AIAA Guidance, Navigation, and Control Conference
13 - 16 August 2012, Minneapolis, Minnesota

AIAA 2012-4682

Planning under Uncertainty using Bayesian Nonparametric Models

Trevor Campbell*, Sameera Ponda†, Girish Chowdhary‡, Jonathan P. How§
Laboratory for Information and Decision Systems (LIDS), MIT, Cambridge, MA

The ability to plan actions autonomously to achieve predefined objectives in the presence of environmental uncertainties is critical to the success of many Unmanned Aerial Vehicle missions. One way to plan in the presence of such uncertainties is by learning a model of the environment through Bayesian inference, and using this model to improve the predictive capability of the planning algorithm. Traditional parametric models of the environment, however, can be ineffective if the data cannot be explained using an a priori fixed set of parameters. In Bayesian nonparametric models (BNPs), on the other hand, the number of parameters grows in response to the data. This paper investigates the use of BNPs in the context of planning under uncertainty. Two illustrative planning examples are used to demonstrate that the additional flexibility of BNPs over their parametric counterparts can be leveraged to improve planning performance and to provide the capability to identify and respond to unforeseen anomalous behaviors within the environment.

I. Introduction

The ability to autonomously plan missions is critical for ensuring the success of Unmanned Aerial Systems (UASs). Autonomous planning typically involves selecting actions that maximize the mission objectives given available information, such as models of the agent dynamics, environment, available resources and mission constraints. In realistic missions these models are often not known precisely, and such uncertanties can have a negative impact on the performance of the UAS if they are ignored. Thus, an important aspect of autonomous planning for UASs is to take the uncertanties in available information into account when making decisions.

Uncertainty can enter the problem at various levels; for example, at the vehicle level, it may appear as modelling uncertainty resulting from inaccurate models of the vehicle. Adaptive control and robust control are examples of methods that have been widely studied to handle modelling uncertainties[1–4]. Uncertainty also enters at the mission level as a result of limited prior knowledge

*S.M. candidate, Dept. of Aeronautics and Astronautics, MIT tdjc@mit.edu
†Ph.D. candidate, Dept. of Aeronautics and Astronautics, MIT sponda@mit.edu
‡Postdoctoral Associate, Dept. of Aeronautics and Astronautics, MIT girishc@mit.edu
§Richard C. Maclaurin Professor of Aeronautics and Astronautics, MIT jhow@mit.edu

about the environment. For example, an accurate model of the environment may not be available a priori, or the environment might change, making it difficult to decide on the best course of action. A typical approach employed throughout the literature is to start with a prior model, and adapt the model to the environment as the agent executes its mission. This approach relies on machine learning techniques, and is in many ways similar to indirect adaptive control. The classical approach to handling mission level uncertainties is to assume the environment has a certain structure and may be modelled by a parametric model. The parameters of this model are then adapted in response to data as it becomes available using Bayesian inference[5–8]. These methods however, can be ineffective when the data cannot be modelled accurately with the a priori assumed model structure.

Bayesian nonparametric models (BNPs), on the other hand, yield a class of Bayesian inference techniques in which the number of parameters grows in response to observed data. BNPs have a flexible structure and, therefore, are particularly useful in situations where it is not clear what the model structure should be a priori. Several authors have explored BNP techniques within machine learning [9–20]. For example, Fox *et al.* explored BNPs for speaker diarization[21], Teh and Jordan used BNPs for document classification[12], and Hjort *et al.* showed that BNPs can be used for classifying life history data[16]. BNPs have also been explored in the robotics community. Rasmussen *et al.* have explored Gaussian Process (GP) based models for regression and classification[20], Ko *et al.* used GP observation and process models for Kalman filtering[22], and Murray-Smith *et al.* proposed a class of indirect adaptive control problems using GPs[23,24]. Several authors have also used GPs for planning in the framework of dynamic programming and reinforcement learning [25,26]. However, BNPs in general do not appear to have been used extensively in the context of planning. This paper attempts to bridge this gap by investigating how a wider class of BNPs can be integrated into typical planning frameworks. In particular, it is shown that planning frameworks employing BNPs can result in improved planning performance when planning under uncertainty.

Section II presents an overview of some BNP techniques that can be used for planning. In Section III, BNPs are applied to two example planning problems and the performance of planners employing BNPs is compared to that of planners using parametric models. Conclusions and suggestions for future work are discussed in Section IV.

II. Introduction to Bayesian Nonparametric Modelling

This section begins by motivating the use of BNPs through a discussion of two common problems in planning and control whose performance can be enhanced through the use of nonparametric models as opposed to fixed parameter models. These problems serve to highlight the need for models whose complexity changes in response to observed data. A short survey of a number of BNPs is then presented, and their applicability to the motivating examples is discussed.

II.A. Motivating Examples

The first planning scenario considered involves performing surveillance and tracking of targets in the environment using an unmanned aerial vehicle (UAV). Traditional target tracking methods

are typically reactive, where the surveillance UAV uses the current available information about the target states to plan its trajectory accordingly. In many instances it is desirable, however, to pursue more intelligent planning strategies that can leverage complex target models to enhance the predictive capabilities of the planning algorithms. For example, in scenarios where one UAV is tasked with tracking multiple targets, resources may be better spent focusing on targets that are exhibiting interesting or anomalous behaviors, rather than spending equal amounts of effort on all targets, including those that are operating as expected. It is beneficial from a planning perspective, therefore, to be able to model and detect what is considered anomalous behavior and to predict when modes flagged as interesting are likely to occur.

There are several types of models that are useful for predicting complex target behavior within planning algorithms. Typical examples used throughout the literature include parametric models such as Hidden Markov Models (HMMs)[27] for modelling switching behaviors and Gaussian Mixture Models (GMMs)[28] for clustering. These models suffer from one crucial limitation: they assume a fixed number of behaviors or clusters, a quantity that is not often known a priori and may be subject to change over time (e.g. targets can learn new behaviors). This motivates the use of more flexible modelling strategies, such as BNPs, that can autonomously learn the number of distinct target behavior modes, the transition dynamics, and the model parameters given the available target data.

The second example of interest considers a multi-agent task allocation problem[29,30]. In this problem, the goal is to allocate several tasks amongst members of a team of autonomous agents such that there are no conflicts, while maximizing the (possibly stochastic) reward received for performing the tasks. If we allow the tasks to involve exogenous agents, two issues arise: 1) the reward distributions for tasks can no longer be assumed known a priori, since the exogenous agents are likely to have an influence upon them; and further, 2) the reward received for doing a task involving one exogenous agent may not contain any information about a future task involving another exogenous agent, as each exogenous agent may have a different influence on the received reward distribution. In order to predict the actions of exogenous agents with which a planning system has not yet interacted, information from previous interactions with exogenous agents must be used effectively.

One way to do this is by sorting exogenous agents into groups or classifications by examining the behaviors they exhibit; for example, if the team of UAVs observes that exogenous agents which move aggressively are often hostile to the team during surveillance tasks, the planner should learn a negative reward for completing tasks involving such aggressive exogenous agents. Thus, by classifying the exogenous agents based on observations of the behaviors they exhibit, both the first and second issues are resolved; the planning algorithm can use past tasks completed involving one type of exogenous agent to predict the reward for a future task involving an exogenous agent of that type. In the presence of process and measurement noise, this introduces the need for a clustering procedure. However, all parametric mixture models require the specification of the number of mixture components (in this case, the number of behaviors exhibited by the exogenous agents), and in realistic scenarios this quantity is generally not known. Hence, a model which can

be learned without this information a priori would be more applicable in realistic scenarios.

These two examples illustrate that a major issue with the use of a parametric model in a planning system is that the number of parameters in the model must be specified in advance, and cannot be inferred from the observed data. This issue is alleviated through the use of BNPs, where the number of parameters is automatically adjusted in response to the observed data and needs not be known a priori. Thus, the working hypothesis in this paper is that utilizing the flexibility of BNPs in planning algorithms can lead to enhanced system performance in many mission scenarios, such as surveillance, tracking, and task allocation.

II.B. Bayesian Nonparametric Models

Bayesian modelling and inference generally involves using measurements to improve a probabilistic model of an observed phenomena. The two major components of a Bayesian probabilistic model are the likelihood and the prior: the likelihood represents the probability of the generated data given a set of parameters, while the prior represents the probability distribution over those parameters (sometimes referred to as latent variables, as they are unknowns whose value must be inferred) before any measurements are made. As measurements become available, the prior is combined with the likelihood of those measurements using Bayes' rule resulting in a posterior distribution, which is an updated distribution over the unobservable quantities given the observations. For example, consider a mixture model consisting of K components:

$$\begin{aligned} \pi|\alpha &\sim \text{Dir}(\alpha_1, ..., \alpha_K), \\ z_i|\pi &\sim \text{Categorical}(\pi), \\ \theta_k|H &\sim H,\ k = 1...K, \\ y_i|z_i, \{\theta_k\} &\sim F(\theta_{z_i}). \end{aligned} \tag{1}$$

In this model, each data point (y_i, z_i), where y_i is an observation and z_i is the index to the corresponding mixture component, is assumed to be generated by rolling a K-sided die with outcome probabilities π (the Categorical distribution) for z_i, and based on that outcome, sampling y_i from the corresponding distribution $F(\theta_{z_i})$ in the K mixture components. Typically, those K distributions are of the same family (e.g. Gaussian) but have different parametrization (e.g. the mean). In this case, the Dirichlet distribution is the prior over the K unknown mixing probabilities π, H is the prior over the unknown $\{\theta_k\}$, and $F(\cdot)$ is the likelihood parametrized by a single value. As measurements become available, the estimates of π and $\{\theta_k\}$ can be improved through Bayesian inference techniques. Eq. (1) is an example of a parametric Bayesian model. In such models, the number of parameters or latent variables in the model are fixed a priori (in this case, the K mixing probabilities and parameters), and are often chosen through expert judgment.

