. Here, the dimension I can either be the dwell time τ or any control input $u \in U$ for v. Then, for each negative sample $\mathbf{c}$ in Neg, we check the value of $\mathbf{c}_{pos}[v][I]$ and $\mathbf{c}[v][I]$. If $\mathbf{c}_{pos}[v][I]$ is larger (line 18), it means the control configuration of the mode v in the dimension I prefers larger values so that it can be evaluated as better samples. Therefore, we randomly generate a value $temp$ from the region $(\mathbf{c}[v][I], \mathbf{c}_{pos}[v][I])$ (line 19) and compare it to the current lower bound of $h_{\mathbf{v}}[v][I]$. If $temp$ is larger, we can shrink the lower bound of $h_{\mathbf{v}}[v][I]$ to $temp$ by calling the function *SetLowerBound* (line 21). If a negative sample c successfully contributes to refine the search space, we can remove it from Neg (line 22) since the new model won't generate c again in next iterations.

If $\mathbf{c}[v][I]$ is larger compared to $\mathbf{c}_{pos}[v][I]$ (line 23), it means the value of the control configuration in the dimension I of the mode v prefers smaller values. In this case, we need to shrink the upper bound of $h_{\mathbf{v}}[v][I]$ to a smaller value in the same way we introduced in line 18-22. After all the negative samples are eliminated from Neg, we return $h_{\mathbf{v}}$ to sample better solutions in next iterations.

3.3.3 Put together. After introducing how we sample candidate control configuration sequences and how to learn from evaluated samples, we now give the complete **CDH** algorithm. The detailed algorithm is depicted in Alg.3. With a fixed mode sequence $\mathbf{v}$ and the objective function of corresponding minimization problem $f_{\mathbf{v}}$ as inputs, the algorithm returns the optimal control configuration sequence $\mathbf{c}^*$ for $\mathbf{v}$ along with its evaluation value $f_{\mathbf{v}}(\mathbf{c}^*)$. Firstly, we initialize the sample size m, the current optimal result $\mathbf{c}^*$ and the iteration number t (line 2). Meanwhile, we initialize the sampling model h_0 by default ranges (line 3).

In each iteration t, we sample m candidate solutions (line 5). Then, each sample $\mathbf{c_i} \in S_t$ will be evaluated by function $f_{\mathbf{v}}(\mathbf{c})$. The configuration that has the minimum value will be used to update $\mathbf{c}^*$ (line 6). After that, h_t is updated by learning from existing configurations (line 7), to sample better configurations in the next iteration. This sampling and learning procedure continues until the objective function converges. Finally, we return the optimal control configuration sequence $\mathbf{c}^*$ for $\mathbf{v}$ along with its evaluation value $f_{\mathbf{v}}(\mathbf{c}^*)$ (line 9).

3.4 Discussion of Optimality

In [39, 40], the authors had given sufficient theoretical analysis of the classification-based DFO framework. They first gave the following definition of $(\epsilon, \delta)-$query complexity to measure the performance of the sampling-based optimization methods.

DEFINITION 9. *Given a function f, an algorithm $\mathcal{A}$, $0 < \delta < 1$ and $\epsilon > 0$, the $(\epsilon, \delta)-$**query complexity** is the number of calls to f such that with probability at least $1 - \delta$, $\mathcal{A}$ finds at least one solution $\widetilde{x} \in X \subset \mathbb{R}^n$ satisfying $f(\widetilde{x}) - f(x^*) \leq \epsilon$, where $f(x^*) = min_{x \in X} f(x)$.*

Algorithm 3 Hybrid Optimal Control Via Classification

Input:
$\mathbf{v}$: The control mode sequence; $f_{\mathbf{v}}$: The objective Function;
1: **function** CDH($\mathbf{v}$,$f_{\mathbf{v}}$)
2: Initialize $m \in N^+$, $\mathbf{c}^* = null$, $t = 0$
3: Initialize the model h_0 by default ranges;
4: **while** *not convergent*
5: $S_t = \bigcup_{0}^{m-1} \{\textbf{Sampling}(\mathbf{v}, h_t)\}$
6: $\mathbf{c}^* = argmin_{\mathbf{c}\in S_t \cup \{\mathbf{c}^*\}} f_{\mathbf{v}}(\mathbf{c})$;
7: h_{t+1} = **Learning**($\mathbf{v}$, h_t,S_t,$f_{\mathbf{v}}$);
8: $t = t + 1$;
9: **return** $\mathbf{c}^*$ and $f_{\mathbf{v}}(\mathbf{c}^*)$
10: **end function**

Then, they proved the following lemma to give an upper bound of the $(\epsilon, \delta)-$query complexity of the general classification-based optimization framework to converge to the optimal result.

Lemma 2. *Given a function f, $0 < \delta < 1$ and $\epsilon > 0$, the $(\epsilon, \delta)-$query complexity of a classification-based optimization algorithm is upper bounded [40].*

The detailed bound of the query complexity in Lemma. 2 and its proof are presented in the appendix of [40]. Due to space limitations, please refer to [40] for details. Finally, based on existing theoretical results, we give the following theorem about the optimality of our hybrid optimal control algorithm and give its proof sketch.

