2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Time-Optimal Trajectory Generation for Dynamic Vehicles: A Bilevel Optimization Approach

Gao Tang*, Weidong Sun*, and Kris Hauser

Abstract— **This paper presents a general framework to find time-optimal trajectories for dynamic vehicles like drones and autonomous cars. Hindered by its nonlinear objective and complex constraints, this problem is hard even for state-of-the-art nonlinear programming (NLP) solvers. The proposed framework addresses the problem by bilevel optimization. Specifically, the original problem is divided into an inner layer, which computes a time-optimal velocity profile along a fixed geometric path, and an outer layer, which refines the geometric path by a Quasi-Newton method. The inner optimization is convex and efficiently solved by interior-point methods. A novel variable reordering method is introduced to accelerate the optimization of the velocity profile. The gradients of the outer layer can be derived from the Lagrange multipliers using sensitivity analysis of parametric optimization problems. The method is guaranteed to return a feasible solution at any time, and numerical experiments on a ground vehicle with friction circle dynamics model show that the proposed method performs more robustly than general NLP solvers.**

I. Introduction

Time-optimal trajectory generation is an important topic in robotics to increase task efficiency. In this paper we present a novel bilevel optimization approach to generate time-optimal trajectories for dynamic vehicles. Existing techniques for time-optimal path generation can be categorized into direct collocation or two-stage approaches. The direct collocation approach optimize a discretized representation of the trajectory, both positions, velocities, and controls [1]. However, this approach has to handle nonlinear dynamics and non-convex constraints, requiring solution of a Nonlinear Programming (NLP) problem. There is no guarantee that an optimal, or even feasible solution is obtained. The two-stage approach is an approximation technique where a geometric path is optimized separately from the speed along the path. Each of these optimization steps is more numerically stable, and Time Optimal Path Parameterization (TOPP) approaches have been specialized to quickly and robustly optimize the speed along the path [2], [3]. However, in the two-stage approach the path is fixed after the first stage, and cannot be optimized further to reduce trajectory time.

We introduce a new bilevel optimization approach that solves the path optimization and speed optimization subproblems hierarchically: the inner subproblem solves a TOPP problem along a geometric path by convex optimization; the outer subproblem refines the geometric path to minimize its optimal traversal time. The key challenge is to calculate the gradient of the optimal traversal time with respect to the path shape, which is needed for the outer subproblem. This gradient is estimated analytically using the dual solution (Lagrange multipliers) of the inner subproblem, which is a by-product that can be obtained "for free" when using a convex solver. The inner optimization must be fast, so we employ a convex interior-point TOPP solver and introduce an efficient KKT factorization with linear scalability to the number of grid points.

The algorithm is applied to a racing ground vehicle whose dynamics obey a friction circle model. A feasible solution is obtained in every inner problem, so the optimizer can be terminated at any time to produce a feasible solution as long as the path is geometrically feasible. Our implementation indeed enforces that the path is guaranteed to be feasible in each outer iteration. Experiments demonstrate the robustness, scalability and any-time feasibility of our approach against state-of-the-art NLP solvers IPOPT [4] and SNOPT [5].

II. Related Work

A. Bilevel Optimization

Bilevel optimization refers to a mathematical program where one optimization problem has another optimization problem as one of its constraints, that is, one optimization task is embedded within another [6]. Bilevel and multi-level optimization techniques have been employed for switching time optimization for switched systems [7]–[10]. This literature has concentrated on calculating derivatives of an objective function with respect to switching times. Other applications include trajectory optimization for legged robots [10], robust control and parameter estimation [11], and optimizing time allocation for minimum-jerk trajectories [12].

Various algorithms are proposed to solve bilevel optimization, and we briefly introduce two that are closely related to our approach and refer readers to [6], [13] for more comprehensive treatments of the topic. The first approach replace lower level optimization problem with its KKT conditions. However, this approach introduce non-convex constraints. The second approach, called *descent method*, seeks to decrease the upper-level objective while keeping the new point feasible. Our method might be categorized as a descent method as we solve the upper-level optimization problem by Quasi-Newton method using gradients provided by the lower-level optimization problem.

This work was partially supported by NSF grant #IIS-1816540. W. Sun was partially supported by the China Scholarship Council (CSC). This research was performed while the authors were affiliated with Duke University.

G. Tang and K. Hauser are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA. e-mail: {gaotang2, kkhauser}@illinois.edu.

W. Sun is with XYZ Robotics Inc, Shanghai, China. e-mail: weidong.sun@xyzrobotics.ai.

* Denotes equal contribution

B. TOPP Solvers

Time-Optimal Path Parameterization (TOPP) aims to find admissible control inputs to a robot that minimizes the traversal time of a pre-specified path subject to dynamic, control and other system constraints. It is a well-studied problem in Robotics and can be efficiently solved by: (1) convex optimization (CO) [2], [14], [15], (2) numerical integration (NI) [3], [16], and (3) reachability analysis (RA) [17]. The best scalability is achieved by RA [17] which scales linearly with problem size. Our TOPP solver is in the CO class, but also achieves linear scalability by solving nonlinear CO directly and customizing the KKT solver. The benefit of CO methods is the ability to calculate Lagrange multipliers, which lets us calculate the gradient of the minimum time with respect to the path. To our knowledge, there are no other methods besides CO that are able to calculate multipliers.

