Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

MODEL-BASED EVOLUTIONARY OPTIMIZATION

Yongqiang Wang

Department of Electrical and Computer
Engineering & Institute for Systems Research,
University of Maryland, College Park,
MD, 20742, U.S.A

Michael C. Fu

The Robert H. Smith School of Business &
Institute for Systems Research,
University of Maryland, College Park,
MD, 20742, U.S.A

Steven I. Marcus

Department of Electrical and Computer
Engineering & Institute for Systems Research,
University of Maryland, College Park,
MD, 20742, U.S.A

ABSTRACT

We propose a new framework for global optimization by building a connection between global optimization problems and evolutionary games. Based on this connection, we propose a Model-based Evolutionary Optimization (MEO) algorithm, which uses probabilistic models to generate new candidate solutions and uses various dynamics from evolutionary game theory to govern the evolution of the probabilistic models. The MEO algorithm also gives new insight into the mechanism of model updating in model-based global optimization algorithms. Based on the MEO algorithm, a novel Population Model-based Evolutionary Optimization (PMEO) algorithm is proposed, which better captures the multimodal property of global optimization problems and gives better simulation results.

1 INTRODUCTION

In the literature of global optimization, there is a class of optimization methods called *model-based* methods that generate new candidate solutions based on probabilistic models, and the probabilistic models are updated by using the previously generated solutions. Some of the well known model-based methods include estimation of distribution algorithms (EDAs) (Mühlenbein and Paaß 1996), cross-entropy (CE) method (Boer et al. 2005, Mannor et al. 2003), and model reference adaptive search (MRAS) (Hu, Fu, and Marcus 2007). EDAs were first proposed by Mühlenbein and Paaß (1996), who introduced probabilistic models to generate new candidate solutions. How to construct high-dimensional models to represent the interdependencies between decision variables is the most crucial and difficult part of the method. CE was first designed as an adaptive algorithm to estimate the probabilities of rare events in stochastic networks (Rubinstein 1997) and then it was modified to solve combinatorial and continuous optimization problems (Rubinstein 1999). One of the critical steps in CE is to minimize the distance between the optimal importance sampling density and a family of parameterized densities that are used to generate new candidate solutions. The threshold parameter that is used to choose elite solutions is adaptively changed and the parameterized probability densities are also updated accordingly. MRAS, introduced by Hu, Fu, and Marcus (2007), incorporates the ideas of EDAs and CE, and guides the updating of the parameterized densities by constructing a series of reference models. Recently, Zhou, Fu, and Marcus (2008) formulate the global optimization problem as a filtering problem and present a new particle filtering-based framework to solve global optimization problems. Bayes updating is used to guide the evolution of the probability models that are used to generate new candidate solutions.

The way of interpreting and updating the probabilistic models makes EDAs, CE, MRAS, and algorithms based on particle filtering different from each other. In the aforementioned *model-based* global optimization algorithms, candidate solutions are generated from probabilistic models, which are updated by these candidate solutions such that better solutions will have a higher chance to be sampled at the next iteration. On the other hand, in evolutionary games, better strategies will spread in the population, and this is somewhat similar to the improvement of models in *model-based* algorithms. This similarity motivates us to connect global optimization problems with evolutionary game theory.

The dynamics that are used to study the evolution of strategies in evolutionary games provides us a powerful tool to investigate the model updating in model-based algorithms.

Game theory studies the strategic interaction of players using different strategies; it has been applied in many areas such as economics, engineering, and biology (Fudenberg and Levine 1998, Shoham, Powers, and Grenager 2004). Recently, authors such as Lambert, Epelman, and Smith (2005) and Garcia, Patek, and Sinha (2007) have applied game theory to solve discrete optimization problems, where they model the optimization problem as a potential game. Fictitious play and joint fictitious play are adopted to obtain the Nash equilibrium and two sampled version of fictitious and joint fictitious play are also proposed in Lambert, Epelman, and Smith (2005) and Garcia, Patek, and Sinha (2007). For a potential game, although fictitious play has been proven to converge to a mixed strategy Nash equilibrium, the mixed strategy equilibrium might not be a feasible solution for the optimization problem. The algorithms in Garcia, Patek, and Sinha (2007) and Lambert, Epelman, and Smith (2005) only work for discrete optimization problems with a finite solution space, and moreover the Nash equilibrium obtained by fictitious play might only be a locally optimal solution.

Evolutionary game theory applies game theory to study the evolution of the number of players playing different strategies in a population setting. After being introduced by the biologist (Smith 1982), evolutionary game theory has become popular in biology and increasingly attracts interest from researchers in other areas. Different from static games, evolutionary games introduce replicator dynamics, which shows that the growth rate of the proportion of players using a certain strategy is equal to the difference between the average payoff of that strategy and the average payoff of the whole population. Replicator dynamics can also be used as a learning algorithm to study the behavior of multiple agents (Tuyls and Parsons 2007).

As in EDAs, MRAS, and CE, we maintain a population of solutions. The main idea of our method is to formulate the global optimization problem as an evolutionary game and to use dynamics in evolutionary game theory to study the evolution of the candidate solutions. The process of searching for the optimal solution is carried out through the procedure of reaching the evolutionary stable strategy (ESS). Specifically, we establish a connection between evolutionary game theory and optimization problems by dividing the solution region of the global optimization problem and letting different players play strategies in different subsets. Differential dynamics such as replicator dynamics is used to govern the evolution of the candidate solutions for the optimization problem. Furthermore, we introduce probabilistic models to generate candidate solutions and formulate the global optimization problem as an evolutionary game with continuous strategy spaces, based on which, a Model-based Evolutionary Optimization (MEO) algorithm is developed. Moreover, to better capture the multimodal property of global optimization problems, we propose to use a population of models to generate candidate solutions and a new Population Model-based Evolutionary Optimization (PMEO) algorithm is proposed, in which evolutionary game theory is used to study the evolution of these models and models with best performances will survive eventually. In preliminary numerical experiments, PMEO outperformed the standard CE method.

