The Numerical Analyses of Global Optimization and Simulated Annealing

Jacques Nel*
jmn23@my.yorku.ca
ID. 212588109

Celina Landolfi*
clandolfi17@gmail.com
ID. 215587892

Gian Alix*
gian.alix@gmail.com
ID. 214760870

KEYWORDS

global optimization, simulated annealing, continuous variables

1 INTRODUCTION

Global optimization, a branch of applied mathematics and numerical analysis, is concerned with finding points on a bounded subset S of $\mathbb{R}^n$ in which some function f assumes its optimal value [1]. The function may have numerous local optima values and instead of searching for local points that cannot be improved, global optimization aims to find the global optimum on the bounded subset [2]. Thus, global optimization is a stronger version of local optimization and is desirable, but not necessary, in many practical applications; however, there are several applications where the use of global optimization is crucial including problems in safety verification, chemistry and hard feasibility problems [2]. In these cases, using local optimization methods could potentially return useless information, could be unrealistic with respect to the real world or could potentially underestimate the problem at hand [2]. Despite the obvious importance of global optimization and the efforts that have been invested into developing algorithms, the results have been unsatisfactory thus far and more work is needed for the optimization of more complicated functions [1]. Currently, numerical methods are used for the optimization of complicated functions, but these often cannot produce optimal results and will return a value close to a global optimum instead. By 'close to', we mean the following:

Definition 1.1 ('Close To' Global Minimum). For $\epsilon > 0, B_f(\epsilon)$ is the set of points with a value close to the minimal point, i.e.

$$B_f(\epsilon) = \{x \in S | \exists_{x_{min}} : |f(x) - f(x_{min})| < \epsilon\} \tag{1.1}$$

Furthermore, we construct a formal definition for what it means for a set of points to be close to a minimal point:

Definition 1.2. Let ϵ be any positive real number. Then we define $B_x(\epsilon)$ to be the set such that:

$$B_x(\epsilon) = \{x \in S \mid \exists x_{\min} \, : \, ||x - x_{\min}|| < \epsilon\} \tag{1.2}$$

and this is a set containing all points close to the minimal.

There are two classes of numerical methods in regards to global optimization: deterministic, in which the minimization process depends on probabilistic events, and stochastic methods, which does not use probabilistic information [1]. Unfortunately, deterministic global optimization methods have several disadvantages including that the global optimum can only be found after an exhaustive search over S, there is no guarantee of success from the method and that many assumptions must be made about f [1]. On the other hand, stochastic methods generally have better computational results than deterministic methods and can almost all be proven to find a global optimum that is asymptotically convergent with probability 1 [1]. With this taken into consideration, the concentration of global optimization methods will be on stochastic methods, specifically simulated annealing.

2 PROBLEM DEFINITION

Let $S \subset \mathbb{R}^n$ be a bounded set on $\mathbb{R}^n$ and $f : S \to \mathbb{R}^n$ be an n-dimensional real-valued function.
The problem that simulated annealing wishes to solve is to find a point $x_{min} \in S$ such that $f(x_{min})$ is a global minimum on S.

Specifically,

$$\forall x \in S : f(x_{\min}) \leq f(x) \tag{2.1}$$

The problem deals with global minimization; however, simulated annealing can find global maximization in the same way by reversing the sign of f without loss of generality [1].

3 THE NUMERICAL METHOD

Since local optimality does not guarantee global optimality, a fundamental concern in global optimization is getting stuck at a local optimum point [1]. In order to avoid and overcome this, a specific class of stochastic optimization is used, which is simulated annealing. Simulated annealing originates from the physical annealing process that is well known in condensed matter physics [1]. Physical annealing is a thermal process where the heating and slow cooling of a metal in a heat bath brings it into a more uniformly crystalline state where the free energy takes its global minimum. The role of temperature in physical annealing is to allow the particles to reach higher energy states in order to overcome energy barriers that would force them into local minima [2].

In greater detail, a state with energy level E_1 is compared to a state with energy level E_2, which is obtained by moving one of the particles into another location by a small displacement. E_2 is only accepted if the movement of the particle brings the system in a state of lower energy (i.e. $E_2 - E_1 \leq 0$). If this is not the case, a movement to a state of higher energy E_2 (also called a deterioration) is only accepted with probability $e^{-\frac{(E_2-E_1)}{kT}}$, where k is the Boltzmann constant and T is the temperature [1]. However, the probability of accepting these deteriorations descends slowly towards zero as this process continues. Thus, the repetition of this process for a large enough number of particle movements using this stochastic acceptance criterion for deteriorations make it possible to overcome becoming stuck at local minima and leads to a (near) global minima [1].

In 1983, Kirkpatrick, Gellatt and Vecchi applied the physical annealing of metal with combinatorial minimization by replacing energy with a cost function and the movement of the particles in

*All authors contributed equally to this research.

the physical system became analogous to a trial in the combinatorial minimization problem [1]. This algorithm proves to have many benefits when applied to combinatorial minimization problems including guarantee of convergence to global minimum, generally applicable to the cost function and easy to implement with good performance [1].

