2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

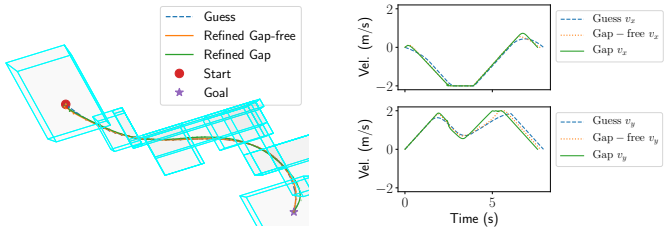
Enhancing Bilevel Optimization for UAV Time-Optimal Trajectory using a Duality Gap Approach

Gao Tang[1], *Student Member, IEEE*, Weidong Sun[2], and Kris Hauser[1], *Senior Member, IEEE*

***Abstract*—Time-optimal trajectories for dynamic robotic vehicles are difficult to compute even for state-of-the-art nonlinear programming (NLP) solvers, due to nonlinearity and bang-bang control structure. This paper presents a bilevel optimization framework that addresses these problems by decomposing the spatial and temporal variables into a hierarchical optimization. Specifically, the original problem is divided into an inner layer, which computes a time-optimal velocity profile along a given 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. The gradients of the outer layer can be analytically obtained using sensitivity analysis of parametric optimization problems. A novel contribution is to introduce a duality gap in the inner optimization rather than solving it to optimality; this lets the optimizer realize warm-starting of the interior-point method, avoids non-smoothness of the outer cost function caused by active inequality constraint switching. Like prior bilevel frameworks, this method is guaranteed to return a feasible solution at any time, but converges faster than gap-free bilevel optimization. Numerical experiments on a drone model with velocity and acceleration limits show that the proposed method performs faster and more robustly than gap-free bilevel optimization and general NLP solvers.**

I. Introduction

Time-optimal trajectory optimization is important for drones, vehicles, and industrial manipulators to complete tasks efficiently, but it is inherently difficult even for linear systems due to its bang-bang control structure. Typical approaches to trajectory optimization, such as Direct Collocation (DC), can handle a wide variety of costs, state constraints, and control constraints by converting the problem into a nonlinear programming (NLP) optimization, which can be solved by numerical techniques [1]. For time minimization, the state and control are functions of time and solved simultaneously with the optimal trajectory time. This approach introduces significant nonlinearity in the dynamics and non-convexity in constraints, so there is no guarantee that an optimal, or even feasible solution is obtained. Alternatively, a two-stage approach approximately solves the problem by optimizing the geometric path separately from the velocity profile, which exploits the speed and robustness of Time Optimal Path Parameterization (TOPP) [2], [3]. The drawback of two-stage optimization is that the path is fixed after the first stage, while it is possible to further to refine it to reduce trajectory time. The authors' recent work has explored the use of *bilevel optimization* to optimize both the path and the velocity in a hierarchical fashion [4]. The gradients of the optimal traversal time with respect to the path can be computed analytically, allowing the outer optimization to be addressed as a standard NLP while the convexity of the TOPP problem lends itself to efficient and reliable Interior-Point Methods (IPMs).



(a) Spatial shape (b) Velocity profile

Fig. 1: (a) Initial and refined trajectory for a randomly generated problem. The feasible space of the environment is split into convex boxes in which the path must lie, and an initial minimum-jerk trajectory is refined to decrease trajectory traversal time with velocity and acceleration limits. With the proposed gap technique, the trajectory converges more quickly toward an optimum. (b) It also achieves a lower execution time compared with a gap-free approach.