Unlike existing CO approaches that convert to SOCP [2] or use Sequential Linear Programming [14], we directly solve the nonlinear convex problem using a primal-dual interior-point method. Our approach is different from [15] where a primal barrier approach is used and a strictly feasible initial point is required. Moreover, primal-dual methods are often more efficient than the primal barrier method, especially when high accuracy is required [18], and we also require the dual variables for our bilevel approach. By using a customized KKT solver, our CO approach scales linearly with the number of collocation points, compared to superlinear complexity in previous methods.

C. Time-Optimal Motion Planning for Vehicles

The topic of generating time-optimal trajectories for vehicles has been extensively studied [15], [19]–[24], and a general overview of motion planning for autonomous vehicles can be found in [25]. TOPP with a half-car model is addressed by Velenis and Tsiotras [24], and is employed as a submodule in Kapania *et al.* [19]. The work by Kapania *et al.*, which bears close resemblance to ours, presents a two-stage iterative method for generating time-optimal trajectories through a race course: their method first runs a forward-backward integration to determine the time-optimal velocity profile along a specified path and then updates the path by minimizing its curvature while obeying constraints. However, minimizing curvature is not the same as minimizing lap time, so their method provides no guarantee of optimality. By contrast, our approach updates the path using the true path gradient with respect to lap time. A similar two-stage iterative optimization technique is also employed by Xu *et al.* [23] where it is applied in the post-optimization stage to refine both the geometric path and speed profile, and can converge to a higher-quality trajectory within a few iterations. However, their technique is only demonstrated on problems of small scale and short horizon.

III. Problem Formulation

A. Time-Optimal Motion Planning Problem

We consider a system with generalized configuration $q \in \mathbb{R}^n$. The environment is represented by $\mathcal{X}_{\text{free}} \in \mathbb{R}^n$, which denotes all the collision-free configurations. We want to find a trajectory in $\mathcal{X}_{\text{free}}$ from a start configuration q_s to a configuration in a goal set $\mathcal{X}_{\text{goal}} \subseteq \mathcal{X}_{\text{free}}$ in a minimum amount of time while respecting the dynamics and control constraints, e.g., bounds on velocity and acceleration. The problem can be mathematically formulated as:

$$\begin{aligned} \underset{q(t),u(t)}{\text{minimize}} \quad & T \\ \text{subject to} \quad & q(t) \in \mathcal{X}_{\text{free}}, \quad \forall t \in [0,T] \\ & q(0) = q_s, \\ & q(T) \in \mathcal{X}_{\text{goal}}, \\ & f(q(t), \dot{q}^2(t), \ddot{q}(t), u(t)) \leq 0, \quad \forall t \in [0,T] \end{aligned} \tag{1}$$

where $u(t)$ is the control input, and $f(\cdot)$ includes constraints including dynamics.

Our bilevel approach optimizes path and time parameterization hierarchically. In next sections, we elaborate how the path is represented and time parameterization is optimized.

B. Path Representation

A feasible *geometric path* is a continuous curve $p : [0,1] \to \mathcal{X}_{\text{free}}$ that connects the start configuration q_s to goal configuration q_g such that $p(0) = q_s, p(1) = q_g$. We represented paths by piecewise Bézier curves with C^2 continuity for each DOF.

In order to achieve collision avoidance, we decompose $\mathcal{X}_{\text{free}}$ into several convex polyhedrons by convex decomposition. After decomposition, a list of connected polyhedrons from the start to the goal are found and the path is split into pieces with each piece contained in a separate polyhedron. This approach lets us easily achieve collision avoidance by constraining each segment to be inside its polyhedron.

For a space decomposed into K convex polygons, the path p of order m is represented by a vector of length nmK, $p = [c_0^1|c_1^1|\cdots|c_m^1|c_0^2|c_1^2|\cdots|c_m^K]^T$ where c_i^k is the ith control point for the kth segment.

If h_i is the number of hyperplanes bounding the ith convex polyhedron, we need a total of $n(m+1)\sum_{i=1}^{K} h_i$ linear inequality constraints to guarantee collision avoidance. This is represented by an inequality

$$Gp \leq g. \tag{2}$$

By the property of Bézier curve, it is sufficient to guarantee the curve is inside the polyhedron by only constraining the control points [12].

To enforce smoothness, we impose C^2 continuity constraints on connecting segments by constraining the control points. Also, the path should obey terminal constraints, i.e. $c_0^1 = q_s, c_m^K = q_g$. We can express these constraints with $(3K-1)n$ linear equality constraints, denoted as:

$$Hp = h. \tag{3}$$

C. Time-Optimal Path Parameterization

The goal of TOPP is to find a time parameterization to a geometric path, so that the traversal time is minimized while satisfying all constraints. As shown in [2], [14], it

can be formulated as a convex optimization problem under appropriate assumptions. Here we give a brief review.