The approach of the simulated annealing algorithm is to generate homogeneous Markov chains of finite length at a finite sequence of descending values of the control parameter [1]. The cooling schedule consists of a set of parameters that controls the convergence of the algorithm, as listed below:

- initial value of the control parameter c
- a decrement function for decreasing the value of the control parameter c
- a stop criterion
- a finite length, L, of each Markov chain [1]

The parameters of the cooling schedule are described below in more detail.

3.1 Initial value of the control parameter (c_0)

The basic assumption regarding the initial value of the control parameter is that c_0 should be sufficiently large such that approximately all transitions are accepted at this value. If we let $\chi(c)$ denote the ratio between the number of accepted transitions and number of accepted transitions along the Markov chain at c. The problem of determining c_0 can be posed in terms of requiring the initial acceptance ratio $\chi_0 = \chi(c_0)$ to be close to 1.

The authors of [1] propose that c_0 be determined by the following scheme: The objective function $f(\boldsymbol{x})$ is sampled m times with $\boldsymbol{x} \sim$ Uniform $(\mathcal{S})$. Let $\Delta f = f(\boldsymbol{y}) - f(\boldsymbol{x})$, and $\Delta f^+ = \{\Delta f : \Delta f > 0\}$. Let m_1 denote the number of accepted transitions, with $m = m_1 + m_2$, and let $\overline{\Delta f^+}$ be the average values of Δf_{xy} for which $\Delta f_{xy} > 0$

$$c_0 = \overline{\Delta f^+} \left(\ln \frac{m_2}{m_2 \chi_0 + (1 - \chi_0) m_1} \right)^{-1} \tag{3.1}$$

In practice, eq. (3.1) was found to be problematic. The logarithm can result in a negative c_0 which does not make sense. Furthermore, if m is not sufficiently large, c_0 exhibits very high variance. This occurs because the mean in $\overline{\Delta f^+}$ is very sensitive to extreme values. Quick examination of the histograms of Δf^+ of the Rosenbrock test function suggested that Δf^+ is heavy-tailed.

Instead let's suppose we want the χ_0-th quantile transition to have a high ($p = 0.9$) probability of being accepted. Section 3.1 shows that $\exp -\Delta f / c$ maps Δf onto a probability. If we let $k = \chi$-th quantile of the Δf^+ values, we can determine c_0 with

$$\exp -\frac{k}{c_0} \geq p \implies c_0 = -\frac{k}{\log p} \tag{3.2}$$

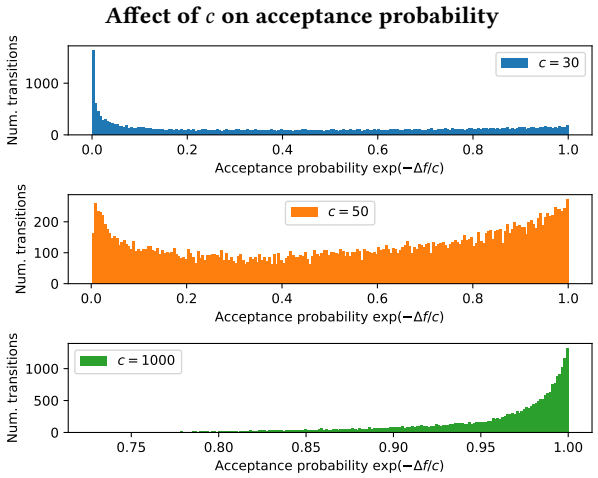


Figure 3.1: Using the Rosenbrock function, 50,000 transitions were computed. Among those, the positive transitions were selected and the exponential function $exp(-\Delta f/c)$ was applied and used as the acceptance criterion. Using three different c values of 30, 50 and 1000, it is shown that a greater value of c leads to more transitions given a higher probability of being accepted.

3.2 Decrement of the control parameter

The decrement function is used to decrease the value of c
The new value of c, c' is calculated by:

$$c' = c \left(1 + \frac{c \ln(1 + \delta)}{3\sigma(c)} \right)^{-1} \tag{3.3}$$

where $\sigma(c)$ is the standard deviation of the values of the cost function of the points of the Markov chain at c and the constant δ is the distance parameter that determines the speed of the decrement of the control parameter c [1].

Particular care must be given towards the $\sigma(c)$ term in the denominator of eq. (3.3). Especially when c is small, towards the later stages of the annealing schedule, a Markov chain at the current c can fail to make any transitions. In such situations, $\sigma(c) = 0$, and we have both a divide-by-zero exception, and c can be set to 0 prematurely, which has the result of triggering the stop condition, before a good solution is reached.

One way to address this issue is to repeatedly sample a Markov chain until some minimum number of transitions are performed. Due to descent direction component of point generation alternative B, the Markov chain will usually make at least one transition, even if it is very small, thus avoiding the divide-by-zero problem.