This paper makes further improvements to the bilevel optimization framework. In [4], the efficiency is limited by the slow convergence rate of the outer optimization, and warm-starting of the inner IPM is not effective, because the prior optimum reaches the boundary of inequality constraints and is far from the central path for the new problem. Furthermore, the cost function in outer optimization is continuous but non-smooth due to changes of active inequalities in the inner optimization. Function smoothness is important for the convergence rate of gradient based methods such as the difference between gradient and sub-gradient methods. To handle these problems, we introduce a *duality gap* which keeps the solution away from the boundary, and deliberately do not solve the optimization to optimality. With a controlled duality gap, inequality constraints are never active and thus there is no switch of active constraint and the solution of IPM is never at the boundary. This trick helps to achieve 1) warm-start of inner optimization and 2) a smoother outer cost function landscape, which significantly accelerates inner and outer optimization, respectively. We prove that solving to a desired duality gap is equivalent to the log-barrier method for penalizing inequality

*This work was supported by NSF grant #IIS-1816540

[1]Gao Tang and Kris Hauser are with Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, USA (gaotang2,kkhauser)@illinois.edu

[2]Weidong Sun is with XYZ Robotics, Shanghai, China weidong.sun@xyzrobotics.ai

constraints, and the gradient computation time is equivalent to the method of [4].

Numerical experiments on an aerial vehicle with velocity and acceleration limits demonstrate the robustness, efficiency, and optimality of our approach over standard bilevel optimization, direct collocation, and simultaneous optimization of the path and velocity profile, using state-of-the-art NLP solvers IPOPT [5] and SNOPT [6].

II. Related Work

A general overview of motion planning for autonomous vehicles can be found in [7]. The time-optimal problem for vehicles has been extensively studied [8]–[14]. The brittleness of spatio-temporal trajectory optimization using direct collocation has been noted by other researchers, leading to a search for alternative, more robust techniques.

A. TOPP Solvers

Time-Optimal Path Parameterization (TOPP) seeks admissible control inputs that minimizes the traversal time of a prespecified path subject to constraints such as system dynamics, control actuation, and velocity limit. Efficient approaches to it include: (1) convex optimization (CO) [2], [13], [15], (2) numerical integration (NI) [3], [16], and (3) reachability analysis (RA) [17]. Our TOPP solver solves the nonlinear optimization problem directly and exploiting sparsity in the KKT solver [4] for efficiency. Since CO methods are used, Lagrange multipliers are by-product and allow for calculation of the gradient of the minimum time with respect to the path. No other methods besides CO can provide such gradient information. Similar to [4], we use primal-dual IPM to solve the nonlinear CO directly, instead of converting to SOCP [2] or using Sequential Linear Programming [15] or primal barrier approach [13]. The sparsity of KKT solver is exploited to achieve linear scalability.

B. Two-stage optimization

TOPP with a half-car model is studied by Velenis and Tsiotras [14], and is used as a submodule in Kapania *et al.* [8]. There is work trying to remove the limitation of the two-stage approach by updating the geometric path too. Kapania *et al.* presents a two-stage iterative method for generating time-optimal trajectories through a race course. However, they are minimizing path curvature to update the path which is different from minimizing lap time. By contrast, our approach updates the path with a goal of minimizing traversal time.

C. Bilevel Optimization

Bilevel optimization refers to an optimization problem with another optimization problem embedded as its constraint [18]. Bilevel optimization techniques have been employed to optimize the switching times for switched systems [19]–[22]. This literature has concentrated on the calculation of the cost function's derivative with respect to switching times. It has also been used for legged robots [22], robust control and parameter estimation [23], and time allocation optimization for minimum-jerk trajectories [24].

Various algorithms are proposed to solve bilevel optimization and a comprehensive treatment is in [18], [25]. We briefly introduce two that are closely related to our approach. The first approach replace inner level optimization problem with its KKT conditions as constraints which may be non-convex such as the strict complimentary slackness condition. The second approach, called *descent method*, seeks to decrease the outer-level objective while keeping the new point feasible. Our method can be categorized as a descent method, as we optimize the outer-level objective by Quasi-Newton method with analytic gradients provided by the inner-level optimization problem. The descent method requires the sensitivity of the optimization which has been studied in [26]–[28].

III. Problem Formulation

A. Time-Optimal Motion Planning Problem

A system has generalized configuration $q \in \mathbb{R}^n$ and operates in an environment represented by $\mathcal{X}_{\text{free}} \in \mathbb{R}^n$, which denotes all the collision-free configurations. The problem is 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 constraints such as dynamics, collision avoidance, and state and control bounds. The problem can be mathematically formulated as:

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

where $u(t)$ is the control input, T is the trajectory duration, and $d(\cdot)$ collects various constraints mentioned before.

