2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Generalizing Trajectory Retiming to Quadratic Objective Functions

Gerry Chen, Frank Dellaert, and Seth Hutchinson

***Abstract*— Trajectory retiming is the task of computing a feasible time parameterization to traverse a path. It is commonly used in the decoupled approach to trajectory optimization whereby a path is first found, then a retiming algorithm computes a speed profile that satisfies kino-dynamic and other constraints. While trajectory retiming is most often formulated with the minimum-time objective (i.e. traverse the path as fast as possible), it is not always the most desirable objective, particularly when we seek to balance multiple objectives or when bang-bang control is unsuitable. In this paper, we present a novel algorithm based on factor graph variable elimination that can solve for the global optimum of the retiming problem with *quadratic* objectives as well (e.g. minimize control effort or match a nominal speed by minimizing squared error), which may extend to arbitrary objectives with iteration. Our work extends prior works, which find only solutions on the boundary of the feasible region, while maintaining the same linear time complexity from a single forward-backward pass. We experimentally demonstrate that (1) we achieve better real-world robot performance by using quadratic objectives in place of the minimum-time objective, and (2) our implementation is comparable or faster than state-of-the-art retiming algorithms.**

I. Introduction

Trajectory optimization is necessary to execute safe, effective robot motions and can be solved either directly as a state-space optimization problem or with a decoupled approach in which a configuration-space path is first found then retimed [1, Ch. 11.2]. We focus on the decoupled approach to separate the trajectory optimization problem into a path planning problem and a retiming problem. In this approach, a path planner first generates a path which e.g. avoids obstacles and solves the robot kinematics. Then, a retiming algorithm is applied to the path to generate a trajectory which satisfies the robot's dynamics and other constraints. Although the resulting solution is only an approximation to the original trajectory optimization problem, it is often more tractable, faster, and reliable, especially when the path planner has good heuristics for kino-dynamic constraints or the task specification defines the path.

The *time-optimal* trajectory retiming problem (also known as time-optimal path parameterization (TOPP), time-optimal path tracking, and several other names), in which the objective is to minimize the time to traverse the path, represents the most common retiming objective. This problem has been studied extensively in the literature and traditionally has taken one of three approaches [2]: convex optimization, dynamic programming, and searching for bang-bang control switching points where the active constraints change.

This work was supported by NSF Grant No. 2008302.

All authors are with the Institute for Robotics and Intelligent Machines (IRIM), Georgia Institute of Technology, Atlanta, GA 30332, USA {gchen,fd27,seth}@gatech.edu

Code available: https://github.com/gchenfc/QOPP-TrajectoryRetiming

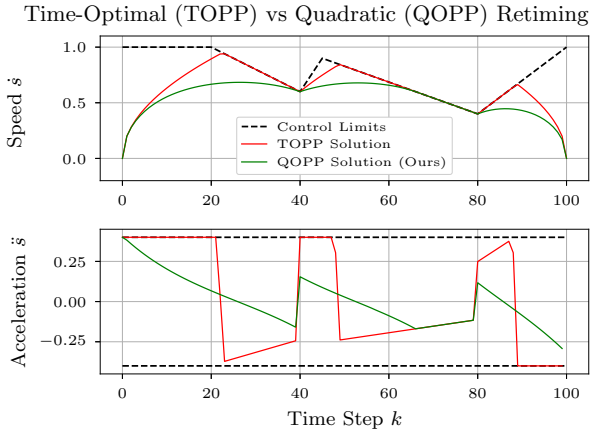


Fig. 1. Using quadratic objectives (green) in place of the minimum-time objective (red) results in less aggressive maneuvers and a more consistent speed profile. Our approach may increase safety margins, improve tracking accuracy, and reduce premature wear by balancing objectives like execution speed and motor torques.

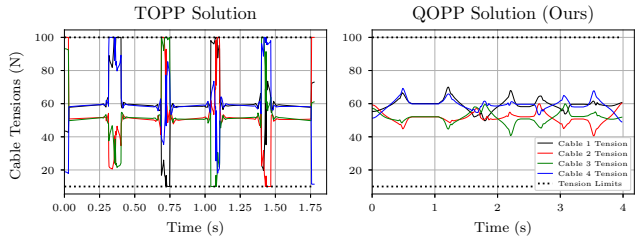


Fig. 2. Open-loop cable robot tensions for the TOPP trajectory frequently hit control limits, losing stiffness and stability. Meanwhile, quadratic objectives (QOPP) enable a tunable tradeoff between duration and safety margin.

Two key precursor techniques enabling efficient algorithms are shared by almost all approaches, dating back to at least [3]–[7]. First, the equations of motions and constraints are reparameterized in terms of the *scalar* time parameterization function (see Sec. II-A for details). This enables Bellman-style forward-backward algorithms. Second, the fact that the time-optimal objective is monotonic implies that the solution must lie on the boundary of the feasible set (bang-bang). Hauser [8], Nagy & Vajk [2], and Pham et. al. [2] all utilize proofs along these lines to justify the use of sequential linear programming (SLP) or greedy speed maximization during the backward pass. This bang-bang approach is efficient but restricts the objective to minimum-time or similar objectives.