3.3 Stop condition

Particular care needs to be given to devising an appropriate stopping condition. First, let $\overline{f}(c_0)$ denote the mean value of the points in the initial Markov chain, and $\overline{f}(c)$ denote the average values of the points in the chain at c. Then $\overline{f}_s(c)$ is the smoothed value of $\overline{f}$ over a number of chains in order to reduce fluctuations of $\overline{f}(c)$. In practice

Figure 3.2: Cooling schedule for Branin

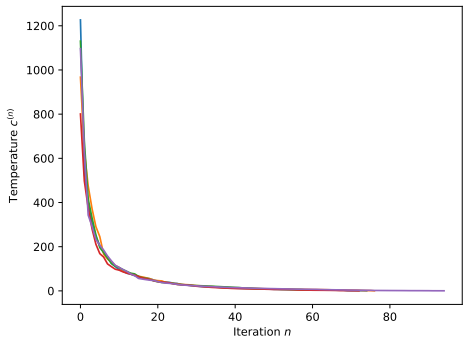


Figure 3.3: Cooling schedule for Goldstein-Price

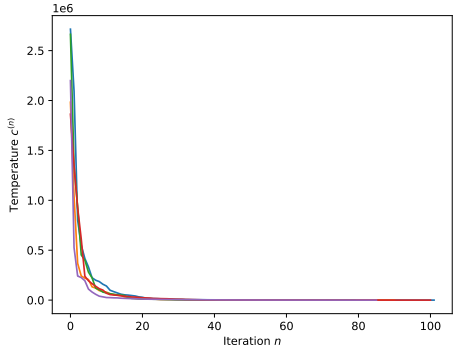


Figure 2 and Figure 3 represent an example Cooling Schedule for Branin and Golden-Price respectively, where each colour on the curve represents a different run, for 4 total runs.

the smoothing is applied using a small smoothing parameter $0 < \gamma < 1$ and

$$\begin{aligned} \overline{f_s}^{(n+1)} &= \gamma \overline{f}^{(n+1)} + (1-\gamma)\overline{f_s}^{(n)} \\ \overline{f_s}^{(0)} &= \overline{f}^{(0)} \end{aligned} \tag{3.4}$$

The algorithm is terminated if:

$$\left| \frac{d\overline{f_s}(c)}{dc} \frac{c}{\overline{f}(c_0)} \right| < \epsilon_s \tag{3.5}$$

where ϵ_s is the stop parameter, a small positive real number. The derivative in eq. (3.5) is simply calculated using a finite difference method [1].

3.4 Length of the Markov chains

Let n be the dimension of S and let L_0 be a constant called the standard length. Then

$$L = L_0 \cdot n \tag{3.6}$$

is the length of the Markov chain, which must be sufficiently large in order to enable the algorithm to explore the neighbourhood of a given point in all directions [1].

A pseudo-code for the algorithm can be found in **Algorithm ??**.

3.5 Generation of Points

There are several ways to generate new points from a given point and two alternatives are used for the purposes of this project, Alternative A and Alternative B:

(A) A uniform distributions on S:

$$g_{xy} = \frac{1}{m(S)} \tag{3.7}$$

A disadvantage about Alternative A is that no structural information about function values is used [1].

(B) Either a point is drawn from a uniform distribution over S or a step is made into a descent direction from the current point:

$$g_{xy} = \begin{cases} LS(x) & \text{if } w > t \\ \dfrac{1}{m(S)} & \text{if } w \leq t \end{cases} \tag{3.8}$$

where t is a fixed number in the interval [0,1) and w is a random number from $U[0,1)$. LS(x) is a Local Search procedure (i.e Steepest Descent Method) that generates a point y in a descent direction of x. [1]

4 THEORETICAL RESULTS

The mathematical model of the simulated annealing algorithm is based on the theory of Markov chains and thus, the following definitions are necessary:

Definition 4.1 (Markov Chains). A *Markov chain* in the simulated annealing algorithm is a sequence of trials. $X(k)$ is a random variable denoting the kth trial by simulated annealing, and the outcome of a trial is a point $x \in S$ that is dependent on the previous trial.

Definition 4.2 (The Generation Probability Distribution). g_{xy}, as seen in **Section 3.5**, is the *generation probability distribution*. That is, g_{xy} is the probability distribution function for generating a point y from point x at a fixed value of the control parameter c.

Definition 4.3 (Acceptance Probability). A_{xy} is the *acceptance probability*. That is, A_{xy} is the probability of accepting point y as a possible new point if x is the current point in a Markov chain.

