2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Robust Pose Graph Optimization Using Stochastic Gradient Descent

John Wang and Edwin Olson

***Abstract*— Robust SLAM methods can allow robots to recover correct maps even in the presence of incorrect loop closures. While these approaches improve robustness to outliers, they are susceptible to getting caught in local minima, a problem which is exacerbated by poor initial estimates.**

In this paper, we describe a stochastic gradient descent optimization approach that exhibits greater robustness to poor initial estimates. Our approach can either be used as a standalone optimization system or in conjunction with existing methods such as Gauss-Newton solvers. Using a combination of synthetic and real-world datasets, we demonstrate that our proposed approach is able to recover correct pose graphs significantly more frequently than other methods when large initialization errors are present.

I. Introduction

Simultaneous Localization and Mapping (SLAM) systems must overcome two major problems with real-world data. First is the problem of noise, particularly noise in odometry sensors. While noise may be overcome by fusing redundant observations, extreme noise can lead to poor initial estimates from which it is impossible to recover the robot's true trajectory. Second is the problem of incorrect data associations, or loop closures. A robot may use visual similarities or laser-scan matching to determine whether it has visited a place before. Perceptual aliasing, which occurs when two distinct places appear the same, can lead to incorrect loop closures.

To date, much work in SLAM has centered around solving the first problem of correcting for positional error, sidestepping the second difficulty by assuming perfect data association. Many such SLAM algorithms are incapable of dealing with erroneous loop closures, and even a single false loop closure can introduce unrecoverable errors into the robot's inferred trajectory. Conventionally, the data association problem has been addressed by building "front-ends" to filter out poor associations. Although this approach greatly improves the quality of SLAM-generated maps, these methods alone are inadequate. In a long-running system, even an extremely low error rate can result in loop closure errors, which will accumulate over time to cause errors in the inferred trajectory and the resulting map.

Recent work in SLAM has explicitly modeled the possibility of incorrect data associations inside the "back-end" solver [1]–[3]. Some of these robust back-ends use extensions to the pose graph model, such as switchable constraints or max-mixture constraints. These robust constraints introduce many local minima into the optimization problem. This can be illustrated intuitively: suppose a loop closure constraint is modeled as a Gaussian mixture with a "null hypothesis" component (a very high covariance Gaussian) that represents discarding that loop closure. A solution which discards a correct loop closure may actually decrease the χ^2 error (eq. 3), since the "null hypothesis" may contribute very little χ^2 error to the graph. A local minimum occurs if the graph is in a state where activating any correct loop closure would increase the χ^2 error. These local minima in the χ^2 error function make the max-mixtures optimization problem difficult. Optimization problems with local minima are more sensitive to the initial values of the variables, making them less robust to odometry noise. This creates a trade-off between robustness to loop closure errors and robustness to initialization error.

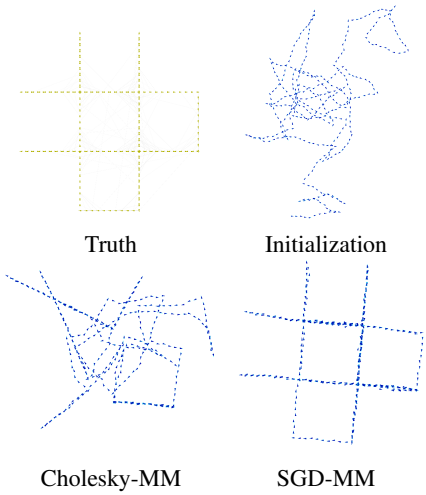

Fig. 1: An example Manhattan world pose graph with 40 false loop closures and odometry noise $\sigma = 0.24$. Cholesky-MM, the prior state of the art, becomes stuck in a local minimum due to the poor initialization. Our proposed method SGD-MM escapes the local minimum and recovers an accurate grid-shaped trajectory.

