Distributed Particle Filter Based on Particle Exchanges

Rui Tang
CAS, Faculty of EEMCS, Mekelweg 4
Delft University of Technology (TUD)
2628 CD Delft, The Netherlands
R.Tang@tudelft.student.nl

Ellen Riemens
CAS, Faculty of EEMCS, Mekelweg 4
Delft University of Technology (TUD)
2628 CD Delft, The Netherlands
E.H.J.Riemens@tudelft.nl

Raj Thilak Rajan
CAS, Faculty of EEMCS, Mekelweg 4
Delft University of Technology (TUD)
2628 CD Delft, The Netherlands
R.T.Rajan@tudelft.nl

***Abstract*—We propose a novel distributed particle filter whose transmitted quantities are particles. The fusion process of particles is implemented in a distributed and iterative fashion. To reduce the communication overhead, we adopt the Gaussian process-enhanced resampling algorithm, which reduces the size of local particle set, while still ensures acceptable filtering performance. To determine the local particle set after the communication, we propose two solutions. Our first algorithm (GP-DPF) adopts a "scoring mechanism", allowing local agents score the received particles and using the scores as the selection criterion. Our second proposed solution (FA-DPF) is a meta-heuristic approach, which uses the well known firefly algorithm as a selection method for particle-based distributed particle filtering. Our simulations demonstrate the superiority of our proposed algorithms under the condition of limited communication and computational resources against other state-of-the-art distributed particle filters.**

TABLE OF CONTENTS

1. INTRODUCTION

In a multi-agent network, agents cooperate to solve a given problem, and offer solutions beyond the individual knowledge of each agent. In some applications, such as target tracking and environmental monitoring, we aim to estimate certain parameters (or states) of the surrounding environment given partial measurements covered with random disturbances [1], and this can be classified as a filtering problem in signal processing. The physical systems in many applications of multi-agent networks often consist of massive nonlinear and non-Gaussian elements. Particle filtering has been widely studied recently and is shown to outperform Kalman filtering [2]. Using a centralized approach for particle filtering lacks scalability, robustness, and flexibility. Distributed approaches are commonly proposed to overcome these limitations.

978-1-6654-9032-0/23/$31.00 ©2023 European Union

Recent work

To effectively implement DPF, approximation strategies must be utilized to reduce the inter-agent communication burden. Depending on the nature of the quantities transmitted across the network, we could divide DPF into weight-based, posterior-based, likelihood-based, and particle-based methods [2].

Weight-based algorithms aim to achieve consensus on the weight of each particle or the local likelihood value. The advantage of this type of DPF lies in its intuitive theoretical derivation and simple numerical computation. As a price, it has several limitations. First, the assumption that local random number generators are synchronized needs to hold to ensure an identical set of particles at each node. Second, the communication cost is proportional to the number of local particles. To reduce this cost, several approximation algorithms have been proposed. In [3], the combination of auxiliary particle filter and selective gossip is proposed to reduce communication cost. In [4], graph-based signal processing technique is used to approximate the particle weights by exploiting the inner structure of particle distribution.

Posterior-based algorithms use a parametric representation to describe the local posteriors and exchange sufficient statistics instead of particles to reduce communication overhead, but sacrifice some accuracy as a cost. Such DPFs generally require minimal communication resources. However, the design of the approximation algorithm is critical because the local posterior distribution can be very complex. In [5] [6], posteriors are approximated as Gaussian distributions. This is not suitable for the non-linear, non-Gaussian scenarios that the particle filter is used for. The Gaussian mixture model (GMM) is introduced to fit any kind of posterior distribution with an adjustable number of Gaussian components. In [7], the GMM is used to represent the posteriors. An adaptive Gaussian mixture learning algorithm for DPF is proposed in [8], aiming at adaptively choosing the number of Gaussian components in GMM for each agent. In [9], an importance sampling-based nonlinear fusion algorithm from the optimal perspective is proposed. These methods reduce the communication cost, but require computational power.

Likelihood-based algorithms aim at calculating the global likelihood function (GLF) in a distributed way. Here, local likelihood functions are approximated and then transmitted. In [10], the algorithms only work under the prerequisite that the local likelihood functions belong to the exponential family. In [11], the exponential family constraint is removed. The most significant benefit is that each agent can compute the global posterior independently when the GLF is available

locally. However, recent likelihood-based algorithms require the noise at all sensors to follow the same distribution.

As its name suggests, particle-based algorithms specify that the particles themselves are the data transmitted in the network. Little work has been done under this category, considering the high communication requirements. With the gradual maturity of research on particle filtering, more and more proposed resampling algorithms [12][13][14][15][16][17] reduce the need for a large number of particles. Even with a very limited number of particles, the diversity of particles can also be well maintained. The above research advance makes it possible to exchange particles directly when implementing DPF. The main advantage of this type of DPF is its great flexibility in modeling distributions and the elimination of the assumption of independent measurement noise of agents [2].

Overview

Our contribution is to propose a novel particle-based DPF, which reduces the size of the local particle set by selecting the most representative particles from the received set. In section 2, a target tracking problem is used as a motivation for particle filtering, followed by a concise overview of Particle filtering and Gaussian Process Regression. In section 3, the GP-enhanced resampling algorithm and a novel particle-based distributed particle filter are presented. In section 4, the firefly algorithm is used to optimize the information fusion stage in the previous section. In section 5, the performance of the proposed algorithms are compared to some state-of-the-art DPFs. In section 6, the numerical results are presented.

