Equality Constrained Linear Optimal Control With Factor Graphs

Shuo Yang1, Gerry Chen2, Yetong Zhang2, Frank Dellaert2, and Howie Choset1

Abstract—This paper presents a novel factor graph-based approach to solve the discrete-time finite-horizon Linear Quadratic Regulator problem subject to auxiliary linear equality constraints within and across time steps. We represent such optimal control problems using constrained factor graphs and optimize the factor graphs to obtain the optimal trajectory and the feedback control policies using the variable elimination algorithm with a modified Gram-Schmidt process. We prove that our approach has the same order of computational complexity as the state-of-the-art dynamic programming approach. Furthermore, current dynamic programming approaches can only handle equality constraints between variables at the same time step, but ours can handle equality constraints among any combination of variables at any time step while maintaining linear complexity with respect to trajectory length. Our approach can be used to efficiently generate trajectories and feedback control policies to achieve periodic motion or repetitive manipulation.

I. INTRODUCTION

The Equality Constrained Linear Quadratic Regulator (EC-LQR) is an important extension [1], [2] of the Linear Quadratic Regulator (LQR) [3]. The standard finite-horizon discrete-time LQR problem contains (1) quadratic costs on the state trajectory and the control input trajectory and (2) system dynamics constraints which enforce that the current state is determined by a linear function of the previous state and control. In the EC-LQR, auxiliary constraints are introduced to enforce additional linear equality relationships on one or more state(s) and/or control(s).

In many important problems, auxiliary constraints do not follow the Markov assumption yet non-Markovian auxiliary constraints are rarely considered in existing EC-LQR approaches. We classify auxiliary constraints in EC-LQR problems into two categories which we term local constraints and cross-time-step constraints. A local constraint only contains a state and/or control from the same time step. Examples of local constraints include initial and terminal conditions on states, contact constraints, and states along a predefined curve. In contrast, a cross-time-step constraint involves multiple states and controls at different time instances. Such non-Markovian constraints are pervasive in many robotics applications. For example, a legged robot’s leg configuration must return to the same state after a period of time during a periodic gait [4]. In optimal allocation with resource constraints [5], the sum of control inputs is constrained to be some constant. Our goal is to solve for both optimal trajectories and optimal feedback control policies to EC-LQR problems with local and cross-time-step constraints in linear time with respect to the trajectory length.

Reformulating control problems as inference problems [6]–[9] is a growing alternative to common trajectory optimization [10]–[12] and dynamic programming (DP) approaches for optimal control [1], [2], [13]. While trajectory optimization focuses on open-loop trajectories rather than feedback laws and a method using DP to handle cross-time-step constraints has yet to be proposed, control as inference may offer the advantages of both. Factor graphs in particular are a common tool for solving inference problems [14] and have recently been applied to optimal control [15], [16].

In this paper we propose a novel formulation using factor graphs [14] to efficiently solve the EC-LQR problem with both local and cross-time-step constraints in linear time with respect to trajectory length. We demonstrate how to represent the EC-LQR problem as a factor graph, and apply the variable elimination (VE) algorithm [17] on the factor graph to solve for the optimal trajectories and optimal feedback control policies. The flexibility of the factor graph representation allows cross-time-step constraints with arbitrary numbers of variables to be seamlessly handled. As long as the maximum number of variables involved in all constraints is bounded, the computational complexity of our method grows linearly with the trajectory length. The approach in this paper matches the computational complexity of standard dynamic programming techniques [2], but also has the added benefit of handing cross-time-step constraints.

1 Shuo Yang and Howie Choset are with the Robotics Institute and Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh. Emails: {shuoyang, choset}@andrew.cmu.edu
2 Gerry Chen, Yetong Zhang, and Frank Dellaert are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta. Emails: {gchen328, yzhang3333, fd27}@gatech.edu

Fig. 1: The factor graph representation of an Equality Constrained Linear Quadratic Regular (EC-LQR) problem. Circles with letters are states or controls. Filled squares and circles represent objectives and constraints that involve the state or controls to which they are connected. The red square represents a cross-time-step constraint.
II. RELATED WORK

Trajectory optimization methods typically transcript a problem into a Quadratic Programming (QP) [18] or Non-Linear Programming (NLP) [10] problem which can be efficiently solved to obtain open-loop trajectories of nonlinear systems. Local controllers can be used to track the open-loop trajectories generated [12]. Designing local controllers that obey equality constraints motivates EC-LQR problems.

