Proceedings of Machine Learning Research vol 144:1–13, 2021

Optimal Cost Design for Model Predictive Control

Avik Jain AVIKJ@BERKELEY.EDY
Lawrence Chan CHANLAW@BERKELEY.EDU
Daniel S. Brown DSBROWN@BERKELEY.EDU
Anca D. Dragan ANCA@BERKELEY.EDU
InterACT Lab, University of California, Berkeley, USA

Abstract

Many robotics domains use some form of nonconvex model predictive control (MPC) for planning, which sets a reduced time horizon, performs trajectory optimization, and replans at every step. The actual task typically requires a much longer horizon than is computationally tractable, and is specified via a cost function that cumulates over that full horizon. For instance, an autonomous car may have a cost function that makes a desired trade-off between efficiency, safety, and obeying traffic laws. In this work, we challenge the common assumption that the cost we optimize using MPC should be the same as the ground truth cost for the task (plus a terminal cost). MPC solvers can suffer from short planning horizons, local optima, incorrect dynamics models, and, importantly, fail to account for *future replanning* ability. Thus, we propose that in many tasks it could be beneficial to purposefully choose a different cost function for MPC to optimize: one that results in the MPC *rollout* having low ground truth cost, rather than the MPC *planned trajectory*. We formalize this as an optimal cost design problem, and propose a zeroth-order optimization-based approach that enables us to design optimal costs for an MPC planning robot in continuous MDPs. We test our approach in an autonomous driving domain where we find costs different from the ground truth that implicitly compensate for replanning, short horizon, incorrect dynamics models, and local minima issues. As an example, the learned cost incentivizes MPC to delay its decision until later, implicitly accounting for the fact that it will get more information in the future and be able to make a better decision.

Keywords: Model Predictive Control, Optimal Cost Design, Planning under Uncertainty

1. Introduction

Robotics planning algorithms often sacrifice optimality for tractability. One of the most well known and widely used planning paradigms in robotics is nonconvex model predictive control (MPC) (Lee, 2011; García et al., 1989; Poignet and Gautier, 2000; Künhe et al., 2005; Zhang et al., 2016; Ostafew et al., 2016; Thananjeyan et al., 2020; Sripathy et al., 2021). MPC optimizes trajectories based on a ground-truth cost function, C, that describes the task. Rather than trying to find a globally optimal trajectory with respect to C, MPC uses four heuristics to enable tractable planning: (1) *Short Horizon*: MPC plans for a finite horizon which is typically much shorter than the full horizon of the problem, leading to potentially myopic behavior; this is only partially mediated by learning a terminal cost added to the final state of the trajectory (Zhong et al., 2013; Karnchanachari et al., 2020); (2) *Local Optimization*: MPC performs local trajectory optimization, which while efficient, often leads to only locally, but not globally optimal behavior; (3) *Replanning*: MPC repeatedly replans a new trajectory following each observation and takes the first action from this plan; however, this can lead to suboptimal behavior since the generated plans do not take into account the fact that the robot will replan based on future observations; this is in contrast to contingency planning (Hardy and Campbell, 2013) and full POMDP solvers (Lovejoy, 1991; Shani et al., 2013), which explicitly account for the fact that the robot will gather new information and be able to replan in the future; (4) *Approximate Dynamics Model:* MPC plans into the future using an approximate model of dynamics. Repeatedly replanning allows an agent the chance to recover; however, systematic errors in the dynamics can still lead to compounding errors in the actual executed trajectory.

In this paper, we challenge the implicit assumption that MPC should optimize the true cost C, and instead propose that optimizing a surrogate cost C' can actually improve performance of the resulting MPC rollout with respect to C. Rather than focusing on making the MPC *planned trajectory* have low true cost, we should focus on the MPC *rollout* having low true cost. We observe that optimizing the true cost directly via MPC enables the former, but might not be an optimal choice for the latter.