2. PRELIMINARIES

Consider the following state space model

$$\mathbf{x}_n = g(\mathbf{x}_{n-1}) + \mathbf{u}_n \quad (1)$$

$$\mathbf{y}_{n,k} = h_k(\mathbf{x}_n) + \mathbf{v}_{n,k} \quad k = 1, \cdots, K, \quad (2)$$

where the transition model (1) describes the evolution process of the state $\mathbf{x} \in \mathbb{R}^d$ over time, where d is the dimension of the state-space, $g(\cdot)$ is a possibly nonlinear mapping function and $\mathbf{u}_n \in \mathbb{R}^d$ is process noise at the nth time instant. The measurement model (2) explains the relationship between the target state $\mathbf{x}$ and the measurements $\mathbf{y}_{n,k} \in \mathbb{R}^b$ at sensor k at the nth time instant.

Let $\mathbf{y}_n \overset{\text{def}}{=} \{\mathbf{y}_{n,k}, k = 1, \cdots, K\}$ denote the collection of measurements taken at time instant n, and $\mathbf{y}_{1:n} \overset{\text{def}}{=} \{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$ denote the collection of measurements up to time instant n, then our objective is to sequentially estimate the state $\mathbf{x}_n$ based on measurements $\mathbf{y}_{1:n}$. Unless otherwise mentioned, we assume that the measurements taken at different sensors at each time instant n are conditionally independent given the state vector $\mathbf{x}_n$ [2], i.e., the global likelihood function can be factorized as

$$p(\mathbf{y}_n|\mathbf{x}_n) = \prod_{k=1}^{K} p(\mathbf{y}_{n,k}|\mathbf{x}_n). \quad (3)$$

Particle Filtering

The particle filter is often employed to solve filtering problems when the state-space model is non-linear and/or the noise is non-Gaussian. The posterior $p(\mathbf{x}_n|\mathbf{y}_{1:n})$ is approximated by a set of $\{\mathbf{x}_n^m, \omega_n^m\}_{m=1}^M$, where $\mathbf{x}_n^m$ are particles, and ω_n^m are the corresponding weights. We now describe the bootstrap particle filter to estimate the particles and particle weights, given the measurements $\mathbf{y}_{1:n}$, the likelihood $p(\mathbf{y}_n|\mathbf{x}_n)$ and the transitional prior distribution $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ [18].

Bootstrap particle filtering is defined as sequential importance sampling where the importance density $q(\cdot)$ is chosen as the state transition probability density function. Here, M particles $\mathbf{x}_n^m$ are drawn from $q(\cdot)$:

$$q(\mathbf{x}_n|\mathbf{x}_{0:1:n-1}^m, \mathbf{y}_{1:n}) = p(\mathbf{x}_n|\mathbf{x}_{n-1}), \quad (4)$$

and the weights corresponding to these particles are proportional to the likelihood function:

$$\omega(\mathbf{x}_n^m) \propto \omega(\mathbf{x}_{n-1}^m) p(\mathbf{y}_n|\mathbf{x}_n^m), \quad (5)$$

such that the weights are normalized i.e., $\sum_{m=1}^M \omega_n^m = 1$.

Later, we use function $\{\mathbf{x}_n^m, \omega_n^m\}_{m=1}^M = \mathbf{PF}(\{\mathbf{x}_{n-1}^m, \omega_{n-1}^m\}_{m=1}^M, \mathbf{y}_n)$ to denote the particle filtering part, where the input is M weighted particles at time instant $n-1$ as well as measurements $\mathbf{y}_n$ and the output is the updated M weighted particles describing the current posterior distribution.

Gaussian Process

Gaussian process is a powerful non-parametric Bayesian method [19], which could be used to approximate the local posterior distributions in resampling stage. Unlike other parametric regressors, GP has no assumptions on the underlying structure(e.g. linear, quadratic) of the system model. Given the input data set $\mathbf{X} \overset{\text{def}}{=} [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^M]$, where $\mathbf{x}^m$ corresponds to the mth input, and the output $\mathbf{y} \overset{\text{def}}{=} [y^1, y^2, \cdots, y^M]$, where,

$$y^m = f(\mathbf{x}^m) + \eta^m \qquad m = 1, \cdots, M, \quad (6)$$

and $\eta^m \sim \mathcal{N}(0, \sigma_\eta^2)$ is the noise. In GPR, the signal term $f(\mathbf{x})$ is assumed to be GP distributed and allows us to make predictions on new inputs. By definition, the prediction distribution follows a joint multivariate normal distribution which can be written as

$$f(\mathbf{x})|\mathbf{y}, \mathbf{X}, \mathbf{x} \sim \mathcal{N}(\mu(\mathbf{x}), \Sigma(\mathbf{x})), \quad (7)$$

where

$$\mu(\mathbf{x}) = \mathcal{K}(\mathbf{x}, \mathbf{X})(\mathcal{K}(\mathbf{X}, \mathbf{X}) + \sigma_\eta^2 \mathbf{I})^{-1}\mathbf{y}, \quad (8)$$

$$\Sigma(\mathbf{x}) = \mathcal{K}(\mathbf{x}, \mathbf{x}) - \mathcal{K}(\mathbf{x}, \mathbf{X})(\mathcal{K}(\mathbf{X}, \mathbf{X}) + \sigma_\eta^2 \mathbf{I})^{-1}\mathcal{K}(\mathbf{X}, \mathbf{x}). \quad (9)$$

Here, $\mathcal{K}$ is the kernel of GP, which is chosen based on the prior knowledge on the given data set such as smoothness.