Bayesian nonparametric models, on the other hand, do not require the expert specification of the number of parameters; instead, the number of parameters in such models is infinite, while a finite quantity of observed data is assumed to be generated using only a finite subset of those parameters. Table 1 presents a list of standard and recently developed Bayesian nonparametric

Table 1. Typical applications of some BNPs[31]. Acronyms are defined in the text.

Model	Typical Application
GP	Learning continuous functions
DP	Data clustering with an unknown number of clusters
BP	Learning shared latent features with an unknown number of features
HDP-HMM	Learning a Markov model with an unknown number of states
HDP-SLDS-HMM HDP-AR-HMM	Learning hybrid system dynamics (switched linear or autoregressive) with an unknown number of modes
BP-AR-HMM	Learning multiple hybrid systems' dynamics with an unknown number of shared modes
HBP	Classification based on shared latent features

models[31]. The Gaussian process (GP) is perhaps the most well known example of an BNP, and is typically used as a model for continuous functions. This model has seen use in geo-surveying and emerging applications in system identification and other regression related areas [20,22]. The Dirichlet process (DP) and beta process (BP), on the other hand, are popular nonparametric models in the machine learning community (e.g., Switching Linear Dynamical Systems (SLDS) and Autoregressive (AR) models have been used by Fox *et al.* for modelling various physical phenomena whose number of modes are not known a priori[21]), but have seen very little application in planning and control, despite their ability to robustly capture the complexity of a system solely from data measured therein[32,33]. Therefore, we focus our attention on these two models and their hierarchical extensions, the hierarchical Dirichlet process (HDP), and the hierarchical beta process (HBP).

The Dirichlet Process

The Dirichlet process, developed formally in [34], is a stochastic process whose realizations are probability distributions. Before introducing it formally, we first provide an intuitive understanding using the notion of an infinite extension of the Dirichlet distribution. Let the k-dimensional unit simplex be defined as the set of vectors $\pi = (\pi_1, \ldots, \pi_k) \in \mathbb{R}^k$ such that $\sum_{i=1}^{k} \pi_i = 1$, and $\pi_i \geq 0$ for all i. Then, the Dirichlet distribution is a probability distribution over this k-dimensional unit simplex whose density is given by:

$$p(\pi) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\sum_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} (\pi_i)^{\alpha_i - 1}, \tag{2}$$

where $\Gamma(\cdot)$ is the Gamma function, and $\alpha = (\alpha_1, ..., \alpha_k)$ is a parameter vector. A draw from the Dirichlet distribution is denoted $(\pi_1, ..., \pi_k) \sim \text{Dir}(\alpha_1, ..., \alpha_k)$. Because the components in any draw from a Dirichlet distribution sum to 1, it is often used as a prior over mixing probabilities in a mixture model, such as the model in Eq. (1).

If we now let $(\alpha_1, ..., \alpha_k) = (\frac{a}{k}, ..., \frac{a}{k})$ for some $a > 0$ and let $k \to \infty$, we arrive at a Dirichlet distribution whose draws contain an infinite number of components which sum to 1. When used in a mixture model, such as in [35], this prior expresses the fact that data is generated from an infinite number of mixing components. The Dirichlet process is very similar to this conceptual

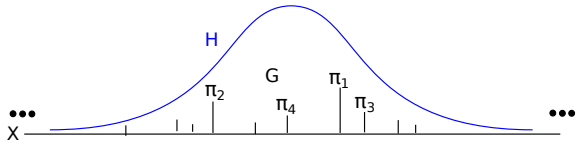


Figure 1. A pictorial representation of $G \sim \mathbf{DP}(\alpha, H)$.

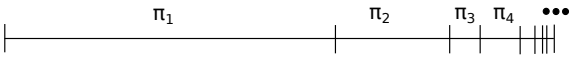


Figure 2. A pictorial representation of the stick breaking procedure.

model; samples from it contain an infinite number of components which sum to one, and it may be used as a prior over mixtures with an infinite number of components. However, it differs in an important way: each component of a sample from a Dirichlet process *also* has a location in some space X. Thus, draws from a Dirichlet process can be thought of as probability measures themselves, having an infinite number of *atoms* (discrete points with measure one) in X whose weights sum to one.

The definition of a Dirichlet process is as follows: Let H be a probability measure on a space X, and let $\alpha > 0$ be a real number. Then, $G \sim \mathrm{DP}(\alpha, H)$ is a probability measure drawn from a Dirichlet process if for any finite partition $\{B_1, ..., B_N\}$ of X (i.e. $B_i \bigcap B_j = \emptyset$ for all $i \neq j$ and $\bigcup_{i=1}^{N} B_i = X$)[34],

$$(G(B_1), ..., G(B_N))|H, \alpha \sim \mathrm{Dir}(\alpha H(B_1), ..., \alpha H(B_N)). \tag{3}$$

From this definition, it can be shown that $E[G(B)] = H(B)$, and that $V[G(B)] = H(B)(1 - H(B))/(\alpha + 1)$. This provides a sense of how H and α influence G: H, called the *base measure*, is essentially the mean of G; and α, called the *concentration parameter*, controls its variance. Further, it can be shown that G is discrete with probability 1, and has the following form[36]:

$$G = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i}, \tag{4}$$

where the weights π_i satisfy $\sum_{i=1}^{\infty} \pi_i = 1$, δ_x is an atom at x, and $\theta_i \sim H$ are the locations of the atoms. A representation of G is shown in Figure 1. This provides a mental picture of what a DP is: it can be thought of as a probability distribution, which, when sampled, yields another probability distribution G having a countably infinite number of atoms. Each atom has independent and identically distributed (i.i.d.) location within X with distribution H (with a slight abuse of notation, we use H to both refer to the measure and its related continuous distribution), where the weights π_i on the atoms can be thought of as being drawn from an infinite dimensional Dirichlet distribution.

In order to sample G from a DP, the *stick breaking construction* is typically used[21]. The stick breaking construction (sometimes denoted $\pi \sim \mathrm{GEM}(\alpha)$) provides an iterative procedure for sampling the weights of Eq. (4), given by:

$$\pi_i = \beta_i \prod_{j=1}^{i-1} (1 - \beta_j), \tag{5}$$

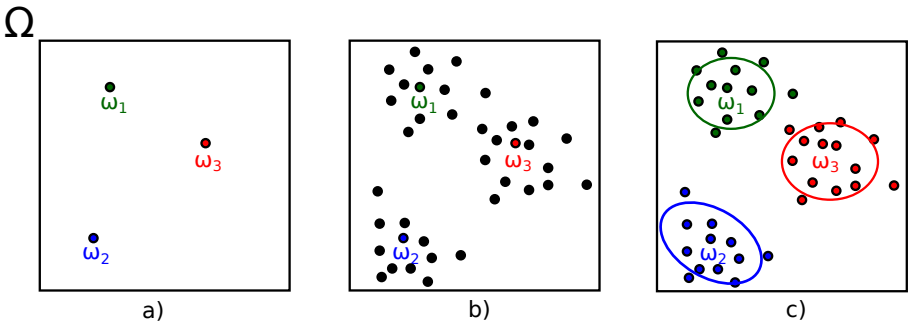


Figure 3. a) The measurements made of behaviors in an ideal scenario. Here, it is easy to tell that there are 3 behaviors exhibited by the agents. b) The measurements made of behaviors in a realistic scenario involving noise. c) The result of clustering those behavior measurements to reveal the 3 underlying behaviors.

where $\beta_i \sim Beta(1, \alpha)$ is the beta distribution. As shown in Figure 2, this corresponds to taking a stick of unit length, repeatedly breaking off beta distributed pieces, and assigning them to locations θ_i in X. Thus, to sample a draw from a DP, we simply alternately sample θ_i from H, and calculate the corresponding π_i by sampling β_i. Of course, there are theoretically an infinite number of (π_i, θ_i) pairs, but the stick breaking process can be terminated to obtain an approximation after a finite number of steps by re-normalizing the weights to sum to 1, or by assigning whatever remaining probability mass there is to the last θ_i.

Due to the form of draws from a DP shown in Eq. (4), we are now able to address some of the issues mentioned in the example problems of Section II.A. To perform clustering on behavior data in the example task allocation problem without assuming that the exogenous agents exhibit a fixed number of behaviors, we can just replace the Dirichlet distribution prior on the mixing probabilities of the Gaussian mixture model with the Dirichlet process as follows[37]:

$$\begin{aligned} G &\sim \mathrm{DP}(\alpha, H) \\ \theta_i | G &\sim G, \ i = 1...N \\ y_i | \theta_i &\sim \mathcal{N}(y_i; \theta_i), \ i = 1...N \end{aligned} \tag{6}$$

where the N observed data points y_i are i.i.d. with a Gaussian distribution with mean θ_i and known variance (this model can be easily extended to the case where the variance of each cluster is also unknown). Nowhere in this model is a number of behaviors assumed; because the θ_i are drawn from $G = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i}$, the number of distinct clusters exhibited in a finite set of observed data can instead be learned through Bayesian inference, as illustrated in Figure 3.

