Tensor-Train-based Algorithms for Aggregate State Estimation of Swarms with Interacting Agents

Citation: Miculescu, David and Karaman, Sertac. 2020. "Tensor-Train-based Algorithms for Aggregate State Estimation of Swarms with Interacting Agents." Proceedings of the American Control Conference, 2020-July.

As Published: 10.23919/acc45564.2020.9147339

Publisher: IEEE

Persistent URL: https://hdl.handle.net/1721.1/137298

Version: Author’s final manuscript: final author’s manuscript post peer review, without publisher’s formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike
Tensor-Train-based Algorithms for Aggregate State Estimation of Swarms with Interacting Agents*

David Miculescu\(^1\) and Sertac Karaman\(^2\)

Abstract—In this paper, we develop an efficient implementation of the gas-kinetic (GK) Probability Hypothesis Density (PHD) filter for aggregate swarm state estimation with interacting agents. We borrow a kinetic/mesoscopic partial differential equation (PDE) model of a swarm of interacting agents from biology moving in a plane with a heading state, which requires the computation of integrals up to five dimensions. In the context of the GK-PHD, we propagate this model by computing in a compressed format called the Tensor Train (TT) format, yielding better memory and runtime properties than a grid-based approach. Under certain assumptions, we prove that TT-GK-PHD has a time complexity of an order of magnitude better than the grid-based approach. Finally, we prove that TT-GK-PHD has a time complexity of an order of magnitude better than the grid-based approach. Under certain assumptions, we show that under certain assumptions, the runtime of TT-GK-PHD is \(O(n^4)\) while the runtime of a grid-based solution similar to the one in [14] is \(O(n^5)\), where \(n\) is the grid resolution of the state space. Third, we validate our analysis in computational experiments and demonstrate the usefulness of our implementation in a scenario which the grid-based solution fails due to hardware limitations.

I. INTRODUCTION

In recent years, large swarms of unmanned aerial vehicles (UAVs) have been designed and developed. In October 2016, 103 Perdix drones were dropped from US fighter jets in a test exercise [1]. Autonomous flying microrobots called RoboBees have been developed at Harvard’s Wyss Institute [2]. Scenarios in which large numbers of these agents swarm together are quickly becoming a reality, and efficient state estimation algorithms are needed to monitor these swarms.

In this work, we consider propagating a swarm density given a dynamics model and periodic measurements of individual agents. Much of the literature on this problem can be divided into two camps. One approach seeks to describe the swarm as a joint density over all the agents in the swarm [3]–[5]. Though this method is useful for general agent interactions, the dimension explosion of the joint density presents a computational challenge. The implementations are usually particle-based [3], [4], which do not scale well with dimension, limiting this approach to state estimation of small swarms.

The other approach models and propagates the swarm in an aggregate sense, [6]–[8]. The implementations of this approach, particle-based and Gaussian Mixture models (GMMs), are more computationally attractive than those of the previous approach [8]. However, the particle-based implementation does not scale well with agent number, while GMMs generally assume linear Gaussian models for agent interactions.

In this paper, we propose a novel implementation of the gas-kinetic (GK) Probability Hypothesis Density (PHD) filter [9] by using the Tensor Train (TT) format [10], a computationally efficient alternative to particle-based filtering in the regime of high agent number. Specifically, we benefit from using the TT format in efficiently propagating a partial differential equation (PDE) model of the swarm, providing more general modeling of agent interactions than GMMs. By employing the principle of separation of variables across dimensions [10], the curse of dimensionality may be alleviated when approximating a multi-dimensional function in the TT format with low Tensor-Train rank. The space complexity of the TT approximation is linear in dimension and polynomial in the TT-rank [10] and has been used previously in modeling multi-dimensional functions in chemistry [11], physics [12], and queueing theory [13].

The contributions of this work include the following. First, we propose a novel Tensor-train-based algorithm TT-GK-PHD that compresses and computes the high dimensional integrals of the GK-PHD filter in the efficient TT format. Second, we provide a general analysis of our algorithm, and show that under certain assumptions, the runtime of TT-GK-PHD is \(O(n^4)\) while the runtime of a grid-based solution similar to the one in [14] is \(O(n^5)\), where \(n\) is the grid resolution of the state space. Third, we validate our analysis in computational experiments and demonstrate the usefulness of our implementation in a scenario which the grid-based solution fails due to hardware limitations.

II. BACKGROUND ON TENSOR TRAIN FORMAT

A. Tensor Train Format

A computationally efficient representation of a tensor, or multidimensional array, is the Tensor-Train format [10]. A d-dimensional tensor \(x\) with mode sizes \(n_1, \ldots, n_d\) is simply an \(n_1 \times \cdots \times n_d\) array. Such an array is represented by \(O(n^d)\) parameters, where \(n := \max_i \{n_i\}\). The TT format attempts to break the representation’s exponential dependence on the dimension \(d\), i.e., the curse of dimensionality, by approximating the tensor \(x\) with a parametric representation based on the principle of separation of variables. We say that \(x\) is in TT format if each element of \(x\) satisfies \(x(i_1, \ldots, i_d) = U_1(i_1)U_2(i_2)\cdots U_d(i_d)\), where \(U_k(i_k)\) is an \(R_{k-1} \times R_k\) matrix with \(R_0 = R_d = 1\). Thus, the TT format requires only \(O(dnR^2)\) parameters, where \(R := \max_k \{R_k\}\). Note that the memory depends linearly in the dimension \(d\), at least
formally. One can decompose $x$, exactly or approximately, into its cores $U_1, \ldots, U_d$ with $d-1$ sequential SVDs, using a procedure TT-SVD [10].

Not only does the TT format enable efficient compression of multidimensional arrays, but it allows for basic linear algebra operations in this compressed format, such as addition, Hadamard product (element-wise multiplication), and inner products with time and space complexities no greater than $O(dnR^3)$ [10], [15].

B. Tensor Train Matrix

By considering a suitable representation of linear operators over tensor spaces, “matrix-vector”-like operations and linear system solving in the TT format can be made computationally efficient [15]–[17], which cannot be performed well in other tensor formats such as the Tucker format. We now discuss a representation for tensor operators called the Tensor-Train-Matrix (TTM) format that facilitates these highly useful operations.

