

A modified probability collectives optimization algorithm based on trust region method and a new temperature annealing schedule

Bo Yang · Ruiming Wu

Published online: 3 March 2015
© Springer-Verlag Berlin Heidelberg 2015

Abstract This article presents a distributed random search optimization method, the trust region probability collectives (TRPC) method, for unconstrained optimization problems without closed forms. Through analyzing the framework of the original probability collectives (PC) algorithm, three potential requirements on solving the original PC model are first identified. Then an interior point trust region method for bound constrained minimization is adopted to satisfy these requirements. Besides, the temperature annealing schedule is also redesigned to improve the algorithmic performance. Since the new annealing schedule is linked to the gradient, it is much more flexible and efficient than the original one. Ten benchmark functions are used to test the modified algorithm. Numerical results show that TRPC is superior to the PC algorithm in iteration times, accuracy, and robustness.

Keywords Random search optimization · Probability collectives · Trust region

1 Introduction

Probability collectives (PC) algorithm is a distributed random search optimization method. It is an extension and formalization of the collective intelligence (COIN) framework with deep connections to game theory, statistical physics, and optimizations for modeling and controlling of the distributed

multi-agents system (MAS) (Bieniawski and Stefan Richard 2005; Wolpert 2006; Kulkarni et al. 2008; Lee et al. 2004).

Insights of PC mainly focus on two points. The first one is that PC is a distributed algorithm. It means each agent independently improves the local objective and updates its own actions according to rewards from the previous round. This process will continue until no further increase in rewards could be made for an individual agent, which, from the perspective of game theory, is exactly the Nash Equilibrium. The second point is that PC algorithm allocates probability values among the possible strategy set instead of operating on variable values. The strategies with higher probability will have more opportunities to show up in the next round, and those with lower values are abandoned in terms of probability. So PC algorithm will ultimately converge to a distribution which has a high probability value (≈ 1) near the optimal strategy, while assigning the other feasible regions a relatively low probability value (≈ 0). In addition, besides the optimal solution, PC also provides a ‘slice’ of the multidimensional objective function, which describes the variation trend of the function. These main features make PC algorithm quite different from the other random search algorithms such as genetic algorithm (GA), simulated annealing (SA) and swarm optimization and allow it to solve problems with discrete, continuous and mixed variables (Kulkarni et al. 2009).

PC process shares the same spirit with various methods such as mutual-information-maximizing input clustering (MIMIC), population-based increased learning (PBIL) and distributed reinforcement learning (DRL). The similarity among these methods is that they all take learning processes and use probability distributions to guide the optimizing procedure instead of operating directly on the variable values. However, each method still maintains its own characteristics. PC uses gradient-based optimization methods and function

Communicated by V. Loia.

B. Yang · R. Wu (✉)
Antai College of Economics and Management,
Shanghai Jiao Tong University, Shanghai 200052, China
e-mail: dwurm@sjtu.edu.cn

B. Yang
e-mail: moriarty4869@sjtu.edu.cn

values to adjust the probability distribution. In comparison, MIMIC directly uses truncated samples and PBIL adopts the idea of GA which simulates the evolution of a population to find the final solution. Besides, PC pays more attention on how individual can influence system's performance with the presence of the other agents than the DRL does and thus is particularly suitable for distributed problems (Kulkarni and Tai 2010a; Kulkarni et al. 2010b; Autry and Brian 2008; Busoniu et al. 2008).

The PC algorithm is first presented by Wolpert and Bieniawski (2004). Since then, lots of modifications and applications have already been developed. Lee et al. (2004) replace the world utility (called Team Game) with private utility (called Aristocrat Utility) to cut down the sample size used in the original algorithm with no bias and low variance. Bieniawski and Stefan Richard (2005) show that the data aging technique can be used to reduce the sample size effectively. Kulkarni and Kang modify the sampling rule on original Monte-Carlo by narrowing the sampling region around the current optimal points (Kulkarni et al. 2008; Lee et al. 2004). Some updating strategies, such as steepest decent method, nearest Newton method and Brower fixed point method have also been developed (Wolpert et al. 2011). The BFGS method is used to optimize the model by Kulkarni et al. (2011). Besides these, Wolpert et al. (2011) classify the original PC as 'delayed sampling PC' and put forward another 'immediate sampling PC' using the importance sampling and parametric machine learning technique.

There are also some applications of the PC algorithm. Numerical results on a set of benchmark functions show that PC method outperforms the traditional GA algorithm in the rate of descent (Huang et al. 2005). PC is also used to improve the Metropolis-Hastings sampling and mechanism design work (Wolpert and Tumer 1999, 2001, 2002). More widely, it has also been used in some combinatorial problems such as multi-depot multiple traveling salesmen problem (MTSP) (Kulkarni and Tai 2010a; Kulkarni et al. 2010b), the single-depot MTSP (Kulkarni and Tai 2010a; Kulkarni et al. 2010b), the fleet assignment problem (Antoine et al. 2004), school table scheduling problems (Autry and Brian 2008) and vehicle routing problems (Kulkarni and Tai 2010a; Kulkarni et al. 2010b).

However, following Wolpert's PC algorithm which is based on the random sampling, some potential improvements still could be figured out. First of all, it is noticed that the gradient derived from the 'Maxent Lagrangian' model directly influences the effect of the original algorithm. In fact, since the gradient contains both the expectation ($E(G)$) and conditional expectation ($E(G|x)$) of the function value, which are estimated through Monte-Carlo Sampling, the original algorithm may not always guarantee a satisfying result due to the estimation error. For example, if the authentic value of the expectation ($E(G)$) is 1,000 and the nearest Newton method

is used to update the probability, one percent estimation error from this number may generate an unbearable outcome, since the error will hugely change the descent direction and destroy the basis of the iteration. Second, the calculation of step size along the descent direction can also be a big challenge for the PC model, because choosing the best step size needs to update the probability and use MC sampling to evaluate the function value repeatedly. Finally, the temperature T is vital to the PC algorithm, so a fixed proportional decrease may not be a satisfying schedule.

With respect to the above analysis, a mixed PC algorithm is presented in this paper, which combines the trust region (TR) method and the PC algorithm. A new adaptive annealing schedule is also adopted to replace the original one. Numerical experiments show that the performance of the proposed algorithm outperforms the original PC algorithm.

The article is organized as follows: in Sect. 2, the original PC algorithm is first reviewed. Then the proposed PC algorithm is elaborated in Sect. 3. And in Sect. 4, the performance of the new algorithm is illustrated by the results of 10 benchmark functions and comparisons with another modified PC algorithm are also made. Finally, conclusions are given in Sect. 5 and future researches are also discussed.

2 The original PC algorithm

2.1 The framework

Different from the conventional 'black box' optimization methods, PC algorithm focuses on the distribution over a strategy set of each variable instead of the variable values. Through adapting distributions, the original problem is inverted to a new problem over a convex space of probability distributions with variables taking real values (probability values) on their feasible regions. Thus powerful deterministic optimization tools could be involved to resolve this problem.

In PC algorithm, variables can be viewed as *intelligent agents*. Every agent independently selects actions from its own particular intervals, gets local rewards based on the objective function per iteration and decides its next round of action according to the previous rewards. The actions that make greater contributions to the global optimization will get higher probability values and could be more likely to appear in the next round while those making little or no contributions will have lower probability and finally disappear from the optimal strategy set in terms of probability. As this iterative process continues, the system will ultimately reach an equilibrium which means the objective function can not be further improved through allocating the probability among strategies and the probability distribution has already converged to the optimal one. At this time, the optimal solution

is the combination of actions with the highest probability in each agent.

Another way to interpret this PC theory is through Game Theory. If the final equilibrium is viewed as a direct choice of a group of fully rational agents, then the whole PC process could be viewed as a bounded rational game: through a learning process, each agent explores both possible moves and payoff matrix and then gradually converges towards the Nash Equilibrium. It is worth noting that during this process, actual moves of every agent are independent and all couplings between the agents occur indirectly; it is not their moves but the separate distributions of the agents that are coupled.

2.2 The original models

Consider the following unconstrained optimization problem:

$$\min_x G(x), \tag{1}$$

where $G(x)$ is the objective function and $x = (x_1, x_2, \dots, x_n)$ is the variable/agent vector.

Let q_i be a distribution over the i th agent’s strategy set. The optimization problem can be seen as a non-cooperative game with n independent players. Each player has its own strategy set in the game and the goal is to reach the Nash equilibrium, so the problem can also be described as

$$\min_q E_q[G] = \int G(x) \prod_i q_i(x_i) dx \tag{2}$$

The subscript in the expectation suggests its value depends on the distribution $q = \prod q_i$.

However, as mentioned above, the full rationality assumption in game theory usually cannot be satisfied. It means the optimal probability distribution cannot be found directly but can only be approximated step by step. In other words, the uncertainty of the final distribution is decreased gradually during the game. So if the Shannon entropy is employed here to measure the uncertainty of the distribution q , it satisfies

$$S(q) = -q(x) \ln q(x) \tag{3}$$

and the entropy should gradually decrease as the objective function approximates the minimum. Note that according to the maximum entropy principle, distribution q should keep the minimal amount of extra information beyond the prior knowledge; thus the entropy should be the largest and equality is used instead of inequality.

So the original problem can also be described as follows:

$$\begin{aligned} \min_q E_q[G] &= \int G(x) \prod_i q_i(x_i) dx \\ \text{s.t. } S(q) &= s \\ \int q_i(x_i) dx_i &= 1 \\ q_i(x_i) &\geq 0 \end{aligned} \tag{4}$$

where s is the *prior knowledge* about the distribution.

The last two constraints are to keep the distribution normalized and will be satisfied through iterations. Using the Lagrange multiplication operator, we can get

$$\min_q L(q, T) = \int G(x) \prod_i q_i(x_i) dx - T(S(q) - s), \tag{5}$$

where T is a Lagrange parameter, also referred to as the *temperature*, which is fixed during one iteration and decreases as this iteration finishes. The objective function (5) is called Maxent Lagrangian and widely used in statistical physics, where it is referred to as *free energy*.

Since q is a product joint distribution, the entropy could be written as a summation form. So minimizing the local/private objective function for each agent will simultaneously minimize the global problem. Thus the objective function could be further written as

$$\begin{aligned} \min_{q_i} L(q_i, T_i) &= \int G(x) \prod_j q_j(x_j) dx - T_i(S(q_i) - s_i), \\ i &= 1, 2, \dots, n \end{aligned} \tag{6}$$

Here the private objective functions are set to equal the global objective function, which is called Team Game (TG). A natural thought of solving this problem is to gradually decrease s_i and find a certain distribution that matches $S(q_i) = s_i$ in each time. Then the distribution will automatically approximate the optimal solution. When s_i becomes sufficiently small, the unique distribution that minimizes the problem can also be found. However, in [Wolpert and Lee \(2004\)](#), temperature T is shown to be non-decreasing with s_i , which means s_i decreases as T decreases. Besides, the process of decreasing temperature is much easier than the one decreasing s_i . So objective (6) can be equally shown as the formula below:

$$\begin{aligned} \min_{q_i} L(q_i, T_i) &= \int G(x) \prod_j q_j(x_j) dx - T_i S(q_i), \\ i &= 1, 2, \dots, n \end{aligned} \tag{7}$$

In statistical physics, equations above are also called Boltzmann equations and solutions to them are

$$q_i(x_i) \propto \exp\left(-\frac{E_{(q_i)}(G(x)|x_i)}{T}\right), \quad i = 1, 2, \dots, n,$$

which is called the Boltzmann distribution. The subscript in the conditional expectation means its value depends on a joint distribution without q_i . It reflects that the probability of agent i choosing pure strategy x_i depends on the effect of that choice on the utilities of the other agents.

3 Trust region PC method (TRPC)

The establishment of the Maxent Lagrangian is excellent, since it successfully converts the primal problem into the probability space and the rest of the work is to solve formula

(7). However, not all gradient-based optimization methods are suitable to this model and some potential requirements should be satisfied. So, in this section, these requirements are first identified and then the new PC algorithm is presented.

