2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Practical and Performant Enhancements for Maximization of Algebraic Connectivity

Leonard Jung[1], Alan Papalia[1,2], Kevin Doherty[3,*], Michael Everett[1]

***Abstract*— Long-term state estimation over graphs remains challenging as current graph estimation methods scale poorly on large, long-term graphs. To address this, our work advances a current state-of-the-art graph sparsification algorithm, maximizing algebraic connectivity (MAC). MAC is a sparsification method that preserves estimation performance by maximizing the algebraic connectivity, a spectral graph property that is directly connected to the estimation error. Unfortunately, MAC remains computationally prohibitive for online use and requires users to manually pre-specify a connectivity-preserving edge set. Our contributions close these gaps along three complementary fronts: we develop a specialized solver for algebraic connectivity that yields an average 2x runtime speedup; we investigate advanced step size strategies for MAC's optimization procedure to enhance both convergence speed and solution quality; and we propose automatic schemes that guarantee graph connectivity without requiring manual specification of edges. Together, these contributions make MAC more scalable, reliable, and suitable for real-time estimation applications.**

Code: `https://github.com/neu-autonomy/MACpp`

I. Introduction

The scalability of state estimation and perception remains a critical challenge for long-term autonomous robotic systems. Many of these problems, such as simultaneous localization and mapping (SLAM), pose graph optimization (PGO), and structure from motion (SFM) may be formulated as an optimization problem over graphs. However, as a system operates over an extended period, the graph may grow unchecked, accumulating a vast number of edges from odometry, loop closures, and other sensor measurements. This unbounded growth renders the underlying optimization problem computationally intractable, jeopardizing the real-time performance essential for autonomous operation.

One paradigm in which this issue is tackled is through *graph sparsification*: the removal of edges while retaining the essential information within the graph. Instead of solving a large, complicated estimation problem over the entire graph, the goal is to approximate it with a sparser one for which the problem can be solved more efficiently. This naturally raises the question of what figure of merit defines a 'good' sparsified graph. One particularly important metric is the algebraic connectivity, which is closely tied to the expected performance of estimators [1], [2].

However, this edge selection problem is unfortunately NP-hard [3]. Recent work [4] posed a convex relaxation that is more amenable to solving, but it remains computationally demanding. We build upon this framework, and improve both its computational efficiency and resulting graph quality. First, we leverage the specific sparsity pattern inherent to the Laplacian graph for PGO graphs and develop a numerical conditioning strategy around this pattern. Secondly, we investigate methods to specify/return a connected subgraph, the selection of which is critical to solution quality, yet has previously been unexplored.

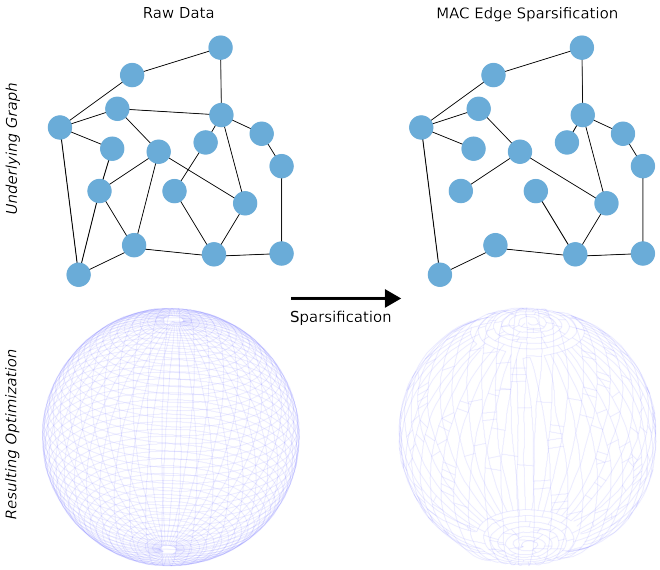

Fig. 1: We sparsify a graph by pruning edges in a way that maintains the graph's structural properties, as quantified by its algebraic connectivity. (Top) Example of a "mock" graph undergoing sparsification. (Bottom) Solution of the estimation problem on the sparsified graph retains the overall shape of the original while using significantly fewer edges.

Our primary contributions are as follows:

1) We present an efficient strategy for computing the Fiedler value via Krylov-subspace methods and sparse matrix factorization, yielding a two-fold acceleration in solution times compared to a general-purpose approach.
2) We investigate the impact of line-search strategies and other variants of the Frank-Wolfe algorithm on solving the maximizing algebraic connectivity problem, and empirically determine that the simple decaying step-size for Frank-Wolfe has the best convergence rate
3) We present a method to remove (restrictive) assumptions that a backbone is given, and present an improved heuristic for determining this (now optional) backbone

This research was funded by the DEVCOM Army Research Laboratory (ARL) in part by W911NF-24-2-006 and SARA CRA W911NF-24-2-0017
[1]Northeastern University, USA. [2]University of Michigan. [3]Boston Dynamics. *This work was conducted in personal time and independently of author's organization.

II. Related Works

The pursuit of computational efficiency in robotics graph optimization problems has motivated a wide range of sparsification techniques. This section reviews related work, beginning with a broad overview of sparsification methods employed in perception and estimation. We then narrow our focus to a more formal class of graph-theoretic sparsification algorithms, within which our work on maximizing algebraic connectivity is situated. Finally, we discuss the key computational considerations for calculating the Fiedler value, a bottleneck in any iterative spectral sparsification routine.

A. Sparsification Techniques in Perception and Robotics

Several methods address memory and computation costs for long-term autonomy using heuristics to sparsify nodes and edges such as [5]–[7]. [8], [9] utilize submodular optimization to select which keyframes or features are most important to track, inherently building a smaller, more computationally efficient graph. Relatedly, point cloud sparsification techniques such as [10] also reduce memory and computational constraints by subsampling sensor data prior to graph construction. We aim to directly solve a relaxed convex optimization problem on the underlying graph's spectral properties for sparsification, while maintaining suboptimality guarantees.

