Overview of Some Global Optimization Algorithms

Chi Zhang

Abstract

This technical note presents several global optimization algorithms, most of which are particle-based. Besides the traditional particle swarm optimization, many model-based population methods are included, e.g., the cross-entropy (CE), the model reference adaptive search (MRAS), the recently proposed controlled particle filter (CPF) etc. This note serves as a reference for the C++ and Python codes related to the optimization project.

Problem statement: We consider the global minimization problem:

$$\min_{x\in\mathbb{R}^d} \; h(x),$$

where $h:\mathbb{R}^d\to\mathbb{R}$ is a real-valued function. The *global minimizer* is denoted as

$$\bar{x} := \arg\min_{x\in\mathbb{R}^d} \; h(x).$$

It is assumed that such a minimizer exists and is unique.

This note provides algorithmic description of several global optimization algorithms. The model-based algorithms are described in Sec. I. The more conventional swarm algorithms are contained in Sec. II. Sec. III presents a recent controlled particle filter algorithm. Throughout the note, particles at iteration n or time t are denoted as $\{X_n^i\}_{i=1}^N$ and $\{X_t^i\}_{i=1}^N$, respectively.

I. Model-based Algorithms

In a model-based optimization algorithms, the search of the global minimizer is guided by a (prescribed) reference distribution (model) that exhibits desirable convergence property. Numerically, these algorithms involve the following iterative steps:

1) generate samples from a prescribed distribution,
2) select "elite" samples according to a selection rule, and
3) estimate the new distribution using the elite samples.

The estimated distribution should stay "close" to the reference model. How these steps are implemented differentiates one model-based algorithm from another.

This section reviews several recent model-based algorithms. Their numerical procedures to carry out the above steps are delineated and compared.

A. Cross-entropy (CE)

At the n-th iteration of a CE algorithm, the particles are sampled i.i.d. from a parametric density $q(\cdot,\theta_n)$ with current parameter value θ_n. The elite samples are selected via a quantile rule,

$$\gamma_{n+1} := \inf\{\gamma : P_{\theta_n}(h(X) \le l) \ge \zeta\},$$

such that only samples with function values below γ_n are used to update the model parameter,

$$\theta_{n+1} := \arg\max_{\theta\in\Theta} \mathsf{E}_{\theta_n}\left[\phi(h(X))\,\mathbf{1}_{\{h(X)\le\gamma_{n+1}\}}\log q(X,\theta)\right]. \tag{1}$$

The update step (1) is shown to minimize the Kullback-Leibler (KL) distance between a reference model p^* and q (c.f., [2]), where

$$p^*_{n+1}(x) := \frac{\phi(h(x))\,\mathbf{1}_{\{h(x)\le\gamma_n\}}\,q(x,\theta_n)}{\mathsf{E}_{\theta_n}\left[\phi(h(X))\,\mathbf{1}_{\{h(X)\le\gamma_n\}}\right]}. \tag{2}$$

This reference model is called the *optimal importance sampling pdf* in [4].

The function ϕ is non-increasing and non-negative. Choosing $\phi(z)\equiv 1$ yields the *standard CE* method, while $\phi(z)=z$ yields the *extended CE* method. Our numerical study onlu considers the standard CE.

The numerical procedure of CE is tabulated in Algorithm 1; see also [4], [2].

Algorithm 1 CE numerical algorithm

1: **Input:** Initial distribution $q(x,\theta_0)$, parameter $\zeta\in(0,1]$

2: **Iteration** n $(n\ge 0)$**:**

3: Sample $\{X_n^i\}_{i=1}^N$ i.i.d. from $q(x,\theta_n)$

4: Calculate the $(1-\zeta)-$quantile $\gamma_{n+1} := \mathbf{h}_{(\lceil\zeta N\rceil)}$, where $\mathbf{h}_{(k)}$ is the k-th order statistic of $\mathbf{h} := \left(h(X_n^i),...,h(X_n^N)\right)$, and $\lceil a\rceil$ denotes the smallest integer greater than a

5: Calculate importance weights,

$$w_n^i \propto \phi(h(X_n^i))\mathbf{1}_{\{h(X_n^i)\le\gamma_n\}}, \qquad \sum_{i=1}^N w_n^i = 1$$

