Pose-graph SLAM sparsification using factor descent

Joan Vallvé, Joan Solà, Juan Andrade-Cetto

Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Llorens Artigas 4-6, 08028, Barcelona, Spain

Abstract

Since state of the art simultaneous localization and mapping (SLAM) algorithms are not constant time, it is often necessary to reduce the problem size while keeping as much of the original graph's information content. In graph SLAM, the problem is reduced by removing nodes and rearranging factors. This is normally faced locally: after selecting a node to be removed, its Markov blanket sub-graph is isolated, the node is marginalized and its dense result is sparsified. The aim of sparsification is to compute an approximation of the dense and non-relinearizable result of node marginalization with a new set of factors. Sparsification consists on two processes: building the topology of new factors, and finding the optimal parameters that best approximate the original dense distribution. This best approximation can be obtained through minimization of the Kullback-Liebler divergence between the two distributions. Using simple topologies such as Chow-Liu trees, there is a closed form for the optimal solution. However, a tree is oftentimes too sparse and produces bad distribution approximations. On the contrary, more populated topologies require nonlinear iterative optimization. In the present paper, the particularities of pose-graph SLAM are exploited for designing new informative topologies and for applying the novel factor descent iterative optimization method for sparsification. Several experiments are provided comparing the proposed topology methods and factor descent optimization with state-of-the-art methods in synthetic and real datasets with regards to approximation accuracy and computational cost.

Keywords: Mobile robotics, SLAM, sparsification, factor recovery, topology

1. Introduction

Simultaneous localization and mapping (SLAM) is a well-studied problem in mobile robotics. It consists in building a map of the environment while localizing the robot in it. It is a challenging problem in many ways. One of its main challenges is its increasing resource demands: the longer the experiment, the larger the problem to solve. Efforts to address this challenge have been focused mainly in two directions, either reducing the computational complexity of the algorithms, or tackling the problem size.

Several improvements have been made to reduce the time complexity of the algorithms, such as the iSAM2 and SLAM++ methods [2, 3], though they have not yet achieved constant-time, constant-memory operation. Thus the issue of ever-increasing problem size calls for sub-optimal strategies to maintain a tractable problem size while keeping as much information as possible.

One of the simplest SLAM approaches aimed at limiting problem size growth consists in considering a temporal or spatial window and discarding the landmarks and/or poses that lie outside this window. This implies giving up loop closures, such as in visual and visual-inertial odometry [4, 5]. Alternatively, hierarchical graph structures can also be considered to keep the problem tractable [6]. The graphs higher up in the hierarchy represent marginalized small sub-graphs and if the error is low enough, only part of the problem is solved at each level in the hierarchy. Another alternative is to reduce the growth of the problem by marginalizing out poses that are close to previously visited ones [7].

Some other methods propose to just discard some information even before it is considered for optimization. For instance, Pose SLAM [8] only adds new robot poses and observations depending on its entropy-based information content. Chouldhary et al. [9] on the other hand propose an entropy-based method to decide which landmarks should be discarded in a trade off between memory footprint and accuracy.

In any case, one important characteristic that must be preserved in any strategy that limits problem size growth is to maintain the network sparsity. The factor graph that represents the network of geometric constraints (factors) linking problem variables (nodes) with sensor measurements is by construction sparse in the number of factors.

[☆]This work has been supported by the Spanish Ministry of Economy and Competitiveness Project EB-SLAM [DPI2017-89564-P]; by the EU H2020 Project LOGIMATIC [H2020-Galileo-2015-1-687534]; and by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI [MDM-2016-0656].

[☆☆]This paper is an extended and revised version of the work of Vallvé et al. [1].

Email addresses: jvallve@iri.upc.edu (Joan Vallvé), cetto@iri.upc.edu (Juan Andrade-Cetto), jsola@iri.upc.edu (Joan Solà)

Most graph-SLAM methods take profit of this sparsity to speed up the computation of the optimal solution.

Furthermore, SLAM being a non-linear problem, those methods that accurately handle relinearization improve the accuracy of the solution.

Normally, the only way of reducing the problem size without loss of information is marginalization. However, marginalization results in a densely connected graph and does not allow for relinearization. These respectively undermine efficiency and accuracy.

Sparsification is known as the problem of finding the best sparse and relinearizable approximation of the result of a marginalization. Kretzschmar and Stachniss [10] present an information-theoretic compression method for pose-graph SLAM that selects the nodes containing the most informative laser scans. They find the subset of measurements that maximize the mutual information of the map for that subset. More recently, the problem has been posed as finding the sparse approximation that minimizes the Kullback-Liebler divergence (KLD) with respect to the dense distribution resulting from marginalization [11, 12, 13]. To solve it, first, the topology of the new factors is built, and finally the optimal mean and covariance of each new factor is computed. While there is a closed form for the simplest topology, *e.g.* the Chow-Liu tree (CLT), iterative optimization is needed for more populated topologies.

The tree topology is the sparsest alternative but it can encode only a small part of the original dense information. To reach more accurate approximations, more populated topologies are required. For this reason, in this paper we focus on designing populated topologies and using iterative optimization for sparsification.

In [1, 14] we introduced factor descent optimization. Given a non-dense factor topology, factor descent iteratively optimizes each of the factors leaving fixed the rest. For each factor, its optimal parameters (mean and information matrix) in terms of KLD are found given the rest of topology factors' parameters.

In this paper, we explore the application of factor descent in the pose-graph SLAM case. Pose-graph SLAM problems are those made only of nodes representing poses, and factors corresponding to relative pose measurements. Its nature can be taken into account in the whole sparsification process. Indeed, the generalized formulation introduced in [1], and completed in [14], is suited to pose-graph SLAM. We found that in such case, some conditions can not take place and the general formulation can be simplified.

The present work focuses also on the topology of the new factors, which was out of the scope in our previous work. The amount of factors that constitute this topology, what we call topology population, controls how dense the sparsification approximation is. While a more populated topology will approximate the dense marginalization better, a less populated one will be faster to compute. In order to handle this trade-off, we propose a new policy to set the topology population. This policy is intuitive, allowing the end user to set the topology population in a reasoned way. Finally, considering the pose-graph SLAM specificities, we devise three new methods to build populated topologies able to encode most of the information of the dense distribution resulting from marginalization.

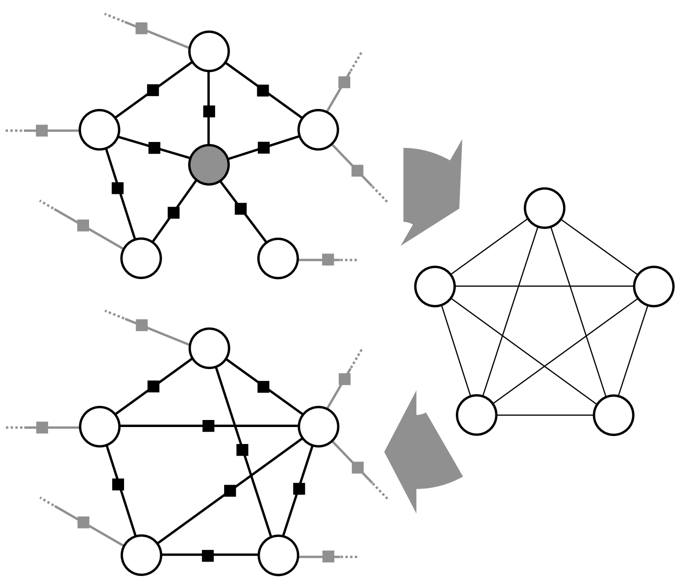
Figure 1: Example of marginalization and sparsification of a node (gray). After cropping a local problem, marginalization produces a dense and non-relinearizable sub-graph (right figure). This subgraph is replaced in the original graph with a sparse approximation (bottom).

Summarizing, the contributions of this paper are:

- The application of the Factor Descent formulation to the specific case of pose-graph SLAM is presented. It results in a simplification of the general case formulation presented in our previous work.
- We propose a novel policy to set the topology population that is more intuitive since it is based on the desired sparsity of the new topology.
- Three new methods to build the topology are presented and compared with the state-of-the-art, in terms of accuracy and computational cost.

2. Node removal and sparsification in pose-graph SLAM

In graph-based SLAM methods, the problem is represented as the optimization of a graph, made of a set of nodes (i.e. variables) and a set of factors (i.e. geometrical constraints according to sensor measurements). The state $\mathbf{x}$ includes variables representing poses of the vehicle along its trajectory and/or some map representation. Each factor expresses the discrepancy or error $\mathbf{e}$ between a measurement $\mathbf{z}$ and its expectation,

$$\mathbf{e}(\mathbf{x}) = h(\mathbf{x}) - \mathbf{z} + \mathbf{v}, \qquad \mathbf{v} \sim \mathcal{N}(0, \mathbf{\Omega}^{-1}) \tag{1}$$

being $h(\mathbf{x})$ the sensor's measurement model, $\mathbf{x}$ the state estimate at the current iteration, and $\mathbf{v}$ the associated measurement Gaussian noise.