This problem is difficult to solve due to its non-convexity since it has to optimize path $x(t)$ and time T simultaneously. Our bilevel approach optimizes path and time parameterization hierarchically.

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 and goal configuration q_g such that $p(0) = q_s, p(1) = q_g$. We represent paths using piecewise Bézier curves with C^2 continuity for each DOF.

In order to be collision-free, we decompose $\mathcal{X}_{\text{free}}$ into several convex polyhedrons by convex decomposition. Decomposing non-convex $\mathcal{X}_{\text{free}}$ into connected convex regions is typical practice, such as [29]. The decomposition returns a list of connected polyhedrons and the path is split into pieces accordingly 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 regions, the path p of order m is represented by a vector of length $n(m+1)K$, $p = [c_0^1|c_1^1|\cdots|c_m^1|c_0^2|c_1^2|\cdots|c_m^K]^T$ where c_i^k is the ith control point for the kth segment.

If h_i is the number of hyperplanes bounding the i-th convex polyhedron, a total of $n(m+1)\sum_{i=1}^{K} h_i$ linear inequality constraints are sufficient to guarantee collision avoidance. This is represented by an inequality like (with properly chosen A_I and v_i):

$$A_I p \leq v_i. \tag{2}$$

By the property of Bézier curve, it is sufficient to guarantee the curve is inside the polyhedron by only constraining the control points [24].

We impose C^2 continuity constraints on neighboring segments to enforce path smoothness. This is achieved by constraining the control points. Also, the path should obey terminal constraints, i.e. $c_0^1 = q_s, c_m^K = q_g$. These linear equality constraints are in the format of:

$$A_E p = v_e. \tag{3}$$

with properly chosen A_E and v_e.

C. Time-Optimal Path Parameterization

TOPP aims to find a time parameterization to a geometric path, so that the traversal time is minimized while satisfying all constraints. Pioneered by [2], [15], it is 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* is the time-parametrized geometric path and 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 $\dot{\square}$ and $\square'$ denotes derivative with respect to time and s, respectively.

In TOPP, values of $p'(s)$ and $p''(s)$ are known and not altered 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 problem, we follow [15], [30] and discretize $s(t)$ into N segments: $\{b_i\}_{i=0}^N$. We assume that $a(s)$ is piecewise 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

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

A wide variety of constraints, i.e. d in (1) can be written as $\alpha(s)\dot{s}^2 + \beta(s)\ddot{s} \leq \gamma(s)$ after substituting Eq. (4) and have to be imposed throughout the trajectory. Due to discretization they are imposed at the mid-points between two consecutive collocation points, i.e. $\alpha(s_{i+1/2})b_{i+1/2} + \beta(s_{i+1/2})a_{i+1/2} \leq \gamma(s_{i+1/2})$. These constraints can be all expressed as linear constraints of b_i, b_{i+1}, after algebraic manipulation.

It turns out all constraints under consideration are linear in b. 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.

D. Bilevel Optimization

A bilevel optimization problem [18] is given by

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

where G_i and H_i denote outer-level constraints, f_0 is the inner-level objective, while $\{f_i(\cdot)\}_{i=1}^m$ and $\{h_i(\cdot)\}_{i=1}^p$ are inner-level constraints.

In some literature, the outer and inner level problems are called leader and follower. The follower makes action x_l by solving some parametric optimization problem based on leader's action x_u. The goal is for the leader to find x_u that optimize its own cost function which also depends on x_l. If we assume the inner 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 x_l is the solution to an optimization problem and usually has no closed-form solution. In our problem, the outer-level constraints only depend on the path itself so it suffices to find an analytic gradient of the cost function. Luckily, the inner and outer problem have the same cost function, so the outer gradient can be computed using sensitivity analysis of the inner problem.