For EC-LQR problems with just local constraints, DP-based approaches can generate both the optimal trajectories and feedback control policies. Solving standard LQR using DP is well understood in control theory [3], [12] tackles EC-LQR with state-only local constraints by projecting system dynamics onto the constraint manifold. [1] extends the DP approach by using Karush-Kuhn-Tucker (KKT) conditions [5] to absorb auxiliary constraints into the cost function, but its computation time grows with the cube of the trajectory length for certain auxiliary constraints. [2] solves the EC-LQR problem into a Quadratic Programming (QP) [18] or Non-Linear Programming (NLP) [10] problem.

LQR with state-only local constraints by projecting system dynamics and feedback control policies. Solving standard LQR using based approaches can generate both the optimal trajectories and its corresponding state trajectory as we focus on in this paper, an equivalent least-squares problem. Designing local controllers that systems. Local controllers can be used to track the open-loop trajectories generated [12].

Fig. 2: Factor graph of a standard LQR problem with trajectory length \( T = 2 \).

A. Problem Formulation

For a robotic system with state \( x_t \in \mathbb{R}^n \) and control input \( u_t \in \mathbb{R}^m \), we define a state trajectory as \( x = [x_0, x_1, \ldots, x_T] \) and control input trajectory as \( u = [u_0, u_1, \ldots, u_{T-1}] \) where \( T \) is the trajectory length. The optimal control input trajectory \( u^* \) and its corresponding state trajectory \( x^* \) are the solution to the constrained linear least squares problem:

\[
\begin{align}
\min_u & \quad x_T^T Q_{xx} x_T + \sum_{t=0}^{T-1} (x_t^T Q_{xx} x_t + u_t^T Q_{uu} u_t) \\
\text{s.t.} & \quad x_{t+1} = F_{xy} x_t + F_{yu} u_t, \\
& \quad G_{xu} x_t + G_{uu} u_t + g_t = 0, \quad t \in \mathcal{C}_1 \\
& \quad G_{xy} x_T + g_{1T} = 0 \\
& \quad \sum_{i \in C_1} S_{xki} x_i + \sum_{j \in C_2} S_{ukj} u_j + s_k = 0, \quad k \in \mathcal{C}_2
\end{align}
\]

where \( Q_{xx}, Q_{xy}, \) and \( Q_{uu} \) are positive definite matrices defining the cost function; \( F_{xy}, \) and \( F_{yu} \) define the system dynamics at time \( t \), constraints \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \) are local auxiliary constraints; and constraint \((1e)\) is a new formulation for cross-time-step constraints. In \((1c)\) and \((1d)\), \( G_{xu} \in \mathbb{R}^{l_t \times n}, \) \( G_{uu} \in \mathbb{R}^{l_t \times m}, \) and \( g_t \in \mathbb{R}^{l_t} \) form local constraints with constraint dimension \( l_t; \mathcal{C}_1 \) is the set of time steps where a local constraint applies; \( G_{xy} \) and \( g_{1T} \) form a local constraint with dimension \( l_T \) on the final step. In the cross-time-step constraint \((1e)\), \( S_{xki} \in \mathbb{R}_{\geq 0}^{l_t \times n}, \) \( S_{ukj} \in \mathbb{R}_{\geq 0}^{l_t \times m}, \) and \( s_k \in \mathbb{R}^{l_t} \) form constraints on a set of states \( x_t \) and controls \( u_j \) for \( k \in \mathcal{C}_2 \) where \( \mathcal{C}_2 \) is the set cross-time-step constraint indices.

B. Standard LQR as a Factor Graph

We demonstrate how to represent standard LQR, Problem \([1]\) without constraints \((1c), (1d), \) and \((1e)\), as the factor graph shown in Figure 2 and subsequently obtain the optimal trajectory and the feedback control policy using VE.