A *time parameterization* is a monotonously increasing scalar function $s(t) : [0, T] \to [0, 1]$, where T is the traversal time of the path. The *trajectory*, which is the time-parametrized geometric path, can be represented as $q(t) = p(s(t)) : [0, T] \to \mathcal{X}_{\text{free}}$. By chain rule, we have

$$\dot{q}(t) = p'(s)\dot{s}(t), \quad \ddot{q}(t) = p'(s)\ddot{s}(t) + p''(s)\dot{s}^2(t) \tag{4}$$

where $\dot{\circ}$ denotes derivative with respect to time and $\circ'$ denotes derivative with respect to s.

In TOPP, values of $p'(s)$ and $p''(s)$ are known when the geometric path is given, the time allocation $s(t)$ has to be optimized. We introduce two variables $a(s) = \ddot{s}, \quad b(s) = \dot{s}^2$ which satisfy the following relationship

$$\dot{b}(s) = b'(s)\dot{s} = \frac{d(b(s))}{dt} = 2\dot{s}\ddot{s} = 2a(s)\dot{s}$$

or more simply,

$$b'(s) = 2a(s). \tag{5}$$

The traversal time T can be written in terms of $b(s)$ as

$$T = \int_0^T 1dt = \int_{s(0)}^{s(T)} \frac{1}{\dot{s}} ds = \int_0^1 \frac{1}{\dot{s}} ds = \int_0^1 \frac{1}{\sqrt{b(s)}} ds. \tag{6}$$

To get a convex optimization problem, following [14], [26], we discretize $s(t)$ into N segments: $\{b_i\}_{i=0}^N$. We assume that $a(s)$ is piece-wise constant between two consecutive discretization points, then $b(s)$ becomes a piece-wise linear function. From (5) we have $2a_i\Delta s_i = b_{i+1} - b_i$ where Δs_i is the size of the ith grid. Now (6) can be manipulated into

$$\begin{aligned} T &= \int_0^1 \frac{1}{\sqrt{b(s)}} ds = \sum_{i=0}^{N-1} \int_{s_i}^{s_{i+1}} \frac{1}{\sqrt{b(s)}} ds \\ &= \sum_{i=0}^{N-1} \frac{2\Delta s_i}{\sqrt{b_i} + \sqrt{b_{i+1}}}. \end{aligned} \tag{7}$$

The path constraints, i.e. f in (1) are imposed at the mid-points between two consecutive collocation points. These constraints can be all expressed as functions b_i, b_{i+1}, and $u_{i+\frac{1}{2}}$ after algebraic manipulation.

It turns out for the dynamic vehicle problem in this paper, all constraints are linear with respect to a, b, u. With objective function (7) being convex, TOPP can be solved by convex optimization. As a result, given a geometric path, we have a reliable approach to solve the corresponding TOPP. In the next section, we introduce several approaches to accelerate TOPP, before introducing how to improve the geometric path.

D. Bilevel Optimization

A bilevel optimization problem [6] is given by

$$\begin{aligned} \underset{x_u \in X_U, x_l \in X_L}{\text{minimize}} \quad & F(x_u, x_l) \\ \text{subject to} \quad & x_l \in \underset{x_l \in X_L}{\text{argmin}}\{ \quad f_0(x_u, x_l) : \\ & \qquad\qquad f_i(x_u, x_l) \le 0, i = 1, \dots, m \\ & \qquad\qquad h_i(x_u, x_l) = 0, i = 1, \dots, p\} \\ & G_i(x_u, x_l) \le 0, \quad i = 1, \dots, M \\ & H_i(x_u, x_l) = 0, \quad i = 1, \dots, P \end{aligned}$$

where G and H denote upper-level constraints, f_0 is the lower-level objective, while $\{f_i(\cdot)\}_{i=1}^m$ and $\{h_i(\cdot)\}_{i=1}^p$ are lower-level constraints.

In some literature, the upper and lower level problems are called leader and follower. The leader makes decision x_u and knows the follower's reaction x_l which is obtained by optimizing its cost function after observing x_u. The goal for the leader is to consider the follower's action and find x_u that optimize its own cost function. If we assume the lower level problem has a unique solution, x_l is a function of x_u. As a result, we can directly optimize x_u by treating x_l as an implicit function of x_u. The problem is the difficulty to find a closed-form expression of x_l as a function of x_u.

In our problem, the upper problem optimizes the geometric path with an objective to minimize the optimal traversal time, which is computed from inner optimization. The upper-level constraints only depend on the path itself. If we treat the inner optimization as a black-box objective function, the outer optimization can be thought of as a nonlinear program with a an expensive objective function. However, optimization tends to be more efficient if the gradient of the objective is available. Luckily, in our problem the inner and outer problem have the same cost function, so the outer gradient can be computed using sensitivity analysis.

