IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Barrier Method for Inequality Constrained Factor Graph Optimization with Application to Model Predictive Control

Anas Abdelkarim[1,2], Daniel Görges[2], and Holger Voos[1]

***Abstract*—Factor graphs have demonstrated remarkable efficiency for robotic perception tasks, particularly in localization and mapping applications. However, their application to optimal control problems—especially Model Predictive Control (MPC)—has remained limited due to fundamental challenges in constraint handling. This paper presents a novel integration of the Barrier Interior Point Method (BIPM) with factor graphs, implemented as an open-source extension to the widely adopted g2o framework. Our approach introduces specialized inequality factor nodes that encode logarithmic barrier functions, thereby overcoming the quadratic-form limitations of conventional factor graph formulations. To the best of our knowledge, this is the first g2o-based implementation capable of efficiently handling the constraints within a unified optimization backend. We validate the method through a multi-objective adaptive cruise control application for autonomous vehicles. Benchmark comparisons with state-of-the-art constraint-handling techniques demonstrate faster convergence and improved computational efficiency. (Code repository: https://github.com/snt-arg/bipm_g2o)**

***Index Terms*—Constrained factor graphs, Interior point method, Robotics control, Robotics perception**

I. Introduction

FACTOR graphs have emerged as a powerful paradigm for robotic perception, enabling efficient modeling of large-scale probabilistic inference problems through their sparse graphical structure [1]. These models are typically solved using gradient-based optimization techniques and have been widely adopted in applications such as Simultaneous Localization and Mapping (SLAM) and situational awareness [2].

However, extending factor graphs to optimal control tasks—particularly Model Predictive Control (MPC) [3], [4]—introduces significant new challenges. Unlike perception problems, control applications require rigorous enforcement of both equality and inequality constraints, *e.g.,* to ensure feasibility and safety [5].

While MPC problems are traditionally formulated using domain-specific modeling tools such as CasADi [6] or AMPL [7], [8], factor graphs offer a compelling unified framework that bridges perception and control. This integration can enable shared representations across both domains, [9], simplifying system architecture and enhancing consistency and performance in complex robotic systems.

This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), MOCCA Project, ref. 17041397. For the purpose of open access, and in fulfilment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission. (*Corresponding author: Anas Abdelkarim.*)

[1] Anas Abdelkarim and Holger Voos are with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg, Luxembourg (e-mail: first.last@uni.lu).

[2] Anas Abdelkarim and Daniel Görges are with the Department of Electrical and Computer Engineering (EIT), RPTU Kaiserslautern–Landau, 67663 Kaiserslautern, Germany (e-mail: {abdelkarim, goerges}@eit.uni-kl.de).

Digital Object Identifier (DOI): see top of this page.

Several works in the literature have addressed the incorporation of equality constraints into factor graphs, *e.g.,* [9], [10]. For inequality constraints, the Augmented Lagrangian (AL) method [11] has been the de facto choice due to its compatibility with the quadratic cost structure of standard factor graphs. However, Interior Point Methods (IPMs) offer superior theoretical properties, including polynomial-time complexity and improved numerical stability, especially for large-scale nonlinear optimization problems [12], [13]. Despite these advantages, IPMs have not yet been systematically integrated into factor graph-based formulations.

The key novelty of this work is the first systematic integration of IPMs into factor graphs via a novel class of inequality factor nodes that encode logarithmic barrier functions. Unlike the quadratic penalties used in AL, these barrier terms are inherently non-quadratic and do not naturally conform to conventional factor graph representations.

We address this challenge by introducing specialized inequality factors capable of directly representing logarithmic barriers, thus enabling efficient and direct enforcement of inequality constraints within the factor graph framework. This advancement allows the computational and convergence benefits of IPMs [14] to be realized in structured probabilistic optimization settings, while preserving the modularity and interpretability of factor graphs.

We implement our proposed method as an open-source extension to the g2o library and validate it against the state-of-the-art AL-based approach in the factor graph literature. Our implementation preserves a shared front-end abstraction across both BIPM and AL methods, simplifying constraint insertion and emphasizing algorithmic modularity. The resulting new solver is evaluated on a high-performance MPC framework for autonomous driving, specifically for a multi-objective adaptive cruise control (MACC) task. This application is representative of real-world robotics control tasks with multiple critical inequality constraints for safety and comfort.

The complete C++ implementation is publicly available for reproducibility and further development under https://github.com/snt-arg/bipm_g2o).

II. Related Work

Factor graphs have been extensively studied as an efficient framework for probabilistic inference in robotic perception [1]. Recent developments, as reviewed comprehensively in [9], have extended factor graphs to optimization-based control, also revealing their potential as a unified representation for both perception and control.

Incorporating equality constraints into least-squares formulations has a long history in optimization. In the context of robotics, several works have adapted these principles to factor graph frameworks. For instance, weighted penalty terms

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

have been used to encode equality constraints in factor graph formulations, with applications in trajectory generation for autonomous vehicles [15]. In addition, Ta et al. [16] utilized SQP to solve optimization problems (OPs) with nonlinear equality-constrained factor graphs, applying them to flight control with obstacle avoidance for a quadrotor.

Moreover, Bai et al. [10] presented a constrained formulation for pose-graph SLAM, where loop-closure cycles are explicitly represented as equality constraints, and proposed an incremental SQP-based SLAM algorithm (iSQP) for solving the optimization. Furthermore, in our previous work [17], we have extended the Gauss-Newton approach, aligning with SQP, to provide a practical g2o implementation of equality constraints.

In this work, we focus on the inequality-constrained case, where direct factor graph integration remains less explored and presents distinct challenges compared to the equality case. Xie *et al.* [18] introduced box constraints into a factor graph-based MPC framework using hinge loss functions as soft penalties. Similar approaches were adopted by King-Smith *et al.* [19] and Yang *et al.* [20]. However, hinge loss functions penalize constraint violations based on current variable estimates, which are often inaccurate in the early stages of the optimization process. This can result in unnecessary penalties or potentially slowing the convergence.

To address these limitations, Augmented Lagrangian (AL) methods have been explored as a more principled approach for constraint enforcement in factor graph optimization. The ICS framework by Sodhi *et al.* [21] integrates AL with the iSAM algorithm to support incremental constrained smoothing, assuming fixed linearization points. Qadri *et al.* [22] introduced Incremental Constrained Optimization (InCOpt) within the iSAM2 framework. They employed the Augmented Lagrangian (AL) method in combination with fluid linearization techniques to achieve more efficient constraint handling. However, they reported that the algorithm may yield underdetermined systems if an inequality constraint is satisfied and inactivated. Building on this line of work, Bazzana *et al.* [5] developed an AL-based solver for SRRG2 and demonstrated its applicability in both pose estimation and real-world MPC tasks.

