Chapter 2

# Generalized Simulated Annealing

Yang Xiang, Sylvain Gubian and Florian Martin

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/66071

**Abstract**

Many problems in mathematics, statistics, finance, biology, pharmacology, physics, applied mathematics, economics, and chemistry involve the determination of the global minimum of multidimensional real-valued functions. Simulated annealing methods have been widely used for different global optimization problems. Multiple versions of simulated annealing have been developed, including classical simulated annealing (CSA), fast simulated annealing (FSA), and generalized simulated annealing (GSA). After revisiting the basic idea of GSA using Tsallis statistics, we implemented a modified GSA approach using the R package GenSA. This package was designed to solve complicated nonlinear objective functions with a large number of local minima. In this chapter, we provide a brief introduction to this R package and demonstrate its utility by solving non-convexoptimization problems in different fields: physics, environmental science, and finance. We performed a comprehensive comparison between GenSA and other widely used R packages, including rgenoud and DEoptim. GenSA is useful and can provide a solution that is comparable with or even better than that provided by other widely used R packages for optimization.

**Keywords:** classical simulated annealing (CSA), fast simulated annealing (FSA), generalized simulated annealing (GSA), GenSA

## 1. Introduction

Determining the global minimum of a multidimensional function is the focus of many problems in statistics, biology, physics, applied mathematics, economics, and chemistry [1–6]. Although there is a wide spectrum of problems, computing the global minimum remains a challenging task, because, for example, modern problem dimensionality is increasing.

The optimization of convex functions is usually reasonably conducted using standard optimization approaches, such as simplex optimization, the steepest descent method, and

the quasi-Newton method. These methods can also provide reasonable results for the study of simple non-convex functions with only a few dimensions and well-separated local minima.

Deterministic methods are usually faster than stochastic methods although they tend to be trapped into a local minimum. To overcome this particular issue, stochastic methods have been widely developed and can determine a good approximation of the global minimum with a modest computational cost. Among stochastic methods, genetic algorithms [7], evolution algorithms [8], simulated annealing (SA) [9], and taboo search [10–12] have been successfully applied.

Among popular approaches, genetic algorithms [7] mimic the process of natural DNA evolution. In this approach, a population of randomly generated solutions is generated. The solutions are encoded as strings and evolve over many iterations toward better solutions. In each generation, the fitness of each individual in the population is evaluated, and in the next generation, strings are generated by crossover, mutation, and selection, based on their fitness. Differential evolution belongs to such genetic algorithms.

Ant colony optimization (ACO) [13] is another set of stochastic optimization methods, which is inspired by ants wandering to find food for the colony. An ant starts wandering randomly while laying down pheromone trails that will influence other ants because they will be attracted (increase in probability) by the trail, and if they eventually locate food, will return and reinforce the trail. To avoid the algorithm converging to local minima, the pheromone trail is set to evaporate proportionally to the time it takes to traverse the trail to decrease its attractiveness. As a consequence, the pheromone density of short paths becomes higher than that of longer paths. The design of ACO perfectly matches graph-based optimization (e.g., traveling salesman problem), but it can be adapted to determine the global minimum of real-valued functions [14] by allowing local random moves in the neighborhood of the current states of the ant.

The SA algorithm was inspired by the annealing process that takes place in metallurgy, whereby annealing a molten metal causes it to achieve its global minimum in terms of thermodynamic energy (crystalline state) [9]. In the SA algorithm, the objective function is treated as the energy function of a molten metal, and one or more artificial temperatures are introduced and gradually cooled, which is analogous to the annealing technique, to attempt to achieve the global minimum. To escape from local minima, this artificial temperature (or set of temperatures) acts as a source of stochasticity. Following the metallurgy analogy, at the end of the process, the system is foreseen to reside inside the attractive basin of the global minimum (or in one of the global minima if more than one global minimum exists). In classical simulated annealing (CSA), the visiting distribution is a Gaussian function (a local search distribution) for each temperature. It has been observed that this distribution is not optimal for moving across the entire search space [5]. Generalized simulated annealing (GSA) was developed to overcome this issue by using a distorted Cauchy-Lorentz distribution.

For a more extensive review of stochastic optimization algorithms, see the review provided by Fouskakis and Draper [15].

The R language and environment for statistical computing will be the language of choice in this chapter because it enables a fast implementation of algorithms, access to a variety of statistical modeling packages, and easy-to-use plotting functionalities. These advantages make the use of R preferable in many situations to other programming languages, such as Java, C++, Fortran, and Pascal [16].

In this chapter, we elaborate further on the background and improvements of GSA and the use of the R package GenSA [17], which is an implementation of a modified GSA. We will also discuss the performance of GenSA and show that it outperforms the genetic algorithm (R package rgenoud) and differential evolution (R package DEoptim) in an extensive testbed comprising 134 testing functions based on the success rate and number of function calls. The core function of GenSA is written in C++ to ensure that the package runs as efficiently as possible. The utility of this R package and its use will be presented by way of several applications, such as the famous Thomson problem in physics, non-convex portfolio optimization in finance, and kinetic modeling of pesticide degradation in environmental science.

## 2. Method

As mentioned above, SA methods attempt to determine the global minimum of any objective function by simulating the annealing process of a molten metal. Given an objective function $f(x)$ with $x = (x_1, x_2, \ldots, x_n)^T$, we attempt to determine its global minimum using SA. The general procedure for SA is as follows:

1. Generate an initial state $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)^T$ randomly and obtain its function value $E^0 = f(x^0)$. An initial temperature $T^0$ is set. *imax* is set to be any big integer.
2. For step $i = 1$ to *imax*,
   - The temperature $T^i$ is decreased according to some cooling function.
   - Generate a new state $x^i = x^{i-1} + \Delta x$, where $\Delta x$ follows a predefined visiting distribution (e.g., Gaussian distribution). $E^i = f(x^i)$ and $\Delta E = E^i - E^{i-1}$.
   - Calculate the probability $p$ of $x^{i-1} \rightarrow x^i$.If $p < random(0,1)$, $x^i$ is set back to its previous state $x^{i-1}$ and $E^i$ is also set back to $E^{i-1}$.
3. Output the final state $x^{imax}$ and its function value $E^{imax}$.

We provide more details of SA methods as follows.

### 2.1. Classical simulated annealing (CSA)

According to the process of cooling and the visiting distribution, SA methods can be classified into several categories, amongwhich CSA [9], fast simulated annealing (FSA) [18], and the GSA [19] are the most common.

In CSA, proposed by Kirkpatrick et al., the visiting distribution is a Gaussian function, which is a local search distribution [5, 19]:

$$g(\Delta x) \propto \exp\left(-\frac{(\Delta x)^2}{T}\right), \tag{1}$$

where $\Delta x$ is the trial jump distance of variable $x$ and $T$ is an artificial temperature in the reduced unit. In a local search distribution, for example, a Gaussian distribution $\Delta x$ is always localized around zero. The jump is accepted if it is downhill of the energy/fitness/objective function. If the jump is uphill, it might be accepted according to an acceptance probability, which is computed using the Metropolis algorithm [20]:

$$p = \min\left[1,\ \exp\left(-\frac{\Delta E}{T}\right)\right]. \tag{2}$$

Geman and Geman [21] showed that for the classical case, a necessary and sufficient condition for having probability 1 of ending at the global minimum is that the temperature decreases logarithmically with the simulation time, which is impossible in practice because this would dramatically increase the computational time.