3.1 Potential requirements

Consider formula (7), we can get

$$\frac{\partial L}{\partial q_i(x_i = j)} = E[G|x_i = j] + T_i[1 + \ln q_i(j)] \quad (8)$$

So gradient-based methods could be used to find the optimal q^* if a fixed *temperature* T is given. Since the closed form of $G(x)$ is not known, here Monte–Carlo sampling is used to estimate the conditional expectation value ($E[G|x_i = j]$) in Eq. (8).

However, the adoption of Monte–Carlo sampling will bring some critical problems that should be considered. First of all, the optimization method should be an interior point method, because in each iteration the values we update are probability values and they are used to perform Monte–Carlo sampling and calculate the entropy. So, the optimization method should automatically guarantee the probability value within $[0, 1]$. (Standardization may not be necessary, it will be discussed later.) Second, the calculation of the step size should be avoided, which can be extremely costly in this model. For example, if a descent direction is given and two step sizes are compared, we need to use them to update probability distributions, sample under the new distributions and evaluate the new function values. When the step size is a real number, this whole method becomes unpractical. Third, since the gradient of L contains conditional expectation ($E[G|x_i = j]$) which may have random errors in practice, the algorithm should not only guarantee an effective descent in each iteration but should also show tolerance to the randomness and estimation error.

Former PC algorithms did not fully solve all of these problems. In the most widely used *Nearest Newton Method*, the interior point requirement is satisfied through the iterative step

$$q_i(j) \leftarrow q_i(j) - \alpha q_i(j) \times \{E(g_i|x_i = j) - E(G) + T[S(q_i) + \ln q_i(j)]\} \quad (9)$$

It can be easily checked that $\sum_j q_i(j) \times \{E(g_i|x_i = j) - E(G) + T[S(q_i) + \ln q_i(j)]\} = 0$. So it seems that the summation of the probability should always equal to one in every iteration if the algorithm starts from a distribution like the uniform distribution. However, this iterative step works well only when the error of estimation $E(G|x_i = j)$ is small, which can hardly be guaranteed in iterations especially in the very beginning. Besides that this equation also can not guarantee every new $q_i(j)$ to be strictly positive during iteration. Though the negative $q_i(j)$ can be set to zero, in an

extremely bad situation there exists possibilities that all of $q_i(j)$ s become negative through this updating. What is worse, the directly setting method could also make the summation of probability smaller than 1, which could influence both the MC sampling and the entropy. Finally, it seems that the parameter α plays a role like step size to ease the error of estimation and keep the iteration on its track. But a small α will also shrink the effect of descent along a correct direction.

Given all of these problems, here a Trust Region method is used to solve the Maxent Lagrangian. Before introducing details, a slight modification of model is needed. Note that the equation constraint

$$\int q_i(x_i) dx_i = 1, \quad i = 1, 2, \dots, n$$

may not be necessary, since the purpose is to set one or a few probability values to be large (≈ 1) while the others to be zero and this equation is needed only when the MC sampling is performed or the entropy is calculated. So the probability can be standardized just before these two calculations and kept within $(0, 1)$ at the other time. In other words, we only need to keep each probability value between 0 and 1 when solving the Maxent Lagrangian but standardize it just for MC sampling and entropy calculation. Thus this constraint can be loosened and the model can be changed to

$$\begin{aligned} \min_{q_i} L(q_i, T_i) &= \int G(x) \prod_j q_j(x_j) dx - T_i S(q_i), \\ & i = 1, 2, \dots, n \\ \text{s.t. } & 0 < q_i(x_i) < 1 \end{aligned} \quad (10)$$

3.2 The trust region probability collectives (TRPC)

Now the original problem has been transformed into a minimization problem subject to bounds and the new algorithm can be discussed in detail. Considering all of the requirements, we redesign the updating rule of T and introduce a modified trust region (TR) method dealing with bounded variables, which was first presented by Coleman et al. (1996), to solve the Maxent Lagrangian.

First, a vector $v(x)$ and an affine scaling matrix $D(x)$ are defined as follows:

Definition 1 Let $v(q_i) = (v_{i1}, v_{i2}, \dots, v_{in})$ be a vector for q_i , and then

$$v_{ij} = \begin{cases} q_{ij} - 1, & \text{if } \nabla L_{ij} < 0 \\ q_{ij}, & \text{if } \nabla L_{ij} \geq 0 \end{cases}$$

where q_{ij} is the probability value of the j th strategy.

Definition 2 For all $v(q_i)$, let

$$D(q_i) = \text{diag}(|v(q_i)|^{-1/2}),$$

where $\text{diag}(\cdot)$ denotes a diagonal matrix.

Based on these two definitions, the quadratic sub-problem in TR can be defined as follows. Here the result is given directly and more details are presented in Appendix A.

$$m_k(s) = g_k^T s + \frac{1}{2} s^T M_k s, \tag{11}$$

where

$$g_k = D_k^{-1} \nabla L_k = \text{diag}(|v(q_i)|^{-\frac{1}{2}})$$

$$M_k = D_k^{-1} B_k D_k^{-1} + \text{diag}(\nabla L_k) J_k$$

J_k plays a role of the *Jacobian* matrix of $|v(q_i)|$ to allow for a second-order Newton process (Coleman et al. 1994). We set $J_k = \text{diag}(\text{sign}(\nabla(L_k)))$.

Following (11), it can be easily checked that the statements below hold.

- (a) $g_k = 0$ iff the first-order necessary conditions for form (10) are satisfied.
- (b) M_k is positive definite and $g_k = 0$ iff the second-order sufficiency conditions are satisfied.
- (c) M_k is positive semi-definite and $g_k = 0$ iff the second-order necessary conditions are satisfied.

So q_i will be a local minimum of model (10) iff $s = 0$ is a solution to the quadratic sub-problem

$$\min_s \left\{ m_k(s) = g_k^T s + \frac{1}{2} s^T M_k s : \|s\| \leq \Delta_k \right\} \tag{12}$$

Hence solving the sub-problem (12) is a reasonable step to attempt when q_i is not a local minimizer.

Here, a so-called *dog-leg* method is used to approximately solve the problem and it can be checked that solutions under this method satisfy the assumptions for sufficient reductions which can guarantee the convergence of the algorithm. Still, the result is directly given here and more details are shown in Appendix B.

Here, three cases are considered:

- (a) If the trust region radius is sufficiently large ($\|s^B\| = \|-M_k^{-1} g_k\| \leq \Delta_k$), s^B is exactly the solution.
- (b) If the trust region radius is rather small ($\|s^U\| = \left\| -\frac{g_k^T g_k}{g_k^T M_k g_k} g_k \right\| \geq \Delta_k$), $\|\alpha s^U\| = \Delta_k$ could be the solution.
- (c) Otherwise, the solution should be a combination of s^B and s^U , namely $s^U + (\tau - 1)(s^B - s^U)$, and τ could be solved by $\|s^U + (\tau - 1)(s^B - s^U)\| = \Delta_k$.

So the whole TRPC algorithm can be described as follows:

Step 1 Check the stop criteria. If $\sum_j \|q_{ij}^k - q_{ij}^{k-1}\| \leq \eta$, then stop the algorithm and the current distribution is the optimal solution. Otherwise, execute the next step.

Step 2 Use Monte–Carlo sampling method to sample from current distribution q^k in the k th iteration and estimate the expectation value $E(G(x))$ and $E(G|x_i)$.

Step 3 According to $E(G|x_i)$, calculate T by the annealing schedule for each agent, and calculate the gradient.

Step 4 Use BFGS method to update B_k as an approximation to the Hessian matrix and obtain s_k by (approximately) solving the sub problem (12). r_k can be evaluated from

$$r_k = \frac{L_k - L(q_k + s_k)}{m_k(s_k)}$$

Then,

- If $r_k < \varepsilon$, set $\Delta_{k+1} = \|s_k\|/4$;
- if $r_k > 0.75$ and $\|s_k\| = \Delta_k$, set $\Delta_{k+1} = 2\|s_k\|$;
- otherwise, set $\Delta_{k+1} = \Delta_k$.

To get an approximation of the exact s_k^* in the sub-problem by *dog-leg* method, define

$$s^B = -M_k^{-1} g_k, \quad s^U = -\frac{g_k^T g_k}{g_k^T M_k g_k} g_k$$

and choose s^B as s^* if $\|s^B\| \leq \Delta_k$. Otherwise, we replace s^* with the following form:

$$\tilde{s}_k(\tau) = \begin{cases} \tau s^U & 0 \leq \tau \leq 1 \\ s^U + (\tau - 1)(s^B - s^U) & 1 \leq \tau \leq 2, \end{cases}$$

where τ is solved through $\|\tilde{s}_k(\tau)\| = \Delta_k$.

Note that, it is necessary to generate another sample set to evaluate $L(q_k + s_k)$.

Step 5 If $r_k > 0$, then $q_{k+1} = q_k + s_k$. Otherwise, let $q_{k+1} = q_k$, and go to Step 1.

3.3 Annealing

Temperature T plays an important role in the performance of PC for balancing the full rational (the expectation) and irrational (the entropy) parts in the Maxent Lagrangian. Like in SA, it is also necessary to gradually decrease T in PC to reach the optimal point. Since a quickly decreasing T may lead to local optimum while a relatively slow reduction may cause too many samplings and iterations, a proper annealing schedule is quite significant.

Traditional annealing schedule is a geometric schedule that decreases T proportionally by multiplying a fixed parameter. However, the temperature T used here is quite different from T used in the simulated annealing algorithm. Since the gradient contains T , it is possible to further describe relationships among the temperature, search scope and accuracy of the algorithm. Thus a much more flexible and efficient schedule can be presented.

Considering the structure of the gradient (8),

$$\frac{\partial L}{\partial q_i(x_i = j)} = E[G|x_i = j] + T_i[1 + \ln q_i(j)]$$

for the i th agent; the conditional expectation describes the payoff of its every strategy given the moves of the other

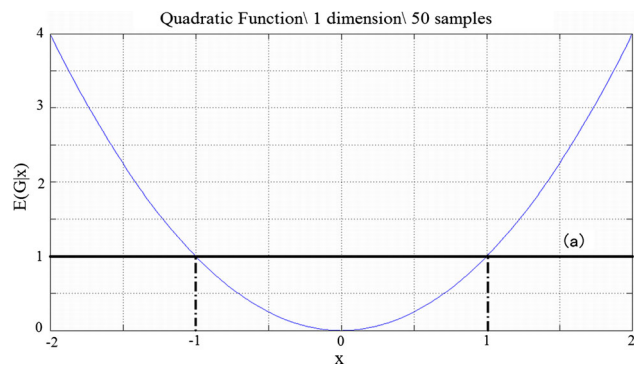


Fig. 1 $E(G|x)$ of function $f(x) = x^2$ with 50 samples

agents. In other words, $E(G|x_i = j)$ tells us what the expected objective value we will get when we choose strategy j for agent i under other variables' current probability distributions. And the picture of the conditional expectation $E(G|x_i = j)$ reflects how the objective value varies according to the different strategies of agent i . It looks just like a 'slice cut' of the objective function. So the optimization of the Maxent Lagrangian can also be interpreted as increasing the probability of strategy with low conditional expectation while decreasing the probability of those with high conditional expectation. Thus, the whole second term in gradient (8) can be treated as a *controller*, through which the negative gradient value can be assigned to the strategy with smaller conditional expectation and positive gradient values are assigned to the larger ones at the same time. And since the optimal direction in TR is roughly along the opposite direction of the gradient, the strategies with negative gradient value can get a higher probability in the next round and strategies with positive gradient value will lower its probability value. So, through controlling T, we can control the positivity and negativity of the gradient and thus control the increase and decrease of the probability value. In practice, the temperature T can be adjusted according to the corresponding probability.

For example, Fig. 1 shows the conditional expectation of quadratic function $f(x) = x^2$, $x \in (-2, 2)$ with 50 samples per iteration. Here $[-2, 2]$ is equally divided with 200 points.