The target surveillance example problem requires learning an HMM with an unknown number of states. Drawing from concepts in the data clustering example, we might expect that an HMM including an infinite number of states is an appropriate choice of model when the actual number of states is unknown. In constructing such a model, we first note that a single draw $G_j \sim \mathrm{DP}(\alpha_j, H)$ from the Dirichlet process could be used as the transition distribution from a single state to the infinite set of other states. To have an infinite number of states, we therefore need to draw an infinite number of G_j. Further, to enforce that all draws $G_j \sim \mathrm{DP}(\alpha_j, H)$ share the same atom locations θ_i (such that each state i corresponds to one of the parameters θ_i), we require that H be

discrete with an infinite number of components. Thus, it seems that H should also be drawn from a Dirichlet process. This results in the hierarchical Dirichlet process (HDP)[38], which is written as:

$$H \sim \mathrm{DP}(\alpha_0, H_0) \quad \text{with} \quad G_j|H \sim \mathrm{DP}(\alpha_j, H),\ j = 1, 2..., \infty$$

where the distributions G_j are given by $G_j = \sum_{i=1}^{\infty} \pi_{ji}\delta_{\theta_i}$. The HDP can be used as a prior over the transition probabilities in an infinite state HMM, thus leading directly to the HDP-HMM[21] listed in Table 1 and shown in Figure 4. This model can then be easily used as the discrete switching component of a hybrid system, such as in an SLDS or AR process[21]. A known issue with the HDP-HMM regarding its practical implementation is that it tends to result in learning multiple redundant states during inference; however, the Sticky HDP-HMM (developed in [21]), which adds a probabilistic bias to states undergoing self-transitions, has been shown to mitigate this issue significantly.

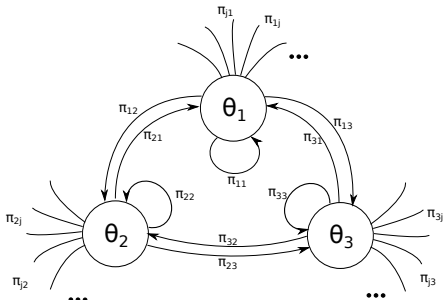


Figure 4. A representation of the HDP-HMM.

The Beta Process

The beta process, like the Dirichlet process, is a stochastic process whose realizations are discrete functions with an infinite number of components. If A is drawn from a beta process, it is typically denoted $A \sim \mathrm{BP}(c, B)$, where $c \in \mathbb{R}, c > 0$ is the *concentration parameter*, and B is some non-negative function on a space X. Unlike the base measure H of the Dirichlet process, B does not have to be a probability distribution; it only has to have a finite total mass b. A has been shown to have the following form[17]:

$$A = \sum_{i=1}^{\infty} q_i \delta_{\theta_i}, \tag{7}$$

where $q_i \in [0, 1]$ are the weights and θ_i are the locations of the atoms. Note that realizations from a beta process are very similar to realizations from a Dirichlet process, except for one key detail: while the weights π_i in a Dirichlet process realization must sum to 1, the weights q_i in a beta process realization must each individually fall within $[0, 1]$ and may sum to a number greater than 1. It has been shown that the expectation of this sum is actually finite if b is finite, despite the infinite number of q_i[16].

An analog to the Dirichlet process stick breaking procedure for approximately sampling from a beta process was developed in [17], given by Algorithm 1, where $\hat{A}_n$ is an approximation of $A \sim \mathrm{BP}(c, B)$. In this algorithm, rather than breaking an infinite number of beta-distributed pieces off a single stick, an infinite number of sticks are broken a single time, each with a beta-distributed resulting length. In addition, unlike the DP stick breaking procedure, there is no requirement for A to be a probability distribution; thus, this algorithm can be terminated at any point without further modification to the current best estimate $\hat{A}_n$.

Because of the requirement that the weights q_i of the beta process realization lie within $[0, 1]$, the beta process has been used as a prior over the probabilities for the Bernoulli process[17]. That is to say, if each observed data point can be represented as an infinite sequence of 1's and 0's which are modelled as being generated by an infinite sequence of Bernoulli trials, the beta process may be used as a prior over the infinite sequence of probabilities for these trials. A major advantage of using the beta process in this way is that the number of 1's in the infinite sequence of any data point is guaranteed to be finite (as long as the total mass b of B is finite), which allows this beta-Bernoulli model to be computationally tractable through a sparse representation of data. Further, if the locations θ_i are seen as features which are either included or not included in each data point (via a 1 or 0 in that index of the sequence, respectively), the beta process can be used as a prior over both the features exhibited by a set of data and the probability that any observation will exhibit each feature.

Algorithm 1 BP Sampling Algorithm[17]

Let $n = 1$ and $\hat{A}_0 = 0$
loop
  Sample $K_n \sim \text{Poi}\left(\frac{cb}{c+n-1}\right)$
  Sample K_n new locations θ_i from B/b
  Sample K_n weights $q_i \sim \text{Beta}(1, c + n - 1)$
  $\hat{A}_n \leftarrow \hat{A}_{n-1} + \sum_{i=1}^{K_n} q_i \delta_{\theta_i}$
  $n \leftarrow n + 1$
end loop

Referring back to the task allocation example problem in Section II.A, we note that each exogenous agent exhibits some subset of an overall shared set of behaviors; this can be represented as a binary feature vector, with a 1 entry representing a behavior exhibited by that agent, and a 0 representing a behavior not exhibited. Using the beta process as a prior over the probabilities of an agent exhibiting each behavior, we can learn a probabilistic model of the behaviors without having to know the number of behaviors exhibited a priori. However, our goal was to classify exogenous agents based on the behaviors they exhibit, and therefore we should use a model having a separate beta process prior A_k for each type k of exogenous agent. This allows exogenous agents within a classification to share the probabilities of exhibiting each behavior, while allowing for different probabilities across different classes.

As in the case of the HDP, we require that B be discrete with a countably infinite number of components, such that each A_k contains the same features (i.e. models the probabilities of the same overall set of behaviors). Thus, B should also be drawn from a beta process; this model is called the hierarchical beta process (HBP)[17,39], shown in Figure 5. Using the HBP model shown in Eq. (8), Bayesian inference can be performed as data becomes available to estimate a posterior distribution of the classification of objects.

$$\begin{aligned} \text{Highest Level Weights and Overall Set of Features} \quad & B \sim \text{BP}(c_0, B_0) \\ \text{Class } k \text{ with Class-Specific Weights and Same Features} \quad & A_k \sim \text{BP}(c_k, B) \quad \forall k \leq n \\ \text{Binary Feature Vector Drawn from Class } k \quad & y_{kj} \sim \text{BeP}(A_k) \quad \forall j \leq n_k \end{aligned} \tag{8}$$

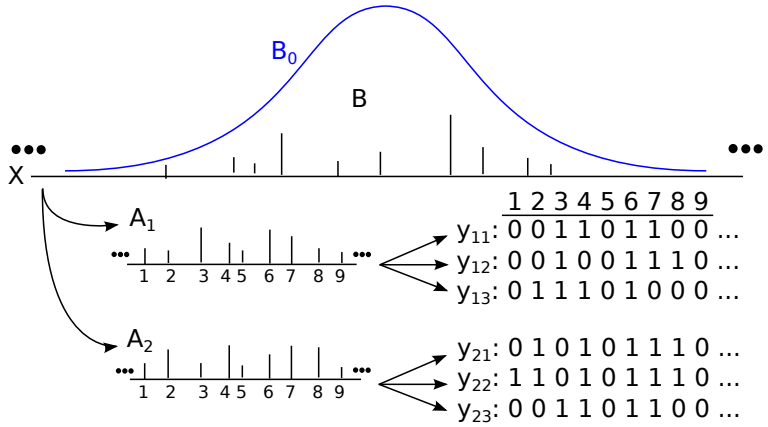


Figure 5. Pictorial representation of a draw from an HBP with two classes, resulting in binary feature vectors that represent behaviors exhibited by exogenous agents. Inference with this model essentially aims to learn the heights of each of the atoms in A_1 and A_2.

Bayesian Nonparametric Inference

Bayesian inference involves finding the posterior distribution $p(D|y)$ of model parameters D given data y. This is typically formulated using Bayes' rule,

$$p(D|y) = \frac{p(y|D)p(D)}{p(y)} \tag{9}$$

For example, in the case of the parametric Gaussian mixture model in Eq. (1), we would have

$$\begin{aligned} p(z, \pi, \theta|y) &= \frac{p(y|z, \pi, \theta)p(z|\pi, \theta)p(\pi)p(\theta)}{p(y)} \\ &= \frac{\prod_i \left[F(y_i; \theta_{z_i})\pi_{z_i}\right] \prod_k \left[H(\theta_k)\right] \mathrm{Dir}(\pi; \alpha_1, ..., \alpha_K)}{p(y)}. \end{aligned} \tag{10}$$

Even in simple cases such as this, the posterior distribution may be impossible to calculate analytically because of the normalization term in the denominator; although we can easily write down an integral formulation of $p(y)$, this integral can be impossible to solve. However, if the prior is *conjugate* to the likelihood, the posterior has the same distribution family as the prior, with different parameters. The Dirichlet distribution is conjugate to the Categorical distribution, and if we pick $H(\theta)$ to be Gaussian, it is conjugate to the Gaussian likelihood $F(y; \theta)$. This avoids the calculation of the integral $p(y)$ in Eq. (10), as we already know the normalization term based on the fact that the posterior is a product of a Dirichlet distribution and Gaussian distribution.

