A.1 Unconstrained Uni-Modal Test Functions

This section includes five unconstrained, uni-modal test functions, none of which should pose a problem for a robust optimizer. Not all sources agree on the initial parameter bounds for these functions, but in practice these variations do not dramatically affect run times or the probability of success. For many EAs, the most difficult function to optimize in this uni-modal test bed is the generalized Rosenbrock function. In addition, some GAs may have problems solving Schwefel's ridge function because it is a highly eccentric, rotated hyper-ellipsoid with dependent parameters.

A.1.1 Sphere

This simple function tests a search method's local optimization speed and its response to changing dimension. To accommodate bit-encoded GAs, early test beds usually specified the initial parameter bounds as [–5.12, 5.12], but Yao and Liu's more recent and widely referenced test bed (Yao and Liu 1997) initializes parameters with values chosen from the interval [–100, 100].

$$
\begin{aligned}
& f(\mathbf{x}) = \sum_{j=0}^{D-1} x_j^2, \\
& -100 \le x_j \le 100, \quad j = 0,2,...,D-1, \\
& f(\mathbf{x}^*) = 0, \quad x_j^* = 0, \quad \varepsilon = 1.0 \times 10^{-6}.
\end{aligned}
\tag{A.1}
$$

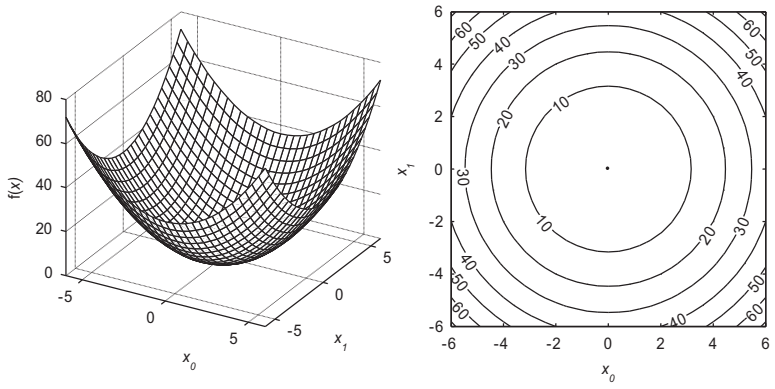

Fig. A.1. The sphere function

A.1.2 Hyper-Ellipsoid

Some literature specifies [–5.12, 5.12] as the bounds for initializing this function, but this book adopts the limits given in Yao and Liu (1997). To decrease the eccentricity of the hyper-ellipsoid, some versions of this function use a term like $(j + 1)^2$ as the pre-factor to x_j instead of putting the parameter index in the exponent. This function can take a long time to solve if an optimizer cannot adapt step sizes to suit each dimension.

$$
\begin{aligned}
& f(\mathbf{x}) = \sum_{j=0}^{D-1} 2^j \cdot x_j^2, \\
& -100 \le x_j \le 100, \quad j = 0,1,...,D-1, \\
& f(\mathbf{x}^*) = 0, \quad x_j^* = 0, \quad \varepsilon = 1.0 \times 10^{-6}.
\end{aligned}
\tag{A.2}
$$

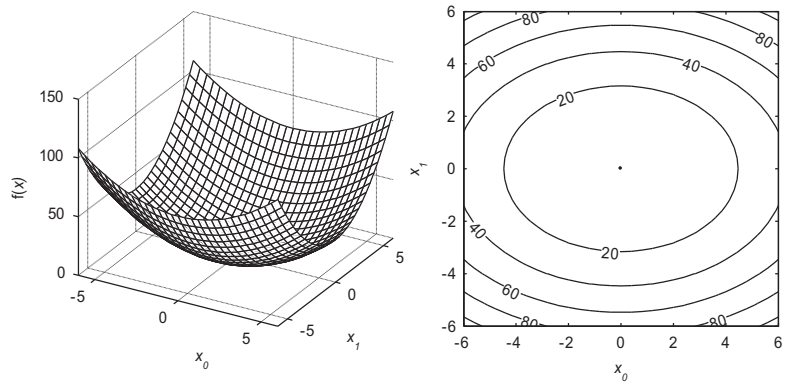

Fig. A.2. The unrotated hyper-ellipsoid

A.1.3 Generalized Rosenbrock

The original Rosenbrock function was just two-dimensional, but it was later generalized to this higher-dimensional version. The ridge in Fig. A.3 shows that this uni-modal function is non-convex. This function exhibits limited parameter dependence that poses a problem for many optimizers. Some studies use [–2.048, 2.048], while others use [–5.12, 5.12] for initial parameter bounds. Yao and Liu initialized parameters with values chosen from [–32, 32], but initial parameter bounds were [–30, 30] for studies in this book.

$$f(\mathbf{x}) = \sum_{j=0}^{D-2} \left(100 \cdot \left(x_{j+1} - x_j^2 \right)^2 + \left(x_j - 1 \right)^2 \right), \tag{A.3}$$
$$-30 \le x_j \le 30, \quad j = 0,1,...,D-1, \quad D > 1,$$
$$f(\mathbf{x}^*) = 0, \quad x_j^* = 1, \quad \varepsilon = 1.0 \times 10^{-6}.$$

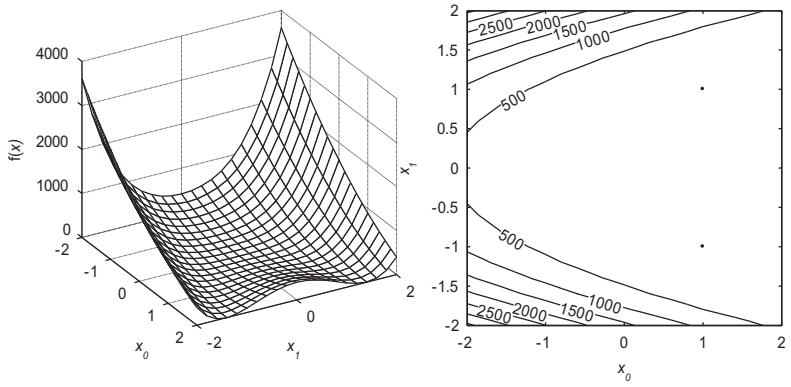

Fig. A.3. Rosenbrock's function

A.1.4 Schwefel's Ridge

When this function is posed as a minimization problem, the "ridge" in its landscape becomes an elliptical "valley". For some EAs, adapting to both the orientation and high eccentricity of the ellipse can be a significant challenge. Some studies have used [–65.536, 65.536] as initial parameter bounds, but this book adopts the bounds published in Yao and Liu (1997).