For brevity, we give the results of first-order sensitivity analysis and refer readers to [26] for further details. Consider the problem of finding the local solution $x_l(x_u)$, i.e. the inner optimization problem as a parametric NLP problem:

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

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

$$\nabla_{x_u} f_0^\star(x_u) = \nabla_{x_u} f_0 + \sum_{i=1}^{m} \lambda_i(x_u) \nabla_{x_u} f_i + \sum_{i=1}^{p} \nu_i(x_u) \nabla_{x_u} h_i. \tag{9}$$

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

E. Interior-Point Method to Certain Duality Gap

For a convex optimization problem with linear constraints in the form of

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & Gx \leq g \\ & Hx = h \end{array} \tag{10}$$

where $f(x)$ is convex. The Karush-Kuhn-Tucker (KKT) conditions for optimality is

$$\begin{array}{l} \frac{\partial f}{\partial x} + G^T\lambda + H^T\nu = 0 \\ Gx + s = g, Hx = h \\ \lambda \geq 0, s \geq 0, \lambda \circ s = 0 \end{array} \tag{11}$$

where λ and ν are Lagrangian multipliers, s is slack variable and $\circ$ denote element-wise product. The IPM solves Eq. (10) using Newton's method with special modification to account for inequality constraints and strict complementary slackness. To progressively move towards optimality, the duality gap $\lambda^T s$ is used to control how far the next iteration is to the optimum. In fact, usually a centering step is used in order to prevent $\lambda \circ s$ from reaching 0 too quickly which slows down convergence.

Our modification of the IPM is to change the right side of the strict complementary slackness condition. Instead of solving to $\lambda \circ s = 0$, we only require IPM to solve to $\lambda \circ s = \mu$ for some value $\mu > 0$. Here, μ is a user-defined parameter for our algorithm. This modification has three positive and one negative effects. First, IPM solves toward $\lambda \circ s = 0$ iteratively from a positive initial value of $\lambda \circ s$, the value of $\lambda \circ s$ gradually converges to 0 from some positive value. As a result, early termination at $\mu > 0$ requires fewer iterations. Second, the solution is not on the boundary of nonnegative orthant and we avoid the difficulty preventing warm start. Third, no inequality constraint is active at termination, so there is no need to worry about the active set switch that causes non-smoothness of the outer gradient.

However, the drawback is that the result of optimization is suboptimal, since the KKT condition is not satisfied. The degree of suboptimality can be tuned by reducing the value of μ. Nevertheless, our experiments suggest that even a small value of μ is beneficial, and the benefits overall are worth sacrificing a small amount of optimality in the inner optimization.

Because the duality gap modifies the KKT condition, the gradient (9) obtained by sensitivity analysis is no longer valid. To obtain the new gradient, we use the equivalence between modified KKT conditions and log barrier method [31, Ch. 11]. The optimum of the barrier-augmented problem

$$\begin{array}{ll} \underset{x,s}{\text{minimize}} & f(x) - \mu \sum \log s \\ \text{subject to} & Gx + s = g, \\ & Hx = h \end{array} \tag{12}$$

satisfies the KKT condition

$$\begin{array}{l} \frac{\partial f}{\partial x} + G^T\lambda + H^T\nu = 0 \\ -\frac{\mu}{s} + \lambda = 0 \\ Gx + s = g, Hx = h \end{array} \tag{13}$$

where the Lagrangian multipliers are λ and ν. Note that the second equality is equivalent to the duality gap condition (11), and non-negative constraints $\lambda > 0$ and $s > 0$ are implicitly imposed. Hence, we apply sensitivity analysis to the log barrier problem to obtain analytic gradient. Specifically, we may reuse (9) but augment the cost function with the log barrier, and drop the inequality constraints.

The algorithm for solving the modified KKT system is basically same with standard IPM algorithm as cvxopt [30] and the only difference is Eq. (17) and Eq. (18b) of [30] are modified to account for duality gap μ. The algorithm returns multipliers λ, μ and slack variables s which are used to compute the cost with barrier as in Eq. (12) and analytic gradient as in Eq. (9). Warm starting is implemented by keeping the solution (x, s, λ, ν) from the last problem and using them as the initial guess for the next problem. It is possible that some inappropriate step is taken in outer optimization, and warm start fails to solve the inner optimization problem. In this circumstance, a regular initial guess is used.