Implicitly accounting for horizon issues: As an example, consider an autonomous car (orange) driving in the scenario shown in Figure 1(a). Here, the optimal behavior with respect to the robot's true cost is to switch lanes to maintain its speed. From the perspective of an MPC planner with a short planning horizon, the cost of switching lanes outweighs the benefit of maintaining its speed accumulated over the short horizon. A learned surrogate cost function C' incentivizes lane changing, thereby decreasing the true cumulative rollout cost. Prior work seeks to address the issue of short horizon by learning a value function or terminal cost, evaluated at the final state of the MPC trajectory (Zhong et al., 2013; Lowrey et al., 2018; Hoeller et al., 2019; Karnchanachari et al., 2020). However, while prior work assumes you should optimize C with an added terminal cost, we investigate replacing C with a learned surrogate C'. The learned terminal cost will not be perfect (otherwise we would just act greedily with respect to it), so there is still headroom for a surrogate cost function C' to improve performance even when we add a terminal cost. Our experimental results indicate that our proposed approach is complementary to learning a terminal cost: while learning a terminal cost results in better MPC performance by alleviating short planning horizon issues, using a learned terminal cost along with a learned surrogate cost function results in even better performance. More importantly, a short horizon is not the only reason MPC can result in suboptimal behavior, MPC may also suffer from local optima and inaccurate dynamics models, and fails to account for future replanning.

Implicitly accounting for local optimization issues: Consider the scenario in Figure 1(b). Here the optimal behavior with respect to the robot's true cost is to merge into the rightmost lane while avoiding collisions; however, there is a human car in the way. Because MPC only searches locally for a trajectory, it does not discover that a significant reduction in speed will let the human car pass and provide a collision free path to the right lane, and instead chooses to stay in the left lane. A learned surrogate cost function C' smooths the optimization landscape and pushes the car out of the left lane, enabling MPC to more easily discover the globally optimal trajectory.

Implicitly accounting for future replanning: Even more interesting is the scenario shown in Figure 1(c), where the robot car is trying to travel as fast as possible while staying on the road and avoiding collisions; however, there is a human car driving slowly on the lane line in front of the robot, and the robot is uncertain about which lane the human will eventually merge into. Running MPC on the true cost function results in an overly conservative behavior that picks a lane to merge into and slows down to avoid a collision. This is because MPC plans ahead and sees that no matter which lane it picks, there is a chance that the human picks the same lane. By contrast, the learned surrogate cost function C' actually *encourages MPC to defer the decision of which lane to choose until later, when it will have more information about which lane the human driver chooses.* Thus, MPC with the learned surrogate cost function results in an emergent contingency planning behavior.

Implicitly accounting for an approximate dynamics model: Consider planning using a deterministic dynamics model, but where the true dynamics are subject to systematic noise, e.g., wind blowing across the highway. Learning a cost function can implicitly correct for the mismatch between the MPC dynamics model and the true dynamics by shaping the cost such that the car implicitly adjusts its steering to compensate for the wind.

We demonstrate that we can address the above issues via *optimal cost design*[1]: given a ground-truth cost function C and a robot planning algorithm, we find a surrogate cost function C' such that optimizing for C' results in robot trajectories with minimal cost under the true cost function C (Singh et al., 2009). Prior work on optimal cost design has focused on learning cost functions for bounded rational agents in the context of reinforcement learning, and approaches optimal cost design using Bellman backups, tree-based planning, or gradient descent on domains with discrete actions and often discrete state spaces (Singh et al., 2009; Sorg, 2011; Sorg et al., 2010). By contrast, we seek to solve the optimal cost design problems for non-linear dynamical systems with continuous states and actions that are optimized via nonconvex MPC. Other related work has investigated cost optimization for continuous control tasks. Marco et al. (2016) use Bayesian optimization to automatically tune the cost for an LQR controller. Karnchanachari et al. (2020) present a method for automatically shaping a sparse MPC cost function by learning a value function; however, while learning a value function can alleviate short horizon problems, it does not necessarily address issues that arise from local optimization, replanning, and approximate dynamics.

We propose a zeroth-order optimization approach for optimal cost design for MPC. Given a set of initial conditions, we seek to learn a surrogate cost function C' such that optimizing C' results in MPC rollouts that have minimum cost under the true cost function C. We show in four case studies that this optimization is tractable and ameliorates the shortcomings of planning with MPC. Our main contributions are as follows: (1) we highlight the implicit, but incorrect, assumption in MPC that we should always optimize the true cost function; (2) we analyze scenarios in which optimizing an alternate cost function via MPC yields lower cumulative costs than optimizing the true cost function; and (3) we explore the robustness of optimized surrogate cost functions to variations in scenario conditions.

