Convergence Rate Evaluation of Derivative-Free Optimization Techniques

Thomas Lux

Roanoke College, Salem VA 24153, USA,
thlux@mail.roanoke.edu,
WWW home page: http://cs.roanoke.edu/~thlux/

Abstract. This paper presents a convergence rate comparison of five different derivative-free numerical optimization techniques across a set of 50 benchmark objective functions. Results suggest that Adaptive Memory Programming for constrained Global Optimization, and a variant of Simulated Annealing are two of the fastest-converging numerical optimization techniques in this set. Lastly, there is a mechanism for expanding the set of optimization algorithms provided.

Keywords: Optimization, Convergence, Metaheuristic, Derivative-free

1 Introduction

Many well defined engineering problems in the real world do not allow for the timely computation of an optimal solution. The task of *Optimization* is to find desirable solutions to problems in spaces that prohibit exhaustive search. In order to find desirable solutions while only sampling a relatively small subset of possible solutions, optimization algorithms make assumptions about the search space for the problem at hand. The goal of an optimization algorithm can vary, but for the purpose of this paper we consider algorithms attempting to find "good" solutions with the fewest computations possible.

The problems that optimization algorithms solve are often formulated as a function which takes a set of parameters as input and then returns the performance of that set of parameters as output. We define an *objective function* as a deterministic function of multiple real-valued parameters from $\mathbb{R}^n$ to $\mathbb{R}$. The computational cost of exhaustively searching for a minimum return value of one of these objective functions grows exponentially with respect to the number of input parameters and their acceptable values.

The class of objective functions that we focus on are those for which each parameter is bounded, and no derivative information is known about the objective function. In mathematical terms we define this as: given derivative-free objective function $f : \mathbb{R}^n \to \mathbb{R}$ minimize $f(\boldsymbol{x})$ subject to $a_i \leq x_i \leq b_i, (a_i, b_i \in \mathbb{R})$.

In this paper we analyze five different numerical optimization techniques, four of which are relatively new, and compare their *rates of convergence* on a set of well-understood objective functions with varying dimensionality. The purpose of this study is to propose a ranking of these algorithms in terms of their

likelihood to converge to an optimal solution in a restricted number of objective function evaluations. The five different numerical optimization techniques that we compare, in order of their creation, are: Simulated Annealing (SA)[1], Adaptive Memory Programming for constrained Global Optimization (AMPGO)[2], Cuckoo Search (CK)[3], Backtracking Search optimization Algorithm (BSA)[4], and quick Artificial Bee Colony (qABC)[5]. In Section 3 we list the objective functions, their bounds, the metrics for convergence evaluation, and the general formulation of each optimization algorithm. In Section 4 we provide figures and tables demonstrating the performance of each algorithm across our set of objective functions.

1.1 Metaheuristic Optimization Algorithms

Metaheuristic algorithms are useful for searching through objective function spaces for which there is no known derivative because they rely strictly on the objective values obtained and the internal heuristics of the algorithm. In [6] the general form of a metaheuristic optimization algorithm and the behavior of a random walk is defined. It is also mentioned that the mechanisms by which optimization algorithms achieve this random walk behavior is quite different. All of the numerical optimization algorithms that we analyze in this paper, with the exception of AMPGO, are metaheuristic algorithms. Each of the metaheuristic algorithms can be generalized to a random walk search model while AMPGO utilizes a memoized approximation of the objective function gradient for convergence.

2 Related Works

Numerical optimization is currently a rapidly expanding field of research. Every year there are more numerical optimization techniques and algorithms introduced and the five being compared in this paper provide only a small subset of all algorithms. This paper focuses specifically on derivative-free optimization techniques because they are useful for optimizing complex objective functions [7, 8]. These five derivative-free algorithms are only a few of large existing set and have been selected because they each have independent assertions claiming them to be the "best" optimization algorithms by various sources within recent years [3–5] [9, 10].