GP-enhanced resampling

In [13], Gaussian process is used to model local posteriors, which is illustrated in Algorithm 1, where the pairs of particles and weights $\{\mathbf{x}^m, \omega^m\}_{m=1}^M$ are the inputs. The mapping function, i.e., local posterior distribution, can be approximated by a Gaussian Process (GP) based on (8)-(9), where the Gaussian kernel is given by

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp(\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{x}'||^2), \quad (10)$$

which is positive and smooth. Let the approximated GP representing the posterior be denoted by $\mathcal{GP}1$. Since the estimation results are particle weights, an additional normalization step has to be performed afterwards. To exploit the uncertainty knowledge provided by $\mathcal{GP}1$ and enable the exploration, we artificially build a second particle set using the function $f(\mathbf{x}) = \mu(\mathbf{x}) + 3\Sigma^{1/2}(\mathbf{x})$. The input for the function is formed by generating $M-1$ intermediary particles from the existing ones, so the second particle set can be denoted as $\{\hat{\mathbf{x}}^m, f(\hat{\mathbf{x}}^m)\}_{m=1}^{M-1}$. This set is then approximated using GPR again. By doing so, the particles with high variance can also be taken into consideration. To decide whether to sample from $\mathcal{GP}1$ or $\mathcal{GP}2$, we define a parameter ζ, and generate a scalar u from a uniform distribution between 0 and 1. If $u \geq \zeta$, we sample from $\mathcal{GP}1$, otherwise from $\mathcal{GP}2$. By tuning the parameter ζ, we can decide whether the final resampling results are more inclined to exploitation or exploration [2]. The final weights after resampling would be $1/M$ uniformly for all weights. It has been verified that the adoption of Gaussian process-enhanced resampling algorithm can guarantee the tracking performance with a limited number of particles under the constant velocity (CV) model [13].

Algorithm 1 Gaussian process enhanced resampling (GP-R)

Require: $\{\mathbf{x}_n^m, \omega_n^m\}_{m=1}^M, \zeta \in [0,1]$
Initialize GP hyperparameter σ
Fit a GP $\mathcal{GP}1$ using particle set $\{\mathbf{x}_n^m, \omega_{n-1}^m\}_{m=1}^M$
for $m = 1, \cdots, M-1$ **do**
  $\hat{\mathbf{x}}_n^m = (\mathbf{x}_n^{m+1} - \mathbf{x}_n^m)/2 + \mathbf{x}_n^m$
  $\{\mu(\hat{\mathbf{x}}_n^m), \Sigma(\hat{\mathbf{x}}_n^m)\} = \mathcal{GP}1(\hat{\mathbf{x}}_n^m)$
  Compute $f(\hat{\mathbf{x}}_n^m) = \mu(\hat{\mathbf{x}}_n^m) + 3\Sigma^{1/2}(\hat{\mathbf{x}}_n^m)$
end for
Fit a GP $\mathcal{GP}2$ using particle set $\{\hat{\mathbf{x}}_n^m, f(\hat{\mathbf{x}}_n^m)\}_{m=1}^{M-1}$
for $m = 1, \cdots, M$ **do**
  Draw $u \sim \mathcal{U}_{[0,1]}$
  if $u \geq \zeta$ **then**
    Draw $\tilde{\mathbf{x}}_n^m$ from $\mathcal{GP}1$
  else
    Draw $\tilde{\mathbf{x}}_n^m$ from $\mathcal{GP}2$
  end if
  Assign weights: $\tilde{\omega}_n^m = 1/M$
end for
return $\{\tilde{\mathbf{x}}_n^m, \tilde{\omega}_n^m\}_{m=1}^M$

3. GAUSSIAN PROCESS BASED PARTICLE FILTER

In this section, a novel particle-based Distributed Particle Filter i.e., Gaussian-Process based DPF (GP-DPF), which is based on the GP-R discussed in the earlier section.

When exchanging particles, the goal is to merge particle sets between neighboring nodes. We design effective algorithms to filter out those most representative particles from the received ones. In this section, we present a method where each agent score each received particle based on its local measurements. We define the score as the likelihood value, e.g., the score of the mth received particle at sensor k at time n is calculated as follows:

$$s_{n,k}^m = p(\mathbf{y}_{n,k}|\mathbf{x}_n^m). \tag{11}$$

[2]In optimization, exploitation means to focus the search on a local space where a current good solution exists while exploration means to explore the search space on a global scale.

In order to achieve the purpose of information fusion, we need to make sure the particle set of each agent is fully mixed. If multiple communication iterations are performed between consecutive time instants, each agent can gradually gain particles from the entire network.

Next, we briefly describe the algorithm presented in Algorithm 2. Each agent first starts with local particle filtering and GP-enhanced resampling, resulting in a set of weighted particles. In each communication iteration, each sensor broadcasts a subset of its local particles, whose size is determined as $\phi M, \phi \in (0,1]$, to its neighbors. The agents then calculate the scores of the received particles and select particles with higher scores. In order to compose the final particle set, $\lceil M/(D_k+1) \rceil$ particles are from the local particle set. The remaining particles are from the particles received from connected agents and are determined in resampling stage to increase the diversity of the particles. D_k is the number of adjacent nodes of agent k and $\lceil \cdot \rceil$ is the ceiling operation. Information will be diffused across the network, integrating the knowledge of all measurements across the agents. Parameters such as ϕ, local particle size M, and the number of iterations L_b can be adjusted to balance between tracking performance and communication overhead.