2. Optimal Cost Design for MPC

We model our problem as a finite horizon, Markov decision process (Puterman, 2014) with state $x \in \mathcal{X}$, controls $u \in \mathcal{U}$, transition function $x_{t+1} = f(x_t, u_t)$ where t is the timestep, and cost C_θ parameterized by θ. Given a starting state x_0, the goal is to optimize controls

1. This problem is usually called *optimal reward design* in the RL setting, e.g., Singh et al. (2009).

$\boldsymbol{u} = [u_0, \dots, u_{T-1}]$ over horizon T to minimize the cumulative cost according to C_θ:

$$\boldsymbol{u}^*(\theta) = \underset{\boldsymbol{u}}{\operatorname{argmin}} \sum_{t=0}^{T-1} C_\theta(x_t, u_t) \ . \tag{1}$$

2.1. Model Predictive Control

It is common in robotics to use online planning in the form of nonconvex model predictive control (MPC) (García et al., 1989; Maciejowski, 2002; Lee, 2011). In MPC we usually do not have access to the true dynamics function f and instead plan with respect to an approximate dynamics model $\hat{f}$. Furthermore, the robot may not get to directly observe everything about the state: even though we may model the world as an MDP, the real world is typically better modeled as a POMDP (Smallwood and Sondik, 1973; Lovejoy, 1991). For example, in the autonomous vehicle setting considered in this paper, we often plan using MPC in the presence of other agents. When planning a driving trajectory, the robot needs to take into account the anticipated behavior of the other cars on the road. Thus, the robot needs a model of the other agents' behavior. Following prior work by Sadigh et al. (2017), we assume access to a model of human behavior parameterized by φ. Thus, the true state is a function of both the robot's controls and the controls of the other agent:

$$x_{t+1}^{\varphi} = f^{\varphi}(x_t, u_t) = f(x_t, u_t, u_t^{\varphi})$$

However, in most cases, the robot does not know φ and instead will maintain a belief over the expected behavior of the human. At time step t, MPC seeks to approximate the optimal controls $\boldsymbol{u}^*$ by optimizing the following objective:

$$\boldsymbol{u}_{t:t+K-1}^{\text{MPC}}(\theta) = \underset{\boldsymbol{u}_{t:t+K-1}}{\operatorname{argmin}} \underset{\varphi}{\mathbb{E}} \left[\sum_{i=0}^{K-1} C_\theta(\hat{x}_{t+i}^{\varphi}, u_{t+i}) \right] \ . \tag{2}$$

where the expectation is taken over the robot's belief over φ and $\hat{x}_{t+1}^{\varphi}$ is given by applying the robot's approximate dynamics model:

$$\hat{x}_{t+i+1}^{\varphi} = \hat{f}^{\varphi}(x_t, u_t) = \hat{f}(\hat{x}_{t+i}, u_t, u_t^{\varphi}); \ \hat{x}_t^{\varphi} = x_t^{\varphi},$$

where $\boldsymbol{u}_{t:t+K-1}^{\text{MPC}}(\theta)$ is found via trajectory optimization over the truncated horizon $K < T$.

At time t, the robot executes its first planned action $u_t^{\text{MPC}}(\theta) = \boldsymbol{u}_{t:t+K-1}^{\text{MPC}}(\theta)[0]$, updates its belief over φ based on the observed action of the other driver, and then replans K steps into the future based on its next state $x_{t+1} = f^{\varphi}(x_t, u_t^{\text{MPC}})$. It is typically intractable to minimize Equation (2) directly, so $\boldsymbol{u}^{\text{MPC}}$ is found via local optimization, for example by performing N steps of gradient descent:

$$\boldsymbol{u}_{t:t+K}^{\text{MPC}}(\theta) \leftarrow \boldsymbol{u}_{t:t+K}^{\text{MPC}}(\theta) - \alpha \nabla_{\boldsymbol{u}} \underset{\varphi}{\mathbb{E}} \left[\sum_{i=0}^{K-1} C_\theta(\hat{x}_{t+i}^{\varphi}, u_{t+i}) \right] \ . \tag{3}$$

The actual MPC rollout, consisting of visited states and executed controls, is as follows:

$$\xi_{\text{MPC}} = \left((x_0^{\varphi}, u_0^{\text{MPC}}(\theta)), (x_1^{\varphi}, u_1^{\text{MPC}}(\theta)), \dots, (x_{T-1}^{\varphi}, u_{T-1}^{\text{MPC}}(\theta)) \right) \tag{4}$$

where $u_t^{\text{MPC}}(\theta) := \boldsymbol{u}_{t:t+K-1}^{\text{MPC}}(\theta)[0]$ and $x_{t+1}^{\varphi} = f^{\varphi}(x_t^{\varphi}, u_t^{\text{MPC}}(\theta))$. Note that while MPC has the ability to anticipate future events via its approximate dynamics model $\hat{f}^{\varphi}$, and can take control actions accordingly by planning K timesteps into the future, the plan is only executed for one step and then changes. Despite replanning at each timestep, MPC plans K steps into the future assuming the entire trajectory will be executed.