Due to the sheer quantity of optimization research papers being published, even the most thorough review papers [6] [11] do not compare all algorithms. Many optimization algorithms that have been proposed and asserted as comparatively better than others are compared strictly to older less powerful algorithms. Compounding this difficulty, much of the source code is written in languages for proprietary software such as Matlab and is not readily usable by the public. For these reasons, more research comparing modern numerical optimization algorithms as well as providing open sourced implementations of the algorithms in a freely available language needs to be done.

3 Methods

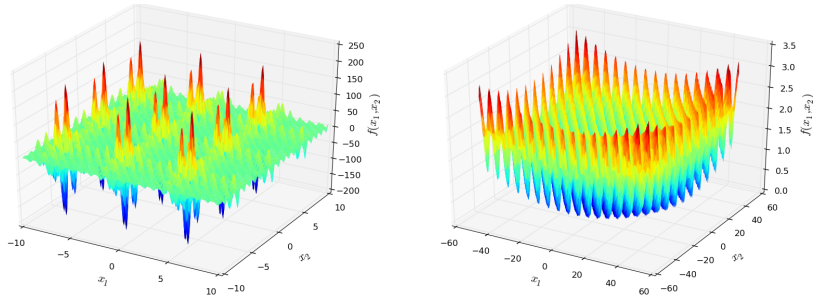
The 50 objective functions chosen for this evaluation are a sampling of functions used on other optimization algorithms as well as in some popular benchmark data sets. The function definitions can be found in the Appendix Table 1, while the number of dimensions used in testing and acceptable ranges for values are all listed in the Appendix Table 2. Plots of a representative subset of these functions in their two dimensional variants are seen in Figure 1.

In order to compare the relative performance of the different optimization algorithms across multiple objective functions, we use *Data Profiles*, *rank 0 probability*, and *best solution probability*. The remaining statistics that are normally included in a comparative study of optimization algorithms (maximum, minimum, average, and standard deviation of objective function values obtained) are available through electronic supplementary materials.

The parameters we use for each of the optimization algorithms are the default suggested values provided by the authors of the algorithms. The one exception to this is the simulated annealing algorithm, for which we use an *acceptance probability* of 0, and add a modification that increases the *temperature* each time a better solution is discovered. For exact parameters and implementation details, please see the referenced source code. Each optimization algorithm is allowed 5000 executions of each objective function. 5000 is selected under the assumption that these optimization algorithms would be used in the future on moderately difficult-to-compute objective functions which require at least 10 seconds to evaluate per iteration under 12-24 hour time constraints.

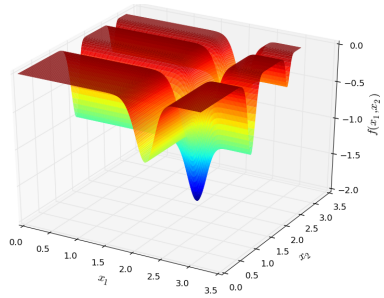
We use a *convergence in mean* stopping criteria over the objective function value obtained by an optimization algorithm after 5000 executions to determine an appropriate number of repeated trials. By using convergence in mean, we attempt to equally weight the results in terms of randomness. We consider the final objective function value obtained by an optimization algorithm to have converged in mean after ($t > 500$) trials if: for 100 sequential trials, the mean of the cumulative final objective function values shifts by less than 0.01%, in terms of the range of mean values encountered. For this set of objective functions and optimization algorithms the average number of trials needed to pass this convergence test was 1106.

Data Profile The *Performance Profile*, introduced by Dolan in [12], compares the *performance ratio* of multiple optimization algorithms given a stopping criteria such as a time restriction or maximum number of executions of the objective function. Performance Profiles can be used to compare optimization algorithms with the only drawback being the need for a singular stopping criteria. *Data Profiles*, created by Moré in [7], address the need for establishing a singular stopping criteria by measuring the convergence of an optimization algorithm after successive executions of the objective function. The primary measurement performed in a Data Profile is at each execution of the objective function. Supposing an optimization algorithm has executed an objective function f, for n iterations,