The way we formulate global optimization problems as evolutionary games provides new insights into the mechanism for generating new candidate solutions and the mechanism of model updating for model-based global optimization algorithms. For example, one special case of the MEO algorithm gives a new explanation for the CE method. This evolutionary game setting for global optimization problems makes it possible to study the convergence property of model-based algorithms by using analytical tools in the evolutionary game theory literature and it also provides new possibilities to develop new algorithms, for example, the PMEO algorithm developed in this paper.

2 EVOLUTIONARY GAME THEORY

Before presenting the main algorithm, we give a brief introduction to game theory and evolutionary game theory. Consider a two player game. A is the payoff matrix for player I and B is the payoff matrix for player II. Player I has the pure strategy set $S_1 = \{1, \cdots, n\}$ and $S_2 = \{1, \cdots, n\}$ is the pure strategy set for player II. The mixed strategy of player I is a probability vector $x = (x_1, \cdots, x_n)^T$ and x_i is the probability of choosing strategy $i \in S_1$. Similarly, B is the payoff matrix for player II with mixed strategy $y = (y_1, \cdots, y_n)^T$. If the game is symmetric, we have $B^T = A$. The expected payoff for player I and player II will be $x^T Ay$ and $x^T By$, respectively.

Evolutionary game theory studies the game in a population setting. Assume there is a population of agents which are programmed to play n different pure strategies in the set $\{1, \cdots, n\}$ and let x_i be the percentage of agents playing pure strategies i for $i \in \{1, \cdots, n\}$ in the population. We assume that

x_i is a differentiable function of time t. If individuals meet randomly and then engage in a symmetric game with a payoff matrix A, then $(Ax)_i$ is the expected payoff for an individual playing strategy i and $x^T Ax$ is the payoff of an agent that is randomly selected from the population. Let us assume that the per capita rate of growth, i.e. the logarithmic derivative $(\dot{\ln x_i}) := \dot{x}_i / x_i$, is given by the difference between the payoff for type i and the average payoff in the population. This yields the replicator equation (Weibull 1995)

$$\dot{x}_i = x_i((Ax)_i - x^T Ax) \qquad \forall i \in \{1, 2, \cdots, n\}.$$

Replicator dynamics is a selection process, according to which, more successful strategies will spread in the population.

2.1 Other Dynamics

Besides replicator dynamics, there are some other dynamics (Hofbauer and Sigmund 2003) and we introduce two as follows.

2.1.1 Imitation Dynamics

The imitation dynamics is given by $\dot{x}_i = x_i \sum_j [\phi_{ij}(x) - \phi_{ji}(x)] x_j$, where ϕ_{ij} is the rate at which an agent playing the strategy j adopts the strategy i. One plausible assumption is that this rate depends only on the payoffs achieved by the two agents, i.e $\phi_{ij}(x) = \phi(f_i, f_j)$, where f_i, f_j are payoff functions and $\phi(u, v)$ defines the imitation rule (the same for all players). The simplest rule is to imitate the better, i.e

$$\phi(u,v) = \begin{cases} 0 & \text{if } u \le v \\ 1 & \text{if } u > v \end{cases}.$$

In this case, the percentage of agents playing a strategy increases if and only if its payoff is larger than the median of the payoffs of all the strategies.

2.1.2 The Brown-von Neumann-Nash Dynamics

The other dynamics is the Brown-von Neumann Nash dynamics (BNN), which is defined as $\dot{x}_i = k_i(x) - x_i \sum_{j=1}^{M} k_j(x)$, where $k_i(x) = \max(0, f_i - \sum_{j=1}^{M} x_j f_j)$ denotes the positive part of the excess payoff for the strategy i. The discrete time version of the above dynamics is given by the following Nash map

$$x_i(t+1) = \frac{x_i(t) + k_i(x(t))}{1 + \sum_{j=1}^{M} k_j(x(t))}.$$

3 CONNECTING OPTIMIZATION AND EVOLUTIONARY GAME THEORY

Consider the following continuous optimization problem:

$$y^\star \in \arg\max_{y \in \mathscr{Y}} H(y) \tag{1}$$

where the solution space $\mathscr{Y} \in \Re^n$ is a nonempty set. The objective function $H(\cdot) : \mathscr{Y} \to \Re$ is a deterministic function bounded from above, i.e $\exists \mathscr{M}$ such that $H(y) \le \mathscr{M} \;\; \forall y \in \mathscr{Y}$. $y^\star$ is the global optimal solution if $H(y^\star) \ge H(y)\; \forall y \neq y^\star,\; y \in \mathscr{Y}$.