Factor graphs can be interpreted as describing either a joint probability distribution with conditional independencies or, as we focus on in this paper, an equivalent least-squares problem derived from minimizing the log-likelihood. A factor graph is a bipartite graph consisting of variables and factors connected by edges, where a factor can be viewed either as...
where \( \exp \) is the exponential function and \( X \) is the estimation of variables. For Gaussian distributions, the probability distribution of variables follows Gaussian distributions. We construct factor graphs to describe probability distributions, and the covariance of the probability distribution, and a joint probability distribution or least squares objective over the variables it is connected to.

We will begin by showing how the probabilistic view of factor graphs is equivalent to a least squares minimization [14]. We construct factor graphs to describe probability distribution of variables \( X = [x, u] \). Factors are conditional probabilities that change the Maximum A Posteriori (MAP) estimation of variables. For Gaussian distributions, the probability distribution for a single objective or constraint factor \( \phi_k \) can be written in matrix form as

\[
\phi_k(X_k) \propto \exp \left\{ -\frac{1}{2} \| A_k x_k - b_k \|^2_{\Sigma_k} \right\}
\]

where \( \exp \) is the exponential function and \( X_k \) contains the variables connected to the factor. \( A_k \) and \( b_k \) are a matrix and a vector respectively having problem-specific values, \( \Sigma_k \) is the covariance of the probability distribution, and \( \| \cdot \|^2_{\Sigma} := (\cdot)\Sigma^{-1}(\cdot) \) denotes the Mahalanobis norm. \( A_k, b_k, \) and \( \Sigma_k \) together define the probabilistic distribution of the factor.

The product of all factors defines the posterior distribution of \( X \) whose MAP estimate is the solution of the least squares problem [14]:

\[
X_{MAP} = \arg \max_X \phi(X) = \arg \min_X -\log(\prod_k \phi_k(X_k))
\]

\[
= \arg \min_X \sum_k ||A_k x_k - b_k||^2_{\Sigma_k} = \arg \min_X ||AX - b||^2_{\Sigma}
\]

where \( A, b \) and \( \Sigma \) stack together all \( A_k, b_k, \) and \( \Sigma_k \) respectively such that each factor, \( \phi_k \) corresponds to a block row in \( A \). Defining the weight matrix \( W := \Sigma^{-1} \), \( X_{MAP} \) minimizes a weighted least squares expression \( (AX - b)^T W (AX - b) \).

The objective factors in Figure 3 are \( \phi_{obj, k}(x_k) \propto \exp \left\{ -\frac{1}{2} ||Q_{x_k}^{1/2} x_k||^2 \right\} \) or \( \phi_{obj, u}(u) \propto \exp \left\{ -\frac{1}{2} ||Q_{u}^{1/2} u||^2 \right\} \), while the constraint factors are \( \phi_{con, k}(x_k) \propto \exp \left\{ -\frac{1}{2} ||x_k + F x_k - F u||^2_{\Sigma_k} \right\} \) where the covariance \( \Sigma_c = 0 \) creates infinite

Fig. 3: Two variable eliminations for the LQR problem. Each sub-figure consists of three rows showing three equivalent representations: the factor graph (top), constrained optimization (middle), and modified Gram-Schmidt process on \( [A_i | b_i] \) (bottom). The arrows in the factor graphs show variable dependencies. The thin horizontal arrows separate cases before and after elimination. Terms and symbols in the same color correspond to the color-coded variable elimination steps in Section III-B. Note that the matrix factorization representation consists of the weight vector, \( W_i \), next to the sub-matrix \( [A_i | b_i] \).

Fig. 4: Two elimination steps for EC-LQR with local constraints. This figure has the same layout as Figure 3.
terms in $W$. When factor graphs have factors with zero
covariance, the least squares problem turns into a constrained
least squares problem which we can solve using e.g. modified
Gram-Schmidt [23].

