DiffuSolve: Diffusion-based Solver for Non-convex Trajectory Optimization

Anjian Li ANJIANL@PRINCETON.EDU
Zihan Ding ZIHAND@PRINCETON.EDU
Adji Bousso Dieng ADJI@PRINCETON.EDU
Ryne Beeson RYNE@PRINCETON.EDU
Princeton University, NJ, USA

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

Optimal trajectory design is computationally expensive for nonlinear and high-dimensional dynamical systems. The challenge arises from solving a non-convex optimization problem with multiple local optima, where traditional numerical solvers struggle to find diverse solutions efficiently without appropriate initial guesses. In this paper, we introduce `DiffuSolve`, a general diffusion model-based solver for non-convex trajectory optimization. An expressive diffusion model is trained on pre-collected locally optimal solutions and efficiently samples initial guesses, which then warm-starts numerical solvers to fine-tune the feasibility and optimality. We also present `DiffuSolve+`, a novel constrained diffusion model with an additional loss in training that further reduces the problem constraint violations of diffusion samples. Experimental evaluations on three tasks verify the improved robustness, diversity, and a $2\times$ to $11\times$ increase in computational efficiency with our proposed method, which generalizes well to trajectory optimization problems of varying challenges.

Keywords: Diffusion Models, Trajectory Optimization, Non-convex Optimizations

1. Introduction

Optimal trajectory design is fundamental in decision-making for autonomous agents. In the open-loop case, the goal is to plan an optimal path and control sequence for an agent to reach a target from an initial state while satisfying the dynamics and safety constraints. These problems often have non-convex structures with nonlinear system dynamics and environmental constraints. For example, within a chaotic dynamical system like the Circular Restricted Three-Body Problem (CR3BP) (Koon et al., 2000), the trajectory optimization problem for the spacecraft has multiple local optima of different qualitative behaviors with various tradeoffs (Hartmann et al., 1998; Russell, 2007). Identifying diverse and locally optimal solutions efficiently remains a significant challenge.

With a control transcription, trajectory optimization can be transformed into a discretized nonlinear program (NLP), where an initial guess is required to get the solution (Betts, 1998). When solving for several solutions, a two-step global search is needed: (i) sampling a distribution on the control space, and (ii) using this sample as an initial guess to the NLP solver that hopefully generates a solution. For each solution, optimality (and also feasibility) is often verified by the solver by checking whether the first-order necessary conditions, the Karush-Kuhn-Tucker (KKT) conditions, are satisfied (Kuhn and Tucker, 2013; Karush, 1939). However, this two-step global search process is often time-consuming for highly non-convex problems and hardly meets the efficiency demands.

Machine learning has also gained popularity in trajectory optimization, but primarily focuses on solving convexified problems such as quadratic Programs (QP) (Chen et al., 2022; Cauligi et al., 2021) rather than learning the solution distribution to the original non-convex problem. Recently, generative

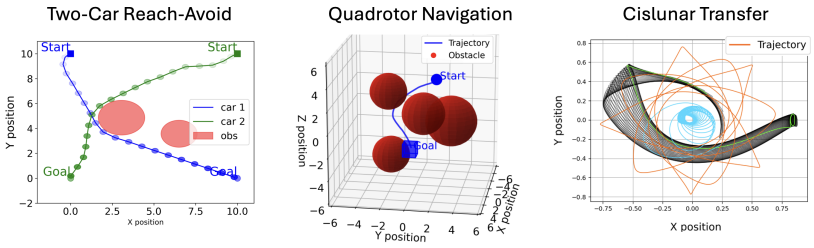

Figure 1: Three trajectory optimization tasks in our experiments (from left to right): *two-car reach-avoid* problem; *quadrotor navigation* problem; *cislunar transfer* problem.

models, particularly diffusion models, have shown good performance in generating trajectories or controls from a distribution (Janner et al., 2022; Ajay et al., 2022; Chi et al., 2023). However, the diffusion model alone without the NLP solver may output trajectories that violate the dynamic and safety constraints, causing catastrophic damage to safety-critical systems. It also lacks the ability to verify the optimality of the sampled trajectories. While efforts have been made to integrate safety constraints into generative models (Yang et al., 2023; Xiao et al., 2023; Botteghi et al., 2023), it is still challenging to achieve solution quality, diversity, feasibility, and computational efficiency at the same time.

In this paper, we propose `DiffuSolve`: a diffusion model-based solver for non-convex trajectory optimization with improved efficiency, diversity, and robustness. Our contribution is two-fold: First, we train a conditional diffusion model to efficiently sample a set of diverse and high-quality trajectory predictions from complex solution distributions. These samples are then used as warm starts for an NLP solver to derive locally optimal and feasible solutions, ensuring the robustness. Second, to further reduce the constraint violation of diffusion samples, we developed `DiffuSolve+` with a novel constrained diffusion model that incorporates a hybrid loss function in training.

`DiffuSolve` has several appealing properties: **(a). Computational Efficiency.** The diffusion model samples high-quality solutions that are close to local optima as warm starts, from which the NLP solver can quickly converge. **(b). Robustness and Diversity.** A constrained diffusion model produces diverse samples with minimal constraint violations, further refined by an NLP solver for enhanced feasibility and optimality. **(c). Automatic Data Generation.** With an NLP solver, we can generate diverse problems and collect corresponding solutions ourselves, allowing for on-demand data augmentation whenever required. **(d). Generalization.** Our framework is suitable for trajectory optimization across various system dynamics and environment settings, with the potential for optimization problems in other areas like finance (Sambharya et al., 2023), etc.