Definition 4.4 (Transition Probability). A point $x \in S$ transformed to a point $y \in T \subset S$, with the probability of generating and accepting a point in T given that $x \notin T$ is called a *transition probability*. Hence, for any current point x in the Markov Chain, then the probability that an element in T is the next point of the

chain is given as:

$$P(T \mid x;\, c) = \begin{cases} \displaystyle\int_{y\in T} p_{xy}(c)\,\mathrm{d}y & \text{for } x \notin T \\ \displaystyle\int_{y\in T} p_{xy}(c)\,\mathrm{d}y + \left(1 - \int_{y\in S} p_{xy}(c)\,\mathrm{d}y\right) & \text{for } x \in T \end{cases} \tag{4.1}$$

where

$$p_{xy}(c) = g_{xy} \cdot A_{xy}(c) \tag{4.2}$$

and

$$P(T \mid x;\, c) = \mathbb{P}\{X(l) \in T \mid X(l-1) = x\,;\, c\} \tag{4.3}$$

4.1 Asymptotic Convergence of the Algorithm

Before asymptotic convergence of the algorithm is proved, the following definitions are necessary:

Definition 4.5 (Stationary Probability Distribution Function). A probability distribution function $r(x, c)$ is said to be *stationary* if the following two conditions are met:

$$(1)\ \forall_{x\in S} : r(x,c) = \int_{y\in S} r(y,c)p_{yx}(c)\mathrm{d}y + r(x,c)\left(1 - \int_{y\in S} p_{xy}(c)\,\mathrm{d}y\right) \tag{4.4}$$

$$(2)\ \int_{x\in S} r(x,c)\mathrm{d}x = 1 \tag{4.5}$$

From this definition, the following is a stationary probability distribution since it meets the two conditions, which will later be used to show that the simulated annealing algorithm converges to a near minimal solution:

$$q(x,c) = \exp\left(-\frac{f(x) - f_{\min}}{c}\right)\left[\int_{y\in S} \exp\left(-\frac{f(y) - f_{\min}}{c}\right)\mathrm{d}y\right]^{-1} \tag{4.6}$$

Definition 4.6 (Probability of Transformation). The probability that a point $x \in S$ is transformed into a point $y \in T \subset S$ in k trials is:

$$P^{(k)}(T \mid x;\, c) = \begin{cases} \displaystyle\int_{y\in T} p_{xy}^{(k)}(c)\,\mathrm{d}y & \text{for } x \notin T \\ \displaystyle\int_{y\in T} p_{xy}^{(k)}(c)\,\mathrm{d}y + \left(1 - \int_{y\in S} p_{xy}(c)\,\mathrm{d}y\right)^{k} & \text{for } x \in T \end{cases} \tag{4.7}$$

where

$$p_{xy}^{(k)}(c) = \int_{z\in S} p_{xz}^{(k-1)}(c)p_{zy}(c)\,\mathrm{d}z + p_{xz}^{(k-1)}(c) + \left(1 - \int_{z\in S} p_{yz}(c)\,\mathrm{d}z\right) + \left(1 - \int_{z\in S} p_{xz}(c)\,\mathrm{d}z\right)^{k-1} p_{xy}(c) \tag{4.8}$$

i.e. $p_{xy}^{(k)}(c)$ is the quasi probability distribution function of transforming x into y in k trials. $p_{xy}^{(k)}(c)$ is the summation of three terms, which are outlined below:
(i). Term 1: the quasi probability distribution function of transforming x into z in $k-1$ trials, and from z to y in the next trial integrated over all z.
(ii). Term 2: the quasi probability distribution function of transforming x into y in $k-1$ trials and then rejecting the kth trial.
(iii). Term 3: the quasi probability distribution function of transforming x into y in one trial after $k-1$ rejected trials from x.

In **Eq. (4.6)**, we present a stationary probability distribution function, which is the necessary requirement for the Simulated Annealing algorithm to converge to the minimum solution. We present the theorem that:

Theorem 4.7. If there is a finite number of local minima for a uniformly continuous function f, then:

$$\forall \epsilon > 0\ :\ \lim_{c\to 0} \int_{y\in B_f(\epsilon)} q(y,c)\,\mathrm{d}y > 1 - \epsilon \tag{4.9}$$

Proof. For a finite number of local minima, we have:

$$\exists \epsilon_1 > 0\ :\ |f(x_{\text{loc}}) - f_{\min}| > \epsilon_1 \tag{4.10}$$

$$\exists \epsilon_2 > 0,\ \forall x_{\min}\ :\ ||x_{\text{loc}} - x_{\min}|| > \epsilon_2 \tag{4.11}$$

where $f_{\min} = f(x_{\min})$, $\forall x_{\min}$ as per **Eq. (2.1)** and x_{loc} is a local, non-global minimum point. Then pick a positive ϵ such that:

$$\epsilon < \frac{1}{4}\min\{\epsilon_1, \epsilon_2\} \tag{4.12}$$

It should be noted that if all minima are global, then select ϵ such that $\exists x \in S : f(x) - f_{\min} > \epsilon$.

As f is uniformly continuous, then:

$$\exists \delta_1 > 0,\ \forall x, y \in S : ||x - y|| \le \delta_1 \Longrightarrow |f(x) - f(y)| < \frac{\epsilon}{2} \tag{4.13}$$

Choosing a δ such that $\delta = \min\{\delta_1/2, \epsilon\}$, we have:

$$\forall y \in B_x(\delta) : f(y) - f_{\min} < \frac{\epsilon}{2} \tag{4.14}$$

where $B_x(\delta)$ is given by **Definition 1.2**.