The mean conditional distribution of $\mathbf{x}$ given the measurements is solved iteratively by minimizing the Mahalanobis squared norm of all linearized errors

$$\Delta\mathbf{x}^* = \arg\min_{\Delta\mathbf{x}} \sum_k \|h_k(\mathbf{x}) - \mathbf{z}_k + \mathbf{J}_k\Delta\mathbf{x}\|^2_{\mathbf{\Omega}_k^{-1}} \qquad (2)$$

being $\mathbf{J}_k$ the Jacobian of the k-th measurement[1].

Imposing a null derivative of the cost in (2) w.r.t $\Delta\mathbf{x}$, the optimal step $\Delta\mathbf{x}^*$ is found and used to update the estimate.

Current methods for solving for $\Delta\mathbf{x}^*$ use Cholesky [16, 17, 3] or QR [18, 19, 2] matrix factorizations. Important speed-ups are obtained with incremental methods [19, 2, 17, 3], which update the problem directly on the factorized matrix only relinearizing it periodically or partially.

Reducing the problem size in graph SLAM is usually approached in two steps: node marginalization and sparsification (see Fig. 1). These two processes do not necessarily have to be immediately consecutive. Sparsification can be postponed depending on the availability of computational resources as in [12].

Once a node is selected for removal, the process is faced locally. The process only involves the immediate surroundings consisting of the node's Markov blanket (all nodes at distance 1) and all its intra-factors (the factors involving only nodes in the Markov blanket). Optionally, this cropped problem can be solved in order to relinearize all intra-factors before proceeding. This yields slightly better results especially in on-line cases [13].

Marginalization of the selected node is performed via Schur complement. This marginalization can be understood as adding a dense factor that substitutes all intra-factors that involve the removed node. This new dense factor has no measurement model associated to it; hence, its error cannot be re-evaluated, and re-linearization is not possible, thus undermining the accuracy of the solution. Moreover, despite there is no information loss in node marginalization, the induced density in the blanket undermines SLAM solvers efficiency since they rely on sparsity.

This brings us to the second step, sparsification, whose goal is to approximate the dense distribution $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, resulting from node marginalization, with a sparse distribution $q(\mathbf{x}) \sim \mathcal{N}(\breve{\boldsymbol{\mu}}, \breve{\mathbf{\Sigma}})$ defined by a new set of (relinearizable) factors. This is usually split in two phases: topology building and factor recovery. The topology defines the arrangement between the Markov blanket nodes and the new set of factors, each factor with a measurement model (see Sec. 4). Given the set of new factors of the chosen topoogy, factor recovery consists on computing their mean and information that best approximate the original distribution.

2.1. Factor recovery through KLD minimization

Given the topology, the aim of factor recovery is finding its factors' means $\breve{\mathbf{z}}_k$ and information $\breve{\mathbf{\Omega}}_k$ that minimize the relative entropy (KLD) of the sparse distribution $q(\mathbf{x})$ w.r.t. the dense one $p(\mathbf{x})$. For Gaussian distributions, the KLD reads,

$$D_{KL} = \frac{1}{2}\Big(\langle\breve{\mathbf{\Lambda}}, \mathbf{\Sigma}\rangle - \ln|\breve{\mathbf{\Lambda}}\mathbf{\Sigma}| + \|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\mathbf{\Lambda}}^{-1}} - d\Big), \qquad (3)$$

where $\langle\cdot,\cdot\rangle$ denotes the matrix inner product and $\breve{\mathbf{\Lambda}} = \breve{\mathbf{\Sigma}}^{-1}$ is the information matrix of $q(\mathbf{x})$, given by

$$\breve{\mathbf{\Lambda}} = \breve{\mathbf{J}}^\top\breve{\mathbf{\Omega}}\breve{\mathbf{J}} = \begin{bmatrix}\dots & \breve{\mathbf{J}}_k^\top & \dots\end{bmatrix} \begin{bmatrix}\ddots & & \\ & \breve{\mathbf{\Omega}}_k & \\ & & \ddots\end{bmatrix} \begin{bmatrix}\vdots \\ \breve{\mathbf{J}}_k \\ \vdots\end{bmatrix}$$

being $\breve{\mathbf{J}}_k$ the Jacobian of the k-th measurement model.

Setting all measurements' means $\breve{\mathbf{z}}_k$ as the expected measurements of the dense distribution, *i.e.* $\breve{\mathbf{z}}_k = h_k(\boldsymbol{\mu})$, the Mahalanobis norm term $\|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\mathbf{\Lambda}}}$ becomes null, and to find $\breve{\mathbf{\Omega}}$ that minimizes (3), the rest of the expression can be simplified as follows. The dimension d of both distributions is constant w.r.t the information of all measurements stored in $\breve{\mathbf{\Omega}}$. The log term can be decomposed as $\ln|\breve{\mathbf{\Lambda}}\mathbf{\Sigma}| = \ln|\breve{\mathbf{\Lambda}}| + \ln|\mathbf{\Sigma}|$, whose second term is also constant w.r.t. $\breve{\mathbf{\Omega}}$. Considering the above, the factors' information that minimizes (3), can be obtained from the constrained problem

$$\begin{aligned}\breve{\mathbf{\Omega}}^* = \arg\min_{\breve{\mathbf{\Omega}}} & \Big(\langle\breve{\mathbf{J}}^\top\breve{\mathbf{\Omega}}\breve{\mathbf{J}}, \mathbf{\Sigma}\rangle - \ln|\breve{\mathbf{J}}^\top\breve{\mathbf{\Omega}}\breve{\mathbf{J}}|\Big) \\ \text{s.t.} \quad & \breve{\mathbf{\Omega}} \in \mathcal{D}, \breve{\mathbf{\Omega}} \succ 0\end{aligned} \qquad (4)$$

where $\mathcal{D}$ refers to the set of block-diagonal matrices.

Sometimes the dense problem has a rank-deficient information matrix $\mathbf{\Lambda}$, and the covariance matrix $\mathbf{\Sigma}$ is not defined. In such cases, a projection $\mathbf{\Lambda} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ such that $\mathbf{D}$ is invertible can be applied. Then, the formulation derived for (4) holds by substituting

$$\begin{aligned}\breve{\mathbf{J}} &\mapsto \breve{\mathbf{J}}\mathbf{U} \\ \mathbf{\Sigma} &\mapsto \mathbf{D}^{-1}.\end{aligned} \qquad (5)$$

This is always the case in pose-graph SLAM problems since all factors correspond to relative pose measurements. The projection $\mathbf{U}$ can be computed by re-parametrizing the problem to relative poses w.r.t an arbitrarily chosen node [12, 11] or using a rank-revealing eigen decomposition [13].

[1]In case of manifolds, equation (1) and the squared Mahalanobis norm in (2) become $\mathbf{e}(\mathbf{x}) = h(\mathbf{x}) \ominus \mathbf{z} \oplus \mathbf{v}$ and $\|h_k(\mathbf{x}) \ominus \mathbf{z}_k + \mathbf{J}_k\Delta\mathbf{x}\|^2_{\mathbf{\Omega}_k^{-1}}$, respectively, with $\mathbf{J}_k = \partial(h_k(\mathbf{x}) \ominus \mathbf{z}_k)/\partial\Delta\mathbf{x}$. The operators $\oplus$ and $\ominus$ indicate addition and subtraction on a manifold, as described in [15].

2.2. Factor recovery in closed form

Certain topologies admit a closed form solution to (4). As proven in [13], when $\breve{\mathbf{J}}$ is invertible, imposing a null derivative of (4) w.r.t. all factor information matrices yields

$$\breve{\boldsymbol{\Omega}}_k^* = (\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}. \tag{6}$$

In the case of a tree topology in pose-graph SLAM, using a projection as (5), the Jacobian is invertible thus the closed form is suitable. However, as has been said and will be illustrated in the results, this topology is often too sparse to accurately approximate the exact dense distribution.