$$f(\mathbf{x}) = \sum_{k=0}^{D-1} \left(\sum_{j=0}^{k} x_j \right)^2, \tag{A.4}$$
$$-100 \le x_j \le 100, \quad j = 0,1,...,D-1,$$
$$f(\mathbf{x}^*) = 0, \quad x_j^* = 0, \quad \varepsilon = 1.0 \times 10^{-6}.$$

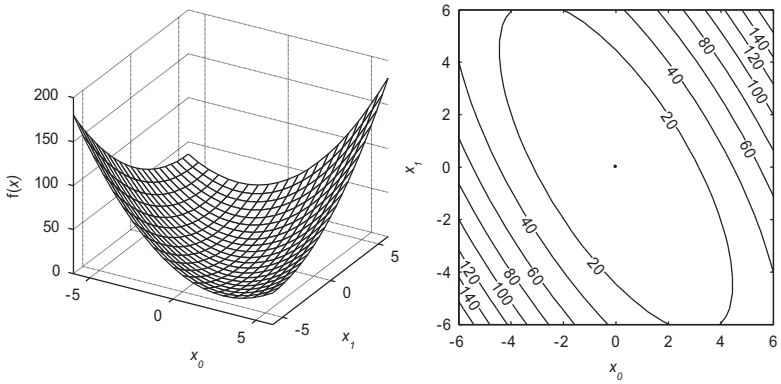

Fig. A.4. Schwefel's ridge function

A.1.5 Neumaier #3

This function also displays elliptical contours that are aligned with coordinate diagonals, but the optimum is not centered in the initial bounding box.

$$
\begin{aligned}
& f(\mathbf{x}) = \sum_{j=0}^{D-1} (x_j - 1)^2 - \sum_{j=1}^{D-1} x_j \cdot x_{j-1}, \quad D > 1, \\
& -D^2 \le x_j \le D^2, \quad j = 0,1,...,D-1, \quad \varepsilon = 1.0 \times 10^{-6}, \\
& f(\mathbf{x}^*) = -D \cdot (D+4) \cdot (D-1)/6, \quad x_j^* = (j+1) \cdot (D-j).
\end{aligned}
\tag{A.5}
$$

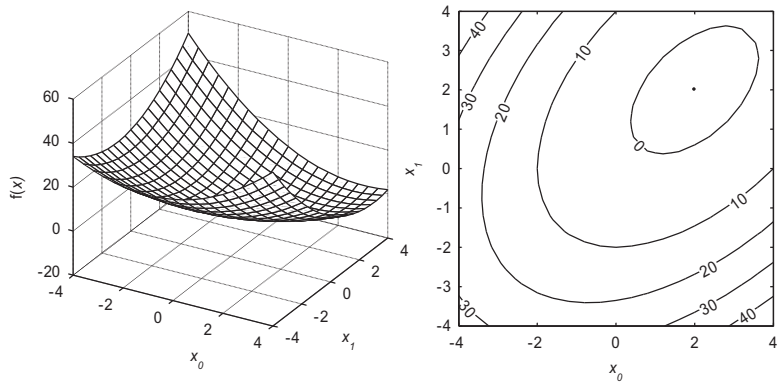

Fig. A.5. Neumaier's function number three

A.2 Unconstrained Multi-Modal Test Functions

Uni-modal functions can reveal how an algorithm responds to dimension, parameter dependence and disparities in step size, but few practical problems are so simple. The following multi-modal functions range from moderately challenging to very difficult depending in part on the dimension at which they are evaluated and on the amount of special knowledge about the function that an optimizer exploits. Not all functions can be evaluated at arbitrarily high dimensions.

A.2.1 Ackley

One of the most commonly cited multi-modal test functions is Ackley's function. At high dimension (e.g., $D \geq 30$), care must be taken with computer code to ensure a precise result. For example, the constant $e = 2.71828\ldots$ in Eq. A.6 is best implemented as $e = \exp(1)$. Bounds are usually given as [–32, 32], but this book uses [–30, 30].

$$f(\mathbf{x}) = -20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{D} \cdot \sum_{j=0}^{D-1} x_j^2}\right) - \exp\left(\frac{1}{D} \cdot \sum_{j=0}^{D-1} \cos(2\pi \cdot x_j)\right) + 20 + e, \tag{A.6}$$

$$-30 \leq x_j \leq 30, \quad j = 0,1,...,D-1,$$

$$f(\mathbf{x}^*) = 0, \quad x_j^* = 0, \quad \varepsilon = 1.0 \times 10^{-6}.$$