Line (a) in Fig. 1 is exactly the second term of the gradient, namely we choose $T_{ij}[1 + \ln(q_{ij})] = -1$. Note that $g_{ij} = E(G|x_i = j) + T_{ij}[1 + \ln(q_{ij})]$, so the gradient value g_{ij} is positive in the intervals $-2 \leq x_{ij} \leq -1$ and $1 \leq x_{ij} \leq 2$ and negative in the interval $(-1 \leq x_{ij} \leq 1)$. Since $x_{ij} \in (-2, -1) \cup (1, 2)$ can generate a larger objective function value than those from the interval $(-1, 1)$, the probability corresponding to the former regions should decrease while the probability to the latter region should increase in the next iteration. Thus the gradient can be used to generate the descent direction needed. Based on this interpretation, the annealing process can be viewed as both lowering line (a) and narrowing the

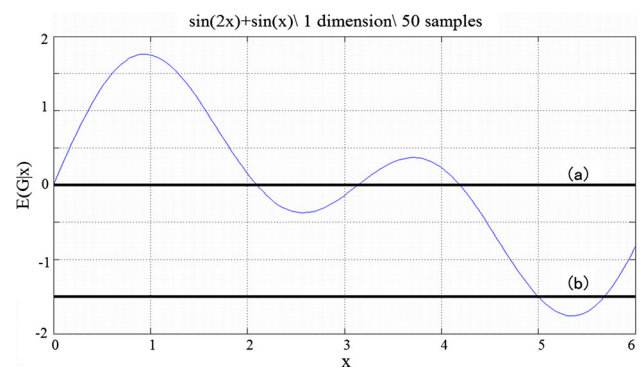


Fig. 2 Function $f(x) = \sin(2x) + \sin(x)$ with 50 samples

probability increasing region on the x -axis. The closer of the controller decreases to 0, the closer of the region narrows to the optimal point. Finally, there will be only one point left that should increase its probability. Besides, as the probability value gradually concentrates on a narrow area, so does the sampling. For example, in Fig. 1, the algorithm will finally assign a high probability (≈ 1) around $x = 0$, so samples generated by Monte-Carlo samplings will be around $x = 0$. Thus $E(G(x))$ will finally approximate a certain value through these iterations.

The new annealing schedule will also give us another insight into the optimization process. Considering the function $f(x) = \sin(2x) + \sin(x)$ for $x \in [0, 6]$, the corresponding $E(G|x)$ is shown in Fig. 2. It is clear that there is a local minimum and a global minimum in this function. At the beginning of the algorithm, the *controller* is set to be a relatively high value (the upper line (a) in Fig. 2). Obviously, the probability of any $x_i = j$ that has $E(G|x_i = j)$ above the line (a) will decrease in the next iteration and the probability values of points near the local and global optimal solution will be increased. In other words, when using line (a), MC sampling will mainly focus on the areas round the local minimum and the global minimum. However as we gradually lower the controller from line (a) to line (b), since the conditional expectation of the local minimal (i.e. $E(G|x_i = \text{local minimum})$) is now above the line (b) and its gradient value changes from negative to positive, the corresponding probability will begin to decrease. So when using line (b), the probability-increasing region will only be the areas around the global optimal point. And as the process continues, it will ultimately be picked out.

Besides that, note that the optimal solution of the Maxent Lagrangian is on the boundary of the feasible region, which means the probability is either close to 1 or 0 with negative and positive gradient, respectively. So the final solution satisfies the first-order necessary conditions of the local optimal point.

The estimation of $E(G|x)$ is another important part in our algorithm. If the conditional expectation is known at the first

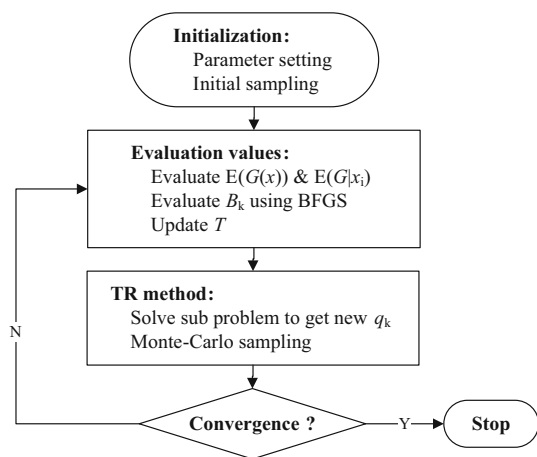


Fig. 3 Procedure of TRPC algorithm

iteration, the optimal point can be directly found. However, $E(G|x)$ may not be as precise as being expected in the beginning or even in the end of the algorithm, especially when the sample size is small. And the randomness in the estimation can hardly be eliminated. So, even though TR method is used, the new PC algorithm still has characteristics of random search methods. In general, the estimation can be viewed as a process that gradually makes $E(G|x)$ clear under the guide of probability. At first, the algorithm equally pays its attention to every point of the space and gets a rough description of the objective function. However, in the following rounds, as the probability concentrating, the algorithm can focus only on a narrower area and describe $E(G|x)$ more accurately than previous rounds did. As this process continues, TRPC gradually narrows the areas which may include the global optimal and finally pick out an interval that indeed contains the optimal point.

In practice, the gradually shrinking controller is a good choice, but here a much easier strategy is used. In each iteration, the minimal $E(G|x)$ is figured out and the largest value in its neighborhood is chosen as the controller. As the algorithm goes on, if the distribution keeps unchanged for several iterations, we sharply narrow our controller to be $\min E(G|x) + \varepsilon$. This strategy can be called the *two-phase schedule*. The selections of the width of neighborhood and value of ε all depend on the objective functions. To deal with this, several TRPC iterations (say 1000) could be ran with very large width and ε to observe the trend of objective function through $E(G|x)$. For functions changing sharply, numerical experiments show that the width and ε with small values can find the global minimum more efficient than large ones.

So far the new TRPC algorithm has been discussed in detail. The whole procedure is described in Fig. 3. It is worth noting that q_i may not be updated in every iteration. If $q_{i+1} = q_i$ in the k th iteration, samples generated in two iterations can be incorporated.

4 Experimental results

To compare arithmetic performance of TRPC with that of the original PC algorithm, the same benchmark functions used in (Huang et al. 2005) are also adopted here. The four benchmark functions are Rosenbrock Function with 10 dimensions, Ackley Function with 10 dimensions, Schaffer’s 7 Function with 2 dimensions and Michalewicz’s Epistatic Function with 10 dimensions. Besides, another 6 functions are also used. In total, the new algorithm is tested by ten benchmark functions, within which 6 of them are of 10 dimensions and 4 are of two dimensions.

As mentioned in Huang et al. (2005), the performance of the original PC algorithm has no difference under the sample size 25 and 50. But in TRPC, it does make sense to increase sample size, so we use 40–50 samples per iteration in our algorithm. Besides, since the original PC algorithm usually needs thousands of iterations while our TRPC algorithm acquires satisfying results within 2,000 iterations, it is inconvenient to show their convergence processes in one figure. So here we show the performance of the original PC and TRPC separately.

In this section, we mainly focus on the first four functions. The results generated by the TRPC algorithm are explicitly explained and comparisons are also made with that of the original PC algorithm. For the other six functions, since the results are very similar, their results are only listed in the table.

The numerical experiments were implemented using Matlab 2010b in Vista operating system and run on a personal computer with Intel(R) Core(TM) 2 2.00 GHz CPU and 2 GB of RAM.

4.1 Comparison with the original PC

(1) The Rosenbrock function

The Rosenbrock function is one particular case discussed in Huang et al. (2005). Its definition is

$$f(x) = \sum_{i=1}^{N-1} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2],$$

where $x = (x_1, x_2, \dots, x_N)$, $-5 \leq x_i \leq 5$ and $N = 10$. The optimal point is $(1, 1, \dots, 1)$ and the corresponding function value is 0.

Figure 4a shows the results obtained by the original PC. The x -axis shows iteration times and the y -axis is the value of $E(G)$. Figure 4b just displayed a detailed view from the range of the $E(G)$ on the interval $[0,1,000]$. In the figure, one can see that the function value is still near 100 after 1,000 iterations by using the original PC. However, for the same function, a satisfactory result is acquired within 1,200 times of iteration by our new TRPC algorithm. The results

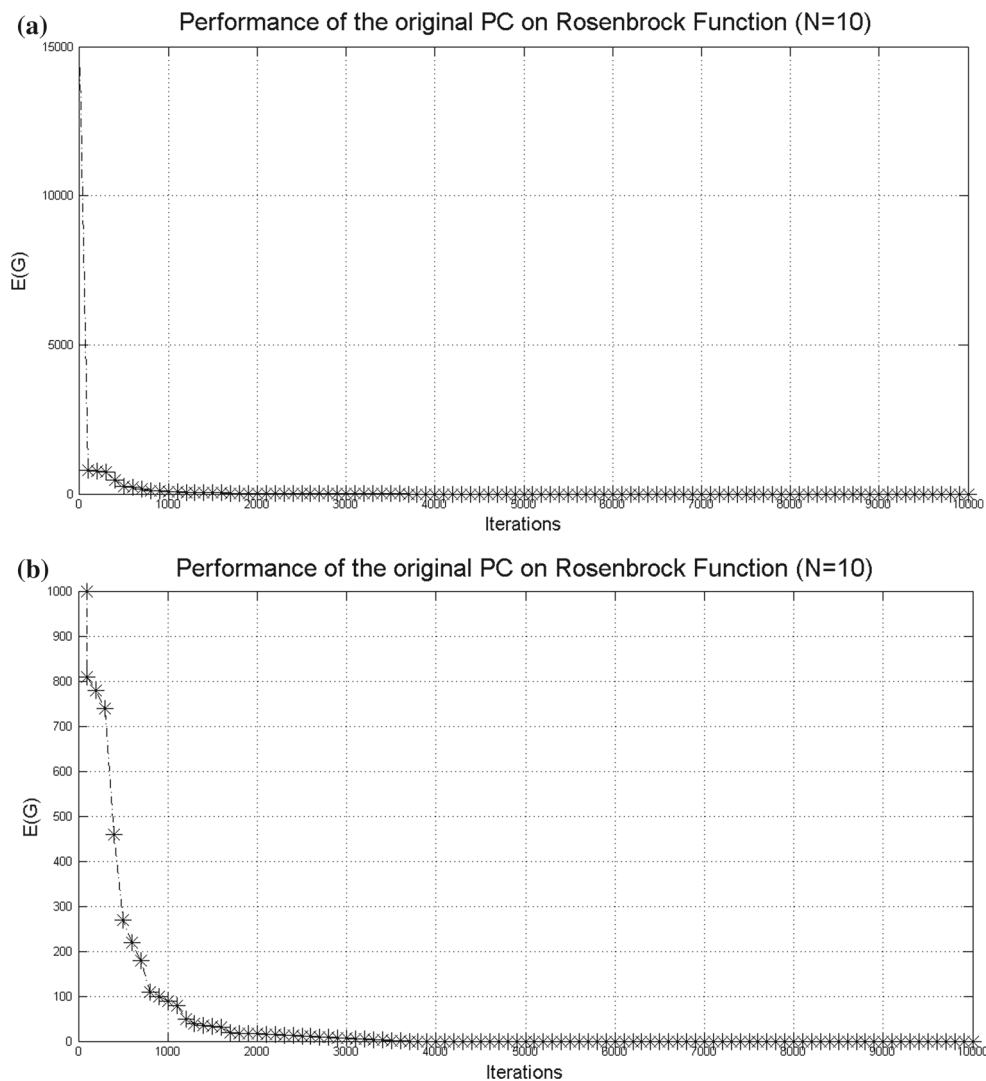


Fig. 4 Performance of the original PC on Rosenbrock function

are shown in Figs. 5, 6. Although TRPC method needs more samples in this case, it sharply decreases the value of objective function from 4,000 to 10 within 10 iterations and the algorithm accurately finds the global optimum.