Theorem 1. *Given a hybrid optimal control problem $\mathcal{P}$ by Def.5 and a depth bound $\mathcal{D}$, the $(\epsilon, \delta)-$**query complexity** of Alg.1 to synthesize the optimal trajectory ξ^* in the bound $\mathcal{D}$ is upper bounded.*

Proof. As introduced in sect.3.1, with the search depth upper bounded by $\mathcal{D}$, Alg.1 can find all the candidate sequences that are not longer than $\mathcal{D}$. We denote the finite number of candidate sequences as $\mathcal{N}_{\mathcal{D}}$. For each candidate mode sequence $\mathbf{v}$, the optimal control synthesis of configurations is transformed into an equivalent minimization problem by Def.8 and Lemma.1. Then, according to Lemma.2, the $(\epsilon, \delta)-$**query complexity** of the **CDH** algorithm to find the optimal solution of the minimization problem is upper bounded. We denote it as $Complexity(\mathbf{v})$. Since we explore all the candidate control mode sequences and solve each minimization problem with bounded $(\epsilon, \delta)-$**query complexity**, the overall $(\epsilon, \delta)-$**query complexity** of Alg.1 is upper bounded by $\sum_{i=0}^{\mathcal{N}_{\mathrm{D}}-1} Complexity(\mathbf{v}_i)$. □

Generally speaking, it's very difficult to claim the optimality of results generated by CDH. The argument is twofold.

- Theoretically, based on Theorem.1, the underlying DFO algorithm can synthesize the optimal trajectory with a high probability given enough query time.
- Practically, as the query upper bound could be enormous, we set a termination condition for Alg.3 to reach an **approximate** optimal solution. Our termination condition is that the values of consecutive iterations keep stable for a long time (e.g. 50 iterations in our setting), or it reaches the maximum iteration steps (e.g. 500 steps).

3.5 Multi-Phase Control Synthesis-Based Tradeoff Between Optimality and Efficiency

As we introduced above, Alg.1 searches for an optimal trajectory with the maximum search depth $\mathcal{D}$. To further improve the scalability of our method to handle complex problems that need large bound $\mathcal{D}$, we propose a multi-phase method based on Alg.1 to solve the whole problem piece by piece in a divide-and-conquer manner.

We split the synthesis problem of the complete trajectory into multiple synthesis problems of sub-trajectories. For each sub-problem, we set a relatively small bound $\mathcal{D}'$ ($\mathcal{D}' < |\xi^*|$) and call Alg.1 to generate an optimal sub-trajectory in the bound of $\mathcal{D}'$. After reaching the control target, we merge these sub-trajectories to become a final feasible solution. Before introducing the algorithm, we first define the concatenation of two trajectories.

Definition 10. *The **concatenation** of two control trajectories $\xi_i = (v_{i0}, c_{i0}), \ldots, (v_{im}, c_{im})$ and $\xi_j = (v_{j0}, c_{j0}), ..., (v_{jn}, c_{jn})$ is the new trajectory*

$$\xi_i \parallel \xi_j = (v_{i0}, c_{i0}), \ldots, (v_{im}, c_{im}), (v_{j0}, c_{j0}), ..., (v_{jn}, c_{jn})$$

Now we introduce how we solve complex control problems, as presented in Alg.4. It first initializes the search bound $\mathcal{D}'$ to a relatively small value (line 1), which is the number of control modes in hybrid systems in our setting. Meanwhile, it initializes the final trajectory $\xi = null$ and the current mode $v = v_{init}$ (line 2). Then, it iterates to generate sub-trajectories by calling **Synthesis** from Alg.1 (line 5) until the control target is fulfilled or it reaches the maximum iteration times $\mathcal{T}$. Every time a new sub-trajectory is generated, it merges the newly found sub-trajectory ξ_s to the final trajectory ξ (line 6) and updates the start mode of the next search to the last state of the previous sub-trajectory (line 7). When the loop stops, the algorithm will return the final trajectory ξ (line 9).

Algorithm 4 Algorithm of Multi-Phase Control Synthesis and Concatenation for Complex Hybrid Control Problems

Input:
H: The hybrid automaton;
J: The objective function;
$\mathcal{T}$: Maximum iteration times;
1: Initialize a relatively small search bound $\mathcal{D}'$
2: Initialize $\xi = null$, $v = v_{init}$, $t = 0$
3: **while** not fulfill the control target **and** $t < \mathcal{T}$
4: Initialize $\xi_s = null$.
5: ξ_s = **Synthesis**($H, J, v, \mathcal{D}'$) // Call Alg.1
6: $\xi = \xi \parallel \xi_s$ // concatenation
7: $v = LastState(\xi_s)$ // update
8: $t = t + 1$
9: **return** ξ.

Discussion about Soundness and Optimality. Here, we give some discussions about the soundness and optimality of the multi-phase control synthesis and concatenation.