Assume that the solution space $\mathscr{Y}$ can be divided into M disjoint subsets $\mathscr{G}_1, \ldots, \mathscr{G}_M$. At each iteration, we plan to generate $N_1, \cdots, N_M$ candidate solutions using some random sampling algorithm in the corresponding subsets $\mathscr{G}_1, \ldots, \mathscr{G}_M$, where $\sum_{i=1}^{M} N_i = N$. Here we implicitly assume that the sampling in subsets $\{\mathscr{G}_i\}$ can be done by some sampling algorithm. Our goal is to sample more around the optimal solution $y^\star$, and thus increase the chance of finding the optimal solution. In other words, we want most of our samples concentrated around the optimal solution as we run our optimization algorithm. From an evolutionary game theory perspective, we view the samples (candidate solutions) as agents, which are programmed to play M different pure strategies $\{1,\cdots,M\}$. Here by playing the pure strategy i, we mean that an agent can sample a candidate solution in the subset $\mathscr{G}_i$. Assume that there are N_i agents playing the pure strategy i. An agent playing a pure strategy i will receive a payoff f_i no matter whom this agent is playing against.

The payoff f_i is defined as

$$f_i = \frac{1}{N_i}\sum_{j=1}^{N_i} H(y_{ij}),$$

where $\{y_{ij}, j=1,\cdots,N_i\}$ are candidate solutions generated from the subset $\mathscr{G}_i$. Define $x_i = \frac{N_i}{N}$, which is the percentage of agents playing the pure strategy i for all $i \in \{1,\cdots,M\}$. The evolution of the number of agents playing different strategies is governed by the replicator dynamics

$$\dot{x}_i = x_i(f_i - \sum_{j=1}^{M} x_j f_j). \tag{2}$$

From (2), it is easy to see that if the payoff of the strategy i is bigger than the average payoff, i.e $f_i > \sum_{j=1}^{M} x_j f_j$, the number of agents playing i will increase. From the viewpoint of simulation-based optimization, more samples will be assigned to the more promising area - the subset $\mathscr{G}_i$.

Note that replicator dynamics (2) is a differential equation. However our optimization algorithm is simulation-based; therefore we need a discretized version of replicator dynamics, which is given as follows. In matrix games, the discrete replicator dynamics is given by

$$x_i(t+1) = x_i(t)\frac{(Ax(t))_i + c}{x(t)^T Ax(t) + c},$$

where c is some constant to make sure that the denominator is not zero. In our setting for the global optimization problem, the corresponding replicator dynamics is

$$x_i(t+1) = x_i(t)\frac{f_i^t + c}{\sum_{j=1}^{M} x_j(t) f_j^t + c}, \tag{3}$$

where f_j^t is the payoff if the strategy j is adopted at time t. We can see from (3) that the percentage of agents playing each strategy changes by a fraction proportional to the averaged payoff of the corresponding strategy at each iteration. The percentage of the number of agents playing a strategy increases only if its payoff is greater than the average payoff and the amount of increase depends on the difference of the payoff of the particular strategy and the average payoff.

4 MODEL-BASED EVOLUTIONARY OPTIMIZATION

In Section 3, we partition the solution region into several subsets, and in the corresponding evolutionary game, there is only a finite number of strategies. The approach to partitioning the solution region and to generating candidate solutions in the resulting subsets can be critical to solving the global optimization problem and requires further investigation.

We take a step further and consider the partition in which each subset contains a single point in the solution region. Assume that there is a probabilistic model defined on the solution region, from which candidate solutions are generated. From the viewpoint of an evolutionary game, consider an evolutionary game with a continuous strategy space. Denote $\mathbb{P}_t$ a probability measure defined on the solution region $\mathscr{Y} \subseteq \Re^n$, which is also the strategy space of the game. Every single point $y \in \mathscr{Y}$ can be viewed as an agent that plays the pure strategy y. The percentage of the number of agents playing the pure strategy y at time t is $\mathbb{P}_t(dy)$. An agent playing the pure strategy y always obtains a fitness $H(y)$ no matter who else is playing the game. Different from evolutionary games with a finite number of pure strategies, the percentage of the number of agents adopting different strategies in the continuous game is described by the probability measure $\mathbb{P}_t$ defined on the strategy space $\mathscr{Y}$. By evolutionary game theory (Oechssler and Riedel 2002), the evolution of this probability measure is governed by some dynamics such as replicator dynamics. Let $\mathscr{A}$ be a measurable set in $\mathscr{Y}$. If the replicator dynamics with a continuous strategy space is adopted, we have

$$\dot{\mathbb{P}}_t(\mathscr{A}) = \int_{\mathscr{A}} (H(y) - E_{\mathbb{P}_t}[H(Y)])\mathbb{P}_t(dy), \tag{4}$$

where $E_{\mathbb{P}_t}[H(Y)]$ is the expectation of $H(Y)$ under the measure $\mathbb{P}_t$. From (4), we can see that if $H(y)$ outperforms $E_{\mathbb{P}_t}[H(Y)]$ at y, the probability measure around y will increase. However since the probability measure doesn't have a specific form, it would be very difficult to use (4) directly. If we assume that there is a probability density function p_t, such that $\mathbb{P}_t(dy) = p_t dy$, then (4) becomes

$$\dot{p}_t(y) = (H(y) - E_{\mathbb{P}_t}[H(Y)])p_t(y), \tag{5}$$

which governs the evolution of the probability density function on the continuous strategy space.

If we use $p_t(y)$ as our model to generate candidate solutions for the global optimization problem (1), the differential equation (5) can be used to update the model $p_t(y)$, with the final goal of making the probability density function $p_t(y)$ concentrated on a small set containing the global optimal solution. Then the global optimization problem can be easily solved by sampling according to the obtained probability density function.