Tangentially, [11] use spectral properties to specifically solve the PGO problem by leveraging sparsity; our method can be seen as a preprocessing step for any optimization problem over graphs.

B. Fiedler Value Graph Sparsification

Our work is most directly related to spectral graph sparsification methods that aim to preserve the algebraic connectivity (the Fiedler value) of the graph Laplacian. This line of research is rooted in the foundational work of [12], which established the importance of the Fiedler value as a measure of graph connectivity.

The MAC (maximizing algebraic connectivity) algorithm [13] represents the state-of-the-art in Fiedler value-based graph sparsification. Our work addresses several of its key computational limitations to enhance its practicality.

The connection between algebraic connectivity and system performance has been extensively studied in theory and demonstrated empirically in multiple previous works. [14] also uses the Fiedler value for sparsification and additionally computes the number of edges to remove. [15] formulates the problem of adding edges to maximize the Fiedler value as a convex semidefinite program (SDP). Methods like OASIS [16] also solve a similar smallest eigenvalue sparsification problem for sensor placement.

Like [13], it is important to distinguish our goal from that of spectral sparsification [17], which aims to preserve the entire spectrum of the original graph Laplacian. Our objective is narrower: we seek only to maximize the Fiedler value. This is a less restrictive condition that allows for more aggressive sparsification and is directly related to the performance metrics of estimation problems, such as the Cramer-Rao lower bound [1].

C. Fiedler Value Calculation

A practical bottleneck in the MAC algorithm is the frequent computation of the Fiedler value and its corresponding eigenvector. Shift-invert Lanczos methods and other Krylov subspace techniques [18] are both well-established methods for computing eigenvalues. The Tracemin-Fiedler [19] algorithm presented a method more tailored to finding the Fiedler value. As iterative solvers, the performance of these routines is heavily dependent on the conditioning of the matrix. Advanced vertex ordering (e.g., [20]) and preconditioning techniques [21] can improve this. Our work uses several of these techniques along with properties of a graph Laplacian to drastically decrease computation time.

III. Background

Many robotic perception problems are traditionally posed as estimation over graphs $\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E}, w)$, where the variables of interest are the nodes $\mathcal{V}$, measurements are the edges $\mathcal{E}$, and precision are the edges w.

Graphs found in robotics applications can easily exceed 10^5 edges. Since estimation scales poorly in both memory and computational complexity with the number of edges, sparsification is crucial for enabling reasonable performance.

A. Algebraic Connectivity

Let $\delta(i)$ denote the set of edges incident to node i (i.e., the neighbors of i). The algebraic connectivity, also known as the Fiedler value, of a graph is the second smallest eigenvalue λ_2 of the Laplacian $L \in \mathbb{R}^{n \times n}$ of graph $\mathcal{G}$ where:

$$L_{ij} = \begin{cases} \sum_{e \in \delta(i)} w_e, & i = j \\ -w_{ij}. & \{i,j\} \in \mathcal{E} \\ 0 & \{i,j\} \notin \mathcal{E} \end{cases}$$

This graph Laplacian has several key properties of note:

- It is always positive semi-definite (PSD).
- It has at least one zero eigenvalue, which corresponds to the "all ones" eigenvector $\mathbf{1}$. More generally, the multiplicity of the zero eigenvalue equals the number of connected components in the graph
- The Laplacian can be written as the sum of the Laplacian of the subgraphs induced by each of its edges.

We refer to the eigenvector associated with the Fiedler Value as the Fiedler vector q_2 and the two together as the Fiedler Pair.

The algebraic connectivity is inversely related to the worst-case error for solutions of many estimation problems [2], [13]; that is, if we aim to determine which states and measurements should be pruned while minimally changing the quality of our solution, the resulting graph should maximize the algebraic connectivity of the graph with nodes corresponding to robot poses x_i and edge weights equal to w_{ij}.

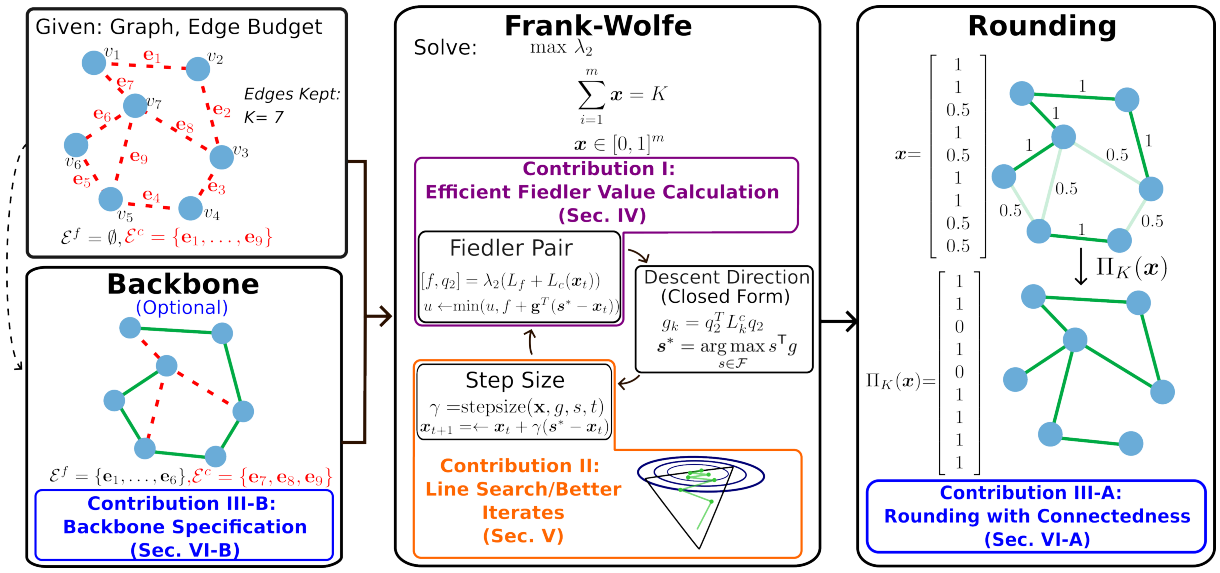

Fig. 2: **Overview of MAC algorithm with highlighted contributions:** MAC [13] is given a graph and prespecified "backbone" (fixed edge set), then applies a Frank–Wolfe procedure with a decaying step size, and finally rounds the solution to a binary one. To reduce total runtime and broaden MAC's applicability to problems without a pre-specified backbone, we investigate three main questions: **1.** Can we decrease MAC's runtime through Faster Fiedler value calculations (Section IV)?; **2.** Can we increase MAC's convergence rate through intelligent step size and linesearch routines (Section V)?; **3.** Can we extend MAC to problems without a given fixed edge set through rounding functions Π_K(Section VI-A) and automatic backbone edge specification (Section VI-B)?