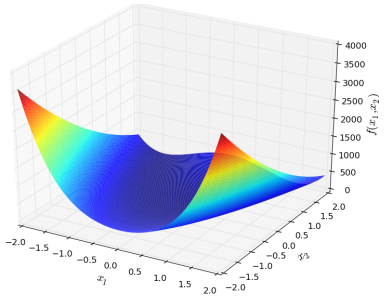


(a) Shubert Function

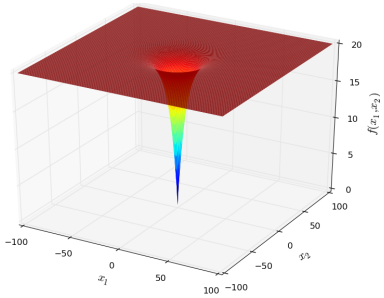
(b) Griewank Function



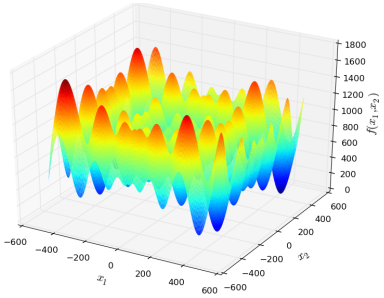
(c) Michalewicz Function



(d) Rosenbrock Function



(e) Easom Function



(f) Schwefel Function

Fig. 1. The 2 dimensional variants of some of the objective functions, provided as a sampling of the types of spaces used in testing. These plots were generated by the code freely available at [9].

starting with initial solution $\mathbf{x}_0$, where the best solution obtained by any optimization algorithm in the comparison set at the n^{th} execution of the objective function is f_L, the performance of that optimization algorithm can be measured by the percentage of solutions that pass the following criteria:

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L) \tag{1}$$

$\tau > 0$ is the convergence tolerance, and represents how close an optimization algorithm should be after n executions of the objective function to f_L given a specific f_0. We provide plots for each $\tau \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

Rank 0 Probability It is often desirable to rank optimization algorithms in terms of their performance. This becomes difficult when the algorithms incorporate randomness, but given some number of independent repeated trials we can still use simple counting techniques to calculate the *probability* that a selected algorithm will be ranked in a certain position. Particularly the position of interest for comparative studies is often that of the best algorithm, which we refer to as rank 0. Consider the following explanation.

Given some set of optimization algorithms A and an optimization algorithm of interest $\beta \in A$, consider objective function f where each optimization algorithm has been allowed n executions of f. At the n^{th} execution, β along with each other algorithm in A, has some set of objective function values obtained from repeated trials T_β, where $n(T_\beta)$ is the number of repeated trials. Now, assuming that the values in T_β are independent and hence equally likely, we can count the number of ways that a trial value from T_β could be the smallest of trial values selected for each algorithm. This counts the number of ways that β could be the best algorithm in A after n executions of the objective function. Using a simple counting technique, we have that the rank 0 probability R_0 of β, can be computed as:

$$R_0(\beta) = \frac{1}{n(T_\beta)} \sum_{v \in T_\beta} \frac{\prod_{\alpha \in A \setminus \{\beta\}} n(\{i \in T_\alpha \text{ s.t. } v \leq i\})}{\prod_{\alpha \in A \setminus \{\beta\}} n(T_\alpha)} \tag{2}$$

Note that the entire computation of rank 0 probability can be done in $alog(a)$ with $a = n_{avg}(T_\alpha)$ [1] for each execution of the objective function. This is not prohibitive with any reasonable number of trials.