It should be noted that $E(G)$ is used here for the convenience of comparing with Huang's numerical results, but it is not the final objective value. The final optimal value comes from $G(x^*)$ with setting x^* the point with the highest probability value in each agent. In addition, the accuracy of the result is closely related to the range of intervals and discretization.

Other outputs of the new algorithm are shown in Fig. 6. Figure 6a presents the curve of $E(G|x)$, and Fig. 6b shows the probability values of x_{ij} . Here, $[-2, 2]$ is equally divided into 200 intervals to discretize the continuous function. So x -axis ranges from 0 to 200 in both figures. In Fig. 6a, the curve x_1 depicts $E(G|x_1)$, the $\times 10$ depicts $E(G|x_{10})$, and the other variables share the same unlabeled curve. Since the closed form

of the objective function is not known, Figure 6a provides some useful information about it. First, $(-1, 1, 1, \dots, 1)$ is a local optimal point since the curve x_1 is close to the x -axis at $x_1 = -1$ when the other variables have values 1. Second, a slight change of x_{10} will not cause big changes in the objective value because the curve x_{10} is relatively flat around its optimal solution. Third, the variables x_2, \dots, x_9 are symmetric because their curves overlap. The final probability concentrates on point $(1, 1, \dots, 1)$ and the exact values are shown in Table 1. It is clear that $f^* = 0$.

(2) The Ackley function

The Ackley function is another case discussed in Huang et al. (2005). The definition of the function is

$$f(x) = -ae^{-b\left(\frac{\sum_i x_i^2}{N}\right)^{1/2}} - e^{\frac{\sum_i \cos(cx_i)}{N}} + a + e,$$

where $a = 20$, $b = 0.2$, $c = 2\pi$, $N = 10$, $-3 \leq x_i \leq 3$. The optimal point is $(0, 0, \dots, 0)$ and $f(0, 0, \dots, 0) = 0$.

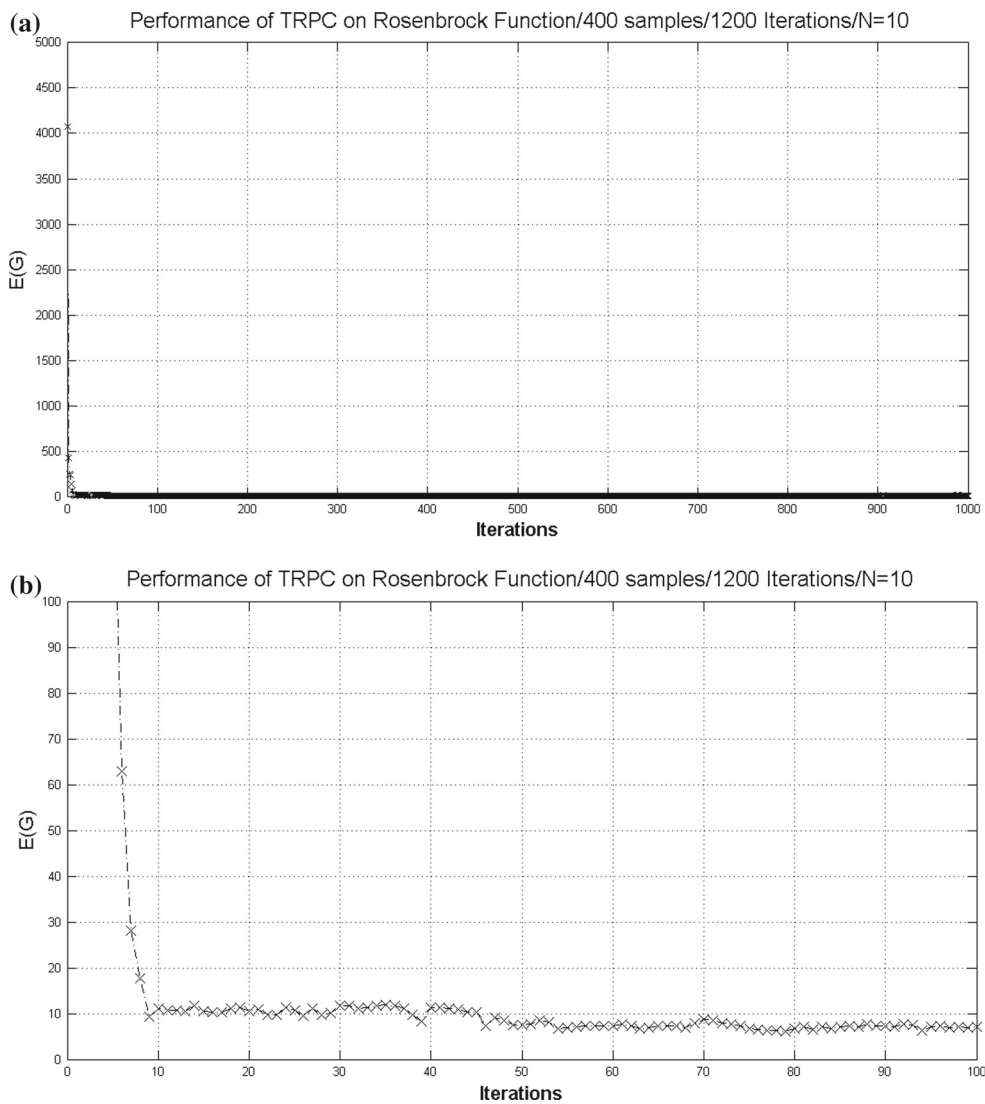


Fig. 5 $E(G)$ by TRPC in Rosenbrock function

Figure 7a, b show the results of TRPC and Fig. 8 displays the results of the original PC. This experiment may be more persuasive than the Rosenbrock function, because we only use 40 samples per iteration for our TRPC in this case and obtain the satisfying result within 1,500 iterations. However, in contrast, the original PC algorithm needs more than 8,000 iterations. The result once again shows that TRPC method outperforms the original PC algorithm.

Similar to the Rosenbrock function, Fig. 9a provides us with information about the objective function through $E(G|x)$ and Fig. 9b shows the optimal point should be $x = (0, 0, \dots, 0)$. It is clear that this function only has one global optimum and all x_i are symmetric. The final probability is shown below (Table 2).

In this case, it is worth noting that the f^* is 0.0427 since x_3 deviates from 0 by one unit (0.03 in this case). The reason for this deviation is the *two phase schedule*. The ε we use in min

$E(G|x) + \varepsilon$ is too small, so even TRPC finds the right candidate area in the first phase, the probability concentrates too fast on the false optimum without exploring other potential points around it. This problem could be solved using more phases in the annealing schedule.

(3) The Schaffer’s function F_7

The third benchmark function used in the literature is Schaffer’s function F_7 . The function is defined as

$$f(x) = (x_1^2 + x_2^2)^{0.25} [\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1],$$

where $-2 \leq x_i \leq 2$ for $i = 1, 2$. The optimal point is (0,0) and the function value at this point is 0.

Figure 10a was obtained by TRPC using one point every ten iterations to compare with the result of the original PC algorithm, which is shown in Fig. 10b. At first glance, these two figures above seemingly make no difference between TRPC and PC. But if we look at the probability, it is obvious

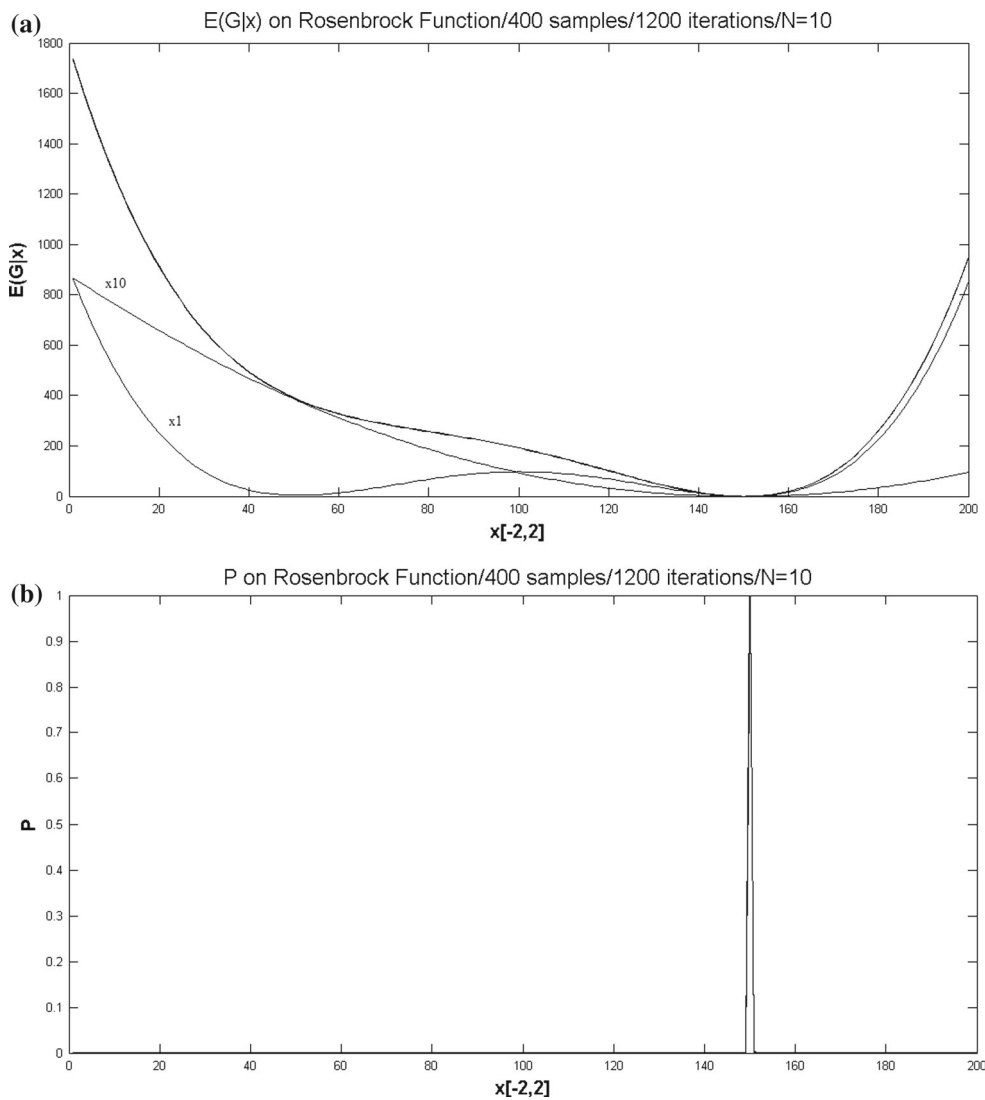


Fig. 6 $E(G|x)$ and P in Rosenbrock function

Table 1 Probability values in Rosenbrock function

P Prob Agt	146	147	148	149	150	151	152	153	154
q_1	2.56E-08	2.56E-08	2.56E-08	2.56E-08	0.999995	2.56E-08	2.56E-08	2.56E-08	2.56E-08
q_2	5.29E-08	5.29E-08	5.29E-08	5.29E-08	0.999989	5.29E-08	5.29E-08	5.29E-08	5.29E-08
q_3	1.24E-08	1.24E-08	1.24E-08	1.24E-08	0.999998	1.24E-08	1.24E-08	1.24E-08	1.24E-08
q_4	1.37E-08	1.37E-08	1.37E-08	1.37E-08	0.999997	1.37E-08	1.37E-08	1.37E-08	1.37E-08
q_5	4.87E-08	4.87E-08	4.87E-08	4.87E-08	0.999990	4.87E-08	4.87E-08	4.87E-08	4.87E-08
q_6	6.71E-08	6.71E-08	6.71E-08	6.71E-08	0.999987	6.71E-08	6.71E-08	6.71E-08	6.71E-08
q_7	1.01E-07	1.01E-07	1.01E-07	1.01E-07	0.999980	1.01E-07	1.01E-07	1.01E-07	1.01E-07
q_8	5.50E-08	5.50E-08	5.50E-08	5.50E-08	0.997517	0.002473	5.50E-08	5.50E-08	5.50E-08
q_9	7.19E-08	7.19E-08	7.19E-08	7.19E-08	0.999986	7.19E-08	7.19E-08	7.19E-08	7.19E-08
q_{10}	1.60E-08	1.60E-08	1.60E-08	0.000801	0.999196	1.60E-08	1.60E-08	1.60E-08	1.60E-08

that the optimal point has already been found within 2,000 iterations by TRPC. The gap between the $E(G)$ and the optimal value 0 is due to the randomness and discretization.