Soundness: Whether the trajectory resulting from the concatenation is still valid? As we can see in Alg.4, every time we merge a new sub-trajectory into the end of the existing trajectory, we update the states of the system simultaneously. As a result, the values we

use to generate new sub-trajectories are all extended from the last state of the existing trajectory, and the whole trajectory after the concatenation is still valid.

Optimality: Whether the trajectory resulting from the concatenation is still optimal? Actually, this is a tradeoff between performance and optimality. For any hybrid optimal control problems $\mathcal{P}$, if the length of its optimal trajectory has not reached bound $\mathcal{D}'$, $|\xi^*_{\mathcal{P}}| \leq \mathcal{D}'$, we can approximate the optimal trajectory by Theorem.1. In this case, Alg.4 will stop after just one iteration. On the other side, if $|\xi^*_{\mathcal{P}}| > \mathcal{D}'$, Alg.4 can synthesize a feasible solution very efficiently. But the result cannot be guaranteed to be as optimal as the integral control anymore, but only piecewise optimal instead.

4 EXPERIMENT

Now we present our experiments on two complex systems: the vehicle system and the quadcopter drone system. These two classical systems were given in [22]. All the experiments here are conducted on an Ubuntu machine with a 4 core, 2.7 GHz CPU and 8GB RAM.

4.1 Vehicle Optimal Control

The vehicle control system has two control modes: *Forward* and *Turn*. The flow conditions are given as follows:

$$\dot{x} = v_{el}cos\theta \quad \dot{y} = v_{el}sin\theta \quad \dot{v_{el}} = u_1 \quad \dot{\theta} = u_2$$

where x and y are the Cartesian coordinates of the car, v_{el} and θ are the velocity and the angular orientation with respect to the x-axis. u_1 and u_2 are control inputs corresponding to the velocity and angular acceleration respectively. We set $u_1 \in [-0.3, 0.3]$, $u_2 = 0$ in *Forward* mode and $u_1 = 0$, $u_2 \in [-\frac{\pi}{6}, \frac{\pi}{6}]$ in *Turn* mode. As for invariants of vehicle control systems, we require $v_{el} \leq 8$ in *Forward* mode and require $-1.5 \leq \theta \leq 3.14$ in *Turn* mode.

About discrete transitions, the system can jump from one control mode to all the other control modes, including the current mode itself. All the transitions have no guard conditions. The control target of the hybrid optimal control problem is to drive the car from the initial waypoint w_0 to a target waypoint $\hat{w}$ while avoiding all the boundaries and obstacles. The given objective function is:

$$J(\xi) = \|(x, y)^{(\xi)} - \hat{w}\|_1 + \gamma \sum_{i=0}^{|\xi|-1} I(v_{el})L_i(\tau_i, u_1, u_2)$$

which consists of a control target function denoting the Manhattan distance between the current position and the target position and cost functions. The cost function also has two parts: the running cost L respective to τ_i and control inputs u_1, u_2, and a piecewise function I with respect to v_{el}. Usually, it costs more fuel for a car to get the same acceleration when it has a higher speed. So we set $I = 1$ if $0 < |v_{el}| \leq 5$, $I = 2$ if $5 < |v_{el}| \leq 10$ and $I = 3$ if $|v_{el}| > 10$. Obviously, I is not continuous everywhere to compute gradients[1].

We calculate the optimal control trajectory of the vehicle for three control missions shown in Fig.2, where 2(a), 2(b) and 2(c) present the scenes of *Go Straight*, *Turn Right* and *Cross Intersection*, respectively. The constraints of these control missions consist of invariants of each control mode and external constraints from obstacles. It takes around 200 iterations for our algorithm to synthesize the approximate optimal trajectories for these control missions.

[1]Here, we use the piece-wise control functions to show our method can handle discontinuous objective functions. Our method can also work with normal objective functions. Therefore, it's not a restriction compared to the state-of-the-art hybrid optimal control algorithms.

The moving trajectories of the vehicle are also presented in Fig.2. In each plot, blue points represent the *Forward* mode and orange points represent the *Turn* mode. As we can see from the plots, under the optimal control solutions generated by our methods, the vehicle successfully and smoothly fulfills the control targets without any collision into the boundaries or the obstacles.

We repeat the control missions for 100 times. In each round, the initial states, target points and objective functions are all the same. Table.1 shows the statistical results. The success rates of three control missions of the vehicle are all 100%, which shows the feasibility of our algorithm to solve complex hybrid optimal control problems. Meanwhile, our method is efficient since the average time costs to synthesize optimal solutions of three control missions are 13.1 secs, 12.9 secs, 9.1 secs, respectively. What's more, in terms of the stability, our method also performs well that the standard deviations of time costs for three tasks are all less than 4 seconds.

