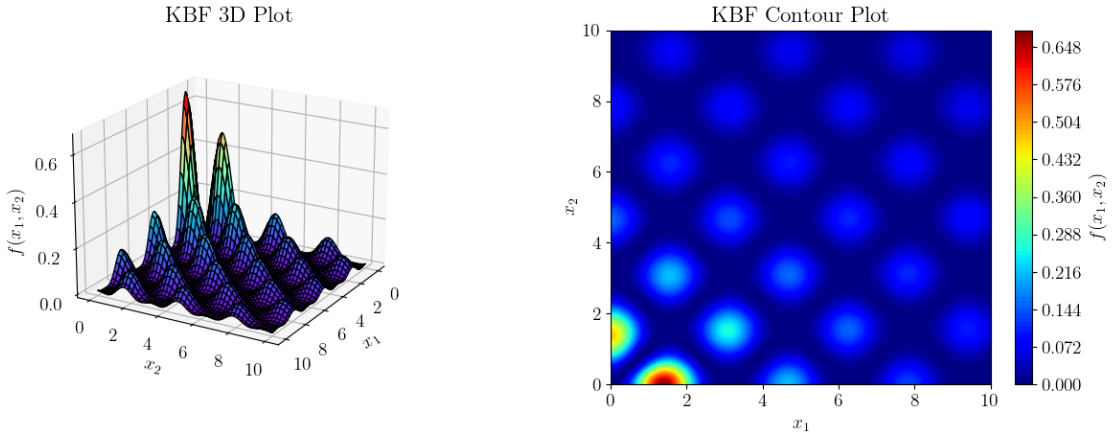
4M17 Coursework #2
Optimisation Algorithm Performance Comparison

Candidate No: 5730E

December 11, 2023

1 Abstract

This report conducts a comparative analysis of two optimisation algorithms applied to minimise Keane's Bump Function, (KBF). In particular, the study focuses on a Continuous Genetic Algorithm, (CGA), as well as an alternative algorithm not covered in the lectures: Q-learning, (QEG). Specifically, the Q-learning approach is adapted with the epsilon-greedy strategy to introduce a level of stochasticity into the optimisation process, thus aligning it with the requirements of this assignment.



(a) Surface plot.

(b) Contour plot.

Figure 1: The two-dimensional visualisation of the Keane's Bump Function, (KBF).

2 Introduction

2.1 Keane's Bump Function

To compare the performances of the two algorithms, the Keane's Bump Function, (KBF), is used as the objective function. In particular, the n-dimensional constrained optimisation problem is defined as the maximisation of:

$$f(\mathbf{x}) = \left| \frac{\sum_{i=1}^{n} (\cos(x_i))^4 - 2\prod_{i=1}^{n} (\cos(x_i))^2}{\sqrt{\sum_{i=1}^{n} i \cdot x_i^2}} \right| \quad (1)$$

$$\begin{aligned} \text{subject to} \quad & 0 \leq x_i \leq 10 \quad \forall i \in \{1, \ldots, n\} \\ & \prod_{i=1}^{n} x_i > 0.75 \\ & \sum_{i=1}^{n} x_i < \frac{15n}{2} \end{aligned} \quad (2)$$

The two-dimensional form of the function has been plotted in Figure 1. Some notable properties are as follows:

- The function is undefined at the origin, (0, 0). This is due to the division by zero in the denominator of Equation 1. Otherwise, the function is continuous and differentiable everywhere.
- The function is highly multi-modal. Its global maximum is located on the constaint boundary $x_n = 0$, where x_n denotes the final variable in the n-dimensional space. However, there are many local maxima located inside the feasible region, all of which have quite similar amplitudes.
- The function is nearly symmetric about the line $x_1 = x_2$. This stems from its construction in 1, using the sums of squared, symmetric terms, x_i^2, $(\cos(x_i))^2$, and $(\cos(x_i))^4$. This results in some invariance regarding the order of the input variables. Overall, the peaks consistently manifest in pairs, yet there is a notable pattern wherein one peak always surpasses its counterpart in magnitude.

