MS2MP: A Min-Sum Message Passing Algorithm for Motion Planning

Salman Bari, Volker Gabler and Dirk Wollherr*

Abstract-Gaussian Process (GP) formulation of continuoustime trajectory offers a fast solution to the motion planning problem via probabilistic inference on factor graph. However, often the solution converges to in-feasible local minima and the planned trajectory is not collision-free. We propose a message passing algorithm that is more sensitive to obstacles with fast convergence time. We leverage the utility of min-sum message passing algorithm that performs local computations at each node to solve the inference problem on factor graph. We first introduce the notion of compound factor node to transform the factor graph to a linearly structured graph. We next develop an algorithm denoted as Min-sum Message Passing algorithm for Motion Planning (MS2MP) that combines numerical optimization with message passing to find collisionfree trajectories. MS2MP performs numerical optimization to solve non-linear least square minimization problem at each compound factor node and then exploits the linear structure of factor graph to compute the maximum a posteriori (MAP) estimation of complete graph by passing messages among graph nodes. The decentralized optimization approach of each compound node increases sensitivity towards avoiding obstacles for harder planning problems. We evaluate our algorithm by performing extensive experiments for exemplary motion planning tasks for a robot manipulator. Our evaluation reveals that MS2MP improves existing work in convergence time and success rate.

I. INTRODUCTION

Kinematic motion planning focuses on finding a trajectory in a robot's configuration space from start state to goal state while satisfying multiple performance criteria such as collision avoidance, joint limit constraints and trajectory smoothness. There are several motion planning approaches that have been proposed so far. These approaches can be roughly divided into two broad categories: sampling-based algorithms and optimization-based algorithms. Samplingbased algorithms [1]–[3] efficiently find collision free trajectories by probing the configuration space and checking the feasibility of robot's configuration. However, trajectories produced by sampling-based algorithms are not smooth and therefore require further post processing.

Trajectory optimization algorithms [4]–[6] start with an initial trajectory that may not be collision-free and then minimize an objective function that penalizes collisions and non-smooth configurations. A drawback of these approaches is that they only tend to find locally optimal solutions and need fine-discritization of trajectory into way points for collision checking in complex environments. Gaussian Process (GP) formulation of continuous-time trajectories [7] can overcome this challenge.



Fig. 1: Different Robot configurations during MS2MP planning phase for an autonomous disassembly setup: (a) at N = 0, (b) at N = 4, (c) at N = 7, (d) at N = 10.

(d)

(c)

Recently, the Gaussian Process Motion Planning (GPMP2) framework [8] has been proposed that represents trajectory as a GP and finds collision-free trajectories with fast, structure-exploiting inference. It fuses all the planning problem objectives which are represented as factors and solves non-linear least square optimization problem via numerical (Gauss-Newton or Levenberg-Marquardt) methods. Although, GPMP2 is fast but the batch non-linear least square optimization approach makes it vulnerable to converging to in-feasible local minima. The approach to combine all the factors of the graph makes it faster than state of the art motion planning algorithms but it comes at the cost of being more prone to getting stuck in local minima. Graph reoptimization could naively help to get out of the in-feasible local minima but it increases the computation time.

In this work, we explore the min-sum algorithm for finding the maximum a posteriori (MAP) estimation of a motion planning problem on factor graph. However, min-sum cannot be efficiently adopted due to the non-linear factors of probabilistic motion models. So, we propose a hybrid algorithm called Min-sum Message Passing algorithm for Motion Planning (MS2MP) that combines numerical optimization methods with min-sum message passing algorithm to solve non-linear factors for MAP estimation. MS2MP combines all the factors attached to the same variable node to form a compound factor node. Then, the messages are passed among adjacent factor and variable nodes to find the MAP estimation of complete graph that generates collision-

^{*}Authors are associated with the Chair of Automatic Control Engineering (LSR), Department Electrical of and Computer Engineering, Technical University of Munich, Germany [s.bari|v.gabler|dw]@tum.de

free trajectories.

II. RELATED WORK

The probabilistic inference view on motion planning is introduced by Toussaint et al. [9], [10]. The motion planning problem is formulated as a probabilistic model which represents the performance criteria (e.g. avoiding obstacles, reaching a goal) as objectives and use probabilistic inference to compute a posterior distribution over possible trajectories. Belief propagation is proposed to solve the approximate inference problem of finding feasible trajectories while factor graphs are used to represent the planning problem.