If the prior is not conjugate to the likelihood, or if we don't have an explicit form of $p(D)$ as is often the case with Bayesian nonparametric models (such as the Dirichlet process Gaussian mixture model in Eq. (6) where we do not have $p(G)$), there are a number of approximate methods for posterior inference available. The most common inference procedures involving BNPs are Markov Chain Monte-Carlo (MCMC) methods, such as Gibbs sampling and the Metropolis-Hastings algorithm[37]. Although it can be hard to tell if or when such algorithms have converged, they are simple to implement and have been successfully used in a number of studies[21]. In order to ameliorate the difficulties of using MCMC inference, other methods (such as variational Bayesian methods[40]) have

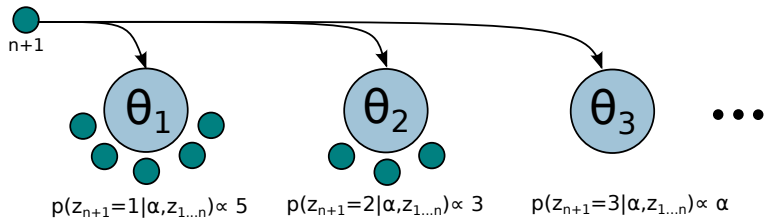


Figure 6. A pictorial representation of the CRP.

been developed.

The basic idea of Gibbs sampling is straightforward: if we have access to the conditional distributions of the parameters we wish to learn, we can approximate sampling from their joint distribution by iteratively sampling from each parameter's conditional distribution. For example, if we have a model with three parameters $\theta_1, \theta_2, \theta_3$ and we wish to sample from the posterior $p(\theta_1, \theta_2, \theta_3|y)$ given the data y, we can approximate this by iteratively sampling $p(\theta_1|y, \theta_2, \theta_3)$, $p(\theta_2|y, \theta_1, \theta_3)$, and $p(\theta_3|y, \theta_1, \theta_2)$. By doing this, we create a Markov chain whose equilibrium probability distribution is that of the full joint probability distribution that we wish to sample from. Therefore, we first let the Markov chain reach its equilibrium distribution by sampling from the conditional distributions T_1 times without recording any of the samples, and then take a sample every T_2 iterations from the algorithm to be a sample from the joint distribution. With the approximate joint distribution samples, we can estimate whatever statistics we might be interested in, such as the mean of the distribution.

This is particularly useful in the context of inference for the $\mathrm{DP}(\alpha, H)$ mixture model due to another construction of the Dirichlet process, the Chinese Restaurant process (CRP). This stochastic process models data as being generated sequentially, and upon generation of each data point, it is assigned a cluster. The name of the CRP comes from an analogy often used to describe it, which is illustrated in Figure 6. If we consider a restaurant with an infinite line of tables, where customers enter and either sit at a previously selected table with probability proportional to the number of customers sitting at that table, or start a new table with probability proportional to α, the table assignments are said to be drawn from a Chinese Restaurant Process. The equivalence of the CRP and the DP is made through de Finetti's theorem[41]: as each sequential data point is observed, it is impossible to tell whether it was sampled i.i.d. from the distribution $G \sim \mathrm{DP}(\alpha, H)$ or whether it was generated from the sequential cluster assignment rules of the CRP.

The CRP gives us the conditional distribution of the cluster assignments for Gibbs sampling with a DP mixture model. If we expand the mixture model of Eq. (6) using the stick breaking procedure to make the cluster assignments z_i for observations y_i explicit, we arrive at the following model:

$$\left.\begin{array}{ll} \pi & \sim \mathrm{GEM}(\alpha) \\ \theta_k & \sim H \\ z_i|\pi & \sim \mathrm{Categorical}(\pi) \\ y_i|z_i, \theta_{z_i} & \sim \mathcal{N}(y_i; \theta_{z_i}) \end{array}\right\} \quad k = 1, 2, ..., \infty, \quad i = 1, 2, ...N,$$

and the CRP provides us with the following probability distribution:

$$P(z_{n+1} = k|\alpha, z_{1,\ldots,n}) = \begin{cases} \frac{n_k}{n+\alpha} & k \leq K \\ \frac{\alpha}{n+\alpha} & k = K+1 \end{cases} \tag{11}$$

where n_k is the number of observations in cluster k. This distribution highlights the reinforcing nature of the CRP; the more popular a cluster is, the more likely the next measurement is going to be in that cluster. Then, we can sample from the posterior $p(z, \theta|y)$ to obtain estimates of the cluster assignments z and parameters θ by iteratively sampling from $p(z_i|y, z_{-i}, \theta)$ and $p(\theta_k|y, z, \theta_{-k})$, where z_{-j} or θ_{-j} are the set of z or θ with the j^{th} entry removed. These two distributions are shown mathematically in (12),

$$\begin{aligned} p(z_i = k|y, z_{-i}, \theta) &= \frac{p(y|z_i, z_{-i}, \theta)p(z_i|z_{-i}, \theta)}{p(y|z_{-i}, \theta)} \propto \begin{cases} \frac{n_k}{N+\alpha-1}\mathcal{N}(y_i; \theta_k) & k \leq K \\ \frac{\alpha}{N+\alpha-1}\int \mathcal{N}(y_i; \theta_k)H(\theta_k)d\theta_k & k = K+1 \end{cases} \\ p(\theta_k|y, z, \theta_{-k}) &= \frac{p(y|z, \theta)p(\theta_k|z, \theta_{-k})}{p(y|z, \theta_{-k})} \propto \prod_{i|z_i=k} \left[\mathcal{N}(y_i|\theta_k)\right] H(\theta_k), \end{aligned} \tag{12}$$

where the first is a discrete distribution, and the second is a Gaussian if H is the Gaussian conjugate prior, which are both easily sampled using well-known methods.

III. Planning with Bayesian Nonparametric Models

In a planning system that uses a probabilistic model of the environment in which it operates, the stochasticity of the model provides a means for observations to influence its underlying parameters through inference, allowing it to improve and provide more accurate predictions regarding the outcomes of possible actions. Bayesian nonparametric models enhance this flexibility by removing the constraint that the number of such underlying parameters must be specified a priori. This provides two advantages within the context of planning: BNPs can attain a higher fidelity than their parametric counterparts, providing better predictive capability to the planning system; and perhaps more subtly, as BNPs are able to change their structure in response to observations they can identify anomalous data, which may actually represent a new characteristic within the environment.

These new capabilities lead to several challenges within the planning problem. Key amongst these are the need to be able to perform efficient real-time (and plan time) inference, which typically involves some form of sampling. Further challenges are developing methods to validate the accuracy of the models, and determining how to appropriately respond within the planner to anomalous occurrences when detected. In the motivating problems discussed in Section II.A, two approaches are taken to this last problem: in the task allocation scenario, exogenous agents exhibiting anomalous behaviors within their set of behaviors

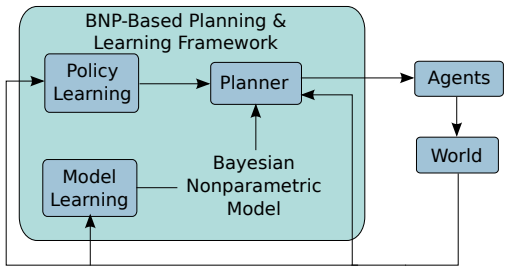


Figure 7. BNP planning and learning framework.

are classified by relating them to previously observed agents exhibiting a similar set of behaviors; and in the target surveillance scenario, the planning algorithm actively learns about anomalous behaviors by preferentially directing agents to observe such behaviors upon their occurrence.

Figure 7 illustrates the general planning framework we use in this work (a version with specific components is shown in each of the example problem sections in Figures 8 and 14). The model learning algorithm provides the planner with a Bayesian nonparametric model of the environment, while the policy learning algorithm provides the planner with an estimate of the utility of each of the possible actions to take based on previous experience.

III.A. Target Tracking and Surveillance

This section describes the implementation and results for the UAV surveillance and target tracking example introduced in Section II.A. The objective is to illustrate how BNP models can be learned from target data and leveraged within the planning framework to improve mission performance. In particular, this example involves collecting time-series data associated with a maneuvering target, and using a Sticky-HDP-HMM, as described in Section II.B, to learn an underlying Markov model that describes the observed target behaviors. Figure 8 shows a general block diagram of the system, specifying how the planning and learning components interact.