Algorithm 2 GP-DPF

Require: $\{\mathbf{x}_{n-1,k}^m, \omega_{n-1,k}^m\}_{m=1}^M, \mathbf{y}_{n,k}, \phi \in (0,1]$
Initialize the number of iterations L_b, degree D_k
$\{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1}^M = \textbf{PF}(\{\mathbf{x}_{n-1,k}^m, \omega_{n-1,k}^m\}_{m=1}^M, \mathbf{y}_{n,k})$
$\{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1}^M = \textbf{GP-R}(\{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1}^M)$
for $i = 1, \cdots, L_b$ **do**
  Randomly draw ϕM particles from $\{\mathbf{x}_{n,k}^m\}_{m=1}^M$
  Sensor k sends $\{\mathbf{x}_{n,k}^m\}_{m=1}^{\phi M}$ to its neighbors
  Compute scores $\{s_{n,k}^m\}_{m=1}^{\phi M D_k}$ of $\cup_{j\in\mathcal{N}_k}\{\mathbf{x}_{n,j}^m\}_{m=1}^{\phi M}$
  Randomly draw $\lceil \frac{M}{D_k+1} \rceil$ particles from $\{\mathbf{x}_{n,k}^m\}_{m=1}^{\phi M}$
  $\{\mathbf{x}_{n,k}^m\}_{m=\lceil \frac{M}{D_k+1} \rceil+1}^{M}$
    $= \textbf{Resample}(\cup_{j\in\mathcal{N}_k}\{\mathbf{x}_{n,j}^m\}_{m=1}^{\phi M}, \{s_{n,k}^m\}_{m=1}^{\phi M D_k})$
  Combine two subsets of particles and get $\{\mathbf{x}_{n,k}^m\}_{m=1}^M$
end for
$\mathbf{x}_{n,k} = \text{mean}(\{\mathbf{x}_{n,k}^m\}_{m=1}^M)$
return $\mathbf{x}_{n,k}, \{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1}^M$
† The whole algorithm is executed at all nodes in parallel $\forall k = 1, \cdots, K$

4. META-HEURISTIC BASED PARTICLE FILTER

In the previous section, a scoring mechanism was proposed as a method to select representative particles from the received ones. However, the limitation is that only particles from the existing ones can be chosen and the results may deviate when the size of particle set is relatively small. In this section, we regard the particle selection process as an optimization problem, i.e., continuously optimizing the local particle set. Various optimization algorithms can be utilized to achieve the above goal. In this paper, we use modern metaheuristics to search for a globally optimal particle set. The Firefly algorithm (FA) [20] is adopted due to its superiority over genetic algorithms and particle swarm optimization.

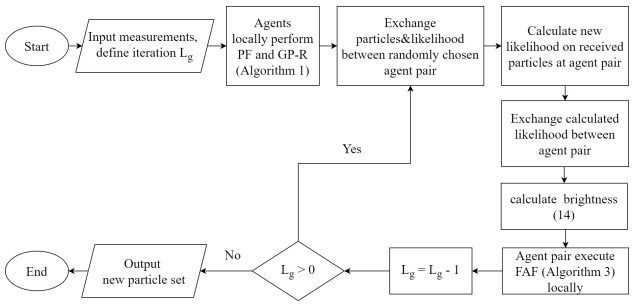


Figure 1: **The flowchart of FA-DPF (the whole block is for a single time instant n).**

Firefly Algorithm

The Firefly Algorithm (FA) mimics the social behavior of fireflies flying through the sky and is based on the following principles.

Attractiveness—The brightness of a firefly at a certain location $\mathbf{x}$ is determined by the objective function, i.e., $I(\mathbf{x}) \propto f(\mathbf{x})$. The attractiveness β is relative and it should be judged by other fireflies. The attractiveness of the brighter of the two fireflies, l and j, to the other should vary with the distance r_{lj} between the two. This relationship is usually described by the following equation:

$$\beta(r_{lj}) = \beta_0/(1+\gamma r_{lj}^2), \tag{12}$$

where β_0 is the attractiveness when $r_{lj} = 0$ and γ is called light absorption coefficient, which determines the strength of the influence of distance on attractiveness. The distance can be defined using the Cartesian distance $r_{lj} = ||\mathbf{x}_l - \mathbf{x}_j||_2$.

Movement—The movement of a firefly l towards a brighter firefly j can be modeled as

$$\mathbf{x}_l = \mathbf{x}_l + \beta_0 e^{-\gamma r_{lj}^2}(\mathbf{x}_j - \mathbf{x}_l) + \alpha\varepsilon, \tag{13}$$

where the second term is due to the attractiveness and the third term is a random term. The use of random term gives FA the ability to explore the search space. Therefore, it is possible to achieve a good balance between local intensive exploitation and global exploration by adjusting $\alpha \in [0,1]$.

Firefly Algorithm Improved Particle-based DPF(FA-DPF)

In this section, we employ the Firefly Algorighm as a selection method for particle-based distributed particle filtering. The pseudo-code of FA-DPF is presented in Algorithm 4. We assume the information of two agents is fused at a time. Hence, the gossip communication protocol is adopted. Agents l and j are selected at a single communication iteration. We define the brightness of each particle as a function of the likelihood value, given measurement information of both agents. The flowchart of the algorithm is depicted in Figure 1, which would be explained in the following. Note, we only describe the behavior of agent l, and the same for agent j.

Each agent performs local particle filtering and GP-enhanced resampling, resulting in a set of weighted particles. The data at agent l should be $\{\mathbf{x}_{n,l}^m, \omega_{n,ll}^m\}_{m=1}^M$, where the first l subscript relates to the particle origin and the second to the likelihood based on agent l's measurements. When the importance density is chosen to be the same as transition function as discussed in section 2, the particle weight is equal to the likelihood value. After the communication round, agent l would hold the data $\{\mathbf{x}_{n,l}^m, \omega_{n,ll}^m\}_{m=1}^M$ and $\{\mathbf{x}_{n,j}^m, \omega_{n,jj}^m\}_{m=1}^M$ from agent j.