### 2.2. Fast simulated annealing (FSA)

In 1987, Szu and Hartley proposed a method called FSA [18], in which the Cauchy-Lorentz visiting distribution, that is, a semi-local search distribution, is introduced:

$$g(\Delta x) \propto \frac{T}{\left(T^2 + (\Delta x)^2\right)^{\frac{D+1}{2}}}, \tag{3}$$

where $D$ is the dimension of the variable space. In a semi-local search distribution, for example, the Cauchy-Lorentz distribution, the jumps $\Delta x$ are frequently local, but can occasionally be quite long. The temperature $T$ in FSA decreases with the inverse of the simulation time, and the acceptance algorithm is the Metropolis algorithm shown in Eq.(2).

### 2.3. Generalized simulated annealing (GSA)

#### *2.3.1. Introduction to GSA*

A generalization of classical statistical mechanics was proposed by Tsallis and Stariolo [19]. In the Tsallis formalism, a generalized statistic is built from generalized entropy:

$$s_q = k\frac{1-\sum p_i^q}{q-1}, \tag{4}$$

where $q$ is a real number, $i$ is the index of the energy spectrum, and $s_q$ tends to information entropy:

$$s = -k\sum p_i \ln p_i \tag{5}$$

when $q \to 1$. Maximizing the Tsallis entropy with the constraints

$$\begin{aligned} &\sum p_i = 1, \text{ and} \\ &\sum p_i^q \varepsilon_i = const, \end{aligned} \tag{6}$$

where $\varepsilon_i$ is the energy spectrum, and the generalized probability distribution is determined to be

$$p_i = \frac{[1-(1-q)\beta\varepsilon_i]^{\frac{1}{1-q}}}{z_q}, \tag{7}$$

where $z_q$ is the normalization constant that ensures that $p_i$ integrates to 1. This distribution pointwise converges to the Gibbs-Boltzmann distribution, where $q$ tends to 1.

The GSA algorithm [19] refers to the generalization of both CSA and FSA according to Tsallis statistics. It uses the Tsallis-Stariolo form of the Cauchy-Lorentz visiting distribution, whose shape is controlled by the parameter $q_v$:

$$g_{q_v}\left(\Delta x(t)\right) \propto \frac{[T_{q_v}(t)]^{-\frac{D}{3-q_v}}}{\left[1 + (q_v - 1)\frac{\left(\Delta x(t)\right)^2}{[T_{q_v}(t)]^{\frac{2}{3-q_v}}}\right]^{\frac{1}{q_v - 1} + \frac{D-1}{2}}}. \tag{8}$$

Please note that $q_v$ also controls the rate of cooling:

$$T_{q_v}(t) = \frac{2^{q_v - 1} - 1}{(1+t)^{q_v - 1} - 1} T_{q_v}(1), \tag{9}$$

where $T_{q_v}$ is the visiting temperature. In turn, a generalized Metropolis algorithm is used for the acceptance probability:

$$p_{q_a} = \min\left\{1,\ [1-(1-q_a)\beta\Delta E]^{\frac{1}{1-q_a}}\right\}, \tag{10}$$

where $\beta = 1/(KT_{q_a})$. The acceptance probability is controlled by the artificial temperature, $T_{q_a}$. When $q_v = 1$ and $q_a = 1$, then GSA is exactly CSA. Another special instance is given by $q_v = 2$ and $q_a = 1$ for which GSA is exactly FSA. Asymptotically, GSA has a similar behavior thanthe stochastic steepest descentas $T \to 0$. A faster cooling than CSA and FSA is achieved when $q_v > 2$.

It has been shown in a few instances that GSA is superior to FSA and CSA. Xiang et al. established that a pronounced reduction in the fluctuation of energy and a faster convergence

to the global minimum are achieved in the optimization of the Thomson problem and Nickel cluster structure [4–6]. Lemes et al. [22] observed a similar trend when optimizing the structure of a silicon cluster.

*2.3.2. Improvement of GSA*

A simple technique to accelerate convergence is as follows:

$$T_{q_a} = \frac{T_{q_v}}{t} \tag{11}$$

where $T_{q_a}$ is the acceptance temperature, $T_{q_v}$ is the visiting temperature, and $t$ is the number of time steps. Our testing shows that this simple technique accelerates convergence [6].

The performance of GSA can be further improved by combining GSA with a local search method, large-scale bound-constrained Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [23] for a smooth function, or Nelder-Mead [24] for a non-smooth function. A local search is performed at the end of each Markov chain for GSA.

## 3. Results

The GenSA R package has been developed and added to the toolkit for solving optimization problems in the Comprehensive R Archive Network (CRAN) R packages repository. The package GenSA has proven to be a useful tool for solving global optimization problems [17].

### 3.1. Usage

The GenSA R package provides a unique function called GenSA whose arguments were described in the associated help [25]:

**par**: Numeric vector. Initial values for the parameters to be optimized over. Default value is NULL, in which case, the initial values will be generated automatically.

**lower**: Numeric vector with a length of par. Lower bounds for the parameters to be optimized over.

**upper**: Numeric vector with a length of par. Upper bounds for the parameters to be optimized over.

**fn**: A function to be minimized, where the first argument is the vector of parameters over which minimization is to take place. It should return a scalar result. The function has to always return a numerical value; however, not applicable (NA) and infinity are supported.

...: Allows the user to pass additional arguments into fn.

**control**: A list of control parameters, discussed below. The control argument is a list that can be used to control the behavior of the algorithm. Some components of control are the following:

- **maxit**: Integer. Maximum number of iterations of the algorithm. Default value is 5000, which could be increased by the user for the optimization of a very complicated objective function.
- **threshold.stop**: Numeric. The program will stop when the objective function value is≤ threshold.stop. Default value is NULL.
- **smooth**: Logical. TRUE when the objective function is smooth, or differentiable almost everywhere, and FALSE otherwise. Default value is TRUE.
- **max.call**: Integer. Maximum number of calls of the objective function. Default value is 10,000,000.
- **max.time**: Numeric.Maximum running time in seconds.
- **temperature**: Numeric. Initial value for the temperature.
- **visiting.param**: Numeric.Parameter for the visiting distribution.
- **acceptance.param**: Numeric.Parameter for the acceptance distribution.
- **verbose**: Logical. TRUE means that messages from the algorithm are shown. Default value is FALSE.
- **trace.mat**: Logical. Default value is TRUE, which means that the trace matrix will be available in the returned value of the GenSA call.