2.2. Problem Statement

Given a robot that plans trajectories using MPC, a true cost function $C = C_\theta$, and a distribution of starting conditions p_0, we seek a surrogate cost function $C' = C_{\theta'}$, parameterized by θ', which is the solution to the following optimal cost design problem:

$$\theta' = \underset{\theta'}{\operatorname{argmin}} \underset{x_0 \sim p_0, \varphi}{\mathbb{E}} \left[\sum_{t=0}^{T-1} C_\theta(x_t^{\varphi}, u_t^{\text{MPC}}(\theta')) \right] , \tag{5}$$

where $u_t^{\text{MPC}}(\theta')$ denotes the first control planned using MPC by solving Equation (2) under the surrogate cost parameters θ' and $x_{t+1}^{\varphi} = f^{\varphi}(x_t^{\varphi}, u_t^{\text{MPC}}(\theta'))$. Note that, critically, Equation (6) seeks a surrogate cost function parameterized by θ' that performs well when the controls from a limited horizon MPC planner are evaluated over the longer, true horizon T. We call this problem Optimal Cost Design for Model Predictive Control or OCD-MPC.

2.3. Finding a Surrogate Cost Function

In theory, there always exists a surrogate cost function that allows MPC to recover the globally optimal trajectory. For example, optimizing a cost that is defined as the L2 distance between the globally optimal controls and the MPC controls, $C'(x_t, u_t) = ||u_t^* - u_t^{\text{MPC}}||^2$, will lead MPC to find the globally optimal controls, assuming that the local optimization converges. However, while this shows that an optimal surrogate cost function exists, such a cost function is impractical as it requires prior knowledge of the globally optimal controls. A more interesting question is: can we improve the behavior according to C with a more practical parameterization of the cost function space?

In this work, we seek to find a surrogate cost function C' by optimizing Equation (6) directly. This optimization may be performed using a variety of methods such as Bayesian optimization (Snoek et al., 2012; Shahriari et al., 2015) or evolutionary search methods (Kennedy and Eberhart, 1995; Pelikan et al., 2002). We chose to use Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2006). CMA-ES is a stochastic zeroth-order (derivative-free) optimization algorithm that is well suited for non-convex optimization problems over continuous search spaces. CMA-ES also has very few hyperparameters that need manual tuning and enables easy parallelization for computational efficiency.

Given a population of potential surrogate costs, we use MPC to generate rollouts under each proposed surrogate cost and then use the cumulative true cost under C as the fitness. We consider environments that have stochastic initializations in terms of car positions and velocities. Thus, we calculate the fitness of a proposal surrogate cost C' as the average true trajectory cost when using MPC to optimize the surrogate cost proposal over n i.i.d.

samples from the initial state distribution p_0. This results in the following fitness function:

$$\text{fitness}(\tilde{\theta}) = \frac{1}{n}\sum_{j=1}^{n}\sum_{t=0}^{T-1} C_\theta\left(x^{\varphi}_{j,t}, u^{\text{MPC}}_{j,t}(\tilde{\theta}) \mid x^{\varphi}_{j,0} \sim p_0\right) \quad , \tag{6}$$

where $x^{\varphi}_{j,t}, u^{\text{MPC}}_{j,t}$ are the t-th state-action pair from the j-th MPC rollout ξ_j.

In Section 3 we discuss four case studies that demonstrate the ability of OCD-MPC to optimize a surrogate cost function for a single initial state ($n = 1$) sampled from p_0 and used at both training and test time. These experiments allow us to better analyze the qualitative and quantitative performance of OCD-MPC. In Section 4, we explicitly study the generalization capability of OCD-MPC, where we optimize a surrogate cost funtion C' under varying numbers of initial states n and then evaluate the learned cost function on a larger held-out set of test initializations drawn from p_0.