The particle brightness function is given in (14). Now we do not only need the likelihood based on measurements at l: ω_{ll}^m, we also require ω_{lj}^m.

$$I(\mathbf{x}_{n,l}^m) = \omega_{ll}^m \omega_{lj}^m. \tag{14}$$

To obtain this value, we introduce another communication round. Agent l now receives $\{\omega_{n,lj}^m\}_{m=1}^M$ from agent j. This completes the communication and the data held by agent l is $\{\mathbf{x}_{n,l}^m, \omega_{n,ll}^m, \omega_{n,lj}^m, \mathbf{x}_{n,j}^m, \omega_{n,jj}^m, \omega_{n,jl}^m\}_{m=1}^M$.

After determining the corresponding brightness, i.e, $\{I(\mathbf{x}_{n,l}^m), I(\mathbf{x}_{n,j}^m)\}_{m=1}^M$, we can proceed to the firefly algorithm based fusion (FAF). First, for each particle drawn from the set $\{\mathbf{x}_{n,l}^m\}_{m=1}^M$, taking $\mathbf{x}_{n,l}^p$ as an example, we make a comparison between $I(\mathbf{x}_{n,l}^p)$ with $\max(\{I(\mathbf{x}_{n,j}^m)\}_{m=1}^M)$. If $I(\mathbf{x}_{n,l}^p) < \max(\{I(\mathbf{x}_{n,j}^m)\}_{m=1}^M)$, we randomly draw a particle $\mathbf{x}_{n,j}^q$ from the set $\{\mathbf{x}_{n,j}^m\}_{m=1}^M$ meeting the requirement that $I(\mathbf{x}_{n,j}^q) > I(\mathbf{x}_{n,l}^p)$. The above operations can be performed efficiently by introducing a sorting algorithm. Then we move the particle $\mathbf{x}_{n,l}^p$ towards $\mathbf{x}_{n,j}^q$ based on the following movement rule which is rephrased from (13):

$$\mathbf{x}_{n,l}^p = \mathbf{x}_{n,l}^p + \beta_0/(1+\gamma r_{pq}^2)(\mathbf{x}_{n,j}^q - \mathbf{x}_{n,l}^p) + \alpha\varepsilon, \tag{15}$$

where $r_{pq} = ||\mathbf{x}_p - \mathbf{x}_q||_2$ and ε follows the same distribution as the process noise. The above procedure needs to be repeated M times until each particle in particle set $\{\mathbf{x}_{n,l}^m\}_{m=1}^M$ has been processed. Multiple iterations can be executed to ensure information from the entire network is fully fused. The pseudo-code for FAF is presented in Algorithm 3.

Algorithm 3 Firefly Algorithm-based Fusion(FAF)

Require: $\{\mathbf{x}_{n,l}^m, I(\mathbf{x}_{n,l}^m)\}_{m=1}^M, \{\mathbf{x}_{n,j}^m, I(\mathbf{x}_{n,j}^m)\}_{m=1}^M$
Initialize Firefly algorithm hyperparameters β_0, α, γ
for $p = 1, \cdots, M$ **do**
  if $\max(\{I(\mathbf{x}_{n,j}^m)\}_{m=1}^M) > I(\mathbf{x}_{n,l}^p)$ **then**
    Randomly draw $\mathbf{x}_{n,j}^q$ meeting $I(\mathbf{x}_{n,j}^q) > I(\mathbf{x}_{n,l}^p)$
    $r_{pq} = ||\mathbf{x}_{n,l}^p - \mathbf{x}_{n,j}^q||_2$
    $\mathbf{x}_{n,l}^p = \mathbf{x}_{n,l}^p + \beta_0/(1+\gamma r_{pq}^2)(\mathbf{x}_{n,j}^q - \mathbf{x}_{n,l}^p) + \alpha\varepsilon$
  end if
end for
return $\{\mathbf{x}_{n,l}^m\}_{m=1}^M$
† ε is generated from the distribution same as the process noise.

Algorithm 4 FA-DPF

Require: $\{\mathbf{x}_{n-1,k}^m, \omega_{n-1,k}^m\}_{m=1,k=1}^{M,K}, \{\mathbf{y}_{n,k}\}_{k=1}^K$

Initialize the number of gossip iterations L_g

Execute the following tasks $\quad \forall k = 1, \cdots, K$

$$\begin{cases} \{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1}^M = \textbf{PF}(\{\mathbf{x}_{n-1,k}^m, \omega_{n-1,k}^m\}_{m=1}^M, \mathbf{y}_{n,k}) \\ \{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1}^M = \textbf{GP-R}(\{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1}^M) \\ \text{Calculate the particles' weight } \{\omega_{n,kk}^m\}_{m=1}^M \end{cases}$$

for $i = 1, \cdots, L_g$ **do**

  Randomly select 2 adjacent sensors l, j

  Exchange particles and weights

  Run parallel at sensors l, j:

$$\begin{cases} \text{Calculate weight } \omega_{n,lj}^m \text{ of set } \{\mathbf{x}_{n,l}^m\}_{m=1}^M \\ \text{Calculate weight } \omega_{n,jl}^m \text{ of set } \{\mathbf{x}_{n,j}^m\}_{m=1}^M \end{cases}$$

  Exchange weights $\{\omega_{n,jl}^m\}_{m=1}^M, \{\omega_{n,lj}^m\}_{m=1}^M$

  Calculate particles' brightness at sensors l, j

  Run **FAF** at sensors l, j

end for

$\mathbf{x}_{n,k} = \text{mean}(\{\mathbf{x}_{n,k}^m\}_{m=1}^M) \qquad \forall k = 1, \cdots, K$

return $\{\mathbf{x}_{n,k}\}_{k=1}^K, \{\mathbf{x}_{n,k}^m, \omega_{n,k}^m\}_{m=1,k=1}^{M,K}$

5. PERFORMANCE ANALYSIS

In this section we present the performance of the proposed algorithms in terms of communication overhead and computational complexity, as shown in Table 1. We include GL-DPF [4] and GM-DPF [9] for comparison as well, but the derivation is omitted.