B. Maximization of Algebraic Connectivity

Given an edge budget K, our objective is to find the set of K edges that maximizes the algebraic connectivity (Fiedler value) of the sparsified graph. Defining L_k as the Laplacian of the subgraph induced by edge e_k, m as the number of edges to optimize over, we formalize this as the following problem, where $L(\boldsymbol{x}) = \sum_{k=1}^{m} \boldsymbol{x}_k L_k$:

Problem 1. Fiedler Value Maximization

$$\begin{aligned} &\max_{\boldsymbol{x}\in\{0,1\}^m} \lambda_2\left(L(\boldsymbol{x})\right) \\ &\sum_{k=1}^{m} \boldsymbol{x}_k = K \end{aligned} \tag{1}$$

However, due to the integrality constraint $\boldsymbol{x}_k \in \{0,1\}$ this problem is NP-Hard. [13] solves this via a convex relaxation:

Problem 2. Convex Relaxation of Fiedler Value Maximization

$$\begin{aligned} \boldsymbol{x}^* = &\arg\max_{\boldsymbol{x}\in[0,1]^m} \lambda_2\left(L(\boldsymbol{x})\right) \\ &\sum_{k=1}^{m} \boldsymbol{x} = K \end{aligned} \tag{2}$$

As the Fiedler value of a Laplacian is a concave function [22] of $\boldsymbol{x}$, Problem 2 is concave and can be efficiently solved to global optimality. However, due to the relaxation of the integrality constraint, the solution of Problem 2 may not be a feasible point for Problem 1.

If the solution to Problem 2 results in a non-integral solution, a rounding function is used to return a valid solution:

$$\boldsymbol{x}^*_{\{0,1\}^m} = \Pi_K(\boldsymbol{x}^*) \tag{3}$$

where $\Pi_K : [0,1]^m \to \{0,1\}^m$ is any function that rounds $\boldsymbol{x}$ to an integral solution, and $\sum_k \boldsymbol{x}_k = K$. This rounding function can drastically impact the final graph: for instance, poor rounding may result in a disconnected graph. Therefore, the choice of rounding strategy plays a critical role in the final solution quality.

C. Frank-Wolfe for Maximizing Algebraic Connectivity

[13] demonstrates that the Frank-Wolfe algorithm is particularly well-suited to solve Problem 2. The Frank-Wolfe algorithm consists of two steps. First, a descent direction is found from minimizing the linearized objective function

$$\boldsymbol{s}^{(t)} = \arg\min_{\boldsymbol{s}\in\mathcal{F}} \langle \nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{s}\rangle \tag{4}$$

Then, the next iterate is found by taking a step in the descent direction $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \gamma(\boldsymbol{s}^{(t)} - \boldsymbol{x}^{(t)})$. Although the Frank-Wolfe algorithm is designed for smooth convex functions, it is easily adapted to non-smooth, concave functions such as the Fiedler Value by negating and using sub-gradients.

For Problem 2, each Frank-Wolfe iteration requires only a single evaluation of the Fiedler value and its corresponding eigenvector. A super-gradient (the concave analog to the sub-gradient) $g \in \mathbb{R}^m$ of $\lambda_2(\boldsymbol{x})$ can then be found:

$$g = (I_m \otimes q_2)^{\mathsf{T}} \begin{pmatrix} L_1 \\ \vdots \\ L_m \end{pmatrix} q_2 \tag{5}$$

where q_2 is a Fiedler vector, I_m is the identity matrix, and $\otimes$ denotes the Kronecker product..

Algorithm 1 MAC Algorithm [13]

Input: $\mathcal{E}^f \leftarrow$ Fixed Edges, $\mathcal{E}^c \leftarrow$ Candidate Edges, $T \leftarrow$ Total Iterations, $\epsilon_u \leftarrow$ Duality Gap, $\boldsymbol{x}^{(0)} \leftarrow$ Initial Guess, $K \leftarrow$ Number of Candidate Edges to Keep
Output: Edges Kept

function MAC($\mathcal{E}^f, \mathcal{E}^c, T, \boldsymbol{x}^{(0)}$)
  for $t = 0, \dots, T-1$ **do** ▷ Frank-Wolfe
    $f \leftarrow \lambda_2(L_f + \sum_{k=1}^{m} \boldsymbol{x}_k^{(t)} L_k^c)$
    $\boldsymbol{g}, \boldsymbol{s} \leftarrow$ from (5) and (6), respectively
    $u \leftarrow \min(u, f + \boldsymbol{g}^\mathsf{T}(\boldsymbol{s} - \boldsymbol{x}^{(t)})$ ▷ Dual
    if $|\frac{u-f}{f}| < \epsilon_u$ **then**
      break
    end if
    $\gamma \leftarrow \frac{2}{2+t}$ ▷ Naive Step Size
    $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \gamma(\boldsymbol{s} - \boldsymbol{x}^{(t)})$
  end for
  return $\Pi_K(\boldsymbol{x}^{(T)})$ ▷ Rounding
end function

The descent direction $\boldsymbol{s} = [\boldsymbol{s}_1, \dots, \boldsymbol{s}_m]^\mathsf{T}$problem is then solved in closed form

$$\boldsymbol{s}_k = \begin{cases} 1, & k \in \mathcal{S}^*, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where S^* corresponds to the indices of g with the K largest entries.