However, the *time-optimal* retiming problem is not always the most desirable objective [9], particularly in applications where we seek to balance multiple objectives or where bang-bang control is unsuitable. For example, cable-driven parallel robots maintain stiffness primarily through internal

tension which diminishes the closer they are to torque limits. Thus balancing speed with torque margin is desirable to maintain stability and safety [10], [11]. This and several other applications in balancing robot safety, stability, and wear with speed of operation motivate the use of quadratic objectives, which can minimize the sum of multiple squared errors instead of or in addition to hard constraints with TOPP.

Surprisingly alternate retiming objectives are rarely considered in the literature. Dynamic programming approaches [5], [7], [12] discretize not only in time but also in state space. Some approaches address other objectives, especially energy-minimization [13], [14], but apply only to specific objectives, e.g. integral of a time-independent running cost. Direct transcription approaches tend to be the most general [9], [15] but, even with second-order cone problem or sparse linear algebra solvers, do not fully exploit the structure of this scalar-function optimal control problem.

In this paper, we present a novel, linear-time algorithm for solving *Quadratic* Objective Path Parameterization (QOPP). Our algorithm matches the performance of TOPP-RA [16] on TOPP problems while also solving QOPP problems in linear time. Our contributions are as follows:

1) We share a **re-interpretation** of the TOPP-RA [16] approach as factor graph variable elimination.
2) We **extend** the TOPP-RA approach to solve the trajectory retiming problem with **quadratic objectives**.
3) We describe how the quadratic case can be extended to arbitrary **nonlinear objectives** with iteration.
4) We write and benchmark a fast, C++ **implementation**.
5) We validate the utility of objectives other than minimum time with real-world robot **experiments**.

II. Approach

In this section, we start by providing a brief recap of the standard reparameterizations used in trajectory retiming. We then introduce factor graphs by explaining the TOPP-RA (*R*eachability *A*nalysis) algorithm using a factor graph interpretation. Finally, we use the factor graph approach to extend the TOPP-RA algorithm to the QOPP problem.

A. Reparameterization

We define the TOPP problem as follows. Given a path $\boldsymbol{q}(s) : [0,1] \to \mathbb{R}^n$, we seek to find a reparameterization, $s(t)$ is a monotonically increasing function from $[0,T] \to [0,1]$, which minimizes the total time T while satisfying a set of general first and second order constraints:

$$s^*(t) = \underset{s(t)}{\arg\min} \quad T \tag{1a}$$

$$\text{subject to}$$

$$\boldsymbol{A}(s)\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T\boldsymbol{B}(s)\dot{\boldsymbol{q}} + \boldsymbol{f}(s) \in \mathscr{C}(s), \tag{1b}$$

$$\boldsymbol{A}^v(s)\dot{\boldsymbol{q}} + \boldsymbol{f}^v(s) \in \mathscr{C}^v(s), \tag{1c}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{f}$ denote coefficient matrix-, tensor-, and vector- valued functions for general second order constraints; $\boldsymbol{A}^v, \boldsymbol{f}^v$ denote coefficient matrix- and vector- valued functions for general first-order constraints; and $\mathscr{C}, \mathscr{C}^v$ denote convex polytope-valued functions of admissible values for the corresponding constraints. For notational convenience, we will assume these to be closed polytopes (so we can use $\leq$), but everything applies to open intervals as well. Arguments for $\boldsymbol{q}(s(t)), \dot{\boldsymbol{q}}(s(t)), \ddot{\boldsymbol{q}}(s(t))$, and $s(t)$ in (1b) and (1c) were omitted for readability. Equations (1b) and (1c) apply to $t \in [0,T]$ (omitted for readability). Although rigid-body equations of motion are most commonly notated with $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ instead of $\dot{\boldsymbol{q}}^T\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}$, the latter is also standard [1], [17] and valid for conservative systems.

1) Re-writing Constraints in Terms of $s(t)$: Differentiating $\boldsymbol{q}(s(t))$ with respect to t yields

$$\dot{\boldsymbol{q}}(s(t)) = \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}s}\dot{s} \tag{2}$$

$$\ddot{\boldsymbol{q}}(s(t)) = \frac{\mathrm{d}^2\boldsymbol{q}}{\mathrm{d}s^2}\dot{s}^2 + \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}s}\ddot{s}. \tag{3}$$

Substituting into (1b) and (1c) yields constraints of the form:

$$\boldsymbol{a}(s)\ddot{s} + \boldsymbol{b}(s)\dot{s}^2 + \boldsymbol{c}(s) \in \mathscr{C}(s) \tag{4}$$

$$\boldsymbol{a}^v(s)\dot{s} + \boldsymbol{c}^v(s) \in \mathscr{C}^v(s), \tag{5}$$

where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{a}^v, \boldsymbol{c}^v$ are vector functions of s.