Despite these advances, AL methods are known to be highly sensitive to hyperparameter tuning, including penalty coefficients, their update strategies, and the maximum number of inner and outer iterations. This sensitivity often leads to fragility in practical applications. Our analysis reveals that the tuning parameters used in [5] differ significantly from those suitable in our application, highlighting the strong context-dependence of AL-based methods.

In contrast, Barrier Interior Point Methods (BIPMs) offer a compelling alternative. They exhibit faster convergence rates [23], require fewer hyperparameters, and demonstrate greater robustness across a variety of problem settings [12]. However, BIPMs have not been systematically integrated into factor graph optimization, primarily due to the incompatibility between logarithmic barrier terms and the sum-of-squares cost structure of conventional factor graphs.

This motivates our work, which introduces a novel class of inequality factor nodes that encode barrier terms, enabling the first direct integration of BIPM into factor graph-based control frameworks. This integration fills a major methodological gap and expands the capabilities of factor graphs for constrained optimization in robotics and autonomous systems.

III. Preliminaries

Factor graphs are probabilistic graphical models consisting of two types of nodes: variable nodes, which represent unknown states or parameters, and factor nodes, which encode constraints or relationships among these variables. In maximum a posteriori (MAP) inference, the goal is to maximize the joint probability distribution defining the factors, which factorizes according to the graph structure. This can be expressed as

$$X^{\text{MAP}} = \underset{X}{\operatorname{argmax}} \prod_{j=1}^{r} \exp\left(-\frac{1}{2}\left\|e_j\left(X\right)\right\|_{\Omega_j}^2\right), \tag{1}$$

where X is the stacked vector containing all variable nodes, $\|e\|_{\Omega}^2 = e^{\mathrm{T}}\Omega e$ denotes the Mahalanobis norm, $\exp(.)$ is the exponential function, and $.^{\mathrm{T}}$ represents the transpose operator for vectors or matrices. The term e_j is the error function associated with factor node j, and Ω_j is the positive-definite information matrix of edge j, typically the inverse of the covariance matrix. The MAP objective can be reformulated as a weighted least squares problem [9, §III.B]

$$X^{\text{MAP}} = \underset{X}{\operatorname{argmin}} \quad \sum_{j=1}^{r} \left\|e_j\left(X\right)\right\|_{\Omega_j}^2 \tag{2}$$

In the error functions are typically highly nonlinear. Therefore, a first-order linear approximation is applied around a reference point X^i

$$\hat{e}_j\left(X\right) = e_j(X^i) + \mathcal{J}_{e_j}(X^i)[X - X^i], \tag{3}$$

where $\mathcal{J}_{e_j}(X^i)$ is the Jacobian matrix of the error function associated with factor node j, evaluated at the linearization point X^i.

Applying the first-order optimality condition, which states that the gradient of the cost function vanishes at an optimum, the Gauss-Newton update step can be derived as

$$\sum_{j=1}^{r} \underbrace{\left\|\mathcal{J}_{e_j}(X^i)\right\|_{\Omega_j}^2}_{\mathrm{H}_{e_j}^i} \Delta X_{\text{gn}}^i = \sum_{j=1}^{r} \underbrace{-\mathcal{J}_{e_j}(X^i)^{\mathrm{T}}\Omega_j e_j(X^i)}_{\mathfrak{b}_{e_j}^i} \tag{4}$$

Each factor node contributes a block to the overall linear system used to compute the update step. We refer to such factor nodes as "cost factor nodes" to distinguish them from the constraint factor nodes introduced later.

In the following section, we describe how inequality constraints are integrated into the factor graph formulation using the Barrier Interior Point Method (BIPM).

IV. Methodology

This section presents the BIPM from an optimization perspective [24] and describes incorporating inequality constraints into the factor graphs. Consider the constrained OP

$$\min_{X} \quad \sum_{j=1}^{r} \left\|e_j(X)\right\|_{\Omega_j}^2 \tag{5a}$$

$$\text{s.t.} \quad h_j(X) = 0 \qquad j = 1, \cdots, l \tag{5b}$$

$$g_j(X) \leq 0 \qquad j = 1, \ldots, q \tag{5c}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

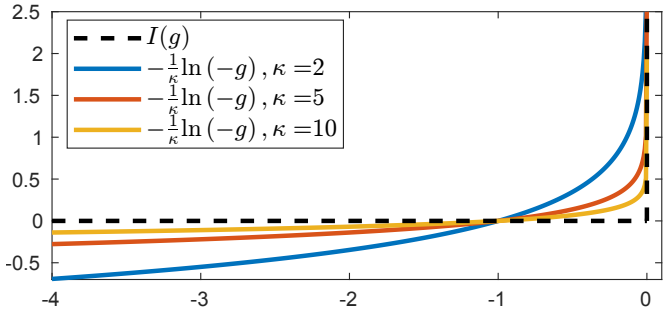

Fig. 1. The approximation of the barrier function

where $h_j : \mathbb{R}^n \to \mathbb{R}^{d_{h_j}}$ are equality constraints and $g_j : \mathbb{R}^n \to \mathbb{R}^{d_{g_j}}$ are inequality constraints. For handling the equality constraints, we adopt the framework presented in [17].

A. Barrier Interior Point Method

The core idea of the Interior Point Method (IPM) is to incorporate inequality constraints into the objective function using a barrier function. With an ideal barrier function $I(\cdot)$, the OP is reformulated as

$$\min_X \quad \sum_{j=1}^{r} \|e_j(X)\|^2_{\Omega_j} + \sum_{j=1}^{q} I(g_j(X)), \tag{6a}$$

$$\text{s.t.} \quad h_j(X) = 0 \qquad j = 1, \ldots, l \tag{6b}$$

The ideal barrier function is defined as

$$I(g) = \begin{cases} 0 & g \leq 0 \\ \infty & g > 0 \end{cases} \tag{7}$$

The barrier term $(\sum_{j=1}^{q} I(g_j))$ remains zero as long as the inequality constraints $g_j \leq 0$ are satisfied, preserving the original objective function. However, once X exits the feasible region defined by these constraints, the cost function becomes infinite. Thus, to minimize the cost in (6a), the solution X must remain strictly within the feasible region.