Given the above properties, the KBF is a challenging function to optimise. The presence of multiple, similar-amplitude local maxima makes it difficult for an optimisation algorithm to converge to the global maximum. On the other hand, all control variables share the same nature, (continuous variables), and exhibit identical scales. Additionally, all constraints are of the inequality type, and the feasible space is non-disjoint.

The problem becomes more complicated with the inclusion of the constraints outlined in 2. Figure 2 illustrates the resulting feasible region carved out of the original function space. Notably, the contstraint boundaries are non-linear, and the feasible region is non-disjoint. The problem complexity is additionally excaberated by the presence of multiple optima along the constraint boundaries, including the global maxima that we seek to identify.

These properties make the KBF a suitable candidate for the comparative analysis of the two optimisation algorithms, as discussed in the previous work [1].

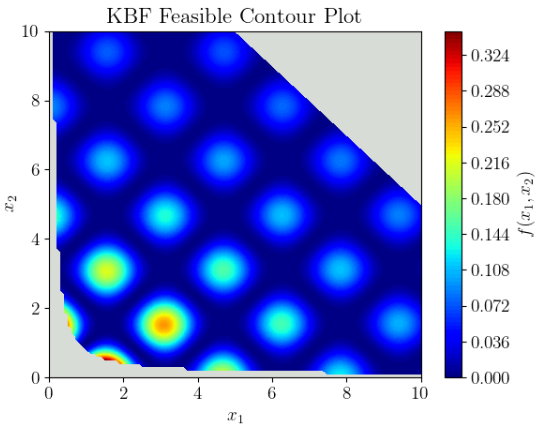


Figure 2: The three-dimensional visualisation of the Keane's Bump Function, (KBF).

2.2 Continuous Genetic Algorithm

The discrete nature of the GA presented in [5] makes it unsuitable for the optimisation of the KBF. An implementation of a Continuous Genetic Algorithm, (CGA), is used instead, which lends itself better to the problems presented in 1-2.

The CGA, a technique inspired by natural selection and genetics, presents itself as particularly well-suited to tackling challenges associated with multiple local optima. Furthermore, the algorithm lends itself well to parallelisation with low implementation effort, and offers ample opportunities for modifications and adaptations, supported by a rich body of literature on the subject.

The primary difference between the CGA and the GA

in [5] is the representation of individuals, (solutions of the state space), within the population. Rather than representing an individual as a vector of binary values or bits, (0s and 1s), the CGA uses a real-valued vector of floating-point numbers to represent each individual, as discussed in [2]. This allows for a direct representation of the problem, and eliminates the need for a decoding function, which reduces overhead in function evaluations.

This adjustment marks a significant departure from conventional GAs, aligning the algorithm more closely with Evolution Strategies (ES), another member of the evolutionary algorithms family presented in [6]. However, the algorithm presented in 2.2.1 is still classified as a GA in accordance with the differences presented in [3], given that mutation does not serve as the primary search mechanism for exploring the state space. Instead, it functions as a non-adaptive, background operator.

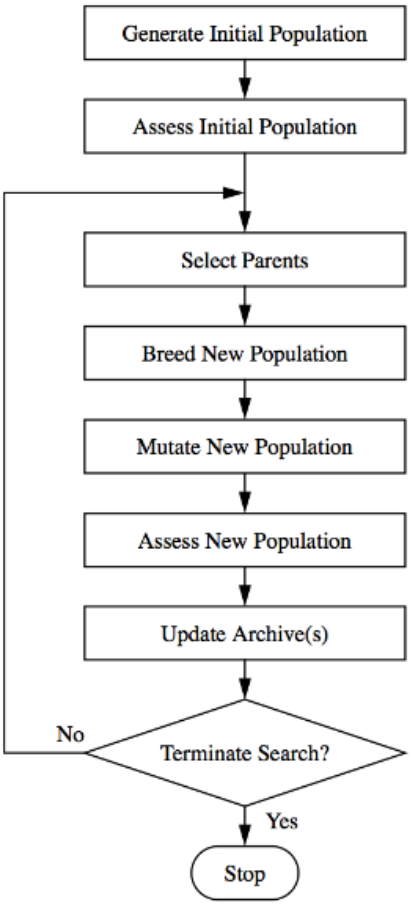


Figure 3: A flowchart depicting the CGA process, taken from [5].