D. MAC Algorithm

The MAC algorithm [13] is the leading method for scalable sparsification and serves as our primary foundation.

It first assumes sets of fixed edges, forming a connected subgraph $\mathcal{E}^f$, and candidate edges $\mathcal{E}^c$ are given, where $\mathcal{E} = \mathcal{E}^f \cup \mathcal{E}^c, \mathcal{E}^f \cap \mathcal{E}^c = \emptyset$. Next, using the Frank-Wolfe Algorithm, it solves Problem 2, with the Laplacian rewritten in terms of the edge sets and selection vector $\boldsymbol{x}$

$$L(\boldsymbol{x}) = \lambda_2(L_f + \sum_{k=1}^{m} \boldsymbol{x}_k L_k^c) \tag{7}$$

where L_f and L_k^C are the Laplacians corresponding to edges in $\mathcal{E}^f$ and edge $e_k \in \mathcal{E}^c$, respectively. Finally, it rounds the solution via Madow Sampling [23].

We target two specific limitations of the MAC framework: its computational cost and its requirement on a user-specified spanning tree "backbone". Our improvements directly address these challenges to enhance the algorithm's efficiency and generality. Specifically, we focus on 1) improving computational efficiency and 2) removing the user-specified backbone requirement.

First, we aim to make MAC faster. The primary computational burden lies in the Frank-Wolfe optimization loop, which is expensive due to the calculation of the Fiedler vector and value (the Fiedler pair) in each iteration. To accelerate MAC, we pursue a two-pronged approach:

1) **Improved Fiedler Pair Calculation** (Section IV): We develop a fast, numerically stable algorithm that leverages the inherent graph structure to calculating the Fiedler vector and value.
2) **Improved Convergence Rate** (Section V): We investigate step size selection for the Frank-Wolfe algorithm to increase its convergence rate, reducing the total number of iterations required

Algorithm 2 Fiedler Pair Calculation

Input: : $L \in \mathbb{R}^{n \times n} \leftarrow$ Laplacian, $\sigma \in \mathbb{R} \leftarrow$ Shift parameter, $k \in \mathbb{N} \leftarrow$: Number of eigenvalues to compute, $\mathbf{x}_0 \in \mathbb{R}^n \leftarrow$: Initial guess vector
Output: : Fiedler Vector q_2, Fiedler Value λ_2

function FIEDLERSHIFTINVERT($L, \sigma, k, \mathbf{x}_0$)
  Step 1: Shifted System Factorization
  $A \leftarrow L - \sigma I$ ▷ Shifted matrix
  $p \leftarrow \text{AMD}(A)$ ▷ Approx. minimum degree [20]
  $R \leftarrow \text{Cholesky}(A(p,p))$ ▷ Cholesky factorization
  $\text{sol}_A(\mathbf{b}) \leftarrow R^{-1}(R^{-\mathsf{T}}(\mathbf{b}(p)))$ ▷ Solves x in $Ax = \mathbf{b}$
  Step 2: Projected Shift-Invert Operator
  $\mathcal{P}(\mathbf{v}) \leftarrow \mathbf{v} - (\mathbf{1}^\top \mathbf{v}/n)\mathbf{1}$ ▷ Projection onto $\mathbf{1}^\perp$
  $\mathcal{T}(\mathbf{v}) \leftarrow \mathcal{P}(\text{sol}_A(\mathcal{P}(\mathbf{v})))$ ▷ Shift-invert operator
  Step 3: Krylov-Schur Method [18]
  $\mathbf{v}_0 \leftarrow \mathcal{P}(\mathbf{x}_0)/\|\mathcal{P}(\mathbf{x}_0)\|$ ▷ Normalized initial vector
  $m \leftarrow \min(2k + 10, n)$ ▷ Subspace dimension
  $(\mathbf{V}_m, \mathbf{H}_m) \leftarrow \text{KrylovSchur}(\mathcal{T}, \mathbf{v}_0, m)$
  Step 4: Eigenvalue Recovery
  $\lambda_i \leftarrow \sigma + 1/\mathbf{H}_{ii}$
  Sort λ_i ascending, take $\lambda_f \leftarrow \lambda_1$
  $\mathbf{x}_f \leftarrow \mathcal{P}(\mathbf{V}_m \mathbf{y}_1)/\|\mathcal{P}(\mathbf{V}_m \mathbf{y}_1)\|$ ▷ Fiedler vector
  return $\mathbf{x}_f, \lambda_f$
end function

Second, the original MAC algorithm requires a predefined "backbone" $\mathcal{E}^f$ to maintain connectivity after rounding (e.g., odometry). We instead consider the case where $\mathcal{E}^f$ is not given or empty, and propose two solutions: first, in Section VI-A we develop a rounding function Π_K that returns a connected graph. Alternatively, in Section VI-B we propose a heuristic method for automatically determining a strong candidate backbone.

IV. Fiedler Value Calculation

While much of the current research on numerical Fiedler-value solvers [19], [24] focus on scalability to large graphs (exceeding 10^6 edges), most graphs found in robotics have between $10^3 \sim 10^5$ edges. At these sizes, we can still utilize the Cholesky Factorization of the *entire* Laplacian under real-time constraints. While it is counterintuitive to tailor our sparsification process to already smaller graphs, it makes our solver well-suited to online, incremental sparsification, where we prune the graph as it grows. Using this observation, we develop a fast Fiedler value calculation routine, detailed in Algorithm 2.

We leverage three structural properties of the Laplacian. First, as the Fiedler Value for a Laplacian Graph is the closest eigenvalue to zero, we apply shift-invert conditioning

[25]. Next, for the inversion, we perform an approximate minimum degree ordering [20] before taking the Cholesky factorization, which, for graphs found in robotics (e.g. pose-graphs), dramatically decreases the number of non-zeros and thus computation time for using our Cholesky factorization. Finally, in our iterative eigenvalue solver we use the fact that the first eigenvalue of our Laplacian will always be zero with an eigenvector of $\mathbf{1}^\mathsf{T}$, and deflate (project onto $\mathbf{1}^\perp$) our candidate eigenvector $\mathbf{v}$ at each iteration. To iteratively solve for the eigenvalues of our shift-inverted system, we utilize the Krylov-Schur algorithm [18], and recover our original Fiedler pair with a cheap sort and projection step.