In practice, the ideal barrier function is non-differentiable and therefore unsuitable for gradient-based optimization. To address this, a differentiable logarithmic approximation is used

$$\hat{I}(g) = -\frac{1}{\kappa}\ln(-g), \qquad g < 0 \tag{8}$$

The approximation $\hat{I}(g)$ converges to the ideal barrier $I(g)$ for $g < 0$ as $\kappa > 0$ increases as illustrated in Fig. 1. The resulting approximate OP becomes

$$\min_X \quad \sum_{j=1}^{r} \|e_j(X)\|^2_{\Omega_j} - \frac{1}{\kappa}\sum_{j=1}^{q} \ln(-g_j(X)) \tag{9a}$$

$$\text{s.t.} \quad h_j(X) = 0 \qquad j = 1, \ldots, l \tag{9b}$$

The BIPM (or primal-barrier IPM) solves a sequence of such approximated problems. It starts with a small κ and strictly feasible guess-point (*i.e.,* satisfying $g_j(X) < 0$). At each iteration, a step is taken using backtracking to ensure the updated solution remains feasible. Once an optimal point is found for the current κ, the barrier parameter is increased by a factor $\nu > 1$, and the process repeats until convergence is achieved for a sufficiently large κ.

Although the resulting problem is an equality-constrained optimization, it cannot be directly solved using standard equality-constrained factor graph frameworks. This is because the cost terms derived from the inequality constraints are not quadratic. For instance, attempting to represent the logarithmic barrier term with a factor node defined by

$$e_{\cdot}(X) = \sqrt{-\ln(-g_j(X))}, \text{ and } \Omega_{\cdot} = 1/\kappa, \tag{10}$$

is problematic. The argument of the square root is not guaranteed to be non-negative for all feasible X, making the expression potentially undefined or complex. Therefore, special care is required in how such constraints are incorporated into factor graph-based solvers.

B. Inequality Factor Node

To incorporate the BIPM into factor graphs, we define specialized inequality factor nodes. These nodes use tailored error functions and information matrices to efficiently encode inequality constraints. For clarity, we present the formulation considering a simplified OP with a single nonlinear inequality constraint $g_j : \mathbb{R}^n \to \mathbb{R}^{d_{g_j}}$.

Assuming g_j is nonlinear, we linearize it around the current estimate X^i, similar to how error functions are handled in conventional cost factor nodes, *i.e.,*

$$g_j(X) \approx \hat{g}_j(X) = g_j(X^i) + \mathcal{J}_{g_j}(X^i)[X - X^i] \tag{11}$$

From the BIPM perspective, the OP containing only this inequality factor node becomes

$$\min_X \quad \underbrace{-\frac{1}{\kappa}\sum_{j=1}^{d_{g_j}} \ln(-\hat{g}_{n,j}(X))}_{f(X)} \tag{12}$$

To solve this unconstrained problem, we adopt a Gauss-Newton approach by enforcing the first-order optimality condition, *i.e.,* setting the gradient of $f(X)$ to zero:

$$\nabla f(X) = -\sum_{j=1}^{d_{g_j}} \frac{\overbrace{\nabla \hat{g}_{n,j}(X)}^{\mathcal{J}^{\mathsf{T}}_{g_{n,j}}(X^i)}}{\kappa\, \hat{g}_{n,j}(X)} = 0 \tag{13}$$

Using a first-order Taylor expansion around the current estimate, the gradient is approximated as

$$\nabla f(X^i + \Delta X) \approx \nabla f(X^i) + \mathcal{J}_{\nabla f}(X^i)\Delta X = 0, \tag{14}$$

where $\mathcal{J}_{\nabla f}(X^i)$ is the Hessian matrix of $f(X)$ and $\hat{g}_{n,j}(X^i) = g_{n,j}(X^i)$. This yields the following linear system

$$\underbrace{\sum_{j=1}^{d_{g_j}} \frac{\mathcal{J}^{\mathsf{T}}_{g_{n,j}}(X^i)\mathcal{J}_{g_{n,j}}(X^i)}{\kappa[g_{n,j}(X^i)]^2}}_{\mathrm{H}^i_{g_j}} \Delta X^i = \underbrace{\sum_{j=1}^{d_{g_j}} \frac{\mathcal{J}^{\mathsf{T}}_{g_{n,j}}(X^i)}{\kappa\, g_{n,j}(X^i)}}_{\mathfrak{b}^i_{g_j}} \tag{15}$$

This update step can be interpreted as a second-order update step, where the matrix $\mathrm{H}^i_{g_j}$ represents a Gauss–Newton approximation of the Hessian matrix of the cost term (12). In particular, we apply the first-order approximation to the first-order optimality condition, *i.e.,* (13), which yields a second-order update step—equivalent to applying the optimality condition directly to a second-order approximation of the cost term. This explains why the solver can handle nonlinear constraints.

Moreover, the inequality constraints are linearized within the barrier function, while the barrier function itself remains

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

in its nonlinear form. At each optimization step, the solver iteratively updates the linearized inequality constraints, thereby accommodating nonlinearities. A similar principle applies to nonlinear cost factors in factor graph literature, where error functions are linearized at each iteration yet still enable efficient optimization in nonlinear problems, as presented in Section III (*i.e.,* (3)).

Based on (15), we define a BIPM-compatible inequality factor node characterized by

- an *error function* derived directly from the inequality constraint:
$$e_{g_j}(X) = g_j(X), \tag{16}$$
- a *diagonal information matrix* that encodes the barrier scaling:
$$\Omega_{g_j} = \kappa^{-1} \cdot \mathrm{diag}\left([g_j(X)]^{-2}\right) \tag{17}$$

The contribution of the inequality factor to the linear system is then expressed as

$$\mathrm{H}^i_{g_j} = \left\| \mathcal{J}_{e_{g_j}}(X^i) \right\|^2_{\Omega_{g_j}} \tag{18a}$$

$$\mathfrak{b}^i_{g_j} = \mathcal{J}_{g_j}(X^i)^{\mathrm{T}} \Omega_{g_j} e_{g_j}(X^i). \tag{18b}$$

Notably, the residual vector $\mathfrak{b}^i_{g_j}$ differs from conventional cost factor terms in that it excludes the negative sign (cf. (4) and (18)). This distinction is important and must be accounted for when integrating inequality factors into the overall solver framework.

To simplify implementation, the library provides a dedicated class for inequality factor nodes. The factor is defined independently of the underlying solver, while the library automatically constructs the corresponding linear system based on the selected optimization algorithm.