4.2 Quadcopter Drone Optimal Control

In this section, we solve the optimal control of a classical quadcopter drone system from [22]. In the system, x and z denotes the position of the drone along the horizontal axis and the height above the ground. θ means the roll angle of the drone. The HA model of this system has three modes: *Cruise*, *Rise* and *Dive*. Each of them has flow equations as follows

$$\ddot{x} = \frac{1}{M}sin(\theta)\sum_{k=1}^{3} T_k, \quad \ddot{z} = \frac{1}{M}cos(\theta)\sum_{k=1}^{3} T_k - g$$
$$\ddot{\theta} = \frac{L}{I_z}(T_1 - T_3)$$

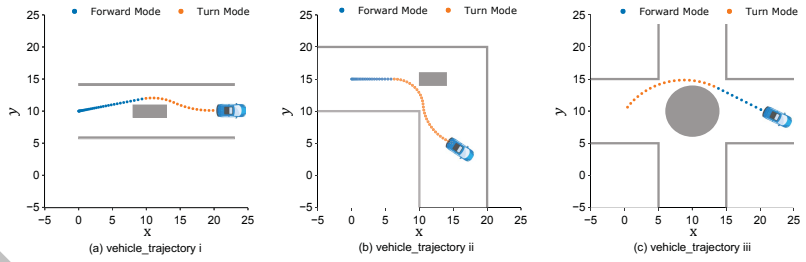
where T_1 and T_3 are thrusts applied at the opposite ends of the quadcopter along the x-axis, while T_2 is the sum of the thrusts of other rotors at the center of mass of the quadcopter. Meanwhile, M, I_z, L represent the mass, moment of inertia about z-axis and the distance from center of mass of each of the rotors T_1, and T_3. For the *Cruise* mode, we set $T_1 = T_3 = 0$, $T_2 \in [0, 16]$. For the *Rise* mode, we set $T_1 = 0$, $T_2 = Mg$, $T_3 \in [0, 2]$ and for the mode *Dive*, we set $T_1 \in [0, 2]$, $T_2 = Mg$, $T_3 = 0$. As for invariants of drone control systems, we require $-2 \leq \dot{z} \leq 2$ in all the three control modes.

About discrete transitions, similar to the vehicle control system, the drone can jump from one control mode to all the other control modes, including the current mode itself. All the transitions have no guard conditions either. The objective of the problem here is to control the drone to fly from the initial waypoint w_0 to the target way point $\hat{w}$. The objective function is listed below:

$$J(\xi) = \|(x, z)^{(\xi)} - \hat{w}\|_1 + \gamma \sum_{i=0}^{|\xi|-1} I(\dot{x}, \dot{z})L_i(\tau_i, T_1, T_2, T_3)$$

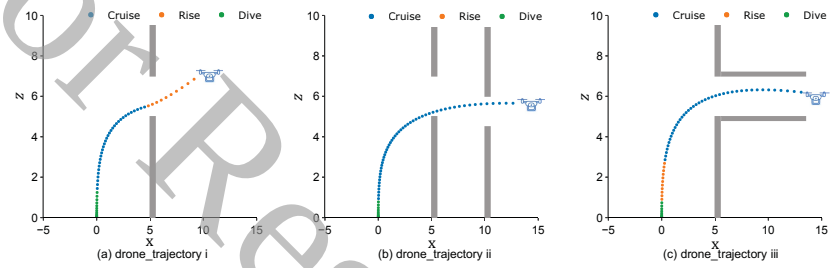
The objective function here is also composed of a control target function denoting the distance and cost functions. As introduced in the vehicle control problem, we also define a piecewise function I respective to the horizontal velocity $\dot{x}$ and the vertical velocity $\dot{z}$ to balance the fuel cost when the drone has different speeds.

We calculate optimal trajectories for three control missions which are shown in Fig.3. It takes around 300 iterations for our algorithm to synthesize the approximate optimal trajectories. In each plot, blue points, orange points and green points are flying trajectories regarding the *Cruise* mode, the *Rise* mode and the *Dive* mode, respectively. As we can see from the plots, the drone can avoid all the obstacles according to the optimal control solutions generated by our method.



In each plot, blue points and orange points are moving trajectories of the vehicle with respect to the *Forward* mode and *Turn* mode, respectively.

Figure 2: The moving trajectories of the vehicle for three control missions: (a) *Go Straight*, (b) *Turn Right*, (c) *Cross Intersection*.



In each plot, blue points, orange points and green points are flying trajectories of the drone with respect to the *Cruise* mode, *Rise* mode and *Dive* mode, respectively.

Figure 3: The flying trajectories of the drone for three control missions: (a) *Turn*, (b) *Two Turns*, (c) *Keep Forward*.

We also repeat the missions for 100 times and present the statistical data in Table.1. The success rates of synthesizing feasible solutions remain 100%. In terms of efficiency, although it costs more computation time than vehicle control missions, the trajectories can still be synthesized within 50 seconds. Moreover, our method keeps synthesizing trajectories stably since the maximum standard deviation of time costs of three tasks is about only 9 seconds.

Table 1: Statistical results of repeating experiments.