The objectives of the UAV surveillance mission consist of two components: to collect data obtained from tracking a ground target and use this data to build a complex target model including observed behaviors, underlying vehicle dynamics, and transitions between behavior modes; and to embed this learned model into the planning framework so as to enhance mission performance. Two particular example missions were executed and are described in detail below. The first involved using a learned model to predict anomalous target behavior and elicit an appropriate planning response. The second involved maximizing the mission reward given a reward function that depended on the underlying target dynamics and behaviors. For both examples, a ground target simulation was created and used to drive the target vehicles using the underlying Markov model shown in Figure 11. For the first mission, a simple 5-state Markov model was used (excluding the anomaly state), and for the second mission, a 6-state Markov model including the suspicious anomalous behavior mode was used to drive the target. The learning for both examples consisted of collecting target data and running Gibbs sampling to learn a Sticky HDP-HMM of the observed target behavior. Figure 9 shows the output and progress of the learning algorithm for the first example. Figure 9(a) shows the learned behavior modes and classification labels for the observations[a], (b) shows the true obser-

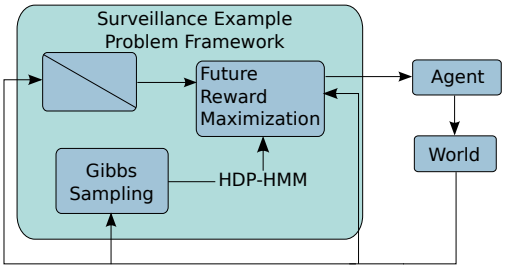
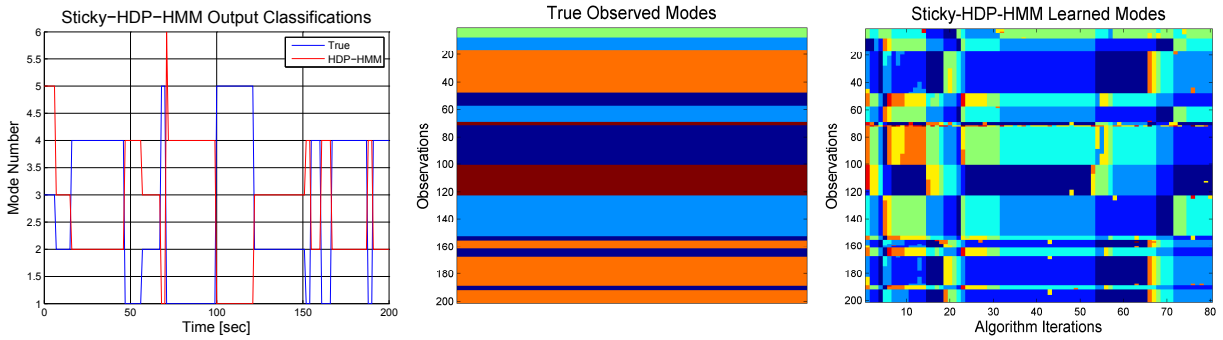


Figure 8. BNP planning and learning framework, with components specified for the surveillance problem.

[a]Note that the learning algorithm discovers modes through clustering of observations, but the clusters do not have a pre-specified ID or index. Therefore the label numbers for the learned clusters may be different than the label numbers for the true classification labels. This has no impact on the correctness of the model, since the metric of interest is whether the same observations are clustered together in the learned model and the true case, independent of the particular cluster IDs used.



(a) Sticky-HDP-HMM Output (b) True Observation Labels (c) Sticky-HDP-HMM Labeling

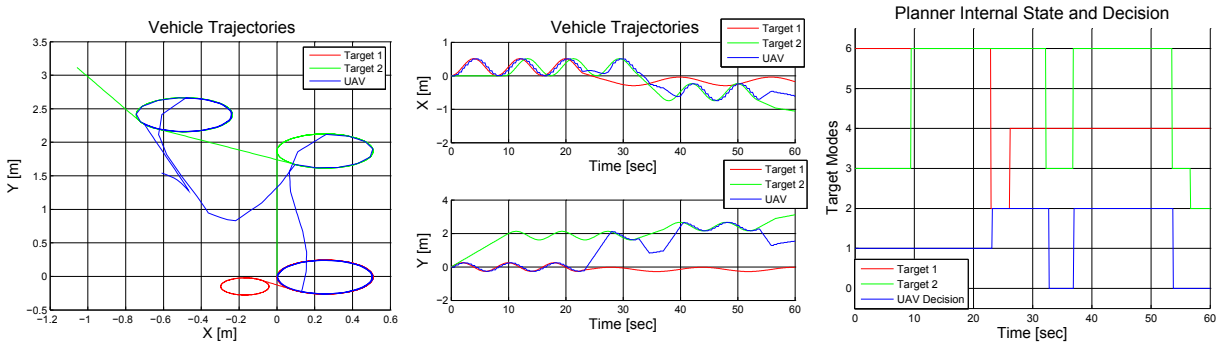
Figure 9. Sticky-HDP-HMM Learning: Results illustrating the performance of the Sticky-HDP-HMM algorithm for learning underlying Markov models from vehicle time-series data. (a) Sticky-HDP-HMM output labels vs. true vehicle mode history. (b-c) Image plots of observation labels vs. algorithm iterations: (b) true observation labels, and (c) Sticky-HDP-HMM observation labels.

vation clusters, where the observations are enumerated on the vertical axis, and the colors represent the particular clusters, and (c) shows the algorithm clustering results as a function of the number of iterations. Note that the first half of the clustering sample iterations displayed are thrown away as part of the burn-in period, and only the latter half are used in the final cluster assignments.

Results

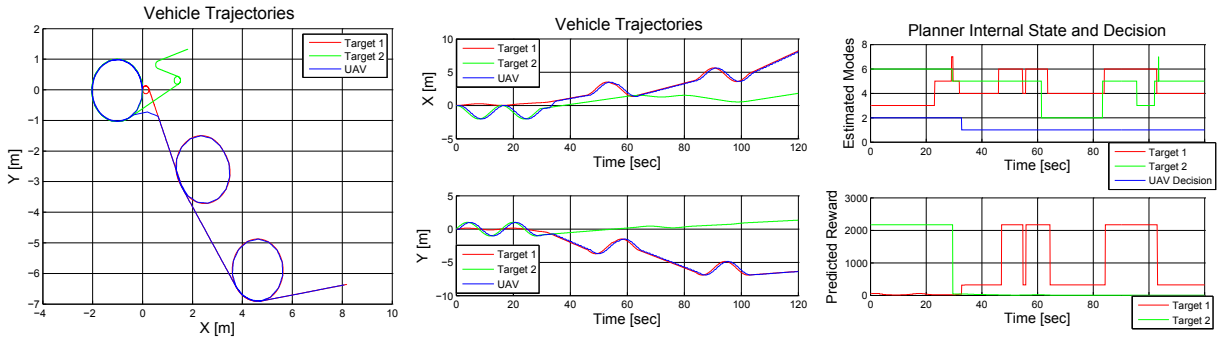
For the first example, the planning task was to use a UAV to perform surveillance on multiple ground targets, and to predict what behavior mode the targets were exhibiting, using the estimated target states and mode parameters provided by the learned model. If an anomaly was detected, then the planned UAV response was to create a target-following task to track the anomalous vehicle more closely (directly overhead), in order to obtain finer resolution data about the newly detected behavior mode. If no anomalies were detected, then the UAV resumed normal surveillance operation. A simulation involving a surveillance UAV tracking two ground vehicles was used to test the performance of this algorithm. The results for this example are shown in Figure 10. Figures 10(a)-(b) show the UAV and target trajectories (top view and trajectories vs. time respectively). Figure 10(c) shows the target mode history for both targets (where mode 6 was the anomalous mode), and the corresponding planner decisions (where 1 implied "Follow Target 1", likewise for 2, and 0 represented resuming normal surveillance). As shown in the trajectory plots, the UAV remains equidistant between both targets when the vehicles are behaving as expected, so as to minimize the time-to-target for either vehicle. When the planner detects an anomaly however, the UAV tracks the anomalous target more closely, following a trajectory directly overhead the suspicious vehicle. Once the anomalous target resumes normal operation, the surveillance UAV returns to a position between both targets.

For the second example, the planning task was to track ground targets using a surveillance UAV flying overhead, where rewards were obtained for being within tracking radius of the targets. However, in this scenario, the reward function was dependent upon the underlying behavior mode parameters, where the basic reward was 1, a reward of 2 was obtained when the targets were



(a) UAV and Target trajectories (b) Trajectories vs. Time (c) Target Modes and Planner Decisions

Figure 10. Surveillance and tracking example showing anomaly detections and planning responses using a learned target behavior model.



(a) UAV and Target trajectories (b) Trajectories vs. Time (c) Target Modes and Planner Decisions

Figure 12. Surveillance and tracking example where the planning algorithm leverages target behavior predictions over a time horizon to improve mission rewards.

tracked while exhibit turning modes, and a high reward of 100 was obtained if the targets were caught speeding and turning erratically (e.g. possible evasive maneuver detected). To test the performance of this algorithm, ground targets were driven using the 6-mode Markov model shown in Figure 11 (including the anomalous mode), and the Sticky-HDP-HMM was relearned using target data simulated from this new model. The results for a mission involving a UAV and two ground targets are presented in Figure 12, where (a) and (b) show the UAV and target trajectories, and (c) shows the estimated target modes and planning decisions (top) and the planner discounted reward function for both targets (bottom). Looking closely at plot (c), one can see that at around 32 seconds the planner switches from tracking Target 2 to tracking Target 1. This corresponds to Target 1 entering "Mode 4", which corresponds to "Forward Fast" mode in the Markov model. This is because, as observed in Figure 11, the target can only enter the suspicious mode from the "Forward Fast" mode. The planner therefore leverages this information to sacrifice immediate gains in favor of potential future higher rewards. In particular, in this example, the turning modes have higher rewards than the forward and stop modes, but the probability of reaching the anomalous mode in the future is highest when the current target is in "Forward Fast", therefore, the predictive planner chooses this mode over the turning modes.