C. BIPM Algorithm for Factor Graph Optimization

We describe here the BIPM solver, which follows the standard primal-barrier interior point framework. In the convex setting, such methods enjoy well-established global convergence guarantees and polynomial-time complexity bounds, see [12, §11] and [13] for detailed theoretical analyses. Our adaptation preserves these properties in convex problems, while allowing applicability to nonconvex OPs. The solver involves a backtracking line search to maintain the feasibility of inequality constraints throughout the optimization process. Specifically, after computing the update ΔX, we start with an initial step size $\zeta = 1$ and iteratively reduce it by a factor $\alpha \in (0, 1)$ until all inequality constraints evaluated at $X + \zeta \Delta X$ are strictly negative. Second, we implement an outer loop to gradually increase the barrier parameter κ, thereby refining the approximation of the ideal barrier function and improving solution accuracy. Finally, comprehensive termination criteria ensure the optimization process stops only when both optimality and constraint satisfaction have been achieved:

- Update convergence: $\|\Delta X\|_2 < \epsilon_x$
- Inequality feasibility: $\|\max(g(X), 0)\|_\infty < \epsilon_g$
- Equality satisfaction: $\|h(X)\|_\infty < \epsilon_h$

where $h(X)$, and $g(X)$ represent the aggregated equality and inequality constraint vector, respectively.

Algorithm 1 summarizes the complete BIPM solver, where unsubscripted vectors aggregate all relevant subvectors. Feasibility is maintained via backtracking line search, while the barrier parameter κ is increased iteratively to tighten the approximation. The system matrix H and residual vector $\mathfrak{b}$ correspond to all optimization variables, including Lagrange multipliers γ. While presented in global form for clarity, practical implementations assemble H and $\mathfrak{b}$ from local factor contributions via sparse indexing, and impose limits on total and inner-loop iterations for efficiency and scalability.

Algorithm 1: Barrier-IPM Solver for Factor Graphs

Input: $\{e_j, \Omega_j\}_{\text{j=1:r}}$, $\{h_j\}_{\text{j=1:l}}$, $\{g_j\}_{\text{j=1:q}}$, $\kappa > 0$, $\alpha \in (0, 1)$, $\epsilon_x > 0$, $\epsilon_g > 0$, $\epsilon_h > 0$, $\kappa_{\text{final}} > 0$, $\nu > 1$, γ_0, feasible X

initialize $\gamma_{h_j} \leftarrow \gamma_0$ for all $j = 1 : l$, $X \leftarrow [X, \boldsymbol{\gamma}]$, $\mathrm{H} \leftarrow 0$, $\mathfrak{b} \leftarrow 0$;
repeat
  repeat
    foreach e_j, Ω_j **do**
      $\mathrm{H} \leftarrow \mathrm{H} + \mathrm{H}_{e_j}$;
      $\mathfrak{b} \leftarrow \mathfrak{b} + \mathfrak{b}_{e_j}$ (Eq. 4);
    foreach h_j **do**
      $\mathrm{H} \leftarrow \mathrm{H} + \mathrm{H}_{e_{h_j}}$;
      $\mathfrak{b} \leftarrow \mathfrak{b} + \mathfrak{b}_{e_{h_j}}$ [17, (Eq. 17)];
    foreach g_j **do**
      $\mathrm{H} \leftarrow \mathrm{H} + \mathrm{H}_{e_{g_j}}$;
      $\mathfrak{b} \leftarrow \mathfrak{b} + \mathfrak{b}_{e_{g_j}}$ (Eq. 18);
    $\Delta X \leftarrow$Solve ($\mathrm{H}\Delta X = \mathfrak{b}$)
    Set step size $\zeta \leftarrow 1$;
    while $g(X + \zeta \Delta X) \geq 0$ **do**
      $\zeta \leftarrow \alpha\zeta$;
    $X \leftarrow X + \zeta \Delta X$;
  until $\|\Delta X\|_2 < \epsilon_x$ *and* $\|\max(g, 0)\|_\infty < \epsilon_g$ *and* $\|h\|_\infty < \epsilon_h$;
  $\kappa \leftarrow \nu\kappa$;
until $\kappa \geq \kappa_{final}$;
return X

V. Results and Performance Evaluation

This section presents a comparative evaluation of the proposed Barrier Interior Point Method (BIPM)-based solver against an Augmented Lagrangian (AL) baseline. Both solvers are applied to the Multi-objective Adaptive Cruise Control (MACC) problem for battery electric vehicles (BEVs), formulated as a Model Predictive Control (MPC) task. For evaluation, simulations were conducted using the comprehensive validated vehicle model and scenario described in our previous work [25]. The real driving tests performed in [25] demonstrate that the simulation model accurately represents real driving scenarios, which makes the evaluation and comparison of the novel solver-based model reliable. This setup will also guide a real experimental validation of the proposed solver as future work.

For the AL implementation, we adopted the approach described in [5] to construct the linear systems corresponding to the constraints and update the Lagrangian variables. Both AL and BIPM utilize identical stopping criteria and involve an inner and outer loop. In the BIPM framework, the inner loop solves the OP at a fixed barrier parameter κ, while the outer

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

loop updates this parameter. In contrast, AL solves the OP for fixed Lagrangian variables and penalty parameter ρ, with the outer loop updating both the Lagrangian variables and the penalty parameter.

A. Factor Graph Representation of MACC

In this work, we focus on the MACC problem, selected as a representative benchmark due to its multiple objectives and a variety of inequality constraints, including both hard and soft constraints, and later extended to nonlinear and nonconvex cases in Section V-E. Following the framework in [25], the OP for MACC is given by

$$\min_{X} \sum_{i=k}^{k+N-1} (\underbrace{\|p(v_i, F_{t,i})\|^2_{\Omega_p}}_{\phi_1} + \underbrace{\|F_{b,i}\|^2_{w_1}}_{\phi_2} + \underbrace{\|\delta_{\text{far},i}\|^2_{w_2}}_{\phi_3} + \underbrace{\|\delta_{F_t,i}\|^2_{w_3}}_{\phi_4} + \underbrace{\|\delta_{F_b,i}\|^2_{w_4}}_{\phi_5}) \tag{19a}$$

subject to

$$v_{i+1} - \left(v_i + \frac{T_s}{m_{\text{eq}}}(F_{t,i} - F_{b,i} - \bar{F}_{\text{resist},i})\right) = 0 \tag{19b}$$

