Particle Filter Optimization: A Brief Introduction

Bin Liu[†,♯(✉)], Shi Cheng[‡], Yuhui Shi[§]

[†]School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, 210023 China
[♯] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023 China
bins@ieee.org
[‡]School of Computer Science, Shaanxi Normal University, Xi'an, 710062 China
cheng@snnu.edu.cn
[§]Department of Electrical & Electronic Engineering, Xi'an Jiaotong-Liverpool University , Suzhou, 215123 China
Yuhui.Shi@xjtlu.edu.cn

Abstract. In this paper, we provide a brief introduction to particle filter optimization (PFO). The particle filter (PF) theory has revolutionized probabilistic state filtering for dynamic systems, while the PFO algorithms, which are developed within the PF framework, have not attracted enough attention from the community of optimization. The purpose of this paper is threefold. First, it aims to provide a succinct introduction of the PF theory which forms the theoretical foundation for all PFO algorithms. Second, it reviews PFO algorithms under the umbrella of the PF theory. Lastly, it discusses promising research directions on the interface of PF methods and swarm intelligence techniques.

Keywords: Particle filter, Optimization, Swarm intelligence

1 Introduction

Particle filters (PFs), also known as Sequential Monte Carlo methods [1], consist of a set of generic type Monte Carlo sampling algorithms to solve the state filtering problem [2, 3]. The objective of state filtering is to compute the posterior distributions of the states of the dynamic system, given some noisy and/or partial observations. The PF theory provides a well-established methodology for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions [4–6]. Under mild assumptions, the convergence of the PF methodology has been proved [7, 8]. With the notable exception of linear-Gaussian signal-observation models, the PF theory has become the dominated approach to solve the state filtering problem in dynamic systems [9–11]. In the last two decades, applications of the PF theory have expanded to diverse fields such as object tracking [2], navigation and guidance

[12, 13], robotics [14], fault diagnosis [15], signal processing in digital communication systems [16], financial time series [17], biological engineering [18] and optimization [19, 20].

As a generic type mutation-selection sampling approach [21], the PF theory based algorithm uses a set of particles (also called samples) to represent the posterior distribution given some noisy and/or partial observations. The working mechanism of the PF algorithm allows the state-space model to be nonlinear and the initial state and noise distributions to take any form required. Provided that the particles are properly placed, weighted and propagated, the posterior distributions can be estimated sequentially over time. Despite many advantages, the PF theory is usually faced with a fatal problem, termed sample impoverishment, due to the suboptimal sampling mechanism used [1, 22]. The population based swarm intelligence techniques consist in one branch of approaches to reduce the effect of sample impoverishment. The basic principle is to treat the particles and the likelihood function involved in the PF theory as the individuals and the fitness function involved in swarm intelligence methods, respectively. Based on the above principle, a number of swarm intelligence algorithms, e.g., the particle swarm optimization (PSO) [23, 24], the genetic algorithm [25, 26], artificial fish swarm [27], have been used to improve the PF methods in reducing the effect of sample impoverishment.

Recently a class of optimization algorithms, termed particle filter optimization (PFO), has been proposed [28–31]. Although the PF theory has revolutionized probabilistic state filtering for dynamic systems, the PFO algorithms have not attracted enough attention from the community of optimization. The focus of this paper just lies in the introduction of the PF theory and the PFO algorithms.

The remainder of the paper is organized as follows. In Section 2, we present the basic components of the PF theory. In Section 3, we make a brief overview on the PFO algorithms under the umbrella of the PF theory. In Section 4, we compare two versions of PFO algorithms with a benchmark PSO algorithm in performance. Lastly in Section 5, we conclude the paper and make several remarks on the interface between the PF theory and the swarm intelligence techniques.

2 Particle Filter Theory

Here we give a brief overview on the PF theory, which forms the theoretical foundation for the PFO algorithms discussed in Section 3. Refer to [3, 32] on further details on the PF theory.