V. Improving Convergence Rate

We investigated multiple step size line search and Frank-Wolfe variations for improving the convergence rate of MAC.

Previously, MAC utilized a decaying step size at each iteration:

$$\gamma_k = \frac{2}{2+k} \tag{8}$$

Although inexpensive to compute and sharing the same worst-case $O(1/k)$ convergence rate as other step size rules, this simple scheme is often outperformed in practice. In particular, the exact line search and backtracking line search algorithms [26] enjoy linear convergence rates for strongly convex/concave functions with polytope constraints. Although the Fiedler value is not strongly concave, we investigated if the empirical performance gains of both backtracking and exact line search algorithms translate to MAC. The Pair-wise and Away-steps variants of Frank-Wolfe [27] also present similar improvements to the base Frank-Wolfe algorithm. Thus, we implemented MAC for both different line search strategies and Frank-Wolfe variants. Interestingly, as we will discuss in Section VII-B, Frank-Wolfe with a decaying step size is both the cheapest in terms of total time taken, and the most robust in terms of attaining a higher optimized value.

VI. Backbone Specification and Removal

One key assumption of MAC is that a fixed set of edges $\mathcal{E}^f$ is given that connects the graph. We improve on this assumption by 1) proposing a method that automatically specifies informative spanning-tree backbones and 2) removing the need for fixed edges entirely. As a result, no user-specified backbone must be given.

A. Rounding with Connectedness

A simple backbone-free approach is to solve the relaxed edge selection problem (Problem 2) with $\mathcal{E}^f = \emptyset$ and round the solution using the techniques in [13]. However, these rounding methods do not guarantee connectivity, potentially yielding a disconnected graph. To address this, we propose a rounding procedure that ensures connectivity, selects exactly K edges, and maximizes total weight according to the relaxed solution $\boldsymbol{x}^*$. Formally, we aim to solve

$$\begin{aligned} \max_{\mathbf{w}\in\{0,1\}^m} \quad & \mathbf{w}^\top \boldsymbol{x}^* \\ \text{s.t.} \quad & \sum_{i=1}^{m} \mathbf{w}_i = K, \quad \mathcal{G}(\mathcal{V}, \mathcal{E}_\mathbf{w}) \text{ connected}, \end{aligned} \tag{9}$$

where $\mathcal{E}_\mathbf{w}$ is the set of edges indicated by $\mathbf{w}$. This can be solved efficiently by first computing a maximum spanning tree using $\boldsymbol{x}^*$ as weights, then adding the remaining $K-(n-1)$ edges with the highest-weight.

	Dataset	Nodes	Edges	Avg. Solve time (ms)		
				Tracemin	Eigs	Ours
SFM	IMC-gate	5436	216985	274 (-84%)	1820	**68(-96%)**
	BAL-392	28713	128565	189 (-4%)	198	**23 (-88%)**
	Replica-office	1989	63921	332 (719%)	40	**10 (-73%)**
SNL	sphere2500-snl	2500	4949	12 (291%)	**3.3**	4.8 (45%)
	city10k-snl	10000	20687	46 (116%)	21	**15(-27%)**
	grid3D-snl	8000	22236	104 (64%)	63	**45(-28%)**
PGO	sphere2500	2500	4949	13 (143%)	5.4	**4.7 (-13%)**
	city10000	10000	20687	46 (199%)	15	**15 (-0.4%)**
	grid3D	8000	22236	88 (-4%)	92	**44 (-51%)**
	tiers	9769	17553	143 (304%)	35	**17 (-50%)**

TABLE I: **Fiedler Solve Time**. Frank–Wolfe is run on sparsified SfM, PGO, and SNL graphs using *Tracemin*, *Eigs*, and *Our* solver; the time to find the Fiedler Value are reported with percent change ($\frac{t-t_{Eigs}}{t_{Eigs}}$) against *Eigs* (faster, slower), **bold** marks the fastest.

B. Spectrally-Informed Spanning Tree

Alternatively, rather than using a given set of connected fixed edges, we can automate fixing edges before solving Problem 2. However, multiple connected subgraphs or spanning trees may exist for a given graph. Identifying the optimal one requires solving the original unrelaxed problem (Eq. 1), which is NP-hard; consequently, we rely on heuristic methods to guide the selection.

A natural heuristic for this task is the effective resistance distance, a graph-theoretic quantity that measures the importance of an edge to overall connectivity. Formally, for an edge (i,j), the effective resistance is defined as

$$R_{ij} = (\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} L^\dagger (\mathbf{e}_i - \mathbf{e}_j) \tag{10}$$

where $\mathbf{e}_i \in \mathbb{R}^m$ is the standard basis vector with 1 as its i^{th} component, and $L^\dagger$ is the Moore-Penrose pseudo-inverse of L [28]. Effectively, edges with high effective resistance are bottlenecks—critical connections whose absence would severely disrupt flow. This can arise from an inherently weak connection (low weight) or being a bridge between otherwise disconnected components. We wish to choose a spanning tree that specifically contains edges that are important because of the topology of the graph, and thus we weight each edge's effective resistance by the edge's weight, that is $w_{ij}^R = R_{ij} w_{ij}$. Then, we determine $\mathcal{E}^f$ as the edges of the maximum spanning tree with weights w_{ij}^R.