(5) can always be rewritten in the form of (4) since $\dot{s}$ is a scalar, positive function. Thus (5) can be written $\dot{s}_{min} \leq \dot{s} \leq \dot{s}_{max}$ and squared. Our TOPP problem is now:

$$s^*(t) = \underset{s(t)}{\arg\min} \quad T \tag{6a}$$

$$\text{subject to} \quad \boldsymbol{a}(s)\ddot{s} + \boldsymbol{b}(s)\dot{s}^2 + \boldsymbol{c}(s) \in \mathscr{C}(s). \tag{6b}$$

2) Discretization: As is common for TOPP convex optimization and dynamic programming approaches (but not all switching-search approaches), we discretize in s:

$$a_k\ddot{s}_k + b_k\dot{s}_k^2 + c_k \in \mathscr{C}_k. \tag{7}$$

Defining:

$$x := \dot{s}^2, \quad u := \ddot{s}, \tag{8}$$

we can rewrite (6b) as

$$\boldsymbol{a}_k u_k + \boldsymbol{b}_k x_k + \boldsymbol{c}_k \in \mathscr{C}_k \tag{9}$$

and, remarkably, $\frac{\mathrm{d}x}{\mathrm{d}s} = 2\dot{s}\frac{\mathrm{d}\dot{s}}{\mathrm{d}s} = 2\frac{\mathrm{d}\dot{s}}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}t} = 2\ddot{s} = 2u$. Then assuming a zero-order hold on u (piecewise constant between intervals), we introduce another constraint from the relationship between x and u:

$$x_{i+1} = x_k + 2u_k\Delta_s. \tag{10}$$

The minimum-time objective can also be discretized:

$$T = \int_0^1 \frac{1}{\dot{s}(t)}ds = \sum_{k=0}^{N-1} \frac{2\Delta_s}{\sqrt{x_k} + \sqrt{x_{k+1}}}, \tag{11}$$

where the summation holds exactly for the piecewise-constant assumption on u [18, Sec 6.1.1].

Finally, the objective $\min \sum T$ can be achieved by greedily selecting the maximum $\dot{s}_k$ from the set of reachable values at each time step, as long as Δ_s is sufficiently small [16]. Notationally, we can express this "greedy" selection optimization problem as:

$$\underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}}{\text{maximize}} \quad \sum_{i=0}^{N} w^k x_k \tag{12a}$$

subject to

$$\boldsymbol{a}_k u_k + \boldsymbol{b}_k x_k + \boldsymbol{c}_k \in \mathscr{C}_k, \quad i = 0, \dots, N, \tag{12b}$$

$$x_{i+1} - x_k - 2u_k\Delta_s = 0 \ , \quad i = 0, \dots, N-1, \tag{12c}$$

$$x_k > 0 \ , \quad i = 0, \dots, N \tag{12d}$$

for some very large w, which denotes that each x_k should be taken greedily and irrespective of any other x_k. Define $u_N := 0$ for notational simplicity of (12b).

Intuitively, the equivalence of the optimization problems despite a different objective function is due to the facts that (a) if a larger x_k never sacrifices x_{k+1} to be smaller, then the greedy approach works, and (b) there exists some critical threshold Δ_s^* such that, when $\Delta_s \leq \Delta_s^*$, the former is true for all k. Specifically, the former is true when the dynamics coefficient $2\Delta_s$ is smaller than $\boldsymbol{a}_k/\boldsymbol{b}_k$ for all k. Though the details of the proof require additional machinery to address the fact that $\boldsymbol{a}_k/\boldsymbol{b}_k$ is not defined for vectors, the intuition is the same and given in [16].

Although we could remove u_k from the optimization problem (12) by substituting $u_k = (x_{i+1} - x_k)/2\Delta_s$ from (12c), we opt not to because (1) it will make min-effort objectives more natural when we extend to quadratic objectives and (2) the factor graph elimination will do this automatically.

B. Solving the TOPP Problem with Factor Graphs

Although [16] solves the LP in $\mathcal{O}(N)$ time using a reachability analysis approach, we can instead use factor graph variable elimination to derive an equivalent algorithm. We hope to be clear enough that factor graph elimination is intuitive to readers, but we provide a brief introduction in the Appendix for those who feel more comfortable with one.

The Factor Graph for TOPP: The factor graph for this problem is given in Fig. 3. We will proceed eliminating one variable at a time in the order $u_0, x_0, u_1, x_1, \dots, x_{N-1}, x_N$.

Eliminating u_0: Following the reachable set elimination ordering, we first eliminate u_0 by solving the LP derived by collecting only the terms in (12) that contain u_0:

$$\begin{aligned} u_0^*(x_0, x_1) = \ \underset{u_0}{\arg\max} \quad & \text{(nothing)} \\ \text{subject to} \quad & \boldsymbol{a}_0 u_0 + \boldsymbol{b}_0 x_0 + \boldsymbol{c}_0 \in \mathscr{C}_0, \\ & x_1 - x_0 - 2u_0\Delta_s = 0. \end{aligned}$$

In this case, although we have no objective function, the solution is obvious because the dynamics fully constrain u_0:

$$u_0^*(x_0, x_1) = \frac{1}{2\Delta_s}(x_1 - x_0). \tag{15}$$

This optimal assignment is called a *conditional* (this would be $p(u_0|x_0, x_1)$ in PGM literature) and is denoted by arrows in Fig. 4. We then substitute $u_0^*(x_0, x_1)$ into (12b) to create a new factor on the separator $\mathcal{S}(u_0) = \{x_0, x_1\}$:

$$\boldsymbol{a}_0 \left((x_1 - x_0)/(2\Delta_s)\right) + \boldsymbol{b}_0 x_0 + \boldsymbol{c}_0 \in \mathscr{C}_0. \tag{16}$$

After eliminating u_0, our factor graph looks like Fig. 4.