The PF theory addresses the problem how to sample from a set of target probabilistic density functions (pdf) $\{\pi_k\}_{k\in[1,\ldots,K]}$ in a sequential manner, namely first sample from π_1, then from π_2 and so on. In the context of sequential Bayesian state inference, π_k could be the posterior distribution of the state x_k given the data collected until the kth time step, i.e., $\pi_k = p(x_k|y_{1:k})$, where y denotes the collected data and $y_{1:k} \triangleq \{y_1, \ldots, y_k\}$. At a given time

k, the basic idea is to obtain a large collection of N weighted random particles $\{w_k^i, x_k^i\}_{i=1:N}, w_k^i > 0, \sum_{i=1}^N w_k^i = 1$, whose empirical distribution converges asymptotically ($N \to \infty$) to π_k, i.e., for any π_k-integrable function ϕ ($\phi(x) \in \mathbb{R}$):

$$\sum_{i=1}^{N} w_k^i \phi(x_k^i) \to \mathbb{E}_{\pi_k}(\phi) \qquad \text{almost surely,} \tag{1}$$

where '$\to$' denotes 'converges to' and

$$\mathbb{E}_{\pi_k}(\phi) = \int_X \phi(x)\pi_k(x)dx. \tag{2}$$

In what follows, any operation involving the superscript i must be understood as performed for $i = 1 : N$, where N is the total number of particles. Assume that we have at hand a weighted sample set $\{w_k^i, x_k^i\}_{i=1:N}$, which satisfies Eqn. (1), a combination of sequential importance sampling (SIS) and resampling ideas [33] is used in the PF theory to carry forward these particles over time, yielding another weighted sample set $\{w_{k+1}^i, x_{k+1}^i\}_{i=1:N}$, which can provide a Monte Carlo approximation to π_{k+1} as follows,

$$\pi_{k+1} \simeq \sum_{i=1}^{N} w_{k+1}^i \delta(x_{k+1} - x_{k+1}^i), \tag{3}$$

where δ denotes the delta-mass function.

According to the principle of importance sampling [33], we first draw random particles x_{k+1}^i from a proposal pdf $q(x_{k+1}|y_{1:k+1})$, then the weights in Eqn. (3) are defined to be

$$w_{k+1}^i \propto \frac{\pi_{k+1}(x_{k+1}^i)}{q(x_{k+1}^i|y_{1:k+1})}. \tag{4}$$

To obtain a sequential solution, the proposal pdf is chosen to factorize such that [3]

$$q(x_{k+1}|y_{1:k+1}) = q(x_{k+1}|x_k, y_{1:k+1})q(x_k|y_{1:k}). \tag{5}$$

Note that $\pi_{k+1}(x_{k+1})$ can be derived based on the Bayesian theorem as follows

$$\begin{aligned} \pi_{k+1}(x_{k+1}) &= \frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|x_k)}{p(y_{k+1}|y_{1:k})}\pi_k(x_k) \\ &\propto p(y_{k+1}|x_{k+1})p(x_{k+1}|x_k)\pi_k(x_k), \end{aligned} \tag{6}$$

where $p(x_{k+1}|x_k)$ is determined by the evolution law of the state sequence $x_1, x_2, \ldots$, and $p(y_{k+1}|x_{k+1})$ denotes the likelihood function. By substituting Eqns. (5) and (6) into (4), we have

$$\begin{aligned} w_{k+1}^i &\propto \frac{p(y_{k+1}|x_{k+1}^i)p(x_{k+1}^i|x_k^i)\pi_k(x_k^i)}{q(x_{k+1}^i|x_k^i, y_{1:k+1})q(x_k^i|y_{1:k})} \\ &= w_k^i \frac{p(y_{k+1}|x_{k+1}^i)p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{1:k+1})}. \end{aligned} \tag{7}$$