The GP formulation of continuous-time trajectories [7] made the probabilistic view on motion planning problem more lucrative because it offers sparse parametrization of the trajectory. Here, a GP is used to represent trajectories as functions that map time to the robot state which provides the benefit of querying the trajectory at arbitrary time steps by exploiting the Markov property of GP priors produced from linear time variant, stochastic differential equation (LTV-SDE). Recently, B-spline [11] and kernel methods [12] have been used in a similar manner to represent trajectories with fewer states in motion planning problems.

The formulation of continuous-time trajectory as GP paved the way to consider an alternative approach for solving the inference problem on factor graph. The GPMP2 algorithm combines all the factor nodes and solves the inference problem using batch numerical optimization approach. Simultaneous Localization and Mapping (SLAM) problems [13] have been also solved in the similar manner. The algorithm often converges to in-feasible local minima due to the holistic approach of solving factor graph and results in trajectories that are not collision-free. In case of batch optimization, several ideas like random initializations and graph-based initialization [14] exist to improve results but they do not deal with the inherent approach of combining all factors and optimizing it at once.

Intuitively, the better approach is to evaluate each node of the factor graph separately to obtain improved results. This is the standard characteristic of message passing algorithms, which in its purest form, allows in-place inference on a factor graph with entirely local processing at graph nodes. Over the past few years, there has been an increased interest in a message passing based optimization algorithm for graphical models, called min-sum message passing algorithm [15], [16]. Message passing algorithms have been used to solve NP-hard combinatorial optimization problems [17], [18]. The min-sum algorithm is equivalent to the max-product algorithm, and is closely related to belief propagation also known as sum-product algorithm [19].

We build upon the success of min-sum message passing algorithm for general factor graphs and thus give an outline how this method can be adopted for GP-based motion planning problems. We transform the factor graph by introducing compound factor nodes and then optimize each node locally to generate messages to compute MAP estimation of the factor graph.



Fig. 2: An example trajectory planning problem formulation on a factor graph. White circles show support states and square boxes show factors.

III. PROBLEM FORMULATION

Suppose a trajectory is represented by $x(t) : t \to \mathbb{R}^D$, where D is the dimensionality of the state. p(x) is the prior on x that actually encourages smooth trajectories and fixes start and goal states, $l(x; \mathbf{e})$ is the likelihood of states xrepresenting collision-free events \mathbf{e} on the trajectory. For example, e_i could be a binary event with $e_i = 0$ if trajectory is collision-free and $e_i = 1$ if trajectory is in collision. Given the prior and likelihood, the optimal trajectory x^* is found by the MAP estimation, which generates the trajectory that maximizes the conditional posterior $p(x|\mathbf{e})$,

$$\boldsymbol{x}^* = \operatorname*{arg\,max}_{\boldsymbol{x}} \underbrace{p(\boldsymbol{x}|\mathbf{e})}_{p(\boldsymbol{x})l(\boldsymbol{x};\mathbf{e})}$$
 (1)

The posterior distribution can be equivalently formulated as MAP inference problem on a *factor graph*. Factor graph formulation for motion planning problem (1) is described in detail in the following section.

A. Factor Graph Formulation

The motion planning problem formulation in (1) consists of two components, prior and likelihood. Our objective is to find a trajectory parameterized by x that satisfies collisionfree events 'e'.

The continuous-time trajectory prior p(x) is represented as functions sampled from GPs [7]. GP based trajectory priors offer the benefit of representing the full trajectory x by means of N support states $x_i \in \mathbb{R}^D$. Given these, a trajectory can be efficiently interpolated between two consecutive support states through Gaussian Process regression (GPR). Similar to prior work [8], we assume x to follow a joint Gaussian distribution

$$\boldsymbol{x}(t) = \left[\boldsymbol{x}_{0},...,\boldsymbol{x}_{N}\right]^{T} \sim \mathcal{N}\left(\boldsymbol{\mu}\left(\boldsymbol{t}\right),\boldsymbol{\mathcal{K}}\right),$$
 (2)

for a set of times $t = t_0, \ldots, t_N$, where $\mu(t)$ is a mean vector and $\mathcal{K}(t, t')$ is the covariance kernel. The kernel \mathcal{K} induces smoothness and puts constraints on start and goal states. We follow [7] by considering a structured kernel

generated from LTV-SDE, such that the GP prior of x results in

$$p(\boldsymbol{x}) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{\mu}\|_{\boldsymbol{\mathcal{K}}}^{2}\right\},$$
 (3)

where $||x - \mu, ||_{\mathcal{K}}^2$ is the Mahalanobis distance. The likelihood function which specifies the probability of avoiding obstacles is also defined as a distribution in the exponential family. The collision-free likelihood is

$$l(\boldsymbol{x}; \mathbf{e}) = \exp\left\{-\frac{1}{2} \|\mathbf{h}(\boldsymbol{x})\|_{\boldsymbol{\Sigma}_{\text{obs}}}^{2}\right\}, \qquad (4)$$

where $\mathbf{h}(x)$ represents the obstacle cost, and $\Sigma_{\rm obs}$ is a hyperparameter matrix.