2.3. Factor recovery via iterative optimization

Other topologies with non-invertible Jacobian $\breve{\mathbf{J}}$ do not admit a closed form solution and the problem has to be solved using iterative optimization. The state-of-the-art literature proposes two different optimization algorithms: Interior Point (IP) and Limited-memory Projected Quasi-Newton (PQN) [20]. IP includes the positive definiteness constraint in the cost function (4) as a log barrier

$$\langle \breve{\mathbf{J}}^\top \breve{\boldsymbol{\Omega}} \breve{\mathbf{J}}, \boldsymbol{\Sigma} \rangle - \ln |\breve{\mathbf{J}}^\top \breve{\boldsymbol{\Omega}} \breve{\mathbf{J}}| - \rho \ln |\breve{\boldsymbol{\Omega}}|. \tag{7}$$

A stricter constraint can be applied instead of the log barrier term to guarantee conservativeness [12]

$$\langle \breve{\mathbf{J}}^\top \breve{\boldsymbol{\Omega}} \breve{\mathbf{J}}, \boldsymbol{\Sigma} \rangle - \ln |\breve{\mathbf{J}}^\top \breve{\boldsymbol{\Omega}} \breve{\mathbf{J}}| - \rho \ln |\boldsymbol{\Lambda} - \breve{\mathbf{J}}^\top \breve{\boldsymbol{\Omega}} \breve{\mathbf{J}}|. \tag{8}$$

The Interior Point method consists of two nested loops. For each log barrier parameter value ρ_i, the problem is solved in the inner loop. After convergence of the inner loop, the outer loop decreases the log barrier, $\rho_{i+1} = \beta \rho_i$ (with $\beta \in (0,1)$) and the inner loop is solved again. The optimization ends when the outer loop reaches a ρ_i value *close enough* to 0, i.e. the solution minimizes the cost function (7) (or (8)) without the contribution of the constraint term. In [13], the inner loop is solved with Newton's method using the gradient and Hessian of the cost function.

Setting the IP parameters β and ρ_0 as well as the inner loop's end conditions is a trade off between convergence speed and robustness to divergence. The IP initial guess must satisfy the constraint, i.e. it must be positive definite. To ensure that the positive definiteness constraint remains satisfied at all times, the contributions to the gradient and the Hessian of the KLD terms and the log barrier term have to be balanced. Relaxing the inner loop end conditions or enhancing the decrease factor β lead to a lower contribution of the log barrier term. This speeds up the method but may converge to a non positive definite result. Proper tuning of these parameters is oftentimes problem-dependent.

On the other hand, PQN does not require the Hessian (it still needs the gradient), nor a feasible initial guess. The positive definiteness constraint is accomplished through the projection $\mathcal{P}(\breve{\boldsymbol{\Omega}})$ onto the positive semi-definite subspace, by setting all negative eigenvalues to zero,

$$\mathcal{P}(\breve{\boldsymbol{\Omega}}) = \mathbf{V} \operatorname{diag}(\max\{0, \lambda_i\}) \mathbf{V}^\top, \tag{9}$$

being $\breve{\boldsymbol{\Omega}} = \mathbf{V} \operatorname{diag}(\lambda_i) \mathbf{V}^\top$ the Eigen decomposition.

PQN has much slower convergence than IP, but it can be initialized closer to the optimal solution. In [21], an initial guess based on the off-diagonal blocks of the dense information matrix is proposed for topologies with factors that involve two nodes

$$\breve{\boldsymbol{\Omega}}_k = \mathbf{J}_{k_1}^{-\top} \boldsymbol{\Lambda}_{k_1,k_2} \mathbf{J}_{k_2}^{-1} \tag{10}$$

being k_1, k_2 the two nodes involved in the factor k (so $\mathbf{J}_{k_1}, \mathbf{J}_{k_2}$ are the non-zero blocks of $\mathbf{J}_k$) and being $\boldsymbol{\Lambda}_{k_1,k_2}$ the off-diagonal block corresponding to the involved nodes. Such initial guess is normally not symmetric nor positive semi-definite, and one usually takes its closest symmetric positive semi-definite approximation [22]. Therefore, since this initial guess is normally a semi-definite positive guess, it cannot be used in the IP method.

Algorithm 1 Factor descent sparsification

```
Input: Dense mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, topology $\mathcal{Z}$
  // Precompute constant variables
  for $\mathbf{z}_k \in \mathcal{Z}$ do
    $\mathbf{J}_k \leftarrow$ evaluateJacobian($\mathbf{z}_k, \boldsymbol{\mu}$)
    $\boldsymbol{\Phi}_k \leftarrow (\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}$
  end for
  k := 1
  while not endConditions() do
    // Compute the information of the rest of factors
    $\boldsymbol{\Upsilon}_k \leftarrow \sum_{j \neq k} \breve{\mathbf{J}}_j^\top \breve{\boldsymbol{\Omega}}_j \breve{\mathbf{J}}_j$
    // k-th factor descent
    $\breve{\boldsymbol{\Omega}}_k \leftarrow \boldsymbol{\Phi}_k - (\breve{\mathbf{J}}_k \breve{\boldsymbol{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1}$
    // Ensure positive semi-definite solution
    if $\breve{\boldsymbol{\Omega}}_k \prec 0$ then
      $\mathbf{V}, \boldsymbol{\lambda} \leftarrow$ eigenDecomposition($\breve{\boldsymbol{\Omega}}_k$)
      $\breve{\boldsymbol{\Omega}}_k \leftarrow \mathbf{V}$diag(max(0, $\boldsymbol{\lambda}$))$\mathbf{V}^\top$
    end if
    // Cycle for all factors
    k++
    if k > N then
      k := 1
    end if
  end while
```

3. Factor recovery with factor descent optimization

Factor descent sparsification (FD) is a novel optimization method for solving (4) inspired in coordinate descent optimization. FD is a cyclic block-coordinate descent method; each step of the cycle consists in solving for a (small) block of variables, those defining one factor's information matrix $\breve{\boldsymbol{\Omega}}_k$, while fixing the rest.

Consider a given topology and an initial guess $\breve{\mathbf{\Omega}} = \text{diag}(\cdots, \breve{\mathbf{\Omega}}_k, \cdots)$. The derivative of (4) w.r.t the k-th factor's information matrix $\breve{\mathbf{\Omega}}_k$ is

$$\frac{\partial D_{KL}}{\partial \breve{\mathbf{\Omega}}_k} = \breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top - \breve{\mathbf{J}}_k (\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top \breve{\mathbf{\Omega}}_k \breve{\mathbf{J}}_k)^{-1} \breve{\mathbf{J}}_k^\top \ , \tag{11}$$

where $\breve{\mathbf{\Upsilon}}_k$ is the information matrix of the problem considering only the rest of factors,

$$\breve{\mathbf{\Upsilon}}_k = \sum_{j \neq k} \breve{\mathbf{J}}_j^\top \breve{\mathbf{\Omega}}_j \breve{\mathbf{J}}_j \ . \tag{12}$$

Descent of the KLD cost is achieved factor by factor. Each time, (4) is solved for one factor while fixing the rest, and this is done cyclically over all the factors until convergence.

Interestingly, considering all factors other than k fixed, the optimal $\breve{\mathbf{\Omega}}_k$ can be computed analytically by finding the null derivative of the KLD (11). This can take different forms depending on the particular properties of $\breve{\mathbf{\Upsilon}}_k$ and $\breve{\mathbf{J}}_k$, as we explored in [14].

In the specific case of pose-graph SLAM, assuming factors with strictly positive definite information, the optimal k-th factor's information matrix when the rest of factors are fixed may take two different forms, as follows.

If $\breve{\mathbf{\Upsilon}}_k$ is invertible, (11) is null in

$$\breve{\mathbf{\Omega}}_k = \underbrace{(\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}}_{\mathbf{\Phi}_k} - (\breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1}. \tag{13}$$

The first term $\mathbf{\Phi}_k$ can be interpreted as the information of the dense distribution projected onto the k-th factor's measurement space. Analogously, the second term is the projection of the information of the rest of the factors onto the measurement space of the k-th factor. In a pose-graph, $\breve{\mathbf{\Upsilon}}_k$ is invertible when without considering the k-th factor, the topology is still fully connected.

Otherwise, when the topology gets disconnected without the k-th factor, (11) is null in

$$\breve{\mathbf{\Omega}}_k = \mathbf{\Phi}_k = (\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}. \tag{14}$$

Since $\mathbf{\Phi}_k$ is constant, all factors with non invertible $\breve{\mathbf{\Upsilon}}_k$ do not require to be iterated. Note that $\mathbf{\Phi}_k$ is exactly the closed form (6). Indeed, the tree topology gets disconnected as soon as any of its factors is not considered.

Since the optimal solution in the factor's subspace is computed in closed form (13), 'iterations' in FD refer to cyclically solving for each of the factors, not on fitting a nonlinear model onto any linear or quadratic function. Then, the FD convergence rate mainly depends on how much the direction to the optimal solution is aligned with the sub-spaces corresponding to each factor (see Fig. 2 for an illustrative example).