The SIS algorithm thus consists of recursive propagation of the weights and support points as each measurement is received sequentially. A common problem with the SIS algorithm is the degeneracy phenomenon, which means that after a few iterations, all but one particles will have negligible weights. This degeneracy indicates that a large computational effort is devoted to updating particles whose contribution to the approximation to π_k is almost zero. In conventional PF algorithms, a resampling step is used as a practical way to reduce the effect of degeneracy. The basic idea of resampling is to eliminate small weight particles and to replicate large weight particles. The resampling step involves generating a new set x_k^{i*} by resampling (with replacement) N times from an approximate discrete representation of $p(x_k|y_{1:k})$ given by

$$p(x_k|y_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(x_k - x_k^i) \tag{8}$$

so that $\Pr(x_k^{i*} = x_k^j) = w_k^j$. Note that after the resampling step, the particles with bigger weights will be replicated into multiple copies, while particles with much smaller weights will be dropped out from the resulting particle set. So the resampling procedure can be regarded as a selection method in terms of replicating bigger weight particles and deleting smaller weight ones. Refer to [1, 34] for more details on different approaches to implement the resampling step.

3 Particle Filter Optimization Algorithms

In this section, we give a brief overview on PFO algorithms, which are grounded in the PF theory presented in Section 2. The basic principle of PFO algorithms consists of transforming the objective function to be a target pdf and then performing PF to simulate the target pdf. The hope is that the optimum of the objective function can be assessed from the resulting samples.

Here we focus on the maximization problem

$$\max_{x \in X} f(x) \tag{9}$$

where f denotes the objective function, X denotes the nonempty solution space defined in $\mathbb{R}^d$, d is the dimension of x, f ($f(X) \in \mathbb{R}$) is a continuous real-valued function that is bounded on X. We denote the maximal function value as f^*, i.e., there exists an x^* such that $f(x) \leq f^* \triangleq f(x^*)$, $\forall x \in X$.

Based on the PF theory as presented in Section 2, the working mechanism of PFO algorithms consists of sampling from a sequence of target distributions $\{\pi_k\}_{k=1:K}$ sequentially, whereas the purpose of sampling is different. In conventional PF theory, the purpose of sampling is to obtain a numerical approximation to the stochastic integrals (such as the one in Eqn. (2)) required for state inference, which is otherwise intractable in computation. For PFO algorithms, the purpose of sampling is to find the optimum, namely x^*.

A basic PFO algorithm consists of K sequentially processed iterations. Specifically, the kth ($k > 1$) iteration consists of the following steps. First, generate a particle set, $\{x_k^i\}_{i=1:N}$, from the proposal pdf $q(\cdot|\{x_{k-1}^i\}_{i=1:N})$, where $\{x_{k-1}^i\}_{i=1:N}$ is the outputted particle set of the $(k-1)$th iteration. Then, weight the particles by $w_k^i \propto \pi_k(x_k^i)/q(x_k^i)$, $\sum_{i=1}^N w_k^i = 1$. Finally, perform resampling, obtaining an equally weighted particle set $\{x_k^{i*}\}_{i=1:N}$ and then set $x_k^i = x_k^{i*}$ and $w_k^i = 1/N$.

The PFO algorithms can differentiate from each other by the definitions of the target pdf π_k and of the proposal pdf q. A specific definition of π_k and q determines how the objective function is implanted in the sampling process and how the random samples (i.e., candidate solutions) are generated, respectively.

3.1 On the definition of the target pdf

Now we focus on the different definitions of π_k in the context of PFO. A simple definition of the target pdf is shown to be [32]

$$\pi_k(x) \propto (f(x))^{\phi(k)}, \tag{10}$$

where $0 < \phi(k)$ and $\phi(k) < \phi(k+1)$, $\forall k \in [1, \ldots, K]$. The underlying idea is to make the initial target pdf π_1 (usually corresponding to a $\phi(1) < 1$) easy to sample from and the final target π_K concentrate around a very proximal area of the ideal optimum, i.e., x^*.