6: Update the model parameter,

$$\theta_{n+1} := \arg\max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^N w_n^i \log q(X_n^i,\theta) \tag{3}$$

7: Assign $n=n+1$ if the stopping rule is not satisfied; otherwise terminate

The parameter update (3) is identical to the maximum likelihood estimation of the parameter θ in $q(x,\theta)$. When q is chosen as Gaussian, i.e., $\theta = (\mu, \Sigma)$, the update formulae are given explicitly as,

$$\mu_{n+1} = \sum_{i=1}^{N} w_n^i X_n^i,$$
$$\Sigma_{n+1} = \sum_{i=1}^{N} w_n^i (X_n^i - \mu_{n+1})(X_n^i - \mu_{n+1})^T.$$

Note that only the elite samples (i.e., samples satisfying $h(X_n^i) \leq \gamma_n$) have non-zero weights and contribute to the estimated mean and covariance. This parameter update is also included in MRAS and MEO to be presented next.

B. Model Reference Adaptive Search (MRAS)

The MRAS algorithm resembles CE but with a slightly different reference model, defined recursively as,

$$\begin{aligned} p_{n+1}^*(x) &:= \frac{\phi(h(x))\,\mathbf{1}_{\{h(x)\leq\gamma_n\}}\,p_n^*(x)}{\mathsf{E}_{p_n^*}\left[\phi(h(X))\,\mathbf{1}_{\{h(X)\leq\gamma_n\}}\right]}, \\ p_0^*(x) &:= \frac{\mathbf{1}_{\{h(x)\leq\gamma_0\}}}{\mathsf{E}_{\theta_0}\left[\mathbf{1}_{\{h(X)\leq\gamma_0\}}/q(X,\theta_0)\right]}, \end{aligned} \tag{4}$$

where q is chosen from a parametrized family such as Gaussians. Correspondingly, the parameter update, as a counterpart of (1) in CE, becomes,

$$\theta_{n+1} := \arg\max_{\theta\in\Theta} \mathsf{E}_{\theta_n}\left[\frac{\phi(h(X))^n}{q(X,\theta_n)}\,\mathbf{1}_{\{h(X)\leq\gamma_n\}}\log q(X,\theta)\right]. \tag{5}$$

The numerical procedure of MRAS is tabulated in Algorithm 2; see also [2]. Additional numerical treatment is included:

(i) The quantile level ζ is adaptive, and the sequence $\{\gamma_n\}_{n\geq 0}$ is guaranteed to be non-increasing.

(ii) A smoothing parameter ν is introduced to "damp" the parameter update (see Line 17), which effectively prevents adversely fast convergence that leads to premature solutions.

Algorithm 2 MRAS numerical algorithm

Input: Initial distribution $q(x,\theta_0)$, parameters $\zeta_0 \in (0,1]$, $\varepsilon \in (0,1)$, $\nu \in [0,1]$

Iteration n **(**$n \geq 0$**):**

Sample $\{X_n^i\}_{i=1}^N$ i.i.d. from $q(x,\theta_n)$

Calculate the $(1-\zeta_n)-$ quantile $\tilde{\gamma}_n(\zeta_n) := \mathbf{h}_{(\lceil \zeta_n N \rceil)}$, where $\mathbf{h}_{(k)}$ is the k-th order statistic of $\mathbf{h} := \big(h(X_n^i),...,h(X_n^N)\big)$, and $\lceil a \rceil$ is the smallest integer greater than a

if $n = 0$ Assign $\gamma_n = \tilde{\gamma}_n(\zeta_n)$ and $\zeta_{n+1} = \zeta_n$

if $n \geq 1$ and $\tilde{\gamma}_n \leq \gamma_{n-1} + \varepsilon/2$ **then**

  Assign $\gamma_n = \tilde{\gamma}_n(\zeta_n)$ and $\zeta_n = \zeta_{n-1}$

else

  Find largest $\zeta \in (0, \zeta_{n-1})$ such that $\tilde{\gamma}_n(\zeta) \leq \gamma_{n-1} + \varepsilon/2$

  if Such a ζ exists **then**

    Assign $\gamma_n = \tilde{\gamma}_n(\zeta)$ and $\zeta_n = \zeta$

  else

    Assign $\gamma_n = \tilde{\gamma}_{n-1}$ and $\zeta_n = \zeta_{n-1}$

  end if

end if

Calculate importance weights,

$$w_n^i \propto \frac{\phi(h(X_n^i))^n}{q(X_n^i)} \mathbf{1}_{\{h(X_n^i) \leq \gamma_n\}}, \quad \sum_{i=1}^N w_n^i = 1$$