3.1. Positive-definiteness

According to (13), $\breve{\mathbf{\Omega}}_k$ is positive definite only if

$$(\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1} \succ (\breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1}. \tag{15}$$

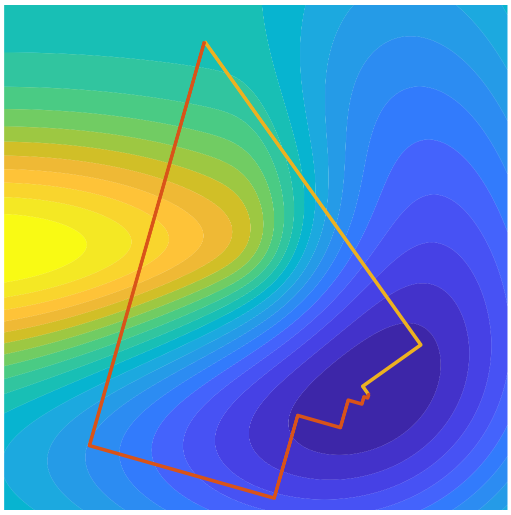
Figure 2: A simplified representation to illustrate the effect in the convergence speed of the alignment of the variables w.r.t. the direction to the optimal solution. Analogously to FD, the optimization of two variables (representing two factors' information) is performed by alternatively fixing one of the variables and analytically computing the optimal solution for the other variable. In the badly aligned case (orange) more iterations are required than in the good aligned case (yellow) to reach the optimal solution.

This happens when the projection of the dense distribution information onto the measurement space is *larger* than that of the rest of the factors in any direction. When the rest of new factors exactly explain the original distribution in some direction, $\breve{\mathbf{\Omega}}_k$ would have a null eigenvalue in that direction. Further, a negative eigenvalue of $\breve{\mathbf{\Omega}}_k$ means that the approximation without the k-th factor is not conservative and the optimal k-th factor would be the one that subtracts this excess of information. After each iteration, we impose a positive definite result setting all null and negative eigenvalues of $\breve{\mathbf{\Omega}}_k$, if any, to a small positive value ϵ, using *e.g.* $\breve{\mathbf{\Omega}}_k = \mathbf{V}\,\text{diag}(\max\{\epsilon, \lambda_i\})\mathbf{V}^\top$ with $\{\text{diag}(\lambda_i), \mathbf{V}\} = eig(\breve{\mathbf{\Omega}}_k)$.

3.2. Initial guess

Since the positive definite constraint is imposed after each factor descent, the initial guess is not required to be in a strictly feasible point. Thus, we can use the off-diagonal blocks based initialization (10).

Moreover, factor descent can also be used to build an initial guess by just applying the first cycle of (13) considering null initial information in all factors. Then, in the first cycle, only the previously computed factors contribute to $\breve{\mathbf{\Upsilon}}_k$, and therefore it can be computed incrementally,

$$\breve{\mathbf{\Upsilon}}_k = \breve{\mathbf{\Upsilon}}_{k-1} + \breve{\mathbf{J}}_{k-1}^\top \breve{\mathbf{\Omega}}_{k-1} \breve{\mathbf{J}}_{k-1}. \tag{16}$$

4. Topology

Whichever factor recovery method is used to solve (4), its result is only the best approximation that can be achieved for a given topology. But different topologies produce different approximations. Therefore, the selection of the topology is critical in terms of the accuracy of the sought sparse approximation. From this viewpoint, the accuracy of the solution depends on how much the selected topology can explain the dense distribution.

In this paper we focus on pose-graph SLAM problems. In pose-graph SLAM, all factors correspond to relative measurements between pairs of nodes of the same dimension. The simplest topology using relative measurements is a spanning tree. The Chow-Liu tree (CLT) defines the tree topology that can encode more information from the dense distribution. To do this, it recalls on mutual information (MI) between all pairs of nodes in order to measure which nodes are more correlated.

However, in order to evaluate the MI either using the canonical form or the information form, a non-singular dense information matrix $\mathbf{\Lambda}$ is required. Precisely, in the pose-graph SLAM case the information matrix resulting of node marginalization is singular. The main objective of computing the MI between pairs of nodes is measuring how a potential factor between each pair would explain the original dense distribution. Then, working on a projected sub-space as (5) is not a suitable solution since we lose the track of each node.

Carlevaris et al. [11] compute the MI of each pair of nodes using the information form. Since the dense information matrix $\mathbf{\Lambda}$ is singular, to prevent from null determinants a Tikhonov regularization is used within all determinants

$$\mathbf{I}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \log \frac{|\tilde{\mathbf{\Lambda}}_{ii} + \epsilon \mathbf{I}|}{|\tilde{\mathbf{\Lambda}}_{ii} - \tilde{\mathbf{\Lambda}}_{ij} \tilde{\mathbf{\Lambda}}_{jj}^{-1} \tilde{\mathbf{\Lambda}}_{ji} + \epsilon \mathbf{I}|}. \tag{17}$$

In order to obtain $\tilde{\mathbf{\Lambda}}_{ii}, \tilde{\mathbf{\Lambda}}_{ij}$ and $\tilde{\mathbf{\Lambda}}_{jj}$, all variables different from i and j should be marginalized out from $\mathbf{\Lambda}$ via Schur complement.

Conversely, Mazuran et al. [13] compute the MI using the covariance form. But first, the Tikhonov regularization is performed to get an approximation of the covariance matrix $\hat{\mathbf{\Sigma}} = (\mathbf{\Lambda} + \epsilon \mathbf{I})^{-1}$,

$$\mathbf{I}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \log \frac{|\hat{\mathbf{\Sigma}}_{ii}||\hat{\mathbf{\Sigma}}_{jj}|}{\begin{vmatrix} \hat{\mathbf{\Sigma}}_{ii} & \hat{\mathbf{\Sigma}}_{ij} \\ \hat{\mathbf{\Sigma}}_{ji} & \hat{\mathbf{\Sigma}}_{jj} \end{vmatrix}}. \tag{18}$$

Note that (17) and (18) are not strictly equivalent since the regularization is performed in different places. No matter how the MI is computed, the CLT tree is built by incrementally taking the pair of nodes with largest MI that do not create a cycle.

However, even being the most informative tree, since a tree topology is the sparsest topology, the CLT is usually too simple to approximate the original distribution. For this reason, the so-called sub-graph (SG) topology is proposed in [13]. It is built departing from the CLT and complementing it with more factors, also based on the already computed MI values.

Alternatively, the cliquey topology [13] takes the CLT and converts pairs of independent factors into one single factor by correlating them. However, it implies breaking the homogeneity of factors creating factors that involve more than two nodes. We propose to use only relative pose measurement factors to maintain the pose-graph SLAM nature.

Differently to the CLT-based methods, Eckenhoff et al. [12] proposed an ℓ_1-regularized KLD minimization to compute the topology that will encode the most information

$$\min_{\breve{\mathbf{\Lambda}}} \quad \langle \breve{\mathbf{\Lambda}}, \mathbf{\Sigma} \rangle - \ln |\breve{\mathbf{\Lambda}}| + \lambda \|\breve{\mathbf{\Lambda}}\|_1. \tag{19}$$

Then, the topology is built setting relative measurement factors according to the significant off-diagonal blocks of the resulting $\breve{\mathbf{\Lambda}}$.

4.1. Topology population

We call population to the number of factors that are contained in a sub-graph. Determining the topology population K is a compromise between sparsity and accuracy. More populated topologies achieve better approximations, however they undermine the sparsity of the resulting graph and also solving the factor recovery becomes computationally more expensive. While using tree topologies allows the use of closed form factor recovery solution, complementing it with some more factors (SG) leads to constrained iterative optimization. Moreover, all mentioned iterative factor recovery methods become slower the more populated the topologies are. While the gradient and Hessian get larger for the IP and PQN methods, factor descent has more factors to iterate over.

A policy to fix the population size of the topology should be adopted. The topology must satisfy two conditions: It has to connect all nodes, and there should be no two factors connecting the same pair of nodes since that would be redundant. Thus the topology population K, i.e., the number of factors, for a given Markov blanket of size n should be in the interval

$$K \in \left[n-1, \tfrac{n(n-1)}{2} \right]. \tag{20}$$

In [13] the population is fixed proportional to that of a spanning tree

$$K = \left\lceil \gamma(n-1) \right\rceil, \quad \text{with} \quad \gamma \geq 1. \tag{21}$$

Then, while for small Markov blankets the result is quite dense, for bigger ones it becomes very sparse.

In [12], a threshold is used to consider *significant* the resulting off-diagonal values of (19). However, the magnitude of the information matrix entries depends on the noise of the original measurements, so the tuning of this threshold is highly problem dependant.

A sparsification application has two opposite aims: sparsity and accuracy of the simplified graph. For this reason, we propose a new population policy proportional to the total amount of possible factors,

$$K = \left\lceil \alpha \frac{n(n-1)}{2} \right\rceil, \quad \text{with} \quad \alpha \in (0,1]. \tag{22}$$

The parameter α fixes the fill-in ratio of the resulting information matrix. It is a more intuitive parameter to be tuned by the end user depending on the application specifications. Also, given its quadratic form, the fill-in ratio adapts better than a tree-proportional one to different Markov blanket sizes.