In [28, 35], the target pdf π_k is defined to be

$$\pi_k(x) = \frac{\varphi(f(x) - y_k)\pi_{k-1}(x)}{\int \varphi(f(x) - y_k)\pi_{k-1}(x)dx}, \text{for} \quad k > 1, \tag{11}$$

where $\varphi(\cdot)$ denotes a positive and strictly increasing pdf that is continuous on its support $[0, \infty)$; y_k is the maximum objective function value that has been found at the kth iteration, taken here as a noisy observation of the optimal function value $f(x^*)$. Eqn. (11) implies that π_k is tuned towards the more promising area where $f(x)$ is greater than y_k, since $\varphi(f(x) - y_k)$ is positive if $f(x) \geq y_k$ and is zero otherwise. Note that such a definition of the target pdf can theoretically lend the estimate of $f(x^*)$ to monotonically increasing and asymptotically converge to the true optimal function value [28]. A similar definition of the target pdf can be found in [20], and it is shown that PFO algorithms derived based on such kind of target pdf definitions are equivalent to the cross entropy methods [36].

The Boltzmann distribution is another choice in defining π_t for PFO algorithms [29, 31]. Specifically, the target pdf at the kth iteration is defined to be

$$\pi_k(x) = \frac{1}{Z_k} \exp\left(\frac{f(x)}{T_k}\right), \tag{12}$$

where $Z_k = \int \exp(f(x)/T_k)dx$ is the normalization constant, and T_k is referred to as the annealing temperature at the kth iteration. It is proved that the temperature T_k does not have to be monotonically decreasing, as long as $T_k \to 0$

and the absolute change $\triangle k = \frac{1}{T_k} - \frac{1}{T_{k-1}}$ is monotonically decreasing to 0 as $k \to \infty$ [31]. So the Boltzmann distribution based definition allows more flexible temperature cooling schedule.

3.2 On the definition of the proposal pdf

One common practice in defining the proposal pdf q for PFO algorithms is to use a symmetric such as normal distribution [31]. Specifically, it generates the value of x_k^i from a proposal

$$q(\cdot|\{x_{k-1}^j\}_{j=1:N}) = \mathcal{N}(x_{k-1}^i, \Sigma), \tag{13}$$

where $\mathcal{N}(x, \Sigma)$ denotes a normal distribution with mean x and covariance matrix Σ. Note that such a definition just corresponds to the local random walk proposal, which has been widely used in the Metropolis-Hastings algorithm [37].

The mainstream approach to define $q(x_k^i|\{x_{k-1}^j\}_{j=1:N})$ consists of representing it by a parametric model. The resulting PFO algorithm was therefore termed the model based optimization approach [35]. The optional model forms include but not limited to Gaussian, mixture of Gaussian pdfs and mixture of Student's t pdfs [38]. Note that the underlying idea of seeking a model representation of the proposal to generate new particles (i.e., the candidate solutions in the context of optimization) also appeared in the cross entropy methods [39], the adaptive importance sampling methods [40, 41] and the estimation of distribution algorithms (EDA) [42].

3.3 On practical issues in implementing PFO algorithms

Here we present several practical issues that have been left out but are important in designing a specific PFO algorithm.

First, the temperature cooling schedule needs to be determined. The most simple and straightforward way is to set T_k as a constant $\forall k \in [1, \ldots, K]$, such as in [19, 20, 28, 35]. In addition, an empirically selected monotonically decreasing (corresponding to Eqn. (12)) or increasing (corresponding to Eqn. (10)) cooling schedule is usually the choice. Recently some efforts have been made in designing an adaptive annealing temperature schedule based on information gathered during randomized exploration [38, 43]. Despite this progress, how to derive a theoretically optimal as well as practical cooling schedule remains an open problem.

Second, the form of the model in defining the proposal pdf needs to be fixed up. The mixture model has shown to be a flexible and powerful choice, especially for multimodal objective functions [38]. Given a specific form of the model for representing the proposal, a corresponding parameter estimation approach needs to be selected for determining the values of the model's parameters.

In addition, the sample size N, the resampling strategy and the number of iterations can be specified empirically based on the amount of available resources in computation, memory and time.