To evaluate the performance of the planning algorithms described above using the learned Sticky-HDP-HMMs, a set of Monte Carlo simulations were run. For comparison with the Sticky-HDP-HMM, a Baum-Welch learning algorithm[42] was implemented, which learns observation labels, mode parameters, and transition probabilities, assuming a fixed number of modes. The results for both example scenarios described above are displayed in Figure 13. As shown in the plots, the system performance using a Baum-Welch algorithm with a lower number of modes than the actual is quite poor. As the number of fixed modes used for training the Baum-Welch algorithm approaches the correct number of modes, the performance of the system increases. In contrast, the Sticky-HDP-HMM algorithm does not require the specification of a fixed number of modes a priori, but learns these from the data, achieving comparable performance to the Baum-Welch which uses the true number of modes. This motivates the use of BNP strategies that are more flexible than their parametric counterparts, especially for situations in which the true number of modes is typically not known well. An interesting observation in Figure 13(a) is that the performance of the system does not monotonically increase as the number of assumed modes approaches the true value. In particular, in this example, using 3 modes produces consistently higher performance than using 4 modes. Even though this seems counterintuitive, the reason is that the HMM using Baum-Welch is forced to cluster multiple true modes into a single modelled mode, and thus learns parameters which model multiple clusters as well as possible but does not accurately capture the true underlying model. In the 4 mode model the "Forward Slow" and "Stop" modes were combined, and the resulting parameters for that mode led to the "Forward Slow" mode erroneously being identified as anomalous. The 3 mode model consistently combined different modes and generally correctly identified the "Forward Slow" mode as being normal. Because of how frequently the targets entered that mode, the effect that the correct identification of this mode had on the score was sufficient to cause the 3 mode model to perform better than the 4 mode model.

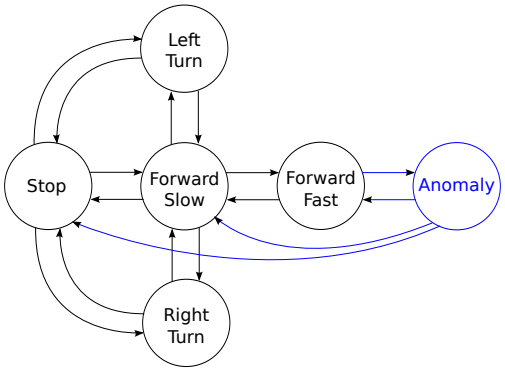
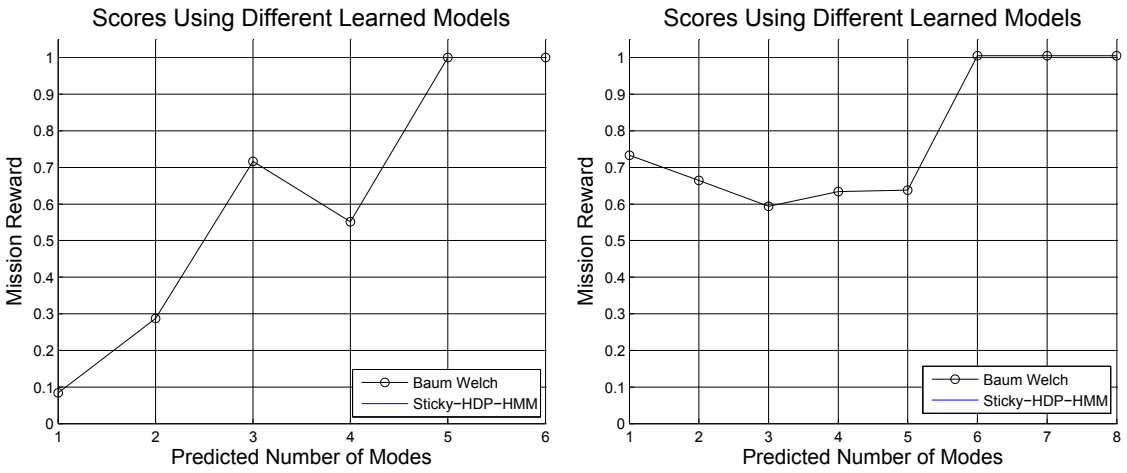


Figure 11. Markov model describing target behavior modes and transitions.

This demonstrates that in these stochastic settings, pursuing a heuristic learning strategy, such as increasing the number of assumed modes until the performance stops increasing, is not always advisable since the score using different numbers of fixed modes may exhibit local maxima. This motivates the use of Bayesian nonparameteric models that can learn the number of modes within a mathematically consistent framework.

III.B. Task Allocation

We return now to the task allocation example introduced in Section II.A. Even the simplest forms of the multi-agent task allocation problem in general are combinatorial problems[43], for which the optimal solution can be very difficult to find efficiently. One formulation of this problem is as



(a) Anomaly Detection and Tracking

(b) Behavior Prediction

Figure 13. Monte Carlo results comparing the performance of the planner using an HMM with a fixed number of modes vs. using a Sticky HDP-HMM.

follows:

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \quad & \sum_{i=1}^{N_a}\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta})\; x_{ij} \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) \leq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^{N_a \times N_t},\; \boldsymbol{\tau} \in \{\mathbb{R}^+ \cup \emptyset\}^{N_a \times N_t} \end{aligned} \tag{13}$$

where N_a and N_t are the number of agents and tasks respectively, $\mathbf{x} \in \{0,1\}^{N_a \times N_t}$ is a set of $N_a \times N_t$ binary decision variables, x_{ij}, which are used to indicate whether or not task j is assigned to agent i; $\boldsymbol{\tau} \in \{\mathbb{R}^+ \cup \emptyset\}^{N_a \times N_t}$ is a set of real-positive decision variables τ_{ij} indicating when agent i will execute its assigned task j (where $\tau_{ij} = \emptyset$ if task j is not assigned to agent i); $\boldsymbol{\theta}$ is a set of planning parameters, such as the classification of an exogenous agent; $\mathbf{c}_{ij}$ is the reward agent i receives for task j given the overall assignment and parameters; and $\mathbf{G} = [\mathbf{g}_1 \dots \mathbf{g}_{N_c}]^T$, with $\mathbf{b} = [b_1 \dots b_{N_c}]^T$, is a set of N_c possibly nonlinear constraints of the form $\mathbf{g}_k(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) \leq b_k$ that capture transition dynamics, resource limitations, etc.

A common approach to dealing with this complexity is to use a sequential greedy allocation algorithm, where tasks are allocated by iteratively finding the task/agent pair which results in the greatest net reward increase, and allocating that task to that agent. Sequential greedy solutions have been shown to provide acceptable approximations which are typically much faster to compute than the optimal solution [44]. However, in our task allocation example problem, there are two added complexities: the desirability of completing a certain task is stochastic with unknown distribution a priori, as we do not know how the exogenous agents will affect the scores of the tasks; and we would like the planning algorithm to classify the exogenous agents by the behaviors they exhibit, but we do not know what those behaviors are or how many there are a priori.

Given the identification of behaviors with a DP mixture model and the classification of the

exogenous agents based on these behaviors with the HBP, as discussed in Section II, this problem is transformed into a concurrent reward learning and task allocation problem; essentially, we must solve the task allocation problem as stated above, except we are no longer given $\mathbf{c}_{ij}$ for each exogenous agent classification $\boldsymbol{\theta}$. In other words, we have transformed the problem statement expressed in Eq. (13) to the following:

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \quad & \sum_{i=1}^{N_a}\sum_{j=1}^{N_t} E_r\left[\mathbf{c}_{ij}(\mathbf{x},\boldsymbol{\tau},r(\boldsymbol{\theta}))\right]\ x_{ij} \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) \leq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^{N_a \times N_t},\ \boldsymbol{\tau} \in \{\mathbb{R}^+ \cup \emptyset\}^{N_a \times N_t} \end{aligned} \tag{14}$$

where $r(\boldsymbol{\theta})$ is the base reward of doing a task involving an exogenous agent of classification $\boldsymbol{\theta}$ with probability distribution $p(r|\boldsymbol{\theta})$, and $E_r[\cdot]$ is the expectation over r. This introduces an extra level of complexity to the task allocation process, because we are not given $p(r|\boldsymbol{\theta})$ and must now include some notion of our uncertainty in the expectation of $\mathbf{c}_{ij}$ for each task to be allocated.

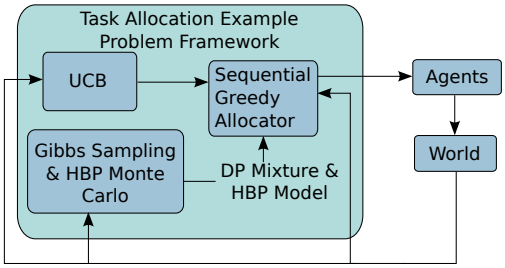


Figure 14. NPBM planning and learning framework, with components specified for the task allocation problem.

This is a problem of deciding between *exploitation* and *exploration*; that is to say, the planner now must decide whether to complete a task because it has a high expected reward or because we want to reduce the uncertainty in its reward. This question has been studied extensively in the reinforcement learning community, particularly in the context of the *N-armed Bandit* problem[45]. In that scenario, an autonomous agent is presented with a number of possible actions to take, each with an unknown but stationary reward distribution, and the agent's goal is to repeatedly select actions to take which maximize its reward over time. At each step, the agent can choose to exploit what it has learned by selecting what it believes to be the highest reward action, or to explore by selecting what appears to be a suboptimal action for the purpose of reducing its uncertainty in the reward provided by that action to ensure that it has correctly identified the optimal action. There are a number of solutions to that problem[45], but one which appears useful in the present context, the UCB algorithm[46], is shown in Algorithm 2. There are two major advantages to adapting this algorithm for task allocation over the alternatives. First, the algorithm has a logarithmic upper bound on regret (the difference between the optimal score and the received one)[46]; and second, the selection of which action to take next is deterministic and greedy, which makes it easy to integrate in common sequential greedy-based task allocation algorithms.