Consider operator $A$ of size $(m_1 \times n_2 \times \cdots \times m_d) \times (n_1 \times n_2 \times \cdots \times n_d)$, which operates on $d$-dimensional tensors. Operator $A$ is in TTM format if the following holds for each element of $A$: $A((i_1, j_1), \ldots, (i_d, j_d)) = V_1(i_1, j_1)V_2(i_2, j_2)\cdots V_d(i_d, j_d)$, where $V_k(i_k, j_k)$ is an $R_{k-1} \times R_k$ matrix with $R_0 = R_d = 1$. Arrays $V_1, \ldots, V_d$ are called the cores of $A$, and the parameters $R_k$ are called the TTM ranks of $A$. The cores of $A$ are of dimension $R_{k-1} \times m \times n \times R_k$; thus, the complexity of the TTM format is higher than that of the TT format. Specifically, TTM format requires $O(dmnR^2)$ parameters, where $m := \max_k \{m_k\}$, $n := \max_k \{n_k\}$ and $R := \max_k \{R_k\}$. One may use the TT-SVD procedure to compute the TTM cores of $A$.

Many classes of operators are known to have efficient TTM representations, i.e., the TTM-ranks are low enough that there are significant storage savings when compared to the full number of parameters $O(m^dn^d)$. These include Laplace and Laplace Inverse Operators [17], certain operators associated with PDEs in chemistry and physics [18]–[20], and also multidimensional matrices with circulant, block, or banded structure [21], [22]. We will use the TTM format to compress high-dimensional kernels in a PDE used in swarm biology [14].

C. Tensor Train Procedures

a) TT-round - Controlling TT-rank growth: To address TT-rank growth, we employ the procedure TT-round which finds an approximate TT decomposition with quasi-optimal TT-ranks [10]. Let tensor $x$ be in TT format with maximum TT-rank $R$, and $\epsilon$ be the prescribed accuracy in the Frobenius norm. The procedure TT-round($x, \epsilon$) returns a Tensor-Train $y$ with quasi-optimal TT-ranks in $O(dnR^3)$ operations such that $\|x - y\|_F \leq \epsilon \|x\|_F$.

b) TT-AMEEn - Iterative linear solver in TT-format: The procedure TT-AMEEn is an alternating least squares (ALS) method that minimizes the residual of a linear system in TT format by sequentially updating the TT cores of an initial guess until a convergence tolerance is met [23]. Each core update is the solution of an auxiliary, smaller, and better-conditioned linear system, which is manageable with standard linear solvers. Let $v \leftarrow$ TT-AMEEn($A, b, \epsilon$) denote the approximate solution returned by TT-AMEEn for the linear system $Ax = b$ with prescribed accuracy $\epsilon$: $\|Av - b\|_F \leq \epsilon \|b\|_F$. Here, $A$ is given appropriately in the TT format, while $b$ and $v$ are given in the TT format. For a symmetric and positive definite (SPD) matrix $A$, TT-AMEEn was proven to converge geometrically [23]. TT-AMEEn works well in practice for matrices close to SPD, and even general matrices. One advantage of the TT-AMEEn procedure is its rank-revealing nature, and thus, the TT-ranks of $v$ need not be known a priori. The time and space complexity of TT-AMEEn is summarized in the following lemma.

Lemma II.1 (Lemma 1 from [24]) We solve $Ax = b$, where the TT-ranks of $A \in \mathbb{R}^{n \times n}$ are all bounded above by $R_A$, and the TT-ranks of $b$ are all bounded above by $R_b$. Let the TT-ranks of iterates $v_k$ and solution $v_\infty$ all be bounded above by $R_v$. Then, one iteration of TT-AMEEn has space complexity $O(dnR^2 + dR^2 R_A + dR R_b)$ and time complexity $O(dnR^2 R_A + dnR^2 R_A + dnR^2 R_b)$.

c) TT-mv - Iterative TT matrix-vector product: At times, TT-round is inefficient when rounding a direct product of tensors. In such cases, it is generally more efficient to compute a low TT-rank approximation iteratively. To this end, we use a procedure called TT-mv that is based on TT-AMEEn. To compute an approximation $y \approx Ax$, one can minimize $\|y\|^2 - 2y^TAx$, which may be solved iteratively with TT-AMEEn($I, Ax, \epsilon$), where $I$ is the identity matrix. Because this method is based upon the TT-AMEEn procedure, the TT-ranks of the approximate matrix-vector product need not be known a priori.

Corollary II.2 Let $y$ be the output of TT-mv($A, x, \epsilon$), where the TT-ranks of $A \in \mathbb{R}^{n \times n}$ are all bounded above by $R_A$, the TT-ranks of $x$ are all bounded above by $R_x$, and $\epsilon$ is the prescribed accuracy. Let the TT-ranks of the iterates $y_k$ and the solution $y$ be bounded above by $R_y$. Then, one iteration of TT-mv($A, x, \epsilon$) has space complexity $O(dnR^2 + dR_y R_{Ax})$ and time complexity $O(dnR^2 R_{Ax})$, where $R_{Ax} := R_A R_x$.

III. PROBLEM DEFINITION AND ALGORITHM

A. Problem Definition

We borrow a mesoscopic model from swarm biology that propagates a swarm density, due to its ability to model general agent interactions, such as local attraction, repulsion and alignment [14], [25], [26]. This model may also be adapted to include goal regions and even interactions with other swarms. We consider inter-agent attraction, repulsion and alignment and also goal attraction. We use the PDE from [14] to propagate the density $\mu(\phi, y, x)$ of agents in
where $\gamma$ is the (constant) agent speed, $\bar{e}_\phi$ is the direction vector $(\cos \phi, \sin \phi)$, $\nabla$ is the spatial gradient $(\partial_x, \partial_y)$, $\lambda(\phi, y, x)$ is the rate at which agents in state $(\phi, y, x)$ turn, and $\tau$ is the rate at which agents at position $(x, y)$ reorient themselves from $\phi'$ to $\phi$.