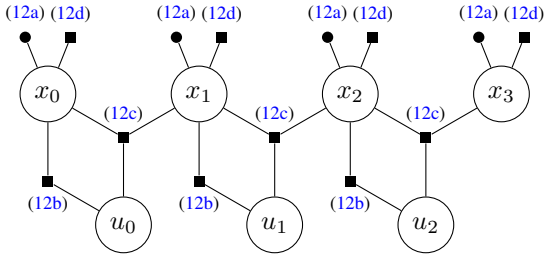
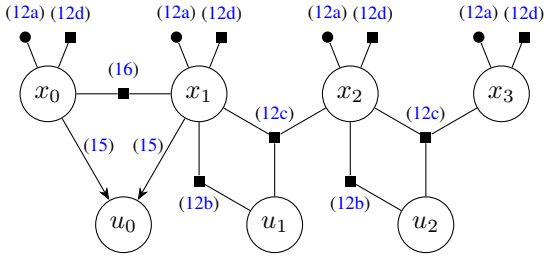


Fig. 3. Factor graph graphically representing a 4-timestep instance of the TOPP problem (12). Each variable node represents a variable x_k or u_k in the LP. Each factor node represents a constraint (square) or objective (dot).



$$\underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}}{\text{maximize}} \quad \text{(12a)}$$

$$\text{subject to} \quad \text{(16)}, \text{(12b)}, \text{(12c)}, \text{(12d)}$$

Fig. 4. Factor graph (top) and the equivalent optimization problem (bottom) after eliminating u_0. The arrows denote a *conditional* $u_0^*(x_0, x_1)$ and the new factor (16) is the result of substituting $u_0^*(x_0, x_1)$ into (12b).

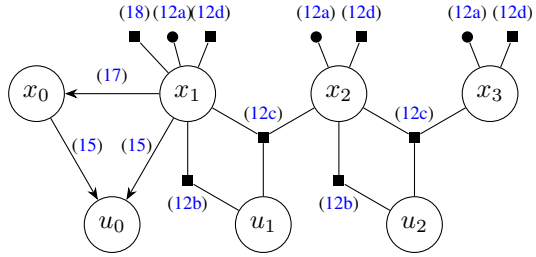
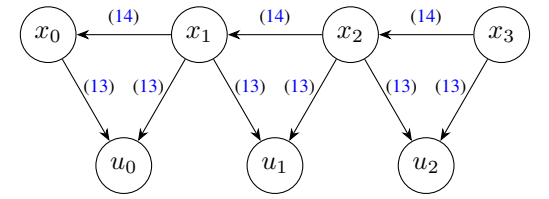


Fig. 5. Factor graph after eliminating u_0, x_0. The arrow (17) represents a 1-d LP we will backsubstitute at the end, and the new factor (18) denotes the scalar inequality constraint propagated to x_1 after eliminating x_0.



$$u_k^*(x_k, x_{k+1}) = \frac{1}{2\Delta_s}(x_{k+1} - x_k). \tag{13}$$

$$\left.\begin{aligned} x_k^*(x_{k+1}) = \ \max_{x_k} \quad & x_k \\ \text{s.t.} \quad & \frac{\boldsymbol{a}_k}{2\Delta_s}(x_{k+1} - x_k) + \boldsymbol{b}_k x_k + \boldsymbol{c}_k \in \mathscr{C}_k, \\ & x_{k,min} \leq x_k \leq x_{k,max}, \\ & x_k > 0. \end{aligned}\right\} \tag{14}$$

Fig. 6. After eliminating all the variables, we obtain a *Bayes Net*. Arrows denote conditionals (13)(14) which we can efficiently back-substitute.

Eliminating x_0: Next, eliminate x_0 much the same way:

$$x_0^*(x_1) = \max_{x_0} \quad x_0 \tag{17a}$$

$$\text{s.t.} \quad \frac{\boldsymbol{a}_0}{2\Delta_s}(x_1 - x_0) + \boldsymbol{b}_0 x_0 + \boldsymbol{c}_0 \in \mathscr{C}_0, \tag{17b}$$

$$x_0 > 0. \tag{17c}$$

For the purposes of variable elimination, we don't need to symbolically solve this LP, but instead we just need 2 things:

1) conditional: $x_0^*(x_1)$, and
2) new factor: the resulting objectives/constraints on x_1 after we substitute $x_0 = x_0^*(x_1)$.

For 1, we do not need an analytical expression (yet) so we just store the conditional as the optimization problem (17).

For 2, the new factor will consist of an objective component and a constraint component.

The objective component is easy: we can ignore it because we will select x_1 greedily. More formally, our new objective factor will be $x_0^*(x_1)$, but because we also have a pre-existing factor wx_1 where w is a very large number, our new objective term is negligible in comparison ($x_0^*(x_1) \ll wx_1$).

The constraint component of our new factor can be solved with 2 LPs the same way as in [16]. Since x_1 is a scalar, the resulting constraint on x_1 will take the form:

$$x_{1,min} \leq x_1 \leq x_{1,max}. \tag{18}$$

We compute the smallest and largest possible values of x_1 that satisfy (17b), (17c):

$$x_{1,min} = \underset{x_0,x_1}{\text{minimize}} \quad x_1 \quad \text{subject to (17b), (17c)}, \tag{19}$$

$$x_{1,max} = \underset{x_0,x_1}{\text{maximize}} \quad x_1 \quad \text{subject to (17b), (17c)}. \tag{20}$$

These are very easy to solve in just a few dozen lines of code since we need only optimize over 2, scalar variables.