Consider a point $x_0 \in S \setminus B_x(\delta)$ where $f(x_0) - f_{\min} = \epsilon$. Note that this is possible due to the continuity of f. Then therefore:

$$\begin{aligned}
\lim_{c\to 0} q(x_0, c) &= \lim_{c\to 0} \frac{\exp(-(f(x_0) - f_{\min}/c))}{\int_{y\in S} \exp(-(f(y) - f_{\min})/c)} \, \mathrm{d}y \\
&= \lim_{c\to 0} \frac{\exp(-\epsilon/c)}{\int_{y\in S} \exp(-(f(y) - f_{\min})/c)} \, \mathrm{d}y \\
&= \lim_{c\to 0} \left[\int_{y\in S} \exp((\epsilon - (f(y) - f_{\min}))/c) \, \mathrm{d}y \right]^{-1} \\
&= \lim_{c\to 0} \left[\int_{y\in S\setminus B_x(\delta)} \exp((\epsilon - (f(y) - f_{\min}))/c) \, \mathrm{d}y \right. \\
&\qquad \left. + \int_{y\in B_x(\delta)} \exp((\epsilon - (f(y) - f_{\min}))/c) \, \mathrm{d}y \right]^{-1} \\
&\leq \lim_{c\to 0} \left[\int_{y\in B_x(\delta)} \exp((\epsilon - (f(y) - f_{\min}))/c) \, \mathrm{d}y \right]^{-1} \\
&\leq \left[\lim_{c\to 0} \int_{y\in B_x(\delta)} \exp((\epsilon - \epsilon/2)/c) \, \mathrm{d}y \right]^{-1} \\
&= \left[\lim_{c\to 0} \exp(\epsilon/2c) m(B_x(\delta)) \right]^{-1} \qquad (4.15)
\end{aligned}$$

and can clearly be seen that this approaches to 0 as $c \to 0$. And so with the Lebesgue measure of S, $m(S)$, then we have:

$$\exists c_0 > 0, \, \forall c < c_0 \, : \, q(x_0, c) < \frac{\epsilon}{m(S)} \qquad (4.16)$$

And hence:

$$\forall c < c_0, \, \forall x \in S^+(x) \, : \, q(x, c) \leq q(x_0, c) < \frac{\epsilon}{m(S)} \qquad (4.17)$$

and that:

$$\forall c < c_0, \, \forall x \in S^-(x) \, : \, f(x) - f_{\min} < \epsilon \qquad (4.18)$$

with $S^+(x)$ and $S^-(x)$ defined as follows:

$$S^+(x) = \{y \in S \,|\, f(y) \leq f(x)\} \qquad (4.19)$$

$$S^-(x) = \{y \in S \,|\, f(y) > f(x)\} \qquad (4.20)$$

$\forall c < c_0$, then we have:

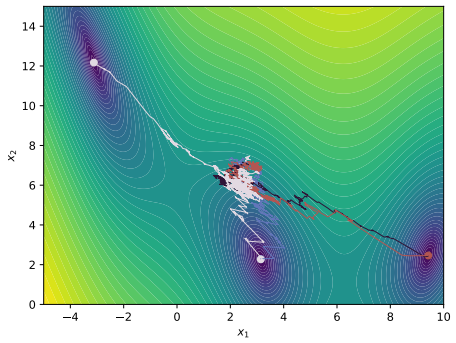
$$\begin{aligned}
1 &= \int_{y\in S} q(y, c) \, \mathrm{d}y \\
&= \int_{y\in S^-(x_0)} q(y, c) \, \mathrm{d}y + \int_{y\in S^+(x_0)} q(y, c) \, \mathrm{d}y \\
&< \int_{y\in B_f(\epsilon)} q(y, c) \, \mathrm{d}y + \int_{y\in S^+(x_0)} \frac{\epsilon}{m(S)} \, \mathrm{d}y \\
&\leq \int_{y\in B_f(\epsilon)} q(y, c) \, \mathrm{d}y + \epsilon \qquad (4.21)
\end{aligned}$$

Know that $B_f(\epsilon) = S^-(x_0)$. Because of **Eq. (4.11)** and **Eq. (4.12)**, then there is no local minimum in $B_f(\epsilon)$. Thus showing that

$$\lim_{c\to 0} \int_{y\in B_f(\epsilon)} q(y, c) \, \mathrm{d}y > 1 - \epsilon \qquad (4.22)$$