Best Solution Probability It could also be interesting to know which optimization algorithms achieved the best solutions and how often each algorithm was the one to achieve the best solution in the benchmark objective function set. We consider an algorithm to have achieved the best possible solution if it is within 1% of the best objective function value obtained by any other algorithm

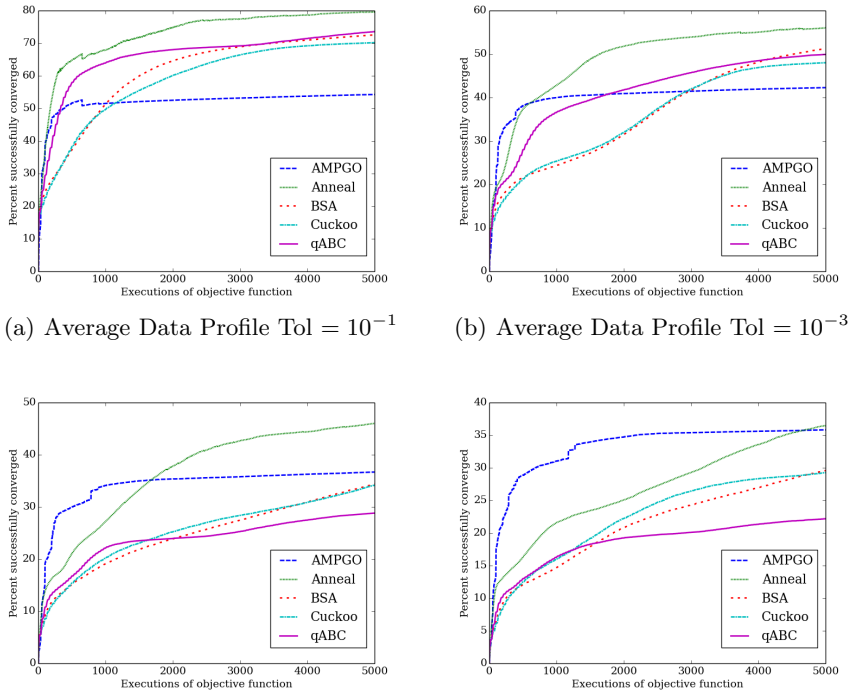
[1] The computation of $n(\{i \in T_\alpha \text{ s.t. } v \leq i\})$ in the upper product sum can be done in $log(n(T_\alpha))$ if each T_α is sorted. n_{avg} represents the average number of trials for all $\alpha \in A$

after the same number of executions of the objective function. For objective function f, the 1% distance, $f_{1\%}$ is defined from the range $max_f - min_f$, where max_f is the maximum value of f that any optimization algorithm achieved and min_f is the minimum. The best solution probability bsp of optimization algorithm α after n executions of each objective function f, in the set of benchmark objective functions F, is:

$$bsp(\alpha) = \frac{n(\{f \in F \text{ s.t. } min(T_\alpha) - min_{f,n} < f_{1\%}\})}{n(F)} \tag{3}$$

where T_α is the set of trials for α after n executions of f and $min_{f,n}$ is the minimum value achieved for f after n iterations by any optimization algorithm.

For verification and reuse, the code for each of these algorithms is available at the following web address implemented in python3 with Numpy and SciPy: *https://github.com/thlux/Convergence_Rate_Evaluation*.



(a) Average Data Profile Tol = 10^{-1}

(b) Average Data Profile Tol = 10^{-3}

(c) Average Data Profile Tol = 10^{-5}

(d) Average Data Profile Tol = 10^{-7}

Fig. 2. The average data profile results across all 50 objective functions

4 Results

The four different tolerance values for data profiles display important nuances related to this set of optimization algorithms. As can be seen in Figure 2, Annealing converges at least weakly more often than any other optimization algorithm, qABC is close behind. As we shift from weak convergence ($\tau = 10^{-1}$) toward stronger convergence ($\tau = 10^{-7}$), the results transition to show that AMPGO is the optimization algorithm most capable of converging quickly. It is clear from the four different tolerance values that Annealing and AMPGO are the two contenders for fastest converging optimization algorithms. Annealing converges more consistently, but not as quickly as AMPGO.