Building upon this body of previous work, we present a robust SLAM optimization algorithm that is resilient to both poor initial estimates and incorrect data associations. Our formulation uses a combination of the Gaussian max-mixtures model and the stochastic gradient descent solver, which is capable of quickly escaping local minima and often finding the global minimum. The main contributions of this paper are:

- An efficient stochastic gradient descent solver for Gaussian max-mixture pose graphs

The authors are with the Computer Science and Engineering Department, University of Michigan, Ann Arbor, MI 48109, USA.
{jnwang,ebolson}@umich.edu; http://april.eecs.umich.edu

- A stochastic gradient descent-based bootstrapping procedure for Gauss-Newton solvers that increases robustness to local minima
- An evaluation that demonstrates the improved robustness of our approach against initialization error

II. Prior Work

In non-robust pose graph SLAM, Gauss-Newton methods that rely on factorizing the information matrix are often used to solve the resulting non-linear optimization problem. Such methods are sensitive to initial conditions, and a poor initialization can prevent the algorithm from converging upon the true solution. A stochastic gradient descent method has been shown to increase convergence from poor initial estimates in non-mixture pose graphs [4].

Previous methods to increase robustness involved front-end validation of loop closures. Such methods include Atlas's nearest-neighbor feature matching [5], Joint Compatibility Branch and Bound (JCBB) [6], and graph consistency approaches such as CCDA [7], SCGP [8], and spectral clustering [9].

Robust SLAM formulations are also not new. Latif, Cadena, and Neira developed the Realizing, Reversing, Recovering (RRR) algorithm [1] to check the consistency between clusters of edges in a graph. This algorithm is based on the observation that correct loop closures are mutually consistent, while incorrect loop closures are often outliers. It bears the most resemblance to conventional front-end verification systems, and could be considered an online extension of the joint compatibility test, where loop closure hypotheses can be reexamined for consistency in the future.

Sunderhauf and Protzel introduce the concept of switchable constraints [3]. By adding a switch variable for each loop closure, which takes on a real value between 0 and 1, each edge can be turned on, off, or partially on by having the switch variable take on a value between 0 and 1. These variables essentially modify the shape of the pose graph, allowing the optimization to turn off false loop closure constraints.

Relative constraints have been used in existing SLAM techniques. Olson et al. showed its usefulness for pose graph optimization [4], which was later extended by Grisetti et al. to use a spanning tree parameterization [10]. Relative bundle adjustment techniques have been shown to efficiently solve the general SLAM problem, which includes both poses and landmarks [11], [12].

Olson and Agarwal described the Gaussian max-mixtures SLAM model [2], on which this work is based. This formulation makes the edge constraints themselves more expressive, allowing multimodal distributions to be used as edge constraints. In particular, loop closure edges can use a robust cost function which incorporates a high-covariance "null hypothesis" component. This work introduces the Cholesky-MM algorithm, which uses a Gauss-Newton solver to solve max-mixtures SLAM problems.

III. Background

This work focuses on a special case of SLAM that only considers pose-to-pose constraints. Such a problem arises naturally from pose matching methods, but can also be obtained by marginalizing out landmarks. Landmarks can alternatively be included directly in the state vector, but our experiments only consist of pose-to-pose links.

Initial state estimates have a significant effect on the quality of the final solution. Two common approaches are using only the odometry information and computing a minimum spanning tree (MST) using all available edges. In our approach, we use odometry as the initial pose estimate. The goal is to optimize over the state vector x, representing a series of robot poses, given our observations z. Making the usual assumption that individual observations are conditionally independent, we can factor this probability distribution.

$$P(x|z) \propto \prod_i P(z_i|x)$$

Traditionally, each edge constraint $P(z_i|x)$ is assumed to be Gaussian with some covariance Σ_i.

$$P(z_i|x) \propto e^{-\frac{1}{2}(f_i(x)-z_i)^T\Sigma_i^{-1}(f_i(x)-z_i)}$$

The observation model $f_i(x)$ is non-linear, so we must linearize $f_i(x) \approx f_i(x_0) + J_i\Delta x$. The maximum likelihood trajectory may be computed by minimizing the negative log probability (eq. 1). We use the notation $r = z - f(x_0)$ to represent the residual.