We demonstrate the efficacy of our method on three non-convex trajectory optimization problems: *two-car reach-avoid*, *quadrotor navigation*, and *cislunar low-thrust transfer* for a spacecraft.

2. Related Work

Machine Learning for Optimization. Machine learning methods have been widely used to solve a variety of optimization problems with different techniques, including convex Quadratic Programming (QP) (Zeilinger et al., 2011; Zhang et al., 2019; Chen et al., 2022; Sambharya et al., 2023), mixed-integer programming (Cauligi et al., 2021), combinatorial optimization (Sun and Yang, 2023; Huang et al., 2023) and black-box optimization (Kumar and Levine, 2020; Trabucco et al., 2022; Krishnamoorthy et al., 2023), etc. To improve the constraint satisfaction in optimization problems, neural networks have been incorporated with fully differentiable constraint function (Donti

et al., 2021). Conditional variational autoencoder (CVAE) and long short-term memory (LSTM) are proposed to warm-start the general nonlinear trajectory optimization problems (Li et al., 2023).

Diffusion Models for Trajectory Generation. The diffusion models (Sohl-Dickstein et al., 2015; Song et al., 2020b; Ho et al., 2020) are able to sample multi-modal distributions through a forward diffusion and a reverse sampling process. Both classifier-guided diffusion (Dhariwal and Nichol, 2021) and classifier-free guidance (Ho and Salimans, 2022) are proposed to achieve conditional generation with diffusion models. In robotics, diffusion models are used for generating controls or trajectories (Ajay et al., 2022; Janner et al., 2022; Liang et al., 2023; Sridhar et al., 2023; Chi et al., 2023; Mishra et al., 2023; Ding et al., 2024). Many efforts have been made to improve the diffusion model for creating safe trajectories for safety-critical systems, such as control barrier functions (Xiao et al., 2023; Botteghi et al., 2023) and collision-avoidance kernels (Chang et al., 2023). To adapt to new constraints in testing, prior work investigated composing diffusion models to improve the generalizability (Power et al., 2023; Yang et al., 2023).

Different from existing work, our paper focuses on highly non-convex trajectory optimization problems with many local optima and aims to find diverse solutions efficiently. Instead of incorporating constraints in the sampling process, our constrained diffusion model in `DiffuSolve+` includes a novel loss in training, which can be combined with the current work to further reduce the violation. Moreover, `DiffuSolve` and `DiffuSolve+` are equipped with an NLP solver that finetunes the diffusion samples with verified feasibility and optimality.

3. Preliminaries

Non-Convex Trajectory Optimization. We consider a general open-loop trajectory optimization problem with known dynamical models and environment. With the direct transcription (Betts, 1998), this trajectory optimization problem can be discretized as the following nonlinear program $\mathcal{P}_y$:

$$\mathcal{P}_y := \begin{cases} \min\limits_{x} & J(x;y) \\ s.t., & g_i(x;y) \leqslant 0,\ i = 1,2,...,l \\ & h_j(x;y) = 0,\ j = 1,2,...,m \end{cases} \tag{1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the decision variable, and $y \in \mathcal{Y} \subset \mathbb{R}^k$ represents various problem parameters. $J \in C^1(\mathbb{R}^n, \mathbb{R}^k; \mathbb{R})$ is the objective function. Each $h_i \in C^1(\mathbb{R}^n, \mathbb{R}^k; \mathbb{R})$ and $g_j \in C^1(\mathbb{R}^n, \mathbb{R}^k; \mathbb{R})$ are equality and inequality constraint functions, respectively. In trajectory optimization, x might include a sequence of control u, the time variable t required to reach the target, etc. Parameters y can include control limits, target and obstacle setup, etc. The objective J can be time-to-reach, fuel expenditure, etc. The constraints g and h could be for dynamical models, goal-reaching, obstacle avoidance, etc.

For a fixed problem $\mathcal{P}_y$ with various local optimum (feasible) x^*, the user first provides multiple initial guesses x^0 for the decision variables x, typically drawn from a uniform distribution or some heuristic approach. Then a solver π performs a gradient-based optimization until a local optimum x^* is found and verified by checking KKT conditions. Popular methods include Interior Point Method (IPM) (Wright, 1997) and Sequential Quadratic Programming (SQP) (Boggs and Tolle, 1995).

Diffusion Probabilistic Model. As expressive generative models, the diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020b) contain a forward diffusion process and a reverse generation process. The forward diffusion process iteratively adds Gaussian

noise to sample x_k at the k-th step as $q(x_{k+1}|x_k) = \mathcal{N}(x_{k+1}; \sqrt{1-\beta_k}x_k, \beta_k\mathbf{I}), 0 \leqslant k \leqslant K-1$ where β_k is a given variance schedule. With the diffusion steps $K \to \infty$, x_K becomes a random Gaussian noise. From above one can write a closed form x_k in each sampling step k:

$$x_k = \sqrt{\bar{\alpha}_k}x_0 + \sqrt{1-\bar{\alpha}_k}\varepsilon, \tag{2}$$

where $\bar{\alpha}_k = \prod_{k'=1}^{K}(1-\beta_{k'})$ and $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$ is a random sample from a standard Gaussian.