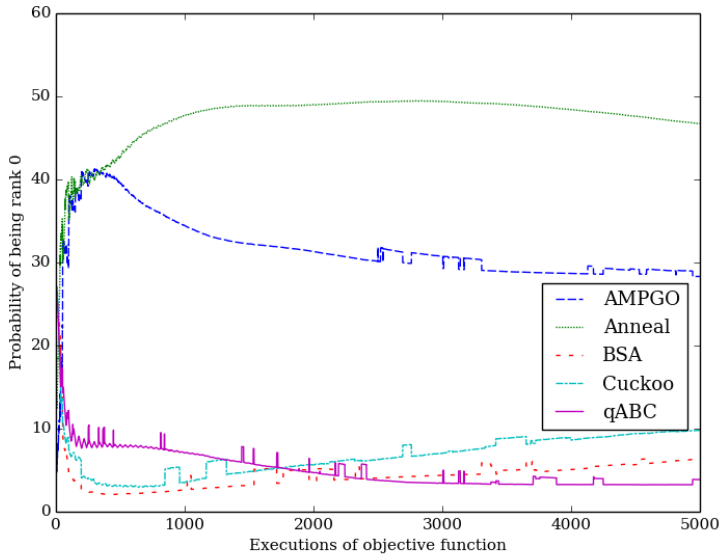


Fig. 3. The average rank 0 probability across all 50 objective functions

The rank 0 probability results in Figure 3 also suggest that Annealing and AMPGO are close competitors. For the first 20 executions of the objective function, the random sampling provided by the populations of qABC, BSA, and Cuckoo Search allow for the fastest convergence. Beyond 20 and until 500 executions of the objective function, it is unclear which of Annealing or AMPGO is a better choice. Given more than 500 executions of the objective function, Annealing is the algorithm most likely to find the best solutions in this set of objective functions. Notably, Annealing only beats AMPGO by a factor of 50%, and would only be more than likely to produce the best solution with more than two trials.

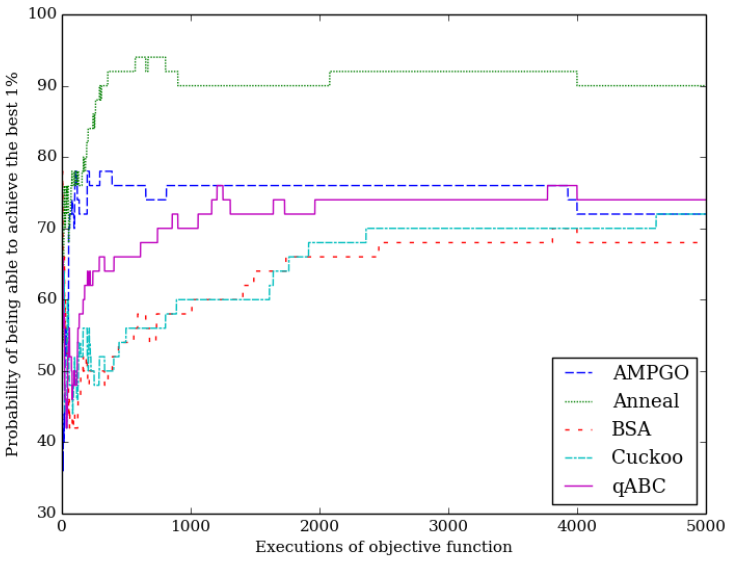


Fig. 4. The average probability of being able to achieve within 1% of the best solution across all 50 objective functions

Lastly, the best solution probability results in Figure 4 suggest that given a large number of repeated trials, all of the algorithms are more than 50% likely to find the equivalent of a "best" solution. Once again, Annealing and AMPGO are the highest performing algorithms for the first 500 executions. The high performance of Annealing in this plot suggests that Annealing does a better job of performing global search though the objective function solution spaces given many trials.

The purpose of this study is to compare the expected convergence rates of each of these algorithms for optimizing complex objective functions. These results suggest that given a mostly unknown complex objective function with bounds and time constraints on optimization, where it is expected that the unknown objective function resembles at least one of the functions in this test set, modified Annealing or AMPGO are most likely to produce optimal results with the fewest executions of the objective function.