The default values of the control components are set for a complicated optimization problem. For typical optimization problems with medium complexity, GenSA can determine a reasonable solution quickly; thus,using the variable **threshold.stop** to the expected function value is recommended to make GenSA stop at an earlier stage. A maximum run time can be also set by **max.time** argument or **max.call** argument for setting the maximum run time or number of calls, respectively.

### 3.2. Performance

An extensive performance comparison of currently available R packages for continuous global optimization problems has been published [26]. In this comparison, 48 benchmark functions were used to compare 18 R packages for continuous global optimization. Performance was measured in terms of the quality of the solutions determined and speed. The author concluded that GenSA and rgenoud are recommended in general for continuous global optimization [26]. Based on this conclusion, we set out to perform a more extensive performance test by including more benchmark functions and the additional algorithm DEoptim. The SciPy Python scientific toolkit provides an extensive set of 196 benchmark functions. Because these 196

benchmark functions are coded in Python, we had to convert the Python code to R code. As a result, a subset containing 134 functions was available for this test. One hundred runs using a random initial starting point were performed for every combination of the 134 benchmark functions and the aforementioned three methods. We used a local search method to further refine the best solution provided by Deoptim, because this technique provides a more accurate final result [17]. The default values of the parameters of every package were used in this comparison. A tolerance of 1e-8 was used to establish whether the algorithm determines the global minimum value.

The reliability of a method can be measured by the success rate%, which is defined as the number of successful runs over 100 runs. For each testing function, the number of function calls required to achieve the global minimum was recorded for every method, and we refer to this as the number of function calls. Please note that rgenoud required a longer time to complete 100 runs compared with GenSA and DEoptim. **Table 1** shows the success rate% and the average number of function calls (in parentheses).

The mean of the success rate% over all the benchmark functions is 92, 85, and 86% for GenSA, DEoptim, and rgenoud, respectively. Because the number of function calls changes dramatically, the median rather than the mean of the number of function calls is provided: 244.3 for GenSA, 1625.9 for Deoptim, and 1772.1 for rgenoud.

A heatmap of the success rate% for GenSA, DEoptim, and rgenoud is displayed in **Figure 1**. The color scaling from red to green represents the success rate% from 0 to 100. Clearly, GenSA has a larger green region (high success rate%) than DEoptim and rgenoud.

Both the success rate% and number of function calls show that GenSA performed better than DEoptim and rgenoud in the testbed composed of 134 benchmark functions.

### 3.3. Applications

#### *3.3.1. The Thomson problem in physics*

The physicist J.J. Thomson posed the famous Thomson problem after proposing his plum pudding atomic model, based on his knowledge of the existence of negatively charged electrons within neutrally charged atoms [27]. The objective of the Thomson problem is to determine the minimum electrostatic potential energy configuration of $N$ equal point charges constrained at the surface of a unit sphere that repel each other with a force given by Coulomb's law. The Thomson model has been widely studied in physics [28–31]. In the Thomson model, the energy function is

$$E = \frac{1}{2}\sum_{j\neq i}\frac{1}{|\vec{r}_i - \vec{r}_j|}. \tag{12}$$

The number of metastable structures (local minima) of the Thomson problem grows exponentially with $N$ [28]. The region containing the global minimum is often small compared with those of other minima [30]. The Thomson problem has been selected as a benchmark for global

| Function name | GenSA | DEoptim-LBFGS | rgenoud |
|---|---|---|---|
| AMGM | 100% (93.0) | 100% (62.6) | 100.0% (88.8) |
| Ackley01 | 100% (746.2) | 100% (1710.0) | 100.0% (1840.1) |
| Ackley02 | 100% (182.9) | 100% (1703.6) | 100.0% (2341.6) |
| Ackley03 | 100% (352.5) | 100% (1420.2) | 100.0% (1860.9) |
| Adjiman | 100% (33.3) | 100% (1133.9) | 100.0% (1677.5) |
| Alpine01 | 100% (756.0) | 70% (2756.8) | 95.0% (46852.2) |
| Alpine02 | 100% (56.4) | 100% (913.6) | 100.0% (1688.0) |
| BartelsConn | 100% (263.1) | 100% (1539.8) | 100.0% (2343.5) |
| Beale | 100% (145.6) | 100% (1311.8) | 100.0% (1711.0) |
| BiggsExp02 | 100% (85.6) | 100% (763.3) | 100.0% (1710.5) |
| BiggsExp03 | 100% (190.7) | 100% (2614.4) | 100.0% (1975.9) |
| BiggsExp04 | 100% (3498.8) | 88% (8182.5) | 100.0% (4234.5) |
| BiggsExp05 | 98% (40117.8) | 14% (10864.2) | 17.0% (13871.5) |
| Bird | 100% (112.3) | 100% (1777.3) | 100.0% (1702.9) |
| Bohachevsky1 | 100% (875.1) | 100% (1107.5) | 100.0% (2673.1) |
| Bohachevsky2 | 100% (1372.9) | 100% (1155.2) | 76.0% (2554.5) |
| Bohachevsky3 | 100% (623.4) | 100% (1342.4) | 96.0% (2596.9) |
| BoxBetts | 100% (129.2) | 100% (1866.3) | 100.0% (2018.8) |
| Branin01 | 100% (42.2) | 100% (1747.6) | 100.0% (1694.9) |
| Branin02 | 100% (1495.7) | 28% (2830.9) | 96.0% (1752.3) |
| Brent | 100% (11.0) | 100% (987.5) | 100.0% (1687.9) |
| Bukin02 | 100% (39.9) | 100% (1477.4) | 100.0% (1679.7) |
| Bukin04 | 100% (217.9) | 100% (1029.4) | 100.0% (1744.2) |
| Bukin06 | 0% (NA) | 0% (NA) | 0.0% (NA) |
| CarromTable | 100% (119.5) | 100% (2040.9) | 100.0% (1729.6) |
| Chichinadze | 100% (517.4) | 100% (1063.9) | 92.0% (2219.8) |
| Colville | 100% (4515.8) | 100% (8230.9) | 100.0% (2996.1) |
| CosineMixture | 100% (22.0) | 100% (3875.3) | 100.0% (1670.6) |
| CrossInTray | 100% (70.8) | 100% (1512.8) | 100.0% (1772.1) |
| CrossLegTable | 0% (NA) | 0% (NA) | 2.0% (51829.0) |
| CrownedCross | 0% (NA) | 0% (NA) | 2.0% (16045.0) |
| Cube | 100% (1717.2) | 100% (2030.3) | 52.0% (21649.8) |
| DeVilliersGlasser01 | 100% (2343.8) | 0% (NA) | 1.0% (43919.0) |
| DeVilliersGlasser02 | 2% (173501.0) | 0% (NA) | 0.0% (NA) |
| Deb01 | 100% (57.1) | 100% (4000.0) | 100.0% (1700.8) |
| Deb03 | 100% (78.8) | 100% (4028.9) | 100.0% (1708.1) |