F. Duality Gap Bilevel Optimization Algorithm

Our overall bilevel optimization method is presented in Algorithm 1. The algorithm takes an initial guess of the geometric path p_0, the linear constraints A_I, v_i, A_E, v_e on the path, and the duality gap μ. It also maintains a set of inner-optimization warm-start parameters, z_{ws}. In each iteration, the TOPP solver (Gap-TOPP) takes a path p, desired duality gap μ, and optional warm-start guess z as input, and outputs an optimized cost J, Lagrange multipliers λ and time parameterization $\{b_i\}_{i=0}^N$. The gradient of the path p is computed in the Get-Gradient subroutine using Lagrange multipliers, and later used to update the path. The outer-level optimization is essentially a nonlinear optimization with linear constraints and the optimality conditions are checked by the optimizer. At the end of the algorithm, we solve the gap-free TOPP problem to get lower cost.

Any gradient-based method can be used as the Take-A-Step function which updates p based on gradient ∇ 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 outer-level optimization are linear so feasibility is always guaranteed. In our implementation the gap parameter μ is set to a small value and fixed throughout the algorithm.

IV. Numerical Experiment

We evaluate our method on random instances of "forests" using the environment generator of Gao et al. [29]. We generated 101 tests, each with a random environment and randomly sampled feasible start points and goal points. We refer to [24] for a visualization of the environment and statistics on number of segments. In all experiments, computation is performed on a PC running Ubuntu 16.04 with 4.00 GHz CPU and 32 GB memory without parallelization. Implementations are mainly based on Python, but the IPM solver is written in C++ based on open-source cvxopt [30]. The NLP solvers are implemented

Algorithm 1 Bilevel-Solver $(p_0, A_I, v_i, A_E, v_e, \mu)$

$p \leftarrow p_0$
$z_{ws} \leftarrow nil$
for $i \leftarrow 0$ to *max-iterations* **do**
  $J, \{b_i\}_{i=0}^N, \lambda, \nu \leftarrow$ Gap-TOPP(p, μ, z_{ws})
  $z_{ws} \leftarrow (\{b_i\}_{i=0}^N, \lambda, \nu)$
  $\nabla \leftarrow$ Get-Gradient(λ, ν)
  $p \leftarrow$ Take-A-Step$(p, J, \nabla, A_I, v_i, A_E, v_e)$
  if optimality-conditions-satisfied **then**
    break
$J, \{b_i\}_{i=0}^N \leftarrow$ TOPP(p)
return $J, p, \{b_i\}_{i=0}^N$
function GAP-TOPP(p, μ, z)
  if z is not nil **then** ▷ Warm start
    $\bar{x}, \bar{\lambda}, \bar{\nu} \leftarrow z$
    if $\bar{x}$ is dynamically infeasible for p **then**
      Clear warm start
  $x, \lambda, \nu \leftarrow$ solve Eq. (13) with guess $\bar{x}, \bar{\lambda}, \bar{\nu}$
  return $J(x)$ (Eq. (12)), x, λ, ν

in compiled languages for efficiency, but they are called using cost and constraint functions written in Python.

In this problem setting, a heuristic-based graph search finds a tentative path from the start to goal. Box-type corridors are generated around the path to obtain a list of safe corridors from start to goal that overlap with neighbors. The trajectory is thus divided into several pieces with each piece constrained in one safe corridor. A demonstration of the box-type corridors is shown in Fig. 1. We set the log barrier parameter $\mu = 0.001$ in all examples, and keep it constant throughout optimization. However, after optimization terminates we re-solve with $\mu = 0$ to obtain the actual minimum time. To initialize the optimization, we use a minimum jerk trajectory computed using methods in [24] with or without time allocation refinement. The outer optimization has a iteration limit of 80 to limit computation time.

A. Gap vs Gap-free

Here we show the effect of solving our modified KKT system (Gap) by comparing with bilevel optimization where the inner optimization is solved to optimality (Gap free).

