# FAST INITIALIZATION OF PARTICLE FILTERS USING A MODIFIED METROPOLIS-HASTINGS ALGORITHM: MODE-HUNGRY APPROACH

*Volkan Cevher and James H. McClellan*

Georgia Institute of Technology
Atlanta, GA 30332-0250

## ABSTRACT

As a recursive algorithm, the particle filter requires initial samples to track a state vector. These initial samples must be generated from the received data and usually obey a complicated distribution. The Metropolis-Hastings (M-H) algorithm is used for sampling from intractable multivariate target distributions and is well suited for the initialization problem. Asymptotically, the M-H scheme creates samples drawn from the exact distribution. For the particle filter to track the state, the initial samples need to cover only the region around its current state. This region is marked by the presence of modes. Since the particle filter only needs samples around the mode, we modify the M-H algorithm to generate samples distributed around the modes of the target posterior. By simulations, we show that this ”mode hungry” algorithm converges an order of magnitude faster than the original M-H scheme for both unimodal and multi-modal distributions.

## 1. INTRODUCTION

Particle filters use Bayes' rule to update an arbitrary posterior recursively as the observations arrive. In the filter mechanics, posteriors are represented by particles that are discrete realizations distributed according to the underlying distribution (either directly or by weighting). Hence, the particle filter can estimate any statistics of the posterior, and the estimation accuracy can be improved up to the theoretic bounds by increasing the number of particles. Particle filters are becoming widely popular in signal processing due to (i) the advances in the current computer systems (ii) their easy implementation [1, 2].

As a recursive algorithm, the particle filter needs initialization (or initial samples) to track a state vector as the observations arrive in sequence. The algorithm's estimation performance not only depends on how it is constructed around the data models, but also on its initial samples. Hence, it is necessary to generate good initial samples from the data itself. If the initial samples do not sufficiently cover the state space, important features of the distribution (such as multi-modality) can be missed. This can significantly degrade the estimation performance of the filter.

Monté Carlo Markov chain (MCMC) simulation techniques offer attractive solutions to this type of sampling problem. In MCMC terms, the initialization problem becomes the estimation of a posterior density $\pi(x)$ by simulating a Markov chain $x^{(1)}$, $x^{(2)}$, $\ldots, x^{(t)}$ whose stationary distribution is the target posterior $\pi$ as $t \to \infty$. Among the MCMC sampling techniques, the Metropolis-Hastings (M-H) algorithm is used in this paper to demonstrate the concepts and the results.

Prepared through collaborative participation in the Advanced Sensors Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-02-0008.

The M-H scheme [3,4] provides a basis from which other well-known sampling algorithms such as the Acceptance-Rejection (AR) and Gibbs sampling can be derived as special cases [5, 6]. The algorithm assumes that it is possible to assign probabilities from the distribution to a given realization $x$. These probabilities need not be exact; they can also be given up to a proportionality constant. Note that calculating probabilities given the particles is different (and much easier) than generating particles directly from the, possibly intractable and multivariate, distribution itself.

At each iteration of the algorithm, a candidate generating density $q(x, y) : x \to y$ is used to propose moves (or *jumps*) within the state space. These moves are accepted or rejected with probability $\alpha(x, y)$ so that the following reversibility condition is satisfied:

$$\pi(x)q(x, y)\alpha(x, y) = \pi(y)q(y, x) \quad (1)$$

The reversibility condition is also known as *detailed balance*, *microscopic reversibility*, or *time reversibility* and is a necessary condition for the chain to satisfy if $\pi$ is the stationary distribution [5].

The convergence analysis of the M-H algorithm is an open problem. Many methods have been proposed to detect the convergence of this scheme [7–9]. Although these convergence detection methods are very useful, the M-H algorithm usually takes a large number of iterations to converge, and hence it is usually not considered for real-time applications in its original form. To make the algorithm more attractive, there has been some effort on making the algorithm more efficient. Two fundamentally different approaches are used to speed up the M-H scheme: One approach adaptively alters the jumping rules (i.e., the jump size) defined by the candidate generating density. The other approach uses a mode-search algorithm (usually a grid search) to locate the modes and samples from a mixture of suitable distributions located at these modes to generate a prior for the algorithm. The first method slightly improves the convergence speed but not enough for a real-time initialization. The latter method incurs a high computational cost to begin and is therefore not suitable for our purposes.