In the context of GP-based motion planning, we need to model a belief over continuous, multivariate random variables $\boldsymbol{x}_i \in \mathbb{R}^D$. It can be formulated on a factor graph $\mathcal{G} = (\mathcal{X}, \mathcal{F}, \mathcal{E})$ using conditional probability density functions (PDFs) $p(\boldsymbol{x}_i | \mathbf{e})$ over the variables \boldsymbol{x}_i , given the constraints e. The bipartite graph \mathcal{G} factorizes the conditional distribution over \boldsymbol{x} as

$$p(\boldsymbol{x}|\mathbf{e}) \propto \prod_{m=1}^{M} f_m(\mathcal{X}_m),$$
 (5)

given a subset of M factor nodes $f_m \in \mathcal{F}$ on an adjacent subset of variable nodes $\mathcal{X}_m \in \mathcal{X}$, that are set in relation via the edges \mathcal{E} of the factor graph. Fig. 2 shows an example factor graph for a motion planning problem with N = 4 support states that describe a collision-aware motion planning problem. Thus, the prior is factorized as

$$p(\boldsymbol{x}) \propto f_0^p(\boldsymbol{x}_0) f_N^p(\boldsymbol{x}_N) \prod_{i=1}^{N-1} f_i^p(\boldsymbol{x}_i) f_i^{\text{GP}}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}), \quad (6)$$

where $f_0^p(\boldsymbol{x}_0)$ and $f_N^p(\boldsymbol{x}_N)$ put constraints on fixing the start state $\boldsymbol{\mu}_0$ and goal state $\boldsymbol{\mu}_N$. Constraining the deviation from the prior is included via

$$f_{i}^{p}\left(\boldsymbol{x}_{i}\right) = \exp\left\{-\frac{1}{2}\left\|\boldsymbol{x}_{i}-\boldsymbol{\mu}_{i}\right\|_{\boldsymbol{\mathcal{K}}_{i}}^{2}\right\},$$
(7)

in case they are not colliding with obstacles, while the GP prior of the trajectory is incorporated via

$$f_{i}^{\text{GP}}(\boldsymbol{x}_{i}, \boldsymbol{x}_{i+1}) = \exp\left\{-\frac{1}{2} \|\boldsymbol{\Phi}(t_{i+1}, t_{i})\boldsymbol{x}_{i} - \boldsymbol{x}_{i+1} + \boldsymbol{\mu}_{i,i+1}\|_{\boldsymbol{Q}_{i,i+1}}^{2}\right\},$$
(8)

using state transition matrix $\mathbf{\Phi}(t_{i+1}, t_i)$ and power spectral density matrix $\mathbf{Q}_{i,i+1}$ as introduced in [7].

The collision-free likelihood $l(\mathbf{x}; \mathbf{e})$ is factorized with two types of factors, unary obstacle factors f_i^{obs} and interpolated binary obstacle factors $f_{\tau_j}^{\text{intp}}$ as

$$l(\boldsymbol{x}; \mathbf{e}) = \prod_{i=0}^{N} \left\{ f_{i}^{\text{obs}}(\boldsymbol{x}_{i}) \prod_{j=1}^{N_{ip}} f_{\tau_{j}}^{\text{intp}}(\boldsymbol{x}_{i}, \boldsymbol{x}_{i+1}) \right\}, \quad (9)$$

where N_{ip} is the number of interpolated states between two consecutive support states x_i, x_{i+1} and τ_j is the interpolation time. The unary obstacle factor at each support state is defined according to (4). The interpolated binary obstacle factor is defined as

$$f_{\tau_{j}}^{\text{intp}}\left(\boldsymbol{x}_{i}, \boldsymbol{x}_{i+1}\right) = \exp\left\{-\frac{1}{2}\left\|\mathbf{h}_{\tau_{j}}^{\text{intp}}\left(\boldsymbol{x}_{i}, \boldsymbol{x}_{i+1}\right)\right\|_{\boldsymbol{\Sigma}_{\text{obs}}}^{2}\right\}.$$
(10)

The interpolated binary obstacle factor accumulates the collision cost at interpolated states τ_j . This cost information is utilized to update the associated support states by solving MAP inference problem on the factor graph.

