Self-adaptive Lower Confidence Bound: A New General and Effective Prescreening Method for Gaussian Process Surrogate Model Assisted Evolutionary Algorithms

Bo Liu, Qingfu Zhang, Senior Member, IEEE, Francisco V. Fernández, and Georges Gielen, Fellow, IEEE

Abstract—Surrogate model assisted evolutionary algorithms are receiving much attention for the solution of optimization problems with computationally expensive function evaluations. For small scale problems, the use of a Gaussian Process surrogate model and prescreening methods has proven to be effective. However, each commonly used prescreening method is only suitable for some types of problems, and the proper prescreening method for an unknown problem cannot be stated beforehand. In this paper, the four existing prescreening methods are analyzed and a new method, called self-adaptive lower confidence bound (ALCB), is proposed. The extent of rewarding the prediction uncertainty is adjusted on line based on the density of samples in a local area and the function properties. The exploration and exploitation ability of prescreening can thus be better balanced. Experimental results on benchmark problems show that ALCB has two main advantages: (1) it is more general for different problem landscapes than any of the four existing prescreening methods; (2) it typically can achieve the best result among all available prescreening methods.

I. INTRODUCTION

Many real-world optimization applications involve non-differentiable, black-box and highly multimodal hypersurfaces. Traditional mathematical programming methods find many difficulties to solve these problems. Evolutionary computation (EC) is an active research area and evolutionary algorithms (EAs) are becoming a standard approach to address them [1]. However, the main drawback of EAs is that typically thousands of function evaluations are necessary. When the function evaluation is computationally expensive, as often happens in many simulation-based optimization problems, the time to finish the optimization may become unacceptably long. For instance, mm-wave circuit and antenna synthesis [2] require electromagnetic simulation-based optimization, and several days or months of computation time can be consumed when canonical EAs are used.

To enhance the efficiency, surrogate model assisted evolutionary algorithms (SAEAs) [3] are attracting much attention. The efficiency is significantly enhanced in SAEAs, because many expensive exact function evaluations are replaced by cheap surrogate model predictions. Existing SAEAs can be mainly classified into two categories. The first one includes those using Gaussian Process (GP) meta-modeling with prescreening method (e.g. expected improvement) [4]. This kind of methods can obtain high quality results in a small number of expensive function evaluations (e.g. 100-500), but is not scalable and works well for small scale problems (e.g. about 5 dimensions) [5]. When the number of dimensions increases a lot, the training of the GP model becomes very expensive and the prediction quality degrades. Hence the prescreening methods are difficult to correctly locate new promising candidates for function evaluations in further optimization. The second category includes those using artificial neural networks [6] or support vector machines [3]-based surrogate models and their prediction values to assist evolutionary algorithms. To obtain good results, those methods often need more function evaluations, but they are scalable to medium scale problems (e.g. 6000-8000 exact function evaluations for problems with 20-30 dimensions [7]). To investigate prescreening methods, this work focuses on the former class.

Prescreening methods rank the candidates according to their qualities and select the appropriate subset to perform exact function evaluations for verification and model updating. Existing prescreening methods are the expected improvement (EI) [5], the probability of improvement (PI) [8], the lower confidence bound (LCB) [9] and the most likely improvement (MI) [10]. Among them, EI is the most commonly used method. EI, PI and LCB reward the prediction uncertainty of the GP model: possible promising candidates that currently without a good predicted value may be selected in the subset for exact function evaluations. This prevents the search from being trapped in a locally optimal area due to the prediction uncertainty. In other words, global search is not only accomplished by the EA, but also by the prescreening methods [5]. Hence, it is not a surprise that such methods have high exploration ability and an additional efficiency enhancement compared to directly using the predicted value. EI, PI and LCB try to call for a balance between exploration and exploitation in different ways. However, we found that none of the existing prescreening methods is general enough to be effective for different problem landscapes (see Section IV). For example, EI performs very well for some multimodal problems, but got
the worst result for some unimodal problems. The problem is that the proper prescreening method for an unknown real-world problem cannot be known beforehand.

To address this problem, a new prescreening method, called self-adaptive lower confidence bound (ALCB), is proposed. ALCB is derived from LCB, but the parameter to control how the prediction uncertainty is rewarded, is adaptively adjusted online based on the density of samples in a local area and the function properties. Experimental results on benchmark problems show that in terms of solution quality, not only good performances can be achieved for different problem landscapes, but also that the results are often better than with all existing prescreening methods. On the efficiency aspect, ALCB does not cost more exact function evaluations compared to the existing prescreening methods.