VII. Experimental Results

We evaluated our algorithms on variety of datasets representing three common estimation problems: Pose Graph Optimization (PGO), Sensor Network Localization (SNL),

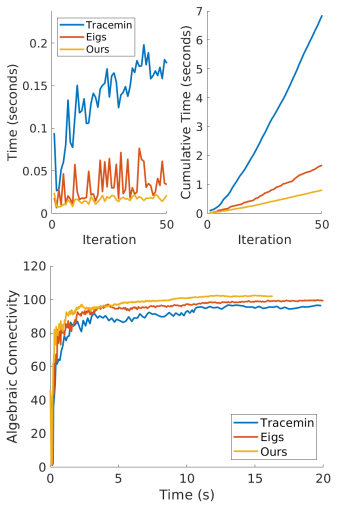

Fig. 3: **TIERS** [29] dataset sparsified to 80% edges via Frank-Wolfe. Notably, each iteration takes significantly less time when using our algorithm due to the faster Fiedler Pair solves (**Top**), resulting in higher algebraic connectivity λ_2 in shorter time (**Bottom**).

and Structure from Motion (SfM). Experiments were conducted on both synthetic and real datasets To generate sensor network localization graphs, pose graphs were converted by replacing poses with landmarks, and transformations as range measurements. We seek to answer several questions:

- **Q1**: Can MAC's runtime be improved through specialized eigenvalue solvers?
- **Q2**: How do different types of graphs found in robotics affect MAC?
- **Q3**: Do the advanced variants of Frank-Wolfe result in fewer iterations to solve Problem 2?
- **Q4**: How can we automatically choose a high quality "backbone"?
- **Q5**: How does the Fiedler value of a graph produced by our rounding scheme compare to that of a graph with a pre-specified backbone?

A. Fiedler Value Calculation

To evaluate Algorithm 2, we compared our algorithm (*Ours*) against two established solvers: the Tracemin-Fiedler algorithm (*Tracemin*) with Cholesky preconditioning [19], and `eigs`, the Krylov–Schur method (*Eigs*) available in MATLAB. Our setup began by running MAC [13] for 50 iterations on each dataset (presented: IMC-gate [30], BAL-392 [31], Replica [32], Sphere, Grid [33], and City [34]) and saving the Laplacians produced at every iterate. We then applied each solver to these Laplacians and recorded runtimes.

To test our hypothesis that eigenvalue computation is the dominant cost in MAC, we next integrated each solver into a Frank–Wolfe routine. The results, summarized in Figure 3, show that our algorithm again outperformed both Tracemin and Eigs, confirming that our approach substantially reduces the overall runtime (**Q1**). Additionally, as seen in Table I, our algorithm consistently demonstrates faster Fiedler Value

		Line Search Type			
	Metric	Naive	Exact	Backtracking	PFW-Exact
Intel	(↓) Cum. time (s)	**0.138**	1.064	0.140	1.006
	(↑) λ_2	**3.260**	2.929	2.753	3.021
	(↓) Avg time (s)	0.003	0.024	**0.003**	0.022
TIERS	(↓) Cum. time (s)	**1.153**	19.317	1.889	14.190
	(↑) λ_2	**63.573**	49.294	42.718	43.699
	(↓) Avg time (s)	**0.026**	0.429	0.042	0.315
MRCLAM7	(↓) Cum. time (s)	**2.342**	32.551	3.210	35.235
	(↑) λ_2	**0.873**	0.829	0.743	0.794
	(↓) Avg time (s)	**0.052**	0.723	0.065	0.782

TABLE II: **Frank-Wolfe Variants**. We report cumulative runtime, the resulting Fiedler value (λ_2), and average runtime per iteration on a single run of each dataset. **Bold** indicates the best solver. Surprisingly, the simple decaying stepsize (*Naive*) consistently outperforms the more advanced variants

solve time across all three different types of graphs (**Q2**).

B. Line Search

We next examined the effect of line-search strategies and Frank–Wolfe variants on several real datasets, including TIERS [29], MR.CLAM7 [35], and Intel [36]. Alongside the simple (*Naive*) decaying step size (8) which requires no additional objective evaluations, we implemented an approximate exact line search (*Exact*) using the `fminbnd` function in MATLAB. Since this approach can be computationally expensive, we also considered an approximate backtracking line search (*Backtracking*) following [26]. Finally, we implemented the pairwise Frank–Wolfe algorithm (*PFW-Exact*) [27]. Of note, across our datasets shown here, the away-step is never taken for an away-steps Frank-Wolfe and thus we do not include it in our results. The results, shown in Table II and Figure 4, reveal that the decaying step size not only achieved the lowest runtime but also produced the highest Fiedler value. In contrast, the other methods took good initial iterates, but stalled later at suboptimal values. Thus, the naive Frank-Wolfe proves to be both faster and produce better iterates than the more advanced variants (**Q3**).

We hypothesize this behavior arises from the nonsmoothness of the Fiedler value. When the Laplacian has repeated Fiedler eigenvalues, the gradient is undefined; although the supergradient remains a valid descent direction, it may carry little useful information. As noted, away-step Frank–Wolfe rarely takes away steps, suggesting the early iterates quickly reach a "good" set of faces of the constraint set. However, moving from these faces to the optimal ones becomes difficult with a conservative line search along an arbitrary supergradient. In contrast, a decaying step size induces a zig-zagging effect that, while typically undesirable, enables exploration of additional faces.