From Fig. 11a, it is clear that there are many local minimums in interval $[-2, 2]$, and TRPC is not trapped in any of them and successfully find the global optimum. The optimal

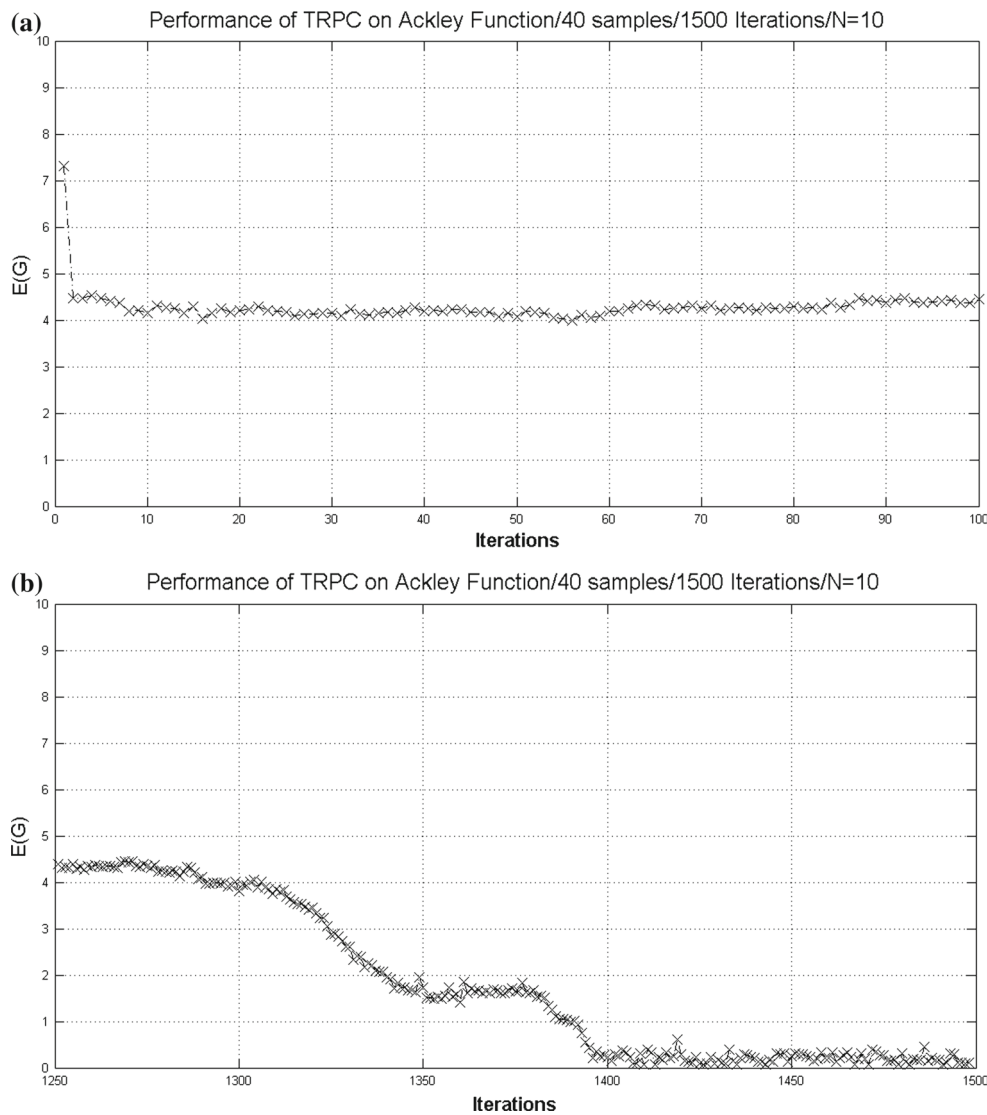


Fig. 7 $E(G)$ by TRPC

function value f^* is 0 in this case. Table 3 below gives the final probability.

(4) The Michalewicz’s epistatic Function

The final test function used in the literature is the Michalewicz’s epistatic function, which has the following form:

$$f(x) = - \sum_{i=1}^N \sin(y_i) \sin^{2m} \left(\frac{iy_i^2}{\pi} \right),$$

where

$$y_i = x_i \cos \frac{\pi}{6} - x_{i+1} \sin \frac{\pi}{6} \text{ if } i \bmod 2 = 1 \text{ and } i \neq N;$$

$$y_i = x_{i-1} \sin \frac{\pi}{6} + x_i \cos \frac{\pi}{6} \text{ if } i \bmod 2 = 0 \text{ and } i \neq N;$$

$$y_N = x_N;$$

$$0 \leq x_i \leq \pi \quad \text{for } i = 1, 2, \dots, N.$$

$$m = 10;$$

The optimal function value is -9.66 when $N = 10$.

Figure 12a shows the $E(G)$ generated by TRPC while Fig. 12b shows the results of original PC. The advantage of the TRPC method is even more obvious in this experiment. Though fluctuating sharply before 1,800 iterations, $E(G)$ finally keeps stable at -9.5526 in TRPC. However, in the original PC algorithm, the objective value is larger than -9 even after 10,000 iterations. The gap between -9.662 and -9.552 is due to the discretization.

Here we pick out $E(G|x_i)$ of x_1, x_4, x_8 and x_{10} . It is interesting to note that x_i has i local minimums in this function. It should be pointed out that for the convenience of coding, Fig. 13 shifts the minimum $E(G|x_i)$ to the point zero, but this

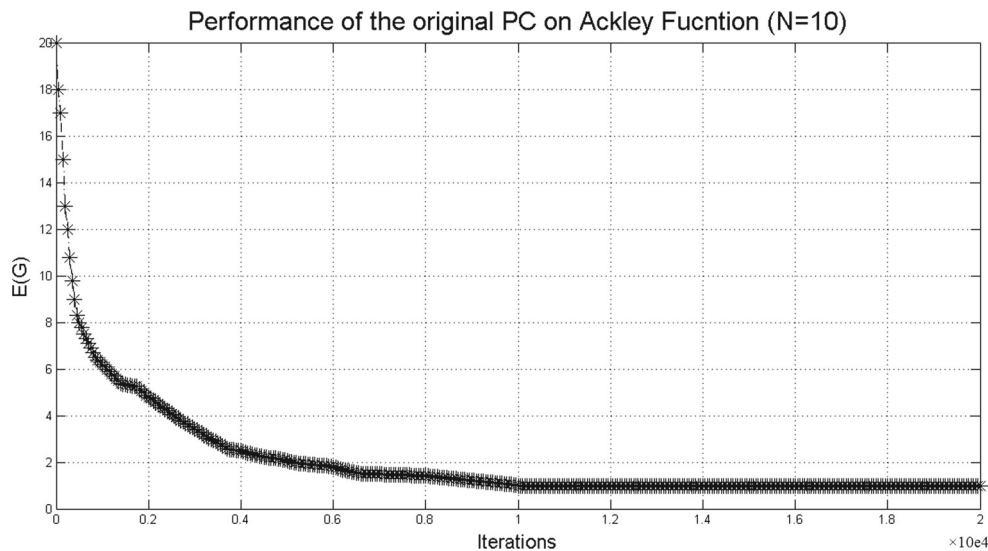


Fig. 8 Performance of the original PC on Ackley function

does not influence the other parts of the algorithm. And the figure below shows probability values. The actual probability value table is omitted since the optimal points are separated from each other and can hardly be put into one table (Fig. 14).

4.2 Other numerical experiments

Besides the functions used in Huang et al. (2005), here another 6 test problems are chosen and the results are listed in Table 4. The fifth column of Table 4 are solutions from TRPC and the last column is the analytical optimal solutions.

4.3 Comparison with another PC modification

Another PC modification here refers to the method mentioned in Kulkarni et al. (2009). For convenience, we call it modified PC (MPC) below. Its main process is shown in the following figure:

The main character of the MPC is that it indeed shrinks the sampling space per iteration. According to the current best solution (or ‘Favorable Strategy’ in the literature), it artificially sets upper and lower bounds (Favorable Strategy $\pm(0.5$ or $0.1)$ *Favorable strategy) for the sampling space. The advantage of this setting is that it improves the accuracy of the final solution. However, since it still uses the Nearest Newton Method to update the probability, the problems mentioned in Sect. 3.1 still exist and it clearly shows in Fig 15 that a large number of probability updating ($k \cdot n$ times) is needed to ‘sharpen’ the probability of the current best solution per iteration. So, the MPC does not change the updating equation of the original PC; instead, it redesigns the whole process to increase the number of probability updating in each iteration

and limits the sampling space to compensate for the potential ineffectiveness of the iterative step.

In contrast, TRPC totally changes the iterative step and thus can make the probability very close to 0 and 1, which means it can shrink the sampling space in terms of probability but does not actually change its interval. In other words, MPC directly limits the sampling space around the current best solution but TRPC uses temperature T to manipulate the gradient and then controls the increase and decrease of the probability and thus finally locks on the best solution. So, TRPC has more efficient probability updating than that of MPC. But the discretization will have more impacts on TRPC’s final solution than on the MPC’s, since it does not actually reduce the search interval. This could be the major disadvantage of the TRPC method.

Although the Rosenbrock function in Kulkarni and Tai 2009 is 5 dimensions and the range for each variable is different, this will not influence the comparison between these two methods. The reasons for this are as following: First, TRPC could find the optimal solution within 1,200 times function evaluations even in a harder problem (10 dimensions), while MPC needs 2.047×10^5 – 3.591×10^5 iterations. This clearly shows TRPC needs less function evaluations. Second, the actual search interval in both methods is not related to the dimension. In TRPC, it is $[-2, 2]$ for each variable from the beginning to the end. The discretization is also on the same interval. But in MPC, the actual search interval is determined by the lower and upper bounds (favorable strategy ± 0.1 *favorable strategy) and the final discretized intervals are $[-1, -0.9]$ for x_1 , $[1, 1.1]$ for x_5 and $[0.9, 1.1]$ for the other variables. Since the final search interval in MPC is much smaller than that of TRPC, the variation of the optimal solution in MPC ($\pm 10^{-3}$) is also less than TRPC’s ($\pm 10^{-2}$).