The rest of the paper is organized as follows. Section II describes basic concepts and the framework of the SAEA used in this paper. Section III first analyzes the existing prescreening methods, and then elaborates the ALCB method. Experimental results based on benchmark problems are shown in section IV. The concluding remarks are presented in section V.

II. THE BASIC SAEA FRAMEWORK

Because the prescreening methods are only suitable for Gaussian Process (GP) surrogate model, GP is briefly introduced first. More details can be found in [10].

GP predicts a function value \( y(x) \) at some design point \( x \) by modeling \( y(x) \) as a stochastic variable with mean \( \mu \) and variance \( \sigma \). For two points \( x_i \) and \( x_j \), their correlation is defined as:

\[
\text{Corr}(x_i, x_j) = \exp(-\sum_{l=1}^{d} \theta_l | x_{il} - x_{jl} |^p_l)
\]

(1)

\[\theta_l > 0, 1 \leq p_l \leq 2\]

where \( d \) is the dimension of \( x \) and \( \theta_l \) is the correlation parameter which determines how fast the correlation decreases when \( x_{il} \) moves in the \( l \) direction. Parameter \( p_l \) is related to the smoothness of the function with respect to \( x_{il} \). The optimal values of \( \mu, \sigma \) and \( \theta_l \) are determined by maximizing the likelihood function of the observed data. The function value \( y(x') \) at a new point \( x' \) can be predicted as:

\[
\hat{y}(x') = \hat{\mu} + r^T R^{-1} (y - I \hat{\mu})
\]

\[
\hat{\mu} = (I^T R^{-1} I)^{-1} I^T R^{-1} y
\]

\[
R_{i,j} = \text{Corr}(x_i, x_j), \quad i, j = 1, 2, \ldots, n
\]

\[r = [\text{Corr}(x', x_1), \text{Corr}(x', x_2), \ldots, \text{Corr}(x', x_n)]^T\]

Vectors \( x = (x_1, x_2, \ldots, x_n) \) and \( y = (y_1, y_2, \ldots, y_n) \) represent the already evaluated data points and their objective function values, respectively. \( I \) is a \((n \times 1)\)-dimensional vector of ones.

The prediction uncertainty is:

\[
\hat{s}^2(x') = \hat{\sigma}^2[I - r^T R^{-1} r + (I - r^T R^{-1} r)(I^T R^{-1} I)^{-1}]
\]

\[
\hat{\sigma}^2 = (y - I \hat{\mu})^T R^{-1} (y - I \hat{\mu}) n^{-1}
\]

(3)

Differential evolution (DE) [11] is used as the search engine in this work. DE uses a simple differential operator to create new candidate solutions and a one-to-one competition scheme to greedily select new candidates. The DE mutation and the crossover operators that are described below are used in this work.

**DE mutation:**

For each target individual \( i \) of the iteration \( t \), a mutant vector for the \((t+1)\)-th iteration is generated:

\[
V_i(t+1) = x_{\text{best}}(t) + F(x_r_1(t) - x_r_2(t))
\]

(4)

where indices \( r_1 \) and \( r_2 \) (\( r_1, r_2 \in \{1, 2, \ldots, NP\} \)) are randomly chosen and are mutually different and also different from the current index \( i \). \( NP \) is the population size. \( x_{\text{best}}(t) \) is the best individual of the current population. \( F \) is the scaling factor.

**DE crossover:**

For each target individual \( i \) of the iteration \( t \), a trial vector for the \((t+1)\)-th iteration is generated whose components are built as:

\[
u_{i,j}(t+1) = \begin{cases} v_{i,j}(t+1), & \text{if } \text{rand}(i, j) \leq CR \text{ or } j = \text{randn}(i), \\ x_{i,j}(t), & \text{otherwise,} \end{cases}
\]

(5)

where \( \text{rand}(i, j) \) is an independent random number uniformly distributed in the range \([0,1]\). Parameter \( \text{randn}(i) \) is a randomly chosen index from the set \( \{1, 2, \ldots, d\} \), where \( d \) is number of dimensions of \( x \). Parameter \( CR \in [0,1] \) is a constant called the crossover parameter.

Using GP and DE as basic elements, an SAEA can be constructed as shown in Algorithm 1.