Table 1: **Analysis of Communication and Computation Complexity. (L_1: the number of randomized gossip iterations in GL-DPF; K: the number of agents; z: the number of reserved Laplacian transfer coefficients; M: the size of local particle set; N_{knn}: the number of nearest neighbors in k-NN; d: the dimension of the state space; L_2: the number of communication iterations in GM-DPF; C:the number of components in GMM; L_{EM}: the number of EM iterations; L_f: the number of iterations for fusion stage; L_{ft}: the number of iterations for fine-tuning; L_b: the number of communication iterations in GP-DPF; ϕ: a scalar between (0,1]; L_g: the number of communication iterations in FA-DPF)**

Algorithm	Communication	Computation
GL-DPF	$\mathcal{O}(2L_1 z/K)$	$\mathcal{O}(M^3 + N_{knn}Md)$
GM-DPF	$\mathcal{O}(L_2 C d^2) - \mathcal{O}(KL_2 C d^2)$	$\mathcal{O}([(L_{EM}+K)L_f M + M + KL_{ft}]Cd^2)$
GP-DPF	$\mathcal{O}(2\phi dML_b K)$	$\mathcal{O}(M^3 + (M-1)^3)$
FA-DPF	$\mathcal{O}(L_g M(2d+4)/K)$	$\mathcal{O}(2L_g M log(M) d/K + M^3 + (M-1)^3)$

Communication Overhead

GP-DPF—Particles are transmitted across the network. Let L_b denote the number of broadcast iterations. There are $2|E|\phi dM$ scalars to be transmitted in every iteration where $|E|$ is the number of links in the multi-agent network. Since $|E|$ ranges from $\mathcal{O}(K)$ to $\mathcal{O}(K^2)$ for a connected network, the worst case communication complexity per agent during each time step is $\mathcal{O}(2\phi dML_b K)$.

FA-DPF—In addition to exchanging particles, the likelihood values also need to be exchanged for FA-DPF. Hence, in each iteration, a total of $2M$ particles as well as $4M$ likelihood values need to be exchanged. Let L_g denote the number of iterations, the average communication complexity per agent is $\mathcal{O}(L_g M(2d+4)/K)$.

Computational Complexity

GP-DPF—Exact GP inference is quite computationally demanding, requiring $\mathcal{O}(M^3)$ time complexity and $\mathcal{O}(M^2)$ space complexity [21], which prevents us from using more particles. This is also the bottleneck of GP-enhanced resampling. Finally, since GPR is performed twice in GP-DPF, the computation complexity is approximately $\mathcal{O}(M^3 + (M-1)^3)$. Note: in computing the computational complexity, we assume hyper-parameters training for GP is excluded.

FA-DPF— In the FA-DPF, a sorting algorithm, requiring $\mathcal{O}(Mlog(M)d)$ [22], is added. Hence, under the gossip protocol, the average complexity per agent is $\mathcal{O}(2L_g Mlog(M)d/K + M^3 + (M-1)^3)$.

6. SIMULATION

In this section, we present the performance evaluation of our proposed algorithms compared to other state-of-the-art DPFs through simulation.

Simulation Scenario and Setup

We consider a target tracking problem where a single agent moves following the Wiener process acceleration model [23] and 9 sensor nodes are deployed in an area of 200×200. The state vector (16) of the target contains 6 elements, including the position, velocity and acceleration in two-dimensional space.

$$\mathbf{x}_n = [x_{n,1} \quad x_{n,2} \quad \dot{x}_{n,1} \quad \dot{x}_{n,2} \quad \ddot{x}_{n,1} \quad \ddot{x}_{n,2}]^T \tag{16}$$

The transition function is defined as follows,

$$\mathbf{g}(\mathbf{x}_n) = \mathbf{D}\mathbf{x}_n + \mathbf{u}_n, \tag{17}$$

where

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & t & 0 & \frac{1}{2}t^2 & 0 \\ 0 & 1 & 0 & t & 0 & \frac{1}{2}t^2 \\ 0 & 0 & 1 & 0 & t & 0 \\ 0 & 0 & 0 & 1 & 0 & t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{18}$$

t is the state transition interval, which is set to 1 in our case. $\mathbf{u}_n$ follows a multivariate Gaussian distribution $\mathcal{N}(0, \mathbf{R})$, where

$$\mathbf{R} = \sigma_u^2 \begin{bmatrix} \frac{1}{20}t^5 & 0 & \frac{1}{8}t^4 & 0 & \frac{1}{6}t^3 & 0 \\ 0 & \frac{1}{20}t^5 & 0 & \frac{1}{8}t^4 & 0 & \frac{1}{6}t^3 \\ \frac{1}{8}t^4 & 0 & \frac{1}{3}t^3 & 0 & \frac{1}{2}t^2 & 0 \\ 0 & \frac{1}{8}t^4 & 0 & \frac{1}{3}t^3 & 0 & \frac{1}{2}t^2 \\ \frac{1}{6}t^3 & 0 & \frac{1}{2}t^2 & 0 & t & 0 \\ 0 & \frac{1}{6}t^3 & 0 & \frac{1}{2}t^2 & 0 & t \end{bmatrix}. \tag{19}$$

To estimate the unknown time-varying state vector of the moving target, we deploy 9 agents at known positions to