$$-\log P(x|z) \propto \sum_i (J_i\Delta x - r_i)^T\Sigma_i^{-1}(J_i\Delta x - r_i) \quad (1)$$

A. Max-Mixtures Model

The max-mixtures model represents each edge distribution $P(z_i|x)$ as a mixture of Gaussians based on a max operator. A mixture model allows us to represent more complex distributions. In our application, we represent the uncertainty of a loop closure as a Gaussian component (the "null hypothesis") with a very large covariance.

$$P(z_i|x) \propto \max_j w_j\mathcal{N}(\mu_{ij}, \Sigma_{ij})$$

The Gaussian components each have mean μ_{ij}, covariance Σ_{ij}, and mixing weights w_j. The max allows the log operator to be "pushed inside" the max (eq. 2). Minimizing this quantity results in a maximum likelihood estimator that selects the most probable Gaussian component for each edge constraint.

$$\begin{aligned}-\log P(x|z) = \sum_i \min_j \Big[&-\log(w_j) + \frac{1}{2}\log(|\Sigma_{ij}|) + \\ &\frac{1}{2}(J_{ij}\Delta x - r_{ij})^T\Sigma_{ij}^{-1}(J_{ij}\Delta x - r_{ij})\Big]\end{aligned} \quad (2)$$

Because of the max selection operator, for each edge i the Jacobian J_{ij} and residual r_{ij} is computed for a single Gaussian component j.

B. Stochastic Gradient Descent

The stochastic gradient descent (SGD) solver optimizes over a single edge constraint i at a time. This allows it to both explore and escape from poor local minima, since different edges will pull the graph in different directions. The cost function χ_i^2 for a single constraint is:

$$\chi_i^2 = \min_j \left[(J_{ij}\Delta x - r_{ij})^T \Sigma_{ij}^{-1} (J_{ij}\Delta x - r_{ij}) \right] \quad (3)$$

Assuming mixture component j is active, we will ignore the other components and drop the j subscript from this point. We begin by finding the gradient of the cost function with respect to Δx. At the current state, where $\Delta x = 0$, the gradient is:

$$\begin{aligned} \nabla \chi_i^2 &= 2J_i^T \Sigma_i^{-1} J_i \Delta x - 2J_i^T \Sigma_i^{-1} r_i \\ &= -2J_i^T \Sigma_i^{-1} r_i \end{aligned}$$

We correct our state estimate by Δx in the direction of the gradient, where the magnitude of Δx is dictated by the learning rate (or step size) λ.

$$\Delta x = -\lambda \nabla \chi_i^2 = 2\lambda J_i^T \Sigma_i^{-1} r_i$$

C. Incremental State Space

The stochastic gradient descent method lends itself to an alternative state space representation. Many SLAM implementations use a global state space, storing an (x, y, θ) pose for each node. This representation preserves sparsity in the information matrix, which is important for computational efficiency, since each edge directly affects only two pose nodes. This overlooks an important property of this problem: a robot's trajectory is cumulative. Intuitively, this means that if a single pose needs some amount of correction, then it is likely that the following poses also need similar correction.

In this method, we use the incremental state space as described in [13]. Instead of using a global state vector $\mathbf{x} = (x_0, y_0, \theta_0, x_1, y_1, \theta_1, \ldots)$, we use the incremental representation $\dot{\mathbf{x}}$. The conversion from global to incremental is given below.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}_0 \\ \hline \\ \dot{\mathbf{x}}_i \\ \\ \hline \vdots \end{bmatrix} = \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{\theta}_0 \\ \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \\ \vdots \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \\ x_i - x_{i-1} \\ y_i - y_{i-1} \\ \theta_i - \theta_{i-1} \\ \vdots \end{bmatrix}$$