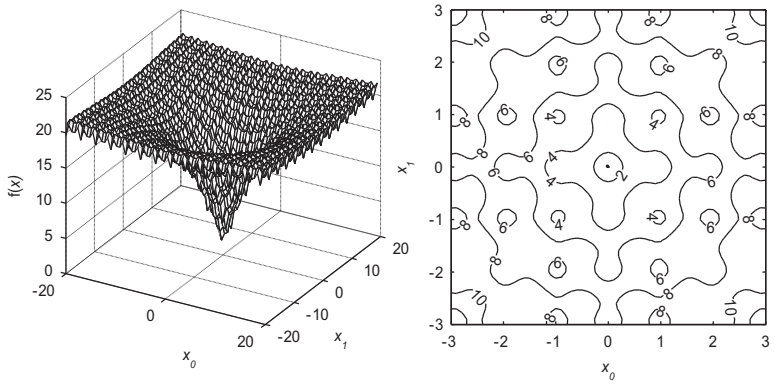

Fig. A.6. The Ackley function at large scale

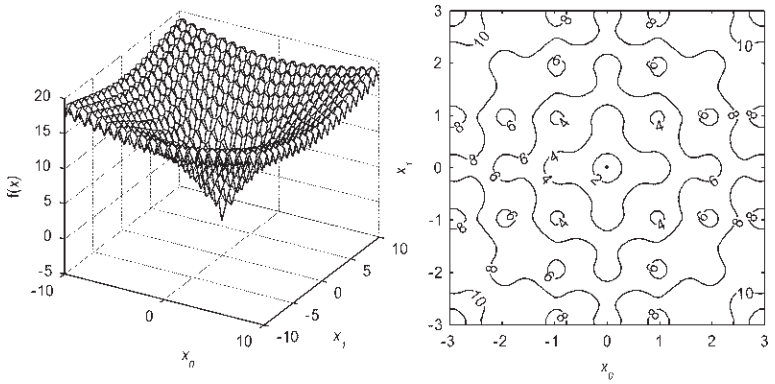

Fig. A.7. The Ackley function at small scale

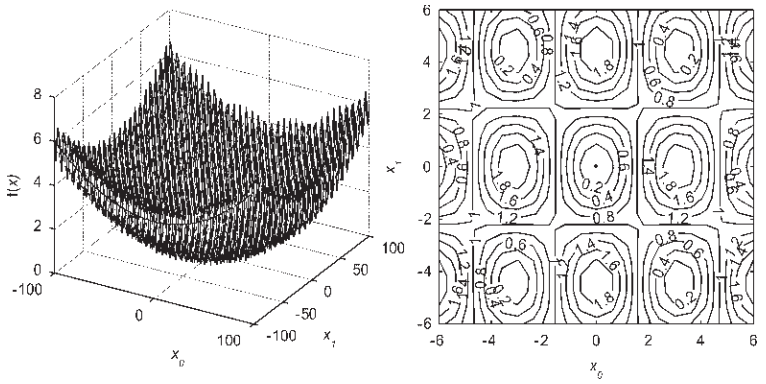

Fig. A.8. Griewangk's function

A.2.2 Griewangk

This mildly parameter-dependent function becomes relatively easier to solve as D increases. The summation term creates a parabolic bowl while the product of cosine terms generates the local optima. As D increases, the contribution from the cosine terms becomes less significant and the local basins of attraction become shallower. At the same time, the relative size of the optimal basin of attraction increases. See Whitley et al. (1996) for details. It is not uncommon that this function will require a relatively large population to forestall premature convergence.

$$f(\mathbf{x}) = \frac{1}{4000} \cdot \sum_{j=0}^{D-1} x_j^2 - \prod_{j=0}^{D-1} \cos\left(\frac{x_i}{\sqrt{j+1}}\right) + 1, \tag{A.7}$$
$$-600 \le x_j \le 600, \quad j = 0,1,...,D-1,$$
$$f(\mathbf{x}^*) = 0, \quad x_j^* = 0, \quad \varepsilon = 1.0 \times 10^{-6}.$$

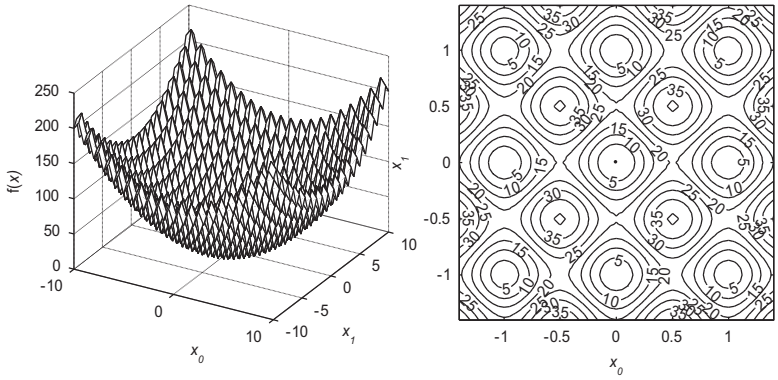

Fig. A.9. Rastrigin's function

A.2.3 Rastrigin

Like the Ackley and Griewangk functions, Rastrigin's function has many local optima arrayed on the side of a larger bowl-shaped depression. This function is separable as written and easily solved by methods that can exploit decomposable functions. It is *much* harder to solve when rotated. Like Rosenbrock's function, Rastrigin's function is a generalization of a two-dimensional function. Like the Ackley and Griewangk's functions, Rastrigin's function is symmetric about its solution. Optimizers that search the vicinity of the mean population vector will do well on these symmetric functions because, like the local minima, the population will also be symmetrically distributed.

$$f(\mathbf{x}) = \sum_{j=0}^{D-1} \left(x_j^2 - 10 \cdot \cos(2\pi \cdot x_j) + 10 \right), \tag{A.8}$$
$$-5.12 \le x_j \le 5.12, \quad j = 0,1,...,D-1,$$
$$f(\mathbf{x}^*) = 0, \quad x_j^* = 0, \quad \varepsilon = 1.0 \times 10^{-6}.$$

A.2.4 Salomon

The landscape for this parameter-dependent function resembles a pond with ripples. Because this function is symmetric, methods that search the vicinity of the population's mean vector will likely perform well.

$$f(\mathbf{x}) = -\cos(2\pi \cdot \|\mathbf{x}\|) + 0.1 \cdot \|\mathbf{x}\| + 1, \tag{A.9}$$
$$\|\mathbf{x}\| = \sqrt{\sum_{j=0}^{D-1} x_j^2},$$
$$-100 \le x_j \le 100, \quad j = 0,1,...,D-1,$$
$$f(\mathbf{x}^*) = 0, \quad x_j^* = 0, \quad \varepsilon = 1.0 \times 10^{-6}.$$