We adapt turning rate $\lambda$ and reorientation rate $\tau$ from [14] to model attraction, repulsion, alignment, and goal attraction forces as follows. Define $\lambda := \lambda_{at} + \lambda_{re} + \lambda_{al} + \lambda_g$, where $\lambda_{at}$ is the turning rate due to inter-agent attraction, and similarly for the inter-agent repulsion, inter-agent alignment and goal attraction terms. Define $\lambda_{at} := \int_{\phi_1}^{\phi_2} \kappa(\phi, y, x, \tilde{y}, \tilde{x}) \mu(\phi, x, y) d\phi \cdot d\tilde{y}d\tilde{x} = \kappa_{at}(\phi, y, x) \mu_{at}(\phi, x, y) d\phi \cdot d\tilde{y}d\tilde{x}$, where goal density $\mu_g$ is the target density constraint for the swarm. The goal density is used to compute the goal attraction rate, while the inter-agent rates use the swarm density. The kernels $\kappa_{at}$, $\kappa_{re}$, $\kappa_{al}$ and $\kappa_g$ have two components, orientation and relative distance with subscripts $o$ and $d$, respectively:

$\kappa_{at}(\phi, y, x, \tilde{y}, \tilde{x}) := \kappa_{at, o}(\phi, \psi) \kappa_{at, d}(\tilde{x} - x, \tilde{y} - y) \mu(\phi, y, x) d\phi \cdot d\tilde{y}d\tilde{x}$

$\kappa_{re}(\phi, \psi, y, x) := \frac{\pi}{2\sigma}(1 - \cos(\phi - \psi))$ for $j \in \{at, re\}$, and $\kappa_{at, o}(\phi, \theta) := \frac{\pi}{2\sigma}(1 - \cos(\phi - \theta))$, and distance kernels $\kappa_{at, d}(x, y) := \int \frac{1}{\sqrt{\pi}e^{|z|^2}} |w(z)| \partial z d\pi z$ for $j \in \{at, re\}$, with parameters $q_j$, $d_j$, and $\lambda_{j}$, and normalization $A_j$.

Define reorientation rate $\tau := \tau_{AT} + \tau_{RE} + \tau_{AL} + \tau_{G}$, where $\tau_{AT} := \int \kappa_{at}(\phi, y, x, \tilde{y}, \tilde{x}) \mu(\phi, y, x) d\phi \cdot d\tilde{y}d\tilde{x}$, and $\tau_{RE}$, $\tau_{AL}$, and $\tau_{G}$ are similar definitions.

We define $\omega_j$, the turning probability from $\phi'$ to $\phi$ as:

$\omega_j(\phi', \phi) := f_{\sigma}(\phi' - \phi - k_j \sin(\phi - \psi))$, where $k_j$, $\sigma$, and $\psi$ are parameters.

Using this model, we simulate a swarm with a fixed and known number of agents $N_a$, while periodic measurements are being taken of individual agents from $J$ independent sensors. Each sensor $s$ takes noisy measurements of a random number of agents in the swarm with agent detection probability $p_s$. Let $M_s$ be the index set of the agents which are sensed by sensor $s$ at the current time step. Let $\ell_s(\cdot ; y) : \mathcal{X} \rightarrow \mathbb{R}$ be the likelihood function for sensor $s$ given measurement $y$. At every time step, sensor $s$ outputs an unordered set of measurements ${y_{n,s}}_{n \in M_s}$, with duration $\Delta t$ between successive measurement-sets.

We would like to estimate the time-varying swarm density $\mu$, by propagating (1) while incorporating the measurements in a principled and computationally efficient manner.

B. Algorithm Description

The GK-PHD filter [9], an extension of the PHD filter [28], was proposed to deal with multitarget tracking for urban environments, in which it is often difficult to resolve individual targets. The GK-PHD filter propagates the density using a two-step process: a time-update followed by a measurement-update. In the time-update, the agent density is propagated according to a continuity PDE. In our work, we use (1) in this step. Due to suitability in handling multi-sensor data, we then do the measurement-update from the iterated-corrector PHD filter [29], [30].

Each iteration, we compute turning rate $\lambda$ and reorientation rate $\tau$, and then update density $\mu$ via (1). We acquire measurements and update the density iteratively: $\mu[s] := \frac{N_a - |M|}{N_a} \mu[s - 1] + \frac{1}{N_a} \sum_{n \in M_s} \int_{\ell_s(\cdot ; y_{n,s})} \mu[s - 1](\cdot ; y_{n,s}) d\phi \cdot d\tilde{y}d\tilde{x}$, where $\mu[0]$ is the density from the time-update, and $\mu[s]$ is the density updated with measurements from sensors $1, \ldots, s$.

Then, the output of the measurement-update is density $\mu[J]$. Although the sensor ordering affects the outputted density, it was shown in practice that the agent densities arising from different sensor orderings do not differ significantly when the detection probability $p_s$ for each sensor $s$ is high [30].

C. Discretization of the PDE

To propagate (1), we discretize (1) and solve the resulting linear system. We employ an implicit Euler scheme to discretize in time the linear operator on the left-hand side of (1), while we explicitly evaluate the nonlinear right-hand side of (1); $(I + \Delta t \frac{\partial}{\partial \phi} \cdot \nabla - \tau^i) \mu[t] = \mu[t] + \Delta t(-\lambda \mu[t] + \int_{-\pi}^{\pi} \tau^i(\phi', y, x) \mu[t](\phi', y, x) \Delta \phi')$, where $\bar{\lambda}$ is the identity operator, and $\mu[t] + \mu[t] + \mu[t] + \mu[t]$ are the updated densities of iterations $i$ and $i + 1$, respectively.

Next, we discretize the equation above on a uniform grid. Denote $X \in \mathbb{R}^n$ as the uniform discretization of state space $\mathcal{X}$, where $n$ is the number of points per axis. We discretize density $\mu$ on grid $X$, i.e., $u := \mu[X]$. We discretize the kernels that generate the rates $\lambda$ and $\tau$ on grid $X$. Denote $K_j$ as the tensor arising from the discretization of $k_j$ on $X$, for $j \in \{at, re, al, g\}$, $\bar{\lambda}$, and $\tau$ on $X$. We discretize the integral above on grid $X$: $V_i := \sum_{\phi} T^i(\phi, \phi', y, x) u_i(\phi', y, x) \Delta \phi'$. The PDE discretized on
grid $X$ may be written as: $(I_{n} + \Delta t \gamma(D_x + D_y)) u^{i+1} = u^{i} + \Delta t(-I'u^{i} + V^{i})$, where $D_x := \cos(\Phi) \otimes I_n \otimes d_x$, $D_y := \sin(\Phi) \otimes d_y \otimes I_n$, and $d_x$ are $n \times n$ central difference matrices in the $x$ and $y$ dimensions, respectively, and $\Phi := [-\pi + \frac{2\pi}{n} : \frac{2\pi}{n} : \pi] \in \mathbb{R}^n$ uniformly discretizes $[-\pi, \pi]$. Define kinematics matrix $A := I_{n+3} + \Delta t (D_x + D_y)$. Thus, we solve the following linear system for density $u^{i+1}$: $Au^{i+1} = u^{i} + \Delta t(-I'u^{i} + V^{i})$.