The data generation process is the reverse denoising process that transforms random noise into data sample $p_\theta(x_{k-1}|x_k) = \mathcal{N}\big(x_{k-1}; \mu_\theta(x_k), \sigma_k\mathbf{I}\big), 1 \leqslant k \leqslant K$, with initial noisy data sampled from a random Gaussian distribution $x_K \sim \mathcal{N}(0, \mathbf{I})$. p_θ is the parameterized sampling distribution, and it can be optimized through an alternative $\varepsilon_\theta(x_k, k)$ which is to predict ε injected for x_k, for every diffusion step k.

For conditional generation, a conditional variable y is added to both processes $q(x_{k+1}|x_k, y)$ and $p_\theta(x_{k-1}|x_k, y)$. The classifier-free guidance (Ho and Salimans, 2022) can be applied to further promote the conditional information, which learns both conditioned and unconditioned noise predictors as $\varepsilon_\theta(x_k, k, y)$ and $\varepsilon_\theta(x_k, k, \varnothing)$. Specifically, ε_θ is optimized with the following loss function:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{(x_0,y)\sim\mathcal{X}\times\mathcal{Y},k,\varepsilon,b}\left\|\varepsilon_\theta\Big(x_k(x_0,\varepsilon), k, (1-b)\cdot y + b\cdot\varnothing\Big) - \varepsilon\right\|_2^2 \tag{3}$$

where x_0 and y are sampled from groundtruth data, $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$, $k \sim \text{Uniform}(1, K)$, $b \sim \text{Bernoulli}(p_{\text{uncond}})$ with given unconditioned probability p_{uncond}, and x_k follows Eq. (2).

4. Methodology

We first propose `DiffuSolve`, a general framework for efficient and robust non-convex trajectory optimization using diffusion models and NLP solvers. Then we introduce `DiffuSolve+` with a novel constrained diffusion model that is designed to generate more feasible solutions.

4.1. DiffuSolve Framework

For similar optimization problems $\mathcal{P}_y$ with the same objective J and constraints g, h but different parameters y, it has been observed that the solutions x^* often exhibit similar structures (Amos et al., 2023; Li et al., 2023). Thus, the key insight of `DiffuSolve` is to use a diffusion model to learn $p(x^*|y)$ based on pre-solved similar problems, so that it can generalize and predict diverse solutions $\tilde{x}^*_{\text{new}}$ to new scenarios where the condition y_{new} value is unseen. However, diffusion models inevitably make prediction errors, which can result in constraint violations. To address this, we use diffusion samples $\tilde{x}^*_{\text{new}}$ as initial guesses to warm-start the NLP solver, which fine-tunes and derives the final solutions x^*_{new}. This approach acts like a safety filter, improving both feasibility and optimality.

Our `DiffuSolve` framework is illustrated in Fig. 2. In the offline data generation process, we use an NLP solver π to collect locally optimal and feasible solutions x^* for various problem instances $\mathcal{P}_y$ with uniformly sampled initial guesses x^0. The solutions with sub-optimal objective values are filtered out manually and the remaining dataset is used as the training data. For `DiffuSolve` training, we use a conditional diffusion model to learn $p(x^*|y)$ with classifier-free guidance (Ho and Salimans, 2022), with the loss function in Eq. (3). In the online testing process when a new problem $\mathcal{P}_{y_{\text{new}}}$ is presented with *a priori* unknown value y_{new}, the diffusion model first predicts diverse solution candidates $\tilde{x}^*_{\text{new}}$ as initial guesses. Then the NLP solver π only requires minor adjustments to derive the final solutions x^*_{new} with improved feasibility and optimality. This warm-starting approach is fully parallelizable for both diffusion sampling and the solving process of π.

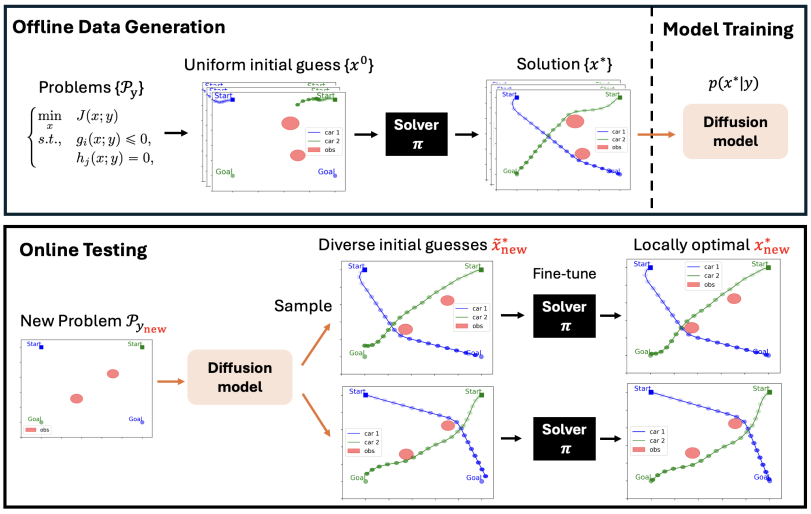

Figure 2: The DiffuSolve framework for solving non-convex trajectory optimization problems.