**Algorithm 1. SAEA using GP surrogate model and DE search engine**

**Step 0:** Initialize the population by Latin Hypercube sampling [12] of the design space. \( 11 \times d-1 \) samples are used [5]. Evaluate the population using exact functions.

**Step 1:** Check if the stopping criterion is met. If yes, output the result; otherwise go to step 2.

**Step 2:** Train the GP surrogate model according to the available samples (population).

**Step 3:** Use the available samples as the current population, and perform mutation and crossover operations according to equations (4)-(5) to obtain each candidate solution’s trial individual.

**Step 4:** According to the values predicted by the GP model from Step 2, use a prescreening method to select the individual with the best potential and evaluate it with exact functions.

**Step 5:** Update the population by adding the point from step 4. Update the best solution obtained so far. Update the training data. Go back to Step 2.

It can be seen from algorithm 1 that only one most promising candidate is evaluated in each iteration. The candidate is decided by the prescreening method. This
framework is used to compare the different prescreening methods in all the experiments.

III. ANALYSIS OF EXISTING PRESCREENING METHODS AND SELF-ADAPTIVE LOWER CONFIDENCE BOUND

In order to propose a new prescreening method having more generality and higher performance, the limitations of the existing prescreening methods need to be investigated first.

A. Analysis of existing prescreening methods

The four existing prescreening methods are MI, EI, PI, and LCB. Considering a minimization problem, they are defined as follows:

1) Most likely improvement (MI):

\[
M(I(x)) = \begin{cases} 
0, & \text{if } \hat{y}(x) > f_{\text{min}} \\
 f_{\text{min}} - \hat{y}(x), & \text{otherwise} 
\end{cases}
\]  

(6)

where \( f_{\text{min}} \) is the current best function value in the population and \( \hat{y}(x) \) is the predicted value of a candidate. \( I(x) \) means the function of improvement compared to \( f_{\text{min}} \).

2) Expected improvement (EI):

\[
E[I(x)] = (f_{\text{min}} - \hat{y}(x))\Phi\left(\frac{f_{\text{min}} - \hat{y}(x)}{s(x)}\right) + \hat{s}(x)\phi\left(\frac{f_{\text{min}} - \hat{y}(x)}{\hat{s}(x)}\right)
\]  

(7)

where \( \phi(\cdot) \) is the standard normal density function, and \( \Phi(\cdot) \) is the standard normal distribution function.

3) Probability of improvement (PI):

\[
P[I(x)] = \Phi\left(\frac{f_{\text{min}} - \hat{y}(x)}{s(x)}\right)
\]  

(8)

4) Lower confidence bound (LCB):

\[
f_{\alpha}(x) = \hat{y}(x) - \alpha s(x), \alpha \in [0, 3]
\]  

(9)

According to (6)-(9), it is clear that MI is very different from the other three prescreening methods. Unlike EI, PI and LCB, MI does not reward any prediction uncertainty and the predicted value is directly used. This means that MI shows the improvement when the response value with maximum probability density function is considered. Therefore, the performance of MI depends on the quality of the GP modeling. In on-line SAEAs, the available samples may be ill distributed, and there may be insufficient information of some areas in the decision space to construct a good local GP model. Hence, MI is more risky than the methods taking prediction uncertainty into account for multimodal problems (class C). The reason is that although EA explores the search space, the prediction values of some promising sub-spaces may not be good and additional samples would not be allocated to those sub-spaces to improve the model. The solution may therefore be trapped in a locally optimal area. However, it is too early to conclude that MI is not suitable for multimodal problems because the smoothing effect [13] may help it to explore the space without a good surrogate model. For class D functions, using low order polynomial regression may make use of the smoothing effect [7]. Nevertheless, whether the smoothing effect can significantly help depends on the problems themselves and often cannot be known beforehand. On the other hand, the advantage of MI is that the exploitation ability is very good when the accuracy of the GP model is acceptable. For typical unimodal problems (class A), SAEAs using MI often perform well [14]. However, for some unimodal problems whose optimal area has a quite flat surface (class B), MI may not work well because the predicted values in that area do not differ much and the prediction error may have a large impact. To summarize, the main drawback of MI is in not rewarding the prediction uncertainty.