Update the model parameter,

$$\tilde{\theta}_{n+1} := \arg\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N w_n^i \log q(X_n^i, \theta)$$

Assign $\theta_{n+1} = \nu\, \tilde{\theta}_{n+1} + (1-\nu)\, \theta_n$

Assign $n = n+1$ if the stopping rule is not satisfied; otherwise terminate

The choice of the function ϕ is similar as in CE. If $\phi(z) = e^{-rz}$ is used, where $r > 0$ is a small parameter, the reference model of MRAS becomes,

$$p_{n+1}^*(x) = \frac{p_n^*(x) \exp(-r\, h(x)) \mathbf{1}_{\{h(X) \leq \gamma_n\}}}{\int p_n^*(y) \exp(-r\, h(y))\, \mathbf{1}_{\{h(X) \leq \gamma_n\}}\, \mathrm{d}y},$$

with p_0^* as a uniform distribution. Without the selection factor $\mathbf{1}_{\{h(X) \leq \gamma_n\}}$, this model is identical to the Bayes' model that is used in our controlled particle filter algorithm (see (11)).

C. Model-based Evolutionary Optimization (MEO)

The reference model in MEO evolves according to the replicator dynamics,

$$\frac{\mathrm{d}p_t^*}{\mathrm{d}t}(x) = -(h(x) - \hat{h}_t)\, p_t^*(x), \tag{6}$$

where $\hat{h}_t := \int h(x)\, p_t^*(x)\, \mathrm{d}x$. In a sample-based implementation, the density is approximated as $p_t^*(x) = \sum_{i=1}^N w_t^i \delta(x - X_t^i)$, and the corresponding odes for the weights are given by,

$$\frac{\mathrm{d}w_t^i}{\mathrm{d}t} = -\big(h(X_t^i) - \hat{h}_n^{(N)}\big)\, w_t^i, \tag{7}$$

where $\hat{h}_t^{(N)} := \sum_{i=1}^N w_t^i\, h(X_t^i)$. The discrete-time counterpart of (7) is obtained as,

$$w_{n+1}^i = w_n^i - \frac{h(X_n^i) - \hat{h}_t^{(N)}}{\sum_{i=1}^N h(X_n^i)}\, w_n^i,$$

which determines the importance weights of the samples in MEO. The normalization $\sum_{i=1}^N w_{n+1}^i = 1$ is automatically satisfied.

The numerical procedure of MEO is tabulated in Algorithm 3; see also [5].

Algorithm 3 MEO numerical algorithm

1: **Input:** Initial distribution $q(x, \theta_0)$, parameter $\zeta \in (0, 1]$

2: **Iteration** n $(n \geq 0)$**:**

3: Sample $\{X_n^i\}_{i=1}^N$ i.i.d. from $q(x, \theta_n)$ with *equal* weights $w_n^i = 1/N,\ \forall i$

4: Calculate the $(1-\zeta)-$quantile $\gamma_n := \mathbf{h}_{(\lceil \zeta N \rceil)}$, where $\mathbf{h}_{(k)}$ is the k-th order statistic of $\mathbf{h} := \big(h(X_n^i), ..., h(X_n^N)\big)$, and $\lceil a \rceil$ denotes the smallest integer greater than a

5: Calculate importance weights,

$$w_{n+1}^i = w_n^i - \frac{h(X_n^i) - \hat{h}_n^{(N)}}{\sum_{i=1}^N h(X_n^i)}\, w_n^i,$$

6: Update the model parameter,

$$\theta_{n+1} := \arg\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N w_{n+1}^i \log q(X_n^i, \theta)$$