Note that both tree-proportional and fill-in policies can provide populations out of the allowed range (20). Then, the resulting population of (21) and (22) must be clipped to an admissible value.

4.2. Building populated topologies

As the formulation in Sec. 3 shows, the factors of a populated topology affect each other in the computation of the optimal solution. Therefore, it is important to remark that building a sub-graph topology by incrementally taking the most informative factor does not have to produce the most informative sub-graph topology. In consequence, despite being the CLT the most informative tree, this does not mean that the most informative sub-graph will contain one or all the factors in the CLT. Unfortunately, optimally solving this combinatorial problem would require significant computational effort.

Three new alternative methods to build an informative topology are proposed below: downdated mutual information; expected KLD decrease; and off-diagonal block determinant. Figure 3 shows an example of the different factor recovery results, compared to the CLT and to a brute force search of the optimal solution.

4.2.1. Downdated mutual information

As introduced early, mutual information (MI) between pairs of nodes is a useful metric to indicate to what extent a factor relating those nodes contributes to explain the dense distribution. However, once the CLT is built, the MI between the rest of pairs of nodes computed either using (17) or (18) may not be a reliable metric about the amount of information that each new factor contributes since the already present CLT factors already explain some of these correlations.

To tackle this, our first proposal is a variation of the MI-based method to complement the CLT. First the CLT is computed in the same manner, using the MI. Afterwards, instead of directly complementing the CLT with the subsequent MI pairs of nodes, the MI of the remaining pairs of nodes are recomputed taking into account the already added CLT factors. To do this, before recomputing these MIs, we perform a downdate on the regularized covariance $\hat{\mathbf{\Sigma}}$ with the information of all CLT factors. Since we are working in the covariance space, we can rely on the Kalman equations to find an optimal value for the covariance increment,

$$\Delta\hat{\mathbf{\Sigma}} = \sum_{j \in CLT} \hat{\mathbf{\Sigma}} \breve{\mathbf{J}}_j^\top (\breve{\mathbf{\Omega}}_j^{-1} + \breve{\mathbf{J}}_j \hat{\mathbf{\Sigma}} \breve{\mathbf{J}}_j^\top)^{-1} \breve{\mathbf{J}}_j \hat{\mathbf{\Sigma}}. \tag{23}$$

Note that equation (23) is obtained by substituting the Kalman gain into the covariance update, for each factor of the CLT, and then adding them all up. Contrary to the Kalman update, however, this increment is added to the covariance, removing information and thus constituting a *downdate*, $\hat{\mathbf{\Sigma}} \leftarrow \hat{\mathbf{\Sigma}} + \Delta\hat{\mathbf{\Sigma}}$.

To evaluate (23), all CLT factors information matrices $\breve{\mathbf{\Omega}}_j$ are required. Therefore and ideally, factor recovery for all CLT factors should be performed. However, the factor recovery solution for these factors depends on the rest of the factors that will be finally added to the topology, which have not been decided yet. To resolve the situation, we resort to estimating the final CLT factors information $\breve{\mathbf{\Omega}}_j$ as they would be without complementing with more factors. Hence, we approximate each $\breve{\mathbf{\Omega}}_j$ with the closed form factor recovery (6).

After downdating the CLT estimated contribution to the regularized covariance $\hat{\mathbf{\Sigma}}$, the MI of the remaining pairs of nodes can be computed again. Hence, we named it downdated mutual information. It provides a measure of where there is still information that has not been explained by the CLT. Finally, until the topology population is achieved, new factors are added to the topology corresponding to the pairs of nodes with most downdated mutual information.

The more complementary factors are added to the topology, the more unreliable the downdated mutual information becomes, for the same reason as the initial MI values. Then, the downdate could be performed once or recursively after adding each new factor. However, in our experience, this does not contribute significantly to improve the resulting topology compared to its computational cost. Then, we only perform a single downdate after building the CLT.

4.2.2. Expected KLD decrease

Taking profit of our FD method, we propose a new topology building methodology. Departing from the CLT topology, we incrementally build the topology by finding the factor candidate that will decrease the KLD the most. To compute this, for each remaining factor candidate we compute an estimation of its factor recovery solution. Afterwards, the KLD decrease of including each factor in the topology is computed. The candidate with the most expected KLD decrease is chosen.

In order to compute the factor recovery solution estimate, we perform one single FD instance, without iterating.

Despite a single instance of Factor Descent does not provide the final factor recovery solution, it is enough to evaluate how much new information can be provided by

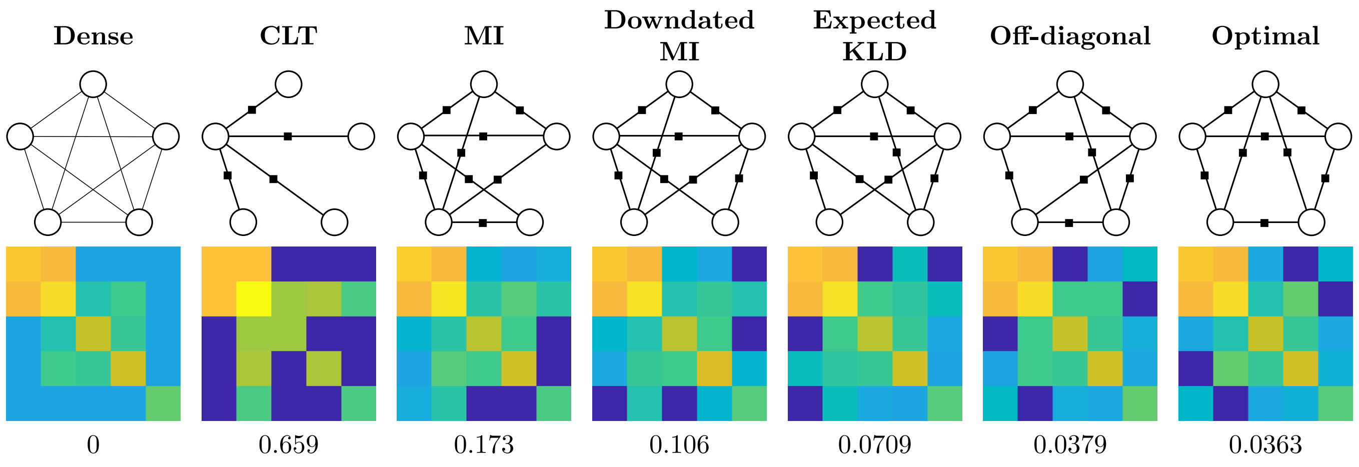

Figure 3: Example of factor recovery results using different topology building methods (top). In the middle row, the information pattern of $\breve{\mathbf{\Lambda}}$ (or $\mathbf{\Lambda}$ in the first case), colors depict log absolute values. In the bottom row, the KLD of each factor recovery solution from the dense distribution.

each factor candidate. This alternative requires several evaluations of KLD and Factor Descent instances. However, note that the resulting topology already provides a good initial guess for the factor recovery equivalent to the first Factor Descent cycle.

4.2.3. Off-diagonal block determinant

Taking inspiration from the off-diagonal block initialization (10), we propose a third method to build the topology. The off-diagonal block entries of the dense information matrix $\mathbf{\Lambda}$ correspond to different pair of nodes and can only be explained by a factor between both nodes. Therefore, this method adds a factor to the topology corresponding to the off-diagonal blocks with the most significant values. To identify them, we make use of the absolute determinant of each off-diagonal block of the information $\mathbf{\Lambda}$.

Note that in this case, since we are dealing with an information matrix, adding factors does not affect the rest of off-diagonal blocks. Thus, no downdating is needed.

Differently from the other methods proposed, this topology is built from scratch using this measure instead of complementing the CLT. For this reason one has to take care in the end of producing a fully connected topology. To guarantee this, the first $n-1$ pairs of factors are chosen such that they build a spanning-tree. Analogously to the CLT procedure, the pairs of nodes that do not create a cycle with largest off-diagonal block absolute determinant, are incrementally added to the topology. Once a tree topology is reached, the no-cycle restriction is no longer applied and other factors can be added until reaching the desired population.

5. Results

In order to test the performance of the Factor Descent sparsification method and the influence of the topology in the factor recovery approximation, three different types of experiments were run on a computer with an Intel Core $i7-4770$ CPU (@3.40GHz x 8). To do so, the Factor Descent and Interior Point MATLAB prototypes were reimplemented in C++ and integrated with our own nonlinear least squares SLAM based on Ceres [23]. Based in our preliminary results indicating its very poor convergence, PQN was discarded from the comparison and it was not implemented.