IV. MAP INFERENCE VIA MIN-SUM MESSAGE PASSING

Given \mathcal{F} as a set of Gaussians, (5) results in a product of Gaussians. Thus, finding the most probable values for \boldsymbol{x} is equivalent to minimizing the negative log of the probability distribution via

$$\boldsymbol{x}^{*} = \operatorname*{arg\,min}_{\boldsymbol{x}} \sum_{m=1}^{M} f_{m}\left(\mathcal{X}_{m}\right). \tag{11}$$

As the individual factors are nonlinear, it is in-feasible to apply min-sum directly to solve the optimization problem in (11). It requires to linearize factors at each step resulting high computation cost. However, we propose to explicitly alter the factor graph structure by introducing *compound factor nodes* and then adopt the min-sum algorithm in which a local node objective function is optimized using the Gauss-Newton method. Our proposed algorithm decreases the tendency of converging to local minima due to distributed approach, while containing the convergence properties, as shown in Appendix A. In the following, the concept of compound factor nodes is outlined in detail.

A. Factor Graph with Compound Factor Nodes

The goal of introducing compound factor nodes is to reshape general factor graphs into a linear representation with unique edges between nodes by combining individual factors as exemplarily shown in Fig. 3 for the graph from Sec. III-A.

Assumption 1 A motion planning problem can be fully described by a factor graph \mathcal{G} with \mathcal{F} consisting of two types of factors, unary factors $f_i : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ and binary factors $f_{ij} : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ connected to two consecutive variables $\mathbf{x}_i, \mathbf{x}_j$ where $\{ij\} \in \mathcal{E}$ and |i - j| = 1.

Definition 1 (Compound factor node) Based on assumption 1, a compound unary factor node is defined as

$$\phi_{i}\left(\boldsymbol{x}_{i}\right) = \sum_{i \in \mathcal{X}} f_{i}\left(\boldsymbol{x}_{i}\right), \qquad (12)$$

also denoted as self-potential and binary compound node

$$\psi_i\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sum_{ij \in \mathcal{E}} f_{ij}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), \quad (13)$$

as the edge-potential of the current graph.



Fig. 3: (a) Complete factor graph structure (b) Factors attached to same variable nodes are combined together to form a compound factor nodes. Then, in (c), a linear graph structure is shown. Unary factors attached to variable nodes are not shown explicitly for clarity. Instead, we show that graph consists of self-potentials $\phi_i(\mathbf{x}_i)$ and edge-potentials $\psi_i(\mathbf{x}_i, \mathbf{x}_j)$.

Algorithm 1: MS2MP: Min-Sum Message Passing				
Algorithm for Motion Planning				
Data: trajectory prior x , factor graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$,				
number of support states N				
Result: optimized trajectory x^*				
Init($m{x}$) Initialize messages from prior				
/* Compound Factor Node Formation (Def. 1) $*/$				
for $i \in \mathcal{X}$ do				
$\left[\begin{array}{ccc} \phi_{i}\left(oldsymbol{x}_{i} ight) & \leftarrow \end{array} ight. \sum_{i\in\mathcal{X}}f_{i}\left(oldsymbol{x}_{i} ight)$				
for $ij \in \mathcal{E}$ do				
$\downarrow \psi_{i}\left(oldsymbol{x}_{i},oldsymbol{x}_{j} ight)=\sum_{ij\ \in\ \mathcal{E}}\ f_{ij}\left(oldsymbol{x}_{i},oldsymbol{x}_{j} ight)$				
/* Min-sum messages calculation */				
for $t = 0, 1,, N - 1$ do				
<pre>/* get variable to factor messages */</pre>				
$m_{x \to f}^{t}\left(\boldsymbol{x}_{i}\right) \& = \phi_{i}\left(\boldsymbol{x}_{i}\right) + m_{\left(\mathcal{F}_{x}/f\right) \to x}^{t-1}\left(\boldsymbol{x}_{i}\right),$				
<pre>/* get factor to variable messages */</pre>				
$m_{f ightarrow x}^{t}\left(oldsymbol{x}_{i} ight), orall \psi_{i}\left(oldsymbol{x}_{i},oldsymbol{x}_{j} ight) \in \mathcal{F}$ see (16)				
<pre>/* update (belief) state */</pre>				
$b_{i}^{t}(\boldsymbol{x}_{i}) \leftarrow \phi_{i}(\boldsymbol{x}_{i}) + \sum_{f \in \mathcal{F}_{x}} m_{f \mapsto x}^{t}(\boldsymbol{x}_{i})$				
$oldsymbol{x}_i \leftarrow rgmin ~ b_i^t \left(oldsymbol{x}_i ight)$				