	S. Rate	*Mean*	*Std*	*Max*	*Min*
Vehicle (a)	100%	13.1	2.0	17.8	7.7
Vehicle (b)	100%	12.9	2.1	19.8	7.6
Vehicle (c)	100%	9.1	3.6	30.4	3.7
Drone (a)	100%	37.2	5.0	59.3	27.7
Drone (b)	100%	43.1	4.9	57.4	32.4
Drone (c)	100%	49.0	9.1	69.4	34.3

S.Rate stands for the success rate to generate the final optimal feasible solutions.
Mean/Std means the mean/standard deviation value of time (seconds).
Max/Min means the maximum/minimum time (seconds).

4.3 Compare CDH With Existing Methods

Beside CDH, we also adapt the cross-entropy method (*CEM*) [13] and the genetic algorithm (*GA*) [21] to apply them into our non-differentiable hybrid optimal control synthesis framework. What's more, we implement another two classical sampling-based hybrid control algorithms, Rapidly-exploring Random Trees (*RRTs*) [8] and robust model predictive control based on Monte Carlo and Rejection-Sampling (*MCRS*) [18] to solve the above six control problems. More details about these two methods are given in sect.5. Due to space limitations, we skip the implementation details here. We synthesize optimal solutions for the above six control missions using these methods for 100 times respectively. The comparison results are depicted in Fig.4.

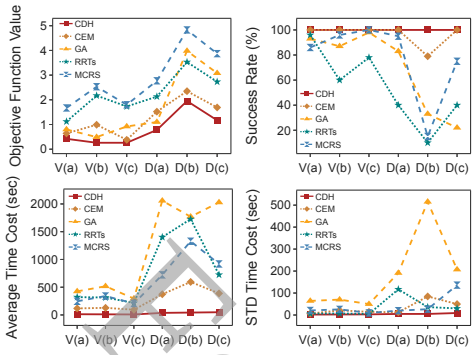
Firstly, we compare the final objective function values achieved by these five methods. We can see that the *CDH* achieves the best objective function values on all the cases. In terms of success rates, *CDH* is the only one that successfully finds optimal solutions for all the control missions in all rounds. For the control missions of $D(a), D(b), D(c)$, which are relatively difficult to solve, the other four methods perform much worse than *CDH*. Especially for $D(b)$, the success rates of *GA*, *RRTs* and *MCRS* are all lower than 50%.

We also give the analysis of time costs. Our *CDH* method outperforms other methods on all the cases. Especially on the drone control missions, only *CDH* generates solutions within one minute. In comparison, *CEM* needs about 10 minutes (600 seconds) and other three methods need over half an hour (1800 seconds). Meanwhile, from the perspective of the standard deviation of time costs, *CDH* is more stable than other four methods as well.

The above experimental results confirm our DFO-based control synthesis method can be easily equipped by existing DFO methods. It also demonstrates that our *CDH* algorithm outperforms existing sampling-based methods substantially.

4.4 Multi-Phase Control Missions

As we introduced in sect.3.5, for complex hybrid control problems that need many control modes switches, if we simply raise $\mathcal{D}$, the number of potential control mode sequences may grow quickly. So we propose the multiple-phase control method to make a tradeoff between optimality and efficiency. In this subsection, we give two complex control missions of the vehicle and drone and compare



In each plot, *V(a), V(b), V(c)* stand for the the three control missions of the vehicle and *D(a), D(b), D(c)* stand for the three control missions of the drone.

Figure 4: The comparison results on objective function values, success rates, average and standard deviation of time costs of CDH, CEM, GA, RRTs and MCRS.

the performance of synthesizing trajectories between the original (integral) version and the multi-phase control.

For the vehicle, we require it to reach the waypoint while avoiding a number of obstacles, as shown in Fig.5. While for the drone control, we generate two complex piecewise functions as boundaries of the flying region (represented by grey lines in Fig.6(a)). The drone is only allowed to fly in the enclosed tight space. As the state space of these two control missions is much more complicated, it needs more mode switches to find a feasible solution.

We first set $\mathcal{D} = 6$ and solve the two problems integrally using Alg.1. The vehicle's trajectory is shown in Fig.5 denoted by a blue car, while the drone's trajectory is shown in Fig.6 denoted by a blue quadcopter. We give results of time costs, numbers of potential mode sequences checked and minimized objective function values in Table.2. For the vehicle control, it explores 12 control sequences and costs 576 seconds to synthesize the final optimal trajectory. The final minimized objective function value is 1.35. While for the drone control, it costs over an hour (3894 seconds) with 189 control sequences checked. The minimized objective function value is 2.23.

Then, we use the multi-phase control synthesis and concatenation algorithm to solve the same problems. For vehicle control, we set $\mathcal{D} = 2$ and the trajectory computed is illustrated in Fig.5, denoted by a green car. As the vehicle system has only two control modes, multi-phase control still checks 12 control sequences during 3 phases. Nevertheless, it saves 88% of time costs since the control sequences checked here are shorter, and the corresponding DFO problems are much easier to solve. The minimized value of the objective function is 1.40, which is increased by 3.7% only.