proving the **Theorem 4.7** as desired. □

Figure 5.1: Average SA trajectories for BR



5 NUMERICAL RESULTS

5.1 Examples of running Simulated Annealing

Table 5.1: Simulated Annealing with BR test to $\tau = 10^{-7}$

	$\boldsymbol{x}^\star$	$f(\boldsymbol{x}^\star)$	#f evals	#∇f evals
1	(9.42478, 2.47500)	0.39789	448	166
2	(9.42478, 2.47500)	0.39789	607	224
3	(9.42478, 2.47500)	0.39789	688	232
4	(-3.14159, 12.27500)	0.39789	567	167
5	(-3.14159, 12.27500)	0.39789	648	195
6	(-3.14159, 12.27500)	0.39789	568	199
7	(3.14159, 2.27500)	0.39789	529	183
8	(3.14159, 2.27500)	0.39789	486	172
9	(3.14159, 2.27500)	0.39789	528	204
10	(-3.14159, 12.27500)	0.39789	527	190
11	(3.14159, 2.27500)	0.39789	607	195
12	(-3.14159, 12.27500)	0.39789	568	201
13	(-3.14159, 12.27500)	0.39789	489	179
14	(-3.14159, 12.27500)	0.39789	608	203
15	(9.42478, 2.47500)	0.39789	728	228
16	(3.14159, 2.27500)	0.39789	528	179
17	(9.42478, 2.47500)	0.39789	728	235
18	(3.14159, 2.27500)	0.39789	567	201
19	(3.14159, 2.27500)	0.39789	649	225
20	(3.14159, 2.27500)	0.39789	767	244

Table 5.1: note that all 3 global minima are found. The following parameters were used and serve as good default paramters:
$L_0 = 20, \delta = 1.1, \epsilon_s = 10^{-6}, \chi = 0.9, \gamma = 10^{-2}$ and $T = 0.5$.

Table 5.2 shows average required number of runs for the benchmark problems RB, GP, BR, H3, and H6, Simulated Annealing is run repeatedly until all known global minima are found for the problem. A solution $\hat{\boldsymbol{x}}_i$ is considered the same if $|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_j^\star|_2 \leq \tau = 10^{-4}$. This process is repeated $M = 50$ times and average number of runs, average function and Jacobian values are reported. In all cases, every

Figure 5.2: Average SA trajectories for GP

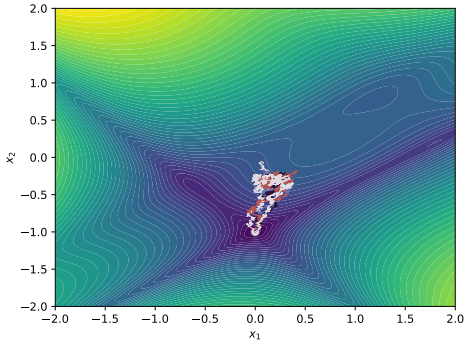


Table 5.2: Average number of runs required to solve problems

	Problem	Avg. num runs	Solutions found	$\#f$ evals	$\#\nabla f$ evals
1	RB	1.0	1	614.58	210.68
1	GP	1.6	1	1072.38	385.96
3	BR	5.18	3	3012.68	1041.06
1	H3	1.12	1	575.08	211.06
1	H6	1.58	1	737.7	284.1

Table 5.3: Global and local minima found by Simulated Annealing

Prob	Type	$\boldsymbol{x}_i^\star$	$f(\boldsymbol{x}_i^\star)$
RB	g	(1.0, 1.0)	0.0
GP	g	(-0.0, -1.0)	3.0
	l	(-0.6, -0.4)	30.0
		(1.8, 0.2)	84.0
BR	g	(-3.14159, 12.275)	0.39789
		(3.14159, 2.275)	0.39789
		(9.42478, 2.475)	0.39789
H3	g	(0.11461, 0.55565, 0.85255)	-3.86278
	l	(0.10934, 0.86052, 0.56412)	-3.08976
H6	g	(0.20169, 0.15001, 0.47687, 0.27533, 0.31165, 0.6573)	-3.32237
	l	(0.40465, 0.88244, 0.8461, 0.57399, 0.13893, 0.0385)	-3.20316

global minimum was found. Test problems BR and H6 show an average higher number of required runs per global minima.

The following Simulated Annealing parameters were used: $l_0 = 20, \delta = 1.1, \epsilon_s = 1e-4, \chi = 0.9, \gamma = 10^{-2}$, and $t = 0.5$.

6 NUMERICAL COMPARISONS

6.1 MS (Multi-start)

Multi-start techniques form a family of heuristic global optimization algorithms. These techniques work in a two-phase process. In the first global phase, starting points are sampled in the feasible region or domain $\mathcal{S}$ of the objective function. The most naive implementation of the global phase is to simply perform a uniform sampling of $\mathcal{S}$. Some members of this family of algorithms utilize sophisticated heuristics to generate starting points such as the Scatter Search technique.

The local phase of multi-start algorithms use the starting points from the global phase to perform a local optimization using one of the many algorithms for local optimization. Multi-start algorithms alternate between the global and local phases until a solution is found [5].

The hardest part of divising an efficient Multi-start method is the selection of an appropriate stopping rule. Variations in the literature range from several addhoc holes to Bayesian stoppic rules. For the sake of comparison we utilize a double-box stopping rule [cite box-cstr here].