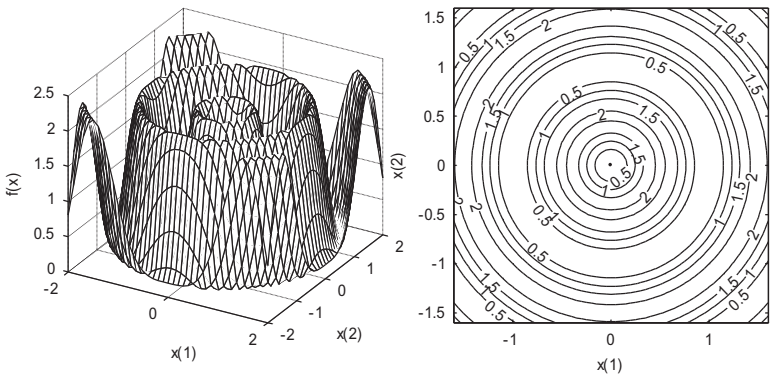

Fig. A.10. Salomon's function

A.2.5 Whitley

Whitley's function is a composition of the Griewangk and Rosenbrock functions. This implementation uses the unweighted full matrix expansion detailed in Whitley et al. (1996). This function's landscape resembles Rosenbrock's function at large scale and Griewangk's function at small scale.

$$f(\mathbf{x}) = \sum_{k=0}^{D-1}\sum_{j=0}^{D-1}\left(\frac{y_{j,k}^2}{4000} - \cos(y_{j,k}) + 1\right), \tag{A.10}$$

$$y_{j,k} = 100\cdot\left(x_k - x_j^2\right)^2 + (1 - x_j)^2,$$

$$-100 \le x_j \le 100, \quad j = 0,1,...,D-1,$$

$$f(\mathbf{x}^*) = 0, \quad x_j^* = 1, \quad \varepsilon = 1.0\times 10^{-6}.$$

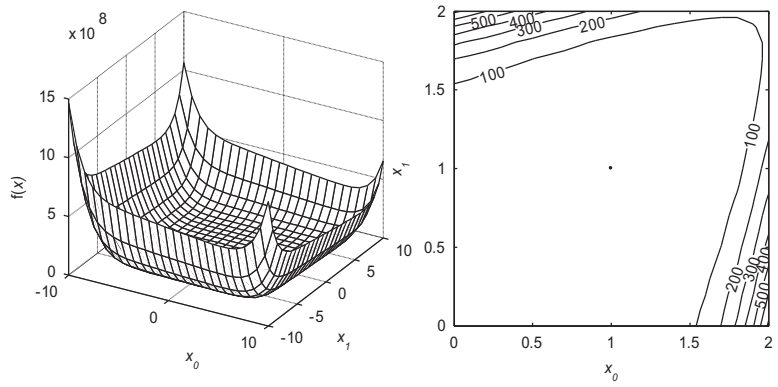

Fig. A.11. Whitley's function (large scale)

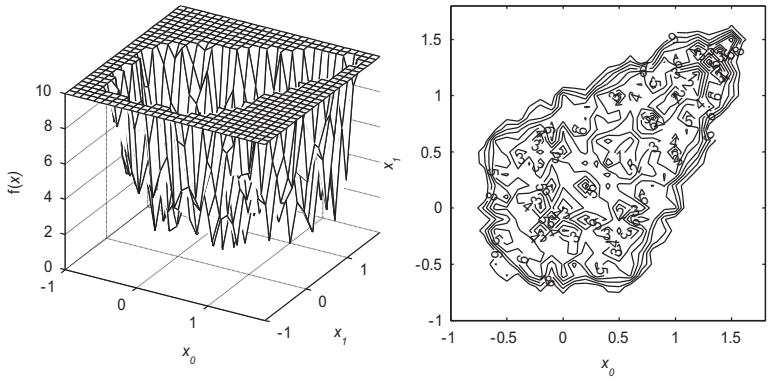

Fig. A.12. Whitley's function with values above 10 clipped

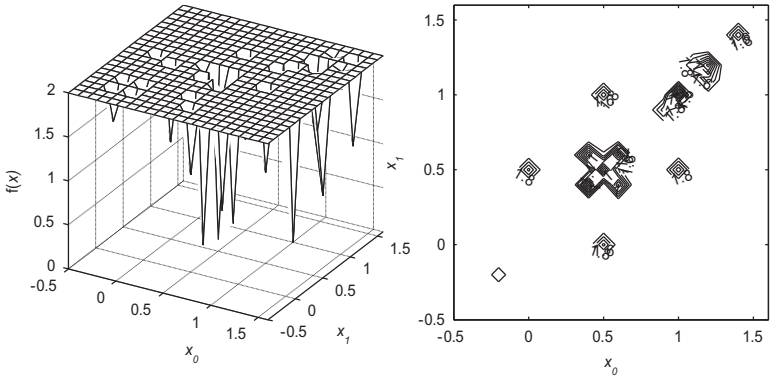

Fig. A.13. Whitley's function with values above 2 clipped

A.2.6 Storn's Chebyshev

The goal of this 2nd ICEO problem (Second ICEO 1997) is to find the coefficients of a polynomial such that the value of the polynomial oscillates between 1 and –1 as its argument, z, varies in the same range. In addition, the polynomial's value is also constrained when $z = 1.2$ and $z = -1.2$. The solution gives the coefficients of a Chebyshev polynomial. The coefficients for a Chebyshev polynomial of degree $D - 1$ can be expressed recursively as $T_{D+1}(z) = 2z \cdot T_D(z) - T_{D-1}(z)$, $D > 0$ and odd, $T_0(z) = 1$, $T_1(z) = z$. The objective function is designed as a three-term error function. The term, p_k, is

the sum of $m + 1$, regularly sampled, squared deviations of the trial vector's objective function value in the [–1, 1] containment zone. Optimal parameter values for this problem grossly differ in magnitude. The picture of the two-dimensional version of this function does not give any indication of the multiple local optima that occur at higher dimensions.