For brevity, we give the results of first-order sensitivity analysis and refer readers to [27] for details. Consider the problem of finding the local solution $x_l(x_u)$, i.e. the inner optimization problem as a parametric NLP problem:

$$\begin{aligned} \text{minimize} \quad & f_0(x_u, x_l) \\ \text{subject to} \quad & f_i(x_u, x_l) \le 0, \quad i = 1, \dots, m, \\ & h_i(x_u, x_l) = 0, \quad i = 1, \dots, p, \end{aligned} \tag{8}$$

where $x_l \in X_L$ is the vector of decision variables and $x_u \in X_U$ is a parameter vector. Under mild assumptions, given a local optimal solution $x_l^\star(x_u)$, and multipliers λ and ν associated with f and h, respectively, then

$$\nabla_{x_u} f_0^\star(x_u) = \nabla_{x_u} f_0 + \sum_{i=1}^m \lambda_i(x_u) \nabla_{x_u} f_i + \sum_{i=1}^p \nu_i(x_u) \nabla_{x_u} h_i.$$

In time-optimal motion planning, x_u is the path parameterization and x_l is time parameterization.

E. Solving Bilevel Optimization

An algorithm for solving bilevel optimization is presented in Algorithm 1. The algorithm takes an initial guess of the geometric path p_0 and the linear constraints on the path. In each iteration, the TOPP solver takes a path p and outputs a cost J, Lagrange multipliers λ and time parameterization $\{b_i\}_{i=0}^N$. The gradient of the path p is computed using Lagrange multipliers, and later used to update the path. The upper-level optimization is essentially a nonlinear optimization with linear constraints and the optimality conditions are checked by the optimizer.

We note that the Get-Gradient function is implemented by sensitivity analysis using Lagrange multipliers. Any gradient-based method can be used as the Take-A-Step function which updates p based on gradient g and possibly its history (in

Quasi-Newton approaches). We use off-the-shelf NLP solver SNOPT to perform Take-A-Step function. Even though we are using a nonlinear solver, the constraints in upper-level optimization are linear so feasibility is always guaranteed.

Algorithm 1 Bilevel-Solver (p_0, G, g, H, h)

$p \leftarrow p_0$
for $i \leftarrow 0$ to *max-iterations* **do**
  $J, \lambda, \{b_i\}_{i=0}^N \leftarrow$ TOPP(p)
  $\nabla \leftarrow$ Get-Gradient(λ)
  $p \leftarrow$ Take-A-Step$(p, J, \nabla, G, g, H, h)$
  if optimality-conditions-satisfied **then**
    break
return $P, \{b_i\}_{i=0}^N$

F. Accelerating TOPP

It is important that the TOPP problem is solved efficiently since it is the lower optimization problem. To this end, a primal-dual interior point method with a customized KKT solver is employed.

1) Nonlinear Objective: Verscheure *et al.* [2] formulated TOPP as a second-order cone programming. However, as noted in [15], this approach is inefficient because the size of the original problem might even be doubled through the introduction of slack variables. In this paper, the nonlinear objective is directly solved.

2) Eliminating Variables: To reduce the number of variables, we replace $\{a_i\}_{i=0}^{N-1}$ with a linear combination of $\{b_i\}_{i=0}^N$, i.e., $a_i = (b_{i+1} - b_i)/2\Delta s_i$.

3) Customized KKT solver: Since an interior point solver spends most of its time in solving the KKT system, an efficient linear solver that exploits the structure of the problem allows significant acceleration. This observation dates back to [28] and we extend the KKT solver described in [15] to primal-dual interior point methods.

For a convex optimization problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_k(x) \preceq 0, \quad k = 1, \dots, m \\ & Gx \preceq h, \\ & Ax = \beta, \end{aligned}$$

the computation time needed for a primal-dual interior point method almost entirely depends upon how fast we can solve the Karush-Kuhn-Tucker (KKT) system [26]:

$$\begin{bmatrix} H & A^T & \tilde{G}^T \\ A & 0 & 0 \\ \tilde{G} & 0 & -W^T W \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \tag{9}$$

where

$$H = \sum_{k=0}^{m} z_k \nabla^2 f_k(x), \quad \tilde{G} = [\nabla f_1(x) \ \cdots \ \nabla f_m(x) \mid G^T]^T,$$

matrix W is a block-diagonal scaling matrix that depends on the current iterates; y and z are the multiplier associated with equality and inequality constraints; and r_x, r_y, r_z are residuals of the KKT system. Matrix $\tilde{G}$ is composed of two parts where the upper parts collects Jacobian of nonlinear inequality constraints, i.e. f_1 to f_m and lower parts collects linear inequality constraints, i.e. G. In our problem W is diagonal matrix since all the cone constraints are non-negative orthants and Nesterov-Todd scaling [26] is used.

A detailed description of how (9) is solved is given in section IV-B where the structure of matrices A and G are described. However, we note that similar structures also appear in TOPP problem of other systems.

IV. Numerical Experiment

We present the dynamics model of the ground vehicle and define how the TOPP problem is formulated. We also illustrate how to exploit KKT system structure, and present a comparison of our framework against off-the-shelf NLP solvers. In all experiments, computation is performed on a PC running Ubuntu 16.04 with 4.00 GHz CPU and 32 GB memory without any parallelization. Implementations are mainly based on Python, but the interior-point solver is written in C++. The NLP solvers are implemented in compiled languages for efficiency, but they are called using cost and constraint functions written in Python.