Let $m(\mathcal{S})$ be the Lebesque measure of the search region $\mathcal{S}$. Since the local search method is deterministic, in the limit of $n \longrightarrow \infty$ runs of the local search method, the algorithm will converge to w unique local minima. Each local minimum $\boldsymbol{x}_i^\star \in \mathcal{S}$ has an associated region of attraction A_i defined as

$$A_i := \left\{ \boldsymbol{x} : \boldsymbol{x} \in \mathcal{S}, \mathsf{LS}(\boldsymbol{x}) = \boldsymbol{x}_i^\star \right\}.$$

Since $\mathcal{S}$ contains w local minima, and the $A_i \cap A_j = \emptyset$, A_i partitions $\mathcal{S}$ i.e.,

$$\bigcup_{i=1}^{w} A_i = \mathcal{S}.$$

Furthermore, if $m(A_i)$ denotes the Lebesque measure of A_i, then

$$m(\mathcal{S}) = \sum_{i=1}^{w} m(A_i).$$

If some initial point $\boldsymbol{x}_0 \sim \text{Uniform}(\mathcal{S})$, then the probability of $\boldsymbol{x}_0$ begin contained in A_i is simply

$$P(\boldsymbol{x}_0 \in A_i) = \frac{m(A_i)}{m(\mathcal{S})}.$$

The double-box stopping rule is based on the idea that we want to ensure that all A_i are sampled in $\mathcal{S}$, but additional sampling results in uncessary computation. Define C has a relative measure of the coverage after the discovery of w local minima, as

$$C = \sum_{i=1}^{w} \frac{m(\mathcal{A}_i)}{m(\mathcal{S})}. \tag{6.1}$$

A sensible heuristic is to stop when $C \longrightarrow 1$. The quantity inside the summation of eq. (6.1) is not calculatable in practice, but as $w \longrightarrow \infty$, we approximate it with

$$C \approx \sum_{i=1}^{w} \frac{L_i}{L}, \tag{6.2}$$

where L_i is the number of starting points of the LS which converged to the local minimizer $\boldsymbol{x}_i$, and L is the total of initial points so far. The quantity in eq. (6.2) is equal to 1 by definition, so another means must be determined. Construct a region $\mathcal{S}_2$ such that $\mathcal{S} \subset \mathcal{S}_2$ and $m(\mathcal{S}_2) = 2 \cdot m(\mathcal{S})$. For each iteration, we sample from $\mathcal{S}_2$ until we have a point in $\mathcal{S}$, in other words points in $A_0 = \mathcal{S}_2 \setminus \mathcal{S}$ are

discared. Also let L_0 denote the number of sampled points in A_0. The total count of sampled points is now given by

$$L = L_0 + \sum_{i=1}^{w} L_i,$$

and the relative coverage C is now

$$C = \frac{1}{m(\mathcal{S})} \sum_{i=1}^{w} m(A_i) = 2 \sum_{i=1}^{w} \frac{m(A_i)}{m(\mathcal{S}_2)}$$

and finally we can approximate relative coverage with

$$C \approx \frac{2}{L} \sum_{i=1}^{w} L_i$$

After n iterations, let M_n denote the number of points in $\mathcal{S}_2$, and n are in $\mathcal{S}$. Then define $\delta_n := n/M_k$ which has an expectation which in the limit of large n

$$\langle \delta \rangle_n = \frac{1}{n} \sum_{i=1}^{n} \delta_i \longrightarrow \frac{m(\mathcal{S})}{m(\mathcal{S}_2)} = \frac{1}{2}$$

The variance is $\sigma_n^2(\delta) = \left\langle \delta^2 \right\rangle_n - \langle \delta \rangle_n^2$ which $\longrightarrow 0$ as $n \longrightarrow \infty$. Finally, the double-box rule is

Double-box stopping rule:

(1) Continue iterating if new minima are found. **(2)** If no new minima are found, let $\sigma_{\text{last}}(\delta)$ be the s.t.d. at the last iteration at which a minimum was found. Contnue iterating while

$$\sigma^2(\delta) < \rho \sigma_{\text{last}}^2(\delta) \tag{6.3}$$

where $\rho \in (0, 1)$ is a paramter that performs exhaustive search when ρ is close to 0 and emphasises less iterations when ρ is close to 1.

Constructing $\mathcal{S}_2$ from $\mathcal{S}$:

In order to apply the double-box stopping rule, we need to construct the box region $\mathcal{S}_2$ such that $m(\mathcal{S}_2) = 2 \cdot m(\mathcal{S})$. Suppose $\mathcal{S} \in \mathbb{R}^n$, then we can scale the bounds of $\mathcal{S} = [l_1, u_1] \times \cdots [l_n, u_n]$ by $2^{1/n}$, i.e.,

$$l_i^{(2)} = l_i - \frac{1}{2}\left(2^{1/n} - 1\right)(u_i - l_i) \text{ for } i = 1, \ldots n$$
$$u_i^{(2)} = u_i + \frac{1}{2}\left(2^{1/n} - 1\right)(u_i - l_i) \text{ for } i = 1, \ldots n$$

6.2 DE (Differential Evolution)

Differential Evolution is global, gradient-free stochastic optimization algorithm. This population-based method mutates each solution candidate by mixing the solution with other members of the population [4]. *scipy.optim.differential_evolution* is used as comparison.

6.3 BH (Basin Hopping)

Basin-hopping is another two-phase global optimization algorithm inspired by energy minimization of clusters of atoms [6]. *scipy.optimize.basinhopping* is used for comparison.