Given Definition 1, (11) results in

$$\boldsymbol{x}^{*} = \arg\min_{\boldsymbol{x}} \sum_{i} \phi_{i} \left(\boldsymbol{x}_{i} \right) + \sum_{ij} \psi_{i} \left(\boldsymbol{x}_{i}, \boldsymbol{x}_{j} \right).$$
(14)

The particular linear representation of factor graph, shown on the right hand side of Fig. 3 is a pair-wise factor graph. Here, min-sum algorithm allows to solve the pair-wise factor graph by iteratively passing messages among nodes.

B. Min-sum Messages Equations

In the min-sum algorithm, each node solves a local optimization problem and traverses messages to the adjacent nodes. Namely, we differentiate between messages passed from variables to factor nodes, denoted as $m_{x\to f}^t$, and messages passed form factor nodes to variable nodes, denoted as $m_{f\to x}^t$. Denoting the adjacent nodes of a factor node as \mathcal{X}_f and the adjacent nodes of a variable node as \mathcal{F}_x , the messages are iteratively updated according to

$$m_{x \to f}^{t}\left(\boldsymbol{x}_{i}\right) = \phi_{i}\left(\boldsymbol{x}_{i}\right) + m_{\left(\mathcal{F}_{x}/f\right) \to x}^{t-1}\left(\boldsymbol{x}_{i}\right),$$
(15)

$$m_{f \to x}^{t}\left(\boldsymbol{x}_{i}\right) = \min_{\boldsymbol{x}_{j}} \left[\psi_{i}\left(\boldsymbol{x}_{i}, \boldsymbol{x}_{j}\right) + m_{x \to f}^{t-1}\left(\boldsymbol{x}_{j}\right)\right], \quad (16)$$

where (\mathcal{F}_x/f) notates the set-theoretic difference, i.e. the set of factor nodes \mathcal{F}_x except the factor f. From Algorithm 1, at time-step t = 0, all the messages from variable nodes to factor nodes are initialized according to the prior in (15). In the next step, the min-marginal of $\psi_i(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is calculated in (16). Given the messages from adjacent factor nodes, the belief of a variable node is approximated via

$$b_{i}^{t}(\boldsymbol{x}_{i}) = \phi_{i}(\boldsymbol{x}_{i}) + \sum_{f \in \mathcal{F}_{x}} m_{f \to x}^{t}(\boldsymbol{x}_{i}).$$
(17)

The minima of decision variable x_i at t^{th} iteration is

$$\boldsymbol{x}_{i}^{t} \propto \operatorname*{arg\,min}_{\boldsymbol{x}_{i}} \ b_{i}^{t}\left(\boldsymbol{x}_{i}\right).$$
 (18)

At each iteration t, each node optimizes a local objective function by merging incoming messages into the node.

We use numerical optimization similar to [8], [13] to solve the local non-linear objective function. Since, each local objective function is composed of multiple factors, (16) and (18) take the form of non-linear least-square problem as

$$m_{f \to x}^{t}(\boldsymbol{x}_{i}) = \min_{\boldsymbol{x}_{j}} \left[\sum_{ij \in \mathcal{E}} \left\{ \frac{1}{2} \left\| \mathbf{h}_{\tau_{j}}^{intp}(\boldsymbol{x}_{i}, \boldsymbol{x}_{j}) \right\|_{\boldsymbol{\Sigma}_{obs}}^{2} \right\} + \sum_{ij \in \mathcal{E}} \left\{ \frac{1}{2} \left\| f_{ij}^{GP}(\boldsymbol{x}_{i}, \boldsymbol{x}_{j}) \right\|_{\boldsymbol{Q}_{i,j}}^{2} \right\}$$
(19)
$$+ m_{x \to f}^{t-1}(\boldsymbol{x}_{j}) \right],$$
$$\boldsymbol{x}_{i}^{*} = \arg\min_{\boldsymbol{x}_{i}} \left[\sum_{i \in \mathcal{X}} \left\{ \frac{1}{2} \left\| \boldsymbol{x}_{i} - \boldsymbol{\mu}_{i} \right\|_{\boldsymbol{\mathcal{K}}_{h}}^{2} \right\} + \sum_{i \in \mathcal{X}} \left\{ \frac{1}{2} \left\| \mathbf{h}_{i}(\boldsymbol{x}_{i}) \right\|_{\boldsymbol{\Sigma}_{obs}}^{2} \right\}$$
(20)
$$+ \sum_{f \in \mathcal{F}_{x}} m_{f \to x}^{t}(\boldsymbol{x}_{i}) \right].$$



Fig. 4: Planned trajectory using MS2MP for a 6 degree-of-freedom (DOF) COMAU Racer 5 robot.