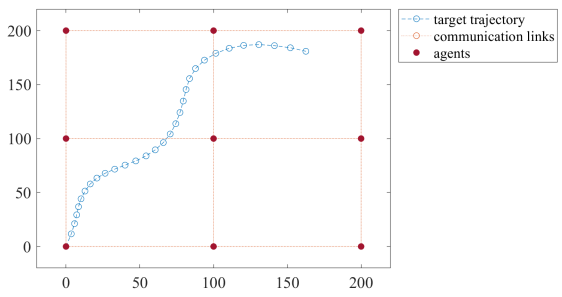


Figure 2: **Multi-agent network, communication links and target trajectory(We here have 9 agents with known positions, and our goal is estimate the 6D state vector of the unknown agent. The dotted blue line is one unknown trajectory realization.**

perform the measurement tasks, i.e., the target's range and Doppler (range rate). We denote the position of the kth sensor as $\mathbf{l}_k = (l_{k,1}, l_{k,2})$, then the measurement vector can be expressed as

$$\mathbf{h}_k(\mathbf{x}_n) = [h_{k,range}(\mathbf{x}_n) \quad h_{k,doppler}(\mathbf{x}_n)]^T, \tag{20}$$

where the two elements are specified as follows,

$$h_{k,range}(\mathbf{x}_n) = \sqrt{(x_{n,1} - l_{k,1})^2 + (x_{n,2} - l_{k,2})^2}, \tag{21}$$

$$h_{k,doppler}(\mathbf{x}_n) = \frac{\dot{x}_{n,1}(x_{n,1} - l_{k,1}) + \dot{x}_{n,2}(x_{n,2} - l_{k,2})}{\sqrt{(x_{n,1} - l_{k,1})^2 + (x_{n,2} - l_{k,2})^2}}, \tag{22}$$

and the measurement noise $\mathbf{v}_{n,k}$ follows $\mathcal{N}(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\sigma_v^2 & 0\\0 & \sigma_w^2\end{bmatrix})$. We set σ_u to 0.5, σ_v to 1, and σ_w to 1. The initial target state is set to $[0, 0, 4, 13, -1, -3]^T$. Figure 2 shows the sensor network and a realization of the target trajectory. We set α to 1, γ to 0.03, and β_0 to 0.5 for FAF.

Communication Overhead

In particle-based DPF, since the communication overhead is determined by both the particle set size and the number of communication iterations, we change the communication overhead by fixing the number of particles and varying the number of communication iterations. The number of particles for each proposed algorithm is chosen to be the elbow point of the function describing the relationship between tracking error and particle set size under the condition that the number of iterations is sufficiently large, i.e. $M = 50$ for GP-DPF, and $M = 10$ for FA-DPF. It also shows when firefly algorithm is used, the number of local particles can be further reduced as long as there are enough iterations. For GL-DPF and GM-DPF, we fix the number of particles to $M = 2000$. For GL-DPF, we keep $z = 500$ Laplacian transform coefficients.

In Figure 3, we show the trajectory error ARMSE for each DPF algorithm as a function of the count of scalars transmitted between sensors in the network. We plot the tracking error of centralized particle filter where all measurements are gathered in the fusion center as a benchmark, as shown by the flat dotted line. As expected, there is a trade-off between estimation accuracy and communication cost. From the figure, we can see that GM-DPF has the smallest communication overhead while GL-DPF requires much higher communication resources. Our proposed GP-DPF and FA-DPF show good estimation performance under limited communication resources. Although the communicated scalars cannot be compressed to the level of GM-DPF, their demands for communication resources are far less than that of GL-DPF. In addition, The FA-DPF can achieve faster convergence speed and lower estimation error.

Time Complexity

Time complexity is also an important metric for algorithms when it comes to practical implementation. The hyper-parameters in each algorithm are mostly problem specific, making the theoretical comparison challenging. We plot the running time required for different algorithms at a given tracking error, i.e., ARMSE around 2, under 50 Monte Carlo experiments, as shown in Figure 4,

From the figure, we can see that the time complexity of the GM-DPF is far larger than that of other DPFs, and hence, it is difficult to apply with high real-time requirements. In contrast, the complexity of the GL-DPF algorithm is relatively lower but still higher than the proposed algorithms in this paper, showing our proposed algorithms more appealing to practical use.

7. CONCLUSIONS

We have proposed a novel particle-based distributed particle filter in this paper. In section 2, two key points of particle-based DPF are stated. The first one is to resort to efficient resampling algorithms to reduce the size of local particle set and therefore solve the problem of high communication overhead. In this paper, GP-enhanced resampling is adopted. The second one is to design algorithms determining which particles to be kept after particle exchange to improve estimation performance. We have given two solutions in this paper. First, a "scoring mechanism" is proposed. Second, one of the most well-known metaheuristics, firefly algorithm, is used to evolve the local particle set when new information is acquired. Numerical results shows the potential of our proposed algorithms compared with some state-of-the-art DPF in terms of both communication overhead and time complexity. In the future, more combinations of efficient resampling algorithms and metaheuristics can be explored.

REFERENCES

[1] F. Zhao, L. J. Guibas, and L. Guibas, *Wireless sensor networks: an information processing approach*. Morgan Kaufmann, 2004.

[2] O. Hlinka, F. Hlawatsch, and P. M. Djuric, "Distributed particle filtering in agent networks: A survey, classification, and comparison," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 61–81, 2012.

[3] D. Üstebay, M. Coates, and M. Rabbat, "Distributed auxiliary particle filters using selective gossip," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 3296–3299.

[4] M. Rabbat, M. Coates, and S. Blouin, "Graph laplacian distributed particle filtering," in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1493–1497.