| Function name | GenSA | DEoptim-LBFGS | rgenoud |
|---|---|---|---|
| Decanomial | 100% (1519.3) | 100% (741.4) | 100.0% (2050.8) |
| DeckkersAarts | 100% (74.6) | 100% (1525.0) | 100.0% (1988.7) |
| Dolan | 100% (2504.7) | 1% (10293.0) | 82.0% (25067.4) |
| DropWave | 100% (3768.7) | 85% (4009.8) | 83.0% (2973.3) |
| Easom | 97% (5077.5) | 100% (1343.0) | 100.0% (1875.7) |
| EggCrate | 100% (122.9) | 100% (912.2) | 100.0% (1697.2) |
| ElAttarVidyasagarDutta | 100% (675.7) | 93% (1625.9) | 100.0% (2115.7) |
| Exp2 | 100% (85.3) | 100% (781.2) | 100.0% (1707.7) |
| Exponential | 100% (20.6) | 100% (580.3) | 100.0% (1682.5) |
| FreudensteinRoth | 100% (395.4) | 83% (2620.7) | 100.0% (1700.9) |
| Gear | 100% (2225.8) | 100% (1118.0) | 93.0% (1609.6) |
| Giunta | 100% (39.6) | 100% (592.3) | 100.0% (1686.6) |
| GoldsteinPrice | 100% (158.7) | 100% (1023.0) | 100.0% (1703.5) |
| Gulf | 100% (1739.0) | 100% (4076.4) | 1.0% (1810.0) |
| Hansen | 100% (149.7) | 100% (104.0) | 100.0% (132.0) |
| HimmelBlau | 100% (53.7) | 100% (2384.8) | 100.0% (1698.7) |
| HolderTable | 100% (138.0) | 100% (2010.3) | 100.0% (1701.7) |
| Hosaki | 100% (49.8) | 100% (583.2) | 100.0% (1699.5) |
| Infinity | 100% (225.8) | 100% (113.4) | 100.0% (492.0) |
| JennrichSampson | 0% (NA) | 0% (NA) | 0.0% (NA) |
| Keane | 100% (21.4) | 100% (679.1) | 100.0% (629.5) |
| Leon | 100% (128.0) | 100% (1338.2) | 35.0% (2156.5) |
| Levy05 | 100% (152.8) | 100% (144.7) | 100.0% (224.6) |
| Levy13 | 100% (867.3) | 100% (1138.5) | 100.0% (1771.7) |
| Matyas | 100% (33.8) | 100% (967.7) | 100.0% (1702.4) |
| McCormick | 100% (42.2) | 100% (747.8) | 100.0% (1705.4) |
| Michalewicz | 100% (97.3) | 100% (69.0) | 100.0% (157.3) |
| MieleCantrell | 100% (2935.1) | 100% (1942.7) | 100.0% (3438.9) |
| Mishra03 | 81% (138334.7) | 0% (NA) | 24.0% (5745.0) |
| Mishra04 | 0% (NA) | 0% (NA) | 13.0% (1833.2) |
| Mishra05 | 100% (1289.1) | 77% (1396.9) | 100.0% (1680.0) |
| Mishra06 | 0% (NA) | 0% (NA) | 0.0% (NA) |
| Mishra07 | 100% (38.9) | 100% (3055.9) | 100.0% (1679.7) |
| Mishra08 | 100% (1519.3) | 100% (741.4) | 100.0% (1900.8) |
| Mishra09 | 100% (80.0) | 85% (4832.0) | 99.0% (12356.0) |

http://dx.doi.org/10.5772/66071

| Function name | GenSA | DEoptim-LBFGS | rgenoud |
|---|---|---|---|
| Mishra11 | 100% (30.9) | 100% (3152.2) | 100.0% (1613.1) |
| MultiModal | 100% (134.5) | 100% (481.8) | 100.0% (1691.6) |
| NewFunction01 | 1% (243864.0) | 0% (NA) | 27.0% (9823.3) |
| NewFunction02 | 0% (NA) | 0% (NA) | 33.0% (12260.0) |
| Parsopoulos | 100% (29.9) | 100% (3267.7) | 100.0% (1683.9) |
| Paviani | 100% (423.3) | 100% (18764.2) | 100.0% (2168.7) |
| PenHolder | 100% (94.3) | 100% (1391.3) | 100.0% (1701.6) |
| Plateau | 100% (35.5) | 100% (22.4) | 100.0% (27.6) |
| Powell | 100% (692.4) | 100% (6501.1) | 100.0% (2052.4) |
| Price01 | 100% (27.4) | 100% (2107.0) | 100.0% (1686.6) |
| Price02 | 100% (1242.4) | 92% (4031.3) | 85.0% (2087.7) |
| Price03 | 100% (3840.5) | 64% (2574.6) | 56.0% (1855.1) |
| Price04 | 100% (440.1) | 100% (1075.4) | 94.0% (6413.9) |
| Qing | 0% (NA) | 0% (NA) | 0.0% (NA) |
| Quadratic | 100% (26.0) | 100% (872.2) | 100.0% (1695.8) |
| Quintic | 100% (928.2) | 76% (2694.1) | 36.0% (42496.5) |
| Rastrigin | 100% (482.4) | 98% (3753.6) | 100.0% (1840.2) |
| Ripple01 | 100% (5742.4) | 71% (4053.3) | 99.0% (4758.6) |
| Ripple25 | 100% (414.7) | 99% (3165.8) | 100.0% (1771.5) |
| RosenbrockModified | 100% (1343.5) | 24% (3625.6) | 95.0% (2329.0) |
| RotatedEllipse01 | 100% (26.0) | 100% (1345.8) | 100.0% (1699.9) |
| RotatedEllipse02 | 100% (28.0) | 100% (1218.1) | 100.0% (1700.8) |
| Salomon | 95% (9790.3) | 86% (4050.4) | 21.0% (3143.1) |
| Schaffer01 | 99% (7105.6) | 92% (3092.8) | 59.0% (4520.3) |
| Schaffer02 | 100% (2078.9) | 100% (2125.2) | 100.0% (2677.8) |
| Schaffer03 | 100% (2786.3) | 91% (4096.2) | 99.0% (3236.3) |
| Schaffer04 | 100% (2920.9) | 98% (4063.5) | 86.0% (6505.1) |
| Schwefel01 | 100% (131.2) | 100% (651.8) | 100.0% (1734.5) |
| Schwefel04 | 100% (63.9) | 100% (842.8) | 100.0% (1698.8) |
| Schwefel06 | 100% (3536.0) | 100% (2302.3) | 100.0% (1790.4) |
| Schwefel20 | 100% (2079.8) | 100% (1670.2) | 100.0% (1779.1) |
| Schwefel21 | 100% (2158.6) | 100% (1853.2) | 100.0% (1774.8) |
| Schwefel22 | 100% (2772.0) | 100% (1678.6) | 100.0% (1775.0) |
| Schwefel26 | 100% (131.3) | 100% (1648.2) | 100.0% (1687.5) |
| Schwefel36 | 100% (361.3) | 100% (1298.7) | 100.0% (1930.6) |