For drone control, we set $\mathcal{D} = 3$ and the trajectory computed is illustrated in Fig.6, denoted by a green quadcopter. Compared to integral control, multi-phase control only checks 42 control sequences, reduced by 77%. As a result, it dramatically improves the efficiency, by 94%, to synthesize feasible solutions within only 216 seconds. About the optimality, the minimized objective function here is 2.49, which is still close to the integral control.

Table 2: Performance comparison between integral control and multi-phase control.

	Type	$\mathcal{D}$	*P*	*Time (sec)*	*Path Num*	*Minimized J*
Vehicle	Integral	6	1	576	12	1.35
	Multi-Phase	2	3	66 (88%↓)	12 (0%↓)	1.40 (3.7%↑)
Drone	Integral	6	1	3894	189	2.23
	Multi-Phase	3	2	216 (94%↓)	42 (77%↓)	2.49 (11.6%↑)

$\mathcal{D}$ is the search bound. *P* is the number of phases. *Minimized J* stands for the objective function value of *J* after minimized.

5 RELATED WORK

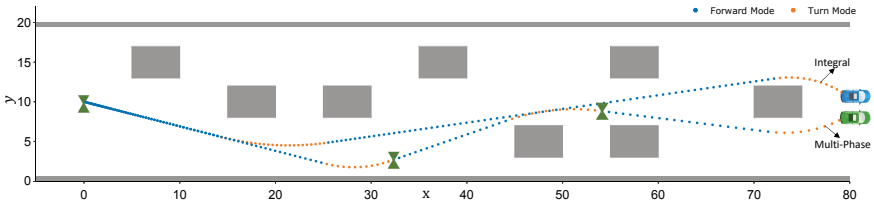
In this section, we make a review of classical hybrid control algorithms for nonlinear systems proposed during the past two decades.

For Differentiable Systems. Here, differentiable means the objective function, constraints are differentiable and control inputs can either be discrete or continuous. Bengea [4] considered the optimal control for switching systems and found sufficient and necessary conditions for the optimality. Xu and Antsakils [37, 38] firstly presented the classical bi-level hierarchical optimization algorithm, based on available theoretical results about necessary conditions for the existence of optimal controls [6, 9, 24, 34]. Egerstedt [16, 17] used a steepest descent algorithm to find the optimal value based on a simpler formula for the gradient. Caldwell [11], Johnson [27] and Liu [31] compared the second-order method to the first-order method and emphasized the importance of the second-order method due to its faster convergence rate in consideration of online applications [15]. In addition, Axelsson [3] and Gonzalez [22, 23] improved the mode insertion method by introducing a single-mode insertion technique and its variant. Ali [1] proposed a method to improve efficiency by changing dwell times of multiple modes at a time.

Beside the conventional gradient-based methods, novel computational methods were also proposed to solve hybrid optimal control problems of differentiable systems. Nilsson [33] proposed a counter-example guided abstraction refinement procedure to synthesize switching protocols. Ding [14] took advantage of the reachability analysis technique to compute the set of states in reach-avoid controllers synthesis. For the specific control problems where the control input space is compact and convex, Zhao [41] proposed using convex relaxtions to synthesize the optimal controller.

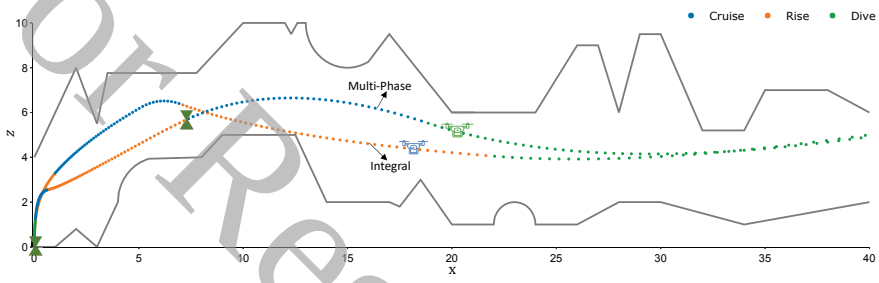
For Non-differentiable Systems. On the other hand, for hybrid optimal control of non-differentiable systems, sampling-based methods have achieved considerable success. Rapidly-exploring Random Trees (RRTs) [30] is one of the classical sampling-based exploration algorithms for quickly searching feasible trajectories. Branicky [7, 8] used RRTs to solve nonlinear control problems and extended them to the case of hybrid systems. However, the above searching graph-based algorithms cannot guarantee the optimality of the synthesized trajectory.

Farahani [18] proposed a robust model predictive control approach which took advantage of Monte Carlo simulation and Rejection-Sampling (MCRS) to solve the worst-case MPC problem. Although MCRS can be used to solve hybrid optimal control problems, its ability to generate feasible trajectories for complex control problems is restricted. To generate maximally-satisfying controllers for STL specifications, Vasile [35] explicitly used bounds on the quantitative satisfaction of a formal specification as a heuristic to guide sampling. The authors gave the asymptotic optimality analysis of



The trajectory computed by integral control is denoted by the blue car, while the trajectory computed by multi-phase control and concatenation is denoted by the green car. For the multiple-phase control, we mark the beginning of each sub-trajectory by a green double-triangle.