5 Future Work

This paper presents a comparison of the convergence rates of five different derivative-free optimization algorithms on a set of 50 objective functions. The goal of this work is to provide insight as to which optimization algorithm will

produce the best results on real-world problems. There are many potential directions this study could take with that goal in mind.

It is necessary to continue introducing new derivative-free optimization algorithms to this test set. In order to encourage the global comparison of new optimization algorithms, the source code for all of this research is readily available in Python. Any new algorithm can be introduced to this comparison by porting existing code to a python function header with an example provided.

This study could be repeated with real-world complex optimization problems to see if the projected results hold under the noise of different objective functions. Finally, the modified Annealing algorithm used in this paper may be improved by the techniques introduced in [13].

6 Conclusion

Comparative study suggests that AMPGO and modified Simulated Annealing converge more reliably on objective functions in our set than do qABC, BSA and Cuckoo search. More optimization algorithms need to be incorporated into this form of comparison in order to expand the search for the fastest-converging general optimization algorithms for bounded derivative-free optimization on difficult-to-compute objective functions.

Appendix

Table 1. These are the mathematical formulations of the functions used to compare the optimization algorithms. For the definitions of the Needle Eye, Penalty02, Rana, and Zero Sum functions please see code provided in electronic supplementary materials.

Name	Definition
Ackley	$-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$
Adjiman	$\cos(x_1)\sin(x_2) - \frac{x_1}{(x_2^2+1)}$
Alpine	$\sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$
Beale	$(x_1x_2 - x_1 + 1.5)^2 + (x_1x_2^2 - x_1 + 2.25)^2 + (x_1x_2^3 - x_1 + 2.625)^2$
Bohachevsky	$\sum_{i=1}^{n-1} \left[x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7\right]$
Cosine Mixture	$-0.1\sum_{i=1}^{n}\cos(5\pi x_i) - \sum_{i=1}^{n} x_i^2$
Deceptive	$-\left[\frac{1}{n}\sum_{i=1}^{n} g_i(x_i)\right]^{\beta}$
Deflected Corrugated Spring	$0.1\sum_{i=1}^{n}\left[(x_i-\alpha)^2 - \cos\left(K\sqrt{\sum_{i=1}^{n}(x_i-\alpha)^2}\right)\right]$
Drop Wave	$-\frac{1+\cos\left(12\sqrt{\sum_{i=1}^{n} x_i^2}\right)}{2+0.5\sum_{i=1}^{n} x_i^2}$
Easom	$a - \frac{a}{e^{b\sqrt{\frac{c}{n}}}} + e - e^{\frac{d}{n}}, c = \sum_{i=1}^{n} x_i^2, d = \sum_{i=1}^{n}\cos(cx_i)$