4 Performance Evaluation

Here we focus on an optimization task in searching the global maximum of the minus Michalewicz function defined as below

$$f(\mathrm{x}) = \sum_{i=1}^{d} \sin(x_i) \left(\sin\left(\frac{i x_i^2}{\pi} \right) \right)^{2m}. \tag{14}$$

This function is multi-modal, having $d!$ local maxima. The parameter m defines the steepness of this function. A larger m leads to a more difficult search. We use the recommended value $m = 10$ here. We evaluate on the hypercube $x_i \in [0, \pi]$, for all $i = 1, \ldots, d$. The global maximum $f(\mathrm{x}^\star)$ in 5 and 10 dimensional cases are 4.687658 and 9.66015, respectively.

We use the Trelea type vectorized PSO [44] as the benchmark algorithm for performance comparison. A basic version of the PFO algorithm (BPFO) [20], and one of the most advanced PFO algorithms, termed posterior exploration based Sequential Monte Carlo (PE-SMC) [38], are involved in this comparison. The key difference between BPFO and PE-SMC lies in the definitions of the target and the proposal pdfs. Refer to [20] and [38] for details on the BPFO and PE-SMC algorithms, respectively. Following [44], we set the swarm size of the PSO algorithm to be 50, and the maximum number of iterations to be 10000. For BPFO and PE-SMC, the particle size is set to be 2000 and 5000 for the 5D and 10D cases, respectively. The termination condition is that in the last 10 iterations, no better solution is found. Each algorithm is ran 100 times independently for each case and the convergence results are listed in Table 1. Boldface in the table indicates the best result obtained. As is shown, PE-SMC beats PSO, while PSO is preferable to BPFO, for both cases. Refer to [38] for more results on performance comparisons between the PSO and PFO algorithms.

d	Goal($f(\mathrm{x}^\star)$)	PSO	BPFO	PE-SMC
5	4.687658	$4.6624 \pm 4.01 \times 10^{-2}$	4.2103 ± 0.1648	$\mathbf{4.6875} \pm 2.81 \times 10^{-4}$
10	9.66015	9.4562 ± 0.16	5.9357 ± 0.2852	$\mathbf{9.6596} \pm 1.77 \times 10^{-2}$

Table 1: Convergence results of the involved algorithms

5 Concluding remarks

In this paper, we provided a very brief introduction to the PF theory and the PFO algorithms. It is shown that the success of PFO algorithms depends on appropriate designs of the target and proposal pdfs. In what follows, let us make several remarks on the relationships between the PFO algorithms and the swarm intelligence methods. Our intention is to provide some insight into the behaviour

of the PFO algorithm and its relationship with some existing swarm intelligence algorithms.

It is shown that, PFO algorithms equipped with annealed target pdfs, as defined in Eqns. (10) and (12), find a straightforward relationship with the simulated annealing (SA) method [45]. The SA is a serially processed algorithm relying on a Markov Chain Monte Carlo sampling mechanism, while it is natural to find parallel implementations for the PFO algorithms. In addition, the PFO algorithm can provide a forward looking mechanism for automatically determining an appropriate annealing temperature value for the following iteration based on the information collected at the current iteration [38]. If the target pdf in the PFO algorithm keeps invariant per iteration, namely $\pi_k(x) = \pi_{k-1}(x)$, $\forall k \in [2, \ldots, K]$, the PFO algorithm just degenerates into the estimation of distribution algorithm in concept, although their sampling and selection mechanisms look different in form. Actually the PFO algorithm can be regarded as a generic type swarm intelligence technique, in which the candidate solutions are generated, weighted and propagated based on probability theorem, other than meta-heuristics. Inherited from the PF theory, the PFO algorithm could own satisfactory theoretical properties in e.g., convergence and global searching, while it consumes much computational resources in always maintaining a strict probabilistic interpretation of the landscape of the evaluated solutions in each iteration. We argue that a promising future direction lies in developing novel approaches that can make best use of the advantages and bypass the disadvantages of the PFO and the meta-heuristics based algorithms.