D. Tensor Train Implementation of GK-PHD

In Algorithm 1, we detail the TT implementation of the GK-PHD filter, which we call TT-GK-PHD. Lines 1-4 constitute the setup of TT-GK-PHD, setting parameters, initializing densities, pre-computing variables in TT format. Parameter $N_I$ is the total number of iterations for state estimation. We pre-compute kernels $K^{TT}_j$ in TT format $K_j$ for $j = a, r, e, g, a, l, d, r, e, a, l, d$ and $g$, and pre-compute kernels $K^{TT}_j$ in TT format $K_j$ for $j = a, l, o$ and $a, l, o$, up to accuracy $\epsilon$. Not knowing the TT-ranks of these kernels a priori, we employ rank-revealing procedures TT-nv and TT-cross [31], an efficient alternative to TT-SVD. In lines 6-11 we do the time-update of TT-GK-PHD, and in lines 12-23 we do the measurement-step while sensor measurements arrive each iteration in line 15. We prefer to do the measurement-step of TT-GK-PHD without the TT format for optimal runtime performance. In line 5, we transition to the next iteration, and the a posteriori density $u^{TT}$ of the current iteration is outputted in line 23. Line 6 calls Algorithm 2 to compute $I^{TT}$. Line 7 calls Algorithm 3 to compute $V^{TT}$. Define $\Sigma_{j, d}(u)$ as the partial summation of a $d$-dimensional tensor $u$ across mode $j$, resulting in a contracted tensor of dimension $d - 1$. For example, $\Sigma_{1,3}(u^{TT})$ contracts the goal density $u^{TT}$ across the first mode, the heading. Define $I_n$ as an $n$-vector of ones.

E. FFT-GK-PHD: Grid-based implementation of GK-PHD

To benchmark TT-GK-PHD, we develop a grid-based implementation to perform the time-update similar to [14]. We detail our benchmark filter FFT-GK-PHD in Algorithm 4. As in [14], we rewrite the integrals in $l$ and $T$ as convolutions, and then evaluate them via the discrete Fourier Transform. We refer the reader to [14] for more details. Lines 1-4 of Algorithm 4 is the FFT-GK-PHD setup. Lines 6-10 is the time-update. In line 9, the linear system is solved via MATLAB’s middivide function. Lines 11-20 is the measurement-update, essentially identical to the measurement-update of TT-GK-PHD.

IV. ANALYSIS

A. Complexity of the setup of TT-GK-PHD

We first analyze the time and space complexity of the TT-GK-PHD setup (lines 1-4 of Algorithm 1). The computational burden is line 3 in computing the interaction kernels in TT/TTM format with TT-SVD and TT-round. We also use TT-cross [31], an efficient alternative to TT-SVD, specifically in the TT approximation of $w_j$. $C$ is the number

Algorithm 1 TT-GK-PHD

1: Set filter resolution $n$; number of iterations $N_I$
2: Set goal density $u^{TT}_g$; swarm density $u^{TT}$
3: Compute interaction kernels $K^{TT}_i, K^{TT}_j$
4: Compute kinematics matrix $A^{TT}$
5: for iterations $i = 1, \ldots, N_I$ do
6: Compute turning rate $I^{TT}$
7: Compute reorientation $V^{TT}$
8: Compute $g^{TT} \leftarrow$ TT-SVD($-I^{TT} \otimes u^{TT} + V^{TT}, \epsilon$)
9: Compute $b^{TT} \leftarrow$ TT-round($u^{TT} + g^{TT} \Delta t, \epsilon$)
10: Set $u^{TT} \leftarrow$ TT-AMEn($A^{TT}, b^{TT}, \epsilon$)
11: Normalize: $u^{TT} \leftarrow \frac{1}{\|u^{TT}\|} u^{TT}$
12: Convert to sparse array: $u \leftarrow$ full($u^{TT}$)
13: for each sensor $s = 1, \ldots, J$ do
14: Reset: $v_s = 0$
15: Acquire measurements of sensor $s$: $\{y_{s,n}\}_{n \in M_s}$
16: for each measurement $y_{s,n}$ from sensor $s$ do
17: Compute likelihood: $\ell_s(X; y_{s,n})$
18: $v_{s,n} \leftarrow u \otimes \ell_s(X; y_{s,n})$
19: $v_s \leftarrow v_s + \frac{1}{\|v_{s,n}\|} v_{s,n}$
end for
21: Sensor $s$ update: $u \leftarrow \frac{N_a - |M_s|}{N_a} u + \frac{1}{N_a} v_s$
end for
23: Convert to TT format: $u^{TT} \leftarrow$ TT-SVD($u, \epsilon$)
24: end for

Algorithm 2 Compute turning rate $I^{TT}$

1: $u^{TT}_{re} \leftarrow \Sigma_{1,3}(u^{TT})\Delta \theta$
2: $u^{TT}_{re} \leftarrow \Sigma_{1,3}(u^{TT})\Delta \theta$
3: $I^{TT}_{al} \leftarrow$ TT-round($K^{TT}_{al} \otimes u^{TT} \Delta x \Delta y, \epsilon$)
4: $I^{TT}_{re} \leftarrow$ TT-round($K^{TT}_{re} \otimes u^{TT} \Delta x \Delta y, \epsilon$)
5: $I^{TT}_g \leftarrow$ TT-round($K^{TT}_g \otimes u^{TT} \Delta x \Delta y, \epsilon$)
6: $I^{TT}_a \leftarrow K^{TT}_{al} \otimes (1 \otimes u^{TT})$
7: $I^{TT}_{al} \leftarrow \Sigma_{1,4}(I^{TT}_{al})\Delta \theta$
8: $I^{TT}_{re} \leftarrow$ TT-round($K^{TT}_{al} \otimes u^{TT} \Delta x \Delta y, \epsilon$)
9: $I^{TT} \leftarrow$ TT-round($I^{TT}_{al} + I^{TT}_{re} + I^{TT}_g + I^{TT}_{al}, \epsilon$)