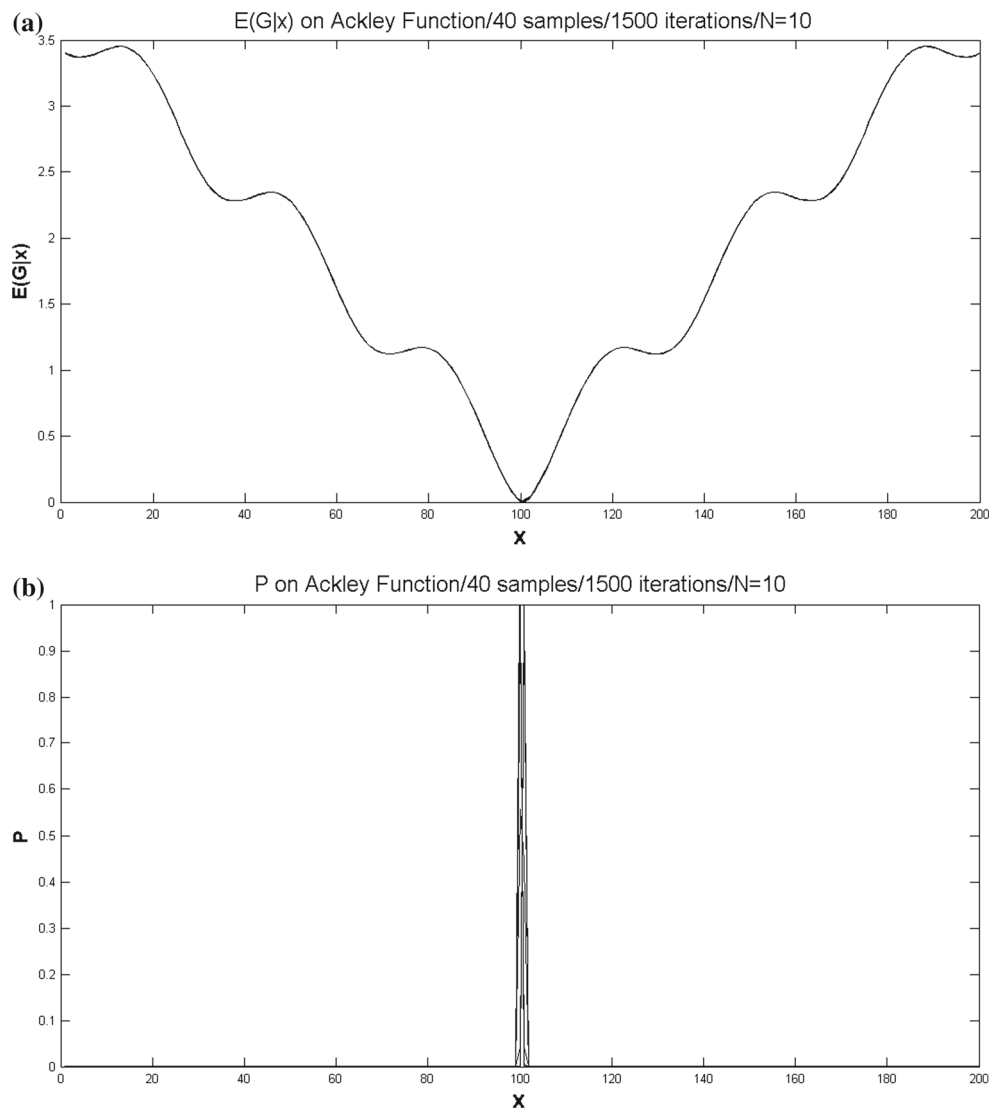


Fig. 9 $E(G|x)$ and P in Ackley function

Table 2 Probability values in the Ackley function

P Prob Agt	96	97	98	99	100	101	102	103	104
q_1	6.30E-05	6.30E-05	6.30E-05	6.30E-05	0.983253	4.28E-03	6.30E-05	6.30E-05	6.30E-05
q_2	8.49E-09	8.49E-09	8.49E-09	8.49E-09	0.556417	4.44E-01	8.49E-09	8.49E-09	8.49E-09
q_3	2.00E-08	2.00E-08	2.00E-08	2.00E-08	2.04E-08	1.00E+00	2.00E-08	2.00E-08	2.00E-08
q_4	9.80E-05	9.80E-05	9.80E-05	9.80E-05	0.979586	1.02E-03	9.80E-05	9.80E-05	9.80E-05
q_5	4.31E-05	4.31E-05	4.31E-05	4.31E-05	0.99143	4.31E-05	4.31E-05	4.31E-05	4.31E-05
q_6	1.11E-05	1.11E-05	1.11E-05	1.11E-05	0.997797	1.11E-05	1.11E-05	1.11E-05	1.11E-05
q_7	8.67E-05	8.67E-05	8.67E-05	8.67E-05	0.9395	4.33E-02	8.67E-05	8.67E-05	8.67E-05
q_8	8.46E-05	8.46E-05	8.46E-05	8.46E-05	0.941093	4.22E-02	8.46E-05	8.46E-05	8.46E-05
q_9	3.38E-08	3.38E-08	3.38E-08	3.38E-08	0.999993	3.38E-08	3.38E-08	3.38E-08	3.38E-08
q_{10}	4.57E-08	4.57E-08	4.57E-08	4.57E-08	0.999991	4.57E-08	4.57E-08	4.57E-08	4.57E-08

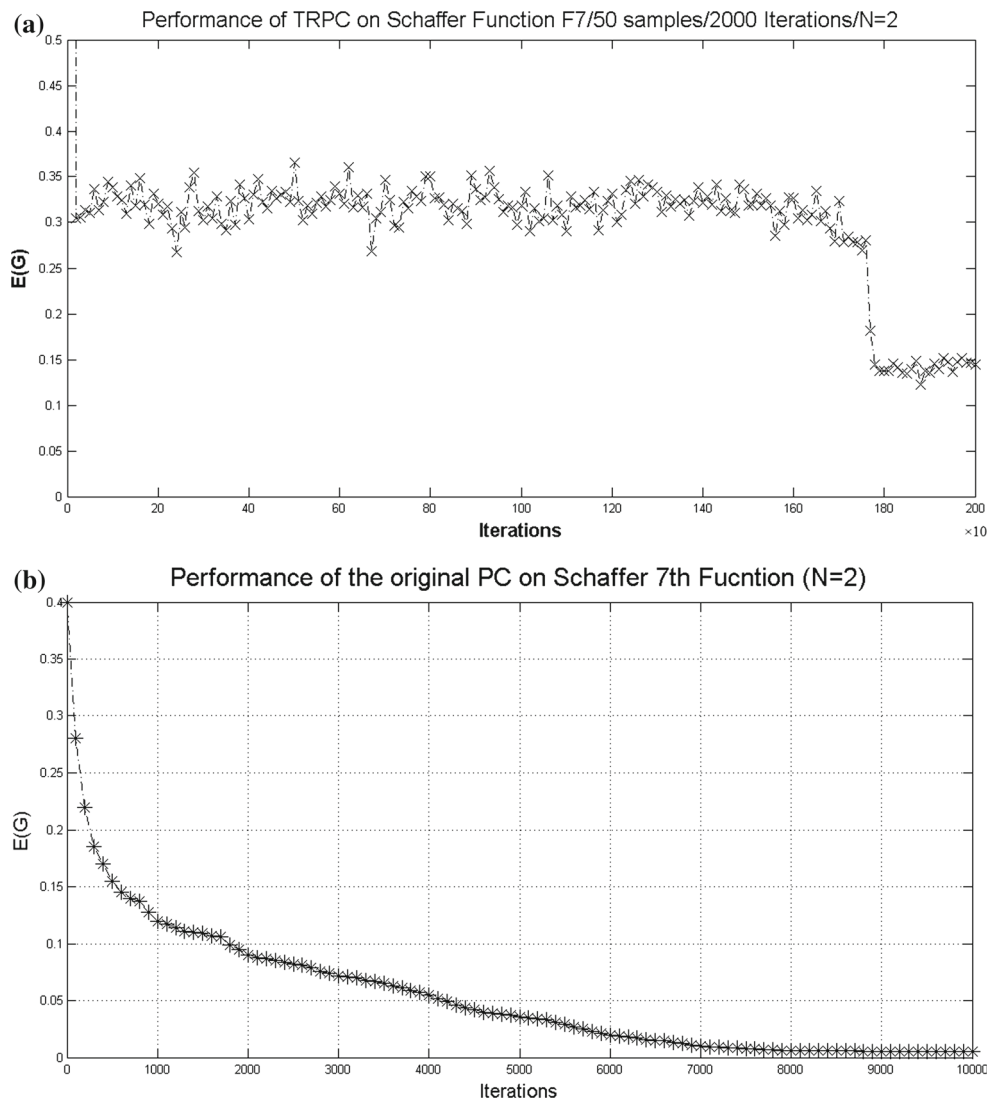


Fig. 10 $E(G)$ by TRPC and PC in Schaffer's function F_7

Third, since both TRPC and MPC do not change the structure of the Maxent Lagrangian, the range for each variable in these two methods could be different.

5 Conclusions

In this paper, a modified PC algorithm is presented by combining the TR and a new annealing schedule. The original probability collectives (PC) algorithm is a kind of heuristic algorithm, which focuses on adapting the distribution over the strategy set of each variable. In order to improve its performance, potential requirements of solving the Maxent Lagrangian are first identified. It is found that the gradient-based optimization should not only be an interior point method but also can tolerate randomness and at the same time avoid computation of the step size. So an interior point TR

method is used to meet all these demands. Another improvement is the adaptation of a new annealing schedule which replaces the geometrically decreasing one. For checking the performance, ten benchmark functions are used and numerical results show that the new TRPC method outperforms the original PC algorithm significantly.

Analysis in this article shows that the TRPC method still has improvement potentials and future researches could focus on the following points. First, PC algorithm is inherently designed as a distributed algorithm, so parallel computing could be used to redesign the whole algorithm. Second, the annealing schedule used here is a basic form. Other much more efficient schedules could be developed on this basis and the algorithm could be even more powerful. Third, since PC algorithm could solve both discrete and continuous problems, there are huge potentials to solve real problems with it.

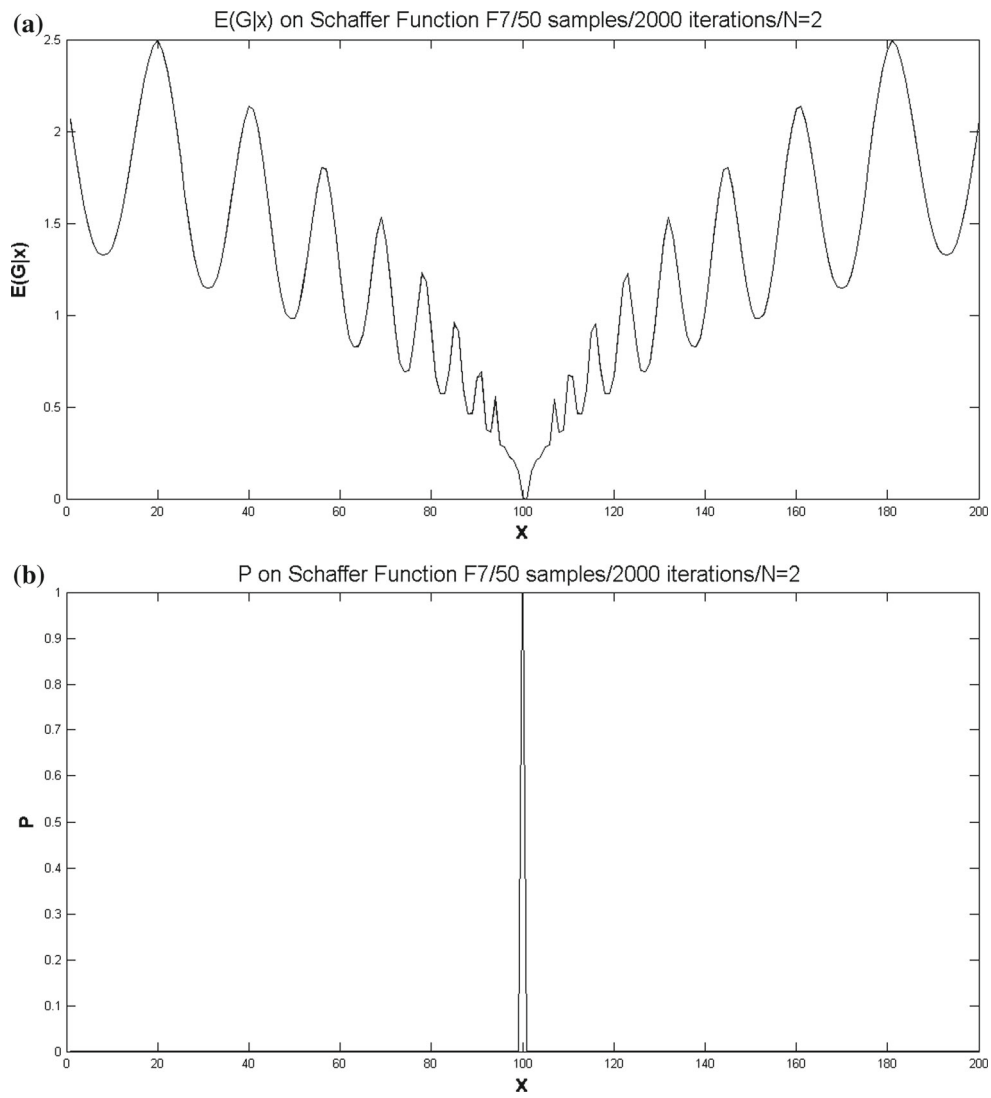


Fig. 11 $E(G|x)$ and P in Schaffer’s function F_7

Table 3 Probability values in the Schaffer’s function F_7

P ProbAgt	96	97	98	99	100	101	102	103	104
q_1	2.57E-09	2.57E-09	2.57E-09	2.57E-09	0.999999	2.57E-09	2.57E-09	2.57E-09	2.57E-09
q_2	1.05E-10	1.05E-10	1.05E-10	1.05E-10	1	1.05E-10	1.05E-10	1.05E-10	1.05E-10