3. Experiments

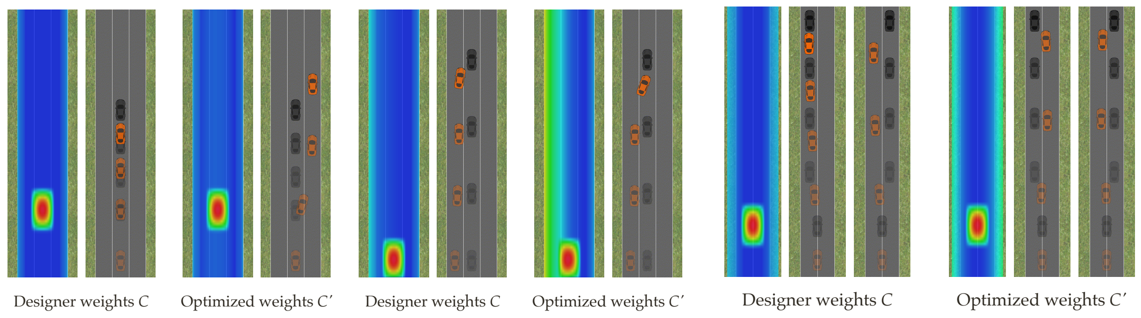
In this section, we empirically test whether OCD-MPC can learn surrogate cost functions that implicitly compensate for issues due to short horizons, local optima, failing to account for future replanning, and planning using an approximate dynamics model.

Driving Simulator and Implementation Details: We use an autonomous driving simulator based on Sadigh et al. (2016, 2017).[2]The robot's true cost function C_θ is modeled as a linear function of the following continuous features: the squared difference between its current forward velocity and a target velocity, a human car collision spike, a smoothed indicator function for grass or road, distance from the center of the closest lane, and the distances from the center of each lane.[3] MPC is run with a receding horizon of $K = 5$. The true horizon is $T = 15$ in Scenarios 1 and 2 and $T = 20$ in Scenario 3. Unless otherwise specified, the planner optimizes controls with control initializations which go straight, steer right, and steer left. MPC uses 100 steps of gradient descent for each initialization, and returns the plan with the lowest cost. We normalize cost function weights to have unit $l2$-norm. For CMA-ES, we used the PyCMA implementation (Hansen et al., 2019). We set $\sigma_0 = 0.05$, and use the default increasing population schedule. We stop CMA-ES after evaluating 85 candidate cost functions and return the cost function weights with the highest fitness. We also compare CMA-ES to a random search baseline, which searches for a high-fitness surrogate cost function by uniformly sampling 85 unit-norm weight vectors.

3.1. Scenario 1: Suboptimality Due to Short Horizon

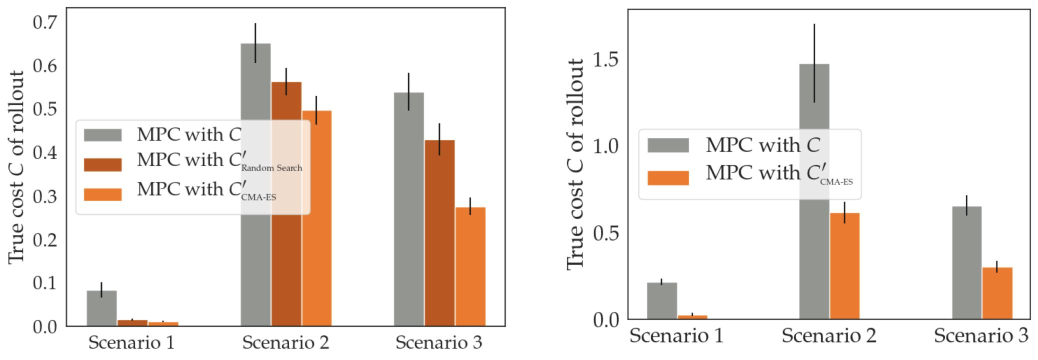
We first provide an example of how OCD-MPC can alleviate problems resulting from MPC using a shorter planning horizon than the true horizon of the problem. We consider the driving scenario shown in Figure 1(a) where the robot car is colored orange and there is a fixed speed human car (black). This true cost function in this scenario encodes costs for collisions, going off road, driving on lane lines, and traveling slower than a target speed.

2. Code and videos available at: `https://sites.google.com/berkeley.edu/ocd-mpc/`
3. We implemented these features following Sadigh et al. (2016), except we replace Gaussian functions with smooth bump functions and logistic functions with smooth step functions.



(a) Scenario 1: Short Horizon (b) Scenario 2: Local Opt. (c) Scenario 3: Replanning

Figure 1: (a)-(c) Qualitative comparisons over three scenarios where planning with MPC under the true cost function C results in suboptimal performance, but where these suboptimalities can be aleviated via a surrogate cost function C', which when optimized via MPC results in better performance under the true cost function C. The MPC robot is colored orange.



(a) No dynamics mismatch (b) Dynamics mismatch

Figure 2: Quantitative Costs under C for the different scenarios in Figure 1 when there is (a) no dynamics mismatch and (b) a mismatch between the true dynamics and the approximate dynamics model used for MPC planning (see Section 3.4). Error bars represent standard error over 7 trials.