4.1 General Model-based Evolutionary Optimization Algorithm

There are many dynamics in evolutionary game theory that can be used to govern the evolution of the probabilistic model $p_t(y)$. We use the following general representation to describe these dynamics,

$$\dot{p}_t(y) = D(H(y), E_{p_t}[H], p_t(y)), \tag{6}$$

where D is a function of $H(y), E_{p_t}[H]$, and $p_t(y)$. The corresponding discretized version is

$$p_{k+1}(y) = D_d(H(y), E_{p_k}[H], p_k(y)).$$

Based on the above analysis, we give the following Model-based Evolutionary Optimization (MEO) algorithm.

Model-based Evolutionary Optimization Algorithm

```
0.  Initialization. Choose $\rho \in (0,1]$ and an initial p.d.f /p.m.f $p_0$ defined on $\mathscr{Y}$. Let $k = 0$.
1.  Quantile calculation. Calculate the $1-\rho$ quantile $\gamma_k$,
        $$\gamma_k = \sup_l \{l : P_k(H(y) \geq l) \geq \rho\}.$$
    If $\gamma_k < \gamma_{k-1}$ and $k > 1$, let $\gamma_k = \gamma_{k-1}$. Let $k = k+1$ and go to step 2.
2.  Updating the probabilistic model.
        $$p_k(y) = D_d(H(y)I_{\{H(y)\geq\gamma_{k-1}\}}, E_{p_t}[HI_{\{H\geq\gamma_{k-1}\}}], p_{k-1}(y)).$$
3.  Stop if some stop criterion is satisfied; otherwise go to step 1.
```

The MEO algorithm requires the specification of a parameter ρ, which is the proportion of samples that will be used to update the probabilistic model; ρ also defines a sequence of $1-\rho$ quantiles $\{\gamma_k, k = 1, 2, \cdots\}$. These quantiles are used to obtain a sequence of nondecreasing thresholds that are used to select samples for model updating. In MEO, the form of the density p_k is not specified and it might be hard to generate candidate solutions from a general density p_k. The choice p_k is crucial to the MEO algorithm. Assume that we have a group of candidate solutions $\{y_t^i\}_{i=1}^N$ generated from p_t; then the probability density function p_t can be approximated as

$$\hat{p}_t(y) = \sum_{i=1}^N w_t^i \delta(y - y_t^i),$$

where δ denotes the Dirac function, and $\{w_t^i\}_{i=1}^N$ are weights satisfying $\sum_{i=1}^N w_t^i = 1$. If we use this approximation $\hat{p}_t$ as our probabilistic model, we can rewrite (6) as

$$\sum_{i=1}^{N} \frac{\partial w_t^i}{\partial t} \delta(y - y_t^i) = D\Big(H(y), \sum_{i=1}^{N} w_t^i H(y_t^i), \sum_{i=1}^{N} w_t^i \delta(y - y_t^i)\Big),$$

which is equivalent to

$$\frac{\partial w_t^i}{\partial t} = D\Big(H(y_t^i), \sum_{i=1}^{N} w_t^i H(y_t^i), w_t^i\Big). \tag{7}$$

The discrete time version of (7) is

$$w_{k+1}^i = D_d\Big(H(y_k^i), \sum_{i=1}^{N} w_k^i H(y_k^i), w_k^i\Big).$$

In particular, for replicator dynamics, we have

$$w_{k+1}^i = \frac{H(y_k^i)}{\sum_{i=1}^N w_k^i H(y_k^i)} w_k^i.$$

Although we have an updated density approximation $\hat{p}_{k+1} = \sum_{i=1}^N w_{k+1}^i \delta(y - y_k^i)$, it cannot be used to generate new candidate solutions directly. Hence we construct a new continuous density to approximate $\hat{p}_{k+1}$, which is done by projecting $\hat{p}_{k+1}$ onto some parameterized family of distributions g_θ. Specifically, we try to minimize the Kullback-Leibler (KL) distance between the parameterized distribution g_θ and $\hat{p}_{k+1}$:

$$\theta_{k+1} = \arg\min_{\theta \in \Theta} \mathscr{D}_{KL}\big(\hat{p}_{k+1} \| g_\theta\big), \tag{8}$$

where Θ is the domain of θ, and the KL distance is defined as

$$\mathscr{D}_{KL}\big(\hat{p}_{k+1}, g_\theta\big) = \int_{y \in \mathscr{Y}} \ln \frac{\hat{p}_{k+1}}{g_\theta} \hat{p}_{k+1} dy = \int_{y \in \mathscr{Y}} \ln \hat{p}_{k+1} \hat{p}_{k+1} dy - \int_{y \in \mathscr{Y}} \ln g_\theta \hat{p}_{k+1} dy.$$

Since the first term does not depend on the parameter θ, the minimization problem (8) is equivalent to

$$\max_{\theta \in \Theta} \int_{y \in \mathscr{Y}} \ln g_\theta \hat{p}_{k+1} dy,$$

which can be rewritten as

$$\max_{\theta \in \Theta} \sum_{i=1}^{N} w_{k+1}^i \ln g_\theta(y_k^i).$$

Based on the above analysis, a Monte Carlo simulation version of the MEO algorithm is given as follows.

Simulated Model-based Evolutionary Optimization Algorithm

0. Initialization. Let N be the total number of candidate solutions generated at each iteration. Choose $\rho \in (0,1]$ and an initial p.d.f /p.m.f p_0 defined on $\mathscr{Y}$. Let $k = 0$.
1. Quantile Calculation. Generate N candidate solutions $\{y_k^i\}_{i=1}^N$ from p_k. Calculate the $1-\rho$ quantile γ_k of $\{y_k^i\}_{i=1}^N$. If $\gamma_k < \gamma_{k-1}$ and $k > 1$, let $\gamma_k = \gamma_{k-1}$. Let $k = k+1$ and go to step 2.