After eliminating u_0, x_0 we have Fig. 5. We repeat the elimination process on $u_1, x_1, \ldots, x_{N-1}, x_N$ until all variables are eliminated. The result is the *Bayes Net* in Fig. 6.

Back-substitution: We solve the Bayes Net by back-substitution. The final elimination step will produce a *marginal* on x_N ($p(x_N)$ in PGM literature) of the form:

$$\begin{aligned} x_N^* = \max_{x_N} \quad & x_N \\ \text{s.t.} \quad & x_{N,min} \leq x_N \leq x_{N,max}, \\ & x_N > 0, \end{aligned} \tag{21}$$

whose solution is clearly $x_N^* = x_{N,max}$.

Then, we can compute $x_{N-1}^* = x_{N-1}^*(x_N)$ by substituting $x_N \leftarrow x_N^*$ into the conditional (14) and solving the now single-variable scalar LP (which is just iterating through the inequalities to find the lower bound) for x_{N-1}. This process is repeated until all variables are evaluated, and the resulting sequence $x_0^*, \ldots, x_N^*, u_0^*, \ldots, u_{N-1}^*$ is the solution to (12).

Finally, the optimal time parameterization $s^*(t)$ can be obtained by integrating $\dot{x}^* = \dot{s}^*$. We defer to [16] for the intracacies of parameterizing solution. As in [16], zero-inertia points are accurate in the limit $\Delta_s \to 0$. The time optimal trajectory is $\boldsymbol{q}^*(t) = \boldsymbol{q}(s^*(t))$.

C. Solving the QOPP Problem with Factor Graphs

The variable elimination algorithm naturally extends to other objectives because it remains unchanged no matter the objectives or constraints; only the algebra of each elimination step changes. Let us then define our (discretized) general quadratic objective problem as:

$$\underset{\substack{x_0,\ldots,x_N,\\ u_0,\ldots,u_{N-1}}}{\arg\min} \quad \sum_{i=0}^{N} \tilde{x}_k Q_k \tilde{x}_k + \tilde{u}_k R_k \tilde{u}_k + \tilde{x}_k N_k \tilde{u}_k \tag{22a}$$

$$\text{subject to} \quad \text{(12b), (12c), (12d)}, \tag{22b}$$

where *scalars* $\tilde{x}_k := x_k - x_{k,desired}$, $\tilde{u}_k := u_k - u_{k,desired}$, and Q_k, R_k, N_k are state, control, and cross cost weights.

Elimination of u_k is identical to the TOPP case; see (13). Similarly, the inequality constraint portion of the new factor when eliminating x_k is identical to the TOPP case; see (18)–(20). Then, the new objective portion of the new factor when eliminating x_k is the only portion that changes (which requires analytically computing the conditional as well).

Let us begin by computing the conditional $\tilde{x}_0^*(\tilde{x}_1)$:

$$\tilde{x}_0^*(\tilde{x}_1) = \underset{\tilde{x}_0}{\arg\min} \quad \tilde{x}_0 Q_0' \tilde{x}_0 + \tilde{x}_1 R_0' \tilde{x}_1 + \tilde{x}_0 N_0' \tilde{x}_1 \tag{23a}$$

$$\text{s.t.} \quad \frac{\boldsymbol{a}_0}{2\Delta_s}(x_1 - x_0) + \boldsymbol{b}_0 x_0 + \boldsymbol{c}_0 \in \mathscr{C}_0, \tag{23b}$$

$$x_0 > 0. \tag{23c}$$

where Q_0', R_0', N_0' were derived after eliminating $\tilde{u}_0$ by substituting $\tilde{u}_0^* = \frac{1}{2\Delta_s}(x_1 - x_0)$:

$$Q_0' := Q_0 + \frac{1}{4\Delta_s^2} R_0 - \frac{1}{2\Delta_s} N_0 \tag{24}$$

$$R_0' := \frac{1}{4\Delta_s^2} R_0 \tag{25}$$

$$N_0' := -\frac{1}{2\Delta_s^2} R_0 + \frac{1}{2\Delta_s} N_0. \tag{26}$$

In contrast to the TOPP case where we could use the bang-bang property to delay solving of the LP until back-substitution, we must actually solve this QP symbolically as a function of x_1. Fortunately, this is tractable for a 2-dimensional (2 scalar variables) QP.

The solution to (23) is well-known to be piecewise linear. As an intuition, without inequality constraints this is just the minimum of a scalar quadratic function:

$$\tilde{x}_{0,unconstr}^*(\tilde{x}_1) = -b/(2a) \tag{27}$$

$$= -\frac{N_0'}{2Q_0'} \tilde{x}_1. \tag{28}$$

After inclusion of inequality constraints, the solution will be unchanged when no constraints are violated, but be the convex feasible boundary when any are violated.

We then substitute the piecewise linear $\tilde{x}_0^*(\tilde{x}_1)$ into (23a) to obtain a piecewise quadratic new objective on $\tilde{x}_1$. When eliminating $\tilde{x}_1$, the problem will have a similar form as (23) but with a piecewise quadratic in place of (23a). Nevertheless, $\tilde{x}_1^*(\tilde{x}_2)$ will still result be piecewise linear solution as is a well-known result in parametric QP literature.