| Function name | GenSA | DEoptim-LBFGS | rgenoud |
|---|---|---|---|
| SixHumpCamel | 100% (83.3) | 100% (909.2) | 100.0% (1695.7) |
| Sodp | 100% (64.3) | 100% (477.7) | 100.0% (1697.5) |
| Sphere | 100% (17.4) | 100% (730.2) | 100.0% (1683.2) |
| Step | 100% (471.0) | 100% (219.3) | 100.0% (1131.5) |
| Step2 | 100% (433.4) | 100% (222.0) | 100.0% (1177.1) |
| StyblinskiTang | 100% (94.7) | 100% (861.4) | 100.0% (1824.2) |
| TestTubeHolder | 100% (839.0) | 98% (3957.5) | 100.0% (2088.5) |
| ThreeHumpCamel | 100% (86.3) | 100% (817.5) | 100.0% (1699.0) |
| Treccani | 100% (49.1) | 100% (1015.2) | 100.0% (1696.4) |
| Trefethen | 64% (20972.8) | 0% (NA) | 46.0% (29439.5) |
| Trigonometric02 | 100% (1766.2) | 100% (1319.6) | 99.0% (4809.6) |
| Ursem01 | 100% (47.7) | 100% (682.1) | 100.0% (1753.8) |
| Ursem03 | 100% (584.0) | 100% (1645.7) | 100.0% (1777.7) |
| Ursem04 | 100% (210.2) | 100% (1372.3) | 100.0% (1847.3) |
| UrsemWaves | 100% (2565.0) | 26% (4025.8) | 50.0% (1674.8) |
| VenterSobiezcczanskiSobieski | 100% (698.3) | 100% (1748.0) | 100.0% (2004.8) |
| Vincent | 100% (41.4) | 100% (3013.4) | 100.0% (1703.5) |
| Wavy | 100% (535.1) | 100% (3110.8) | 100.0% (1745.0) |
| WayburnSeader01 | 100% (156.1) | 96% (2258.3) | 93.0% (16665.5) |
| WayburnSeader02 | 100% (262.8) | 100% (1772.3) | 100.0% (1826.0) |
| Wolfe | 100% (50.3) | 100% (3252.9) | 100.0% (1720.7) |
| XinSheYang02 | 100% (1048.0) | 87% (2284.5) | 100.0% (1768.5) |
| XinSheYang04 | 100% (457.0) | 88% (3329.7) | 100.0% (1777.5) |
| Xor | 100% (26204.2) | 48% (18245.1) | 100.0% (1852.9) |
| YaoLiu09 | 100% (482.3) | 98% (3753.6) | 100.0% (1824.3) |
| Zacharov | 100% (59.7) | 100% (944.3) | 100.0% (1696.6) |
| Zettl | 100% (123.4) | 100% (846.8) | 100.0% (1712.5) |
| Zirilli | 100% (131.0) | 99% (702.3) | 100.0% (1695.1) |

**Table 1.** The success rate% and average number of function calls (in parentheses) for GenSA, DEoptim, and rgenoud.

optimization algorithms in a number of previous studies [4, 5, 28, 32]. The Thomson problem has been solved by both deterministic methods, including steepest descent [28], and stochastic methods, including (but not limited to) constrained global optimization (CGO) [29], the GSA algorithm [4, 5], genetic algorithms [30], and the Monte Carlo method [31, 33]. Typically, deterministic methods with multiple starts can provide a good solution when there are fewer point charges, whereas stochastic methods have to be used when $N$ is large.