EI, PI and LCB reward the prediction uncertainty. Therefore, the quality of a candidate is globally considered. These methods therefore have the ability to jump out of local optima (such as class C). EI, PI and LCB reward prediction uncertainty in different ways. PI rewards candidates with a possibly small improvement but with high probability (\( s \) is small) and candidates with a possible large improvement but with low probability (\( s \) is large). Moreover, a larger \( s \) value is only rewarded when the predicted value of a candidate is worse than the current best function value. The EI values, on the other hand, first decrease and then grow again with the increase of \( s \) [14]. LCB always rewards \( s \) to a fixed extent. For these three methods, although they have good exploration ability, they may select individuals which are not really good but with a large \( s \) value. The extent to reward the prediction uncertainty is in fact a problem of balancing between exploration and exploitation. A larger extent of rewarding emphasizes exploration, and a smaller extent of rewarding emphasizes exploitation. EI and PI consider this balance, but the proposed methods often do not work well for different problem landscapes. EI, PI and LCB have been compared in [14] and they show similar performances in the experiments. It is known that LCB uses an optimistic rewarding criterion.

![Fig. 1. Some typical function surfaces](image)

Some interesting shapes of function surfaces to analyze prescreening methods are provided in Fig. 1. Class A is a typical unimodal function surface. Class B is also a unimodal function surface but the area near the global optimal solution is quite flat. Class C is a typical complex multimodal function, whose global and local optimal points have a large difference and there are numerous local optima. Class D is also a multimodal function, but the values of the global and local optima do not differ much and the surface seems to be repeating itself with small vibrations.
Hence, it can be seen that the balance between exploration and exploitation of EI and PI does not work as good as expected from the result of [14]. More explorations are often made and local exploitation may not be sufficient in many cases. The experiments in Section IV also confirm this problem. To summarize, the main problem of EI, PI and LCB is the over-rewarding of the prediction uncertainty in many cases.

B. Self-adaptive lower confidence bound

From the above analysis, it can be concluded that the key factor to make a good balance between exploration and exploitation is to appropriately reward the prediction uncertainty. Taking LCB for example, it is intuitive that for a candidate with a predicted value within 0.5 $\hat{s}$ of the true function value, using 2 $\hat{s}$ to reward and select the promising candidate is not good for exploitation. To decide on the appropriate extent of rewarding, $\hat{s}$ can give us key information. According to (3), $\hat{s}$ is decided by two factors: (1) the property of the function itself which controls the $y$ value in $\sigma^2$; and (2) the density of the available samples in a local area, which affects the $R$ values. Therefore, $\hat{s}$ results from the combined effect of the function property and the density of the samples in a local area. In SAEAs, the prediction uncertainty should tend to decrease in the evolution process since more samples are becoming available and the density of samples is becoming higher. For example, in the beginning of the optimization, the real values of most candidates may be within 2 $\hat{s}$ of the prediction value, while at the end of the optimization, it may be within 1 $\hat{s}$. Also, for different problems, the rate of decrease of the prediction uncertainty throughout the evolution process may be different. For example, for some problems, the promising candidates near the end of the optimization may be within 1 $\hat{s}$ of the prediction values, while for some other problems, it may be within 1.5 $\hat{s}$. Therefore, adaptive methods, whose rewarding criterion is determined beforehand like EI, PI or LCB, can hardly obtain “appropriate rewarding”. Therefore, a self-adaptive method is necessary.

The adaptive methods are integrated implicitly in the formula for EI and PI. LCB, on the other hand, has an explicit parameter $\omega$ describing the extent of rewarding. A larger $\omega$ emphasizes more on exploration, and a smaller $\omega$ emphasizes more on exploitation. Hence, we select to revise LCB (see (9)). Instead of using $\omega = 2$ like most of the existing works, our algorithm adaptively revises $\omega$ according to candidates with similar $\hat{s}$ values. A database is first constructed. For the $11 \times d$-1 initial samples, there is no predicted value. Apart from these, the promising candidates from the first $N_t$ iterations construct the database. In these iterations, $\omega = 2$ is used. For all the individuals in the database, the real function values, the $\hat{s}$ values, and the predicted function values are available. We then calculate the deviation between the predicted values and the real function values, which are classified into 5 classes: $\leq 0.5\hat{s}$, $0.5\hat{s} - 1\hat{s}$, $1\hat{s} - 2\hat{s}$, $2\hat{s} - 3\hat{s}$, $> 3\hat{s}$. The deviation values are rounded upwards to the larger value in each class, while for the last class, $3\hat{s}$ is used. Then, each individual in the database has an extent of deviation, such as $0.5\hat{s}$, $2\hat{s}$. Only values of 0.5,1,2,3 are allowed to represent the extent of deviation. Once a new candidate is evaluated, it is added to the database. The inputs of ALCB are a group of trial individuals after DE operations, $P$. After GP prediction, each individual in $P$ has its predicted value and its $\hat{s}$ value. The output of ALCB is the selected most promising candidate. The ALCB algorithm works as follows.