Because the human car ahead of the robot is traveling slower than the robot's target speed, the robot must either switch lanes to maintain speed or slow down and stay in the center lane—the former behavior is preferred and has lower true cost.

Optimizing the true cost with MPC yields suboptimal behavior as shown in Figure 1(a) (left). The finite horizon planner used by MPC only plans $K = 5$ timesteps into the future, during which the car incurs costs for being between on or close to the lane line, but does not plan far enough into the future to realize the benefit of increased speed afforded by a lane change. Thus, the myopic planner plans to slow down and stay in the middle lane. By contrast, Figure 1(a) (right) shows that OCD-MPC finds a cost function C' which causes lane switching even with the default short-horizon MPC planner ($K = 5$). This yields lower true cumulative cost under C, mitigating the effects of short-horizon planning. In Figure 2(a) we show the quantitative difference in performance (measured under the true cost function C) between MCP rollouts optimized using C and MPC rollouts optimized with the learned cost function C'. Figure 2(*a*) shows that, for Scenario 1, OCD-MPC with CMA-ES achieves a 86% reduction in true cost compared to to directly optimizing the true cost with MPC. We also found that CMA-ES is more effective at decreasing the true rollout cost than the random search baseline. Figure 1(*a*) also shows heatmaps for the true and optimized cost functions. We analyzed the learned surrogate cost function, C', and found that, while the true cost provides no lane preference, the learned surrogate cost penalizes the robot for staying in the center of its lane. This explicitly guides the agent to switch lanes, whereas lane switching is only implicitly implied by C via a penalty for slowing down.

Learned Terminal Costs: As noted in the introduction, it is common to use a learned terminal cost to alleviate finite horizon issues with MPC. To compare OCD-MPC to this alternative, we learned a terminal cost using POLO (Lowrey et al., 2018) on the true reward C. We found that running MPC with C and the learned terminal cost reduces MPC rollout costs by 60%, compared to not using a terminal cost. However, running MPC with the learned surrogate cost C' plus the learned terminal cost reduces the cost by a full 80%. This suggests that learning terminal costs and optimal cost design are complimentary: OCD-MPC can still improve performance, even when MPC uses a learned terminal cost.

3.2. Scenario 2: Suboptimality due to Local Optimization

Next, we consider how OCD-MPC can alleviate problems associated with only planning locally optimal trajectories via MPC. Consider the scenario shown in Figure 1(*b*), where the robot car is initialized to the left of the human car and its task is to maneuver into the right lane. The true cost function C penalizes collisions, going off road, being far from the right lane, and traveling slower than the target speed. Figure 1(*b*)(left) shows that optimizing C via MPC with a single control initialization converges to a local optima. By contrast, OCD-MPC is able to learn a surrogate cost function C' which causes lane changing behavior without augmenting the number of MPC control initializations. As shown on the right in Figure 1(*b*), the robot is able to optimize a trajectory using C' that acheives the desired behavior encoded in the true cost C: the surrogate cost trajectory has the robot car (orange) slow down and then change lanes. The quantitative difference in performance under the true cost function C is shown in Figure 2(*a*). For Scenario 2, we found that OCD-MPC results in a 24% reduction in true cost compared to optimizing the true cost directly with MPC, and outperforms random search. We also plot heatmaps in Figure 1(*b*) representing the true and learned cost functions. The surrogate cost function reduces the penalty for collision and increases the cost of being far from the right lane. This changes the shape of the reward surface so that even with a single control initialization, gradient descent discovers the globally optimal lane switching behavior via vanilla MPC on C'.

3.3. Scenario 3: Suboptimality due to Replanning

In our third scenario, we consider suboptimal behavior that results from the fact that MPC plans trajectories without accounting for future replanning. We consider the scenario shown in Figure 1(*c*) in which the robot is between two lanes, approaching a slow human car which is also between lanes. The true cost incentivizes avoiding collisions, avoiding driving off-road, staying in the center of a lane, and maintaining a high speed. The robot knows the human will merge into one of the two lanes in the future, thinks both cases are equally likely, and plans based on expected cost under its belief distribution. Because the robot car incurs cost for traveling slowly behind the human car, the optimal behavior is to merge into whichever lane becomes free, once the human chooses a lane. However, MPC sees a human ahead of it in the worst case regardless of which lane it chooses, causing the robot to slow down to avoid a collision and to arbitrarily pick a lane to merge into to avoid driving on the lane line. MPC does not perform contingency planning and cannot foresee that it can maintain a higher speed and will be in a better position to choose a lane in future timesteps.