http://dx.doi.org/10.5772/66071

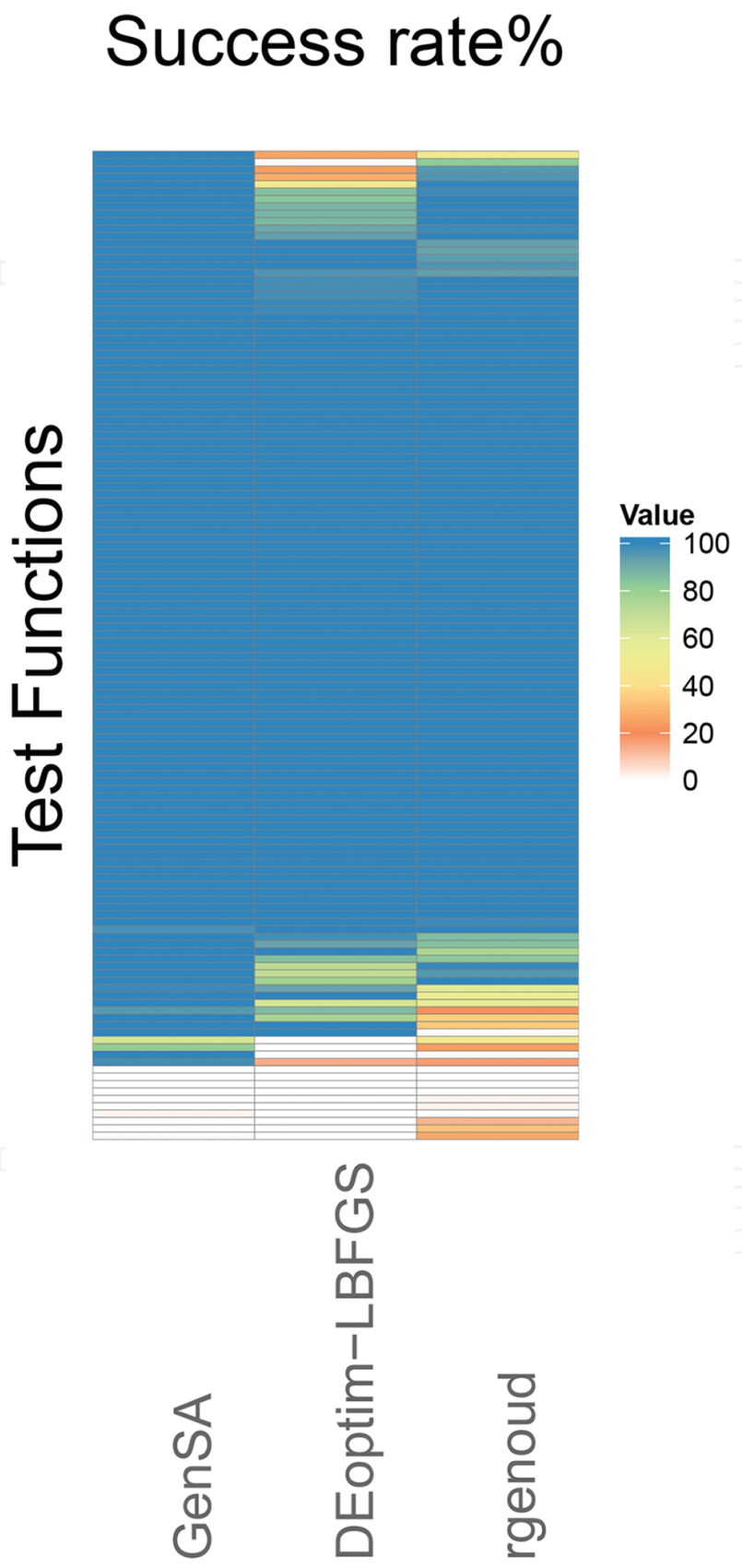


**Figure 1.** Heat map of the success rate% for GenSA, DEoptim, and rgenoud.

We can define an R function for the Thomson problem as follows:

```
>Thomson.fn<- function(x){
fn.call<<- fn.call + 1
x <- matrix(x, ncol = 2)
y <- t(apply(x, 1, function(z){
c(sin(z[1]) * cos(z[2]),
sin(z[1]) * sin(z[2]), cos(z[1]))}))
n <- nrow(x)
tmp<- matrix(NA, nrow = n, ncol = n)
index<- cbind(as.vector(row(tmp)), as.vector(col(tmp)))
index<- index[index[, 1] < index[, 2],, drop=F]
rdist<- apply(index, 1, function(z){
tmp<- 1/sqrt(sum((y[z[1],] - y[z[2],])^2))
})
res<- sum(rdist)
return(res)
}
```

In this example, we chose six point charges because our purpose is only to show how to use our package GenSA. The global energy minimum of six equal point charges on a unit sphere is 9.98528137 [28].

We applied GenSA with default settings to the Thomson problem. As the number of point charges is small, GenSA can determine the global minimum easily. We set the maximum number of function calls allowed, max.call, to 600:

```
>n.particles<- 6 # regular octahedron with global minimum
9.98528137
>lower.T<- rep(0, 2 * n.particles)
>upper.T<- c(rep(pi, n.particles), rep(2 * pi, n.particles))
>require(GenSA)
>options(digits = 9)
>set.seed(1234)
>fn.call<<- 0
>out.GenSA<- GenSA(par = NULL, lower = lower.T, upper = upper.T,
fn = Thomson.fn, control = list(max.call=600))
>print(out.GenSA[c("value", "counts")])
$value
[1] 9.98528137
$counts
[1] 600
>cat("GenSA call function", fn.call, "times.\n")
GenSA call function 600 times.
```

The global minimum 9.98528137 for six point charges is determined within 600 function calls.

http://dx.doi.org/10.5772/66071

*3.3.2. Kinetic modeling of pesticide degradation*

Various types of pesticides have been widely used in modern agriculture. It is important to calculate the concentration of a pesticide in groundwater and surface water. We will show how GenSA can be used to fit a degradation model for a parent compound with one transformation product. All the data and models are from the R packages mkin [34] and FOCUS (2006) [35].

After loading the library, we obtain the data (FOCUS Example Dataset D). The observed concentrations are in the column named "value" at the time specified in column "time" for the two observed variables named "parent" and "m1."

```
>require(mkin)
>require(GenSA)
>require(deSolve)
>options(digits = 9)
>set.seed(1234)
>str(FOCUS_2006_D)
'data.frame':44 obs. of3 variables:
$ name : Factor w/2 levels "m1","parent": 2 2 2 2 2 2 2 2 2 2 ...
$ time :num0 0 1 1 3 3 7 7 14 14 ...
$ value: num99.5 102 93.5 92.5 63.2 ...
```

The measured concentration of parent and m1 change with time is displayed in the lower panel of **Figure 2**.

According to the kinetic model displayed in the upper panel of **Figure 2**, we define the derivative function as follows:

```
>df<- function(t, y, parameters){
+d_parent<- -parameters["k_parent_sink"] * y["parent"] -
+parameters["k_parent_m1"] * y["parent"]
+d_m1 <- parameters["k_parent_m1"] * y["parent"] -
+parameters["k_m1_sink"] * y["m1"]
+return(list(c(d_parent, d_m1)))
+}
```

There is one initial concentration, parent_0, and three kinetic parameters, k_parent_m1, k_parent_sink, and k_m1_sink, whose values need to be fitted out by fitting the concentration curves. The initial concentration of m1, m1_0, is always zero. We define the objective function, fn, as the sum of the squares of residuals (deviations predicted from observed values for both the parent and m1):

```
>fn<- function(x,
+names_x = c("parent_0", "k_parent_sink", "k_parent_m1",
 "k_m1_sink"),
+names_y = c("parent", "m1"),
+names_parameters = c("k_parent_sink", "k_parent_m1",
 "k_m1_sink"),
```

```
+tspan = if (!is.null(dat.fitting))
+sort(unique(dat.fitting[["time"]])) else NULL,
+df, dat.fitting = NULL){
+m1_0 = 0
+names(x) <- names_x
+y0 <- c(x["parent_0"], m1_0)
+names(y0) <- names_y
+parameters<- x[c("k_parent_sink", "k_parent_m1", "k_m1_sink")]
+stopifnot(!is.null(tspan))
+out<- ode(y0, tspan, df, parameters)
+if (is.null(dat.fitting)){
+rss<- out
+} else{
+rss<- sum(sapply(c("parent", "m1"), function(nm){
+o.time<- as.character(dat.fitting[dat.fitting$name == nm,
"time"])
+sum((out[, nm][match(o.time, out[, "time"])] -
+dat.fitting[ dat.fitting$name == nm, "value"])^2,
na.rm = TRUE)
+}))
+}
+return(rss)
+}
```

Then, we use GenSA to estimate the four parameters. As the model is not complicated, we limit the running time of GenSA to 5seconds by setting max.time=5:

```
>res<- GenSA(fn = fn, lower = c(90, rep(0.001, 3)),
+upper = c(110, rep(0.1, 3)),
+control = list(max.time = 5), df = df,
+dat.fitting = FOCUS_2006_D)
>names(res$par) <- c("parent_0", "k_parent_sink", "k_parent_m1",
"k_m1_sink")
>print(round(res$par, digits = 6))
parent_0 k_parent_sinkk_parent_m1k_m1_sink
99.5984910.0479200.0507780.005261
```

GenSA successfully determines the correct value of the initial concentration of the parent and the three kinetic parameters. The fitting curves for the parent and m1 are displayed in the lower panel of **Figure 2**.