$$d_{i+1} - \left(d_i + \frac{T_s}{2}(v_{p,i} + v_{p,i+1} - (v_i + v_{i+1}))\right) = 0 \tag{19c}$$

$$d_{\min} + h_{\text{safety}} v_{i+1} - d_{i+1} \leq 0 \tag{19d}$$

$$d_{i+1} - (d_{\min} + h_{\text{track}} v_{i+1}) + \delta_{\text{far},i} \leq 0 \tag{19e}$$

$$-F_{t,i} \leq 0 \tag{19f}$$

$$F_{t,i} - F_{t,\max} \leq 0 \tag{19g}$$

$$F_{t,i} - (a_{20} + a_{21} v_i) \leq 0 \tag{19h}$$

$$-F_{b,i} \leq 0 \tag{19i}$$

$$F_{b,i} - F_{b,\max} \leq 0 \tag{19j}$$

$$v_{i+1} - v_{\max,i+1} \leq 0 \tag{19k}$$

$$-v_{i+1} \leq 0 \tag{19l}$$

$$|F_{t,i} - F_{t,i-1}| - \delta_{F_t,i} \leq 0 \tag{19m}$$

$$|F_{b,i} - F_{b,i-1}| - \delta_{F_b,i} \leq 0 \tag{19n}$$

Here, $X = [v, d, F_t, F_b, \delta_{\text{far}}, \delta_{F_t}, \delta_{F_b}]$ denotes the variable nodes of the factor graph: host velocity, inter-vehicle distance, traction and braking forces, and slack variables. T_s is the sampling time, and $i \in \{k, \ldots, k+N-1\}$ indexes the control horizon. The preceding vehicle velocity is v_p, while $\bar{F}_{\text{resist}}$ models resistive forces. Finally, h_{safety} and h_{track} are time-headway parameters defining safe and tracking distances.

It is worth noting that the constraint (19d) represents a form of obstacle avoidance by enforcing a safety distance from the preceding vehicle. Similarly, general obstacle avoidance constraints can be encoded as inequality factors of the form $g(x) = d_{\text{safe}} - \|p(x) - p_{\text{obs}}\| \leq 0$, and handled directly by the proposed BIPM framework without any structural changes.

The corresponding factor graph of the MACC problem is illustrated in Fig. 2.

B. Simulation Setup and Initialization Strategy

The simulative evaluation compares two optimization approaches: the BIPM and the AL method. Both solvers were implemented on identical hardware platforms, consisting of an Intel Core i9 12th Gen processor and 32 GB of RAM running Ubuntu 22.04. To ensure a fair comparison, both methods used

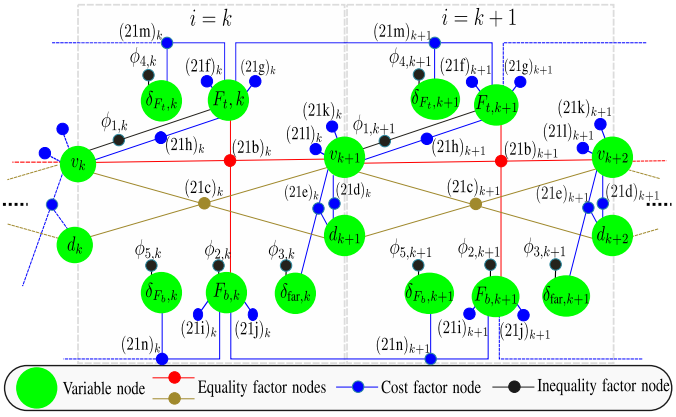

Fig. 2. Factor graph representation of the MACC problem

the same MPC parameters, constraint settings, and stopping criteria for convergence and feasibility. In the simulations, the host vehicle is modeled as an electric vehicle prototype tasked with tracking a leading vehicle following a predefined driving cycle. The cycle conforms to the Real Driving Emissions (RDE) requirements under the European regulation [26]. A total driving duration of approximately 7 minutes was simulated, with the MPC controller executing at a sampling interval of 100 ms. This resulted in approximately 4000 OPs solved throughout the scenario. The simulation captures a variety of realistic urban driving conditions, consistent with typical real-world driving patterns.

Regarding variable initialization, we leverage the problem's physics to construct a feasible initial guess. This is achieved by systematically projecting the warm-start solution from the previously solved MPC problem into the feasible region. Specifically, host vehicle velocities v_{i+1} are first corrected to satisfy (19k), *e.g.*, $v_{i+1} \leftarrow \min\{v_{i+1}^{\text{warm}}, v_{\max,i+1} - \epsilon\}$, with $\epsilon > 0$ ensuring strict feasibility, and then verified against (19l), $v_{i+1} \leftarrow \max\{v_{i+1}^{\text{warm}}, v_{\min,i+1} + \epsilon\}$. Similarly, distances d_i are adjusted with respect to (19d), followed by the slack variables $\delta_{far,i}$ for (19e). The same procedure is applied to traction forces $F_{t,i}$ (19f–19h), their variation slacks $\delta_{F_t,i}$ (19m), braking forces $F_{b,i}$ (19i, 19j), and finally $\delta_{F_b,i}$ (19n). This strategy ensures that the initial guess lies strictly inside the feasible region while remaining close to the nominal operating point.

Remarks: For inequality constraints where finding a feasible initial point is difficult, one can introduce auxiliary slack variables, reformulating the constraint as $g_i(x) + s_i = 0, s_i > 0$. This requires only a positive initialization of s_i. Moreover, mutually exclusive constraints—those that are difficult to satisfy simultaneously—can be relaxed by rewriting them as $g_i(x) + s_i \leq 0$ and adding a penalty term $w_i s_i^2$ to the cost function, where w_i is a weighting factor.

C. Solver Comparison under Different Prediction Horizons

A comparative performance analysis of the two optimization approaches was carried out across different prediction horizons $N \in \{3,\ 6,\ 20\}$. While a horizon length of $N = 3$ was considered in [5], our prior experience suggests that horizons in the range of $N = 5, \ldots, 7$ are more appropriate for the MACC application. To evaluate solver performance at larger problem scales, we also include a longer horizon of $N = 20$.

The problem complexity scales linearly with the horizon length N: the number of decision variables is $X = 7N$,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

with $l = 2N$ equality constraints and $q = 13N$ inequality constraints. Both solvers were configured with a maximum of 300 outer iterations and up to 10 iterations per inner loop. All Lagrange multipliers were initialized to zero.