We model the each observation edge as a rigid body constraint T between poses $\mathbf{x}_a$ and $\mathbf{x}_b$. In terms of global poses $\mathbf{x}_k = (x_k, y_k, \theta_k)$, the equations for the observation model are:

$$f_T(\mathbf{x}) = \begin{bmatrix} \cos(\theta_a)(x_b - x_a) + \sin(\theta_a)(y_b - y_a) \\ -\sin(\theta_a)(x_b - x_a) + \cos(\theta_a)(y_b - y_a) \\ \theta_b - \theta_a \end{bmatrix}$$

We can express the same transformation in terms of incremental poses $\dot{\mathbf{x}}_k = (\dot{x}_k, \dot{y}_k, \dot{\theta}_k)$. Note that the global position $\mathbf{x}_a$ is the sum of the incremental states $\dot{\mathbf{x}}_k$ from 0 to a; for example, $\theta_a = \sum_{k=0}^{a} \dot{\theta}_k$. Substituting in these summations:

$$f_T(\dot{\mathbf{x}}) = \begin{bmatrix} \cos(\sum_{k=0}^{a} \dot{\theta}_k)(\sum_{k=a+1}^{b} \dot{x}_k) + \sin(\sum_{k=0}^{a} \dot{\theta}_k)(\sum_{k=a+1}^{b} \dot{y}_k) \\ -\sin(\sum_{k=0}^{a} \dot{\theta}_k)(\sum_{k=a+1}^{b} \dot{x}_k) + \cos(\sum_{k=0}^{a} \dot{\theta}_k)(\sum_{k=a+1}^{b} \dot{y}_k) \\ \sum_{k=a+1}^{b} \dot{\theta}_k \end{bmatrix}$$

We can now compute the Jacobian of this rigid body constraint with respect to state $\dot{\mathbf{x}}_k$. (Below, x_a, x_b, y_a, y_b, and θ_a are used as shorthand for their respective summations.)

$$J = \frac{\partial f_T}{\partial \dot{\mathbf{x}}_k} = \begin{cases} \begin{bmatrix} 0 & 0 & \begin{matrix} -(x_b - x_a)\sin(\theta_a) \\ +(y_b - y_a)\cos(\theta_a) \end{matrix} \\ 0 & 0 & \begin{matrix} -(x_b - x_a)\cos(\theta_a) \\ -(x_b - x_a)\sin(\theta_a) \end{matrix} \\ 0 & 0 & 0 \end{bmatrix} & k \leq a \\ \begin{bmatrix} \cos(\theta_a) & \sin(\theta_a) & 0 \\ -\sin(\theta_a) & \cos(\theta_a) & 0 \\ 0 & 0 & 1 \end{bmatrix} & a < k \leq b \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \text{otherwise} \end{cases} \quad (4)$$

Note that the Jacobian for $k \leq a$ has non-zero values. Specifically, any rotation of node a will cause an error with respect to the rigid-body transformation T. However, these terms can have very large magnitudes and can cause the optimization to diverge. Thus, following our earlier work, we set these terms to zero. This approximation has good empirical performance, and can be justified intuitively, since a rigid body link between a and b should not have a major effect on nodes prior to a.

IV. Method

We now present the SGD-MM (Stochastic Gradient Descent-Max Mixtures) algorithm, along with a description of some practical implementation details. We also present the hybrid SGD-Cholesky-MM algorithm, which further explores the search space and improves performance on graphs with "strong" false local minima.

A. SGD-MM

For a given edge between nodes a and b, the correction Δx is distributed among $b-a$ nodes, so the total correction s is scaled by $b - a$. Correction is distributed according to the information matrix $J^T \Sigma^{-1} J$, so that less confident nodes (smaller information matrix entries) get more correction, while more confident nodes get less correction. Since it would be computationally expensive to build the information matrix, we instead approximate it using a diagonal matrix M, as shown in the first part of Algorithm 1. The diagonal terms of each W are summed so that M_j approximates the j^{th} diagonal block of $J^T \Sigma^{-1} J$. This can be viewed as a preconditioning step to improve the convergence of gradient descent, where M is a diagonal preconditioning matrix.