4.2. DiffuSolve+ with Constrained Diffusion Model

The original diffusion models in DiffuSolve is optimized through the loss function as Eq. (3) without knowing any constraint information. As a result, the sampled solutions could be close to the groundtruth but still largely violate the constraints. To address this issue, we propose an improved DiffuSolve+ method with a constrained diffusion model (CDM) to minimize the constraint violations in the sampled trajectories. Inspired by the equation loss in PINN (Raissi et al., 2019), we define a violation function $V(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ for differentiable constraints in Eq. (1):

$$V = \sum_{i=1}^{l} \max(g_i, 0) + \sum_{j=1}^{m} |h_j|, \tag{4}$$

where $V(\cdot, \cdot) \in PC^1(\mathbb{R}^n, \mathbb{R}^k; \mathbb{R})$ (piecewise differentiable except at $g_i = 0$ or $h_j = 0$) maps from a sample x and parameter y to the total constraint violation as a scalar value. In fact, each constraint term can be further customized with different weights that allow different treatments. The violation value is expected to decrease during the training process of diffusion models. Therefore, we introduce a newly proposed loss function and the corresponding training process as follows.

DiffuSolve+ Training. In the forward diffusion process, by reformulating the sampling process in Eq. (2), we have $x_0 = \frac{x_k - \sqrt{1-\bar{\alpha}_k}\varepsilon}{\sqrt{\bar{\alpha}_k}}, \varepsilon \sim \mathcal{N}(0, \mathbf{I})$. The noise ε is approximated as the neural-network parameterized model $\varepsilon_\theta(x_k, k, y)$ for condition y at timestep k. Replacing it into the previous formula, the clean sample at timestep $k = 0$ is predicted as $\hat{x}_0 = \frac{x_k(x_0,\varepsilon) - \sqrt{1-\bar{\alpha}_k}\varepsilon_\theta(x_k,k,y)}{\sqrt{\bar{\alpha}_k}}$. The benefit of direct $\hat{x}_0$ prediction is that it does not need iterative diffusion sampling thus avoids gradient backpropagation through multi-step sampling. For the CDM training in DiffuSolve+, we introduce a constraint violation loss $\mathcal{L}_{\text{vio}}$ on predicted $\hat{x}_0$ from x_k, $\forall k \in [K]$:

$$\mathcal{L}_{\text{vio}} = \mathbb{E}_{(x_0,y)\sim\mathcal{X}\times\mathcal{Y}, k\in[K], \varepsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\frac{1}{k} V\left(\text{clip}\left(\hat{x}_0(x_k, \varepsilon_\theta), x_0^{\min}, x_0^{\max}\right), y\right)\right] \tag{5}$$

where the clip function preserves the range of $\hat{x}_0$ to be within the lower $x_0^{\min}$ and upper bound $x_0^{\max}$ for $x_0 \in \mathcal{X}$. This is because in the early training phase, the parameterized ε_θ as an approximation

of ε may not be accurate such that some predicted $\widehat{x}_0$ fall out of the original range of x_0, result in invalid calculation of violation function $V(\cdot,\cdot)$. The scheduling factor $\frac{1}{k}$ tends to penalize less when k is large and the estimation of violation function is less reliable. This may not be a theoretically principled choice but is found effective and convenient in our experiments.

For the violation loss $\mathcal{L}_{\text{vio}}$, we always let the condition variable y appear in the noise prediction $\varepsilon_\theta(x_k, k, y)$ without masking it out, contrary to the probabilistic masking in the original training loss $\mathcal{L}_{\text{diff}}$. An intuitive interpretation of the additional loss term is that it guides the diffusion model training with the gradients of minimizing the violation value for each denoising step.

Finally, our proposed CDM training loss is a combination of the original diffusion training loss $\mathcal{L}_{\text{diff}}$ in `DiffuSolve` in Eq. (3) and the violation loss $\mathcal{L}_{\text{vio}}$ in Eq. (5). $\mathcal{L}_{\text{diff}}$ encourages $\widehat{x}_0$ to be close to true x_0, and $\mathcal{L}_{\text{vio}}$ further enforces the constraint satisfaction of the generated samples. λ is a hyperparameter that helps scale the $\mathcal{L}_{\text{vio}}$ to have the scale as $\mathcal{L}_{\text{diff}}$:

$$\mathcal{L} = \mathcal{L}_{\text{diff}} + \lambda \mathcal{L}_{\text{vio}} \tag{6}$$

Conditional Diffusion Sampling. For sampling from trained CDM models, we adopt guided sampling with classifier-free guidance (Ho et al., 2020; Ho and Salimans, 2022) as introduced in Sec. 3. The guided prediction noise with guidance weight c is:

$$\widehat{\varepsilon}_\theta \leftarrow c \cdot \varepsilon_\theta(x_k, k, y) + (1-c) \cdot \varepsilon_\theta(x_k, k, \varnothing) \tag{7}$$

Then we plug $\widehat{\varepsilon}_\theta$ into ε_θ in the sampling process , and through iterative sampling x_{k-1} from $x_k, k = K, \ldots, 1$ with initial random noise $x_K \sim \mathcal{N}(0, \mathbf{I})$, it produces the generated samples x_0. The x_0 serves as the predicted initial guesses $\tilde{x}^*_{\text{new}}$ in the framework introduced in Sec. 4.1.

5. Experiment

In this section, we first evaluate `DiffuSolve` with a vanilla diffusion model (DM) on three highly non-convex trajectory optimization tasks. Secondly, we present the results of `DiffuSolve+` with our novel constrained diffusion model (CDM). As a standalone method, CDM reduces problem constraint violation of the samples compared to DM. Within `DiffuSolve+`, CDM helps further improve the efficiency of obtaining locally optimal and feasible solutions compared to `DiffuSolve`.