$$
\begin{aligned}
&f(\mathbf{x}) = p_1 + p_2 + p_3, \qquad \text{(A.11)}\\
&p_1 = \begin{cases} (u-d)^2 & \text{if } u < d \\ 0 & \text{otherwise} \end{cases}, \qquad u = \sum_{j=0}^{D-1} x_j \cdot (1.2)^{D-1-j},\\
&p_2 = \begin{cases} (v-d)^2 & \text{if } v < d \\ 0 & \text{otherwise} \end{cases}, \qquad v = \sum_{j=0}^{D-1} x_j \cdot (-1.2)^{D-1-j},\\
&p_k = \begin{cases} (w_k-1)^2 & \text{if } w_k > 1 \\ (w_k+1)^2 & \text{if } w_k < -1, \\ 0 & \text{otherwise} \end{cases} \quad w_k = \sum_{j=0}^{D-1} x_j \cdot \left(\frac{2k}{m} - 1\right)^{D-1-j},\\
&p_3 = \sum_{k=0}^{m} p_k, \quad k = 0,1,...,m, \quad m = 32 \cdot D,\\
&d = T_{D-1}(1.2) \approx \begin{cases} 72.661 & \text{for } D = 9 \\ 10558.145 & \text{for } D = 17 \end{cases}\\
&-2^D \le x_j \le 2^D, \quad j = 0,1,...,D-1, \quad D > 1 \text{ and odd}, \quad \varepsilon = 1.0 \times 10^{-8}\\
&f(\mathbf{x}^*) = 0,\\
&\mathbf{x}^* = \begin{cases} [128, 0, -256, 0, 160, 0, -32, 0, 1] & \text{for } D = 9 \\ [32768, 0, -131072, 0, 212992, 0, -180224, & \\ \quad 0, 84480, 0, -21504, 0, 2688, 0, -128, 0, 1] & \text{for } D = 17. \end{cases}
\end{aligned}
$$

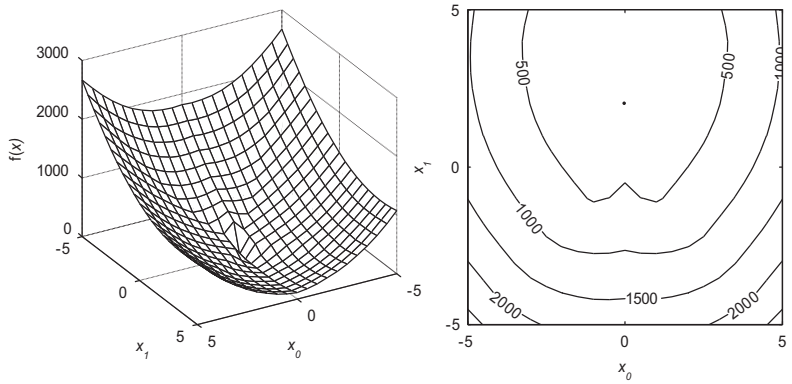

Fig. A.14. Storn's Chebyshev polynomial fitting problem

A.2.7 Lennard-Jones

This problem is based on the Lennard-Jones atomic potential energy function. The goal is to position n atoms in three-dimensional space to minimize their total potential energy. Since neither the cluster's position nor its orientation is specified, optimal parameter values are not unique.

$$f(\mathbf{x}) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \left(\frac{1}{d_{i,j}^2} - \frac{2}{d_{i,j}} \right), \quad d_{i,j} = \left(\sum_{k=0}^{2} \left(x_{3i+k} - x_{3j+k} \right)^2 \right)^3, \tag{A.12}$$

$$-2 \le x_j \le 2, \quad j = 0,1,...,D-1, \quad D = 3 \cdot n, \quad n = 2,3,... \quad \varepsilon = 0.01.$$

Table A.1. Optimal function values for n=2 to n=19 "atoms"

n	$f(\mathbf{x}^*)$	n	$f(\mathbf{x}^*)$
2	–1.0	11	–37.967600
3	–3.0	12	–44.326801
4	–6.0	13	–47.845157
5	–12.712062	14	–52.322627
6	–16.505384	15	–56.815742
7	–19.821489	16	–61.317995
8	–24.113360	17	–66.530949
9	–28.422532	18	–72.659782
10	–32.765970	19	–77.177704

A.2.8 Hilbert

The elements of an $n \times n$ Hilbert matrix, **H**, are $h_{i,j}=1 / (i + j + 1)$, $i = 0, 1, 2, \ldots, n - 1$, $j = 0, 1, 2, \ldots, n - 1$. The goal of this problem is to find $\mathbf{H}^{-1}$, the inverse Hilbert matrix. Because it is ill defined, $\mathbf{H}^{-1}$ becomes increasingly difficult to accurately compute as n increases. For this function, parameters in **x** ($D = n^2$) are first are mapped to a square matrix, **Z**. Next, the identity matrix, **I**, is subtracted from the matrix product **HZ**. The (error) function returns the sum of the absolute value of the elements of **W** = **HZ**–**I**. Like the Chebyshev problem, parameter values are of grossly different magnitude. Equation A.13 provides a sample result for $D = 9$ ($n = 3$).

$$f(\mathbf{x}) = \sum_{i=0}^{n-1}\sum_{k=0}^{n-1} |w_{i,k}|, \tag{A.13}$$

$$\mathbf{HZ} - \mathbf{I} = \mathbf{W} = (w_{i,k}), \quad \mathbf{I} = \begin{bmatrix} 1 & 0 \ldots 0 \\ 0 & 1 \ldots 0 \\ \vdots & \\ 0 & 0 \ldots 1 \end{bmatrix}$$

$$\mathbf{H} = (h_{i,k}), \quad h_{i,k} = \frac{1}{i+k+1}, \quad i,k = 0,1,\ldots,n-1, \quad D = n^2,$$

$$\mathbf{Z} = (z_{i,k}), \quad z_{i,k} = x_{i+nk},$$

$$-2^D \le x_j \le 2^D, \quad j = 0,1,\ldots,D-1, \quad \varepsilon = 1.0 \times 10^{-8},$$

$$f(\mathbf{x}^*) = 0,$$

$$\mathbf{Z}^* = \begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix}, \quad \text{for } n = 3.$$

A.2.9 Modified Langerman

This 2nd ICEO function (Second ICEO 1997) function relies on a vector (**c** in Table A.2) and a matrix (**A** in Table A.3) of real-valued constants. The vector, **c**, contains thirty constants, while **A** is a matrix that contains the coordinates of thirty points in ten dimensions. Points are indexed by rows and coordinates are indexed by columns, e.g., numbers in the k^{th} row are the coordinates of the point $\mathbf{A}_k$, $k = 0, 1, 2, \ldots, 29$. The optimum is the

point in **A** that has the lowest corresponding value of **c**. Although originally designed to use all thirty points in **A**, this implementation, like the code posted for the 2nd ICEO, uses only the first five. Data for both **c** and **A** are available on the CD-ROM that accompanies this book.