This implementation introduces a different scaling term Γ^{-1}, which scales the correction by the largest value of

Algorithm 1 SGD-MM

1: **procedure** SGD-MM(λ_0)
2: $t = 1$
3: **repeat**
4: Randomly permute edges
5:
6: ▷ *Compute M, an approximation of $J^T\Sigma^{-1}J$*
7: Initialize $M_j = 0$ for all j
8: **for** each edge i between edges a, b **do**
9: Compute the Jacobian J_i (Eq. 4)
10: $W = J_i^T\Sigma_i^{-1}J_i$
11: **for** $j \in [a+1, b]$ **do**
12: $M_j = M_j + W$
13: **end for**
14: **end for**
15: $\Gamma = \arg\min |M_j|$
16:
17: ▷ *Compute cumulative weights C*
18: $C_j = \sum_{k=0}^{j} M_k^{-1}$
19: Generate weighted error distribution tree using C
20:
21: ▷ *Modified stochastic gradient descent step*
22: **for** each edge i between edges a, b **do**
23: Compute the Jacobian J_i and residual r_i
24: $\lambda = \lambda_0/t$
25: $s = -(b-a)\lambda\Gamma^{-1}J_i^T\Sigma_i^{-1}r_i$
26: $s_{max} = x_b - (x_a \oplus T_{ab})$
27: $s = \text{clamp}(s, s_{max})$
28: distribute$(a+1, b, s)$
29: **end for**
30:
31: $t = t + 1$
32: **until** converged
33: **end procedure**

M_j^{-1}. This term can be seen as an approximation for $(J^T\Sigma^{-1}J)^{-1}$, the Hessian term in the Gauss-Newton algorithm. This scaling ensures that high confidence measurements with small covariances will not cause extremely large corrections. Finally, the magnitude of s is clamped according to the observation T_{ab}. Since we can calculate the value of s that will exactly satisfy the constraint, we can prevent gradient descent from drastically overshooting.

B. Learning Rate

The learning rate λ has a significant effect on the performance of stochastic gradient descent. When λ is high, the solver takes large steps through the search space and is more likely to jump between different local minima. We ensure convergence by using a learning rate schedule that decays over time. Specifically we use the harmonic series $\lambda = \lambda_0/t$, where t is the iteration number, as suggested by Robbins and Monro [14].

This leaves the initial learning rate λ_0 as a free parameter. In a sense, this parameter captures how much exploration will be performed on a pose graph. This is not a computable quantity, but we can characterize some of the variables that make a graph more complex: the amount of initialization noise, the total path length, and the covariances of the edge constraints. We experimentally found an initial learning rate that worked well for the majority of our datasets, but it is difficult to generally prescribe a way of choosing λ_0.

Other learning rate schedules include "Search Then Converge" [15], which formalizes the intuitive notion of separate exploration and convergence phases. However, since the harmonic learning rate schedule seemed to work for a variety of graphs, we did not attempt to use more sophisticated rate schedules.

C. Implementation and Running Time

Although useful for the derivation above, in practice we do not need to store the individual values as $(\dot{x}_i, \dot{y}_i, \dot{\theta}_i)$. Instead, we store the pose normally as (x, y, θ) and use an error-distribution tree [13]. The error-distribution tree efficiently distributes correction among a contiguous range of states, achieving the same effect as the incremental representation while storing poses in the more useful global representation.

Suppose we have a pose graph with N nodes and E edges. The asymptotic complexity of each iteration is determined by the loops which iterate over each edge (lines 8-14 and 22-29), since $E \geq N$ in a pose graph. Inside both of these loops, some amount of correction is added to a contiguous block of nodes from $a+1$ to b (lines 11-13 and 28). The error distribution tree allows us to accomplish this operation in $\Theta(\log N)$ time. Each pose can also be computed from the error distribution tree in $\Theta(\log N)$ time. Therefore, the running time of each SGD step (lines 23-28) is still $\Theta(\log N)$, and the running time of each iteration is $\Theta(E \log N)$.