C. Backbone

Our final set of experiments compared rounding and spanning-tree strategies. We considered several approaches varying backbone and rounding schemes. First, the *MST*

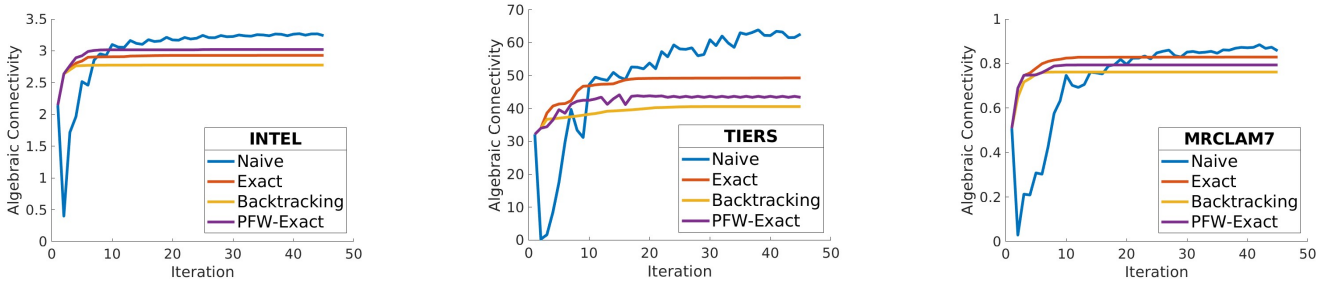

Fig. 4: **Algebraic Connectivity vs Iteration**: Each graph corresponds to the runs in Table II: As seen, the *Exact, Backtracking, and PFW-Exact* variants stall in optimization progress in a few iterates. While the *Naive* step size strategy initially takes a poor step, it does not suffer from this stalling behavior.

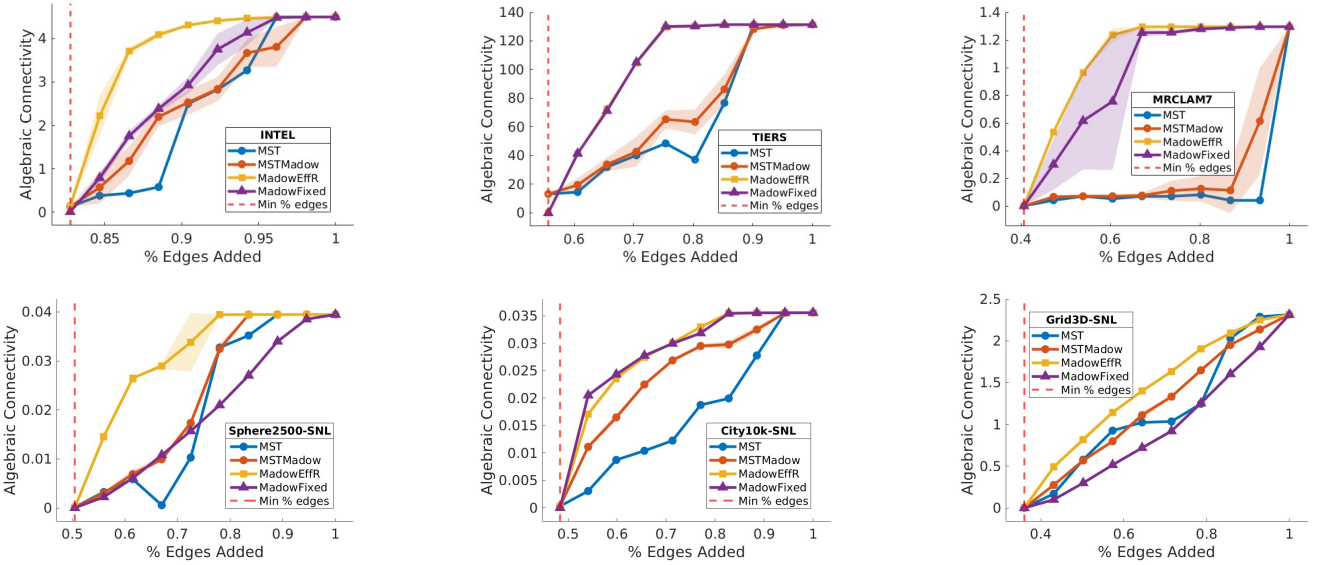

Fig. 5: **Algebraic Connectivity vs Edge Budget**. Graphs of λ_2 vs % edges added. For Madow rounding strategies, shaded error bars indicate the ± 1 standard deviation bound around the mean cost. Our heuristic backbone (*MadowEffR*) generally outperforms all other strategies, while backbone-free approaches (*MST*, *MSTMadow*) generally perform the worst.

strategy, described in Section VI-A, solves using an empty fixed edge set. A second variant, *MST-Madow*, augments *MST* by using Madow sampling [23] to choose the $K-(n-1)$ edges after finding a max spanning tree. We also evaluated *MadowEffR*, which combines the spectral effective resistance backbone from Section VI with Madow sampling. For comparison with prior work, we reproduced the odometry-based fixed backbone of [13] (*MadowFixed*), again coupled with Madow rounding. Because Madow rounding is randomized, each experiment was repeated ten times, and we report both the mean and one-standard-deviation bounds. A subtlety of these comparisons is that MST-based methods must solve a higher-dimensional Frank–Wolfe problem, since all edges are initially treated as unfixed.

As shown in Figure 5, the effective-resistance backbone generally outperforms the alternatives, with only the odometry backbone producing similar performance on select datasets. Thus, our backbone specification heuristic empirically returns stronger sets of fixed edges (**Q4**). Surprisingly, the MST-based methods perform poorly across datasets, despite explicitly selecting max-spanning-tree weights from the solution of the relaxed problem (Problem 2) (**Q5**). We hypothesize this is because the relaxed problem has many symmetries (see [13, Sec. V.A]), and thus spreads the edge selection weights across many potential subgraphs. When rounding on these edge selection weights, we return a mediocre graph. In contrast, by specifying a connected subgraph *before* optimization, we break this symmetry.

VIII. Conclusion

In this paper, we propose several improvements to the Maximizing Algebraic Connectivity (MAC) algorithm. In particular, we significantly enhance computational efficiency by accelerating the Fiedler value computation through a specialized solver. We also uncover a surprising result: the naive step size strategy is not only the fastest approach, but it also achieves higher Fiedler values than more sophisticated alternatives. Finally, we present methods to guarantee connectedness in the returned graph and introduce a spectral-based heuristic that consistently outperforms the original MAC algorithm.

One exciting direction for future work lies in understanding why the naive step size strategy outperforms more advanced Frank–Wolfe variants. A deeper analysis of the Fiedler value optimization landscape may yield insights that inspire more effective first-order solvers. Another promising avenue is to better understand why specifying a backbone *a priori* leads to markedly improved results and whether we can formally characterize the error introduced by such a specification.