7: Assign $n = n + 1$ if the stopping rule is not satisfied; otherwise terminate

D. Particle Filtering for Optimization (PFO)

A particle filtering framework for global optimization is proposed in [9], of which CE and MRAS are special cases. The PFO casts the optimization as a filtering problem with the following state-space model,

$$\begin{aligned} X_{n+1} &= X_n + B_{n+1}, \\ Y_{n+1} &= h(X_{n+1}) + W_{n+1}, \end{aligned}$$

for $n \geq 0$, where $\{B_n\}_{n\geq 1}$ and $\{W_n\}_{n\geq 1}$ are mutually independent sequence of random variables that are also independent of X_0. The global optimizer is viewed as a static hidden variable to be estimated. The reference model is given by the Bayes' rule,

$$\begin{aligned} \text{(Prediction)} \quad p^*_{n+1|n}(x) &= \int K_{n+1}(x|x_n)\, p^*_n(x_n)\, \mathrm{d}x_n, \\ \text{(Update)} \quad p^*_{n+1}(x) &= \frac{\phi(y_{n+1} - h(x))\, p^*_{n+1|n}(x)}{\int \phi(y_{n+1} - h(z))\, p^*_{n+1|n}(z)\, \mathrm{d}z}, \end{aligned} \tag{8}$$

where the transition kernel $K(\cdot|\cdot)$ and the function $\phi(\cdot)$ represent the distribution of the process noise $\{B_n\}_{n\geq 1}$ and the observation noise $\{W_n\}_{n\geq 1}$, respectively.

The implementation of PFO is nearly identical to a bootstrap particle filter, except that a non-increasing sequence of fictitious observations needs to be generated. The numerical procedure of PFO is tabulated in Algorithm 4; see also [9].

Algorithm 4 PFO numerical algorithm

1: **Input:** Initial samples $\{X^i_0\}^N_{i=1} \overset{\text{i.i.d.}}{\sim} p^*_0(x)$, kernel function $K(\cdot|\cdot)$, pdf $\phi(\cdot)$
2: **Iteration** n **(**$n \geq 0$**):**
3: Sample $\widetilde{X}^i_{n+1} \sim K(\cdot|X^i_n)$ for $i = 1, \ldots, N$
4: *Observation generation*: Take y_{n+1} to be a sample function value $h(\widetilde{X}^i_{n+1})$ according to certain rule (e.g., quantile rule). If $n \geq 1$ and $y_{n+1} > y_n$, then set $y_{n+1} = y_n$
5: *Bayes' update*: Calculate importance weights,

$$w^i_{n+1} \propto \phi(y_{n+1} - h(\widetilde{X}^i_{n+1})), \quad \sum_{i=1}^N w^i_{n+1} = 1$$

6: *Resampling*: Generate new samples $\{X^i_{n+1}\}^N_{i=1}$ using resampling with replacement; c.f., [1]
7: Assign $n = n+1$ if the stopping rule is not satisfied; otherwise terminate

It is discussed in [9] that the PFO framework includes both CE and MRAS as special instantiations, with a quantile rule to generate observations, with particular choice of $\phi(\cdot)$, and without process noise. However, while CE, MRAS and MEO generate samples from a parametric

distribution and select only a few elite samples, the PFO is non-parametric and uses resampling to obtain new samples.

E. Sequential Monte-Carlo Simulated Annealing (SMC-SA)

The SMC-SA algorithm, proposed recently in [8], combines PFO with the well-known simulate annealing (SA) [3]. The reference model is the Boltzmann distribution in SA,

$$p_n^*(x) = \frac{\exp(-h(x)/T_n)}{\int \exp(-h(y)/T_n)\,\mathrm{d}y}, \tag{9}$$

where the non-increasing real-valued sequence $\{T_n\}_{n\geq 0}$ is a prescribed cooling schedule. The SMC-SA algorithm estimates the Boltzmann sequence using the importance sampling in PFO followed by an SA move for each sample. The numerical procedure of SMC-SA is summarized in Algorithm 5; see also [8].