**Algorithm 2. Self-adaptive LCB**

For each trial individual $ind_i$ in $P$,

1. Find $C$ individuals from the database whose $\hat{s}$ values have the nearest distances to the $\hat{s}$ value of $ind_i$.
2. Calculate the mean value $mv$ of the extent of deviation of the $C$ individuals.
3. Round $mv$ upwards to the nearest allowed value greater than or equal to it and this value is assigned to $\omega_i$ (see (9)).

End

Use (9) to calculate $f_{\omega_i}(P)$ of the whole group. Select the best candidate. Note that $\omega$ is a vector now.

Because the $\hat{s}$ value reflects the combined effect of the function and the density of samples in a local area, the experiences of other points with similar $\hat{s}$ values are very useful to foresee the extent of reward of a new point. Therefore, $C$ individuals with the nearest $\hat{s}$ values are selected as the reference. Another point that needs to be mentioned is that only 4 allowed rounding values are used to express the extent of deviation. The reason is that the estimation error when predicting $\omega$ for a new point is unavoidable. Hence, there is a balance between the quality of the prescreening and the accurate estimation of $\omega$. Assume that only one allowed value $3\omega$ would be used, then most estimations of the extent of deviation are right (most real values are within $3\hat{s}$ in the GP model), but appropriate rewarding cannot be achieved. In contrast, if we would use many allowed values, it is good for using different levels of rewarding, but many estimations of the extent of rewarding may be wrong, such as $0.2\omega$ is predicted to be $0.25\omega$. Therefore, the use of 4 allowed values is a balance between the above two factors.

IV. EXPERIMENTAL RESULTS

In this section, the ALCB method is tested on benchmark problems with different types of surfaces. Problems with 4 and 6 dimensions are used. The problems (see Appendix) used for test purposes are: the Ellipsoid function (4 dimensional, F1; 6 dimensional F2), the Rosenbrock function (4 dimensional, F3; 6 dimensional F4), the Rastrigin function (4 dimensional, F5; 6 dimensional F6), the Ackley function (4 dimensional, F7; 6 dimensional F8) and the Griewank function (4 dimensional, F9; 6 dimensional F10). 20 runs are performed for each problem: 500 exact function evaluations
are used in each run, and many runs converge within 100-200 exact function evaluations. For the GP model, the exponential correlation function [15] is used. The scaling factor of DE mutation and the crossover rate are set to 0.8. The C value in ALCB (see “algorithm 2”) is set to 10. The average results of 20 runs of EI, MI, LCB (with $\omega = 2$) and ALCB are compared in Table 1. The statistical results are shown in Fig. 2 to Fig. 6, where error bar is used, showing the mean value and standard deviations of EI, MI, LCB and ALCB.

Table 1. Results of EI, MI, LCB and ALCB for the test problems (mean value)

<table>
<thead>
<tr>
<th>Problems</th>
<th>EI</th>
<th>MI</th>
<th>LCB</th>
<th>ALCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>2.33e-6</td>
<td>7.55e-10</td>
<td>4.49e-9</td>
<td>2.67e-9</td>
</tr>
<tr>
<td>F2</td>
<td>2.71e-4</td>
<td>1.53e-5</td>
<td>1.97e-5</td>
<td>1.81e-5</td>
</tr>
<tr>
<td>F3</td>
<td>1.59</td>
<td>2.00</td>
<td>1.45</td>
<td>1.41</td>
</tr>
<tr>
<td>F4</td>
<td>3.58</td>
<td>4.18</td>
<td>3.40</td>
<td>3.13</td>
</tr>
<tr>
<td>F5</td>
<td>1.29</td>
<td>4.77</td>
<td>0.65</td>
<td>0.49</td>
</tr>
<tr>
<td>F6</td>
<td>1.76</td>
<td>4.28</td>
<td>1.35</td>
<td>0.92</td>
</tr>
<tr>
<td>F7</td>
<td>6.16e-4</td>
<td>1.22</td>
<td>1.22e-2</td>
<td>5.24e-4</td>
</tr>
<tr>
<td>F8</td>
<td>0.019</td>
<td>1.76</td>
<td>0.020</td>
<td>0.015</td>
</tr>
<tr>
<td>F9</td>
<td>0.24</td>
<td>0.12</td>
<td>0.15</td>
<td>0.08</td>
</tr>
<tr>
<td>F10</td>
<td>0.51</td>
<td>0.27</td>
<td>0.41</td>
<td>0.38</td>
</tr>
</tbody>
</table>