In order to make a fair comparison available, a *sparsification dataset* was built. This dataset contains several sparsification problems (i.e. node marginalization dense results $\boldsymbol{\mu}, \mathbf{\Lambda}$) of different Markov blanket sizes. To build it, an experiment was run for the Manhattan M3500 dataset [24] storing the result of each marginalization $(\boldsymbol{\mu}, \mathbf{\Lambda})$ of the 80% of the nodes.

As explained, the Interior Point method required a tuning session for its outer loop parameters (initial log barrier factor ρ_0 and log barrier decrease factor β) and the inner loop end condition (minimum gradient threshold ∇_{min}). An extensive search was performed solving all sparsification dataset problems for several combinations of the three parameters. We took the combination that did not diverge in any case and provided the fastest convergence. For our specific experimental conditions, the found values are $\rho_0 = 1, \beta = 0.5$ and $\nabla_{min} = 1$.

5.1. Topology

The first experiment had the purpose of evaluating the performance of the proposed methods to build the topology and the new population policy. All proposed topology methods were compared: decreased mutual information (dMI), expected Kullback-Liebler divergence (eKLD) and off-diagonal determinant (ODD). The comparison includes also the state-of-art method based on mutual information (MI) and the optimal (BEST) topology found using brute force combinatorial search.

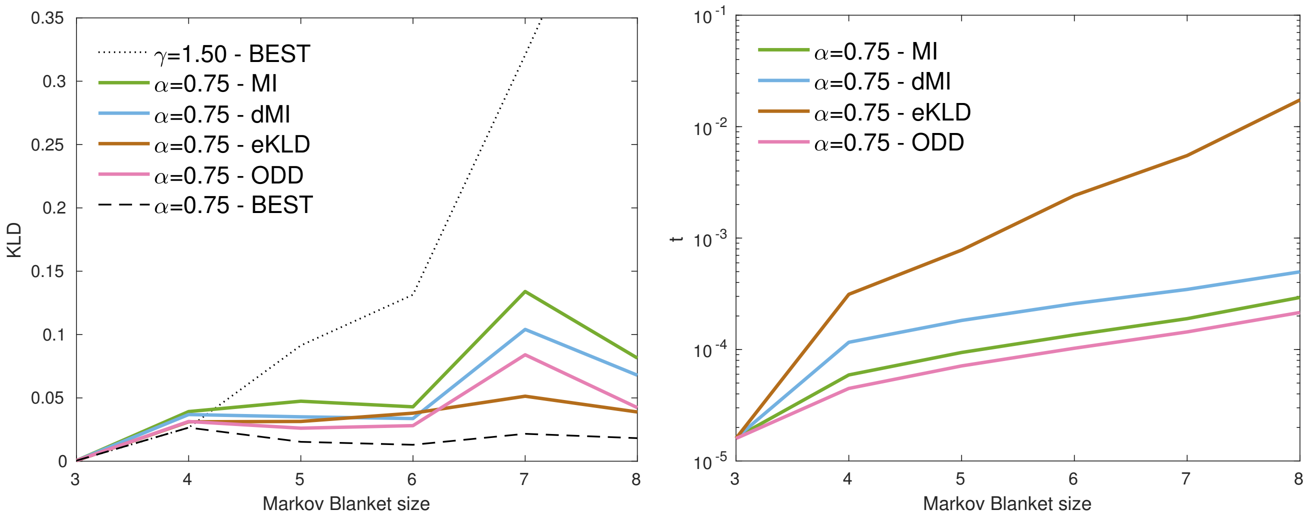

Figure 4: Comparison of topology building methods using different fill-in population policies. The plots show KLD to the optimal factor recovery solution (left) and computational cost (right) for different Markov blanket sizes. In dashed black, the best topology, shown as a baseline. In dotted black, the solution of the best topology fixing the population proportional to the spanning tree population.

Then, for each dense distribution defined by $\boldsymbol{\mu}, \boldsymbol{\Lambda}$ stored in the sparsification dataset, each method built a topology using the same amount of factors. The amount of factors were fixed with a fill-in population policy with $\alpha = 0.75$. The computational cost of the topology building process and the KLD of the optimal factor recovery were stored. The optimal factor recovery was computed using FD and the off-diagonal blocks based initial guess (10). Nevertheless, IP could be used instead without any changes in the experiment, since the KLD difference of the solution using FD or IP is negligible, as demonstrated in our previous work.

The optimal topology is obtained in a brute-force search, for all possible topology combinations keeping the best set of factors in terms of KLD to the original problem. This alternative is only considered as a baseline for comparison since it is computationally prohibitive.

Figure 4 (left) shows the mean KLD of the factor recovery using the different topology methods. The frame to the right in the same figure shows the time required to recover each topology. Note that the plot does not include the time required for factor recovery.

The most computationally expensive method is the expected KLD since it requires a Factor Descent instance and KLD evaluation in all remaining factors for each new factor that is included in the topology. Despite its computational cost, eKLD performs best in terms of KLD for the largest Markov blanket sizes.

The dMI method improves significantly in terms of KLD to the case of MI but slightly increasing its computational cost. It confirms that downdating the regularized covariance avoids adding factors between nodes whose mutual information is already explained by the CLT factors.

The ODD method is the fastest method given its simplicity and provides topologies that can still accurately approximate the original distribution.

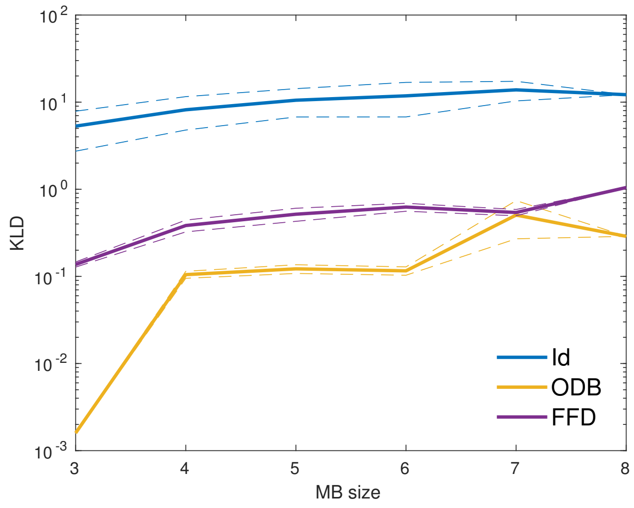

Figure 5: Mean KLD and variance (dashed line) of all three initial guesses (identity matrix, off diagonal blocks, and first factor descent cycle) as a function of the Markov blanket size for the Manhattan experiment with 80% node removal.

The best topology found with brute force combinatorial search using the tree-proportional population policy is plotted in dashed black. It can be observed how the tree-proportional population policy becomes sparse for big Markov blankets leading to higher KLD values.

5.2. Initial guess and convergence rate

Further experiments were designed in order to test the convergence rate of different combinations of optimization method and initial guess on several factor recovery problems.

The IP method using the identity matrix as initial guess as proposed in [13] is compared with FD using all

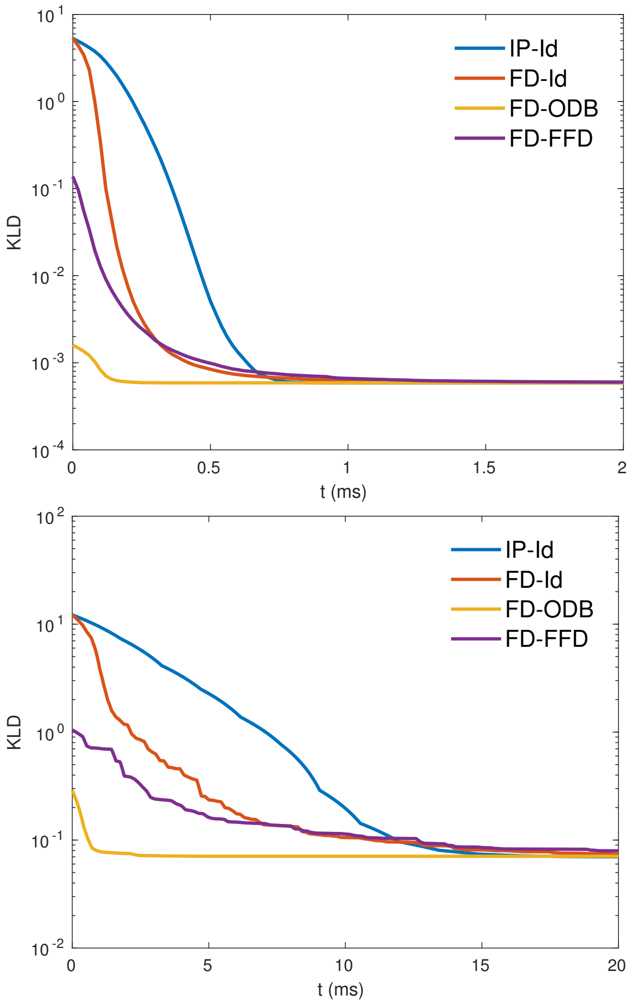

Figure 6: Mean KLD evolution of all factor recovery combinations (methods and initial guesses), for problems with Markov blanket size = 3 (top) and 8 (bottom) in the Manhattan experiment with 80% of node removal.

three mentioned initial guesses: identity matrix (Id), off-diagonal blocks (ODB) [21], and the first FD cycle (FFD) of Sec. 3.2.