Let Problem A be an example composed of 3 boxes shown in Fig. 2. The gradient of the cost function with respect to control points is also shown for both solvers. It can be seen that two solvers may have large gradient difference on the same geometric path. Since the gap solver is smoothing the cost function, its gradient has smaller magnitude. An alternate view of this comparison is shown in Fig. 3. We start from a path that has already been refined by bilevel solver for a few iterations, with the inner optimizations performed by gap-free solver. Then we compute the gradient direction. We perturb the path slightly by taking small steps along the gradient direction, and plot the directional derivative along the gradient. This figure shows that by introducing the duality gap, a much smoother problem is obtained.

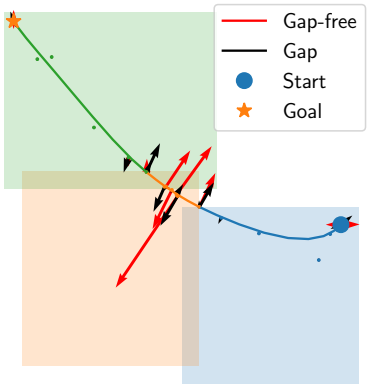


Fig. 2: Environment of problem A. The blue and orange arrow denotes gradient of control points. The gap-free problem has larger gradient.

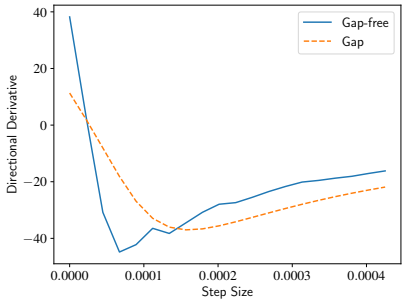


Fig. 3: Profile of directional derivatives ($\nabla f^* \cdot v / \|v\|$) when a path is moved along a direction v with different step sizes. The gap solver achieves smoother derivatives compared with the gap-free solver.

To examine how the duality gap influences the performance of outer optimization, we perform 80 outer iterations on problem A. The summary of results are shown in Tab. I. The gap solver needs fewer number of calls to the inner optimization. On average each inner optimization also requires fewer iteration steps, which is directly proportional to optimization times. As a result, the total computation time is improved. The smoother cost landscape allows more efficient cost improvement given the same amount of outer optimization.

TABLE I: Gap vs Gap-free bilevel opt., on problem A

Method	# TOPP	Avg Iter	Comp time (s)	Final cost (s)
Gap	150	8.71	0.37	2.85
Gap-free	168	20.80	0.66	2.89

We also compare their difference on all the test cases using the same amount of outer iterations. The results are shown in Tab. II and are consistent with the results observed in problem A. Since we are solving inner optimization more efficiently, the Gap method achieves lower computation time. The Cost Improvement column refers to the difference between initial traversal time and optimized traversal time, and shows that the

TABLE II: Gap vs Gap-free bilevel opt., avg. over all test cases

Method	Comp. Time (s)	Cost Improvement (s)
Gap	0.51	0.39
Gap-free	2.06	0.30

smoother cost function obtained by our method leads to faster convergence.

B. Bilevel vs Nonlinear Optimization

We compare our algorithm with two alternative optimization problem formulations, DC and Joint, both solved with SNOPT and IPOPT. Analytical gradients are provided for solver robustness and sparsity is exploited to the best of the authors' ability.

The Joint condition solves the spatial and temporal parameters jointly. The spatial trajectory is initialized directly using the result from minimum-jerk trajectory since they have the same parameterization. The temporal trajectory is initialized by solving TOPP on the initial guess. With this initialization, the initial guess is always feasible. Both SNOPT and IPOPT are both configured to have optimality tolerance of 10^{-5}. IPOPT has a maximum iteration limit of 200 while SNOPT has total iteration (including both major and minor iterations) limit of 10,000.