The VE algorithm is a method to solve (2) while exploiting the
sparsity of $A$ by solving for one variable at a time. For a
variable $\theta_i \in X$, we can identify its separator $S_i$: the set
of other variables sharing factors with $\theta_i$. Then we extract
sub-matrices $A_i$, $W_i$, and sub-vector $b_i$ from the rows of $A$,
$W$, and $b$ such that $[A_i|b_i]$ contains all factors connected to
$\theta_i$. We collect the rows in $[A_i|b_i]$ with finite weights to define
objective function $\phi_i(\theta_i, S_i)$ and rows with infinite weights
to define constraint factor $\psi_i(\theta_i, S_i)$. Then we “eliminate”
variable $\theta_i$ following 3 steps\footnote{In the probabilistic form, steps 2 and 3 would come from factoring
$\phi_i(\theta_i, S_i) \psi_i(\theta_i, S_i) = p(\theta_i|S_i)p(S_i)$. For Gaussian distributions, $\theta_i(\theta_i) =
E[p(\theta_i|S_i)]$ and $\psi_i(\theta_i)\psi_i(S_i) = p(S_i)$.}.

Step 1. Identify all the factors adjacent to $\theta_i$ to get $[A_i|b_i]$.

Split $[A_i|b_i]$ into $\phi_i(\theta_i, S_i)$ and $\psi_i(\theta_i, S_i)$.

Step 2. Solve the (constrained) least squares problem:

$$\theta_i^*(S_i) = \arg\min_{\theta_i} \phi_i(\theta_i, S_i) \text{ s.t. } \psi_i(\theta_i, S_i) = 0$$

using modified Gram-Schmidt or other constrained
optimization methods [5, Ch.10]. $\theta_i^*(S_i)$ denotes that
$\theta_i^*$ is a function of the variables in $S_i$.

Step 3. Substitute $\theta_i \leftarrow \theta_i^*$ by replacing the factors
$\phi_i(\theta_i, S_i)$ and $\psi_i(\theta_i, S_i)$ with $\phi_i(\theta_i^*, S_i)$ and $\psi_i(\theta_i^*, S_i)$ :=
$\psi_i(\theta_i^*, S_i)$, respectively, in $[A|b]$.

We follow an elimination order [19] to eliminate one
variable $\theta_i \in X$ at a time. After all variables are eliminated,
the factor matrix $A$ is effectively converted into an upper-triangular
matrix $R$ allowing $X$ to be solved by matrix back-substitution.
Therefore, one interpretation of the VE algorithm is performing sparse QR factorization on $A$ [14].

To apply VE to the LQR factor graph in Figure 2, we
choose the ordering $x_N, u_{N-1}, x_{N-2}, \ldots, x_0$ and execute Steps
1-3 to eliminate each variable. When eliminating a state $x_i$
for the special case of LQR, the constrained least-squares
problem in Step 2 is trivially solved as $x_i^T = x_{i-1} + F_i u_{i-1} + F_i x_{i-1}$. Additionally, $x_i^T$ will be empty since
$\psi_i^*(S_i)$ is satisfied for any choice of $u_{i-1}$ and $x_{i-1}$. Figure 3a shows the factor graphs, corresponding optimization
problems, and sub-matrices $[W_i][A_i|b_i]$ before and after eliminating $x_2$.

The optimal feedback control policy emerges when eliminating
a control $u_i$. The combined constraint factor $\psi_{u_i}$
is empty (since $\psi_{u_i}^*$ is empty), so Step 2 reduces to an
unconstrained minimization problem. To solve it using QR
factorization, split the objective $[A_i|u_i]|x_i|_2^2 = |R_i u_i + T_i x_i|_2^2 +
|E_i x_i|_2^2$ using the QR factorization $A_i = Q_i R_i$ noting that
$Q_i$ is orthogonal and thus doesn’t change the norm. Then,
$u_i^T = -K_i x_i$, where $K_i := R_i^{-1} T_i$. This efficiently optimizes
the first term and $\phi_{u_i}^*(x_i) = |E_i x_i|_2^2$ is the new factor on $x$. The elimination
is shown in Figure 3b.