5.1. Task setup

We test on three tasks shown in Figure 1 that span diverse domains and present unique challenges: (i). *Two-Car Reach-Avoid*: Multi-agent game-theoretical problem is difficult to scale and features multiple equilibria. (ii). *Quadrotor Navigation*: High-dimensional nonlinear dynamical systems lead to computational inefficiency. (iii). *Cislunar Transfer*: Highly nonlinear chaotic dynamical system in the cislunar space makes the optimization problem slow to solve with numerous local optima. In the test time, we evaluate our method on **new problem $\mathcal{P}_{y_{\text{new}}}$ with unseen y_{new} in training data**.

Two-Car Reach-Avoid. This is a two-player game inspired by (Chen et al., 2018). Each car minimizes the time to reach the goal while avoiding collision with another car and obstacles, shown in Figure 1. The parameter y includes the obstacle positions and sizes that are randomly sampled. Both car_1 and car_2 follows the same dynamics as follows: $\dot{p}_x = v\cos\theta, \dot{p}_y = v\sin\theta, \dot{v} = a, \dot{\theta} = \omega$, where (p_x, p_y, v, θ) denotes the state of each car. p_x, p_y are the position, v is the speed, and θ is the orientation of the car. Each car has two-dimensional controls $u^1 = (a^1, \omega^1), u^2 = (a^2, \omega^2)$, where $a \in [-1, 1]$ is the acceleration and $\omega \in [-1, 1]$ is the angular speed.

Quadrotor Navigation. This a navigation problem of a 10D quadrotor in Figure 1, inspired by (Herbert et al., 2017). The quadrotor finds the minimum time to reach the goal position while avoiding obstacles shown in Figure 1. The problem parameter y includes goal and obstacle positions and sizes that are randomly sampled. The quadrotor's dynamics is defined as follows: $\dot{x} = v_x, \dot{v}_x = g\tan\theta_x, \dot{\theta}_x = -d_1\theta_x + \omega_x, \dot{\omega}_x = -d_0\theta_x + n_0 a_x, \dot{z} = v_z, \dot{y} = v_y, \dot{v}_y = g\tan\theta_y, \dot{\theta}_y = -d_1\theta_y + \omega_y, \dot{\omega}_y = -d_0\theta_y + n_0 a_y, \dot{v}_z = K_T a_z - g$, where x, y, z are the position of the quadrotor. θ_x, θ_y are the pitch and roll angle, and ω_x, ω_y are the corresponding rates. v_x, v_y, v_z are the speed of the quadrotor. d_0, d_1, K_T, n_0 are system parameters, and $g = 9.81$. The controls are $u = (a_x, a_y, a_z)$.

Cislunar Transfer. In this task, we consider a minimum-fuel *cislunar low-thrust transfer* with the dynamical model to be a Circular Restricted Three-Body Problem (CR3BP) (Koon et al., 2000). As shown in Figure 1, the example trajectory (orange) starts from the end of a geostationary transfer spiral (blue) and ends at an arc of the invariant manifold (black). The problem parameter y is the maximum allowable thrust of the spacecraft. Assuming the mass of the spacecraft is negligible, the CR3BP describes the equation of motion of a spacecraft under the gravitational force from the Earth and moon. Let m_1 be the mass of the Earth and m_2 be the mass of the moon, and $\mu = m_2/(m_1+m_2)$, the dynamics of the spacecraft in the CR3BP is as follows: $\ddot{x} - 2\dot{y} = -\bar{U}_x, \ddot{y} + 2\dot{x} = -\bar{U}_y, \ddot{z} = -\bar{U}_z$, where $\bar{U}(r_1(x,y,z), r_2(x,y,z)) = -\frac{1}{2}\left((1-\mu)r_1^2 + \mu r_2^2\right) - \frac{1-\mu}{r_1} - \frac{\mu}{r_2}$ is the effective gravitational potential, $r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2)}$ and $r_2 = \sqrt{(x-(1-\mu))^2 + y^2 + z^2)}$ are the distance from the spacecraft to the sun and moon, respectively. This is a chaotic dynamical system.

5.2. Experiment Setup

Numerical Solvers. For all three tasks, we formulate and solve the optimization problem with Sparse Nonlinear OPTimizer (SNOPT) (Gill et al., 2005) as the solver π, which uses the SQP method and supports warm-starting. For the *cislunar transfer*, we adopt *pydylan*, a Python interface of the Dynamically Leveraged Automated (N) Multibody Trajectory Optimization (DyLAN) (Beeson et al., 2022) to formulate the CR3BP dynamics in addition to SNOPT.

Data collections and model training. For *two-car reach-avoid*, *quadrotor navigation* and *cisluar transfer* task, we each collect 114k, 179k, and 300k locally optimal and feasible solutions as training data, each taking 8 hours, 20 hours, and 3 days on around 200 cpu cores in parallel. For both DM and CDM, we apply a UNet structure with three hidden layers of 512, 512, and 1024 neurons, with a fully connected layer of 256 and 512 neurons to embed the conditional input y. The sampling step is set to be 500. Both DM and CDM are trained with 200 epochs using the Adam optimizer (Kingma and Ba, 2014), which takes 9 hours and 14 hours respectively on one NVIDIA A100 GPU.