In this paper, we propose a new approach called Mode-Hungry M-H (MHMH) to speed up the original scheme for the initialization of the particle filters. The new method modifies the jumping rules according to how the state of the Markov chain is distributed and concentrates the particles on the high probability regions. The algorithm quickly converges on the multi-modal regions of the posterior correctly. The estimated posterior is an approximation to the true posterior around the mode. However, it can be shown that any discrepancies between the estimated and true posteriors can be handled by the particle filter's built-in weighting structure [1, 2]. This property of the particle filters and the convergence speed makes this algorithm very attractive for particle filtering applications.

Organization of the paper is as follows. Section 2 provides the background for the M-H algorithm. Section 3 describes the modification of the basic algorithm and gives pseudo-code for the MHMH algorithm. Computer simulations are given in section 4.

## 2. BACKGROUND

The Metropolis-Hastings scheme [5] is depicted in Figure 1. The objective of the algorithm is to distribute the particles (discrete state samples) according to the target distribution $\pi$ shown on top. Hence, the algorithm recursively redistributes its states around so that, asymptotically, the resulting Markov chain is distributed according to the target distribution. In the figure, the Markov chain at iteration $t$ is represented by $x^{(t)}$. The new chain candidates $y$ are generated by the proposal function $q(x, y)$, which is usually the spherically symmetric random walk:

$$q(x, y) = q(|x - y|) \propto \exp\left[\frac{(x-y)^2}{2\sigma^2}\right] \tag{2}$$

Once the new candidates are generated, the algorithm accepts the moves or keeps the current state according to the acceptance ratio $\alpha(x, y)$ derived from the reversibility condition:

$$\alpha(x, y) = \min\left[\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}, 1\right] \tag{3}$$

In Figure 1, the acceptance ratios are represented by the height of the boxes for each candidate. To accept or reject the new candidate, a random number generator is used to generate uniform random numbers in $(0, 1)$, $u \sim \mathcal{U}(0, 1)$, represented by the black dots in the figure. If $u$ is less than the acceptance ratio for the specific particle, the move is accepted, otherwise, it is rejected. Also, the *acceptance rate* at iteration $t$ is defined to be the number of accepted moves divided by the chain size. Visually, it is the number of arrows in the last stage in Figure 1 divided by the number of particles. Finally, the chain moves to $x^{(t+1)}$ and the scheme is repeated.

Note that the candidate generating (or proposal) function has a significant impact on the efficiency of the algorithm. It should be constructed so that the generated candidates display most of the structural dependence between the different dimensions. $\sigma$ in (2) is defined as the jump size and is the other important variable affecting the algorithm speed. If it is too low, the algorithm takes a longer time to converge since the chain moves very slowly along the target distribution. On the other hand, if it is too high, the algorithm mostly rejects the new candidates and stays frozen. The current MCMC literature concentrates on these two important components of the algorithm for its efficiency [9].

## 3. MODE-HUNGRY APPROACH

The M-H scheme is naturally very slow since it tries to explore the whole parameter space. Hence, any relevant information should be used to increase the convergence speed. The proposal functions should incorporate any dimensional dependence, and the jump sizes

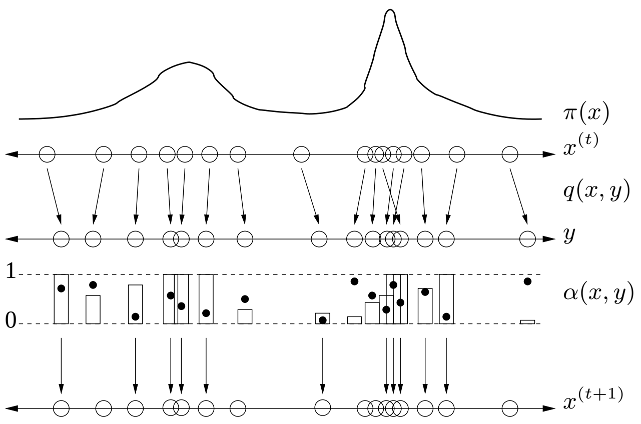

**Fig. 1**. The Metropolis-Hastings scheme is demonstrated. Each circle represents a sample from the chain in the respective state space. The algorithm uses its current state to generate new candidates for its next state using a candidate generating function $q$. The new candidates are accepted or rejected in a way that the Markov chain asymptotically converges to the target posterior $\pi$.

should be modified by monitoring the acceptance rate of the algorithm. In this section, we will modify the M-H algorithm to concentrate around the modes of the target posterior. We call this modification the *Mode-Hungry* approach. Interestingly, note that the adaptive modification of the simulation gives rise to a non-Markov chain since the transition probabilities depends on the previous iteration results. However, the performance of the results themselves provide a justification for their use.