Egg Holder	$-x_1 \sin\left(\sqrt{\lvert x_1 - x_2 - 47\rvert}\right) - (x_2 + 47)\sin\left(\sqrt{\lvert \frac{1}{2}x_1 + x_2 + 47\rvert}\right)$
Exponential	$-e^{-0.5\sum_{i=1}^{n} x_i^2}$
Giunta	$0.6 + \sum_{i=1}^{n}\left[\sin^2\left(1 - \frac{16}{15}x_i\right) - \frac{1}{50}\sin\left(4 - \frac{64}{15}x_i\right) - \sin\left(1 - \frac{16}{15}x_i\right)\right]$
Goldstein Price	$\left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] +$ $\left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$
Griewank	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Katsuura	$\prod_{i=0}^{n-1}\left[1 + (i+1)\sum_{k=1}^{d}\lfloor(2^k x_i)\rfloor 2^{-k}\right]$
Langermann	$-\sum_{i=1}^{5}\frac{c_i \cos\left\{\pi\left[(x_1-a_i)^2+(x_2-b_i)^2\right]\right\}}{e^{\frac{(x_1-a_i)^2+(x_2-b_i)^2}{\pi}}}$
Levy	$\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2$
Michalewicz	$-\sum_{i=1}^{2}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$
Miele Cantrell	$(e^{-x_1} - x_2)^4 + 100(x_2 - x_3)^6 + \tan^4(x_3 - x_4) + x_1^8$
Mishra01	$(1 + x_n)^{x_n} \quad ; \quad x_n = n - \sum_{i=1}^{n-1}\frac{(x_i + x_{i+1})}{2}$
Mishra02	$\left[\frac{1}{n}\sum_{i=1}^{n}\lvert x_i\rvert - \left(\prod_{i=1}^{n}\lvert x_i\rvert\right)^{\frac{1}{n}}\right]^2$
Multi Modal	$-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$
Odd Square	$-e^{-\frac{d}{2\pi}}\cos(\pi d)\left(1 + \frac{0.02h}{d+0.01}\right)$
Pathological	$\sum_{i=1}^{n-1}\frac{\sin^2\left(\sqrt{100x_{i+1}^2 + x_i^2}\right) - 0.5}{0.001\left(x_i - x_{i+1}\right)^4 + 0.50}$
Penalty01	$\frac{\pi}{30}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} +$ $\sum_{i=1}^{n} u(x_i, 10, 100, 4)$
Qing	$\sum_{i=1}^{n}(x_i^2 - i)^2$
Quintic	$\sum_{i=1}^{n}\left\lvert x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4\right\rvert$
Rastrigin	$10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$
Ripple	$\sum_{i=1}^{2} -e^{-2\log 2\left(\frac{x_i - 0.1}{0.8}\right)^2}\left[\sin^6(5\pi x_i) + 0.1\cos^2(500\pi x_i)\right]$
Rosenbrock	$\sum_{i=1}^{n-1}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$
Paviani	$\sum_{i=1}^{10}\left[\log^2(10 - x_i) + \log^2(x_i - 2)\right] - \left(\prod_{i=1}^{10} x_i^{10}\right)^{0.2}$
Plateau	$30 + \sum_{i=1}^{n}\lfloor x_i\rfloor$
Salomon	$1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$
Sargan	$\sum_{i=1}^{n} n\left(x_i^2 + 0.4\sum_{i \neq j}^{n} x_i x_j\right)$
Schwefel01	$418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i\rvert})$
Schwefel02	$\sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$
Shubert	$\left(\sum_{i=1}^{5} i\cos[(i+1)x_1 + i]\right)\left(\sum_{i=1}^{5} i\cos[(i+1)x_2 + i]\right)$
Sine Envelope	$-\sum_{i=1}^{n-1}\left[\frac{\sin^2(\sqrt{x_{i+1}^2 + x_i^2} - 0.5)}{(0.001(x_{i+1}^2 + x_i^2) + 1)^2} + 0.5\right]$
Six Hump Camel	$4x_1^2 + x_1x_2 - 4x_2^2 - 2.1x_1^4 + 4x_2^4 + \frac{1}{3}x_1^6$
Trigonometric	$1 + \sum_{i=1}^{n} 8\sin^2\left[7(x_i - 0.9)^2\right] + 6\sin^2\left[14(x_i - 0.9)^2\right] + (x_i - 0.9)^2$
Ursem Waves	$-0.9x_1^2 + (x_2^2 - 4.5x_2^2)x_1x_2 + 4.7\cos\left[2x_1 - x_2^2(2 + x_1)\right]\sin(2.5\pi x_1)$

Vincent	$-\sum_{i=1}^{n} \sin(10 \log(x))$
Wavy	$1 - \frac{1}{n} \sum_{i=1}^{n} \cos(k x_i) e^{-\frac{x_i^2}{2}}$
Weierstrass	$\sum_{i=1}^{n} \left[\sum_{k=0}^{kmax} a^k \cos\left(2\pi b^k (x_i + 0.5)\right) - n \sum_{k=0}^{kmax} a^k \cos(\pi b^k) \right]$
Whitley	$\sum_{i=1}^{n} \sum_{j=1}^{n} \left[\frac{(100(x_i^2 - x_j)^2 + (1 - x_j)^2)^2}{4000} - \cos(100(x_i^2 - x_j)^2 + (1 - x_j)^2) + 1 \right]$

Table 2. Objective functions used for evaluating the five optimization algorithms.