A. Dynamics Model

We use the front wheel drive friction circle vehicle model [15] which is derived from the observation that the maximum force that can be applied to the car is limited by the friction between tires and the road, with a sufficient powerful engine and braking system. The system dynamics is

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \ddot{q}_x \\ \ddot{q}_y \end{bmatrix} \tag{10}$$

where q_x, q_y is the vehicle's position $\theta = \tan^{-1}(\dot{q}_y/\dot{q}_x)$ denotes the heading angle, u_x, u_y are longitudinal and lateral acceleration. An illustration of the dynamics model is in Fig. 1. The acceleration has to stay in a friction circle, i.e.,

$$\sqrt{u_x^2 + u_y^2} \le \mu g, u_x \le \mu_s \mu g \tag{11}$$

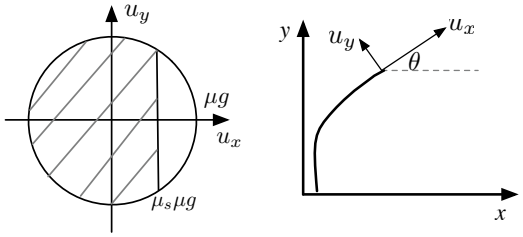
where μ is the friction coefficient, g is gravitational acceleration, $\mu_s = 0.5$ is the weight ratio. The admissible control lies in a convex set. In this model, braking has larger acceleration than speeding up because all four tires are engaged.

In TOPP formulation, the constraints in (10) and (11) are imposed at mid-points between collocation points. For segment i, the constraints are

$$\begin{aligned} &\begin{bmatrix} \cos\theta_{i+\frac{1}{2}} & -\sin\theta_{i+\frac{1}{2}} \\ \sin\theta_{i+\frac{1}{2}} & \cos\theta_{i+\frac{1}{2}} \end{bmatrix} \begin{bmatrix} u_{xi} \\ u_{yi} \end{bmatrix} = \\ &\begin{bmatrix} p_x''(s_{i+\frac{1}{2}}) & p_x'(s_{i+\frac{1}{2}}) \\ p_y''(s_{i+\frac{1}{2}}) & p_y'(s_{i+\frac{1}{2}}) \end{bmatrix} \begin{bmatrix} \frac{b_i + b_{i+1}}{2} \\ \frac{b_{i+1} - b_i}{2\Delta s} \end{bmatrix} \end{aligned} \tag{12}$$

which are linear constraints w.r.t. both b and u. Here $b \equiv \dot{s}^2$ is the squared velocity parameter introduced in Sec. III-C. The friction circle model introduces two inequality constraints for each control, one linear and one nonlinear, i.e.

$$u_{xi} \le \mu_s \mu g, \quad u_{xi}^2 + u_{yi}^2 \le \mu^2 g^2 \tag{13}$$



(a) Friction circle model (b) Vehicle dynamics

Fig. 1. (a) Friction circle model. Shaded area is the feasible region. (b) Vehicle dynamics model. u_x is parallel with velocity direction; θ is heading angle.

As shown in Fig. 1, the admissible control is in a convex set. Additionally, we constrain bounds on b_i The lower bound is 0 since b_i appears in a square root, so it must be positive. We impose an upper bound on b_i, arbitrarily set to 200, for better numerical stability.

B. Customized KKT Solver

In the following analysis optimization variable x is assembled in the form of $[b_0, \ldots, b_N, u_{x0}, u_{y0}, \ldots, u_{xN-1}, u_{yN-1}]$.

In order to solve (9), the first manipulation is to reduce it to the so-called reduced KKT system by eliminating z:

$$\begin{bmatrix} S & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r_x + \tilde{G}^T W^{-1} W^{-T} r_z \\ r_y \end{bmatrix} \tag{14}$$

where

$$S = H + \tilde{G}^T W^{-1} W^{-T} \tilde{G},$$

and $\tilde{G}$ in this problem includes both linear and nonlinear inequality constraints. The nonlinear constraints are linearized at current value of x during iteration. After solving this reduced KKT system, the solution z can be recovered from x, y as:

$$Wz = W^{-T}(\tilde{G}x - r_z),$$

which is cheap since W is diagonal. We note that the steps of eliminating and recovering z are present in both CVXOPT and IPOPT.

The reduced KKT system is sparse (item KKT in Fig. 2), and is composed of the tridiagonal matrix S and the banded block matrix A. However, the default solver in CVXOPT is inefficient at solving it, because it eliminates x and gets an equation on y whose left hand side matrix is $AS^{-1}A^T$. Although computing $S^{-1}A^T$ is relatively cheap since S is tridiagonal, matrix $AS^{-1}A^T$ is not generally sparse (CVXOPT in Fig. 2).