Furthermore, the $\text{cost}_{\text{to-go}}$ (or “value function” [24]),
which commonly appears in DP-based LQR literature, is

$$\text{cost}_{\text{to-go}}(x_1) = x_1^T Q x_1 + x_1^T E_1^2 x_1.$$
the sparse QR factorization result of $A$, we apply back substitution whose computation complexity is $O(T \cdot (n^2 + m^2))$, so the overall computation complexity of solving the trajectory with length $T$ is $O(T \cdot (k_T n^3 + k_S n^2 m + k_S mn^2 + k_S m^3))$, which is the same as the state of the art DP approach [2].

D. EC-LQR with Cross-time-step Constraints

The factor graph’s ability to add factors on any set of variables allows us to add more general auxiliary constraints and objectives than [2], such as cross-time-step constraints. The VE algorithm for solving EC-LQR with cross-time-step constraints remains exactly the same as in Section III-C. Note that cross-time-step objectives could also be handled the same way if desired. Taking Figure 5 as an example, the cross-time-step constraint is $Sx_{n+p} - Sx_n - s = 0$, where $S$ is a selection matrix that selects certain elements from the state, $p$ is the period of the motion cycle, and $s$ is a constant state “advancement” vector. When eliminating $x_{n+p}$, its separator will contain $x_{n+p-1}$. After elimination of $x_{n+p}$, the new constraint to go factor will be connected to not only $x_{n+p-1}$ and $x_n$, but also $x_{n-1}$. Subsequent elimination steps will generate similar factors so that, after all variables are eliminated, the final feedback controllers for control inputs between $x_{n+p}$ and $x_n$ are functions of two states instead of just the current state. Figure 5 illustrates the result in the form of a Bayes Net [14] where arrows represent the variable dependencies.

Our method maintains linear complexity for cross-time-step constraints. From a complexity analysis point of view, as long as the number of variables involved in all constraints is bounded, then the maximum dimension of the separator for any variable will also be bounded thus bounding the complexity of each variable elimination step.

IV. EXPERIMENTS

We run simulation experiments to demonstrate the capability of the proposed method. We implement our method using the Georgia Tech Smoothing And Mapping (GTSAM) toolbox [25]. We compare our approach with three baseline methods implemented in MATLAB. Baseline method 1 is [1], baseline method 2 is [2], and baseline method 3 is using Matlab’s quadprog quadratic programming solver (which does not produce an optimal control policy). We first present comparison experiments for EC-LQR of a toy system with local constraints. We then show our approach handling cross-time-step constraints on an example system motivated by a single leg hopping robot.

A. Cost & Constraint Violation Comparison

The first experiment is to find the optimal trajectory for a simple system with $x_t \in \mathbb{R}^3$ and $u_t \in \mathbb{R}^3$ that are subjected to state constraints. The EC-LQR problem is given by:

$$\min_{u_t} \sum_{t=0}^{T-1} (x_t^T Q_{xx} x_t + u_t^T Q_{uu} u_t)$$

$$\text{s.t.} \quad x_{t+1} = F_t x_t + F_u u_t$$

$$x_0 = [0 \ 0 \ 0]^T, \quad x_N = [3 \ 2 \ 1]^T$$

$$x_{N/2} = [1 \ 2 \ 3]^T$$

where $dt = 0.01$, $F_x = I_3 \times 3 + I_3 \times 3 \cdot dt$, $F_u = I_3 \times 3 \cdot dt$, $T = 100$, $Q_{xx} = 0.01 \cdot I_3 \times 3$, $Q_{uu} = 0.001 \cdot I_3 \times 3$, and $Q_{xu} = 500 \cdot I_3 \times 3$.

Figure 6 compares the optimal state trajectories using the three methods. All three methods arrive at the exact same solution, with 0 constraint violation and identical total cost. This is expected because in this problem the optimal solution is unique. The result supports the correctness of our method.

B. Optimal Controller Comparison

In this experiment we show the optimal controller generated by the proposed method is equivalent to that generated by [2]. To demonstrate our method can solve state-only
constraints as well as state and control local constraints, we replace the state-only constraint (3d) in Problem 3 to be a constraint that contains both the state and the control as

\[ x_{N/2} + u_{N/2} + [1 \ 2 \ 3]^T = 0. \] (4)

We solve this problem to get the optimal controllers \( u_t = -K_s x_t + k_i \) for the two methods. \( K_s \) and \( k_i \) are identical between the two methods, as shown in Figure 7.