D. SGD-Cholesky-MM

For many pose graphs, SGD-MM produces useful solutions on its own. It may also be used to bootstrap the Cholesky-MM solver, a Gauss-Newton based method. The SGD-MM optimization procedure is good at performing a rough alignment of poses, while the Cholesky solver's quadratic convergence is ideal for quickly finding the nearest local minimum. At a fixed interval, it takes the graph produced by SGD and runs Cholesky until convergence (or divergence). Our algorithm repeatedly runs this procedure, keeping the refined graph with the lowest χ^2 error. This best-so-far policy allows us to compensate for SGD's tendency to jump between local minima.

Since the first solution is exactly the same solution returned by Cholesky-MM, its worst-case performance is no worse than Cholesky-MM. We will later show that this procedure produces significantly better graphs than either SGD-MM or Cholesky-MM, and will offer some reasons based on an analysis of the failure cases.

V. Results

Our results come from synthetically-generated "Manhattan" worlds as well as real-world data from the Intel dataset. Our primary claim is that our method increases robustness to

Algorithm 2 SGD-Cholesky-MM

procedure SGD-CHOLESKY-MM(k):
  best = initial graph
  repeat
    Run Cholesky-MM on a copy of the graph
    best = $\min(\text{best}, \arg\min \chi^2(\text{graph}))$
    Run k iterations of SGD-MM
  until interrupted
  return best
end procedure

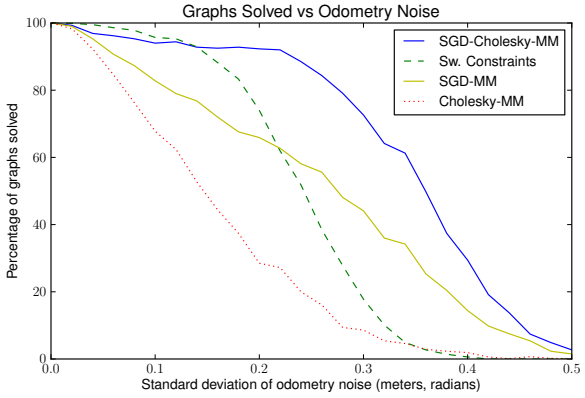

Fig. 2: Percentage of graphs solved for 1000 randomly generated Manhattan worlds with 400 poses, 800 true loop closures, 40 false loop closures, and varying odometry noise. Initial learning rate was $\lambda_0 = 5$ and SGD-Cholesky-MM used $k = 50$. SGD-Cholesky-MM outperforms comparison methods, especially for very poor odometry initializations.

poor initializations. Of course, it is possible to find specific graphs where this is not true, so the bulk of our analysis is based on large numbers of random graphs so that we can statistically characterize the benefits of our approach.

A. Manhattan Worlds

In the Manhattan world experiment, we evaluated both variants of SGD against Cholesky-MM and switchable constraints on a 1000-graph dataset. (To evaluate switchable constraints, we used the authors' original implementation in the g2o framework [16].) Each graph was corrupted with increasing amounts of odometry noise. Fig. 1 shows the result of one of these experiments.

Fig. 2 shows the result of this test for 1000 randomly-generated graphs at increasing noise levels. Mean squared distance error (MSE) compared to ground truth was used to evaluate the map quality. SGD-MM was run until convergence, which we defined as when the map changed an average of $|\Delta x| < 0.001$ over 500 iterations. A threshold of MSE < 10 was used to designate a graph as correctly solved. This corresponds to a map which is usable but not necessarily metrically perfect.