Overall, our findings not only improve the state of the art in maximizing algebraic connectivity but also highlight several intriguing theoretical questions. By bridging the gap between practical heuristics and rigorous analysis, future work has the potential to further advance both the efficiency and reliability of connectivity optimization algorithms.

REFERENCES

[1] Y. Chen, S. Huang, L. Zhao, and G. Dissanayake, "Cramér–rao bounds and optimal design metrics for pose-graph slam," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 627–641, 2021.

[2] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 95–125, 2019.

[3] D. Mosk-Aoyama, "Maximum algebraic connectivity augmentation is np-hard," *Oper. Res. Lett.*, vol. 36, no. 6, p. 677–679, Nov. 2008. [Online]. Available: https://doi.org/10.1016/j.orl.2008.09.001

[4] K. J. Doherty, D. M. Rosen, and J. J. Leonard, "Spectral measurement sparsification for pose-graph slam," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 01–08.

[5] N. Carlevaris-Bianco and R. M. Eustice, "Long-term simultaneous localization and mapping with generic linear constraint node removal," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1034–1041.

[6] G. Kurz, M. Holoch, and P. Biber, "Geometry-based graph pruning for lifelong slam," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3313–3320.

[7] N. Carlevaris-Bianco and R. M. Eustice, "Conservative edge sparsification for graph slam node removal," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 854–860.

[8] D. Thorne, N. Chan, Y. Ma, C. S. Robison, P. R. Osteen, and B. T. Lopez, "Submodular optimization for keyframe selection & usage in slam," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 5033–5039.

[9] L. Carlone and S. Karaman, "Attention and anticipation in fast visual-inertial navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3886–3893.

[10] Z. Dong, J. Pflueger, L. Jung, D. Thorne, P. R. Osteen, C. S. Robison, B. T. Lopez, and M. Everett, "Lidar inertial odometry and mapping using learned registration-relevant features," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 359–366.

[11] G. Moreira, M. Marques, and J. P. Costeira, "Fast pose graph optimization via krylov-schur and cholesky factorization," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 1897–1905.

[12] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973. [Online]. Available: http://eudml.org/doc/12723

[13] K. Doherty, A. Papalia, Y. Huang, D. Rosen, B. Englot, and J. Leonard, "Mac: Graph sparsification by maximizing algebraic connectivity," *arXiv preprint arXiv:2403.19879*, 2024.

[14] J. Nam, S. Hyeon, Y. Joo, D. Noh, and H. Shim, "Spectral trade-off for measurement sparsification of pose-graph slam," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 723–730, 2024.

[15] S. Boyd, "Convex optimization of graph laplacian eigenvalues," 2006.

[16] P. Kaveti, M. Giamou, H. Singh, and D. M. Rosen, "Oasis: Optimal arrangements for sensing in slam," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 13 818–13 824.

[17] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[18] G. W. Stewart, "A krylov–schur algorithm for large eigenproblems," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 601–614, 2002. [Online]. Available: https://doi.org/10.1137/S0895479800371529

[19] M. Manguoglu, E. Cox, F. Saied, and A. Sameh, "Tracemin-fiedler: a parallel algorithm for computing the fiedler vector," in *Proceedings of the 9th International Conference on High Performance Computing for Computational Science*, ser. VECPAR'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 449–455.

[20] P. R. Amestoy, T. A. Davis, and I. S. Duff, "Algorithm 837: Amd, an approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, p. 381–388, Sep. 2004. [Online]. Available: https://doi.org/10.1145/1024074.1024081

[21] D. M. Rosen, "Accelerating certifiable estimation with preconditioned eigensolvers," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 507–12 514, 2022.

[22] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 6605–6611.

[23] W. G. Madow, "On the theory of systematic sampling, ii," *The Annals of Mathematical Statistics*, vol. 20, no. 3, pp. 333–354, 1949.

[24] Y. Hu and J. Scott, "Hslmc73: A fast multilevel fiedler and profile reduction code," 01 2003.

[25] T. Ericsson and A. Ruhe, "The spectral transformation lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems," *Mathematics of Computation*, vol. 35, no. 152, pp. 1251–1268, 1980. [Online]. Available: http://www.jstor.org/stable/2006390

[26] F. Pedregosa, G. Negiar, A. Askari, and M. Jaggi, "Linearly convergent frank-wolfe with backtracking line-search," in *International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, 2020.

[27] S. Lacoste-Julien and M. Jaggi, "On the global linear convergence of frank-wolfe optimization variants," *Advances in neural information processing systems*, vol. 28, 2015.

[28] D. J. Klein and M. Randic, "Resistance distance," *Journal of Mathematical Chemistry*, vol. 12, pp. 81–95, 1993. [Online]. Available: https://api.semanticscholar.org/CorpusID:16382100

[29] X. Yu, I. Catalano, P. T. Morón, S. Salimpour, T. Westerlund, and J. P. Queralta, "Fusing odometry, uwb ranging, and spatial detections for relative multi-robot localization," *arXiv preprint arXiv:2304.06264*, 2023.

[30] A. Chow, E. Trulls, HCL-Jevster, K. M. Yi, lcmrll, old ufo, S. Dane, tanjigou, WastedCode, and W. Sun, "Image matching challenge 2023," https://kaggle.com/competitions/image-matching-challenge-2023, 2023, kaggle.

[31] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *European conference on computer vision*. Springer, 2010, pp. 29–42.

[32] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[33] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3d slam: A survey on rotation estimation and its use in pose graph optimization," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4597–4604.

[34] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[35] K. Y. Leung, Y. Halpern, T. D. Barfoot, and H. H. Liu, "The utias multi-robot cooperative localization and mapping dataset," *Int. J. Rob. Res.*, vol. 30, no. 8, p. 969–974, Jul. 2011. [Online]. Available: https://doi.org/10.1177/0278364911398404

[36] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *The International Journal of Robotics Research*, vol. 33, pp. 965 – 987, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:39883062