The learned surrogate cost function results in emergent contingency planning: the robot maintains its starting velocity and then picks the free lane once the human reveals their lane preference. Figure 2(a) shows that, for Scenario 3, OCD-MPC results in a 49% decrease in true cost compared to optimizing the true cost directly with MPC. Figure 1(c) shows heatmaps for C and C'. The surrogate cost function reduces the lane line penalty and has a larger penalty for not traveling at the target speed, resulting in the robot continuing behind the human and maintaining its speed. The MPC planned trajectory gradient is pushed to one lane by asymmetry after the human starts moving into one of the lanes. Note that in all the above scenarios there is an optimal long term behavior (e.g. switching lanes, staying centered until human steers) which is implicit in the true cost C but can't be produced by optimizing the true cost via MPC. The learned surrogate costs encode this optimal behavior more explicitly, making it easier for MPC to find the optimal solution.

3.4. Suboptimality due to an approximate dynamics model

The above results assume the robot has access to the true dynamics in its planner, as opposed to an approximate dynamics model (that is, $\hat{f} = f$). We now test what happens when this assumption is relaxed. To create a dynamics mismatch we add anisotropic noise to the true dynamics of the simulator. We simulate wind blowing across the highway by adding a small Gaussian distributed force in each of the three scenarios and then reran OCD-MPC on these three modified scenarios. The robot uses the unmodified $\hat{f}$ (deterministic dynamics model with no wind) when performing MPC, but actually moves according to the transition function f where the car is affected by wind. Note that the fitness function for OCD-MPC does roll out trajectories using the true dynamics f. We report our results in Figure 2(b). Not only can OCD-MPC learn a cost function that improves performance in each scenario, in all three scenarios, OCD-MPC is able to learn a cost function that performs similarly to the case without dynamics mismatch.

4. Generalization to unseen initial states

The results above demonstrate that OCD-MPC can improve the performance of MPC; however, is this because OCD-MPC is overfitting to the single initial start state which we optimize for? To test generalization, we varied the initial position and velocity of the car in each of the three scenarios above, and investigate whether we can learn a surrogate cost function that yields lower rollout costs under C across unseen initial conditions. In particular, we sample new initial states using a Gaussian distribution centered around the initial states shown in Figure 1. After sampling n_{init} initial states, we use CMA-ES to solve the optimal reward problem. We then evaluate the true rollout cost using a test set of 24 new initial states from the same distribution. Figure 3 shows the results over 10 replicates.

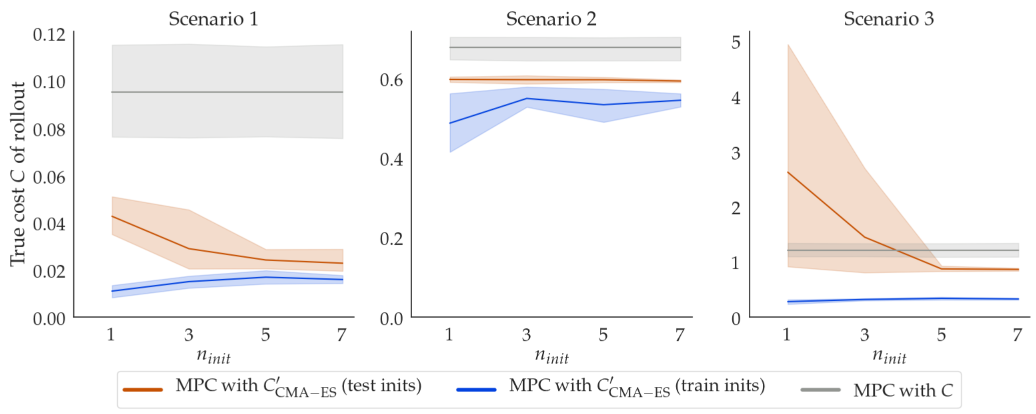


Figure 3: Generalization results from Section 4. In Scenarios 1 and 2, optimizing for a single initialization is sufficient to select a surrogate cost which performs well on unseen state initializations. In Scenario 3 (replanning), we need 5+ initializations to outperform planning with the true cost C. Shaded region represents the bootstrapped 95% confidence interval, calculated over 10 seeds.