*3.3.3. Finance: risk allocation portfolios*

Portfolio selection problems were addressed by mean-risk models in the 1950s. The most popular measures of downside risk are the value-at-risk (VaR) and conditional VaR(CVaR). Portfolio weights for which the portfolio has the lowest CVaR and each investment can

http://dx.doi.org/10.5772/66071

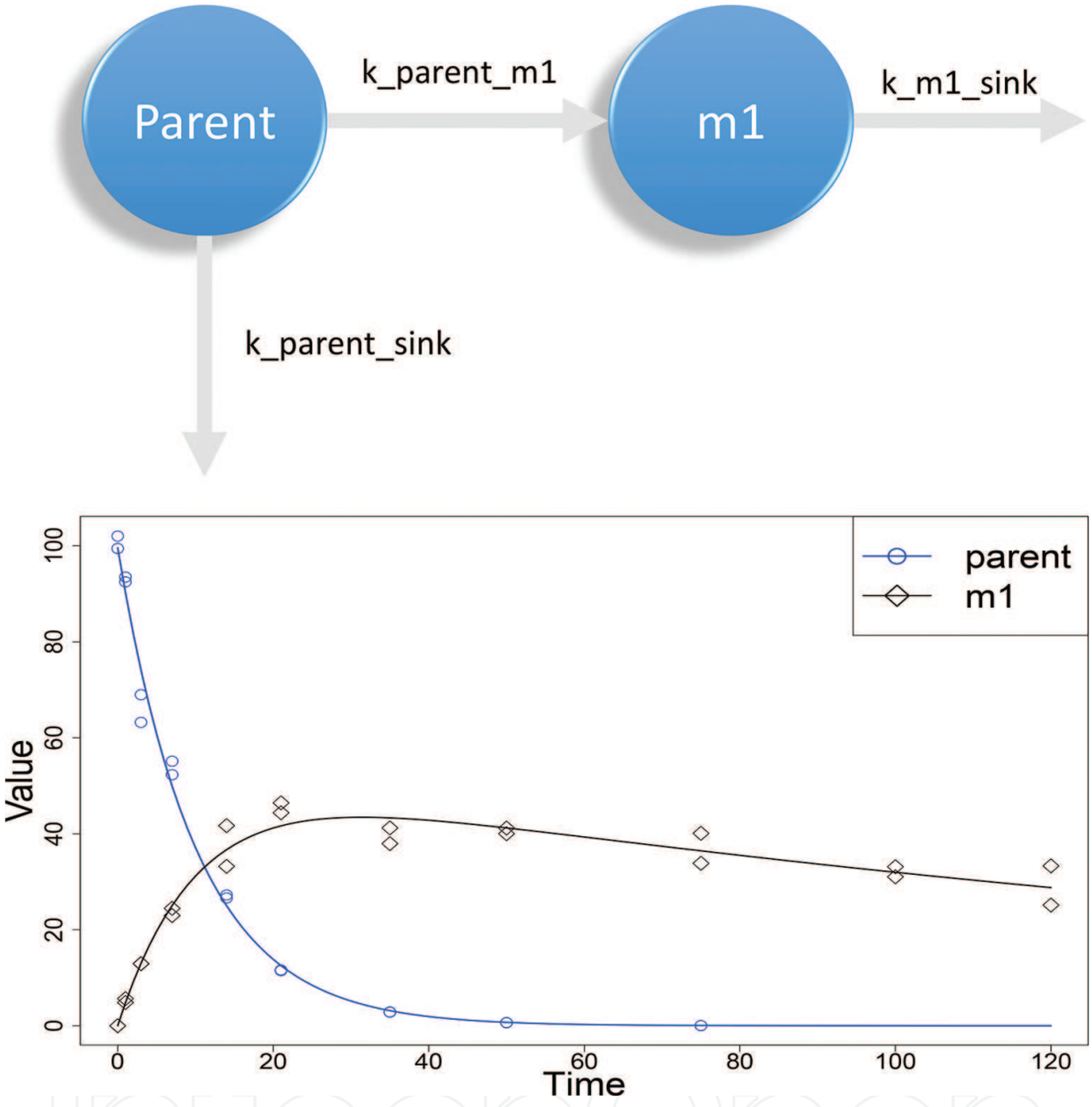


**Figure 2.** Kinetic modeling of pesticide degradation. Upper panel: illustration of pesticide degradation model. Lower panel: experimental concentration data and fitting curves for parent and m1.

contribute at most 22.5% to the total portfolio CVaR risk were estimated using differential evolution algorithms in Mullen et al. [16] and Ardia et al. [36]. The code for the objective function in portfolio optimization is rewritten below from Ardia et al. [36]:

```
>library("quantmod")
> tickers <- c("GE", "IBM", "JPM", "MSFT", "WMT")
>getSymbols(tickers, from = "2000-12-01", to = "2010-12-31")
[1] "GE" "IBM" "JPM" "MSFT" "WMT"
> P <- NULL
>for(ticker in tickers) {
```

```
+tmp<- Cl(to.monthly(eval(parse(text = ticker))))
+P <- cbind(P, tmp)
+}
>colnames(P) <- tickers
> R <- diff(log(P))
> R <- R[-1,]
> mu <- colMeans(R)
> sigma <- cov(R)
>library("PerformanceAnalytics")
>pContribCVaR<- ES(weights = rep(0.2, 5),
+method = "gaussian", portfolio_method = "component",
+mu = mu, sigma = sigma)$pct_contrib_ES
>obj<- function(w) {
+if (sum(w) == 0) { w <- w + 1e-2 }
+w <- w/sum(w)
+CVaR<- ES(weights = w,
+method = "gaussian", portfolio_method = "component",
+mu = mu, sigma = sigma)
+tmp1 <- CVaR$ES
+tmp2 <- max(CVaR$pct_contrib_ES - 0.225, 0)
+out <- tmp1 + 1e3 * tmp2
+return(out)
+}
```

GenSA can be used to determine the global optimum of this function using a bounded search domain from 0 to 1 values for the five parameters to be optimized:

```
>library(GenSA)
>lb<- rep(0, 5) # lower bounds, minimum values for all 5 parameters
are 0
>ub<- rep(1, 5) # upper bounds, maximum values for all 5 parameters
are 1
> out1.GenSA <- GenSA(fn = obj, lower = lb, upper = ub)
```

For non-differentiable objective functions, the smooth parameter in the control argument can be set to FALSE, which means that the Nelder-Mead method is used in the local search:

```
> out2.GenSA <- GenSA(fn=obj, lower=rep(0, 5), upper=rep(1, 5),
+control=list(smooth=FALSE, max.call=3000))
The max.call parameter is set to 3000 to make the algorithm stop
earlier:
> out2.GenSA$value
[1] 0.1141484884
> out2.GenSA$counts
[1] 3000
>cat("GenSA call functions", fn.call.GenSA, "times.\n")
GenSA call functions 3000 times.
>wstar.GenSA<- out2.GenSA$par
>wstar.GenSA<- wstar.GenSA/sum(wstar.GenSA)
```

```
>rbind(tickers, round(100 * wstar.GenSA, 2))
[,1] [,2] [,3] [,4] [,5]
tickers "GE" "IBM" "JPM" "MSFT" "WMT"
"18.92" "21.23" "8.33" "15.92" "35.6"
>100 * (sum(wstar.GenSA * mu) - mean(mu))
[1] 0.03790568876
```

GenSA determined a minimum of 0.1141484884 within 3000 function calls.

## 4. Discussion and conclusions

The discrete optimization problem, in particular, the feature selection problem, exists extensively. GSA can also be used for the discrete problem. Please refer to [37] for details.

GSA is a powerful method for the non-convex global optimization problem. We developed an R package GenSA based on Tsallis statistics and GSA. In an extensive performance testbed composed of 134 benchmark functions, GenSA provided a higher average success rate% and a smaller median of the number of function calls compared with two widely recognized R packages: DEoptim and rgenoud. GenSA is useful and can provide a solution that is comparable with or even better than that provided by other widely used R packages for optimization.