2. Updating the probabilistic model. The discrete approximation of the model is $\hat{p}_k(y) = \sum_{i=1}^{N} w_k^i \delta(y - y_k^i)$, where

$$w_k^i = D_d\Big(H(y_{k-1}^i)I_{\{H(y_{k-1}^i)\geq\gamma_{k-1}\}}, \sum_{i=1}^{N}\frac{1}{N}H(y_{k-1}^i)I_{\{H(y_{k-1}^i)\geq\gamma_{k-1}\}}, 1/N\Big).$$

3. Density projection. Construct g_θ by projecting the density $\hat{p}_k = \sum_{i=1}^{N} w_k^i \delta(y - y_{k-1}^i)$ onto g_θ, where

$$\theta = \arg\max_{\theta\in\Theta}\sum_{i=1}^{N} w_k^i \ln g_\theta(y_{k-1}^i),$$

4. Stop if some stop criterion is satisfied; otherwise go to step 1.

Generally it is not easy to solve the optimization problem (8), which depends on the choice of g_θ. However for g_θ in the exponential family, analytical solutions exist.

Remark 1. *If replicator dynamics is used in step 2 in the Simulated Model-based Evolutionary Optimization Algorithm above, then in step 2, we have*

$$w_k^i = \frac{\frac{1}{N}H(y_{k-1}^i)I_{\{H(y_{k-1}^i)\geq\gamma_{k-1}\}}}{\sum_{i=1}^{N}\frac{1}{N}H(y_{k-1}^i)I_{\{H(y_{k-1}^i)\geq\gamma_{k-1}\}}}.$$

In step 3, we have

$$\theta = \arg\max_{\theta\in\Theta}\frac{\frac{1}{N}H(y_{k-1}^i)I_{\{H(y_{k-1}^i)\geq\gamma_{k-1}\}}}{\sum_{i=1}^{N}\frac{1}{N}H(y_{k-1}^i)I_{\{H(y_{k-1}^i)\geq\gamma_{k-1}\}}}\ln g_\theta(y_{k-1}^i).$$

Now the MEO algorithm becomes the same as the extended CE algorithm in Boer et al. (2005). In CE, the parameterized density function is chosen at the beginning and then the parameter is adaptively updated by an adaptively updated group of elite solutions. In MEO, the probabilistic model is estimated by some weighted Dirac functions and the evolution of the estimated density function is governed by replicator dynamics. Projecting the estimated density function onto a parameterized family of density functions is the final step in MEO. The Dirac function only gives a coarse approximation of the density function. One direction to improve MEO is to explore effective approximations of the density function based on generated samples.

5 POPULATION MODEL-BASED EVOLUTIONARY OPTIMIZATION

In the MEO algorithms given in Section 4, the density approximation $\hat{p}_k$ is inherently multimodal for global optimization problems with many local maxima. The projection of $\hat{p}_k$ onto a family of single mode density functions g_θ, for example, the exponential family, cannot fully capture this multimodal property. Motivated by the work of Hu et al. (2010), which generates candidate solutions from a group of models with the emphasis on optimization of budget allocation, we consider generating candidate solutions from a mixture distribution and focus on studying the evolving behavior of these models in the mixture distribution. We format the global optimization problem as an evolutionary game along the same lines as in Section 4.

5.1 Population Model and Evolutionary Game

Let $\Gamma = \{g_{\theta_1}, \cdots, g_{\theta_M}\}$ be a set of M parameterized probability density/mass functions on $\mathscr{Y}$. We assume that $\{g_{\theta_i}, i = 1, \cdots, M\}$ belong to some common parameterized distribution family $\{g_\theta, \theta \in \Theta\}$, where Θ is the parameter space. We propose to generate N samples from the following distribution:

$$g(y) = \sum_{i=1}^{M} w_i g_{\theta_i}(y), \tag{9}$$

where $w_i \geq 0$ and $\sum_{i=1}^{M} w_i = 1$, which is equivalent to generate $\lceil w_i N \rceil$ samples from g_{θ_i} for $i = 1, \cdots, M$ respectively. From an evolutionary game theory point of view, the action of playing the pure strategy i can be viewed as generating samples from g_{θ_i}; w_i is the percentage of the number of the agents that plays i at each iteration. Assume that the payoff of playing the pure strategy i is $\mathscr{I}_i$. Once we formulate the global optimization problem as an evolutionary game, we can use evolutionary game theory to study the evolution of the probability density function $g(y) = \sum_{i=1}^{M} w_i g_{\theta_i}(y)$. The evolution of $\{w_i, i = 1, \cdots, M\}$ is governed by the replicator dynamics

$$\frac{dw_i}{dt} = w_i(\mathscr{I}_i - \sum_{j=1}^{M} w_j \mathscr{I}_j). \tag{10}$$

It is easy to see from (10) that if $\mathscr{I}_i$, the performance of an agent playing the strategy i, is greater than the average performance $\sum_{j=1}^{M} w_j \mathscr{I}_j$, more agents will play the strategy i. From the viewpoint of optimization, we will sample more points from the more promising model. The evolution of the weights w_i can be viewed as an evolution of the balance between exploration and exploitation when searching for optimal solutions. The discrete time version of (10) is

$$w_i^{k+1} = w_i^k \frac{\mathscr{I}_i}{\sum_{j=1}^{M} w_j \mathscr{I}_j}.$$

5.2 Updating of the Population of Probability Density Functions

From evolutionary game theory, we know how the percentage of agents taking different pure strategies evolves. We also want to update the probability density functions $\{g_{\theta_i}, i = 1, \cdots, M\}$. There are many different ways to do this (Zhang and Mühlenbein 2004, Boer et al. 2005). Here we will adopt the one in Hu, Fu, and Marcus (2007). Given the threshold value γ, we define the exact performance function for probability density function g_{θ_i} as

$$\mathscr{I}_i(\gamma) = E_{\theta_i}[H(y) I_{\{H(y) \geq \gamma\}}], \quad \forall i = 1, \cdots, M.$$