We use the Gauss-Newton algorithm to solve the optimization problems in (19) and (20). Note that our proposed algorithm is different in two aspects from GPMP2 [8]: first, the batch optimization of complete graph is replaced by an iterative local optimization of each node, secondly, message passing is performed to achieve the optimization of the complete factor graph. This behaviour is in fact helpful in avoiding collisions for planning problems when robot has to find its way out of obstacle-rich environment as we empirically show in the experimental validation in the following section.

V. EVALUATION

We evaluated our algorithm on an exemplary motion planning task in a lab environment. Precisely, a 6 DOF COMAU Racer 5 robot is tasked to find collision-free trajectory from the inside of the body of a PC-tower as part of an autonomous disassembly process of Electrical and Electronic equipment. The applicability of the proposed algorithm is also evaluated on the robot platform visualized in Fig. 1. We ran benchmark evaluations against GPMP2 in a simulated scenario. ¹ The simulation scenario is shown in Fig. 4, where the PC-tower is approximated by an occupancy mesh. In this section, we discuss the implementation and evaluation details of our algorithm.

A. GP Prior

We use a constant-velocity prior similar to [8] with the Markovian state consisting of configuration position and velocity. The trajectory is generated from LTV-SDE [7]. The prior factors for the experiment are presented in (6).

B. Collision-free likelihood

Similar to recent state-of-the art motion planning algorithms, e.g. [4], [8], the robot collision body is represented by a set of spheres. We formulate the collision-free unary and binary factors following (9). The obstacle cost function is then obtained by computing the hinge loss

$$\mathbf{c}(d) = \begin{cases} -d + \epsilon & \text{if } d \le \epsilon \\ 0 & \text{if } d > \epsilon \end{cases},$$
(21)

for the spheres. In (21) d(x) represents the signed distance from the sphere to the closest obstacle surface in the workspace, and ϵ is a *safety distance*. ϵ increases the sensitivity of the obstacle constraint before collision can occur. Non-zero obstacle-cost addition enables the robot to not reach too close to the obstacles. For likelihood in (4), the parameter Σ_{obs} is defined as

$$\Sigma_{\rm obs} = \sigma_{\rm obs}^2 \mathbf{I} \tag{22}$$

where the parameter σ_{obs} is used to add weight to the *obstacle cost*.

C. Experiment Setting

Prior trajectory is initialized by a constant velocity straight line from start state to goal state. We initialized the trajectory with N = 11 support states and 4 additional interpolated states between two consecutive support states. In our benchmarks we set $\epsilon = 0.2$ for COMAU Racer arm. The term $\sigma_{\rm obs}$ puts weight on observing the obstacles and it is set according to the planning problem. we set $\sigma_{\rm obs} = 0.001$ for our manipulator planning tasks.

We evaluated the proposed algorithm for 24 unique planning problems for different start configurations inside the PC tower. To make the planning problem much harder the start configurations have been kept very close to the body of PC tower. MS2MP extends the Matlab toolbox from GPMP2 and has thus been benchmarked using identical framework. The benchmarks have been run on a 3.90GHz Intel Core i3-7100 CPU.

D. Discussion

Table I summarizes results for 24 planning problems where a robot has to find its way out of an obstaclerich environment. MS2MP is more successful in finding collision-free trajectories compared to GPMP2 with only approximately a third of the run-time. GPMP2 leads to an early termination of the optimization algorithm due to an increase in absolute error. In this case, a re-optimization of the graph is required in order to naively overcome infeasible local minima. It results in increased run-time for finding collision-free trajectories. However, GPMP2 will still be faster with less success rate if we do not consider the reoptimization step.

TABLE I: Results of 24 planning problems for 6-DOF COMAU Racer robot

	Success (%)	Average time (s)	Max. time (s)
MS2MP	83.3	0.1028	0.1751
MS2MP-no-comp	91.6	0.5625	0.8213
GPMP2	70.8	0.3772	0.3962
GPMP2-no-intp	66.7	0.2724	0.2873

¹For a detailed comparison of GPMP2 against recent state-of-the-art motion planners, we refer the reader to [8], where an extensive benchmark comparison has been outlined already.