Thus, in this example, we use a sequential greedy task allocation algorithm coupled with UCB as a policy learning algorithm in the planning framework shown in Figure 14.

Results

In our task allocation simulation, a homogeneous team of three agents with Dubins' car dynamics operating in a rectangular 2D domain were responsible for completing uniformly spatially dis-

tributed static surveillance tasks whose creation was modelled as Poisson process events. There were 10 classes of exogenous agents and 150 different behaviors in the behavior space $\Omega = \mathbb{R}$, where observations of the behaviors were Gaussian distributed. As each task was created, the involved exogenous agent selected some subset of 150 different behaviors through a hierarchical beta process model, and exhibited this subset by emitting an observation for each selected behavior.

For 5 of the 10 classes, the reward for completing a task was Gaussian distributed with mean sampled from a uniform distribution over the set $[5, 20]$, while the other 5 classes had reward means sampled from a uniform distribution over the set $[-20, -5]$. These reward means were not revealed to the 3 agents. The cost for each task was revealed to the agents prior to completing the task, and as each task was created the cost was sampled from a uniform distribution over $[5, 20]$. The agents were able to reject tasks for which the net reward was expected to be negative.

Algorithm 2 UCB[46]

Try all actions $j = 1, \ldots, J$ once.
$n \leftarrow$ number of actions
$n_j \leftarrow 1, \forall j$
$\bar{x}_j \leftarrow$ reward obtained from action j, $\forall j$
loop
  $k \leftarrow \underset{j}{\operatorname{argmax}} \left(\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}} \right)$
  Try action k, receive reward R
  $\bar{x}_k \leftarrow \bar{x}_k n_k + R$
  $n_k \leftarrow n_k + 1$
  $\bar{x}_k \leftarrow \bar{x}_k / n_k$
  $n \leftarrow n + 1$
end loop

Before starting the simulation, the 3 agents were provided with a class-labeled training data set of behavior observations generated by 20 exogenous agents. To illustrate the effectiveness of Bayesian nonparametric models, both parametric Gaussian mixture models and Dirichlet process mixture models were used to cluster the behavior observations. When using the parametric models, the assumed number of clusters was varied between 0% and 150% of the correct number, and the observations were clustered using expectation maximization to identify cluster parameters and maximum likelihood to assign the observations to individual clusters. When the nonparametric model was used, observations were clustered using Gibbs sampling. Using the cluster assignments of the behavior observations for each training example, a binary feature vector was constructed. The resulting 20 binary feature vectors were used to train an HBP classification model, with the method presented in [17].

As each task was created, the team first identified the behaviors exhibited by the involved exogenous agent using the aforementioned clustering procedures, and classified the agent with the trained HBP model. Note that the exogenous agents created during the simulation occasionally exhibited behaviors not seen in the training data; these were accounted for properly by the DP Gibbs sampling inference, but the maximum likelihood method used in the parametric model inference was forced to assign these behaviors to clusters observed in the training data.

Figure 15 shows the mean and standard deviation of the classification accuracy of the nonparametric model and of the parametric model as the assumed number of observed behaviors is varied between 1% and 150% of the actual number of observed behaviors. While the accuracy of classification using a parametric model depends heavily on the assumed number of behavior clusters, the nonparametric model does not require the specification of this quantity and the classification

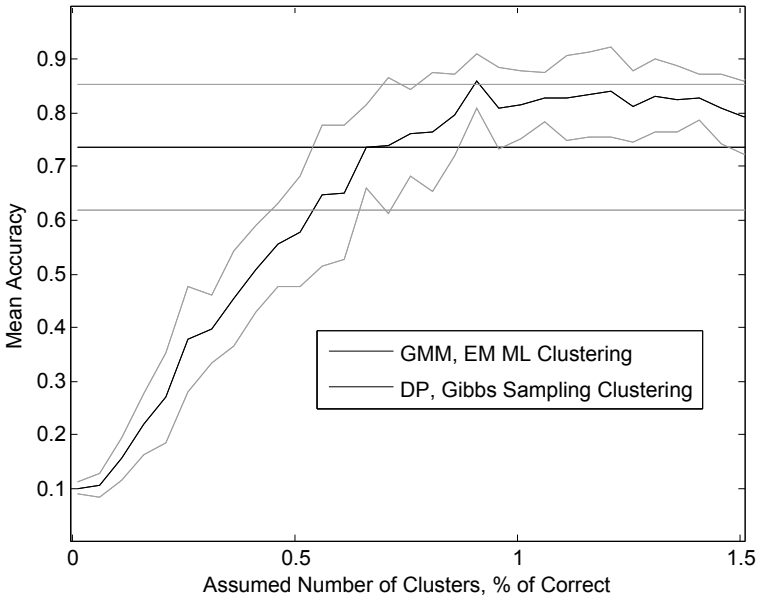


Figure 15. Mean (in black) and 1 standard deviation away from the mean (in grey) of the classification accuracy (higher is better) using a) the parametric Gaussian mixture model with Expectation Maximization and maximum likelihood clustering, assuming a number of clusters between 1% and 150% of the actual number of clusters, and b) the nonparametric Dirichlet process mixture model with Gibbs sampling clustering. For case a), this accuracy was calculated at each 5% increment over 20 trials of 1000 classification attempts, each trial using a random sample of the HBP agent generation model. For case b), the accuracy was calculated over 45 trials of 1000 classification attempts, each trial using a random sample of the HBP agent generation model.

accuracy is independent of it.

Given the estimated classifications of the agents, the team used UCB and a sequential greedy procedure to concurrently allocate and learn the expected reward for the surveillance tasks involving each of the 10 classes. This resulted in a mean overall team regret over 100 trials shown in Figure 16, calculated as the difference in score between the case where rewards and classifications were known and where rewards and classifications were unknown and had to be learned. The regret is shown for a number parametric Gaussian mixture models of behavior observations (as before, where the assumed number of behaviors exhibited in the training data was 25%, 50%, 75% and 100% of the actual number of behaviors), for the Dirichlet process mixture model, and for a case where the agents did not try to classify exogenous agents and instead randomly guessed the classification of each exogenous agent. This test once again illustrates the effectiveness of using BNPs instead of their parametric counterparts when an appropriate choice of model complexity is unknown a priori. It may be noted that at the beginning of the simulation, each of the cases performed equally well on average; this is because UCB is a reinforcement learning algorithm, and therefore made mistakes early on as it learned what the expected rewards were for conducting surveillance on the 10 exogenous agent classes.

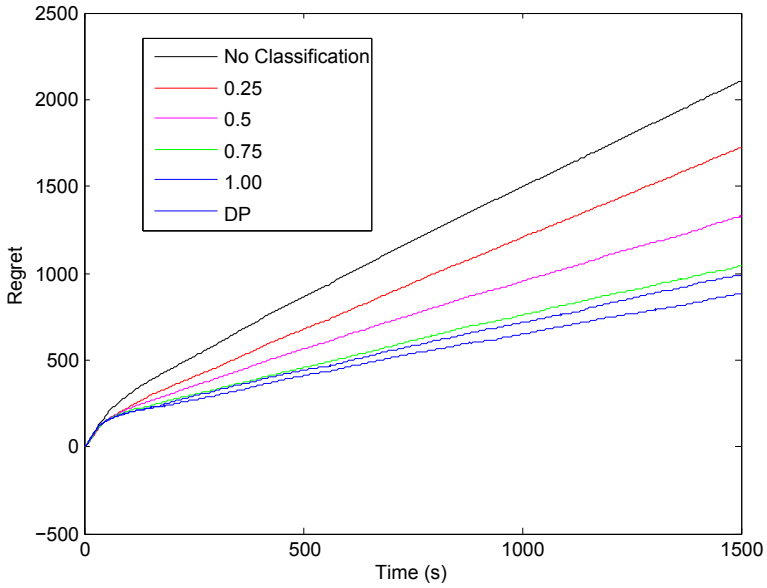


Figure 16. Means of regrets (lower is better) versus time over 100 trials for the task allocation scenario, where a) a parametric Gaussian mixture model with an assumed number of behaviors was used for clustering behavior observations in the training data, b) a Dirichlet process Gaussian mixture model was used for the same purpose, and c) no classification or behavior clustering was done, and the classifications of agents was randomly guessed. For case a), the assumed number of behaviors was 25% (red), 50% (magenta), 75% (green), and 100% (blue).

IV. Conclusion

This paper investigates the use of Bayesian nonparametric models (BNPs) for multi-agent planning. Through two illustrative example problems in target surveillance and task allocation, we have demonstrated that the capability of BNPs to adapt their complexity based on observed data provides two advantages over their parametric counterparts when used in conjunction with planning algorithms. The first is that the increased fidelity leads directly to improvements in the performance of the planning system due to its enhanced predictive capability; and the second is that they enable the planning algorithms to identify and explicitly deal with exogenous agents exhibiting anomalous behaviors, either by directing agents to learn about those anomalous behaviors to reduce the model uncertainty in them, or by using previous experience with similar exogenous agents to infer an appropriate response.