From Table 1 and Fig. 2 to Fig. 6, some conclusions can be drawn. MI performs best for typical unimodal problems (class A, F1, F2). LCB and ALCB got comparable, but slightly worse, results than MI. MI also performs well for the class D problem (F9, F10), where the smoothing effect helps to explore the space. However, for typical multimodal problems (class C, F5-F8), MI provides very bad results. For unimodal problems with a flat surface near its optimal point (class B,
F3, F4), MI also performs bad. For EI, much better exploration ability can be seen from the results of the class C problems (F5-F8). However, its exploitation ability is often worse than ALCB, which is shown by the results of F5-F8. In unimodal problems (F1,F2,F9,F10), EI gives the worst result of all the methods. Compared to EI, LCB with \( \omega = 2 \) shows a better performance in some cases and a worse performance in some other cases. In all the test problems, the results of ALCB are better than EI and LCB, and sometimes much better. It can be seen that the exploitation ability is enhanced a lot, while maintaining the exploration ability. In three of the unimodal problems (F1, F2, F10), MI got the best result, but ALCB also got satisfactory results, often comparable to the results of MI. Therefore, we can conclude that ALCB is general to different problem landscapes and the performance is enhanced compared to the available prescreening methods.

V. CONCLUSIONS

In this work, the advantages and drawbacks of the existing prescreening methods for surrogate model assisted evolutionary algorithms have been analyzed and a new prescreening method, the self-adaptive lower confidence bound (ALCB) method, is proposed. ALCB is general to different problem landscapes and the performance is enhanced compared to the existing prescreening methods. This is achieved by on-line adjusting to which extent the prediction uncertainty is rewarded, which is provided by Gaussian Process surrogate model. The goal of ALCB is to call for a good balance between exploration and exploitation of the prescreening method. The experimental results show the advantages of ALCB. Future works will focus on including local search in ALCB to further enhance its exploitation ability.

ACKNOWLEDGMENT

This research was supported by a special bilateral agreement scholarship of Katholieke Universiteit Leuven, Belgium and Tsinghua University, P. R. China.

APPENDIX

Benchmark functions:

A. Ellipsoid Problem

\[
\begin{align*}
\min f(x) & = \sum_{i=1}^{d} x_i^2 \\
x_i & \in [-5.12, 5.12], i = 1, \ldots, d
\end{align*}
\]

Known minimum: \( x^* = 0, f(x^*) = 0 \).

B. Rosenbrock Problem

\[
\begin{align*}
\min f(x) & = \sum_{i=1}^{d} (100(x_{i+1} - x_i^2)^2 + (1-x_i)^2) \\
x & \in [-2.048, 2.048], i = 1, \ldots, d
\end{align*}
\]

Known minimum: \( x^* = 1, f(x^*) = 0 \).

C. Rastrigin Problem

\[
\begin{align*}
\min f(x) & = 10d + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i)) \\
x_i & \in [-5.12, 5.12], i = 1, \ldots, d
\end{align*}
\]

Known minimum: \( x^* = 0, f(x^*) = 0 \).

D. Ackley Problem

\[
\begin{align*}
\min f(x) & = -20\exp(-0.2\sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2}) - \exp\left(\frac{1}{d} \sum_{i=1}^{d} \cos(2\pi x_i)\right) \\
x_i & \in [-32.768, 32.768], i = 1, \ldots, d
\end{align*}
\]

Known minimum: \( x^* = 0, f(x^*) = 0 \).

E. Griewank Problem

\[
\begin{align*}
\min f(x) & = 1 + \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) \\
x_i & \in [-600, 600], i = 1, \ldots, d
\end{align*}
\]

Known minimum: \( x^* = 0, f(x^*) = 0 \).

REFERENCES