In most cases, the target distribution is multi-modal with a significant number of small modes and a few large modes. The small modes can be due to (i) the nature of the problem, (ii) modelling errors, and (iii) an approximation of the actual target density $\pi$ used in the simulation. In most particle tracking problems, the large modes are of interest since the small modes get rejected by resampling (or weighting) after a few iterations of the filter. Moreover, the small modes tend to trap the chain and increase the convergence time. It takes the M-H scheme a notoriously high number of iterations to break the particles from these small modes and concentrate them on the higher ones [10]. Hence, when generating the target density $\pi$, there is a trade-off between how closely it can be represented versus how long the initialization takes to converge.

With this observation, it is suggested that if the objective of the initialization is to cover only the high probability modes, the particles around the small modes should be redistributed accordingly during the M-H iterations. This, in turn, will result in a faster coverage of the high modes and improve the convergence speed. Hence, at time $t$, the chain $x^{(t)}$ consisting of $N$ particles is separated into two partitions after being sorted with respect to likelihoods. Denote $P_1$ as the set corresponding to the $N - M_t$ high probability particles, and $P_2$ as the $M_t$ low probability particles. During the algorithm execution, a rule needs to be determined to distribute the particles from $P_2$ around $P_1$ so that the algorithm does not get stuck at low probability modes. We propose to use randomly chosen particles from $P_1$ as the candidates $y$ for the particles from $P_2$ and accept their moves with probability 1.

Figure 2 illustrates this Mode-Hungry approach. The target particles from $P_1$ can be chosen in a variety of ways, two of which

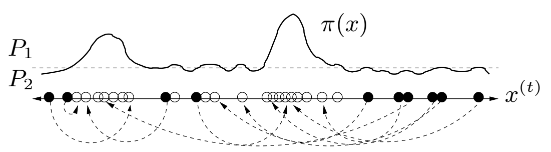

**Fig. 2**. Particles corresponding to the partition $P_2$ are shown in black. In the figure, the particles in $P_2$ are distributed over the states defined by $P_1$ uniformly.

are considered here: proportional to their probabilities and uniformly. If we impose that the cross-jumps from $P_2$ to $P_1$ should be such that particles around one mode should not be quickly consumed by another similar size mode, then it is not prudent to pick particles from $P_1$ proportional to their probabilities. In this case, the highest mode quickly accumulates all the particles and the algorithm results in a single mode distribution. Hence, this rule is recommended if the target posterior is known to be unimodal.

If the particles from $P_2$ are uniformly distributed over $P_1$, the algorithm quickly finds the multiple modes; however, the distributions around the modes can be skewed if they are far apart from each other as illustrated in the next section. This is still acceptable, since any imbalance of the particle distribution will be automatically corrected by the particle filter's weighting mechanism. Note that a good local distribution around the mode is still required to prevent sample degeneracy [1]. The particle filter attaches a weight $w_i$ to each particle $x_i$ coming from the M-H scheme. Then, it estimates an expected value $E[.]$ of the posterior as

$$E[f(x)] = \sum_i w_i f(x_i) \tag{4}$$

assuming that the weights are normalized. Hence, as long as the region around the modes are covered correctly, the weights can correct the imbalances introduced by the initialization since they also depend on the target density $\pi$. It should be noted that the original M-H scheme also has similar issues in multi-modal problems; however, the proposed cross-jumps enable the algorithm to cover the high modes faster. The frequency of these jumps can be determined by monitoring the acceptance rates or how tight the particles distribute themselves around the modes; however, a fixed period ($T_{jump}$) also seem to work. The pseudo-code gives the details of the proposed algorithm.

## 4. EXAMPLES

In this section, our objectives are (i) to demonstrate that the proposed method results in faster convergence, and (ii) to show that the algorithm can handle multi-modal distributions. In the simulations, the initial chain values are generated from a uniform distribution $x_i \sim \mathcal{U}(-50, 50)$ in each dimension.