Consider the following reference distributions at step $k+1$

$$h_i^{k+1}(y) = (1-\lambda) \frac{H(y) I_{\{H(y) \geq \gamma\}} g_{\theta_i^k}(y)}{\mathscr{I}_i(\gamma)} + \lambda g_{\theta_i^k}(y), \qquad \forall i = 1, \cdots, M,$$

where $\lambda \in (0,1)$ is a smoothing parameter.

However, it is not easy to sample from h_i^{k+1}; therefore we adopt the following updating procedure: projecting h_i^{k+1} onto some parameterized distribution family g_θ. Specifically, we will try to minimize the Kullback-Leibler (KL) distance between the parameterized distribution g_θ and h_i^{k+1}. Then we have

$$\theta_i^{k+1} = \arg\min_{\theta \in \Theta} \mathscr{D}_{KL}\big(h_i^{k+1} \| g_\theta\big), \quad \forall i = 1, \cdots, M. \tag{11}$$

The KL distance is defined as

$$\mathscr{D}_{KL}\big(h_i^{k+1} \| g_\theta\big) = \int_{y \in \mathscr{Y}} \ln \frac{h_i^{k+1}}{g_\theta} h_i^{k+1} dy.$$

5.2.1 Projection of Density Function

The exponential family contains a broad class of distributions, such as the Gaussian and binomial. We can obtain closed form solutions for θ_i^{k+1} when updating the distributions in the previous step by using the exponential family. For continuous optimization problems, it is convenient to use multivariate Gaussian distributions with independent components. At iteration k, assume that the parameterized distribution has the following form:

$$g_{\theta_i^k}(y) = \prod_{d=1}^{n} \frac{1}{\sqrt{2\pi(\sigma_{i,d}^k)^2}} \exp\Big(-\frac{(y_d-\mu_{i,d}^k)^2}{2(\sigma_{i,d}^k)^2}\Big),$$

where n is the problem dimension and y_d is the dth element of y.

Note that our algorithm is a simulation-based optimization algorithm, and we use stochastic counterparts to estimate expectations for random variables. Assume that we generate $N_i^k = \lceil w_i^k N \rceil$ samples $\{y_{ij}^k, j=1,\cdots,N_i^k\}$ from $g_{\theta_i^k}$ for all $i=1,\cdots,M-1$, and $N-\sum_{i=1}^{M-1} N_i^k$ samples from $g_{\theta_M^k}$, and calculate the performances $H(y_{ij}^k)$. Then $\mathscr{I}_i(\gamma)$ can be estimated by

$$\hat{\mathscr{I}}_i^k(\gamma) = \frac{1}{\sum_{j=1}^{N_i^k} I_{\{H(y_{ij}^k)\geq\gamma\}}} \sum_{j=1}^{N_i^k} H(y_{ij}^k) I_{\{H(y_{ij}^k)\geq\gamma\}}. \tag{12}$$

Define the "elite" sets $\mathscr{L}_i^k = \{y_{ij}^k : H(y_{ij}^k) \geq \gamma\}$. Use $\hat{\mathscr{I}}_i$ to approximate $\mathscr{I}_i$ and solve the optimization problem (11). We obtain

$$\begin{aligned}\mu_{i,d}^{k+1} &= (1-\lambda)\frac{\sum_{y\in\mathscr{L}_i^k} H(y) y_d}{\sum_{y\in\mathscr{L}_i^k}(H(y))} + \lambda\mu_{i,d}^k \qquad (13)\\ (\sigma_{i,d}^{k+1})^2 &= (1-\lambda)\frac{\sum_{y\in\mathscr{L}_i^k} H(y)(y_d-\mu_{i,d}^{k+1})^2}{\sum_{y\in\mathscr{L}_i^k}(H(y))} + \lambda\big((\sigma_{i,d}^k)^2 + (\mu_{i,d}^{k+1}-\mu_{i,d}^k)^2\big)\end{aligned}$$

for all $d=1,\cdots,n$ and $i=1,\cdots,M$.

5.3 PMEO Algorithm

Based on the above analysis, we give the following Population Model-based Evolutionary Optimization (PMEO) algorithm.

Population Model-based Evolutionary Optimization Algorithm

0. Initialization: Let N be the number of total samples at each iteration. Specify the weights $\{w_i^0, i=1,\cdots,M\}$ and the probability density function $\{g_{\theta_i^0}, i=1,\cdots,M\}$. Let ρ_0 be the quantile parameter. Set $k=0$ and $\gamma=-\infty$.
1. Generate $N_i^k = \lceil w_i^k N \rceil$ samples $\{y_{ij}^k, j=1,\cdots,N_i^k\}$ from $g_{\theta_i^k}$ for all $i=1,\cdots,M-1$, and $N-\sum_{i=1}^{M-1} N_i^k$ samples from $g_{\theta_M^k}$; calculate the performances $H(y_{ij}^k)$. Order the performances from largest to smallest, $H_{(1)}\geq\cdots\geq H_{(N)}$. Let γ^k be the $(1-\rho)$ sample quantile of performances: $\gamma^k = H_{(\lceil(1-\rho)N\rceil)}$. If $\gamma^k>\gamma$, let $\gamma=\gamma^k$. Generate the "elite" sets $\mathscr{L}_i^k = \{y_{ij}^k : H(y_{ij}^k)\geq\gamma^k\}$ for all $i=1,\cdots,M$. Calculate $\hat{\mathscr{I}}_i^k$ by (12).
2. Let

$$w_i^{k+1} = w_i^k \frac{\hat{\mathscr{I}}_i^k}{\sum_{j=1}^{M} w_j \hat{\mathscr{I}}_j^k}.$$

And update the parameter θ_i^{k+1} according to (13) for $i = 1, \cdots, M$.