An analysis of the performance gap between SGD-MM and SGD-Cholesky-MM reveals a specific weakness of both iterative MM solutions, SGD and Cholesky. In many of the

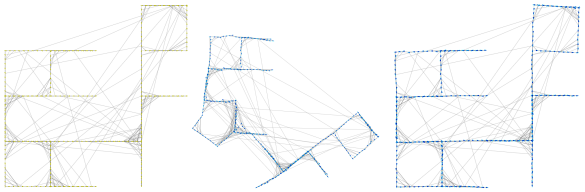

Fig. 3: Example of an incorrect local minimum which causes SGD-MM to fail. On this peninsula-like graph, the incorrect solution is a local minimum with a larger basin of convergence than the correct solution. Because SGD-Cholesky-MM explores multiple local minima and selects the overall best solution, it is able to find the correct solution.

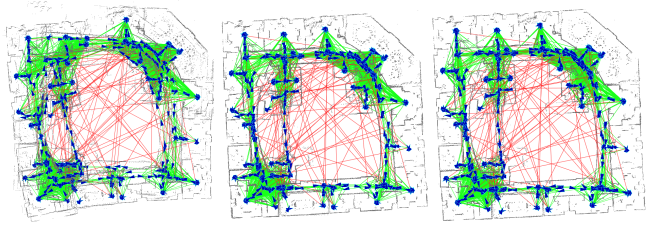

Fig. 4: Intel dataset with 100 false loop closures added (shown in red). Both SGD-based methods are able to produce accurate maps with the walls in alignment.

graphs where SGD-MM failed, the true solution was not the most stable local minimum. Fig. 3 shows an example failure case where the incorrect solution has a larger basin of convergence than the correct solution – both SGD-MM and Cholesky-MM converge upon this incorrect solution most often from a variety of poor initializations.

SGD-Cholesky-MM (Algorithm 2) is designed to improve performance on such graphs by using SGD to generate initializations for Cholesky-MM. Stochastic gradient descent has the tendency to jump in and out of local minima. Running Cholesky-MM to convergence fully explores these local minima. Selecting the graph with lowest χ^2 error gives preference to "better" local minima, even if its basin of convergence is not as large. This is responsible for the improvement over either Cholesky-MM and SGD-MM alone.

B. Intel Dataset

Our test run on the Intel dataset demonstrates the applicability of our method to real-world data. SGD-MM is able to find the true solution and achieve the precision necessary to generate a map (Fig. 4). Since SGD-Cholesky-MM solution converges faster upon the true minimum, it produces a somewhat more refined map.

C. Ring and Ring City Datasets

The Ring and Ring City datasets introduced in [17] illustrate an extreme example where the true solution has a minimal basin of convergence. In this case, we expect Cholesky-MM to fail since its initial position is already in a strong local minimum. SGD methods are more willing to

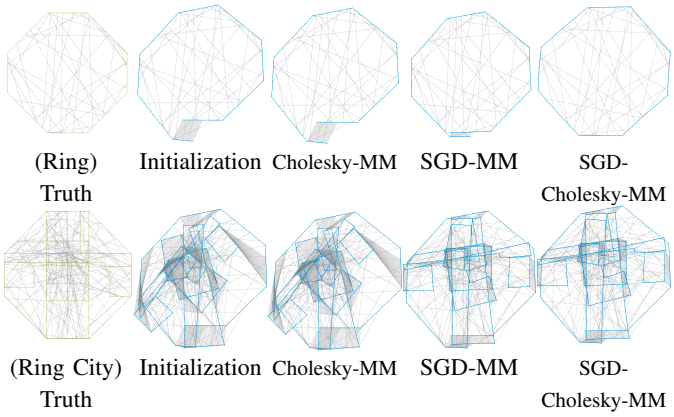

Fig. 5: Results of SGD-MM on Ring and Ring City datasets corrupted with 40 and 200 false loop closures, respectively. Cholesky-MM fails because the graphs start in a stable local minimum. Because SGD methods are more willing to depart from local minima, they are more robust to ring-type failures.