In Scenario 1 and 2, we find OCD-MPC trained on a single initial condition is sufficient to reduce true rollout cost on unseen testing initial conditions. Furthermore, using more initializations during training yields slightly lower mean cost and lower variance on test initializations, even though this yields a higher cost on the training samples. This suggests that we are able to learn a cost function that generalizes across scenario initializations, while only performing slightly worse in any particular scenario. Scenario 3 is more complex, and training on fewer than 5 initializations yields a surrogate cost which occasionally does better than the true cost, but frequently performs worse. However, using more than 5 training initializations yields a surrogate cost which consistently outperforms rollouts generated under the true cost. Scenario 3 reveals the importance of using a training set which provides coverage of the test distribution. To find a C' which is robust to variation in initial state, one should evaluate cost functions on a set of initial states which provides good coverage of the range of possibilities. Furthermore, the existence of scenarios where a surrogate cost function C' can be found which yields lower cost under C and is robust to unseen variations of the scenario suggests that MPC can consistently fails to produce near-optimal behavior even if its cost function accurately encodes true task preferences.

5. Summary and Future Work

In this work, we proposed the Optimal Cost Design problem for MPC (OCD-MPC), as well as a zeroth-order optimization approach to solve this problem. Using this approach, we analyzed three autonomous driving scenarios where optimizing an alternate cost function via MPC yields lower cumulative cost under the true cost function. We also showed a degree of robustness of the learned cost functions to the initial state of our environments. While we focused on autonomous driving, we believe our approach is general enough to work on a variety of continuous control tasks. Future work includes applying OCD-MPC to more dynamic environments with more nuanced human models and investigating the feasibility of optimizing a single cost function that works well across a wider range of tasks.

Acknowledgements

This work is partly supported by AFOSR, NSF NRI SCHOOL, and ONR YIP.

References

Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335 – 348, 1989. ISSN 0005-1098. doi: https://doi.org/10.1016/0005-1098(89)90002-2. URL http://www.sciencedirect.com/science/article/pii/0005109889900022.

Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. URL https://doi.org/10.5281/zenodo.2559634.

Jason Hardy and Mark Campbell. Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. *IEEE Transactions on Robotics*, 29(4):913–929, 2013.

David Hoeller, Farbod Farshidian, and Marco Hutter. Deep value model predictive control. In *Conference on Robot Learning*, pages 990–1004, 2019.

Napat Karnchanachari, Miguel I Valls, David Hoeller, and Marco Hutter. Practical reinforcement learning for mpc: Learning from sparse objectives in under an hour on a real robot. In *2nd Annual Conference on Learning for Dynamics and Control (L4DC 2020)*, 2020.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

F Künhe, J Gomes, and W Fetter. Mobile robot trajectory tracking using model predictive control. In *II IEEE latin-american robotics symposium*, volume 51. Citeseer, 2005.

Jay H Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415, 2011.

William S Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, 28(1):47–65, 1991.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2018.

Jan Marian Maciejowski. *Predictive control: with constraints*. Pearson education, 2002.

Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic lqr tuning based on gaussian process global optimization. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 270–277. IEEE, 2016.

Chris J Ostafew, Angela P Schoellig, Timothy D Barfoot, and Jack Collier. Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking. *Journal of Field Robotics*, 33(1):133–152, 2016.

Martin Pelikan, David E Goldberg, and Fernando G Lobo. A survey of optimization by building and using probabilistic models. *Computational optimization and applications*, 21(1):5–20, 2002.

Ph Poignet and Maxime Gautier. Nonlinear model predictive control of a robot manipulator. In *6th International Workshop on Advanced Motion Control. Proceedings (Cat. No. 00TH8494)*, pages 401–406. IEEE, 2000.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, volume 2. Ann Arbor, MI, USA, 2016.

Dorsa Sadigh, Anca D. Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pages 2601–2606. Cognitive Science Society, 2009.

Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Jonathan Sorg, Richard L Lewis, and Satinder P Singh. Reward design via online gradient ascent. In *Advances in Neural Information Processing Systems*, pages 2190–2198, 2010.

Jonathan Daniel Sorg. *The Optimal Reward Problem: Designing Effective Reward for Bounded Agents.* PhD thesis, 2011.

Arjun Sripathy, Andreea Bobu, Daniel S. Brown, and Anca D. Dragan. Dynamically switching human prediction models for efficient planning. In *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021.

Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, Joseph E Gonzalez, Aaron Ames, and Ken Goldberg. Abc-lmpc: Safe sample-based learning mpc for stochastic nonlinear dynamical systems with adjustable boundary conditions. In *Algorithmic Foundations of Robotics (WAFR)*, 2020.

Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 528–535. IEEE, 2016.

Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez, and Emanuel Todorov. Value function approximation and model predictive control. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 100–107. IEEE, 2013.