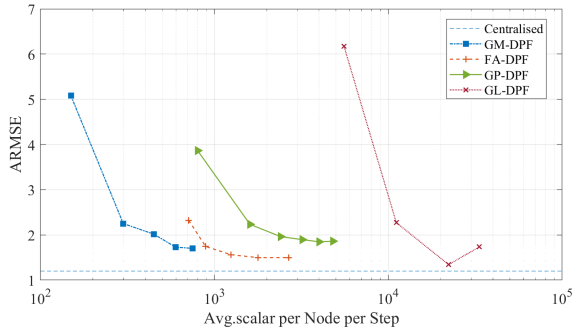


Figure 3: **Tracking performance ARMSE as a function of the communication cost per time step (x-axis in log scale).**

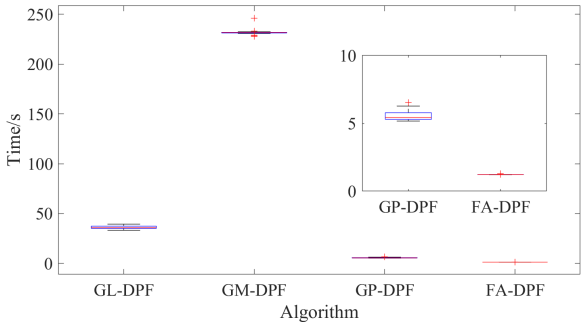


Figure 4: **Time complexity comparison among various DPFs.**

[5] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in *2008 International Conference on Information and Automation*. IEEE, 2008, pp. 302–307.

[6] A. Mohammadi and A. Asif, "Distributed particle filter implementation with intermittent/irregular consensus convergence," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2572–2587, 2013.

[7] D. Gu, "Distributed em algorithm for gaussian mixtures in sensor networks," *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1154–1166, 2008.

[8] J. Li and A. Nehorai, "Adaptive gaussian mixture learning in distributed particle filtering," in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2015, pp. 221–224.

[9] Li, Jichuan and Nehorai, Arye, "Distributed particle filtering via optimal fusion of gaussian mixtures," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 2, pp. 280–292, 2017.

[10] O. Hlinka, O. Slučiak, F. Hlawatsch, P. M. Djurić, and M. Rupp, "Likelihood consensus and its application to distributed particle filtering," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4334–4349, 2012.

[11] O. Hlinka, F. Hlawatsch, and P. M. Djurić, "Likelihood consensus-based distributed particle filtering with distributed proposal density adaptation," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 3869–3872.

[12] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: classification, implementation, and strategies," *IEEE Signal processing magazine*, vol. 32, no. 3, pp. 70–86, 2015.

[13] T. Imbiriba and P. Closas, "Enhancing particle filtering using gaussian processes," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. IEEE, 2020, pp. 1–7.

[14] S. Park, J. P. Hwang, E. Kim, and H.-J. Kang, "A new evolutionary particle filter for the prevention of sample impoverishment," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 801–809, 2009.

[15] S. S. Moghaddasi and N. Faraji, "A hybrid algorithm based on particle filter and genetic algorithm for target tracking," *Expert Systems with Applications*, vol. 147, p. 113188, 2020.

[16] N. Zhou, L. Lau, R. Bai, and T. Moore, "A genetic optimization resampling based particle filtering algorithm for indoor target tracking," *Remote Sensing*, vol. 13, no. 1, p. 132, 2021.

[17] M.-L. Gao, L.-L. Li, X.-M. Sun, L.-J. Yin, H.-T. Li, and D.-S. Luo, "Firefly algorithm (fa) based particle filter method for visual tracking," *Optik*, vol. 126, no. 18, pp. 1705–1711, 2015.

[18] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE proceedings F (radar and signal processing)*, vol. 140, no. 2. IET, 1993, pp. 107–113.

[19] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[20] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*. Springer, 2009, pp. 169–178.

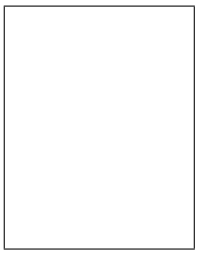
[21] D.-T. Nguyen, M. Filippone, and P. Michiardi, "Exact gaussian process regression with distributed computations," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 1286–1295.

[22] X.-S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International journal of swarm intelligence*, vol. 1, no. 1, pp. 36–50, 2013.

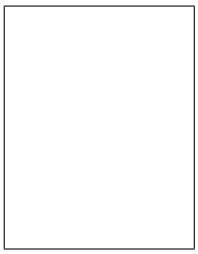
[23] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and data fusion*. YBS publishing Storrs, CT, USA:, 2011, vol. 11.

BIOGRAPHY

Rui Tang received his B.Sc. degree in Beijing Jiaotong University, Beijing, China, in 2016 and the M.Sc. degree in Delft University of Technology, the Netherlands, in 2022. He is currently a wireless communication engineer at Huawei Technologies Co., Ltd. His research interests include distributed optimisation, Bayesian inference, and wireless communication.



Ellen Riemens is a PhD. candidate with the Faculty of electrical engineering, mathematics and computer science (EEMCS) at the Delft university of technology (TUD). She received her B.Sc. and M.Sc. degree at Delft University of Technology, in 2019 and 2021. Her research interests are distributed signal processing and Bayesian inference, with applications to Multi-agent systems.



Dr. Raj Thilak Rajan (S'11, M'17, SM'22) is an Assistant Professor on Distributed and Autonomous Sensor Systems, with the Faculty of electrical engineering, mathematics and computer science (EEMCS) at the Delft university of technology (TUD), and is the Co-director of the Delft Sensor AI Lab. His research interests lie in statistical machine learning and distributed optimisation, with applications to autonomous sensor systems e.g., PNT of swarms.