While the computational complexity of variable elimination for TOPP is exactly identical to [16] (3 trivial LPs for

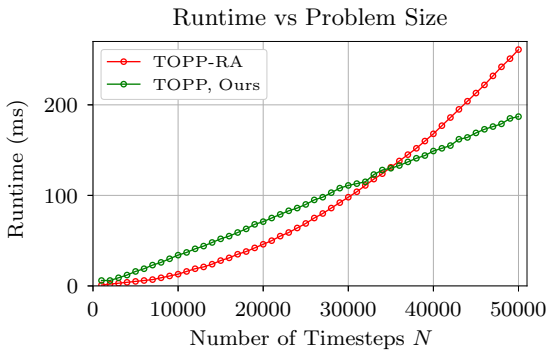


Fig. 7. Computation time for a sample TOPP problem with varying problem size shows (1) our algorithm is linear-time and (2) is faster than TOPP-RA. Both algorithms are implementationed in C++, compiled with identical compiler flags `-O3;-flto`, and run on the same M1 Macbook Air.

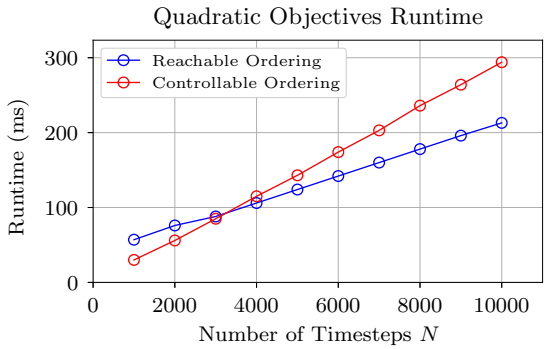


Fig. 8. Runtime plot for a sample quadratic objective retiming problem shows that our algorithm is still $\mathcal{O}(n)$, even with quadratic objectives.

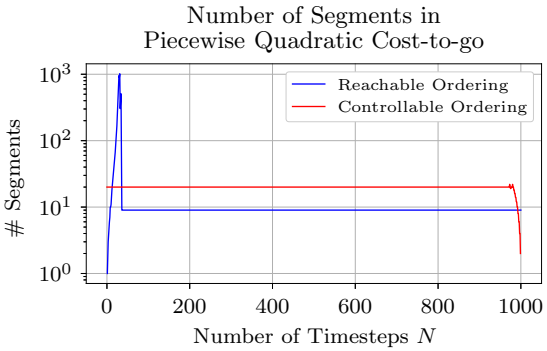


Fig. 9. The number of segments in the piecewise quadratic cost-to-go functions for each time step remains mostly constant throughout the optimization. The rapid growth followed by decay for the reachable ordering explains the apparent non-zero y-intercept in Fig. 8.

each timestep), the worst-case computational complexity for quadratic objectives will be worse because the number of segments in the piecewise quadratic objectives when eliminating $\tilde{x}_k$ could be at most $\sum_{j=0}^{k-1} \mathcal{I}_j + 1$ where $\mathcal{I}_j$ denotes the number of inequality constraints on $\tilde{x}_j$ in the original problem. In practice, however, we found that the number of segments in each piecewise quadratic objective typically did not grow with problem size thus maintaining linear time complexity with respect to the number of timesteps.

III. Experimental Results

To experimentally validate our algorithm, we implement a C++ version of our algorithm [19] to analyze runtime then apply it to a cable robot to demonstrate the efficacy of quadratic objectives over the time-optimal objective.

A. Runtime

To validate the runtime and complexity of our algorithm, we implement a C++ version of our algorithm and compare it to the state-of-the-art TOPP-RA algorithm [20] on a sample problem with varying problem size.

As shown in Figure 7, our implementation is linear-time and has similar speed to TOPP-RA. We test a simple problem: $x_{k+1} = x_k + (0.5)u_k$ with constraint $x_k + u_k \leq 0.1$. The results are identical to 7 decimal places. Although TOPP-RA appears to have a super-linear runtime, in most applications a few hundred timesteps is sufficient so sub-millisecond timing is expected from both algorithms.

Figure 8 shows that the runtime, even with quadratic objectives, remains linear with problem size. Figure 9 further evidences this fact by showing that the number of inequalities carried forward from each time step remains roughly constant throughout the optimization. Objectives $x_N^2 + \sum x_k^2 + u_k^2$ were added to the problem above to generate these results. As discussed in Section II-C, completely general quadratic objectives are supported.

B. Cable-Driven Parallel Robot Application

Operating at control limits for prolonged periods is often undesirable for reasons such as safety, wear and tear, noise, and energy consumption. In such cases, it is preferable to balance multiple objectives such as matching a desired speed while also minimizing control effort.

This is especially true for cable robots, which have inherently low stiffness maintained primarily through internal tension. Approaching control limits necessarily diminishes the possible amount of internal tension causing reduced stability and increased risk of cable slackening. In this section, we formulate the a quadratic objective retiming problem for a cable robot path tracking problem, then quantitatively and qualitatively show results running on a real robot.