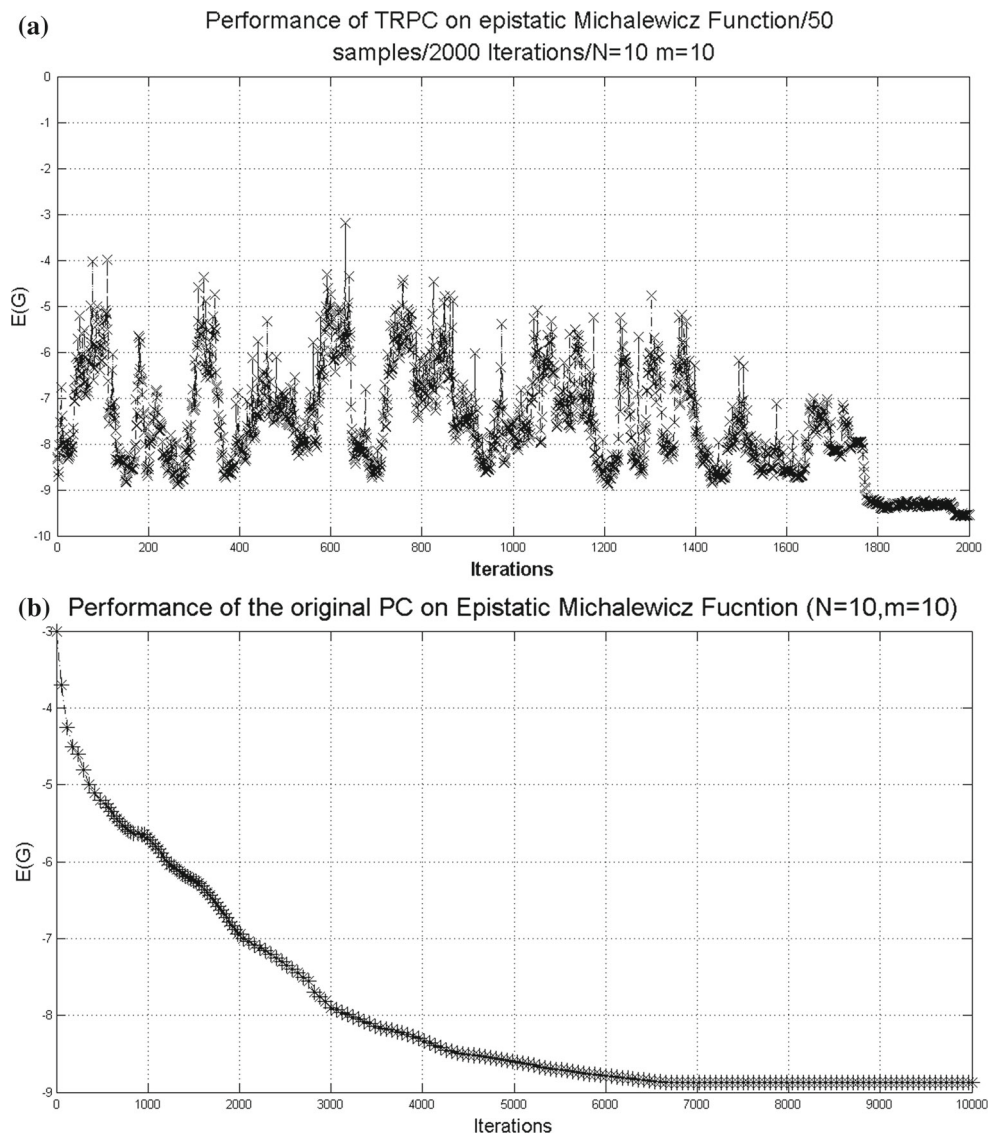


Fig. 12 $E(G)$ by TRPC and PC in Michalewicz epistatic function

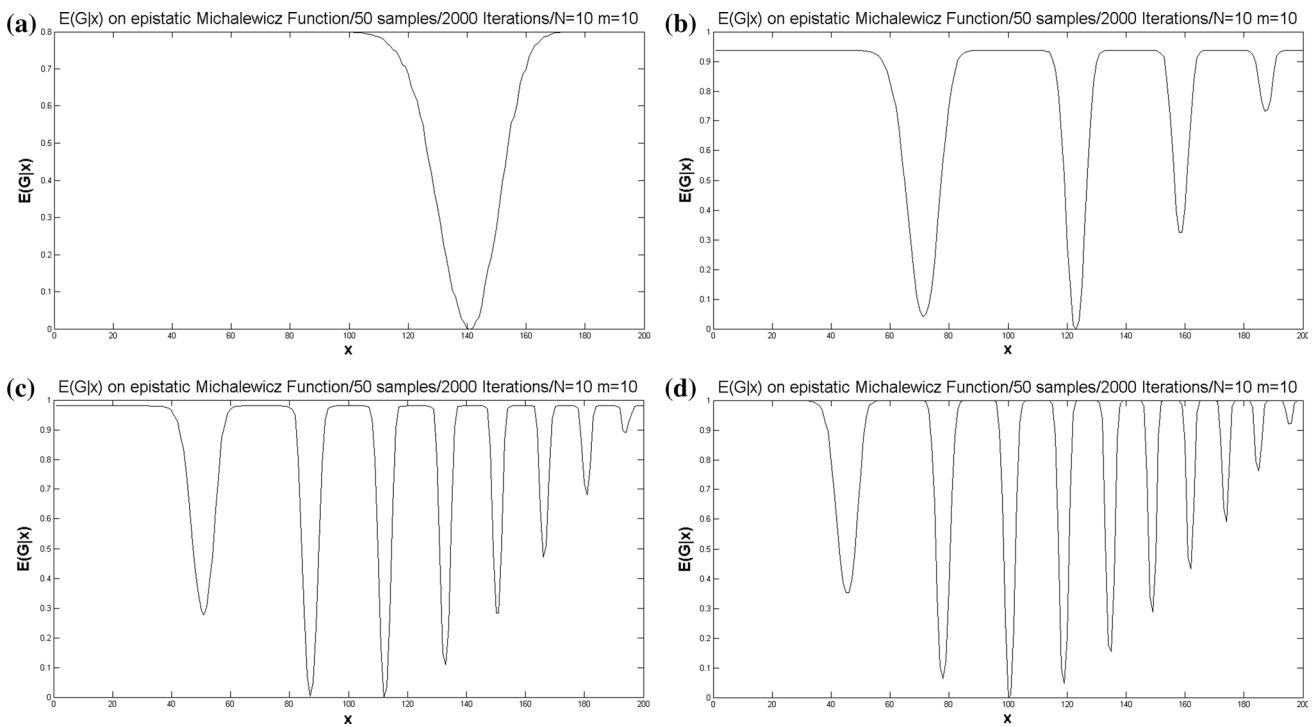


Fig. 13 $E(G|x)$ in Michalewicz epistatic function

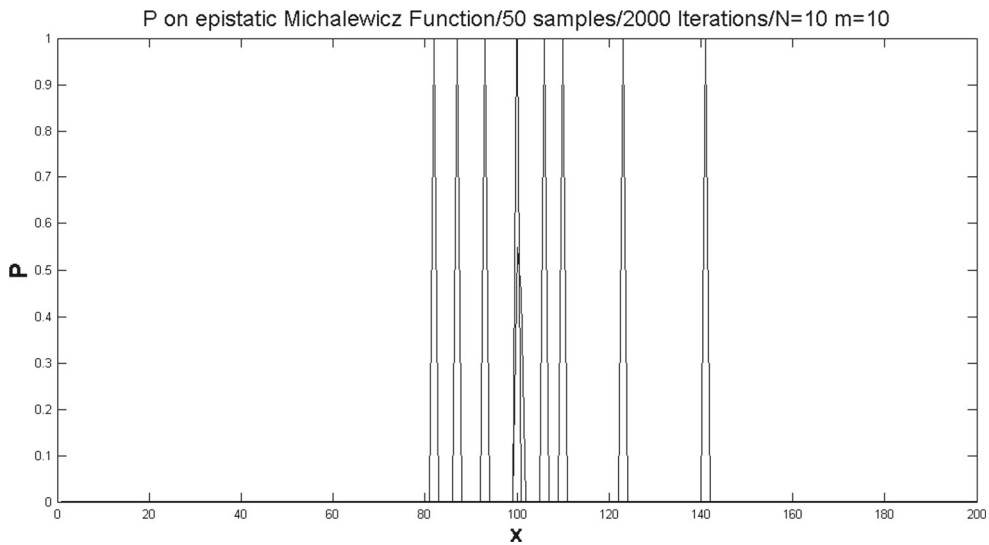
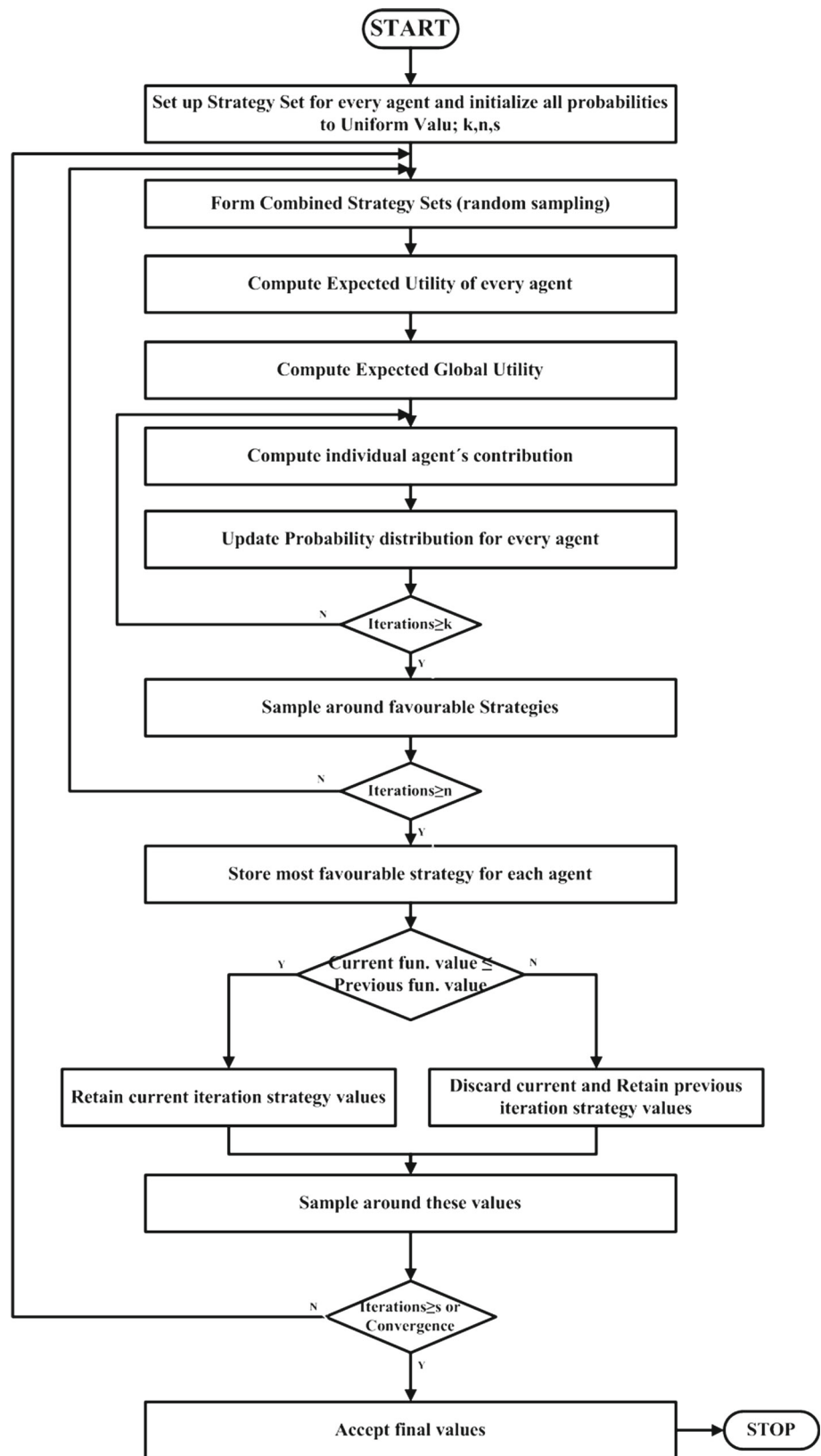