This experiment is also performed using the sparsification dataset. For each stored problem, the same topology is built using a fill-in population policy with $\alpha = 0.75$. Afterwards, the factor recovery problem is solved using all four method-initial guess combinations.

Figure 5 shows the mean and variance of the KLD of each initial guess (i.e. before solving using any optimization method) as a function of the Markov blanket size. It can be observed how the identity matrix is the worst initial guess (note the log scale in the y-axis). The ODB initial guess is much closer to the optimal solution, however it becomes worse for bigger Markov blankets. Despite that FFD is significantly worse than ODB in small Markov blankets in terms of KLD, it achieves similar approximations in big problems.

Figure 6 shows the mean KLD evolution for all four method-initial guess combinations, for Markov blankets of size 3 (top) and 8 (bottom). Obviously, for larger Markov blankets, the convergence of all methods is slower since a bigger Hessian is inverted in IP and more factors should be refined in FD. As explained, FD has a steeper initial converge rate, and the availability of ODB initial guess enhances the performance of FD compared with IP.

5.3. Application

In the two previous experiments, all compared methods solved the same sparsification problems stored in the sparsification dataset. Conversely, in this last battery of experiments, each compared method has its own SLAM system and accumulates the sparsification approximations along the whole experiment.

This was made on two different datasets: the Manhattan M3500 sequence and the Intel Research Lab sequence [24]. The Manhattan M3500 sequence is a large and highly connected problem. Then, it has big Markov blankets, meaning big sparsification problems. Contrarily, the Intel Research Lab sequence has very few loop closures and smaller Markov blankets.

Several combinations of population policy and topology methods were compared. Both tree-proportional and fill-in population policies were used with two different values of their corresponding parameters γ and α, respectively. For each four population combinations, four topology methods are compared, MI, dMI, ODD and eKLD. Additionally, we included in the comparative the CLT topology with closed form factor and the MI topology for all populations combinations with the IP factor recovery initialized by the identity matrix. Note that CLT topology is exactly equivalent to the MI topology method using the tree-proportional population policy with $\gamma = 1.0$.

For each combination of topology method and population policy, an independent experiment was run. In all experiments, the same 80% volume of nodes were marginalized. Since node selection is out of the scope, we applied the simple strategy of marginalizing and sparsifying 4 of every 5 nodes. This is performed periodically every 100 nodes. Therefore, each experiment accumulates the sparsification approximations along the whole dataset showing the differences induced exclusively by the mentioned differences.

The original SLAM graph without removing any node is taken as a baseline. The global KLD between each method and the baseline is computed using (3) for the whole SLAM problem. The same end conditions for all factor recovery methods was used, KLD gradient norm lower than 10^{-3} and a maximum time of 0.20 ms.

As in [13], factors involving previously removed nodes were redirected to the closest existing node. This emulates the effect of data association in real experiments: a loop closure with a removed node will not be detected; instead, the loop will be most probably closed with the closest existing node. In order not to distort the results, this was also applied to the baseline graph.

The results of all experiments are listed in Table 1. Global KLD and position and orientation RMSE of the

Table 1: Comparison of different combinations of population policy, topology building method and factor recovery in different datasets at 80% node reduction.

	Population policy		**Topology**	**Factor recovery**	**KLD**	**RMSE**		**Sparsification total time (s)**	**Topology total time (s)**	**Number of factors**
						position (m)	**orientation (rad)**			
Manhattan	Tree-proportional	$\gamma = 1.0$	MI	Closed form	32.33	0.114	0.0055	**0.08**	**0.058**	1605
		$\gamma = 1.5$	MI	IP-Id	7.45	0.062	0.0032	6.11	***0.070***	2281
			MI	FD-ODB	7.04	0.069	0.0031	5.54	0.071	2282
			dMI		***6.65***	0.055	0.0029	5.92	0.117	2269
			ODD		7.99	0.045	0.0027	***5.42***	0.072	2246
			eKLD		7.18	***0.039***	***0.0023***	6.64	0.371	2270
		$\gamma = 2.0$	MI	IP-Id	2.92	0.025	0.0017	3.23	***0.069***	2624
			MI	FD-ODB	2.63	0.034	0.0018	***1.95***	0.073	2624
			dMI		2.24	***0.024***	***0.0015***	3.31	0.103	2623
			ODD		***1.94***	0.056	0.0021	3.35	0.071	2625
			eKLD		2.66	0.037	0.0016	7.07	0.393	2661
	Fill-in	$\alpha = 0.75$	MI	IP-Id	3.24	0.105	0.0032	6.81	***0.074***	2482
			MI	FD-ODB	2.97	0.119	0.0035	***5.36***	0.076	2483
			dMI		***2.58***	**0.020**	***0.0019***	7.17	0.128	2482
			ODD		3.14	0.047	0.0020	7.05	0.075	2528
			eKLD		4.13	0.053	0.0021	14.36	0.562	2584
		$\alpha = 0.85$	MI	IP-Id	1.13	0.034	0.0014	4.03	***0.073***	2895
			MI	FD-ODB	0.85	0.046	0.0016	2.17	0.078	2895
			dMI		**0.73**	***0.025***	**0.0011**	1.90	0.116	2898
			ODD		0.76	0.031	0.0012	***1.63***	0.077	2931
			eKLD		1.71	0.029	0.0014	6.51	0.686	2949
Intel	Tree-proportional	$\gamma = 1.0$	MI	Closed form	29.16	0.065	0.0105	**0.03**	**0.016**	366
		$\gamma = 1.5$	MI	IP-Id	10.45	0.052	0.0071	1.86	0.018	492
			MI	FD-ODB	9.95	0.045	0.0063	1.96	0.017	492
			dMI		8.92	0.033	0.0051	2.36	0.028	492
			ODD		9.58	***0.023***	0.0050	***1.77***	***0.017***	488
			eKLD		***5.75***	0.029	***0.0024***	2.57	0.074	507
		$\gamma = 2.0$	MI	IP-Id	5.46	0.024	0.0038	***2.37***	0.018	558
			MI	FD-ODB	5.01	***0.023***	0.0036	2.99	0.018	558
			dMI		4.72	0.029	0.0034	2.82	0.024	560
			ODD		4.07	0.032	***0.0025***	2.82	***0.018***	564
			eKLD		***3.15***	0.031	0.0028	3.78	0.070	577
	Fill-in	$\alpha = 0.75$	MI	IP-Id	4.91	0.044	0.0047	***2.98***	0.019	544
			MI	FD-ODB	4.51	0.035	0.0039	3.05	0.020	545
			dMI		3.52	0.022	0.0024	3.90	0.034	546
			ODD		***3.51***	0.030	0.0024	3.41	***0.019***	547
			eKLD		3.71	**0.014**	***0.0019***	6.20	0.114	554
		$\alpha = 0.85$	MI	IP-Id	2.74	***0.015***	0.0018	***1.98***	***0.019***	610
			MI	FD-ODB	2.20	0.016	0.0015	2.17	0.019	610
			dMI		**2.06**	0.016	0.0017	3.14	0.028	611
			ODD		2.09	0.021	**0.0012**	3.29	0.019	611
			eKLD		2.18	0.022	0.0016	3.89	0.110	617

final graph w.r.t. the baseline were computed. Additionally, the table contains the total computational time spent on sparsification and topology building, and the amount of factors of the final graph.

As expected, the closed form factor recovery is the fastest sparsification alternative and the one that produces the sparsest final graph. However, the simplicity of the topology produces worse approximations leading to higher global KLD and RMSE values. In contrast, for the same population policy, the most populated options ($\gamma = 2.0, \alpha = 0.85$) produce much more accurate approximations both in terms of KLD and RMSE. Indeed, as repeatedly stated, more populated topologies can better approximate dense distributions.

For the same topology, both IP and FD achieve similar KLD and RMSE values. However, IP becomes slower in the Manhattan dataset due to the larger Markov blankets.

Comparing dMI and MI, one can see how downdating the regularized covariance always produces a topology that is able to encode more information producing better approximations. While in some cases, eKLD produces informative topologies that can better encode the dense information, in some cases it produces worse approximations and its computational cost is significantly higher than the rest of topology methods. While ODD outperforms MI in terms of KLD for the most populated topologies

($\gamma = 2.0$ and $\alpha = 0.85$), it produces similar or worse results in sparser cases. ODD is in some cases the fastest method competing with MI. This is because both MI and ODD only evaluate the information of all factor candidate once.