1) Formulating the Cable Robot Tracking Problem: Due to the nature of cable-driven parallel robots, it makes most sense to define the path tracking problem in the task space rather than the joint space. Just as serial manipulators can express task space constraints in the joint space via the Jacobian (but the other direction is more difficult), cable robots can express joint space constraints in the task space via the wrench matrix, $\boldsymbol{W}$: defined by $\boldsymbol{F} = \boldsymbol{W}\boldsymbol{t}$ implying $\dot{\boldsymbol{q}} = \boldsymbol{W}^T\dot{\boldsymbol{x}}$, where $\boldsymbol{F}, \boldsymbol{W}, \boldsymbol{t}, \dot{\boldsymbol{x}}$ denote the force on the end-effector, wrench matrix, cable tensions, and task space velocity, respectively. Computing feasible polytopes $\mathscr{C}(s) := \{\boldsymbol{F} : \exists \boldsymbol{t} \in [\boldsymbol{t}^-, \boldsymbol{t}^+] \text{ s.t. } \boldsymbol{F} = \boldsymbol{W}(s)\boldsymbol{t}\}$, we define then our standard dynamics and constraints in the form (note: vector

$\boldsymbol{x}$ is distinct from scalar x):

$$\boldsymbol{A}(s)\ddot{\boldsymbol{x}} + \dot{\boldsymbol{x}}^T \boldsymbol{B}(s)\dot{\boldsymbol{x}} + \boldsymbol{f}(s) \in \mathscr{C}(s) \tag{29a}$$

$$\boldsymbol{A}^v(s)\dot{\boldsymbol{x}} + \boldsymbol{f}^v(s) \in \mathscr{C}^v(s). \tag{29b}$$

We then apply the process in Section II-A to convert to (7).

For objectives, we seek to match a setpoint speed using objective $q' \left\| \dot{\boldsymbol{x}}^T \dot{\boldsymbol{x}} - v_d^2 \right\|^2$ and maximize motor torque margin using objective $r \left\| \boldsymbol{F} - \boldsymbol{W}\boldsymbol{t}_m \right\|^2$, where q', r are weighting factors, v_d is the desired speed, and $\boldsymbol{t}_m$ is the arithmetic mean of the minimum and maximum allowable tensions, similar to [10], [21]. These can be expressed as:

$$q \left\| x - x_d \right\|^2 + r \left\| \boldsymbol{A}u + \boldsymbol{B}x + \boldsymbol{C}' \right\|^2, \tag{30}$$

with $q := q' \left\| \boldsymbol{x}' \right\|^4$, $x_d := v_d^2 / \left\| \boldsymbol{x}' \right\|^2$, $\boldsymbol{x}'$ denotes the derivative of $\boldsymbol{x}$ with respect to s, and $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}'$ are the same coefficients as in the equations of motion except subtracting off $\boldsymbol{W}\boldsymbol{t}_m$. These can, in turn, be combined to form a single quadratic objective in the form $Q_k \tilde{x}_k^2 + R_k \tilde{u}_k^2 + N_k \tilde{x}_k \tilde{u}_k$, computed by software. Note we still keep (29) for safety.

C. Robot behavior with QOPP vs TOPP

We solve the cable robot path tracking problem for a 2m/s star-shaped trajectory with 1000 discretization points. Figs. 2 and 10 show the solution and execution result, respectively. Tracking error measured with OptiTrack Motion Capture.

IV. Discussion

By extending the trajectory retiming problem to a wider domain, we potentially open opportunities for sharing techniques with the broader trajectory optimization community.

One such opportunity is to generalize to arbitrary nonlinear objectives by employing an SQP strategy in which a nonlinear optimization problem is repeatedly approximated as a QP problem over a step direction [15]. This linear-time algorithm is specific to the scalar structure of the problem, so specialized trajectory retiming solvers may outperform general solvers even for complicated objectives.

Another opportunity is to perform trajectory optimization with alternating path and retiming optimizations. Whereas time-optimal retiming would not apply due to differing objectives, introducing quadratic objectives brings us closer to sharing the same objectives, which would enable semi-decoupled approaches. GTDynamics also uses GTSAM and uses a time-scaling variable. Experiments in variable ordering have suggested such an approach may be promising [22].

V. Conclusions and Future Works

In this work, we proposed a linear time algorithm for solving trajectory retiming problems with quadratic objectives (QOPP). We discussed some applications of quadratic objectives, such as balancing objectives of fast speed and keeping distance from control limits for improving safety and robot wear. We described the algorithm by first reinterpreting TOPP-RA using factor graphs, then extending the factor graph algorithm to handle quadratic objectives. Finally, we experimentally validated our algorithm's runtime and performance improvement on a cable robot path tracking problem.

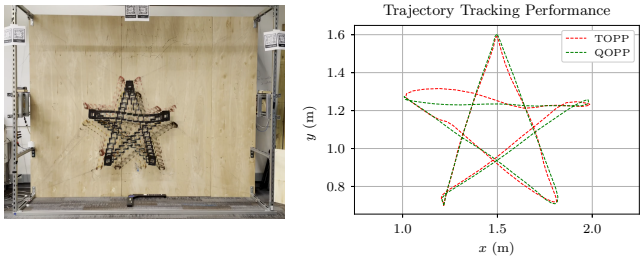


Fig. 10. TOPP (red) and QOPP (ours, green) used for a cable robot path following task show that TOPP generates significantly more error (62mm max tracking error) than QOPP (42mm max tracking error) due to saturating controls. We combine video frames to visualize the robot moving (left).

APPENDIX

In this section we provide a brief introduction to factor graph variable elimination for optimal control.