$$f(\mathbf{x}) = \sum_{k=0}^{m-1} c_k \cdot \left(\exp\left(\frac{-\left\|\mathbf{x} - \mathbf{A}_k\right\|^2}{\pi} \right) \cdot \cos\left(\pi \cdot \left\|\mathbf{x} - \mathbf{A}_k\right\|^2\right) \right), \tag{A.14}$$

$$\left\|\mathbf{x} - \mathbf{A}_k\right\|^2 = \sum_{j=0}^{D-1} \left(x_j - a_{k,j}\right)^2, \quad k = 0,1,...m-1 \le 29,$$

$$\mathbf{c} = \left(c_k\right), \quad \mathbf{A} = \left(a_{k,j}\right),$$

$$0 \le x_j \le 10, \quad j = 0,1,...,D-1, \quad D \le 10, \quad \varepsilon = 0.001,$$

$$\text{for } m = 5 : f\left(\mathbf{x}^*\right) = \mathbf{c}_5 = -0.96500, \quad \mathbf{x}^* = \mathbf{A}_5.$$

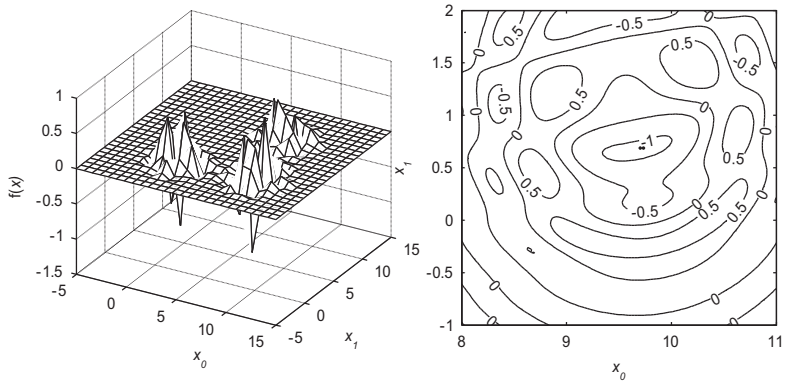

Fig. A.15. The Modified Langerman function

Table A.2. Values for **c**=(c_k)

k	c_k	k	c_k	k	c_k	k	c_k	k	c_k
0	0.806	6	0.524	12	0.463	18	0.828	24	0.332
1	0.517	7	0.902	13	0.714	49	0.964	25	0.817
2	0.100	8	0.531	14	0.352	20	0.789	26	0.632
3	0.908	9	0.876	15	0.869	21	0.360	27	0.883
4	0.965	10	0.462	16	0.813	22	0.369	28	0.608
5	0.669	11	0.491	17	0.811	23	0.992	29	0.326

Table A.3. Values for $\mathbf{A}=(a_{j,k})$. The columns are counted by j (parameter index) while the points, $\mathbf{A}_k$, are numbered by row and are counted by k.

9.681	0.667	4.783	9.095	3.517	9.325	6.544	0.211	5.122	2.020
9.400	2.041	3.788	7.931	2.882	2.672	3.568	1.284	7.033	7.374
8.025	9.152	5.114	7.621	4.564	4.711	2.996	6.126	0.734	4.982
2.196	0.415	5.649	6.979	9.510	9.166	6.304	6.054	9.377	1.426
8.074	8.777	3.467	1.863	6.708	6.349	4.534	0.276	7.633	1.567
7.650	5.658	0.720	2.764	3.278	5.283	7.474	6.274	1.409	8.208
1.256	3.605	8.623	6.905	4.584	8.133	6.071	6.888	4.187	5.448
8.314	2.261	4.224	1.781	4.124	0.932	8.129	8.658	1.208	5.762
0.226	8.858	1.420	0.945	1.622	4.698	6.228	9.096	0.972	7.637
7.305	2.228	1.242	5.928	9.133	1.826	4.060	5.204	8.713	8.247
0.652	7.027	0.508	4.876	8.807	4.632	5.808	6.937	3.291	7.016
2.699	3.516	5.874	4.119	4.461	7.496	8.817	0.690	6.593	9.789
8.327	3.897	2.017	9.570	9.825	1.150	1.395	3.885	6.354	0.109
2.132	7.006	7.136	2.641	1.882	5.943	7.273	7.691	2.880	0.564
4.707	5.579	4.080	0.581	9.698	8.542	8.077	8.515	9.231	4.670
8.304	7.559	8.567	0.322	7.128	8.392	1.472	8.524	2.277	7.826
8.632	4.409	4.832	5.768	7.050	6.715	1.711	4.323	4.405	4.591
4.887	9.112	0.170	8.967	9.693	9.867	7.508	7.770	8.382	6.740
2.440	6.686	4.299	1.007	7.008	1.427	9.398	8.480	9.950	1.675
6.306	8.583	6.084	1.138	4.350	3.134	7.853	6.061	7.457	2.258
0.652	2.343	1.370	0.821	1.310	1.063	0.689	8.819	8.833	9.070
5.558	1.272	5.756	9.857	2.279	2.764	1.284	1.677	1.244	1.234
3.352	7.549	9.817	9.437	8.687	4.167	2.570	6.540	0.228	0.027
8.798	0.880	2.370	0.168	1.701	3.680	1.231	2.390	2.499	0.064
1.460	8.057	1.336	7.217	7.914	3.615	9.981	9.198	5.292	1.224
0.432	8.645	8.774	0.249	8.081	7.461	4.416	0.652	4.002	4.644
0.679	2.800	5.523	3.049	2.968	7.225	6.730	4.199	9.614	9.229
4.263	1.074	7.286	5.599	8.291	5.200	9.214	8.272	4.398	4.506
9.496	4.830	3.150	8.270	5.079	1.231	5.731	9.494	1.883	9.732
4.138	2.562	2.532	9.661	5.611	5.500	6.886	2.341	9.699	6.500

A.2.10 Shekel's Foxholes

This 2nd ICEO version of the Shekel's foxholes function (Second ICEO 1997) also relies on the set of points listed in **A** and on the constants in **c**, but unlike the Modified Langerman function, this function uses all thirty points. Minima for both $D = 5$ and $D = 10$ are provided below. This function is hard for optimizers that tend to prematurely converge.