Baselines. **(1) Uniform**: This method solely uses an NLP solver with uniformly sampled initial guesses $x \in \mathcal{X}$, which solves problems from scratch without leveraging any problem-specific prior information. **(2) Optimal Uniform**: This method uniformly samples from collected locally optimal solutions $\{x^*\}$ as initial guesses for the NLP solver. **(3) CVAE_LSTM**: This method (Li et al., 2023) first uses a conditional variational auto-encoder (CVAE) to sample non-control variables, e.g. time variables, and then an long short-term memory (LSTM) is adopted to sample the control sequence.

Although there exist graph-based methods for path planning, such as the Rapidly-exploring Random Trees (RRT) (LaValle, 1998), A* (Hart et al., 1968), etc., they do not handle well with the general case of nonlinear objective and dynamical systems in complex environment. Thus, the comparison against them is not provided in our experiments.

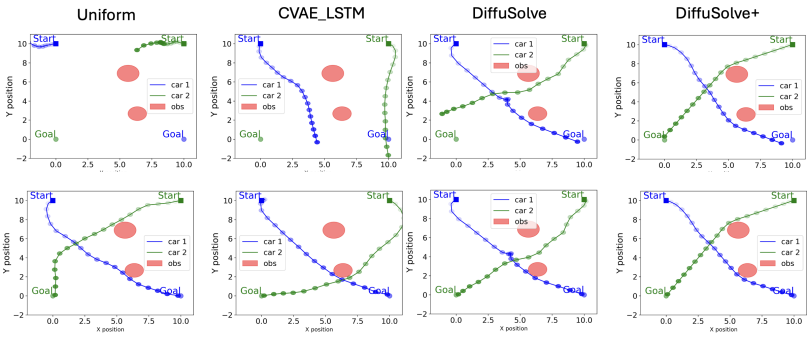

Figure 3: In *two-car reach-avoid* problem: First row: raw samples (no solver); Second row: corresponding solver derived locally optimal and feasible solutions.

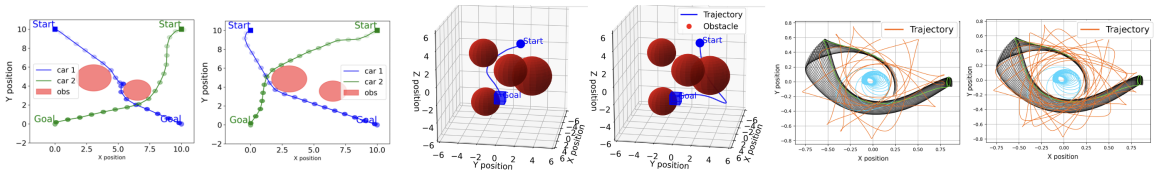

Figure 4: Diverse trajectory solutions for *two-car reach-avoid*, *quadrotor navigation* and *cislunar transfer* problem with `DiffuSolve` method, given the same conditional input.

Diversity Measure. We adopt the Vendi Score (Friedman and Dieng, 2022) as a quantitative metric for measuring the diversity of the model samples, with the similarity kernel $k(x, y) = e^{-||x-y||}$.

5.3. DiffuSolve Results

In this section, we evaluate `DiffuSolve`'s performance on solution optimality, feasibility, diversity, and computational efficiency with a vanilla diffusion model (DM) against baseline methods.

5.3.1. Optimality and Feasibility

We generate 600 samples from each method as warm starts for the NLP solver. In Table 1, we present the percentage (ratio) of locally optimal and feasible solutions over 600 samples which are fine-tuned by the NLP solver. Note that before finetuning, no method can directly generate locally optimal and feasible solutions. When using the samples as warm starts for the NLP solver, our proposed `DiffuSolve` obtains a greater number of locally optimal and feasible solutions than any non-diffusion methods.

Method	Two-car	Quadrotor	Cislunar
DiffuSolve+	**94.83**	**100.00**	–
DiffuSolve	**94.50**	**99.83**	**54.00**
CVAE_LSTM	63.17	99.67	29.17
Uniform	64.17	99.17	17.50
Optimal Uniform	83.00	98.50	23.33

Table 1: Percentage (%) of locally optimal and feasible solutions over 600 samples.

We also visualize the sampled trajectories and the corresponding solver-derived trajectories in Figure 3. For the Uniform and CVAE_LSTM method, the generated trajectories are far from feasible, thus cannot serve as good initial guesses for the NLP solver. `DiffuSolve` already generates near-feasible trajectories and it's easy for NLP solver to derive locally optimal and feasible solutions.

Task	Method	Solving Time			Vendi Score
		Mean($\pm$STD)	25%-Quantile	Median	
Two-car	DiffuSolve+ (Ours)	**17.86** $\pm$ **14.28**	**7.68**	**13.83**	**3873.60**
	DiffuSolve (Ours)	18.82 $\pm$ 14.33	8.77	15.61	3574.66
	CVAE_LSTM	36.24 $\pm$ 20.55	19.45	33.26	1162.72
	Uniform	46.17 $\pm$ 20.63	29.62	43.54	**5256.75**
	Optimal Uniform	25.15 $\pm$ 19.75	10.82	20.89	3708.58
Quadrotor	DiffuSolve+ (Ours)	**6.65** $\pm$ **3.55**	**4.47**	**6.05**	5458.23
	DiffuSolve (Ours)	7.30 $\pm$ 4.79	4.78	6.62	5428.63
	CVAE_LSTM	9.05 $\pm$ 8.30	5.00	6.74	309.18
	Uniform	16.87 $\pm$ 13.60	9.28	12.48	**5985.10**
	Optimal Uniform	13.94 $\pm$ 12.08	7.66	10.13	**5575.83**
Cislunar	DiffuSolve (Ours)	**50.81** $\pm$ **70.30**	**9.97**	**26.68**	**5999.96**
	CVAE_LSTM	141.31 $\pm$ 125.60	38.41	106.48	1403.38
	Uniform	199.34 $\pm$ 116.59	114.02	174.35	**6000.00**
	Optimal Uniform	177.35 $\pm$ 123.22	70.03	153.97	5918.08