6.4 Evaluation metrics

6.4.1 Solution diversity. Given that *mathcalS* is partitioned by regions of attraction A_i around each local miminum, one way to measure the efficiency of a global optimization method is to measure the frequency with which the method converges to each global minimum. A more efficient method will converge to each global minimum with more uniform probability, implying that less total iterations are needed to find all the global minima.

Suppose that $f : \mathcal{S} \longrightarrow \mathbb{R}$ has exactly w global minima. After a large number M of runs, the method has converged to a global minimum $\boldsymbol{x}_i^\star$ w_i times. We can define a *solution diversity score* as

$$\xi_M = \sum_{i=1}^{w} \left| \frac{w_i}{M} - \frac{1}{w} \right|. \tag{6.4}$$

Generally, the regions of attraction A_i for all local and global minima could have different Lebesque measures, so a technique that uses global uniform sampling would oversample some solutions and undersample others.

An ideal technique would converge to solution $\boldsymbol{x}_i$ on M/w occasions after M runs. Equation (6.4) is minimized by such a technique yielding uniform convergence to all minima. A lower ξ_M indicates a more efficient algorithm, whereas if some $\boldsymbol{x}_i^\star$ are over represented (and some are under represented), ξ_M would be higher, and thus indicate an inefficient algorithm.

6.4.2 Number of evaluations of f and ∇f: The number of evalaluations of f and ∇f is another obvious indication of the computational efficiency of a given optimization method. Our implementation, and methods to which we compare Simulated Annealing facilate the counting of these function evaluations in all sub procedures of a given method.

7 CONCLUSIONS

something....

REFERENCES

[1] Anton Dekkers and Emile Aarts. 1991. Global Optimization and Simulated Annealing. *Mathematical Programming* 50 (1991), 367–393.

[2] Arnold Neumaier. 2003. Complete Search in Continuous Global Optimization and Constraint Satisfaction.

[3] H.H. Rosenbrock. 1960. An automatic method for finding the greatest or least value of a function. *Comput. J.* 3 (1960), 175–184.

[4] R Storn and K Price. 1997. Differential Evolution-a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11 (1997), 341–359.

[5] Zsolt Ugray, L. Lasdon, J. Plummer, Fred W. Glover, J. P. Kelly, and R. Martí. 2007. Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization. *INFORMS J. Comput.* 19 (2007), 328–340.

[6] D.J. Wales and J.P.K Doye. 1997. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *Journal of Physical Chemistry* 101 (1997), 5111–5116.

A BENCHMARK FUNCTIONS USED

A.1 RB (Rosenbrock)

The Rosenbrock function is a widely used non-convex test function with a one global minimum inside of a long, narrow valley. It is defined as

$$f(x_1, x_2) := (a - x)^2 + b\left(x_2 - x_1^2\right)^2$$

The global minimum is at $\boldsymbol{x}^\star = \begin{pmatrix} a & a^2 \end{pmatrix}^T$. The typical values are $a = 1$ and $b = 100$ [3].

A.2 GP (Goldstein-Price)

The Goldstein-Price[1] test function in $\mathbb{R}^2$ is given by

$$f(x_1, x_2) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right]$$

$$\times \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$$

$\mathcal{S} = \{\boldsymbol{x} : -2 \leq x_i \leq 2, i = 1, 2\}$. It has a global minimum at $\boldsymbol{x}^\star = \begin{pmatrix} 0 & -1 \end{pmatrix}^T$.

A.3 H3 and H6 (Hartmann's family)

$$f(\boldsymbol{x}) = -\sum_{i=1}^{m} c_i \exp\left(-\sum_{j=1}^{n} a_{ij}\left(x_i - p_{ij}\right)^2\right)$$

where for H3 in $\boldsymbol{x} \in \mathbb{R}^4$, with $m = 4$ and $n = 3$ is given by

$$A = [a_{ij}] = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix},$$

$$P = [p_{ij}] = \begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.038150 & 0.5743 & 0.8828 \end{bmatrix}$$

where for H6 in $\boldsymbol{x} \in \mathbb{R}^4$, with $m = 4$ and $n = 6$ is given by

$$A = [a_{ij}] = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix},$$

$$P = [p_{ij}] = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

A.4 S5, S7 and S10 (Shekel's Family)

The Shekel's family of functions, S5, S7, and S10 [1], is given as:

$$f(x) = -\sum_{i=1}^{m} ((x - a_i)^T (x - a_i) + c_i)^{-1}$$

which has a dimension of $n = 4$, and $m = 5$, $m = 7$, and $m = 10$ for the S5, S7, and S10 respectively. Let $x = (x_1, x_2, ..., x_n)^T$, and $a_i = (a_{i1}, a_{i2}, ..., a_{in})^T$.

Thus for S5:

$$A = [a_{ij}] = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.4 \end{bmatrix},$$

Then for S7:

$$A = [a_{ij}] = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{bmatrix},$$

Then for S10:

$$A = [a_{ij}] = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix},$$