Algorithm 3 Compute reorientation $V^{TT}$

1: $u^{TT}_{re} \leftarrow I_n \otimes I_n \otimes \Sigma_{1,3}(u^{TT})\Delta \theta$
2: $u^{TT}_{re} \leftarrow I_n \otimes I_n \otimes \Sigma_{1,3}(u^{TT})\Delta \theta$
3: $T^{TT}_{AT} \leftarrow$ TT-nv($K^{TT}_{AT} \otimes u^{TT} \Delta x \Delta y, \epsilon$)
4: $T^{TT}_{RE} \leftarrow$ TT-nv($K^{TT}_{RE} \otimes u^{TT} \Delta x \Delta y, \epsilon$)
5: $T^{TT}_g \leftarrow$ TT-nv($K^{TT}_g \otimes u^{TT} \Delta x \Delta y, \epsilon$)
6: $T^{TT}_{AL} \leftarrow K^{TT}_{AL} \otimes (1 \otimes I_n \otimes u^{TT})$
7: $T^{TT}_{AL} \leftarrow \Sigma_{1,5}(T^{TT}_{AL})\Delta \theta$
8: $T^{TT}_{AL} \leftarrow$ TT-round($K^{TT}_{AL} \otimes T^{TT}_{AL}, \epsilon$)
9: $T^{TT} \leftarrow$ TT-round($T^{TT}_{AT} + T^{TT}_{RE} + T^{TT}_g + T^{TT}_{AL}, \epsilon$)
10: $V^{TT} \leftarrow$ TT-SVD($T^{TT} \otimes (1 \otimes u^{TT}), \epsilon$)
11: $V^{TT} \leftarrow \Sigma_{2,4}(V^{TT})\Delta \phi$
Algorithm 4 FFT-GK-PHD

1: Set filter resolution $n$; number of iterations $N_I$
2: Set goal density $N_s$; swarm density $u$
3: Compute interaction kernels $K_j$
4: Compute sparse kinematics matrix $A$
5: for iterations $i = 1, \ldots, N_I$ do
6: Compute turning rate $t$
7: Compute reorientation $V$
8: Compute $b \leftarrow u + \Delta t(-l \odot u + V)$
9: Set $u \leftarrow A^T b$
10: Normalize: $u \leftarrow \frac{1}{\sqrt{\sum_n u_n^2}} u$
11: for each sensor $s \in 1, \ldots, J$ do
12: Reset: $v_s \leftarrow 0$
13: Acquire measurements of sensor $s$: $\{y_{s,n}\}_{n \in M_s}$
14: for each measurement $y_{s,n}$ from sensor $s$ do
15: Compute likelihood: $\ell_s(X; y_{s,n})$
16: $v_{s,n} \leftarrow u \odot \ell_s(X; y_{s,n})$
17: $v_s \leftarrow v_s + \frac{1}{1+n_s} v_{s,n}$
18: end for
19: Sensor $s$ update: $u \leftarrow \frac{N_s - |M_s|}{N_s} u + \frac{1}{N_s} v_s$
20: end for
21: end for

Proposition IV.1 The setup of TT-GK-PHD requires the computation of the kernels up to some accuracy $\epsilon$: $K^{\text{AT}}_{ct}$, $K^{\text{AT}}_{ct}$, $K^{\text{RE}}_{ct}$, $K^{\text{G}}_{ct}$, $K^{\text{AL,O}}_{ct}$, $K^{\text{AL,D}}_{ct}$, $K^{\text{AL,O}}_{ct}$, and $K^{\text{AL,D}}_{ct}$. The time complexity is $O(n^3 + nC(R_{\text{at}} + R_{\text{g}} \text{al,d}) + (R_{\text{at}} + R_{\text{g}} \text{al,d})^3)$ and the space complexity is $O(n^3 + n(R_{\text{AL,O}} + R_{\text{AL,D}})^3 + (R_{\text{at}} + R_{\text{g}} \text{al,d})^3)$.

This proposition follows from the multiplicative construction of the kernels, and that the TT-rank of the product of Tensor-Trains is the product of the TT-ranks. Though this is a worst-case analysis of the complexity, we find in practice that the TT-ranks of these kernels are much lower. For example, the attraction and repulsion kernels actually have lower TT-ranks than the TT-ranks of the goal attraction kernel, due to the distance component. Lastly, due to the complex expression of the kernels, our insight of how the TT-ranks of the kernels trades-off with approximation error $\epsilon$ is limited, and we leave further analysis for future work.

B. Complexity of the time-update of TT-GK-PHD

We make use of the following simplification in our analysis of the time-update of TT-GK-PHD (lines 6 to 11 of Algorithm 1).

Assumption IV.2 In Algorithm 1, the TT-ranks of kernels $K^{\text{AT}}_{ct}$, $K^{\text{AT}}_{ct}$, $K^{\text{G}}_{ct}$, $K^{\text{AL,O}}_{ct}$, $K^{\text{AL,D}}_{ct}$, $K^{\text{AT}}_{ct}$, and $K^{\text{AL,D}}_{ct}$ are upper bounded by $R_{\text{at}}$, $R_{\text{g}}$, $R_{\text{al,d}}$, $R_{\text{AT}}$, $R_{\text{RE}}$, $R_{\text{G}}$ and $R_{\text{AL,D}}$, respectively.

Line 6 of Algorithm 1 computes the turning rate $t^{\text{TT}}$ via Algorithm 2. The burden of Algorithm 2 is TT-round in lines 3-5 and 8-9, with time complexities $O(n(R_{\text{at}}^3))$, $O(n(R_{\text{g}}^3))$, and $O(n(R_{\text{al,d}}^3))$, respectively, and the space complexities $O(n(R_{\text{at}}^2))$, $O(n(R_{\text{g}}^2))$, and $O(n(R_{\text{al,d}}^2))$, respectively. We summarize this below.

Proposition IV.3 The time complexity of Algorithm 2 is $O(n^4(R_{\text{at}}^3 + R_{\text{g}}^3 + R_{\text{al,d}}^3))$. The space complexity of Algorithm 2 is $O(n^3(R_{\text{at}}^2 + R_{\text{g}}^2 + R_{\text{al,d}}^2))$.