For the BIPM solver, the barrier method parameters were set to $\kappa_0 = 0.5$, $\kappa_{\text{final}} = 1500$, with an update factor $\nu = 8$. For the AL solver, we adopted the algorithm described in [9, §IV.D.2], initializing the penalty parameters as $\rho_0 = 0.5$, $\rho_{\max} = 5\text{e}5$, and updating them by a factor $\rho_\nu = 20$.

Table I presents a comparative analysis of solver performance for BIPM and AL across varying MPC prediction horizons. The reported metrics include the average, maximum, and minimum number of total iterations, along with the corresponding solver computation times in milliseconds.

The BIPM solver consistently outperforms the AL method in terms of both iteration count and computational time across all tested horizons. For example, at $N = 6$, BIPM required an average of only 24.7 iterations compared to 37.9 for AL, while also achieving a faster average computation time (32.1 ms vs. 41.3 ms). The performance gap becomes more pronounced at larger horizons: for $N = 20$, BIPM reduced the average number of iterations by approximately 29%, while the AL reached the maximum iteration limit. These results demonstrate the superior scalability and computational efficiency of the BIPM solver as problem complexity increases.

TABLE I
PERFORMANCE COMPARISON OF AL AND BIPM METHODS

	N	Iterations Number ; Calculation time (ms)		
		avg	max	min
AL	3	31.3 ; 17.9	76 ; 76.5	**18** ; **10.2**
BIPM		**23.6** ; **16.6**	**36** ; **62.9**	**18** ; 12.5
AL	6	37.9 ; 41.3	89 ; 120.9	20 ; **22.5**
BIPM		**24.7** ; **32.1**	**65** ; **106.5**	**19** ; 24.6
AL	20	49.5 ; 170.7	300 ; **1042.6**	31 ; 108.5
BIPM		**35.3** ; **151.3**	**263** ; 1178.3	**23** ; **98.8**

Figure 3 illustrates the closed-loop behavior of the MACC problem using both the BIPM and AL solvers under identical configurations. The subplots show the reference and actual velocities of the preceding and following vehicles, the inter-vehicle distance, and the control force input over time. Both solvers produce nearly indistinguishable tracking performance and control effort, which is expected since they operate under the same prediction model, constraints, controller settings, and stopping criteria. This confirms that, when appropriately tuned, both optimization methods can achieve comparable control quality despite their differing algorithmic structures. However, as detailed below, parameter tuning tends to be more challenging and sensitive in the case of the AL algorithm.

D. Solver Performance Under Parameter Tuning

To evaluate the impact of parameter tuning on solver performance, we present Tables II and III, which summarize the behavior of the BIPM and AL solvers under various configurations. Cases 2 and 8, highlighted in bold, were previously presented in Table I for a horizon length of $N = 6$. The parameters are varied based on these two baseline cases in order to examine the influence of parameter tuning on solver performance, considering them as reference configurations. In these tables, Inner Iter denotes the maximum number of inner iterations allowed. The standard deviation (SD) reflects the variability in the total number of iterations across multiple runs, with higher values indicating less consistent convergence behavior.

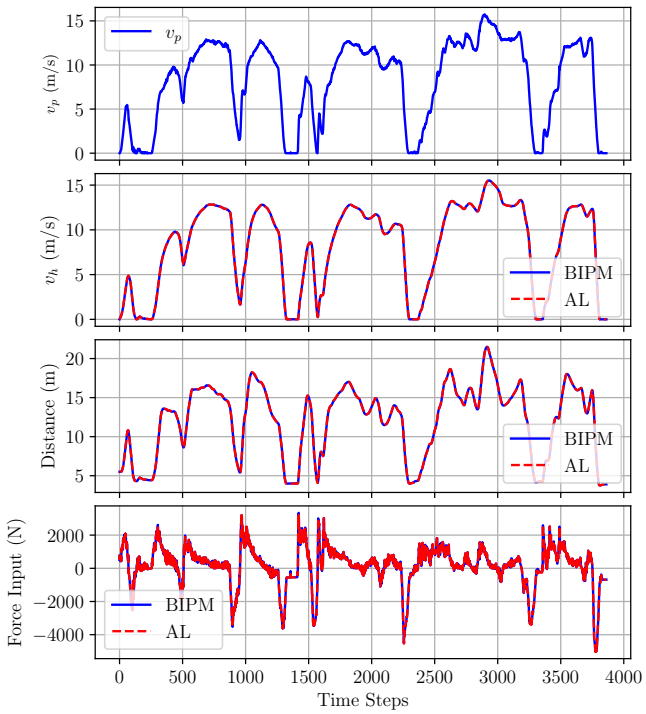

Fig. 3. Comparison of control performance between BIPM and AL solvers applied to the MACC problem. The v_p profile in the first subplot is the same for both the AL and the BIPM.

TABLE II
INFLUENCE OF PARAMETER TUNING ON BIPM SOLVER PERFORMANCE.

Case	ν	κ_0	κ_{final}	Inner Iter	Avg. Iter	Max Iter	SD
1	4	0.05	$1.5 \cdot 10^3$	10	27.3	56	2.2
2	**8**	**0.5**	$\mathbf{1.5 \cdot 10^3}$	**10**	**24.7**	**65**	**2.5**
3	50	0.5	$1.5 \cdot 10^3$	10	24.4	155	6.9
4	50	0.5	$1.5 \cdot 10^3$	25	29.9	58	3.2
5	8	10	$1.5 \cdot 10^3$	10	21.2	110	9.8
6	8	10	$1.5 \cdot 10^3$	25	22.5	73	4.8
7	20	2	$1.5 \cdot 10^6$	15	27.6	70	3.2

a) BIPM Solver Tuning Analysis.

Cases 1–3 demonstrate the impact of the barrier update factor ν on solver performance, aligning with the theoretical framework of Boyd *et al.* [12]. The barrier parameter update factor ν exhibits a key trade-off: smaller values (*e.g.,* $\nu = 8$) increase outer iterations but require fewer inner steps per iteration, while larger values (*e.g.,* $\nu = 50$) reduce outer iterations at the expense of increased inner computation. Notably, the total iteration count remains relatively insensitive to ν variations. Case 3, with $\nu = 50$ and only 10 inner iterations, shows higher standard deviation, indicating difficulty in following the central path—the trajectory of solutions to the barrier problem as the barrier parameter decreases. Increasing the maximum

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE III
INFLUENCE OF PENALTY PARAMETER TUNING ON AL SOLVER PERFORMANCE.