We customize our KKT solver by applying a reordering technique which optimizes the sparsity pattern of the reduced KKT system, an idea borrowed from [15]. Reordering the KKT system is essentially applying a permutation matrix P to (14):

$$P \begin{bmatrix} S & A^T \\ A & 0 \end{bmatrix} P^T P \begin{bmatrix} x \\ y \end{bmatrix} = P \begin{bmatrix} r_x + \tilde{G}^T W^{-1} W^{-T} r_z \\ r_y \end{bmatrix}$$

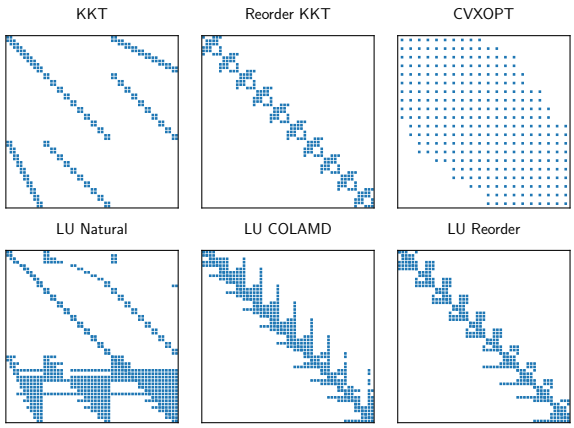


Fig. 2. Top row: sparsity patterns of the KKT matrices of different solvers, showing reduced KKT matrix (KKT); manually reordered KKT matrix, with bandwidth of 6 (**ours**); and the method used in CVXOPT, which uses $AS^{-1}A$ after removing entries with magnitude smaller than 10^{-6}. This is essentially dense. Bottom row: sparsity of lower and upper parts after SuperLU factorization with different reordering methods. Natural reordering gives 668 nonzeros; COLAMD column permutation gives 457 nonzeros; and our custom reordering gives 383 nonzeros.

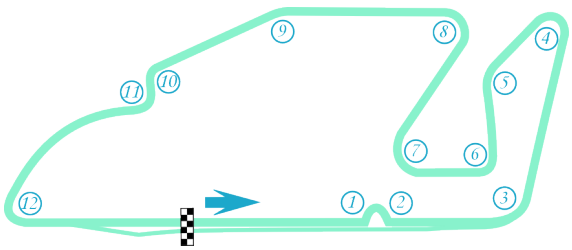
TABLE I
COMPARISON OF RUNNING TIME (IN SEC.) ON KKT SOLVERS

Solver	N=25	N=50	N=100	N=200
CVXOPT	0.119	0.749	7.130	48.940
LU+Reorder **(Ours)**	0.010	0.017	0.032	0.069
LU+Natural	0.011	0.028	0.093	0.414
LU+COLAMD	0.009	0.018	0.035	0.072

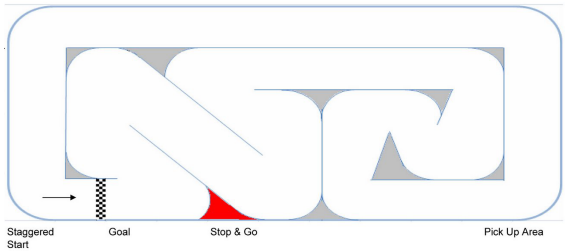
Here y is a $2N$ vector associated with equality constraints, of the order $[y_{x0}, y_{y0}, \ldots, y_{xN-1}, y_{yN-1}]$. After permutation, the new order of $[x^T y^T]^T$ becomes $[b_0, w_0, \ldots, b_{N-1}, w_{N-1}, b_N]$ where $w_i = [u_{xi}, u_{yi}, y_{xi}, y_{yi}]$. By doing so, the permuted matrix has a small bandwidth (Reorder KKT in Fig. 2). We use an LU solver to solve the permuted sparse matrix.

We further compare our customized KKT solver against the SuperLU [29] solver with different orderings to reduce fill-ins. The resulting sparsity patterns are shown in the bottom row of Fig. 2. If SuperLU is used with natural reordering (no reordering), many fill-ins are required. COLAMD reordering performs better, but not as well as our custom reordering.

Tab. I compares scalability of each method. In this environment, we use a fixed number of 9 convex regions and vary the discretization of the grid over the speed variable. Specifically, each polygon is assigned N grid points in the s dimension. The CVXOPT solver forms dense matrices and scales cubically in N. SuperLU with natural ordering has approximately quadratic scalability. SuperLU with COLAMD reordering performs slightly worse than our reordering method, and this is not unexpected since it has more fill-ins. The table also shows that our solver scales linearly.



(a) First track



(b) Second track

Fig. 3. Layout of the two tracks.

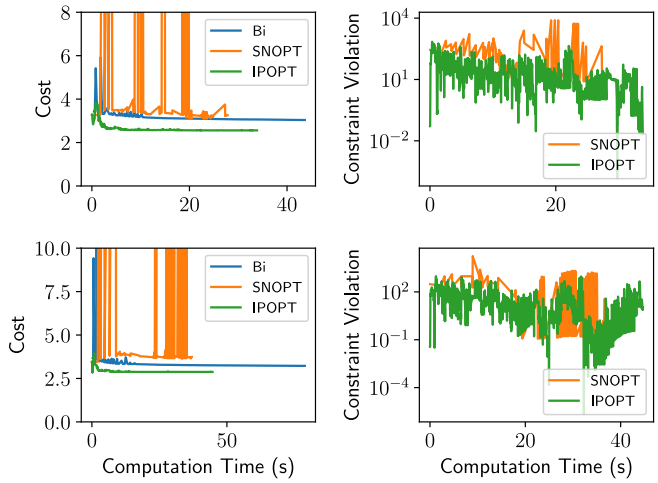


Fig. 4. Cost and constraint violation over time for bi-level optimization and two NLP solvers. The top and bottom rows show results for Track 1 and Track 2, respectively. Function values are recorded whenever a the objective and constraint function is evaluated, so the spikes are caused by line search with inappropriate step length. The right side shows that neither SNOPT nor IPOPT reach a feasible iterate after the initial guess.