6 Acknowledgments

This work was partly supported by the National Natural Science Foundation (NSF) of China under grant Nos. 61571238, 61302158 and 61571434, the NSF of Jiangsu province under grant No. BK20130869, China Postdoctoral Science Foundation under grant No.2015M580455, China Postdoctoral International Academic Exchange Program and the Scientific Research Foundation of Nanjing University of Posts and Telecommunications under grant No. NY213030.

References

1. Liu, J.S., Chen, R.: Sequential monte carlo methods for dynamic systems. Journal of the American statistical association **93**(443) (1998) 1032–1044
2. Gordon, N.J., Salmond, D.J., Smith, A.F.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings F (Radar and Signal Processing) **140**(2) (1993) 107–113
3. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Trans. on Signal Processing **50**(2) (2002) 174–188
4. Del Moral, P.: Feynman-Kac Formulae. Springer (2004)
5. Del Moral, P.: Measure-valued processes and interacting particle systems. application to nonlinear filtering problems. Annals of Applied Probability (1998) 438–495

6. Del Moral, P.: Non-linear filtering: interacting particle resolution. Markov processes and related fields **2**(4) (1996) 555–581
7. Crisan, D., Doucet, A.: A survey of convergence results on particle filtering methods for practitioners. IEEE Trans. on signal processing **50**(3) (2002) 736–746
8. Hu, X.L., Schön, T.B., Ljung, L.: A basic convergence result for particle filtering. IEEE Trans. on Signal Processing **56**(4) (2008) 1337–1348
9. Daum, F.: Nonlinear filters: beyond the kalman filter. IEEE Aerospace and Electronic Systems Magazine **20**(8) (2005) 57–69
10. Ristic, B., Arulampalam, S., Gordon, N.: Beyond the kalman filter. IEEE Aerospace and Electronic Systems Magazine **19**(7) (2004) 37–38
11. Chen, Z.: Bayesian filtering: From kalman filters to particle filters, and beyond. Statistics **182**(1) (2003) 1–69
12. Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., Nordlund, P.: Particle filters for positioning, navigation, and tracking. IEEE Trans. on Signal Processing **50**(2) (2002) 425–437
13. Gordon, N., Salmond, D., Ewing, C.: Bayesian state estimation for tracking and guidance using the bootstrap filter. Journal of Guidance, Control, and Dynamics **18**(6) (1995) 1434–1443
14. Pozna, C., Precup, R.E., Földesi, P.: A novel pose estimation algorithm for robotic navigation. Robotics and Autonomous Systems **63** (2015) 10–21
15. De Freitas, N.: Rao-blackwellised particle filtering for fault diagnosis. In: Proc. of IEEE Aerospace Conf. Volume 4., IEEE (2002) 4–1767
16. Clapp, T.C., Godsill, S.J.: Fixed-lag blind equalization and sequence estimation in digital communications systems using sequential importance sampling. In: Proc. of IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP). Volume 5. (1999) 2495–2498
17. Fearnhead, P.: Using random quasi-monte-carlo within particle filters, with application to financial time series. Journal of Computational and Graphical Statistics **14**(4) (2005) 751–769
18. DiMaio, F., Kondrashov, D.A., Bitto, E., Soni, A., Bingman, C.A., Phillips, G.N., Shavlik, J.W.: Creating protein models from electron-density maps using particle-filtering methods. Bioinformatics **23**(21) (2007) 2851–2858
19. Chen, X., Zhou, E.: Population model-based optimization with Sequential Monte Carlo. In: Proc. of the 2013 Winter Simulation Conference. (2013) 1004–1015
20. Zhou, E., Fu, M.C., Marcus, S., et al.: Particle filtering framework for a class of randomized optimization algorithms. IEEE Trans. on Automatic Control **59**(4) (2014) 1025–1030
21. Whiteley, N., Johansen, A.M.: Recent developments in auxiliary particle filtering. Barber, Cemgil, and Chiappa, editors, Inference and Learning in Dynamic Models. Cambridge University Press **38** (2010) 39–47
22. Doucet, A., Godsill, S., Andrieu, C.: On sequential monte carlo sampling methods for bayesian filtering. Statistics and computing **10**(3) (2000) 197–208
23. Zhang, M., Xin, M., Yang, J.: Adaptive multi-cue based particle swarm optimization guided particle filter tracking in infrared videos. Neurocomputing **122** (2013) 163–171
24. Tong, G., Fang, Z., Xu, X.: A particle swarm optimized particle filter for nonlinear system state estimation. In: IEEE Congress on Evolutionary Computation (CEC). (2006) 438–442
25. Kwok, N.M., Fang, G., Zhou, W.: Evolutionary particle filter: re-sampling from the genetic algorithm perspective. In: Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems. (2005) 2935–2940