3. If a stopping rule is met, then stop; otherwise set $k = k+1$ and go to step 1.

Note that we consider maximization problems in the above algorithm, which can be easily adjusted for solving minimization problems.

6 NUMERICAL EXAMPLES

We use some benchmark examples to demonstrate the effectiveness of the PMEO algorithm. We consider minimizing the following objective functions.

E1. Dejong's 5th function($n = 2$).

$$H_1(y) = \left[0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(y_i - a_{j,i})^6}\right]$$

where
$a_{j,1} = \{-32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0,$
$16, 32, -32, -16, 0, 16, 32\}$;
$a_{j,1} = \{-32, -32, -32, -32, -32, -16, -16, -16, -16, -16, 0, 0, 0, 0, 0, 16, 16, 16,$
$16, 16, 32, 32, 32, 32, 32, \}$

E2. Rosenbrock function ($n = 20$).

$$H_2(y) = \sum_{i=1}^{n-1} 100(y_{i+1} - y_i^2)^2 + (y_i - 1)^2,$$

E3. Powell singular function ($n = 20$).

$$H_3(y) = \sum_{i=1}^{n-2} \left[(y_{i-1} + 10y_i)^2 + 5(y_{i+1} - y_{i+2})^2 + (y_i - 2y_{i+1})^4 + 10(y_{i-1} - y_{i+2})^4\right]$$

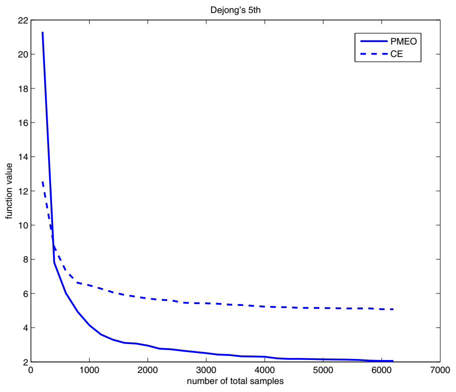
E4. Griewank function ($n = 20$).

$$H_4(y) = 1 + \sum_{i=1}^{n} 8\sin^2(7(y_i - 0.9)^2) + 6\sin^2(14(y_i - 0.9)^2) + (y_i - 0.9)^2.$$

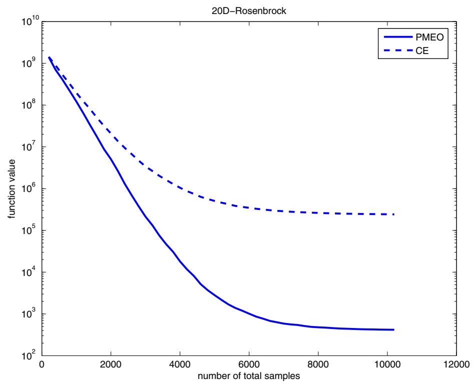
In this experiment, for the exponential family, we use multivariate normal distributions with independent components. For CE, each element of the initial mean vector is uniformly selected from $[-50, 50]$, and the covariance matrix is a diagonal matrix with each diagonal element equal to 500. For PMEO, the initial mean and covariance matrices are chosen the same way as in the CE method. For the mixture distribution in PMEO, we set $M = 5$. At each step, we generate $N = 200$ samples. We run the algorithm 100 independent times. Figure 1 to Figure 4 show the comparison between the CE method and the PMEO method. We can see that PMEO has comparably better performance than CE given the same computational effort.

7 CONCLUSION

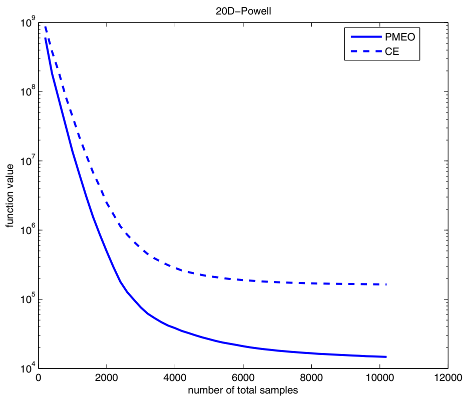
We establish a connection between global optimization problems and evolutionary games by formulating the global optimization problem as an evolutionary game. Based on this connection, we propose a model-based evolutionary optimization (MEO) algorithm, which includes the extended CE algorithm as an instantiation. We also propose a population model-based evolutionary optimization algorithm, which better captures the multimodal property of global optimization problems. Simulation results show the effectiveness of the proposed method.