Name	**Dimensions**	**Bounds**	**Name**	**Dimensions**	**Bounds**
Adjiman	2	$[-1, 2]_x$ $[-1, 1]_y$	Beale	2	[-4.5, 4.5]
Egg Holder	2	[-512, 512]	Goldstein Price	2	[-2, 2]
Langermann	2	[0,10]	Shubert	2	[-10,10]
Six Hump Camel	2	[-5, 5]	Ursem Waves	2	$[-0.9, 1.2]_x$ $[-1.2, 1.2]_y$
Drop Wave	4	[-5.12, 5.12]	Whitley	4	[-10.24, 10.24]
Miele Cantrell	4	[-1, 1]	Weierstrass	4	[-0.5, 0.5]
Rastrigin	4	[-5.12,5.12]	Katsuura	4	[0,10]
Salomon	4	[-100, 100]	Deceptive	8	[0,1]
Giunta	8	[-1, 1]	Griewank	8	[-600, 600]
Trigonometric	8	[-500, 500]	Paviani	8	[2.001,9.999]
Sargan	8	[-100, 100]	Zero Sum	8	[-10, 10]
Plateau	16	[-5.12, 5.12]	Michalewicz	16	[0,π]
Mishra11	16	[-10, 10]	Odd Square	16	[-5*pi, 5*pi]
Qing	16	[-500, 500]	Rosenbrock	16	[-5,10]
Alpine01	16	[-10, 10]	Bohachevsky	24	[-15, 15]
Easom	24	[-100,100]	Levy03	24	[-10, 10]
Multi Modal	24	[-32, 32]	Penalty02	24	[-50, 50]
Quintic	24	[-10, 10]	Vincent	24	[0.25,10]
Ackley	48	[-32,32]	Cosine Mixture	48	[-1, 1]
Wavy	48	[-pi, pi]	Pathological	48	[-100, 100]
Schwefel22	48	[-100, 100]	Deflected Corrugated Spring	96	[0,2*alpha]
Mishra02	96	[0,1 + 1e-9]	Penalty01	96	[-50, 50]
Exponential	96	[-1, 1]	Ripple01	96	[0,1]
Schwefel	96	[-512,512]	Sine Envelope	96	[-100, 100]

Acknowledgments This research project was funded by the Roanoke College Mathematics Computer Science and Physics Department. The python code for AMPGO and the benchmarking library were a result of the freely available work done by Andrea Gavana at [9].

References

1. Kirkpatrick, S.: Optimization by simulated annealing: Quantitative studies. Journal of statistical physics **34**(5-6) (1984) 975–986
2. Lasdon, L., Duarte, A., Glover, F., Laguna, M., Martí, R.: Adaptive memory programming for constrained global optimization. Computers & Operations Research **37**(8) (2010) 1500–1509
3. Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, IEEE (2009) 210–214
4. Civicioglu, P.: Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and Computation **219**(15) (2013) 8121–8144
5. Karaboga, D., Gorkemli, B.: A quick artificial bee colony (qabc) algorithm and its performance on optimization problems. Applied Soft Computing **23** (2014) 227–238
6. Civicioglu, P., Besdok, E.: A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artificial Intelligence Review **39**(4) (2013) 315–346
7. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization **20**(1) (2009) 172–191
8. Floudas, C.A., Pardalos, P.M.: Encyclopedia of optimization. Springer Science & Business Media (2009)
9. Gavana, A.: Global optimization benchmarks and ampgo (2014) [Online; accessed 2016-04].
10. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of global optimization **39**(3) (2007) 459–471
11. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization **56**(3) (2013) 1247–1293
12. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical programming **91**(2) (2002) 201–213
13. Alizamir, S., Pardalos, P.M., Rebennack, S.: Improving the neighborhood selection strategy in simulated annealing using the optimal stopping problem. INTECH Open Access Publisher (2008)