For readers new to Factor Graph Variable Elimination:

Think of factor graph variable elimination as a graphical representation for the process of solving optimization problems "one variable at a time". The optimal value for that variable is given as a function of the other variables still in the problem, and a new optimization problem is created by substituting the optimal value for that variable into the original problem. A key detail is that only the variables that share a factor with the eliminated variable are involved in the elimination process, thereby *exploiting the sparsity of the problem automatically*. This process is repeated until only one variable remains, at which point the solution is returned.

More formally, an optimization problem can be described with a factor graph by denoting each of the variables to be optimized as a "variable" node and each of the optimization objective terms and constraints as a "factor" node, where an edge connects each variable a factor depends on (see Fig. 3 as an example). Then to solve the optimization problem, the variable elimination algorithm "eliminates" (solves) one variable, x, at a time, passing its constraints and objective terms as a new factor on the *separator*, $\mathcal{S}(x)$: the set of variables sharing a factor with the eliminated variable. More complete descriptions of factor graph elimination for solving optimal control problems can be found in [10], [23], [24].

GTSAM [25] is a mature C++ software library that implements factor graph variable elimination, including with quadratic objectives and linear equality constraints. Architecturally, it allows easy extension to handle additional factor types, such as inequality constraints, which we do later in this paper. Clickable link back to II-B.

For readers familiar with factor graphs (especially GTSAM):

Consider that a factor with "zero-covariance" is a constraint. Then, for example, a graph containing only Gaussian factors is equivalent to solving an equality-constrained linear least squares problem. Please refer to [23] for additional details, which we extend in this paper to handle inequality constraints on scalar variables for certain problem structures.

REFERENCES

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

[2] Á. Nagy and I. Vajk, "Sequential Time-Optimal Path-Tracking Algorithm for Robots," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1253–1259, Oct. 2019.

[3] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, Sept. 1985. [Online]. Available: https://doi.org/10.1177/027836498500400301

[4] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, June 1985.

[5] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, Apr. 1987.

[6] J.-J. Slotine and H. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 118–124, Feb. 1989.

[7] S. Singh and M. C. Leu, "Optimal Trajectory Generation for Robotic Manipulators Using Dynamic Programming," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, no. 2, pp. 88–96, June 1987. [Online]. Available: https://doi.org/10.1115/1.3143842

[8] K. Hauser, "Fast Interpolation and Time-Optimization on Implicit Contact Submanifolds," in *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, June 2013. [Online]. Available: http://www.roboticsproceedings.org/rss09/p22.pdf

[9] D. Verscheure, B. Demeulenaere, J. Swevers, J. De 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.

[10] G. Chen, S. Hutchinson, and F. Dellaert, "Locally Optimal Estimation and Control of Cable Driven Parallel Robots using Time Varying Linear Quadratic Gaussian Control," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 7367–7374.

[11] G. Chen, S. Baek, J.-D. Florez, W. Qian, S.-W. Leigh, S. Hutchinson, and F. Dellaert, "GTGraffiti: Spray Painting Graffiti Art from Human Painting Motions with a Cable Driven Parallel Robot," in *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE, May 2022, pp. 4065–4072.

[12] K. Shin and N. McKay, "A Dynamic Programming approach to trajectory planning of robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 491–500, June 1986.

[13] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-4563%28200005%2917%3A5%3C233%3A%3AAID-ROB1%3E3.0.CO%3B2-Y

[14] Z. Shiller, "Time-energy optimal control of articulated systems with geometric path constraints," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, pp. 2680–2685 vol.4.

[15] J. T. Betts and W. P. Huffman, "Path-constrained trajectory optimization using sparse sequential quadratic programming," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 1, pp. 59–68, 1993. [Online]. Available: https://doi.org/10.2514/3.11428

[16] H. Pham and Q.-C. 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.

[17] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

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

[19] G. Chen, "C++ Implementation of "Generalizing Trajectory Retiming to Quadratic Objective Functions"," Sept. 2023. [Online]. Available: https://github.com/gchenfc/gtsam/tree/features/gerry/trajectory_retiming/gtsam_unstable/retiming

[20] H. Pham, Q.-C. Pham, J. Mirabel, and EdsterG, "Toppra: Time-Optimal Path Parameterization," 2023. [Online]. Available: https://github.com/hungpham2511/toppra

[21] A. Pott, T. Bruckmann, and L. Mikelsons, "Closed-form Force Distribution for Parallel Wire Robots," in *Computational Kinematics*, A. Kecskeméthy and A. Müller, Eds. Berlin, Heidelberg: Springer, 2009, pp. 25–34.

[22] M. Xie, A. Escontrela, and F. Dellaert, "A factor-graph approach for optimization problems with dynamics constraints," 2020.

[23] S. Yang, G. Chen, Y. Zhang, H. Choset, and F. Dellaert, "Equality Constrained Linear Optimal Control With Factor Graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 9717–9723.

[24] F. Dellaert and M. Kaess, *Factor Graphs for Robot Perception*. Foundations and Trends in Robotics, Vol. 6, 2017. [Online]. Available: http://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf

[25] F. Dellaert and GTSAM. Contributors, "Borglab/gtsam," Georgia Tech Borg Lab, May 2022. [Online]. Available: https://github.com/borglab/gtsam