26. Han, H., Ding, Y.S., Hao, K.R., Liang, X.: An evolutionary particle filter with the immune genetic algorithm for intelligent video target tracking. Computers & Mathematics with Applications **62**(7) (2011) 2685–2695
27. Xiaolong, L., Jinfu, F., Qian, L., Taorong, L., Bingjie, L.: A swarm intelligence optimization for particle filter. In: Proc. of the 7th World Congress on Intelligent Control and Automation. (2008) 1986–1991
28. Zhou, E., Fu, M.C., Marcus, S.I.: A particle filtering framework for randomized optimization algorithms. In: Proc. of the 40th Conference on Winter Simulation. (2008) 647–654
29. Ji, C., Zhang, Y., Tong, M., Yang, S.: Particle filter with swarm move for optimization. In: Parallel Problem Solving from Nature (PPSN). (2008) 909–918
30. Medina, J.C., Taflanidis, A.A.: Adaptive importance sampling for optimization under uncertainty problems. Computer Methods in Applied Mechanics and Engineering **279** (2014) 133–162
31. Zhou, E., Chen, X.: Sequential monte carlo simulated annealing. Journal of Global Optimization **55**(1) (2013) 101–124
32. Del Moral, P., Doucet, A., Jasra, A.: Sequential monte carlo samplers. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **68**(3) (2006) 411–436
33. Liu, J.S., Chen, R., Logvinenko, T.: A theoretical framework for sequential importance sampling with resampling. In: Sequential Monte Carlo methods in practice. Springer (2001) 225–246
34. Kitagawa, G.: Monte carlo filter and smoother for non-gaussian nonlinear state space models. Journal of computational and graphical statistics **5**(1) (1996) 1–25
35. Hu, J., Wang, Y., Zhou, E., Fu, M.C., Marcus, S.I.: A survey of some model-based methods for global optimization. In: Optimization, Control, and Applications of Stochastic Systems. Springer (2012) 157–179
36. De Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method. Annals of operations research **134**(1) (2005) 19–67
37. Hastings, W.K.: Monte carlo sampling methods using markov chains and their applications. Biometrika **57**(1) (1970) 97–109
38. Liu, B.: Posterior exploration based Sequential Monte Carlo for global optimization. arXiv preprint arXiv:1509.08870 (2015)
39. Rubinstein, R.: The cross-entropy method for combinatorial and continuous optimization. Methodology and computing in applied probability **1**(2) (1999) 127–190
40. Oh, M.S., Berger, J.O.: Integration of multimodal functions by monte carlo importance sampling. Journal of the American Statistical Association **88**(422) (1993) 450–456
41. Cappé, O., Douc, R., Guillin, A., Marin, J.M., Robert, C.P.: Adaptive importance sampling in general mixture classes. Statistics and Computing **18**(4) (2008) 447–459
42. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation **1**(3) (2011) 111–128
43. Yang, C., Kumar, M.: An information guided framework for simulated annealing. Journal of Global Optimization **62**(1) (2015) 131–154
44. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters **85**(6) (2003) 317–325
45. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., et al.: Optimization by simulated annealing. Science **220**(4598) (1983) 671–680