Obviously, for a given population policy option, all four topology methods produce similarly sparse final graphs. Differences are due to concatenating sparsification processes. As we observed in the first results, the fill-in population policy produces more populated topologies in big Markov blankets. Then, to achieve similar approximations in big Markov blanket problems, a higher value of γ is needed for the tree-proportional policy leading to too much populated topologies in small problems. For this reason, the fill-in policy with $\alpha = 0.75$ achieves similar KLD and RMSE than tree-proportional with $\gamma = 2$ using less factors.

The benefits of performing marginalization and sparsification instead of solving the entire graph highly depend on the application: the duration of the experiment, computational resources, etc. To provide an estimate of such benefits, we compare the execution of SLAM sessions with and without applying the marginalization and sparsification strategies discussed in this paper. Table 2 compares the CPU time required to solve the SLAM problem in each of the experiments compared in Table 1. That is, the computational cost of solving the whole SLAM problem versus the time needed to solve the reduced versions at 20% the original size with the different population policy and topology methods.

In all the experiments, the SLAM problem was solved each time a node was added, using our solver based on Ceres (batch algorithm) performing one iteration of the NLS problem (2), and marginalization and sparsification was performed periodically. On average, the reduced SLAM problems were solved 2.4 to 4.7 times faster than the whole SLAM problem depending on the topology population used in the sparsification. In our implementation, the accumulated time saved ranged from $93s$ to $117s$ (a decrease of 60% to 77%) for the Manhattan world, and from $7.6s$ to $8.2s$ (a decrease of 70% to 78%) for the Intel map. The more populated the topology, the lower the time savings in computing a solution, but the closer the approximation to that of the original problem.

Table 2: Computational times for solving the SLAM problem for the original and the sparsified graphs (80% of node reduction).

			SLAM solving time	**% of solving original graph**
Manhattan	Original graph		0.0558 s	100%
	Tree-prop.	$\gamma = 1.0$	0.0139 s	25.0 %
		$\gamma = 1.5$	0.0170–0.0190 s	30.4–34.0 %
		$\gamma = 2.0$	0.0195–0.0221 s	35.0–39.7 %
	Fill-in	$\alpha = 0.75$	0.0199–0.0224 s	35.7–40.1 %
		$\alpha = 0.85$	0.0225–0.0238 s	40.3–42.7 %
Intel	Original graph		0.0127 s	100%
	Tree-prop.	$\gamma = 1.0$	0.00277 s	21,7 %
		$\gamma = 1.5$	0.00329–0.00361 s	25.7–28.2 %
		$\gamma = 2.0$	0.00368–0.00404 s	28.8–31.6 %
	Fill-in	$\alpha = 0.75$	0.00356–0.00396 s	27.8–30.9 %
		$\alpha = 0.85$	0.00384–0.00408 s	30.1–31.9 %

6. Conclusions and future work

Marginalization and sparsification are valuable tools to reduce the size of SLAM problems without undermining the accuracy and efficiency of the solution. This procedure can be understood as a time investment. A small computational time is spent once to reduce the problem size, and the benefits (time savings) are obtained each time the SLAM problem is solved. Furthermore, the marginalization and sparsification of nodes can be distributed along the experiment and even postponed depending on resources availability.

Deciding the topology of the new factors to approximate the original distribution is a trade off between accuracy and computational cost. For the simplest topologies, e.g., spanning tree, factor recovery has a closed form solution but the approximations are less accurate. On the contrary, more populated topologies prevent the use of a closed form, thus iterative optimization is required to solve for factor recovery. Furthermore, the more populated the topology is, the more computationally expensive the factor recovery becomes, regardless of which iterative factor recovery method is used. In exchange, the more populated the topology is, the better the original graph information can be encoded, reaching better approximations.

We described the Factor Descent optimization method for factor recovery and its application to the pose-graph SLAM case. Factor Descent achieves a KLD-optimal factor recovery solution with less computational cost than state-of-the-art methods.

Furthermore, three new methods for building populated pose-graph topologies and a new policy to determine the topology population are proposed, outperforming the state-of-art approaches.

In future work we consider the application of factor descent and new topologies for different SLAM setups such as visual-inertial systems.

7. References

[1] J. Vallvé, J. Solà, J. Andrade-Cetto, Factor descent optimization for sparsification in graph slam, in: Proc. Eur. Conf. Mobile Robots, IEEE, Paris, 2017, pp. 1–6.

[2] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, F. Dellaert, iSAM2: Incremental smoothing and mapping using the Bayes tree, Int. J. Robotics Res. 31 (2) (2011) 216–235.

[3] V. Ila, L. Polok, M. Solony, P. Svoboda, SLAM++-A highly efficient and temporally scalable incremental SLAM framework, Int. J. Robotics Res. 36 (2) (2017) 210–230.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, P. Furgale, Keyframe-based visual-inertial odometry using nonlinear optimization, Int. J. Robotics Res. 34 (3) (2015) 314–334.

[5] M. Li, A. I. Mourikis, High-precision, consistent EKF-based visual-inertial odometry, Int. J. Robotics Res. 32 (6) (2013) 690–711.

[6] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, C. Hertzberg, Hierarchical optimization on manifolds for online 2D and 3D mapping, in: Proc. IEEE Int. Conf. Robotics Autom., Anchorage, 2010, pp. 273–287.

[7] H. Johannsson, M. Kaess, M. Fallon, J. Leonard, Temporally scalable visual SLAM using a reduced pose graph, in: Proc. IEEE Int. Conf. Robotics Autom., Karlsruhe, 2013, pp. 54–61.

[8] V. Ila, J. M. Porta, J. Andrade-Cetto, Information-based compact Pose SLAM, IEEE Trans. Robotics 26 (1) (2010) 78–93.

[9] S. Choudhary, V. Indelman, H. Christensen, F. Dellaert, Information-based reduced landmark SLAM, in: Proc. IEEE Int. Conf. Robotics Autom., Seattle, 2015, pp. 4620–4627.

[10] H. Kretzschmar, C. Stachniss, Information-theoretic compression of pose graphs for laser-based SLAM, Int. J. Robotics Res. 31 (11) (2012) 1219–1230.

[11] N. Carlevaris-Bianco, M. Kaess, R. M. Eustice, Generic node removal for factor-graph SLAM, IEEE Trans. Robotics 30 (6) (2014) 1371–1385.

[12] K. Eckenhoff, L. Paull, G. Huang, Decoupled, consistent node removal and edge sparsification for graph-based SLAM, in: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Daejeon, 2016, pp. 3275–3282.

[13] M. Mazuran, W. Burgard, G. D. Tipaldi, Nonlinear factor recovery for long-term SLAM, Int. J. Robotics Res. 35 (1-3) (2016) 50–72.

[14] J. Vallvé, J. Solà, J. Andrade-Cetto, Graph SLAM sparsification with populated topologies using factor descent optimization, IEEE Robotics and Automation Letters 3 (2) (2018) 1322–1329.

[15] R. Smith, M. Self, P. Cheeseman, Estimating uncertain spatial relationships in robotics, in: Autonomous Robot Vehicles, 1990, pp. 167–193.

[16] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, g^2o: A general framework for graph optimization, in: Proc. IEEE Int. Conf. Robotics Autom., Shanghai, 2011, pp. 3607–3613.

[17] L. Polok, V. Ila, M. Solony, P. Smrz, P. Zemcik, Incremental block Cholesky factorization for nonlinear least squares in robotics, in: Robotics: Science and Systems, Berlin, 2013.

[18] F. Dellaert, M. Kaess, Square root SAM: Simultaneous localization and mapping via square root information smoothing, Int. J. Robotics Res. 25 (12) (2006) 1181–1204.

[19] M. Kaess, A. Ranganathan, F. Dellaert, iSAM: Incremental smoothing and mapping, IEEE Trans. Robotics 24 (6) (2008) 1365–1378.

[20] M. Schmidt, E. Berg, M. Friedlander, K. Murphy, Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm, in: Artificial Intelligence and Statistics, 2009, pp. 456–463.

[21] M. Mazuran, G. D. Tipaldi, L. Spinello, W. Burgard, Nonlinear graph sparsification for SLAM, in: Robotics: Science and Systems, Berkeley, 2014, pp. 1–8.

[22] N. J. Higham, Computing a nearest symmetric positive semidefinite matrix, Linear algebra and its applications 103 (1988) 103–118.

[23] S. Agarwal, K. Mierle, Others, Ceres solver, http://ceres-solver.org.

[24] L. Carlone, http://www.lucacarlone.com/index.php/resources/datasets.