Table 2: Computational time (sampling + solving) statistics and Vendi Score.

5.3.2. COMPUTATIONAL EFFICIENCY

For the locally optimal and feasible solutions obtained in Table 1, Table 2 presents the computational time statistics, involving both sampling time from the model and the NLP solver time. In Table 2, the `DiffuSolve` with a vanilla DM is 2 $\times$ to 11 $\times$ faster than the uniform method based on the top 25%-quantile of the computational time for deriving the locally optimal and feasible solutions. Also in the histogram plots in Figure 5, `DiffuSolve` obtains more solutions than baseline methods within a short period of time. These results coincide with Figure 3 that since the `DiffuSolve` generates samples that are close to local optima, it takes a shorter time for the NLP solver to converge.

Thus, it motivates us to set a proper cut-off time for the solver and allow more initial guesses to be fed, which could further improve the efficiency of `DiffuSolve` to derive solutions.

5.3.3. DIVERSITY

In this paper, our proposed `DiffuSolve` aims to learn the solution distribution of the non-convex problem, instead of predicting a single solution. For example, for the *two-car reach-avoid* problem, there are multiple equilibrium points that are locally optimal and feasible. In Figure 4, we present 2 qualitatively different solutions with `DiffuSolve` for each task, showing its ability to generate diverse solutions even within a shorter period of time. The Vendi Score is also evaluated on the samples from each method in Table 2. While the Uniform method certainly generates the most diverse samples, the `DiffuSolve` is comparable to Optimal Uniform and outperforms CVAE_LSTM in sample diversity. Especially for the *quadrotor navigation*, though the solving time for CVAE_LSTM is competitive in Table 2, the Vendi Score shows that its generated solutions are not diverse at all.

5.4. DiffuSolve+ Results

In this section, we present the results of `DiffuSolve+` on the *quadrotor navigation* and *two-car reach-avoid* task. The proposed CDM is evaluated both as a standalone method for reducing constraint violations and within the `DiffuSolve+` framework for improving computational efficiency.

Since the current *cislunar transfer* task originally adopted from (Beeson et al., 2022) uses an adaptive-stepsize integrator for the dynamics constraint, it requires extra effort to incorporate it into the neural network and is beyond our scope. We leave the CDM for this task as future work.

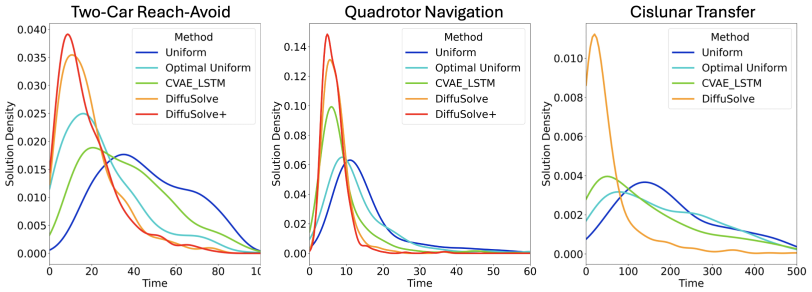

Figure 5: The histogram of computational time (including model sampling time and solver time) for different methods to find locally optimal and feasible solutions.

5.4.1. CONSTRAINT VIOLATION

As a standalone method without the NLP solver, our proposed CDM from Sec. 4.2 has fewer constraint violations compared to the vanilla DM and CVAE_LSTM, displayed in Table 3. Also shown in Figure 3, the CDM generates a trajectory that is the closest to the locally optimal and feasible solutions - only car 1 cannot reach the goal but it is already near. It will be easy for the NLP solver to close this gap within `DiffuSolve+`.

Method	Two-car		Quadrotor	
	Mean(±STD)	Median	Mean(±STD)	Median
DiffuSolve+	**3.00±15.36**	**1.44**	**3.06±7.47**	**0.38**
DiffuSolve	10.99±35.59	4.52	3.85±8.52	0.63
CVAE_LSTM	282.72±186.40	292.90	20.19±22.32	12.96

Table 3: Constraint violation statistics (no solver).

5.4.2. COMPUTATIONAL EFFICIENCY

For `DiffuSolve+` with the proposed CDM, the computational efficiency is further improved as shown in Table 2. The intuition is clear from Figure 3: when the `DiffuSolve+` sample is closer to the local optima, it takes the NLP solver an even shorter time to derive the final solution. We also present the histogram of the computational time in Figure 5. Here we visualize the density of locally optimal and feasible solutions obtained within a time range, where `DiffuSolve+` in red is able to obtain more solutions than `DiffuSolve` within a short period of time from 600 initial guesses.