Line 7 of Algorithm 1 computes $V^{\text{TT}}$ via Algorithm 3. Algorithm 3 is dominated by TT-INV, TT-round and TT-SVD. The number of iterations that TT-INV or TT-AMEEn performs. The time complexities of TT-INV in lines 3-5 are $O(n^4R_{\text{AT}}^3)$, $O(n^4R_{\text{RE}}^3)$, respectively, and the space complexities are $O(n^3R_{\text{AT}}^3)$, $O(n^3R_{\text{RE}}^3)$, respectively. In lines 8-9, the time complexities of TT-round are $O(n^4R_{\text{AL,D}}^3)$ and $O(n^4(R_{\text{AT}}^3 + R_{\text{RE}}^3 + R_{\text{G}}^2 + R_{\text{AL,D}}^2))$, respectively, and the space complexities are $O(n^3R_{\text{AL,D}}^3)$ and $O(n^3(R_{\text{AT}}^2 + R_{\text{RE}}^2 + R_{\text{G}}^2 + R_{\text{AL,D}}^2))$, respectively. In line 10, the time and space complexities of TT-SVD are $O(n^4R^3)$ and $O(n^3R^2)$, respectively. We assume that $nR$ upper bounds the TT-ranks of this TT-SVD output, where $R := O(R_{\text{AT}} + R_{\text{RE}} + R_{\text{G}} + R_{\text{AL,D}})$. We summarize these results below.

Proposition IV.4 The time complexity of Algorithm 3 is $O(n^4(C(R_{\text{AT}}^3 + R_{\text{RE}}^3 + R_{\text{G}}^3) + n^4R_{\text{AL,D}}^3))$. The space complexity of Algorithm 3 is $O(n^3(R_{\text{AT}}^2 + R_{\text{RE}}^2 + R_{\text{G}}^2 + R_{\text{AL,D}}^2))$.

In lines 8 and 9 of Algorithm 1, TT-SVD and TT-round each have time and space complexities of $O(n^4)$ operations and $O(n^3)$ memory, respectively. In line 10, the time and space complexities of TT-AMEEn are $O(n^4C)$ and $O(n^3)$, respectively, since the maximum TT-rank of the kinematics matrix $A^{\text{TT}}$ is 3. We summarize the complexity of the time-update below.

Lemma IV.5 The time complexity of the state-update of TT-GK-PHD (lines 6-11 of Algorithm 1) is $O(n^4(C + R_{\text{at}}^3 + R_{\text{g}}^3 + R_{\text{al,d}}^3 + C(R_{\text{at}}^2 + R_{\text{RE}}^2 + R_{\text{G}}^2 + R_{\text{AL,D}}^2)))$, while the space complexity is $O(n^3(R_{\text{at}}^2 + R_{\text{g}}^2 + R_{\text{al,d}}^2 + R_{\text{AT}}^2 + R_{\text{RE}}^2 + R_{\text{G}}^2 + R_{\text{AL,D}}^2))$.

C. Complexity of the measurement-update of TT-GK-PHD

In line 12, we convert density $u^{\text{TT}}$ into a sparse array, a more suitable format for the measurement-update. This conversion has time complexity of $O(n^4)$ and space complexity of $O(n^3)$. Line 17 computes the likelihood of a measurement $y_{s,n}$. Since there are $O(N_s)$ measurements per sensor and $J$ sensors, the total time and space complexities are $O(JN_sn^3)$ and $O(n^3)$, respectively. Lines 18, 19 and 21 have time and space complexities of $O(n^3)$. Line 23 recasts the density
back into TT format via TT-SVD, with time and space complexities $O(n^4)$ and $O(n^3)$, respectively. We summarize this below.

**Proposition IV.6** The time and space complexities of the measurement-update of TT-GK-PHD (lines 12 to 23 of Algorithm 1) are $O(n^4 + J_{Ni} n^3)$ and $O(n^3)$, respectively.

**D. Complexity of FFT-GK-PHD**

Lines 1-4 of Algorithm 4 is the FFT-GK-PHD setup. The setup complexity is dominated by line 3, the kernel computations, requiring $n^2$ FFTs of vectors of length $O(n^2)$. Thus, the time and space complexity of the FFT-GK-PHD setup is $O(n^4 \log n)$ and $O(n^3)$, respectively. Lines 6-10 is the time-update of FFT-GK-PHD. In line 6, the time and space complexity of computing $l$ is $O(n^3 \log(n) + O(n^3)$, respectively. In line 7, the time and space complexity of computing $V$ is $O(n^5)$ and $O(n^4)$, respectively. Line 8 has time and space complexities of $(n^3)$. Line 9 solves the sparse linear system via MATLAB’s mldivide function, where $A$ is sparse with $O(n^3)$ nonzero elements, and banded diagonal with bandwidth $O(n^2)$. Using an LU decomposition, this can be solved with time complexity $O(n^2)$ and space complexity $O(n^5)$ [32]; however, we observe in practice better complexity properties using MATLAB’s mldivide function. The time complexity of the time-update of FFT-GK-PHD is at least $O(n^7)$, and the space complexity is $O(n^4)$. Lines 11-20 is the measurement-update of FFT-GK-PHD. Similar to TT-GK-PHD, the time and space complexity of the measurement-update of FFT-GK-PHD is $O(J_{Ni} n^3)$ and $O(n^3)$, respectively.

**V. COMPUTATIONAL EXPERIMENTS**

All experiments in this section were carried out on Amazon Web Services (AWS) r5.4xlarge instance with 8 physical cores, 128 GB of RAM and Ubuntu 16.04 with MATLAB R2019a.

**A. Swarm Scenario**

Consider a swarm with 400 interacting agents. We run this scenario for 70 iterations with time step $\Delta t = 0.2 s$, and agent speed $\gamma = 1$. Agent states $(\phi, y, x)$ were initialized according to a multivariate Gaussian mixture with three components. The scenario is divided into two stages. For the first 25 iterations, the goal density is a bivariate Gaussian distribution centered at $(x, y) = (-2, -2)$, while in the latter stage the goal density is centered at $(x, y) = (2, 2)$. The state space is discretized with $n = 2^8$ points per axis. Sensors are located at $(s_x, s_y) = (6, -6), (-6, 0),$ and $(0, 6)$. Each sensor $s$ has detection probability $p_s = 0.9$ and takes a three-dimensional measurement of an agent’s state $(\phi, y, x)$:

$$\sqrt{(x - s_x)^2 + (y - s_y)^2}, \text{atan}2(y - s_y, x - s_x), \phi,$$

a distance, bearing, and heading measurement, corrupted by additive Gaussian noise with covariance matrix $\text{diag}(0.3^2, 0.05^2, 0.1^2)$.