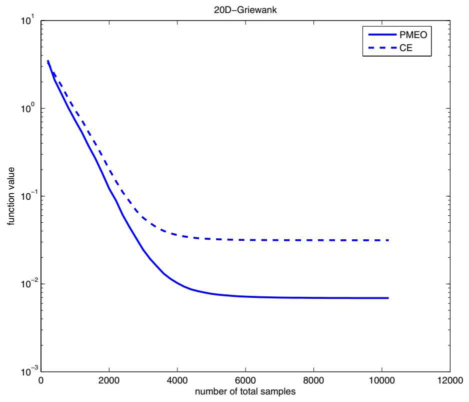
(a) H_1



(b) H_2



(c) H_3



(d) H_4

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (NSF) under Grants DMI-0540312, CNS-0926194, CMMI-0856256, EECS-0901543, and by the Air Force Office of Scientific Research (AFOSR) under Grants FA9550-07-1-0366 and FA9550-10-1-0340.

REFERENCES

Boer, P. T. D., D. P. Kroese, S. Mannor, and R. Y. Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of Operation Research* 134:19–67.

Fudenberg, D., and D. Levine. 1998. *The theory of learning in games*. The MIT Press.

Garcia, A., S. D. Patek, and K. Sinha. 2007. A decentralized approach to discrete optimization via simulation: Application to network flow. *Operations Research* 55 (4): 717–732.

Hofbauer, J., and K. Sigmund. 2003. Evolutionary game dynamics. *Bulletin (New Seires) of the American Mathematical Society* 40 (4): 479–519.

Hu, J., H. Chang, M. Fu, and S. Marcus. 2010. Dynamic sample budget allocation in model-based optimization. *Journal of Global Optimization*:1–22.

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

Lambert, T. J., M. A. Epelman, and R. L. Smith. 2005. A fictitious play approach to large-scale optimization. *Operations Research* 53 (3): 477–489.

Mannor, S., R. Y. Rubinstein, and Y. Gat. 2003. The cross-entropy method for fast policy search. *International Conference on Machine Learning*:512–519.

Mühlenbein, H., and G. Paaß. 1996. From recombination of genes to the estimation of distributions: I. binary parameters. In *Parallel Problem Solving from Nature - PPSN IV*, ed. I. R. Hans-Michael Voigt, Werner Ebeling and H.-P. Schwefel, 178–187. Springer.

Oechssler, J., and F. Riedel. 2002. On the dynamics foundation of evolutionary stability in continuous models. *Journal of Economic Theory* 107:223–252.

Rubinstein, R. Y. 1997. Optimization of computer simulation models with rare events. *European Journal of Operations Research* 99:89–112.

Rubinstein, R. Y. 1999. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability* 2:127–190.

Shoham, Y., R. Powers, and T. Grenager. 2004. Multi-agent reinforcement learning: a critical survey. In *AAAI Fall Symposium on Artificial Multi-Agent Learning*. Citeseer.

Smith, J. M. 1982. *Evolutionary and the theory of games*. Cambridge Univ. Press.

Tuyls, K., and S. Parsons. 2007. What evolutionary game theory tells us about multiagent learning. *Artificial Intelligence* 171 (7): 406–416.

Weibull, J. 1995. *Evolutionary game theory*. The MIT Press.

Zhang, Q., and H. Mühlenbein. 2004. On the convergence of a class of estimation of distribution algorithm. *IEEE Trans. on Evolutionary Computation* 8:127–136.

Zhou, E., M. Fu, and S. Marcus. 2008. A particle filtering framework for randomized optimization algorithms. In *the Winter Simulation Conference*, 647–654.

AUTHOR BIOGRAPHIES

YONGQIANG WANG is a Ph.D. student in the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland. He received the B.S. degree (with highest honor) in electrical engineering and M.E. degree from Zhejiang University in 2003 and 2006, respectively. He has been selected as a Future Faculty Fellow by the A. J. Clark School of Engineering at the University of Maryland. His research interests lie in the areas of simulation-based optimization, Markov decision process, and stochastic control, with applications towards supply chain management and financial engineering. His email address and personal website are <yqwang@umd.edu> and <www.ece.umd.edu/˜ yqwang>.

MICHAEL C. FU is Ralph J. Tyser Professor of Management Science in the Robert H. Smith School of Business, with a joint appointment in the Institute for Systems Research and an affiliate appointment in the Department of Electrical and Computer Engineering, all at the University of Maryland. He received degrees in mathematics and EE/CS from MIT, and an M.S. and Ph.D. in applied mathematics from Harvard University. His research interests include simulation and applied probability modeling, particularly with applications towards manufacturing systems, supply chain management, and financial engineering. He has served as Stochastic Models and Simulation Department Editor of *Management Science* and as Simulation Area Editor of *Operations Research*. He is co-author (with J.Q. Hu) of the book, *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, which received the INFORMS Simulation Society's Outstanding Publication Award in 1998. He is a Fellow of INFORMS and IEEE. His email address is <mfu@umd.edu>.

STEVEN I. MARCUS is a Professor in the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland. He received the B.A. degree in electrical engineering and mathematics from Rice University in 1971, and the S.M. and Ph.D. degrees in electrical engineering from MIT in 1972 and 1975, respectively. From 1975 to 1991, he was with the Department of Electrical and Computer Engineering, University of Texas, Austin, where he was the L.B. (Preach) Meaders Professor in Engineering. At the University of Maryland, he has served as Director of the Institute for Systems Research from 1991-1996, and as Chair of the Electrical and Computer Engineering Department from 2000-2005. He is former Editor-in-Chief of the *SIAM Journal on Control and Optimization*. Currently, his research is focused on stochastic control and estimation, Markov decision processes, and hybrid systems, with applications in manufacturing and telecommunication networks. He is a Fellow of IEEE and SIAM. His email address is <marcus@umd.edu>.