6. Conclusion and Discussion

This paper presents `DiffuSolve`: a general diffusion model-based solver for non-convex trajectory optimization. It can generate locally optimal, feasible, and diverse solutions with high computational efficiency. We also propose `DiffuSolve+` which includes a novel constrained diffusion model that further improves constraint violation and computational efficiency. It is also a general acceleration framework that has the potential to solve optimization problems in other areas such as finance, etc.

Limitations also exist for the current paper. We don't spend extra effort to optimize the diffusion model's sampling speed. Therefore, the current `DiffusSolve` and CDM may not be able to achieve real-time optimization in highly dynamic environments. This can be improved with existing acceleration techniques such as stride sampling, DDIM (Song et al., 2020a), etc. In addition, although most constraint functions in trajectory optimization are differentiable (almost everywhere), the current CDM does not work for non-differentiable constraints. But transforming them into differentiable functions or applying approximation with appropriate bump functions or neural networks is possible.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Brandon Amos et al. Tutorial on amortized optimization. *Foundations and Trends® in Machine Learning*, 16(5):592–732, 2023.

Ryne Beeson, Amlan Sinha, Bindu Jagannatha, Devin Bunce, and David Carroll. Dynamically leveraged automated multibody (n) trajectory optimization. In *AAS/AIAA Space Flight Mechanics Conference*, Charlotte, NC, 8 2022. American Astronautical Society.

John T Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

Nicolò Botteghi, Federico Califano, Mannes Poel, and Christoph Brune. Trajectory generation, control, and safety with denoising diffusion probabilistic models. *arXiv preprint arXiv:2306.15512*, 2023.

Abhishek Cauligi, Preston Culbertson, Edward Schmerling, Mac Schwager, Bartolomeo Stellato, and Marco Pavone. Coco: Online mixed-integer control via supervised learning. *IEEE Robotics and Automation Letters*, 7(2):1447–1454, 2021.

Junwoo Chang, Hyunwoo Ryu, Jiwoo Kim, Soochul Yoo, Joohwan Seo, Nikhil Prakash, Jongeun Choi, and Roberto Horowitz. Denoising heat-inspired diffusion with insulators for collision free motion planning. *arXiv preprint arXiv:2310.12609*, 2023.

Mo Chen, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Robust sequential trajectory planning under disturbances and adversarial intruder. *IEEE Transactions on Control Systems Technology*, 27(4):1566–1582, 2018.

Steven W Chen, Tianyu Wang, Nikolay Atanasov, Vijay Kumar, and Manfred Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model. *arXiv preprint arXiv:2402.03570*, 2024.

Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *arXiv preprint arXiv:2210.02410*, 2022.

Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

John W Hartmann, Victoria L Coverstone-Carroll, and Steven N Williams. Optimal interplanetary spacecraft trajectories via a pareto genetic algorithm. *The Journal of the Astronautical Sciences*, 46:267–282, 1998.

Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522. IEEE, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Junwei Huang, Zhiqing Sun, and Yiming Yang. Accelerating diffusion-based combinatorial optimization solvers by progressive distillation. *arXiv preprint arXiv:2308.06644*, 2023.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

William Karush. Minima of functions of several variables with inequalities as side constraints. *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Wang Sang Koon, Martin W Lo, Jerrold E Marsden, and Shane D Ross. Dynamical systems, the three-body problem and space mission design. In *Equadiff 99: (In 2 Volumes)*, pages 1167–1181. World Scientific, 2000.

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. *arXiv preprint arXiv:2306.07180*, 2023.

Harold W Kuhn and Albert W Tucker. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer, 2013.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *Advances in Neural Information Processing Systems*, 33:5126–5137, 2020.

Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

Anjian Li, Amlan Sinha, and Ryne Beeson. Amortized global search for efficient preliminary trajectory design with deep generative models. *arXiv preprint arXiv:2308.03960*, 2023.

Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*, 2023.

Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.

Thomas Power, Rana Soltani-Zarrin, Soshi Iba, and Dmitry Berenson. Sampling constrained trajectories using composable diffusion models. In *IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning*, 2023.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ryan P Russell. Primer vector theory applied to global low-thrust trade studies. *Journal of Guidance, Control, and Dynamics*, 30(2):460–472, 2007.

Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. End-to-end learning to warm-start for real-time quadratic optimization. In *Learning for Dynamics and Control Conference*, pages 220–234. PMLR, 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. *arXiv preprint arXiv:2310.07896*, 2023.

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *arXiv preprint arXiv:2302.08224*, 2023.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pages 21658–21676. PMLR, 2022.

Stephen J Wright. *Primal-dual interior-point methods*. SIAM, 1997.

Wei Xiao, Tsun-Hsuan Wang, Chuang Gan, and Daniela Rus. Safediffuser: Safe planning with diffusion probabilistic models. *arXiv preprint arXiv:2306.00148*, 2023.

Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional diffusion-based continuous constraint solvers. *arXiv preprint arXiv:2309.00966*, 2023.

Melanie Nicole Zeilinger, Colin Neil Jones, and Manfred Morari. Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization. *IEEE Transactions on Automatic Control*, 56(7):1524–1534, 2011.

Xiaojing Zhang, Monimoy Bujarbaruah, and Francesco Borrelli. Safe and near-optimal policy learning for model predictive control using primal-dual neural networks. In *2019 American Control Conference (ACC)*, pages 354–359. IEEE, 2019.