This scenario could not be computed with FFT-GK-PHD even on a computing system with 760 GB of RAM, due to insufficient memory when storing the interaction kernels. We ran TT-GK-PHD, having a setup runtime of 200 s, using 18 GB to store the interaction kernels, and having an average runtime per iteration of 426 s with a standard deviation of 67 s. There is a trade-off between the kernel TT-ranks and the accuracy of the kernel decomposition. The error in approximating the kernels ranged from $10^{-11}$ to $10^{-15}$. The highest of the kernel TT-ranks was 223 for the goal attraction $K_G$. To compare, the kernel TT-ranks for attraction, alignment and repulsion was 134, 108 and 106, respectively. We cannot report errors in the state estimation since the ground truth FFT-GK-PHD was not computable. In Fig. 1 we plot heat maps of the swarm density at iterations 5, 30 and 60. Access the following weblinks to view a videostream of these heatmaps across all iterations: https://tinyurl.com/rusqtg2 and https://tinyurl.com/qnw8jjv.

**B. Computational Study of TT-GK-PHD**

We study the complexity of TT-GK-PHD from a computational perspective. We ran a scenario as that in Section V-A except with 300 agents and measurement noise covariance matrix $\text{diag}(0.5^2, 0.1^2, 0.2^2)$. We ran TT-GK-PHD and FFT-GK-PHD with the same initial swarm density for three grid resolutions: $n = 2^5, 2^6$ and $2^7$ grid points per axis.

We compute errors as follows. Let $\text{FTT}^i(u_0)$ and $\text{TTR}^i(u_0)$ denote swarm densities at iteration $i$ of FFT-GK-PHD and TT-GK-PHD initialized with swarm density $u_0$, respectively. Since TT-GK-PHD approximates FFT-GK-PHD, we take the swarm densities of FFT-GK-PHD as ground truth. The relative error of TT-GK-PHD at iteration $i$ is computed

$$\frac{||\text{FTT}^i(u_0) - \text{TTR}^i(u_0)||_F}{||\text{FTT}^i(u_0)||_F}$$

in the Frobenius norm. One might expect this error to grow unbounded with iteration, but we find that this error stays bounded, due to the inherent stability of the GK-PHD filter. For each grid resolution $n$, the maximum relative error of TT-GK-PHD across all iterations was found to be 0.109, 0.0384 and 0.0119 for grid resolutions $n = 2^5, 2^6$ and $2^7$, respectively. Though the stability properties of the PHD filter have not been well-studied [33], this stable estimation error suggests that the GK-PHD filter is robust to small model perturbations, and we leave analytical results in this area for future work.

In Fig. 2 the logarithm of the setup runtimes is plotted against the logarithm of the grid resolution $n$. The line slopes for TT-GK-PHD and FFT-GK-PHD are 1.9 and 3.6, respectively, taking into account the differing logarithm bases. Note that the scaling factor of the setup runtime for TT-GK-PHD is better than for FFT-GK-PHD. In our analysis, we estimated time complexities $O(n^3)$ and $O(n^4 \log n)$ for TT-GK-PHD and FFT-GK-PHD, respectively. Though our analysis overpredicts the time complexities, the relative performance gain of TT-GK-PHD of an order to two orders of magnitude is predicted by the analysis. The computational burden of the setup runtime is in computing the goal attraction kernel.

In Fig. 3 the compression of the kernels is plotted against the logarithm of the grid resolution $n$. The compression is defined as the ratio of the total memory of the kernels used in
Fig. 1. Heatmaps of the spatial agent density computed by TT-GK-PHD at iterations 5, 30 and 60 (from left to right) for the swarm described in Section V-A. In the top plots, the arrow encodes the maximum of the marginal distribution of heading $\phi$ at that position $(x, y)$. In the bottom plots, the arrow depicts the state of an individual agent.

Fig. 2. Logarithm of the setup runtime versus logarithm of grid resolution. FFT-GK-PHD to the total memory of the kernels used in TT-GK-PHD. The line slope is 1.6, implying that the TT-ranks of the kernels grow sublinearly with grid resolution. The relative error for approximating the kernels was on the order of $10^{-11}$. Even though the relative errors in approximating the kernels in TT format is quite small, the state estimation errors, though stable, turn out to be significantly higher, due to the compounding of errors across iterations. We found that reducing the approximation error in the kernels produces higher TT-ranks, but no significant improvement in the state estimation error.

In Fig. 4 the mean runtime per iteration is plotted against the logarithm of the grid resolution $n$, along with two standard deviations lower and upper bounds. TT-GK-PHD outperforms FFT-GK-PHD across all grid resolutions. The line slopes are 3.2 and 4.1 for TT-GK-PHD and FFT-GK-PHD, respectively. In our analysis, we estimated time complexities of $O(n^4)$ and $O(n^5)$, respectively, suggesting that MATLAB is employing certain sublinear routines. Our analysis correctly predicts the relative improvement of TT-GK-PHD in the iteration runtime, a one order of magnitude gain. Lastly, as also noted in [14], the computational burden during the runtime of FFT-GK-PHD is in handling the align-
ment rate, due to limitations in the FFT approach. However, the TT ranks of the alignment kernels are remarkably low and effectively handle these time-consuming integrations in the state estimation.

REFERENCES

[1] T. Demerly, Watch U.S. F/A-18 Hornets Unleash Swarm of Mini-Drones in First Test, 2017 (accessed December 7, 2018), https://theaviationist.com/2017/01/11/watch-u-s-fa-18-hornets-unleash-swarm-of-mini-drones-in-first-test/.

[2] Autonomous Flying Microbots (RoboBees), 2018 (accessed December 7, 2018), https://wyss.harvard.edu/technology/autonomous-flying-microbots-robobees/.

[3] Z. Khan, T. Balch, and P. Dellaportas, “Mcm-based particle filtering for tracking a variable number of interacting targets,” IEEE transactions on pattern analysis and machine intelligence, vol. 27, no. 11, pp. 1805–1819, 2005.

[4] S. K. Pang, J. Li, and S. J. Godsill, “Models and algorithms for detection and tracking of coordinated groups,” in Aerospace Conference, 2008 IEEE. IEEE, 2008, pp. 1–17.