### 4.1. Single Mode Gaussian Example

$N = 1000$ particles are simulated to obtain a three dimensional Gaussian distribution: $\pi(\mathbf{x}) \sim \mathcal{N}(\mu, \mathbf{\Sigma})$. The mean and covariance of the distribution are the following:

$$\mu = \begin{bmatrix} 20 \\ -15 \\ 45 \end{bmatrix} \quad \mathbf{\Sigma} = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 2 & -.5 \\ 0.5 & -.5 & 0.5 \end{bmatrix} \tag{5}$$

**Mode-Hungry Metropolis-Hastings**

```
• At time $t$, decide if cross-jumping is needed for $x^{(t)}$, i.e., every $T_{jump}$ iterations
• If not, then for each particle $x_i$, use the M-H scheme:
    – generate a candidate $y_i$ using $q(x_i, y_i)$
    – calculate the acceptance ratio
          $\alpha(x_i, y_i) = \min\left[\frac{\pi(y_i)q(y_i, x_i)}{\pi(x_i)q(x_i, y_i)}, 1\right]$
    – sample $u \sim \mathcal{U}(0, 1)$
    – if $u \le \alpha(x_i, y_i)$, set $x_i^{(t+1)} = y_i$, else, $x_i^{(t+1)} = x_i^{(t)}$.
• If yes, then use the Mode-Hungry scheme:
    – determine a subpartition of size $M_t < N$
    – order the current particles according to their probabilities in descending order: $x_i \rightarrow x_j^*$ where $x^*$ is the ordered particle set
    – generate candidates $y^*(1)$ for $x^*(1) = \{x_j^* | j : j = 1, 2, \ldots, N - M_t\}$ using $q(., .)$
    – calculate the acceptance ratio $\alpha(x^*(1), y^*(1))$ and set $x_j^{(t+1)}$ to $x_j^*(1)$ or $y_j^*(1)$ accordingly for $j = 1, 2, \ldots, N - M_t$
    – distribute $M_t$ candidates $y^*(2)$ from $x^*(1)$ uniformly
    – set $x_j^{(t+1)}$ to $y^*(2)$ for $j = N - M_t + 1, \ldots, N$
                                                                  ■
```

For the Mode-Hungry M-H (MHMH) algorithm, $M_t$ is chosen to be 333. As the proposal function $q$, an independent random walk is used:

$$q(\mathbf{x}, \mathbf{y}) \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{3}\frac{(x_i - y_i)^2}{\sigma_i^2}\right] \tag{6}$$

with $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = 6$. The particles belonging to $P_2$ are distributed over $P_1$ at every other iteration of MHMH ($T_{jump} = 2$) till the variances of the distribution come close to the search variances of the proposal function. Fig. 3 displays the estimation results of both algorithms. In the figure, it should be noted that MHMH achieves convergence around the modes much faster than the original M-H scheme. Moreover, both algorithms take approximately $30 - 50$ iterations to achieve the correlation structure in (5) after convergence around the modes.

### 4.2. Multi-Modal Example

In order to demonstrate efficacy of the proposed modifications on the multi-modal distributions, the following target density is used:

$$\begin{aligned} \pi(\mathbf{x}) \propto & \Big(e^{-(x_1-20)^2/2} + \frac{1}{2}e^{-(x+20)^2/2}\Big) \\ & \times \Big(\frac{1}{2}e^{-(x_1-30)^2/10} + e^{-(x+45)^2/10}\Big) \end{aligned} \tag{7}$$

The same set of cross-jumping rules are used as in the previous example ($N = 1000$, $M_t = 333$, and $T_{jump} = 2$). In this example, the random walk proposal is used with $\sigma_1^2 = 5$ and $\sigma_2^2 = 5$ as the respective variances in $x_1$ and $x_2$ directions. The resulting

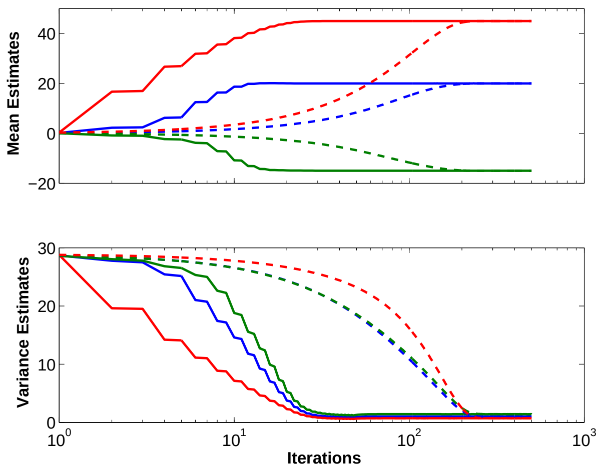

**Fig. 3**. MHMH (solid) and M-H (dashed) algorithms are run 100 times and the averaged estimates are displayed. MHMH is about an order of magnitude faster in this case. The jumps in the MHMH curves are attributed to the redistribution of the low probability particles over the high ones.