We observe that local optimization step at each compound node increases the sensitivity of obstacle avoidance resulting in better success rate in case of MS2MP. A drawback of this approach is that the increased sensitivity towards obstacles affects the trajectory smoothness. In order to overcome this drawback, an additional unary prior factor is introduced in (6) that induces smoothness in case of MS2MP. MS2MPno-comp has the highest success rate because each nonlinear factor is linearized at every message passing step. It results in high computation cost, almost double the runtime as compared to GPMP2 and the output trajectory is not very smooth. We also benchmarked our algorithm against GPMP2 without interpolation (GPMP2-no-intp) with the same number of support states and no interpolated states. Although, its faster than GPMP2, the success rate is lowest among all the evaluated algorithms. We refer the reader to the accompanying video that shows a trajectory planned by MS2MP.

VI. CONCLUSIONS

Formulation of motion planning problems on factor graphs has set the stage for applying different approaches to generate feasible and smooth trajectories. We build upon the idea of GPMP2 using factor graphs in order to represent a motion planning inference problem. However, a drawback of existing work is that by fusing all factors in the graph and solving a global nonlinear least square problem tends to converge to in-feasible local minima. This tendency increases with the complexity of motion planning problem. We believe that this problem can be avoided using message passing techniques.

Message passing is a strong algorithmic framework for solving MAP estimation problem on factor graphs by performing local computations at each node. We proposes an algorithm called MS2MP for finding collision-free trajectories. It performs local computations at individual nodes, thus decreases the chances of converging to in-feasible local minima. A major benefit of MS2MP is that it can be extended to an incremental method that allows planning in dynamic environments, where each node is optimized online. An added benefit of message passing approach is the possibility of performing parallel computation acorss the graph nodes that can further reduce planning time significantly. For future work, we are considering to investigate the incremental planning and parallel computation of the proposed algorithm.

APPENDIX A

A. Convergence Analysis

The min-sum algorithm converges to the global optimum for tree-structured graphs. However, in our proposed algorithm min-sum message passing differs in two aspects. First, we introduce compound factor nodes by merging factors connected to same variable node(s). Secondly, we have nonlinear factors. The crucial point in solving the inference problem for non-linear factors is that at each iteration t, ϕ_i and ψ_i form local objective functions by combining incoming messages into the node. The MAP estimation using the minsum algorithm often fails to converge if the single node local optimization fails to find a unique solution. Its not possible to construct x^* directly if the local node objective function has more than one optimal solution [20]. For this reason, we assume that the local node objective function has a unique minimum.

Assumption 2 For each node with self-potential $\phi_i(\mathbf{x}_i)$ and edge-potential $\psi_i(\mathbf{x}_i, \mathbf{x}_j)$, where $i \in \mathcal{X}$ and $ij \in \mathcal{E}$, the solution produced by numerical optimization of local node objective function always converges to a unique optimum point.

Based on Assumption 2, the obtained beliefs are the minmarginals of the function $\psi_i(x_i, x_j)$. Similar to maxmarginal local optimality condition [20], we can also define local optimality for the compound factor nodes in the graph.

Definition 2 (Local optimality - compound factor nodes) Solution obtained from a local node objective function is locally optimal (min-consistent) according to [21] if for all nodes $\phi_i(\mathbf{x}_i), \psi_i(\mathbf{x}_i, \mathbf{x}_j)$

$$\min_{\boldsymbol{x}_{j}} \left[\psi_{i} \left(\boldsymbol{x}_{i}, \boldsymbol{x}_{j} \right) + \sum_{x \in \mathcal{X}_{f}} m_{x \to f}^{t} \left(\boldsymbol{x}_{i} \right) \right] = b_{i}^{t} \left(\boldsymbol{x}_{i} \right). \quad (23)$$

Remark 1 For a factor graph \mathcal{G} , if the local node ϕ_i optimization converges to unique solution x_i^* , then the node *i* is eliminated from the overall graph with the remaining graph $\mathcal{G}/|i|$ [20, proposition 1]. Proceeding in the same manner will result in the the optimized x^* .

Proposition 1 For a linear-structured factor graph G with compound factor nodes, if there exists a unique x_i^* which is the optimum point for node $\phi_i \in \mathcal{X}$ then for the global objective function (14), the min-sum message passing algorithm converges to a x^* together which optimizes entire graph.