Fig. 14 P in Michalewicz epistatic function

Table 4 Other test functions and results

Function	Dimensions	Sample size	Iterations	TRPC_ X^*	Opt_points
Rastrigin function	10	40	1,000	$X_{(8)} = 0, X_8 = 0.05$	$[0, \dots, 0]$
Shuber function	2	40	2,000	$[-1.432, -0.82]$	$[-1.425, -0.8]$
Griewank function	10	40	1,000	$X_1 = 0.044; X_{10} = 0.088; X_i = 0$	$[0, \dots, 0]$
Sum squares function	10	40	1,000	$X_5 = 0.05; X_{(5)} = 0$	$[0, \dots, 0]$
Easom function	2	40	1,400	$[0, 0]$	$[0, 0]$
Schaffer2 function	2	40	1,000	$[0, 0.025]$	$[0, 0]$

Fig. 15 MPC process



Appendix A

This appendix provides more detailed explanations about the interior TR method we use in Sect. 3. However, the proof of the convergence is still omitted here. Readers who are interested in this could check (Coleman et al. 1996) for more information.

The method we consider is mainly for the problem with a smooth nonlinear objective function subject to bounds on variables:

$$\min_{x \in \mathbb{R}^n} f(x), l \leq x \leq u, \tag{13}$$

The feasible region is $F \stackrel{\text{def}}{=} \{x : l \leq x \leq u\}$ and the strict interior is $\text{inf}(F) \stackrel{\text{def}}{=} \{x : l < x < u\}$. It can be easily checked that model (10) follows this form with $l = 0$ and $u = 1$.

The main idea of this interior TR method is to transform the problem above into a corresponding TR sub-problem which has the same form with the TR sub-problem of an unconstrained problem. To implement this, first we define a vector $v(x)$ and an affine scaling matrix $D(x)$ as follows:

Definition 3 Let $v(x_i) = (v_{i1}, v_{i2}, \dots, v_{in})$ be a vector for x_i ; then

$$v_i = \begin{cases} x_i - u_i, & \text{if } \nabla f(x)_i < 0 \\ x_i - l_i, & \text{if } \nabla f(x)_i \geq 0 \end{cases}$$

Note that v_i measures each component's distance from the current point $x = (x_1, x_2, \dots, x_n)$ to the bound l and u .

Definition 4 For all $v(x_i)$, let

$$D(x) = \text{diag}(|v(x_i)|^{-1/2}),$$

where $\text{diag}(\cdot)$ denotes a diagonal matrix.

Assume x^* is a local minimizer for (13), so the first-order necessary conditions for x^* should be

$$\text{first order : } \begin{cases} \nabla f(x^*)_i = 0, & \text{if } l_i < (x^*)_i < u_i, \\ \nabla f(x^*)_i \leq 0, & \text{if } (x^*)_i = u_i, \\ \nabla f(x^*)_i \geq 0, & \text{if } (x^*)_i = l_i, \end{cases}$$

It is worth noting that these first-order conditions to (13) are equivalent to

$$D_*^{-2} \nabla f(x^*) = 0 \tag{14}$$

(14) has the form of the first-order conditions for unconstrained problems and this is exactly why we use the scaling transformation.

So, a Newton step for (14) satisfies

$$(D_k^{-2} H_k + \text{diag}(\nabla f(x_k)) J_k^v) d_k = -D_k^{-2} \nabla f(x_k),$$

where J_k^v is the Jacobian matrix of $|v(x_i)|$; we set $J_k = \text{diag}(\text{sign}(\nabla f))$. The term $\text{diag}(\nabla f(x_k)) J_k^v$ on the left side is to make the scaled Hessian matrix $D_k^{-2} H_k$ positive semi-definite.

Based on this Newton step, we could define our quadratic model for the TR method:

$$\begin{aligned} \psi_k(s) &= s^T \nabla f_k + \frac{1}{2} s^T M_k s \\ \text{s.t. } \|s\| &\leq \delta_k, \end{aligned} \tag{15}$$

where

$$C(x) = D(x) \text{diag}(\nabla f(x)) J^v(x) D(x),$$

$$M(x) = B(x) + C(x).$$

And a slight modification of the quadratic model above is exactly (11) we used in Sect. 3.2 where $\nabla f_k = \nabla L_k$. It's also obvious that the statements a~c in Sect. 3.2 hold.

Appendix B

This appendix describes the *dog-leg* method we used to solve the quadratic sub-problem in TR. Still, we omit the convergence proof of this method and only present the results. More information could be found in Gill et al. (1981).

The quadratic sub-problem we consider is

$$\begin{aligned} \min m_c(x_c + s) &= f(x_c) + \nabla f(x_c)^T s + \frac{1}{2} s^T H_c s, \\ \text{subject to } \|s\|_2 &\leq \delta_c. \end{aligned} \tag{16}$$

Note if we define $\psi(s) = m_c(x_c + s) - f(x_c)$, (16) has the same form of (15).

The *dog-leg* method does not find the optimal solution s^* to (16). Instead, it uses two directions to approximate s^* . The first direction is the steepest descent direction and the second is the Newton direction. According to the trust region radius,

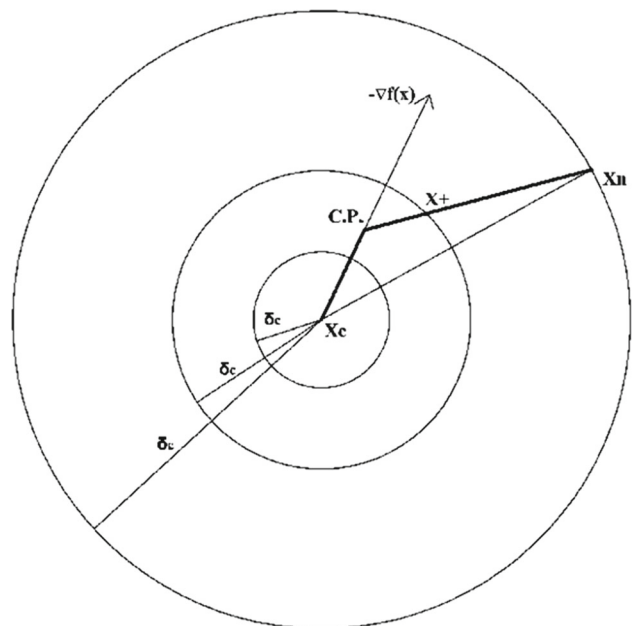


Fig. 16 TR method

different s is chosen as an approximate solution to (16). This idea could be illustrated by the figure below (Fig. 16).

Point C.P. is the Cauchy Point, the minimizer of (16) in the steepest descent direction. Obviously, we need to consider three cases:

Case 1 if $\delta_c \leq \|s^{\text{C.P.}}\|_2$

In this case, we choose $s = \alpha s^{\text{C.P.}}$, $0 < \alpha < 1$ as the final solution.

Case 2 if $\|s^{\text{C.P.}}\|_2 \leq \delta_c \leq \|s^N\|_2$

In this case, we choose $s = s^{\text{C.P.}} + \alpha(s^N - s^{\text{C.P.}})$, $0 < \alpha < 1$ as the final solution.

Case 3 if $\delta_c \geq \|s^N\|_2$

In this case, we choose $s = s^N$ as the final solution

It has been proved that along the curve $x_c \rightarrow \text{C.P.} \rightarrow x_N$, m_c decreases monotonically and there is always $\|s^{\text{C.P.}}\|_2 \leq \|s^N\|_2$. So every solution generated by the *dog-leg* method to the quadratic sub-problem is a sufficient descent and the method ultimately converges. In Sect. 3.2, we use the notation s^B and s^U instead of s^N and $s^{\text{C.P.}}$.

References

- Antoine N et al (2004) Fleet assignment using collective intelligence. In: Proceedings of 42nd aerospace sciences meeting
- Autry Brian M (2008) University course timetabling with probability collectives. NAVAL POSTGRADUATE SCHOOL MONTEREY, CA
- Bieniawski SR (2005) Distributed optimization and flight control using collectives. PhD thesis, Stanford University
- Busoniu L, Robert B, De Bart S (2008) A comprehensive survey of multiagent reinforcement learning. In: IEEE transactions on systems, man, and cybernetics, part C: applications and reviews, vol 38.2, pp 156–172 (2008)
- Coleman TF, Li Y (1994) On the convergence of interior-reflective Newton methods for nonlinear minimization subject to bounds. Math Programm 67(1–3):189–224
- Coleman Thomas F, Li Yuying (1996) An interior trust region approach for nonlinear minimization subject to bounds. SIAM J Optim 6(2):418–445
- Gill Philip E, Murray W, Margaret H (1981) Wright, practical optimization. Academic Press, New York
- Huang C-F et al (2005) A comparative study of probability collectives based multi-agent systems and genetic algorithms. In: Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM, New York
- Kulkarni AJ, Tai K (2008) Probability collectives for decentralized, distributed optimization: a collective intelligence approach. In: IEEE international conference on IEEE systems, man and cybernetics
- Kulkarni AJ, Tai K (2009) Probability collectives: a decentralized, distributed optimization for multi-agent systems. In: Applications of soft computing. Springer, Berlin, pp 441–450
- Kulkarni AJ, Tai K (2010a) Probability collectives: a distributed optimization approach for constrained problems. In: Evolutionary computation (CEC), IEEE congress, pp 1–8
- Kulkarni AJ, Tai K (2010b) Probability collectives: a multi-agent approach for solving combinatorial optimization problems. Appl Soft Comput 10(3):759–771
- Kulkarni AJ, Tai K (2011) A probability collectives approach with a feasibility-based rule for constrained optimization. Appl Comput Intell Soft Comput 12
- Lee CF, Wolpert DH (2004) Product distribution theory for control of multi-agent systems. In: Proceedings of the third international joint conference on autonomous agents and multiagent systems, vol 2. IEEE Computer Society, Washington
- Wolpert DH, Lee CF (2004) Adaptive metropolis-Hastings sampling using product distributions. In: Proceedings of ICCS, vol 4
- Wolpert DH, Tumer K (2002) Collective intelligence, data routing and Braess paradox. J Artif Intell Res 16:359–387
- Wolpert DH (2006) Information theory—the Bridge connecting bounded rational game theory and statistical physics. Complex engineered systems. Springer, Berlin
- Wolpert DH, Bieniawski S (2004) Distributed control by lagrangian steepest descent. In: 43rd IEEE conference on decision and control. CDC, vol 2. IEEE, Washington, D.C
- Wolpert DH, Tumer K (1999) An introduction to collective intelligence. In: Technical report, NASA ARC-IC-99-63, NASA Ames Research Center
- Wolpert DH, Tumer K (2001) Optimal payoff functions for members of collectives. Adv Complex Syst 4(2/3):265–279
- Wolpert DH, Rajnarayan D, Bieniawski S (2011) Probab Collect Optim