$$f(\mathbf{x}) = -\sum_{k=0}^{m-1} \frac{1}{\|\mathbf{x} - \mathbf{A}_k\|^2 - c_k}, \tag{A.15}$$

$$\mathbf{c} = (c_k), \quad \mathbf{A} = (a_{j,k}), \quad k = 0,1,...,m-1, \quad m = 30,$$

$$0 \le x_j \le 10, \quad j = 0,1,...,D-1, \quad D \le 10, \quad \varepsilon = 0.01,$$

$$f(\mathbf{x}^*) = \begin{cases} -10.4056 & \text{for } D = 5 \\ -10.2088 & \text{for } D = 10, \end{cases}$$

$$\mathbf{x}^* = \mathbf{A}_3 \quad \text{for } D = 5,\ 10.$$

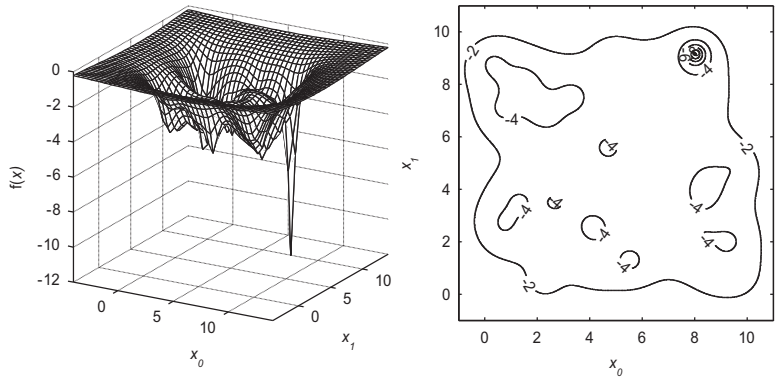

Fig. A.16. The Shekel's foxholes function

A.2.11 Odd Square

This 2nd ICEO function (Second ICEO 1997) resembles Salomon's function except that the ripples are rectangular, not circular. Because the Odd Square is symmetric about the solution, methods that search the vicinity of the population's mean vector will likely do well on this problem. In Eq. A.16, *d* is *D* times the square of the single, largest coordinate difference between the trial vector and the center point, **b**.

$$f(\mathbf{x}) = -\exp\left(\frac{-d}{2\pi}\right)\cdot\cos(\pi\cdot d)\cdot\left(1+\frac{0.02\cdot h}{d+0.01}\right), \tag{A.16}$$

$$d = D\cdot\max\left((x_j - b_j)^2\right), \quad h = \sum_{j=0}^{D-1}(x_j - b_j)^2,$$

$$-5\cdot\pi \le x_j \le 5\cdot\pi, \quad j = 0,1,...,D-1, \quad D \le 20, \quad \varepsilon = 0.01,$$

$$f(\mathbf{x}^*) = -1.14383, \quad \mathbf{x}^* = \text{many solutions near } \mathbf{b},$$

$$\mathbf{b} = [1, 1.3, 0.8, -0.4, -1.3, 1.6, -0.2, -0.6, 0.5, 1.4,$$
$$1, 1.3, 0.8, -0.4, -1.3, 1.6, -0.2, -0.6, 0.5, 1.4].$$

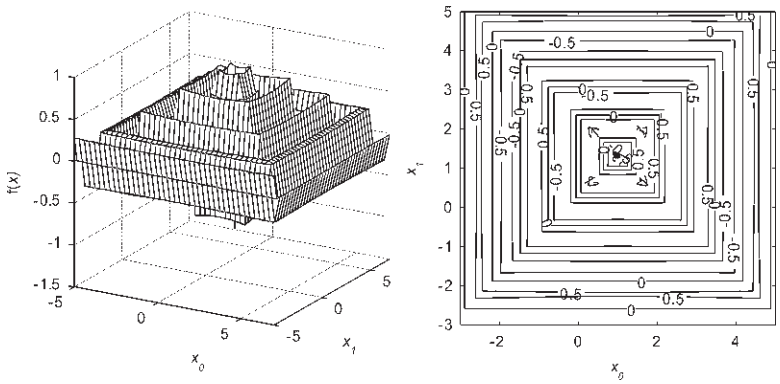

Fig. A.17. The Odd Square function

A.2.12 Katsuura

To be computed accurately, this function needs a floating-point format that supports more than 32 bits of precision when $m \ge 32$. The function "nint()" returns the nearest integer to the argument.

$$f(\mathbf{x}) = \prod_{j=0}^{D-1}\left(1+(j+1)\cdot\sum_{k=1}^{m}\text{nint}\left(2^k\cdot x_j\right)\cdot 2^{-k}\right), \quad k = 0,1,...,m = 32, \tag{A.17}$$

$$-1000 \le x_j \le 1000, \quad j = 0,1,...,D-1,$$

$$f(\mathbf{x}^*) = 1, \quad x_j = 0, \quad \varepsilon = 1.0\times10^{-6}.$$

A.3 Bound-Constrained Test Functions

A.3.1 Schwefel

This classic test function has a solution that lies on a coordinate system diagonal. In this version, the objective function is normalized by D so that $f(\mathbf{x}^*)$ is the same regardless of dimension. Success here can depend heavily on how bound constraints are handled. This function is separable.

$$f(\mathbf{x}) = -\frac{1}{D}\sum_{j=0}^{D-1} x_j \cdot \sin\left(\sqrt{|x_j|}\right), \tag{A.18}$$
$$-500 \le x_j \le 500, \quad j = 0,1,\ldots,D\text{-}1, \quad \varepsilon = 0.01,$$
$$f(\mathbf{x}^*) = -418.983, \quad x_j^* = 420.968746.$$