C. Results on Race Tracks

We test the algorithm on models of two real-world race tracks. Track 1, "Circuit Ricardo Tormo," [1] is a racing circuit with length 3.090 km and width 1.029 km. Track 2, from the Tamiya Asia Cup Finals 2011 [2], is a RC car track with size 25 m by 11 m. Their layouts are shown in Fig. 3.

These two tracks are quite different in terms of both track size and road width. To turn them into nondimensional problems, we use the maximum of track length and width as the length unit, the square root of it as time unit, and the mass of the vehicle as mass unit. With this choice, the gravitational acceleration stays at 9.8, track width and height are at most 1, and other nondimensional quantities like friction coefficient stays the same. In the following results, the cost (traversal time) is reported in nondimensional units.

The tracks are decomposed manually into a sequence of convex regions, where the maximum length of a quadrilateral is limited to 0.15 units. The decomposed tracks are shown in Fig. 5. For the vehicle model, the friction coefficient[3] is chosen as $\mu = 0.9$ between dry road and rubber and $\mu_s = 0.5$. The Bézier curve order is chosen as 6.

Bilevel optimization is compared with NLP solvers SNOPT and IPOPT applied to the joint path and speed optimization problem. The bilevel solver used a constraint feasibility tolerance of 10^{-6}, optimality tolerance of 10^{-6}, and at most 100 major iterations. For SNOPT we used a feasibility tolerance of 10^{-6} and optimality tolerance of 10^{-4}, at most 500 major iterations, and 50,000 minor iterations. For IPOPT we used a feasibility tolerance of 10^{-6}, optimality tolerance of 10^{-2}, and at most 500 iterations. SNOPT and IPOPT are provided with analytic derivatives and problem sparsity are exploited to the best of the authors' ability. Fewer major iterations are used in bilevel optimization because each function evaluation requires a call to the interior-point solver. All three solvers are initialized with the same feasible geometric path, and the timing provided to SNOPT and IPOPT is a feasible point obtained by solving TOPP.

The results for the two tracks are shown in Fig. 4. Bilevel optimization reduces the cost in early stages, but it converges slowly later. Throughout the optimization, the iterates remain feasible. In contrast, the NLP solvers fail to satisfy constraints throughout the optimization process, and fail to converge despite the looser optimality tolerances. The NLP solvers, especially IPOPT, reduce the cost function quickly, but as the right side of Fig. 4 shows, these solutions are indeed infeasible. This suggests high numerical sensitivity despite the existence of a feasible solution. The stability of our method suggests that decoupling path optimization and time allocation results in better numerical stability.

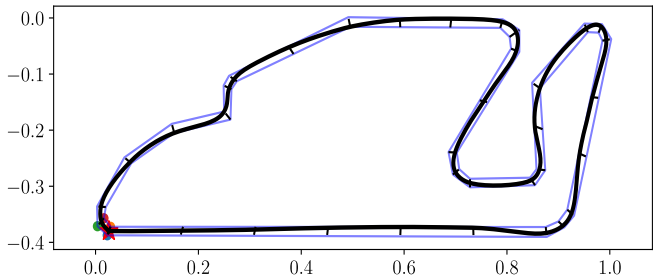
V. Conclusions And Future Work

We present a bilevel optimization framework to solve NLP with spatial and temporal constraints. It is particularly suited for problems in which the time-domain dynamics are convex, since the inner optimization can be solved reliably using interior-point methods. Moreover, if the feasibility of the path can be guaranteed, the bilevel optimization framework is anytime and highly robust. A second contribution is a TOPP solver based on an efficient KKT solver for interior-point method, which scales linearly with the number of collocation points. Experiments show that trajectory optimization on real world racing tracks are solved reliably, while standard NLP solvers fail to maintain feasibility of the solution.

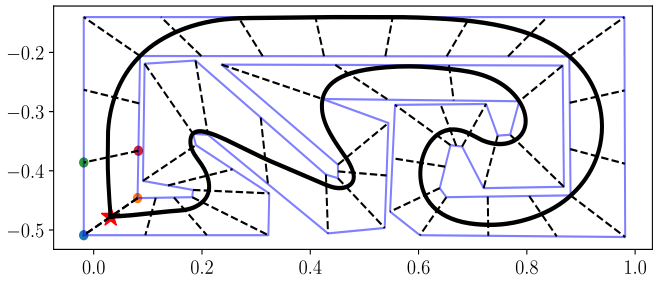
[1] https://formula-e.fandom.com/wiki/Circuit_Ricardo_Tormo, last accessed Feb. 27 2019

[2] http://quantumracing-rc.blogspot.com/2011/09/tamiya-asia-cup-finals-2011-in.html, last accessed Feb. 27 2019

[3] http://hyperphysics.phy-astr.gsu.edu/hbase/Mechanics/frictire.html, accessed Feb. 27 2019



(a) Track 1



(b) Track 2

Fig. 5. Optimal geometric path from bi-level optimization. Convex decomposition is shown by the quadrilaterals with dashed lines. The black curve is the geometric path. The red star denotes the starting position. The solution agrees with intuition that optimal paths should take wide turns, crossing corners when possible.