R is very good for program prototype. When there is a need for heavy computation, other computational languages, such as C/C++, Fortran, Java, and Python,are recommended. Considering both speed and usability, aPython version of GSA, PyGenSA, is being developed and will be released within the SciPy scientific toolkitat the end of 2016.

## Acknowledgements

This work would not have been possible without the R open-source software project. We would like to thank our colleague Julia Hoeng for inspirational discussions and support. This study was funded by Philip Morris International.

## Author details

Yang Xiang†*, Sylvain Gubian† and Florian Martin

*Address all correspondence to: yang.xiang@pmi.com

Philip Morris International R&D, Philip Morris Products S.A., Neuchâtel, Switzerland

† These authors contributed equally to this work

## References

[1] Agostini FP, Soares-Pinto DD, Moret MA, Osthoff C, Pascutti PG. Generalized simulated annealing applied to protein folding studies. Journal of Computational Chemistry. 2006 Aug 1;27(11):1142–55.

[2] Serra P, Stanton AF, Kais S, Bleil RE. Comparison study of pivot methods for global optimization. The Journal of Chemical Physics. 1997 May 1;106(17):7170–7.

[3] De Guire V, Caron M, Scott N, Ménard C, Gaumont-Leclerc MF, Chartrand P, Major F, Ferbeyre G. Designing small multiple-target artificial RNAs. Nucleic Acids Research. 2010 Jul 1;38(13):e140.

[4] Xiang Y, Sun DY, Fan W, Gong XG. Generalized simulated annealing algorithm and its application to the Thomson model. Physics Letters A. 1997 Aug 25;233(3):216–20.

[5] Xiang Y, Gong XG. Efficiency of generalized simulated annealing. Physical Review E. 2000 Sep 1;62(3):4473.

[6] Xiang Y, Sun DY, Gong XG. Generalized simulated annealing studies on structures and properties of Ni n (n =2-55) clusters. The Journal of Physical Chemistry A. 2000 Mar 30;104(12):2746–51.

[7] Holland JH. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press, 839 Greene Street, Ann Arbor, MI 48104-3209; 1975.

[8] Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. 1997 Dec 1;11(4):341–59.

[9] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science. 1983;220(4598):671–80.

[10] Cvijovic D, Klinowski J. Taboo search: an approach to the multiple minima problem. Science. 1995 Feb 3;267(5198):664.

[11] Cvijović D, Klinowski J. Taboo search: An approach to the multiple-minima problem for continuous functions. In Handbook of Global Optimization 2002 (pp. 387-406). Springer Publishing Company, 11 West 42nd Street, 15th Floor, New York, NY 10036, US.

[12] Glover F, Taillard E. A user's guide to tabu search. Annals of Operations Research. 1993 Mar 1;41(1):1–28.

[13] Dorigo M. Optimization, learning and natural algorithms. Ph. D. Thesis, Politecnico di Milano, Italy. 1992.

[14] Toksari MD. Ant colony optimization for finding the global minimum. Applied Mathematics and Computation. 2006 May 1;176(1):308–16.

[15] Fouskakis D, Draper D. Stochastic optimization: a review. International Statistical Review. 2002 Dec 1;70(3):315–49.

[16] Mullen KM, Ardia D, Gil DL, Windover D, Cline J. DEoptim: An R package for global optimization by differential evolution. Journal of Statistical Software. 2011;40(6):1–26.

[17] Xiang Y, Gubian S, Suomela B, Hoeng J. Generalized simulated annealing for global optimization: the GenSA Package. R Journal. 2013 Jun 1;5(1):13–28.

[18] Szu H, Hartley R. Fast simulated annealing. Physics Letters A. 1987 Jun 8;122(3–4):157–62.

[19] Tsallis C, Stariolo DA. Generalized simulated annealing. Physica A: Statistical Mechanics and its Applications. 1996 Nov 15;233(1):395–406.

[20] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. The Journal of Chemical Physics. 1953 Jun 1;21 (6):1087–92.

[21] Geman S, Geman D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Transactions on pattern analysis and machine intelligence. 1984 Jun 6(6):721–41.

[22] Lemes MR, Zacharias CR, Dal Pino A. Generalized simulated annealing: Application to silicon clusters. Physical Review B. 1997 Oct 15;56(15):9279.

[23] Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing. 1995 Sep;16(5):1190–208.

[24] Nash JC. Compact numerical methods for computers: linear algebra and function minimisation. Boca Raton, FL: CRC Press; 1990.

[25] Gubian S, Xiang Y, Suomela B, Hoeng J. Package GenSA. Version 1.1.6. 2016. Available from: https://cran.r-project.org/web/packages/GenSA/GenSA.pdf.

[26] Mullen KM. Continuous global optimization in r. Journal of Statistical Software. 2014 Sep 28;60(6):1–45.

[27] Thomson JJ. XXIV. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. 1904 Mar 1;7(39):237–65.

[28] Erber T, Hockney GM. Equilibrium configurations of N equal charges on a sphere. Journal of Physics A: Mathematical and General. 1991 Dec 7;24(23):L1369.

[29] Altschuler EL, Williams TJ, Ratner ER, Dowla F, Wooten F. Method of constrained global optimization. Physical Review Letters. 1994 Apr 25;72(17):2671.

[30] Morris JR, Deaven DM, Ho KM. Genetic-algorithm energy minimization for point charges on a sphere. Physical Review B. 1996 Jan 15;53(4):R1740.

[31] Wales DJ, Ulker S. Structure and dynamics of spherical crystals characterized for the Thomson problem. Physical Review B. 2006 Dec 27;74(21):212101.

[32] Altschuler EL, Pérez–Garrido A. Defect-free global minima in Thomson's problem of charges on a sphere. Physical Review E. 2006 Mar 6;73(3):036108.

[33] Wales DJ, McKay H, Altschuler EL. Defect motifs for spherical topologies. Physical Review B. 2009 Jun 22;79(22):224115.

[34] Ranke J. mkin: Routines for fitting kinetic models with one or more state variables to chemical degradation data, 2013b. URL http://CRAN.R-project.org/package=mkin.

[35] Boesten JJ, Aden K, Beigel CY, Beulke S, Dust M, Dyson JS, Fomsgaard IS, Jones RL, Karlsson S, Van der Linden AM, Richter O. Guidance document on estimating persistence and degradation kinetics from environmental fate studies on pesticides in EU registration. Report of the FOCUS Work Group on Degradation Kinetics, EC Doc. Ref. Sanco/10058/2005, version. 2005;1.

[36] Ardia D, Boudt K, Carl P, Mullen KM, Peterson BG. Differential evolution with DEoptim. R Journal. 2011 Jun 1;3(1):27–34.

[37] Xiang Y, Hoeng J, Martin F, inventors; Philip Morris Products SA, assignee. Systems and methods for generating biomarker signatures with integrated dual ensemble and generalized simulated annealing techniques. United States patent application US 14/409,679.2013 Jun 21.