Figure 5: Trajectories computed by integral and multiple-phase control for complex hybrid optimal control problems of vehicle.



The trajectory computed by integral control is denoted by the blue quadcopter, while the trajectory computed by multi-phase control and concatenation is denoted by the green quadcopter. For the multi-phase control, we mark the beginning of each sub-trajectory by a green double-triangle.

Figure 6: Trajectories computed by integral and multiple-phase control for complex hybrid optimal control problems of drone.

their proposed method. However, the method was designed only for continuous control, thus cannot be applied to solve hybrid optimal control problems directly.

Sum Up. We now conclude the aims and features of the above classical control synthesis algorithms. As we can see from Table.3, most control synthesis algorithms can support hybrid control, except for maximally-satisfying controllers. Among those hybrid control methods, RRTs aim for feasible trajectories and doesn't belong to hybrid optimal control areas. As for non-differentiable behavior support, sampling-based methods don't require differentiable systems. Finally, compared to MCRS which also supports hybrid optimal control of non-differentiable system, CDH method has sufficient theoretical analysis on query complexity of approximating optimal solutions. On the other hand, as declared in [16], MCRS cannot guarantee to generate feasible trajectories. The experimental comparison results of our CDH and MCRS in sect.4.3 also prove that our CDH method outperforms MCRS significantly. Last but not least, apart from the optimal control synthesis, we also applied such derivative-free optimization methods into falsification areas and achieve promising performance [36].

6 CONCLUSION

In this paper, we propose a practical and efficient sampling-based algorithm to solve arbitrary hybrid optimal control problems. We transform the control synthesis problem into a minimization problem and solve it by adapting the latest classification-based derivative-free optimization framework. To further improve the scalability, we propose the multi-phase control synthesis and concatenation method to solve complex hybrid optimal control problems in a divide-and-conquer manner. We apply our algorithm to synthesize optimal control trajectories for two complex systems, including six basic control missions and two much more complex missions. The result shows our method can handle such complex problems efficiently and outperforms existing methods significantly.

Table 3: Conclusions of aims and features of different control synthesis approaches.

	Hybrid Control	Hybrid Optimal Control	Non-diff. Behavior	Query Complexity
Bi-level [37, 38]	✓	✓	✗	N/A
Second-order [11, 27]	✓	✓	✗	N/A
Mode-insertion [22]	✓	✓	✗	N/A
Abstract refinement[33]	✓	✓	✗	N/A
Reachability analysis[14]	✓	✓	✗	N/A
Convex relaxation[41]	✓	✓	✗	N/A
RRTs [7, 8, 30]	✓	✗	✓	N/A
Maximally-satisfying Controllers [35]	✗	✗	✓	N/A
MCRS [18]	✓	✓	✓	✗
CDH	✓	✓	✓	✓

ACKNOWLEDGMENTS

The authors want to thank the anonymous reviewers for their valuable advices on improving this paper. This work was supported in part by the Leading-Edge Technology Program of Jiangsu Natural Science Foundation under Grant BK20202001, and in part by the National Natural Science Foundation of China under Grant 62032010.

REFERENCES

[1] Usman Ali and Magnus Egerstedt. 2018. Hybrid optimal control under mode switching constraints with applications to pesticide scheduling. *ACM Transactions*