The DC condition formulates a nonlinear program using direct collocation. Since DC uses a different trajectory parameterization, we use interpolation on the minimum-jerk trajectory for the initial guess. The initial duration of the trajectory is scaled to satisfy velocity and acceleration bound constraints. When solved by IPOPT, the iteration limit is set as 1,000 and optimality tolerance is 10^{-3}. For SNOPT the total iteration limit is set as 50,000 and the optimality tolerance of 10^{-3} is used.

1) Success Rate: Success rate comparisons are shown in Tab. III. Rows SN and IP denotes the use of the NLP solver SNOPT and IPOPT, respectively, and columns No Ref and Ref denotes whether time allocation is used to refine the minimum jerk trajectory using methods in [24].

The bilevel optimization framework maintains inner optimization feasibility by construction and guarantees a feasible solution at any time as long as the initial guess is feasible. As a result, its success rate is 100% as long as IPM is robust. Both Joint and DC rely on an NLP solver and cannot guarantee a feasible solution. In practice, both Joint and DC may end up at an infeasible solution even if the initial guess is feasible. This indicates the strong nonlinearity of time-optimal problem and NLP solver may fail unless a solution close to optimum is provided. Also, it can be seen that SNOPT is better at Joint optimization and IPOPT excels in DC, and providing a refined minimum-jerk trajectory gives higher success rate.

2) Cost and Computation Time: Tab. IV compares our method in terms of the optimized cost and computation time. For DC and Joint, we restrict our comparison to the variants that use the refined initial guess, IPOPT for DC, and SNOPT for Joint. For DC and Joint, we average only the successful results, since the cost of infeasible solution cannot be sensibly measured. The problems that DC and Joint successfully solve are compared with the cost of Bilevel which tends to achieve lower cost. It turns out Bilevel outperforms DC and Joint optimization in terms of both cost and computation time.

TABLE III: Success rate of planning methods

	DC		Joint		Bilevel	
	No Ref	Ref	No Ref	Ref	No Ref	Ref
SN	78.2%	88.1%	69.3%	93.1%	100%	100%
IP	85.1%	93.1%	20.8%	23.8%		

TABLE IV: Optimized costs and computation times, by problem set ("Set X" indicates those problems on which X succeeds)

Method	DC	Joint	Bilevel
Avg Cost on All			10.03
Avg Cost on Set DC	10.2		9.98
Avg Cost on Set Joint		10.71	10.35
Avg Comp. Time (s)	6.42	0.62	0.51

It is unclear why Joint has such a success rate but worse cost function, since successful termination requires a feasible solution that satisfies KKT conditions for optimality. Since joint optimization and bilevel optimization are initialized using the same guess, this might be caused by the NLP solver converging to a local optimum.

V. Conclusion

We present a bilevel optimization framework to solve time-optimal problems with spatial and temporal constraints. We exploit the convexity of TOPP and linearity of spatial constraints to achieve an any-time and highly robust trajectory optimizer. We establish the equivalence of modified KKT condition and log barrier methods. This helps to achieve a highly efficient IPM with warm start and smoother cost function. Numerical experiments show that trajectory optimization on flying robots are solved reliably, while standard NLP solvers fail to maintain feasibility of the solution, take a longer time, and obtain higher cost solutions.

Future work should address in the relation between optimizing log barrier cost and the original cost. The problem of choosing the duality gap and necessity of adjusting it during optimization also requires further investigation. We are also interested in replacing the Bézier curve representation due to its potential conservativeness. Including other constraints such as motor actuation is also a promising direction to extend its area of applications. We are also interested in optimizing types of cost function in the spatial-temporal decomposition framework.

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] G. Tang, W. Sun, and K. Hauser, "Time-optimal trajectory generation for dynamic vehicles: A bileveloptimization approach," in *Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference on*. IEEE, 2019, pp. 0–7.

[5] 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

[6] 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

[7] 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

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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

[14] 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.

[15] 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

[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] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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

[24] 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

[25] 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

[26] 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

[27] A. Shaban, C.-A. Cheng, N. Hatch, and B. Boots, "Truncated back-propagation for bilevel optimization," *arXiv preprint arXiv:1810.10667*, 2018.

[28] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 136–145.

[29] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 344–351.

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

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