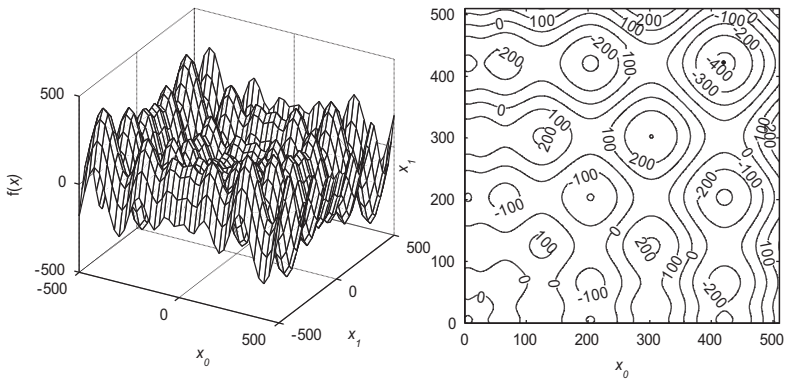

Fig. A.18. Schwefel's function

A.3.2 Epistatic Michalewicz

This 2nd ICEO function (Second ICEO 1997) also has a solution that lies near the limits of the allowed search space.

$$f(x)=\sum_{j=0}^{D-1}\sin\left(y_j\right)\cdot\left(\sin\left(\frac{(j+1)\cdot y_j^2}{\pi}\right)\right)^{2m},\quad m=10, \tag{A.19}$$

$$y_j=\begin{cases}x_j\cdot\cos\left(\dfrac{\pi}{6}\right)-x_{j+1}\cdot\sin\left(\dfrac{\pi}{6}\right)\ \text{if}\ (j+1)\bmod(2)=1\\ x_{j-1}\cdot\sin\left(\dfrac{\pi}{6}\right)+x_j\cdot\cos\left(\dfrac{\pi}{6}\right)\ \text{if}\ (j+1)\bmod(2)=0,\ \ j\neq D-1\\ y_{D-1}=x_{D-1}\ \text{if}\ \ j=D-1\end{cases}$$

$$0\le x_j\le\pi,\quad j=0,1,...,D-1,\quad D>1,$$

$$f\left(\mathbf{x}^*\right)=\begin{cases}-4.68766\ \text{for}\ D=5\\ -9.66015\ \text{for}\ \ D=10\end{cases}$$

$$\mathbf{x}^*=\begin{cases}[2.693170,0.258897,2.074365,1.022922,1.720470]\ \text{for}\ \ D=5\\ [2.693170,0.258897,2.074365,1.022922,2.275369,\\ \ 0.500115,2.137603,0.793609,2.818757,1.570796]\ \text{for}\ \ D=10\end{cases}$$

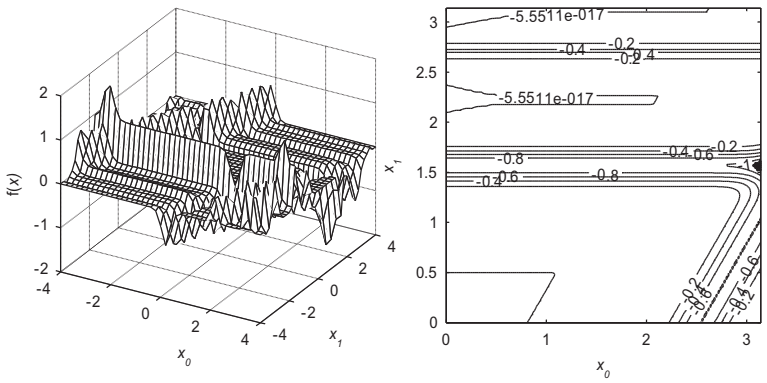

Fig. A.19. The epistatic Michalewicz function

A.3.3 Rana

This is one of the extended functions described in Whitley et al. (1996) in which a two-dimensional primitive function is evaluated with consecutive pairs of parameters, e.g., (0, 1), (1, 2), …, (D – 1, 0), so that the last term pairs the trial vector's first and last parameters (a "full-wrap" evaluation).

$$f(\mathbf{x}) = \sum_{j=0}^{D-1} x_j \cdot \sin(\alpha) \cdot \cos(\beta) + x_{(j+1) \bmod D} \cdot \cos(\alpha) \cdot \sin(\beta), \tag{A.20}$$

$$\alpha = \sqrt{|x_{j+1} + 1 - x_j|}, \qquad \beta = \sqrt{|x_{j+1} + 1 + x_j|},$$

$$-512 \le x_j \le 512, \qquad j = 0,1,...,D-1, \quad D > 1,$$

$$f(\mathbf{x}^*) = -511.708, \quad x_j^* = -512, \quad \varepsilon = 0.01.$$

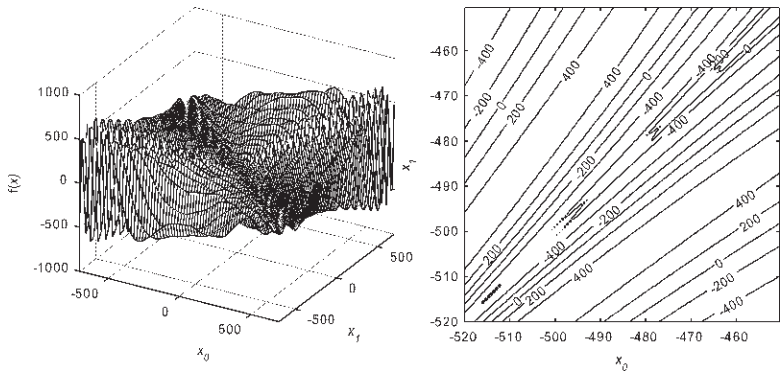

Fig. A.20. Rana's function

References

Michalewicz Z, Shoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4(1):1–32; the test problems are also available via the Internet at: http://www.lut.fi/~jlampine/testset.pdf

Second ICEO (1997) Code for 2nd ICEO test functions is available via the Internet at: http://iridia.ulb.ac.be/~aroli/ICEO/Functions/Functions.html

Yao X, Liu Y (1997) Fast evolution strategies. In: Angeline PJ, Reynolds RG, McDonnell JR, Eberhart R (eds) Evolutionary programming VI, Lecture notes in computer science 1213, Springer, Berlin, pp 151–161

Whitley D, Mathias K, Rana S, Dzubera J (1996) Evaluating evolutionary algorithms. Artificial Intelligence 85:1–32