Proof: In order for the full graph \mathcal{G} converging to a solution, the set of open nodes in the graph needs to hold $\mathcal{G}_{check} = \emptyset$, where \mathcal{G}_{check} is initialized by all nodes in the graph. Thus, given the linear-structured factor graph \mathcal{G} , where each node possesses self and egde-potential functions, the iterative optimization of the min-sum algorithm is run in a monotonically increasing or decreasing manner. Based on Remark 1, individual nodes are removed from \mathcal{G}_{check} in an ordered sequence, thus eventually obtaining $\mathcal{G}_{check} = \emptyset$ for for $t \to \infty$. Referring to Definition 2, the obtained graph directly allows to obtain the converged trajectory.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Horizon 2020 research and innovation programme under grant agreement 820742 of the project "HR-Recycler - Hybrid Human-Robot RECYcling plant for electriCal and eLEctRonic equipment".

REFERENCES

- L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robotics Autom.*, vol. 12, no. 4, pp. 566–580, 1996.
- [2] J. J. K. Jr. and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, San Francisco, CA, USA, April 24-28, 2000.* IEEE, 2000, pp. 995– 1001.
- [3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [4] N. D. Ratliff, M. Zucker, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: gradient optimization techniques for efficient motion planning," in 2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009. IEEE, 2009, pp. 489– 494.
- [5] M. Kalakrishnan, S. Chitta, E. A. Theodorou, P. Pastor, and S. Schaal, "STOMP: stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation, ICRA* 2011, Shanghai, China, 9-13 May 2011. IEEE, 2011, pp. 4569–4574.
- [6] C. Park, J. Pan, and D. Manocha, "ITOMP: incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012.* AAAI, 2012, pp. 207–215.
- [7] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression," in *Robotics: Science and Systems X, University of California, Berkeley,* USA, July 12-16, 2014, D. Fox, L. E. Kavraki, and H. Kurniawati, Eds., 2014.
- [8] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuoustime Gaussian process motion planning via probabilistic inference," *Int. J. Robotics Res.*, vol. 37, no. 11, pp. 1319–1340, 2018.
- [9] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th Annual International Conference* on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009, ser. ACM International Conference Proceeding Series, A. P. Danyluk, L. Bottou, and M. L. Littman, Eds., vol. 382. ACM, 2009, pp. 1049–1056.

- [10] M. Toussaint and C. Goerick, "A bayesian view on motor control and planning," in *From Motor Learning to Interaction Learning in Robots*, ser. Studies in Computational Intelligence, O. Sigaud and J. Peters, Eds. Springer, 2010, vol. 264, pp. 227–252.
- [11] M. Elbanhawi, M. Simic, and R. N. Jazar, "Randomized bidirectional b-spline parameterization motion planning," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 406–419, 2016.
- [12] Z. Marinho, B. Boots, A. D. Dragan, A. Byravan, G. J. Gordon, and S. S. Srinivasa, "Functional gradient motion planning in reproducing kernel hilbert spaces," in *Robotics: Science and Systems XII, University* of Michigan, Ann Arbor, Michigan, USA, June 18 - June 22, 2016, D. Hsu, N. M. Amato, S. Berman, and S. A. Jacobs, Eds., 2016.
- [13] F. Dellaert and M. Kaess, "Square root SAM: simultaneous localization and mapping via square root information smoothing," *Int. J. Robotics Res.*, vol. 25, no. 12, pp. 1181–1203, 2006.
- [14] E. Huang, M. Mukadam, Z. Liu, and B. Boots, "Motion planning with graph-based trajectories and Gaussian process inference," in 2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017. IEEE, 2017, pp. 5591–5598.
- [15] C. C. Moallemi and B. V. Roy, "Convergence of min-sum message passing for quadratic optimization," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2413–2423, 2009.
- [16] J. Yedidia, "Message-passing algorithms for inference and optimization," J. Stat. Phys., vol. 145, no. 4, pp. 860–890, 11 2011.
- [17] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, 1962.
- [18] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, 2001.
- [19] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2000, pp. 689–695.
- [20] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Tree consistency and bounds on the performance of the max-product algorithm and its generalizations," *Stat. Comput.*, vol. 14, no. 2, pp. 143–166, 2004.
- [21] N. Ruozzi and S. Tatikonda, "Message-passing algorithms: Reparameterizations and splittings," *IEEE Trans. Inf. Theory*, vol. 59, no. 9, pp. 5860–5881, 2013.