Future work should address the observation that our method reduces the cost significantly in the early stages but converges rather slowly. One possible reason is that the function being optimized is non-smooth, which hinders convergence. Using solvers with warm-start capacity in the lower level optimization has the potential to dramatically speed up this framework. We are also interested in extending the friction circle model to other vehicle dynamical models including aerodynamic drag, weight transfer and other factors, and applying our approach to other dynamic vehicles like drones and fixed-wing aircraft.

REFERENCES

[1] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[2] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, Oct 2009.

[3] Q. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, Dec 2014.

[4] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006. [Online]. Available: https://doi.org/10.1007/s10107-004-0559-y

[5] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM J. on Optimization*, vol. 12, no. 4, pp. 979–1006, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1137/S1052623499350013

[6] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE T. Evolutionary Computation*, vol. 22, pp. 276–295, 2018.

[7] X. Xu and P. J. Antsaklis, "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Transactions on Automatic Control*, vol. 49, no. 1, pp. 2–16, Jan 2004.

[8] M. Egerstedt, Y. Wardi, and F. Delmotte, "Optimal control of switching times in switched dynamical systems," in *IEEE Int. Conf. Decision and Control*, vol. 3, Dec 2003, pp. 2138–2143 Vol.3.

[9] E. R. Johnson and T. D. Murphey, "Second-order switching time optimization for nonlinear time-varying dynamic systems," *IEEE T. Automatic Control*, vol. 56, no. 8, pp. 1953–1957, Aug 2011.

[10] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *IEEE Int. Conf. Robotics and Automation*, 2017, pp. 93 – 100.

[11] B. Landry, Z. Manchester, and M. Pavone, "A differentiable augmented lagrangian method for bilevel nonlinear optimization," *CoRR*, 2019. [Online]. Available: https://arxiv.org/abs/1902.03319

[12] W. Sun, G. Tang, and K. Hauser, "Fast UAV trajectory optimization using bilevel optimization with analytical gradients," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1811.10753

[13] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of Operations Research*, vol. 153, no. 1, pp. 235–256, Sep 2007. [Online]. Available: https://doi.org/10.1007/s10479-007-0176-2

[14] K. Hauser, "Fast interpolation and time-optimization with contact," *Int. J. Robotics Research*, vol. 33, no. 9, pp. 1231–1250, 2014. [Online]. Available: https://doi.org/10.1177/0278364914527855

[15] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *International Journal of Control*, vol. 87, no. 6, pp. 1297–1311, 2014. [Online]. Available: https://doi.org/10.1080/00207179.2013.875224

[16] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985. [Online]. Available: https://doi.org/10.1177/027836498500400301

[17] H. Pham and Q. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, June 2018.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[19] N. R. Kapania, J. Subosits, and J. Christian Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," *J. Dynamic Systems, Measurement, and Control*, vol. 138, 04 2016.

[20] J. h. Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *2013 American Control Conference*, June 2013, pp. 188–193.

[21] F. Bayer and J. Hauser, "Trajectory optimization for vehicles in a constrained environment," in *IEEE Conf. Decision and Control (CDC)*, Dec 2012, pp. 5625–5630.

[22] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *PACIFIC J. MATHEMATICS*, 1990.

[23] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2061–2067.

[24] E. Velenis and P. Tsiotras, "Optimal velocity profile generation for given acceleration limits; the half-car model case," in *IEEE Int. Symp. Industrial Electronics, 2005. ISIE 2005.*, vol. 1, 2005, pp. 361–366.

[25] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE T. Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016. [Online]. Available: https://doi.org/10.1109/TIV.2016.2578706

[26] L. Vandenberghe, "The cvxopt linear and quadratic cone program solvers," 2010. [Online]. Available: http://www.seas.ucla.edu/~vandenbe/publications/coneprog.pdf

[27] A. V. Fiacco and Y. Ishizuka, "Sensitivity and stability analysis for nonlinear programming," *Annals of Operations Research*, vol. 27, no. 1, pp. 215–235, Dec 1990. [Online]. Available: https://doi.org/10.1007/BF02055196

[28] S. J. Wright, "Interior point methods for optimal control of discrete time systems," *Journal of Optimization Theory and Applications*, vol. 77, no. 1, pp. 161–187, Apr 1993. [Online]. Available: https://doi.org/10.1007/BF00940784

[29] X. Li, J. Demmel, J. Gilbert, iL. Grigori, M. Shao, and I. Yamazaki, "SuperLU Users' Guide," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-44289, September 1999, http://crd.lbl.gov/~xiaoye/SuperLU/. Last update: August 2011.