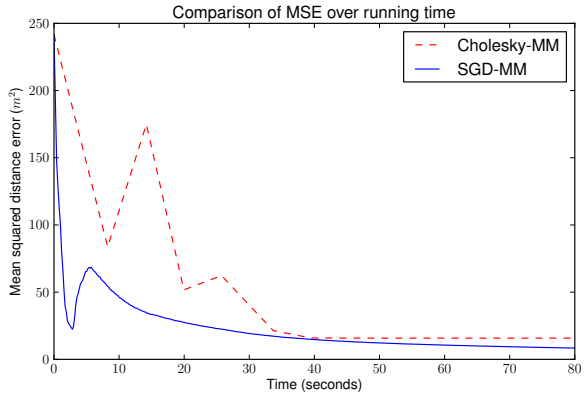

Fig. 6: Plot of error over time on one trial of the CSW 3500-node dataset, with 2100 true and 1000 false loop closures. SGD-MM has reduced the initial error by about 80% before Cholesky-MM has finished its first iteration. (SGD-MM takes a bad step at around $t = 3$s, but recovers.) Both were run on a single 3.4GHz core of an Intel Core i7 using the Java-based april.graph library.

depart from local minima and explore the search space, as shown in Fig. 5. SGD-Cholesky-MM is able to correctly close the loop for the Ring dataset. The Ring City dataset is even more challenging since the true solution requires closing many such ring-type loops, and SGD-Cholesky-MM is not able to close all the loops. However, SGD-based methods show increased robustness to ring-type failures, a specific case of graphs with strong local minima that Cholesky-MM cannot solve.

D. CSW Dataset

We use the CSW dataset [4] to evaluate the running time of our algorithm. CSW is a Manhattan graph with 3500 nodes and 5600 edges. We expect Cholesky-MM to slow down considerably, since the false loop closures cause increased fill-in of the information matrix. SGD-MM is unaffected by fill-in, since its runtime grows as $\Theta(E \log N)$ without regard to the sparsity of the information matrix. As shown in Fig. 6, SGD-MM reduces error more quickly than Cholesky-MM.

VI. Conclusion

We have presented two variants of an SGD solver for robust max-mixture pose graphs that performs well even from poor initializations. SGD-MM can stand alone as a computationally efficient solver for graphs with low initial error. SGD-Cholesky-MM greatly extends the effectiveness of the max-mixtures model on graphs with difficult local minima. Our evaluation characterizes the performance of these two methods and provides some insight into the structure of max-mixtures graph optimization problems.

Acknowledgments

This work was funded by DoD Grant FA2386-11-1-4024.

References

[1] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[2] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *International Journal of Robotics Research*, vol. 32, no. 7, pp. 826–840, July 2013.

[3] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1879–1884.

[4] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2006, pp. 2262–2269.

[5] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework," *International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[6] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 6, pp. 890–897, 2001.

[7] T. Bailey, "Mobile robot localisation and mapping in extensive outdoor environments," Ph.D. dissertation, Citeseer, 2002.

[8] E. Olson, M. Walter, J. Leonard, and S. Teller, "Single cluster graph partitioning for robotics applications," in *Proceedings of Robotics Science and Systems*, 2005, pp. 265–272.

[9] E. Olson, "Recognizing places using spectrally clustered local matches," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1157–1172, December 2009.

[10] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[11] D. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment." in *Robotics: Science and Systems*, 2009.

[12] J.-L. Blanco, J. Gonzalez-Jimenez, and J.-A. Fernandez-Madrigal, "Sparser relative bundle adjustment (SRBA): Constant-time maintenance and local optimization of arbitrarily large maps," in *Robotics and Automation (ICRA), IEEE International Conference on*, May 2013, pp. 70–77.

[13] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.

[14] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[15] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search." IEEE Press, 1992.

[16] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2011.

[17] N. Sunderhauf and P. Protzel, "Switchable constraints vs. max-mixture models vs. RRR: A comparison of three approaches to robust pose graph SLAM," in *Robotics and Automation (ICRA), IEEE International Conference on*, Karlsruhe, Germany, 2013.