Algorithm 5 SMC-SA numerical algorithm

1: **Input:** Initial samples $\{X_0^i\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} p_0^*(x)$, cooling schedule $\{T_n\}_{n\geq 0}$, kernel function $K(\cdot|\cdot)$
2: **Iteration** n $(n \geq 0)$**:**
3: *Importance update*: Calculate importance weights,

$$w_{n+1}^i \propto \exp(-h(X_n^i)/T_n)/p_n^*(X_n^i), \quad n=0$$
$$w_{n+1}^i \propto \exp\big(h(X_n^i)(1/T_n - 1/T_{n+1})\big), \quad n \geq 1,$$

and $\sum_{i=1}^N w_{n+1}^i = 1$ for $n \geq 1$
4: *Resampling*: Generate new samples $\{\widetilde{X}_{n+1}^i\}_{i=1}^N$ using resampling with replacement
5: **for** $i = 1,...,N$ **do**
6: Generate $Y_{n+1}^i \sim K(y|\widetilde{X}_{n+1}^i)$
7: Calculate acceptance probability

$$\zeta_{n+1}^i = \min\ \Big\{\exp\big((h(\widetilde{X}_{n+1}^i) - h(Y_{n+1}^i))/T_{n+1}\big),\ 1\Big\}$$

8: Accept/Reject

$$X_{n+1}^i = \begin{cases} Y_{n+1}^i, \text{ w.p. } \zeta_{n+1}^i \\ \widetilde{X}_{n+1}^i, \text{ w.p. } 1-\zeta_{n+1}^i \end{cases}$$

9: **end for**
10: Assign $n = n+1$ if the stopping rule is not satisfied; otherwise terminate

The following cooling schedule is used in [8],

$$T_n = \frac{|h_n^*|}{\log(n+1)},$$

where h_n^* denotes the best function value found by the algorithm up to iteration n.

F. Sequential Importance Sampling and Resampling (SISR)

The SISR is a non-parametric algorithm soly based on importance sampling and resampling. The reference model is the Bayes' model,

$$p_{n+1}^*(x) = \frac{p_n^*(x)\exp(-\beta\, h(x)\,\Delta t_n)}{\int p_n^*(y)\exp(-\beta\, h(y)\,\Delta t_n)\,\mathrm{d}y}, \tag{10}$$

where $\Delta t_n = t_{n+1} - t_n$ is the time step. The numerical procedure is tabulated in Algorithm 6.

Algorithm 6 SISR numerical algorithm

1: **Input:** Initial samples $\{X_0^i\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} p_0^*(x)$, kernel function $K(\cdot|\cdot)$
2: **Iteration** n $(n \geq 0)$**:**
3: *Importance sampling*: Sample $\widetilde{X}_{n+1}^i \sim K(\cdot|X_n^i)$ for $i = 1,...,N$
4: *Bayes' update:* Calculate the weights,

$$w_{n+1}^i \propto \exp\big(-\beta\, h(\widetilde{X}_{n+1}^i)\,\Delta t_n\big), \quad \sum_{i=1}^N w_{n+1}^i = 1$$

5: *Resampling*: Generate samples $\{X_{n+1}^i\}_{i=1}^N$ using resampling with replacement
6: Assign $n = n+1$ if the stopping rule is not satisfied; otherwise terminate

II. Swarm Algorithms

A. Particle Swarm Optimization (PSO)

PSO is a metaheuristic algorithm that guides particles' position and velocity towards the best solution found by each particle and the entire population. The numerical steps of PSO is summarized below. The variables P_n^i and G_n denote the best position visited by the i-th particle and the entire population, respectively, at the n-th iteration.

Algorithm 7 PSO numerical algorithm

1: **Input:** Initial samples $\{X_0^i\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} p_0^*(x)$, parameters $c_1,\ c_2,\ w,\ \chi$

2: **Iteration** n**:**

3: Generate constants $r_1,\ r_2 \sim \text{Uniform}\,(0,1)$

4: *Velocity update:* Calculate the velocity,

$$V_{n+1}^i = \chi\left(wV_n^i + c_1 r_1 (P_n^i - X_n^i) + c_2 r_2 (G_n - X_n^i)\right), \quad i = 1,...,N$$

5: *Position update:* Calculate the position,

$$X_{n+1}^i = X_n^i + V_{n+1}^i, \quad i = 1,...,N$$

6: Assign $n = n+1$ if the stopping rule is not satisfied; otherwise terminate

III. Controlled Particle Filter

This section presents the controlled particle filter (CPF) algorithm recently proposed in [7], [6].