on Cyber-Physical Systems 2, 1 (2018), 2.
[2] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. 1992. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*. Springer, 209–229.
[3] Henrik Axelsson, Y Wardi, and Magnus Egerstedt. 2008. Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. *Journal of Optimization Theory and Applications* (2008), 167–186.
[4] Sorin C Bengea and Raymond A DeCarlo. 2005. Optimal control of switching systems. *automatica* 41, 1 (2005), 11–27.
[5] Michael S Branicky. 2005. Introduction to hybrid systems. In *Handbook of networked and embedded control systems*. Springer, 91–116.
[6] Michael S Branicky, Vivek S Borkar, and Sanjoy K Mitter. 1998. A unified framework for hybrid control: Model and optimal control theory. *IEEE transactions on automatic control* 43, 1 (1998), 31–45.
[7] Michael S Branicky and Michael M Curtiss. 2002. Nonlinear and hybrid control via RRTs. In *Proc. Intl. Symp. on Mathematical Theory of Networks and Systems*, Vol. 750. Citeseer.
[8] Michael S Branicky, Michael M Curtiss, Joshua A Levine, and Stuart B Morgan. 2003. RRTs for nonlinear, discrete, and hybrid planning and control. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, Vol. 1. IEEE, 657–663.
[9] Michael S Branicky and Sanjoy K Mitter. 1995. Algorithms for optimal hybrid control. In *Proceedings of CDC 1995*. IEEE, 2661–2666.
[10] Eric Brochu, Vlad M Cora, and Nando De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv* (2010).
[11] Timothy M Caldwell and Todd D Murphey. 2010. An adjoint method for second-order switching time optimization. In *Proceedings of CDC 2010*. IEEE, 2155–2162.
[12] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*. Springer, 258–263.
[13] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. 2005. A Tutorial on the Cross-Entropy Method. *Annals OR* 134, 1 (2005), 19–67. https://doi.org/10.1007/s10479-005-5724-z
[14] Jerry Ding and Claire J Tomlin. 2010. Robust reach-avoid controller synthesis for switched nonlinear systems. In *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 6481–6486.
[15] X-C Ding, Yorai Wardi, and Magnus Egerstedt. 2009. On-line optimization of switched-mode dynamical systems. *IEEE Trans. Automat. Control* 54, 9 (2009), 2266–2271.
[16] Magnus Egerstedt, Yorai Wardi, and Henrik Axelsson. 2006. Transition-time optimization for switched-mode dynamical systems. *IEEE Trans. Automat. Control* 51, 1 (2006), 110–115.
[17] Magnus Egerstedt, Yorai Wardi, and Florent Delmotte. 2003. Optimal control of switching times in switched dynamical systems. In *Proceedings of CDC 2003*, Vol. 3. IEEE, 2138–2143.
[18] Samira S Farahani, Vasumathi Raman, and Richard M Murray. 2015. Robust model predictive control for signal temporal logic synthesis. *IFAC-PapersOnLine* 48, 27 (2015), 323–328.
[19] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*. Springer, 379–395.
[20] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. 2009. Hybrid dynamical systems. *IEEE control systems magazine* 29, 2 (2009), 28–93.
[21] David E. Goldberg. 1989. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley.
[22] Humberto Gonzalez, Ram Vasudevan, and Maryam Kamgarpour. 2010. A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems. In *Proceedings of HSCC 2010*. ACM, 51–60.
[23] Humberto Gonzalez, Ram Vasudevan, Maryam Kamgarpour, S Shankar Sastry, Ruzena Bajcsy, and Claire Tomlin. 2010. A numerical method for the optimal control of switched systems. In *Proceedings of CDC 2010*. IEEE, 7519–7526.
[24] Sven Hedlund and Anders Rantzer. 2002. Convex dynamic programming for hybrid systems. *IEEE Trans. Automat. Control* (2002), 1536–1540.
[25] Thomas A Henzinger. 2000. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*. 265–292.
[26] Thomas A Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. 1997. HyTech: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer* 1, 1-2 (1997), 110–122.
[27] Elliot R Johnson and Todd D Murphey. 2009. Second order switching time optimization for time-varying nonlinear systems. In *Proceedings of CDC 2009*. IEEE, 5281–5286.
[28] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. 1998. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation* 14, 1 (1998), 166–171.
[29] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12, 4 (1996), 566–580.
[30] Steven M LaValle and James J Kuffner. 2001. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions* 5 (2001), 293–308.
[31] Jinjin Liu, Kanjian Zhang, Changyin Sun, and Haikun Wei. 2013. Second order transition-time optimization for switched dynamical systems. In *Proceedings of the 32nd Chinese Control Conference*. IEEE, 2382–2386.
[32] Frank Neumann and Ingo Wegener. 2007. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science* 378, 1 (2007), 32–40.
[33] Petter Nilsson and Necmiye Ozay. 2014. Incremental synthesis of switching protocols via abstraction refinement. In *53rd IEEE Conference on Decision and Control*. IEEE, 6246–6253.
[34] Héctor J Sussmann. 2000. Set-valued differentials and the hybrid maximum principle. In *Proceedings of CDC 2000*. IEEE, 558–563.
[35] Cristian-Ioan Vasile, Vasumathi Raman, and Sertac Karaman. 2017. Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3840–3847.
[36] Jiawan Wang, Lei Bu, Shaopeng Xing, and Xuandong Li. 2021. Path-Oriented, Derivative-Free Approach for Safety Falsification of Nonlinear and Nondeterministic CPS. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2021).
[37] Xuping Xu and Panos J Antsaklis. 2000. Optimal control of switched systems: new results and open problems. In *Proceedings of ACC 2000*, Vol. 4. IEEE, 2683–2687.
[38] Xuping Xu and Panos J Antsaklis. 2003. Results and perspectives on computational methods for optimal control of switched systems. In *Proceedings of HSCC 2003*. Springer, 540–555.
[39] Yang Yu and Hong Qian. 2014. The sampling-and-learning framework: A statistical view of evolutionary algorithms. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 149–158.
[40] Yang Yu and Hong Qian. 2016. Derivative-free optimization via classification. In *Proceedings of AAAI 2016*.
[41] Pengcheng Zhao, Shankar Mohan, and Ram Vasudevan. 2017. Optimal control for nonlinear hybrid systems via convex relaxations. *arXiv preprint arXiv:1702.04310* (2017).