Although promising, there are technical challenges associated with the use of BNPs within a planning system. Many current inference methods, such as Markov chain Monte Carlo methods, are computationally intensive and are not incremental. If a nonparametric model is built using an initial set of data, each time new data is observed these algorithms use the entire set of observed data to build an updated model. These algorithms are not meant for real-time data collection situations commonly found in planning and control problems, as computation time increases with the amount of data. Therefore, a current research goal is to develop incremental inference algorithms for BNPs.

Further, if a computationally feasible online inference algorithm were available, it would be advantageous to have a planning algorithm which could adapt its response alongside the continually

evolving model without having to recompute entire plans, or even possibly adapt its set of known responses and actions to better accomodate new and changing circumstances. Therefore, another current research goal is to develop a flexible planner that can adjust its planning strategies within a framework that allows incremental adjustments to plans.

Acknowledgments

This research was supported by ONR MURI Grant N000141110688. Trevor Campbell is also supported in part by a Natural Sciences and Engineering Research Council of Canada PGS-M grant. The authors thank Daniel Levine at MIT for his insights on the example problems considered in this work and the rest of the MURI team for the illuminating discussions.

References

[1] Johnson, E. and Kannan, S., "Adaptive Trajectory Control for Autonomous Helicopters," *Journal of Guidance Control and Dynamics*, Vol. 28, No. 3, May 2005, pp. 524–538.

[2] Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129.

[3] Ioannou, P. A. and Sun, J., *Robust Adaptive Control*, Prentice-Hall, Upper Saddle River, 1996.

[4] Calise, A. J. and Rysdyk, R. T., "Nonlinear adaptive flight control using neural networks," *IEEE Control Systems Magzine*, Vol. 18, No. 6, December 1998, pp. 14–25.

[5] Haykin, S., *Neural Networks a Comprehensive Foundation*, Prentice Hall, USA, Upper Saddle River, 2nd ed., 1998.

[6] Alpaydin, E., *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, The MIT Press, 2004.

[7] Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 1st ed., 2007.

[8] Korb, K. and Nicholson, A., *Bayesian Artificial Intelligence*, CRC Press, Boca-Raton, FL, 2nd ed., 2011.

[9] Fox, E., Choi, D., and Willsky, A., "Nonparametric Bayesian Methods for Large Scale Multi-Target Tracking," *Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC '06)*, Nov 2006, pp. 2009 –2013.

[10] Fox, E., Sudderth, E., and Willsky, A., "Hierarchical Dirichlet processes for tracking maneuvering targets," *Information Fusion, 2007 10th International Conference on*, IEEE, 2007, pp. 1–8.

[11] Fox, E., Sudderth, E., Jordan, M., and Willsky, A., "Nonparametric Bayesian learning of switching linear dynamical systems," *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[12] Teh, Y. W. and Jordan, M. I., "Hierarchical Bayesian nonparametric models with applications," *Bayesian Nonparametrics: Principles and Practice*, edited by N. Hjort, C. Holmes, P. Mueller, and S. Walker, Cambridge University Press, Cambridge, UK, 2010.

[13] Jordan, M. I., "Hierarchical models, nested models and completely random measures," *Frontiers of Statistical Decision Making and Bayesian Analysis: In Honor of James O. Berger*, edited by M.-H. Chen, D. Dey, P. Mueller, D. Sun, and K. Ye, Springer-Verlag, New York, 2010.

[14] Xu, Z., Tresp, V., Yu, K., and Kriegel, H. P., "Infinite Hidden Relational Models," *Uncertainty in Artificial Intelligence (UAI)*, 2006.

[15] Antoniak, C., "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems," *The Annals of Statistics*, , No. 2, 1974, pp. 1152–1174.

[16] Hjort, N. L., "Nonparametric Bayes Estimators Based on Beta Processes in Models for Life History Data," *The Annals of Statistics*, Vol. 18, No. 3, 1990, pp. 1259–1294.

[17] Thibaux, R. and Jordan, M. I., "Hierarchical beta processes and the Indian buffet process," *Proc. International*

Conference on Artificial Intelligence and Statistics, 2007.

[18] Zhou, M., Chen, H., Paisley, J., Ren, L., Sapiro, G., and Carin, L., "Non-Parametric Bayesian Dictionary Learning for Sparse Image Representations," *Proc. Neural Information Processing Systems*, 2009.

[19] Rai, P. and III, H. D., "Nonparametric Bayesian Sparse Hierarchical Factor Modeling and Regression," *Proc. Neural Information Processing Systems*, 2008.

[20] Rasmussen, C. and Williams, C., *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006.

[21] Fox, E. B., *Bayesian Nonparametric Learning of Complex Dynamical Phenomena*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge MA, December 2009.

[22] Ko, J., Klein, D. J., Fox, D., and Hähnel, D., "GP-UKF: Unscented kalman filters with Gaussian process prediction and observation models," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 1901–1907.

[23] Murray-Smith, R. and Sbarbaro, D., "Nonlinear adaptive control using non-parametric Gaussian Process prior models," *15th Triennial World Congress of the International Federation of Automatic Control*, International Federation of Automatic Control (IFAC), 2002.

[24] Murray-Smith, R., Sbarbaro, D., Rasmussen, C., and Girard, A., "Adaptive, cautious, predictive control with Gaussian process priors," *13th IFAC Symposium on System Identification*, edited by P. V. de Hof, B. Wahlberg, and S. Weiland, IFAC proceedings volumes, Elsevier Science, 2003, pp. 1195–1200.

[25] Engel, Y., Mannor, S., and Meir, R., "Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning." *International Conference on Machine Learning (ICML)*, edited by T. Fawcett and N. Mishra, AAAI Press, 2003, pp. 154–161.

[26] Rasmussen, C. and Kuss, M., "Gaussian processes in reinforcement learning," *Advances in Neural Information Processing Systems*, Vol. 16, 2004, pp. 751–759.

[27] Zhu, Q., "Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation," *Robotics and Automation, IEEE Transactions on*, Vol. 7, No. 3, jun 1991, pp. 390 –397.

[28] Miyajima, C., Nishiwaki, Y., Ozawa, K., Wakita, T., Itou, K., Takeda, K., and Itakura, F., "Driver Modeling Based on Driving Behavior and Its Evaluation in Driver Identification," *Proceedings of the IEEE*, Vol. 95, No. 2, 2007, pp. 427 –437.

[29] How, J. P., Fraser, C., Kulling, K. C., Bertuccelli, L. F., Toupet, O., Brunet, L., Bachrach, A., and Roy, N., "Increasing autonomy of UAVs," *Robotics and Automation Magazine, IEEE*, Vol. 16, No. 2, June 2009, pp. 43–51.

[30] Rasmussen, S. and Shima, T., *UAV Cooperative Decision and Control: Challenges and Practical Approaches*, Society for Industrial Mathematics, 2009.

[31] Fox, E., Sudderth, E., Jordan, M., and Willsky, A., "Bayesian Nonparametric Methods for Learning Markov Switching Processes," *IEEE Signal Processing Magazine*, Vol. 27, No. 6, 2010, pp. 43 –54.

[32] Joseph, J. M., *Nonparametric Bayesian Behavior Modeling*, Master's thesis, Massachusetts Institute of Technology, 2008.

[33] Joseph, J., Doshi-Velez, F., Huang, A. S., and Roy, N., "A Bayesian Nonparametric Approach to Modeling Motion Patterns," *Autonomous Robots*, Vol. 31, No. 4, 2011, pp. 383–400.

[34] Ferguson, T., "A Bayesian Analysis of Some Nonparametric Problems," *The Annals of Statistics*, Vol. 1, No. 2, 1973, pp. 209–230.

[35] Rasmussen, C., "The infinite Gaussian mixture model," *Advances in neural information processing systems*, Vol. 12, 2000, pp. 554–560.

[36] Sethuraman, J., "A Constructive Definition of Dirichlet Priors," *Statistia Sinica*, Vol. 4, 1994, pp. 639–650.

[37] Neal, R., "Markov chain sampling methods for Dirichlet process mixture models," *Journal of computational and graphical statistics*, 2000, pp. 249–265.

[38] Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M., "Hierarchical Dirichlet Processes," *Journal of the American Statistical Association*, Vol. 101, No. 476, 2006, pp. 1566–1581.

[39] Teh, Y. W. and Jordan, M., *Bayesian Nonparametrics in Practice*, chap. Hierarchical Bayesian Nonparametric

Models with Applications, 2009.

[40] Blei, D. and Jordan, M. I., "Variational Inference for Dirichlet Process Mixtures," *Bayesian Analysis*, Vol. 1, No. 1, 2006, pp. 121–144.

[41] Teh, Y. W., *Dirichlet Process*, Encyclopedia of Machine Learning, Springer-Verlag New York Inc, 2011, pp. 280–290.

[42] Baum, L. E., Petrie, T., Soules, G., and Weiss, N., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, Vol. 41, No. 1, 1970, pp. 164–171.

[43] Johnson, L. B., Ponda, S. S., Choi, H.-L., and How, J. P., "Asynchronous Decentralized Task Allocation for Dynamic Environments," *Proceedings of the AIAA Infotech@Aerospace Conference*, St. Louis, MO, March 2011.

[44] Choi, H.-L., Brunet, L., and How, J. P., "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, Vol. 25, No. 4, August 2009, pp. 912–926.

[45] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*, The MIT Press, 1998.

[46] Auer, P., Cesa-Bianchi, N., and Fischer, P., "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, Vol. 47, 2002, pp. 235–256.