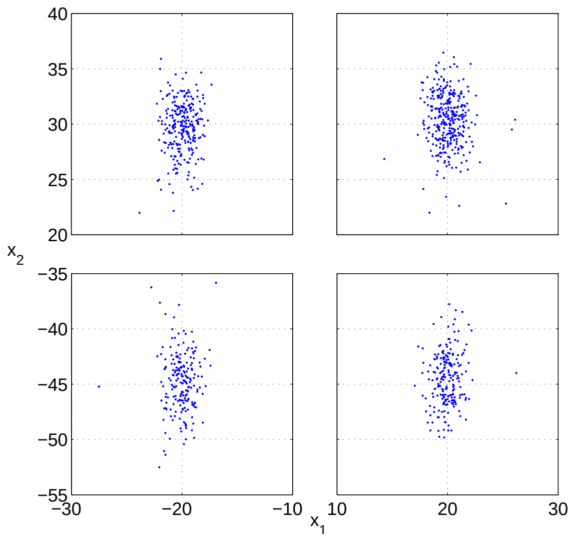

**Fig. 4**. Output distribution of MHMH after 30 iterations. A similar distribution can be obtained by the M-H scheme using approximately 100 iterations.

distribution after 30 iterations of MHMH is shown in Fig. 4.

It should be noted in Fig. 4 that the local distributions of the particles around the modes are correct. However, the number of particles attracted to each mode does not obey the global distribution (two of the modes are higher than the others hence should attract more particles.) This is an issue also common to the M-H scheme where it takes a high number of iterations for the M-H algorithm to break particles from different modes to correctly concentrate them on different modes. This output is still suitable for a particle filter because of the correct local distributions. The inherent weighting of the particles would take care of any discrepancies in the global distribution.

## 5. CONCLUSIONS

In this paper, the M-H algorithm is modified to achieve faster convergence for the initialization of particle filters. The modification depends on the important fact that the particle filter needs only the distribution around the current state estimates. The MHMH algorithm is shown to converge faster than the M-H scheme and is well suited for particle filtering initializations. The effect of the state dimension on the convergence speed of the proposed MHMH algorithm is being studied. In the simulation examples, stopping conditions for the cross jumps depends on monitoring the distribution and seeing if the particles have converged around the modes. The stopping condition is extremely important for MHMH when the target distribution is multi-modal. If the algorithm is allowed to run further, the highest mode naturally consumes the particles around the other modes. This issue can be handled by adaptively changing $M_t$ and $T_{jump}$ and several strategies are being developed.

## 6. REFERENCES

[1] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[2] A. Doucet, N. Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, 2001.

[3] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, “Equations of state calculations by fast computing machines,” *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[4] W.K. Hastings, “Monte Carlo sampling methods using Markov chains and their applications,” *Biometrika*, vol. 57, pp. 97–109, 1970.

[5] S. Chib and E. Greenberg, “Understanding the Metropolis-Hastings algorithm,” *The American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[6] L. Tierney, “Markov chains for exploring posterior distributions,” *The Annals of Statistics*, vol. 22, no. 4, pp. 1701–1728, 1994.

[7] A. Gelman and D.B. Rubin, “Inference from iterative simulation using multiple using multiple sequences,” *Statistical Science*, vol. 7, pp. 457–511, 1992.

[8] S.P. Brooks and A. Gelman, “General methods for monitoring convergence of iterative simulations,” *Journal of Computational and Graphical Statistics*, vol. 7, pp. 434–455, 1998.

[9] A. Gelman, G.O. Roberts, and W.R. Gilks, “Efficient Metropolis jumping rules,” *Bayesian Statistics*, vol. 5, 1996.

[10] P. Clifford et. al., “Discussion on the meeting on the Gibbs sampler and other Markov chain Monte Carlo methods,” *Journal of Royal Statistical Society*, vol. 55, no. 1, pp. 53–102, 1993.