Case	ρ_ν	ρ_0	$\rho_{\max}$	Inner Iter	Avg. Iter	Max Iter	SD
8	**20**	**0.5**	$\mathbf{5 \cdot 10^5}$	**10**	**37.9**	**89**	**9.9**
9	5	0.5	$5 \cdot 10^5$	10	49.3	101	8.8
10	50	0.5	$5 \cdot 10^5$	10	33.6	173	10.0
11	50	0.5	$5 \cdot 10^5$	25	34.2	300	11.9
12	20	10	$5 \cdot 10^5$	10	32.1	157	8.4
13	20	10	$5 \cdot 10^5$	25	32.1	300	10.8
14	20	0.5	$5 \cdot 10^5$	25	38.6	143	11.3
15	20	0.5	$5 \cdot 10^4$	10	69.8	300	72.3
16	20	0.5	$5 \cdot 10^6$	10	40.1	70	7.3

number of inner iterations in Case 4 stabilizes the solver by allowing closer tracking of this path.

Cases 5–6 highlight the effect of initial barrier value κ_0. A large κ_0 increases iteration variance (Case 5), but allowing more inner steps (Case 6) improves stability by better aligning with the central path early on. Finally, Case 7 demonstrates that increasing the final barrier value κ_{final} by a factor of 10^3 increases the average iterations only modestly, suggesting the solver is robust to this parameter when others are well tuned.

b) AL Solver Tuning Analysis.

Cases 8–10 highlight the effect of the penalty update factor ρ_ν. While increasing ρ_ν reduces the average number of iterations (*e.g.,* Case 10 vs. 9), it also increases variation and the likelihood of hitting the iteration limit, as seen in the spike in maximum iterations in Case 10. In contrast, BIPM (Cases 1, 2, and 4) demonstrates more stable behavior with lower variance and predictable performance in similar scenarios.

Increasing the maximum number of inner iterations in AL (Cases 11, 13, and 14) did not yield improved stability. For instance, Case 11 reaches the iteration cap without significant improvement over Case 10. This contrasts with BIPM, where increasing inner iterations (Case 4 vs. Case 3) helped track the central path more effectively, reducing variance and improving convergence.

The initial penalty value ρ_0 also has a complex effect. In Case 12 vs. Case 8, raising ρ_0 decreased the average iteration count but led to higher maximum iteration counts—again highlighting AL's sensitivity to parameter changes.

Finally, adjusting $\rho_{\max}$ has a significant impact on solver behavior. Reducing $\rho_{\max}$ by a factor of 10 (*i.e.,* from $5 \cdot 10^5$ to $5 \cdot 10^4$, as in Case 15) severely degrades performance. Conversely, increasing it to $5 \cdot 10^6$ (Case 16) alters the iteration behavior but does not necessarily improve convergence. In fact, excessively large values of $\rho_{\max}$ may lead to numerical instability and inconsistent progress. For example, in [5], the penalty parameter is constrained to the range $[0.5,\ 2]$, which proves to be unsuitable for our application, where larger and more adaptive scaling is necessary for acceptable solver performance.

c) The Choice of the Backtracking Parameter

In our experiments, we investigated two step-size strategies. The first uses a fixed predefined vector of candidate values

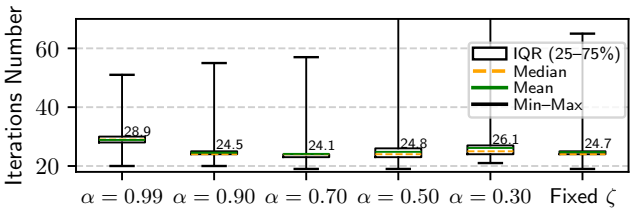

Fig. 4. Box-plot of iteration number across step-size strategies. IQR denotes the interquartile range (25–75%), shown as the height of the box; a taller box indicates higher variability, while a shorter box indicates more consistent iteration number. The y-axis is capped at 70 iterations; the actual maximums are 95 ($\alpha = 0.3$) and 81 ($\alpha = 0.5$).

$$\begin{aligned}\zeta = \{&1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.25, 0.2, 0.15, 0.1,\\ &9 \cdot 10^{-2}, 8 \cdot 10^{-2}, 7 \cdot 10^{-2}, 6 \cdot 10^{-2}, 5 \cdot 10^{-2}, 4 \cdot 10^{-2},\\ &3 \cdot 10^{-2}, 2 \cdot 10^{-2}, 1 \cdot 10^{-2}, 8 \cdot 10^{-3}, 7 \cdot 10^{-3}, 6 \cdot 10^{-3},\\ &5 \cdot 10^{-3}, 4 \cdot 10^{-3}, 3 \cdot 10^{-3}, 2 \cdot 10^{-3}, 1 \cdot 10^{-3} 10^{-3}, 10^{-4},\\ &10^{-5}, 10^{-6}, 10^{-7}, 10^{-9}, 10^{-10}, 10^{-12}, 10^{-12}, 10^{-14},\\ &10^{-16}, 10^{-18}, 10^{-20}, 10^{-25}, 10^{-30}\},\end{aligned}$$

while the second applies a single-parameter reduction strategy, with $\alpha \in \{0.99, 0.9, 0.7, 0.5, 0.3\}$, as specified in the step-size update of Algorithm 1, in line 16.

The results reported in Tables I–III were obtained with the predefined vector. As illustrated in Fig. 4, large α values (*e.g.,* 0.99) keep iterates close to the constraint boundary, increasing iterations and reducing numerical stability, while small α (*e.g.,* 0.30) values lead to fewer inequality evaluations but slower convergence. The predefined vector allows the solver to start with larger steps—avoiding overly small updates in the early phase—and then gradually shift to smaller steps, thereby reducing unnecessary inequality evaluations. Based on our findings, we recommend either the predefined vector or $\alpha \in [0.6, 0.95]$ for practical stability and efficiency.

In summary, while BIPM tuning shows predictable and interpretable effects tied to theoretical properties (*e.g.,* central path tracking), AL solver performance is more sensitive and less predictable, often requiring trial-and-error for practical tuning.

E. The Robustness of the Solver

We next evaluate the robustness of the proposed solver under noisy measurements and nonconvex constraints. Gaussian noise with zero mean was added to the MPC inputs measurements: inter-vehicle distance was perturbed with a standard deviation of 0.5 m, and host velocity with a standard deviation of 1.0 m/s. To introduce nonconvexity, the resistive force was modeled quadratically, making constraint (19b) nonlinear. In addition, constraints (19f)–(19l) were modified to cubic form by applying the transformation $a - b \leq 0; \mapsto; a^3 - b^3 \leq 0$, thereby enforcing stronger nonconvexity.