C. Cross-time-step Constraints

To illustrate an example of how cross-time-step constraints can be used to generate useful trajectories, we use a double integrator system \( x_t = [\text{position, velocity}], u = \text{acceleration} \) with periodic “step placements”. Consider the x-coordinate of a foot of a hopping robot which initially starts in contact with the ground and makes contact with the ground again every 20 time steps. Each contact, it advances forward by 0.6 units and must match the ground velocity (which may be non-zero e.g. on a moving walkway). The problem is:

\[
\min \ x_T^T Q_{xx} x_T + \sum_{t=0}^{T-1} (x_t^T Q_{xx} x_t + u_t^T Q_{uu} u_t) \\
\text{s.t.} \ x_{t+1} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ dt \end{bmatrix} u_t, \ x_0 = [0 \ 0]^T, \\
x_{n_*+20} - x_{n_*} = \begin{bmatrix} -0.6 \\ 0 \end{bmatrix}, n_* = 0, 20, 40, 60, 80 \] (5a, b, c)

The cross-time-step constraints (5c) enforce that contacts must occur at a fixed position relative to and with same velocity as the previous contacts \( p = 20 \) time steps prior. These create constraint factors between two state variables \( p = 20 \) time steps apart, as in Figure 5.

Figure 8 shows the solutions to Problem 5 using Baseline 2 [2], Baseline 3 (QP), and our method, as well as the results when using the same controllers with a perturbed initial state \( x_0 = [0 \ 1.8]^T \) (i.e. walking on a moving walkway with velocity 1.8). We apply some modifications to allow for comparison since Baseline 2 cannot natively handle cross-time-step constraints and Baseline 3 cannot generate an optimal policy, but even so, the adjusted baselines do not generate optimal trajectories from perturbed initial state, as shown in Figure 8 (bottom). For baseline method 2, we convert the cross-time-step constraints to same-time-step constraints \( x_{n_*} = [0.03 n_* \ 0]^T \) for \( n_* = 0, 20, \ldots \) resulting in incorrect constraint violation after perturbing the initial state. An alternative would be to introduce 10 additional state dimensions (two for each cross-time-step constraint) analogous to Lagrange multipliers, but we argue that an approach is not sustainable for online operation and many cross-time-step constraints. For Baseline 3, we re-use the control sequence from Problem 5 for the perturbed case. Our method’s control law produces a state trajectory that is optimal and without constraint violation even with a perturbed initial state as shown in Figure 8 (bottom right).

V. Future Work

The proposed method points to a promising direction in generating trajectories for constrained high dimensional robotics systems such as legged robots and manipulators. Just as LQR is a building block of Differential Dynamic Programming (DDP) [26] and iterative LQR [13], linear factor graphs could be the building block of a more general nonlinear optimal control algorithm, but the following need to be explored first:

- Incorporate inequality constraints into the factor graph by converting them to equality constraints e.g. using barrier or penalty functions [27].
- Extend to nonlinear systems by using nonlinear factor graphs [14].
- Address over-constrained “constraints” in VE.
- Leverage incremental factor graph-based solving using Bayes Trees [28] to do efficient replanning.
- Consider stochastic optimal control with uncertain constraints.
- Combine estimation and optimal control factor graphs together to better close the perception-control loop.

VI. Conclusions

In this paper, we proposed solving equality constrained linear quadratic regular problems using factor graphs. We showed that factor graphs can represent linear quadratic optimal control problems with auxiliary constraints by capturing the relationships amongst variables in the form of factors. Variable elimination, an algorithm that exploits matrix sparsity to optimize factor graphs, is used to efficiently solve for the optimal trajectory and the feedback control policy. We demonstrated that our approach can handle more general constraints than traditional dynamic programming approaches. We believe this method has great potential to solve difficult constrained optimal control problems for a number of complex robotics systems.