[5] R. Prasanth and H. Hoang, “Probability hypothesis density tracking for interacting vehicles in traffic,” in Signal Processing, Sensor Fusion, and Target Recognition XXI, vol. 8392. International Society for Optics and Photonics, 2012, p. 83920J.

[6] R. P. Mahler, “Detecting, tracking, and classifying group targets: a unified approach,” in Signal Processing, Sensor Fusion, and Target Recognition X, vol. 4380. International Society for Optics and Photonics, 2001, pp. 217–229.

[7] N. Correll and H. Hamann, “Probabilistic modeling of swarming systems,” in Springer Handbook of Computational Intelligence. Springer, 2015, pp. 1423–1432.

[8] M. Pace, M. Birattari, and M. Dorigo, “The swarm/potential model: Modeling robotics swarms with measure-valued recursions associated to random finite sets,” IEEE Transactions on Robotics, page submitted, 2013.

[9] R. Mahler, “Urban multitarget tracking via gas-kinetic dynamics models,” in Signal Processing, Sensor Fusion, and Target Recognition XXII, vol. 8745. International Society for Optics and Photonics, 2013, p. 87450B.

[10] I. V. Oseledec, “Tensor-train decomposition,” SIAM Journal on Scientific Computing, vol. 33, no. 5, pp. 2295–2317, 2011.

[11] M. Reis, J. A. Kromer, and E. Klipp, “General solution of the chemical master equation and modularity of marginal distributions for hierarchic first-order reaction networks,” Journal of mathematical biology, pp. 1–43, 2018.

[12] I. Gavilyuk and B. Khoromskij, “Quantized-fft-cayley transform for computing the dynamics and the spectrum of high-dimensional hamiltonians,” Computational Methods in Applied Mathematics Comput. Methods Appl. Math., vol. 11, no. 3, pp. 273–290, 2011.

[13] D. Kressner and F. Macedo, “Low-rank tensor methods for communicating markov processes,” in International Conference on Quantitative Evaluation of Systems. Springer, 2014, pp. 25–40.

[14] R. C. Fetecau and J. Meskas, “A nonlocal kinetic model for predator–prey interactions,” Swarm Intelligence, vol. 7, no. 4, pp. 279–305, 2013.

[15] N. Lee and A. Cichocki, “Fundamental tensor operations for large-scale data analysis in tensor train formats,” arXiv preprint arXiv:1405.7786, 2014.

[16] I. Oseledec, “Approximation of matrices with logarithmic number of parameters,” in Doklady Mathematics, vol. 80, no. 2. Springer, 2009, pp. 653–654.

[17] I. V. Oseledec, “Approximation of $2^d \times 2^d$ matrices using tensor decomposition,” SIAM Journal on Matrix Analysis and Applications, vol. 31, no. 4, pp. 2130–2145, 2010.

[18] V. Kazeev, M. Khammash, M. Nip, and C. Schwab, “Direct solution of the chemical master equation using quantized tensor trains,” PLoS Comput Biol, vol. 10, no. 3, p. e1003359, 2014.

[19] V. Kazeev and C. Schwab, “Tensor approximation of stationary distributions of chemical reaction networks,” SIAM Journal on Matrix Analysis and Applications, vol. 36, no. 3, pp. 1221–1247, 2015.

[20] B. N. Khoromskij, I. V. Oseledec et al., “Dmrg+qtt approach to computation of the ground state for the molecular schrödinger operator,” 2010.

[21] V. A. Kazeev, B. N. Khoromskij, and E. E. Tyrtyshnikov, “Multilevel toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity,” SIAM Journal on Scientific Computing, vol. 35, no. 3, pp. A1511–A1536, 2013.

[22] I. Oseledec, E. Tyrtyshnikov, and N. Zamarashkin, “Tensor-train ranks for matrices and their inverses,” Computational Methods in Applied Mathematics Comput. Methods Appl. Math., vol. 11, no. 3, pp. 394–403, 2011.

[23] V. V. Dolgov and D. V. Savostyanov, “Alternating minimal energy methods for linear systems in higher dimensions,” SIAM Journal on Scientific Computing, vol. 36, no. 5, pp. A2248–A2271, 2014.

[24] S. Dolgov and B. Khoromskij, “Simultaneous state-time approximation of the chemical master equation using tensor product formats,” Numerical Linear Algebra with Applications, vol. 22, no. 2, pp. 197–219, 2015.

[25] R. C. Fetecau, “Collective behavior of biological aggregations in two dimensions: a nonlocal kinetic model,” Mathematical Models and Methods in Applied Sciences, vol. 21, no. 07, pp. 1539–1569, 2011.

[26] R. Eftimie, G. De Vries, M. Lewis, and F. Lutscher, “Modeling group formation and activity patterns in self-organizing collectives of individuals,” Bulletin of mathematical biology, vol. 69, no. 5, p. 1537, 2007.

[27] C. Zmurchok and G. de Vries, “Direction-dependent interaction rules enrich pattern formation in an individual-based model of collective behavior,” PLoS one, vol. 13, no. 6, p. e0198550, 2018.

[28] R. Mahler, “Multitarget moments and their application to multitarget tracking,” LOCKHEED MARTIN TACTICAL DEFENSE SYSTEMS-EAGAN ST PAUL MN, Tech. Rep., 2001.

[29] R. P. Mahler, “Multitarget bayes filtering via first-order multitarget moments,” IEEE Transactions on Aerospace and Electronic systems, vol. 39, no. 4, pp. 1152–1178, 2003.

[30] S. Nagappa and D. E. Clark, “On the ordering of the sensors in the iterated-corrector probability hypothesis density (phd) filter,” in Signal Processing, Sensor Fusion, and Target Recognition XXI, vol. 8050. International Society for Optics and Photonics, 2011, p. 80500M.

[31] I. Oseledec and E. Tyrtyshnikov, “Tt-cross approximation for multidimensional arrays,” Linear Algebra and its Applications, vol. 432, no. 1, pp. 70–88, 2010.

[32] E. Ki¸c¸ and P. Stanica, “The inverse of banded matrices,” Journal of Computational and Applied Mathematics, vol. 237, no. 1, pp. 126–135, 2013.

[33] F. Caron, P. Del Moral, M. Pace, and B.-N. Vo, “On the stability and the approximation of branching distribution flows, with applications to nonlinear multiple target filtering,” Stochastic analysis and applications, vol. 29, no. 6, pp. 951–997, 2011.