Fig. 5 reports the solver performance across ten experimental cases. Case 1 corresponds to the baseline problem with linear constraints and no noise. Case 2 introduces nonlinear constraints but no noise, while Cases 3–10 combine nonlinear constraints with the Gaussian noise model. The results confirm that the solver remains stable under both nonlinear constraints and noisy measurements. Compared to the baseline, the average iteration count increases only marginally (about 2–4 iterations on average), indicating that the solver can absorb perturbations and work with nonlinear constraints. While the maximum number of iterations fluctuates slightly under noisy

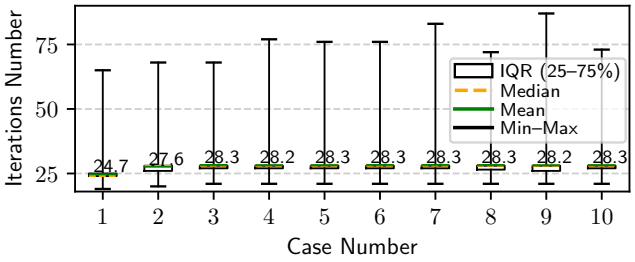

Fig. 5. Box-plot of iterations number across ten test cases under nonlinear constraints and Gaussian noise. Case 1: linear constraints; Case 2: nonlinear without noise; Cases 3–10: nonlinear with noisy measurements.

conditions, the interquartile range remains narrow, highlighting consistent solver behavior. This robustness under noisy measurements captures the primary gap between simulation and real-world testing—sensor uncertainty—and highlights the solver's applicability to real-time MPC

VI. Conclusion and Future Work

This work establishes a principled integration of Barrier Interior Point Methods (BIPM) into factor graph optimization also with inequality constraints as for instance required in optimal control tasks and especially MPC. By introducing inequality factor nodes encoding logarithmic barrier functions, we enable factor graphs to natively support inequality-constrained problems. Our implementation as an extension to the g2o library supports both BIPM and Augmented Lagrangian (AL) methods under a unified frontend, allowing seamless problem formulation with backend-specific solvers.

Empirical results from a multi-objective adaptive cruise control (MACC) task demonstrate the advantages of BIPM in terms of convergence speed and computational efficiency, particularly under large-scale problem settings. The proposed solver achieves comparable control performance to AL while offering improved robustness to hyperparameter tuning and problem scaling.

Finding a strictly feasible initial guess and ensuring stable performance when operating very close to feasibility boundaries remain key challenges for BIPM. Future work could investigate more advanced IPM variants, such as predictor–corrector or infeasible-start methods, to further improve convergence speed and enhance numerical stability in such cases.

References

[1] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[2] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "S-graphs+: Real-time localization and mapping leveraging hierarchical representations," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4927–4934, 2022.

[3] A. Bemporad, "Model predictive control design: New trends and tools," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 6678–6683.

[4] A. Abdelkarim, "Development of numerical solvers for online optimization with application to mpc-based energy-optimal adaptive cruise control," master thesis, Technische Universität Kaiserslautern, 2020. Available online at http://dx.doi.org/10.13140/RG.2.2.11897.28000, accessed: 2025-08-28.

[5] B. Bazzana, H. Andreasson, and G. Grisetti, "How-to augmented lagrangian on factor graphs," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2806–2813, 2024.

[6] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[7] A. Abdelkarim and P. Zhang, "Optimal scheduling of preventive maintenance for safety instrumented systems based on mixed-integer programming," in *7th International Symposium on Model-Based Safety and Assessment IMBSA*. Springer, 2020, pp. 83–96.

[8] A. Abdelkarim, Y. Jia, and D. Gorges, "Optimization of vehicle-to-grid profiles for peak shaving in microgrids considering battery health," in *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2023, pp. 1–6.

[9] A. Abdelkarim, H. Voos, and D. Görges, "Factor graphs in optimization-based robotic control-a tutorial and review," *IEEE Access*, vol. 13, pp. 28 315–28 334, 2025.

[10] F. Bai, T. Vidal-Calleja, and S. Huang, "Robust incremental slam under constrained optimization formulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1207–1214, 2018.

[11] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[12] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[13] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior methods for nonlinear optimization," *SIAM review*, vol. 44, no. 4, pp. 525–597, 2002.

[14] J. Bleyer, "Advances in the simulation of viscoplastic fluid flows using interior-point methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 330, pp. 368–394, 2018.

[15] 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)*. IEEE, 2021, pp. 9717–9723.

[16] D.-N. Ta, M. Kobilarov, and F. Dellaert, "A factor graph approach to estimation and model predictive control on unmanned aerial vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 181–188.

[17] A. Abdelkarim, H. Voos, and D. Görges, "ecg2o: A seamless extension of g2o for equality-constrained factor graph optimization," *arXiv preprint arXiv:2503.01311*, 2025.

[18] M. Xie, A. Escontrela, and F. Dellaert, "A factor-graph approach for optimization problems with dynamics constraints," *arXiv preprint arXiv:2011.06194*, 2020.

[19] M. King-Smith, P. Tsiotras, and F. Dellaert, "Simultaneous control and trajectory estimation for collision avoidance of autonomous robotic spacecraft systems," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 257–264.

[20] P. Yang and W. Wen, "Tightly joining positioning and control for trustworthy unmanned aerial vehicles based on factor graph optimization in urban transportation," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023, pp. 3589–3596.

[21] P. Sodhi, S. Choudhury, J. G. Mangelson, and M. Kaess, "Ics: Incremental constrained smoothing for state estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 279–285.

[22] M. Qadri, P. Sodhi, J. G. Mangelson, F. Dellaert, and M. Kaess, "Incopt: Incremental constrained optimization using the bayes tree," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6381–6388.

[23] D. Steck, "Lagrange multiplier methods for constrained optimization and variational problems in banach spaces," Ph.D. dissertation, Universität Würzburg, 2018.

[24] A. Abdelkarim, Y. Jia, and D. Görges, "An accelerated interior-point method for convex optimization leveraging backtracking mitigation," in *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2023, pp. 1–6.

[25] Y. Jia, A. Abdelkarim, X. Klingbeil, R. Savelsberg, and D. Görges, "Performance evaluation of energy-optimal adaptive cruise control in simulation and on a test track," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 4994–5000, 2023.

[26] European Commission, "Commission regulation (eu) no 2016/427," *Official Journal of the European Union*, vol. L 82, pp. 1–98, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/427/oj