A. Overview

As with a model-based algorithm, a reference model is defined in CPF, given by

$$p^*(x,t) := \frac{p_0^*(x)\exp(-\beta h(x)\,t)}{\int p_0^*(y)\exp(-\beta h(y)\,t)\,\mathrm{d}y}, \tag{11}$$

where β is a positive parameter. It is shown that $p^*(x,t)$ satisfies the replicator model (6) used in MEO. The discrete-time recursive form of (11) coincides with the model (10) used in SISR.

The CPF algorithm consists of a set of controlled interacting particles $\{X_t^i\}_{i=1}^N$ where each particle implements a control-based update,

$$\mathrm{d}X_t^i = u(X_t^i,t)\,\mathrm{d}t, \tag{12}$$

where the *control function* $u(x,t)$ is a solution of the following *Poisson equation*,

$$-\nabla\cdot(\rho(x,t)\nabla\phi(x)) = (h(x) - \hat{h}_t)\rho(x,t), \tag{13}$$

where ρ denotes the density of the particles X_t^i, and $\hat{h}_t := \int h(x)\rho(x,t)\,\mathrm{d}x$. In numerical algorithms, $\hat{h}_t \approx \frac{1}{N}\sum_{i=1}^N h(X_t^i) := \hat{h}_t^{(N)}$.

The control function is then obtained as

$$u(x,t) = \nabla\phi(x).$$

Structurally, the control-based approach is a significant departure from the importance sampling based implementation in the model-based algorithms in Sec. I. It is noted that there are no additional steps, e.g., associated with resampling, reproduction, death, or birth of particles. One interpretation of the control input $u(X_t^i,t)$ is that it implements the "Bayesian update step" in (11) to steer the ensemble $\{X_t^i\}_{i=1}^N$ towards the global minimum $\bar{x}$.

IV. Numerical Algorithm

The numerical procedure of CPF is tabulated in Algorithm 8, which simply discretizes the ODE (12). The solution of the Poisson equation needs to be numerically approximated at each time, evaluated only at the particles. Some numerical schemes for control function approximation can be found in [7].

Algorithm 8 CPF numerical algorithm

1: **Input:** Initial samples $\{X_0^i\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} p_0^*(x)$, time step Δt

2: **At time t:**

3: Calculate $\hat{h}_t^{(N)} := \frac{1}{N}\sum_{i=1}^N h(X_t^i)$

4: Approximate control function at particles to obtain $\{\hat{u}(X_t^1),...,\hat{u}(X_t^N)\}$

5: Update the particles,

$$X_{t+\Delta t}^i = X_t^i + \hat{u}(X_t^i)\,\Delta t, \quad i = 1,...,N$$

(Higher-order discretization method may be used)

6: Assign $t = t+\Delta t$ if the stopping rule is not satisfied; otherwise terminate

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*, 50(2):174–188, 2002.

[2] J. Hu, M. C. Fu, and S. I. Marcus. A model reference adaptive search method for global optimization. *Oper. Res.*, 55(3):549–568, 2007.

[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[4] R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

[5] Y. Wang, M. C. Fu, and S. I. Marcus. Model-based evolutionary optimization. In *Proceedings of the 2010 Winter Simulation Conference*, pages 1199–1210, December 2010.

[6] C. Zhang. A particle system for global optimization. In *Proc. 52nd IEEE Conf. Decision Control*, pages 1714–1719, 2013.

[7] C. Zhang, A. Taghvaeu, and P. G. Mehta. A controlled particle filter for global optimization. *arXiv preprint: 1701.02413*, 2017.

[8] E. Zhou and X. Chen. Sequential Monte Carlo simulated annealing. *J. Global Optim.*, 55(1):101–124, 2013.

[9] E. Zhou, M. C. Fu, and S. I. Marcus. Particle filtering framework for a class of randomized optimization algorithms. *IEEE Trans. Autom. Control*, 59(4):1025–1030, 2014.