2.2.1 Implementation

In accordance with [5], a vector solution of the state space will be referred to as an *individual* or *chromosome*, and a matrix of individuals will be referred to as a *population*. Each individual is represented as a $n \times 1$ vector of real-valued (floating point) numbers, where n denotes the number of variables in the state space. The population is represented as a $m \times n$ matrix, where m denotes the number of individuals in the population, and is dictated as a hyperparameter within the code:

In accordance with the terminology presented in [5], a vector solution of the state space will be referred to as an *individual* or *chromosome*. Correspondingly, a collection of such individuals arranged in a matrix format will be denoted as a *population*. Each individual is delineated as an $n \times 1$ vector of real-valued (floating-point) numbers, where n signifies the number of variables in the state space. The population itself is represented as a $m \times n$ matrix, where m designates the count of individuals in the population. This count is explicitly defined as a hyperparameter within the codebase:

The CGA process is outlined in Figure 3.

Initialisation

The population is initialised with random values uniformly distributed within the bounds of the state space.

Parent Selection

Three selection methods were implemented: proportional selection, tournament selection, and stochastic remainder selection without replacement (SRS). These are outlined and compared in Section 2.2.2.

Three strategies were deliberately implemented to harness the flexibility inherent in the CGA, tailoring it to the optimisation challenges posed by the KBF. The subsequent section, 2.2.2, elucidates the selection method and mutation procedure choices that form the basis of the forthcoming comparison.

Mating Procedure

Similarly, two mating procedures were implemented: crossover and blending. These are also outlined and compared in Section 2.2.2.

Mutation

The mutation procedure proposed by [2] involves perturbing a gene within a chromosome by a normally distributed random number with a mean of zero and a standard deviation of σ. However, the

effectiveness of this procedure is constrained by the selection of σ as an additional hyperparameter, necessitating careful tuning.

Instead, a simpler approach was adopted, where genes within the population have an probability, given by the mutation rate, of being reset to a random value drawn from a uniform distribution within the confines of the state space, mirroring the initialisation procedure.

Evaluate

The population is then evaluated, and the individuals are ranked in order of fitness. Here, the fitness is defined as the negative of the KBF cost function, outlined in 1. This is done to align the algorithm with the maximisation of the KBF.

Termination

For this section of the report, the algorithm terminates after a maximum number of iterations, (defined as a hyperparameter). A more stringent convergence criterion is adopted for the comparison between the two algorithms in Section 3.

2.2.2 Choosing the Selection Method and Mutation Procedure

Given the flexibility of the CGA, multiple selection methods and mutation procedures were implemented within the codebase. These are then selected as hyperparameters within the script, to take advantage of the flexibility offered by the CGA. In particular, the following selection methods were implemented:

Proportional Selection

The probability of an individual being selected for mating is proportional to its fitness:

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

Here, f_i denotes the fitness of the ith individual. This is the simplest selection method, and is implemented in accordance with the theory presented in [5].

Tournament Selection

As outlined in [5], this strategy involves taking a small subset of the population, and selecting the top two individual with the highest fitness. Selection pressure can then be adjusted by varying the size of the subset, controlled by the *TOURNAMENT_SIZE* hyperparameter.

This is a popular selection method, as it is simple to implement, and is known to perform well in practice. It can be improved by including some of the concepts presented in [4], however, these have been avoided considering [5] suggest SRS as a superior alternative.

Stochastic Remainder Selection without Replacement (SRS)

This strategy draws inspiration from [5]. Specifically, a group of chosen individuals is curated by generating an expected number of copies for each individual, denoted as:

$$E_i = N * p_i$$

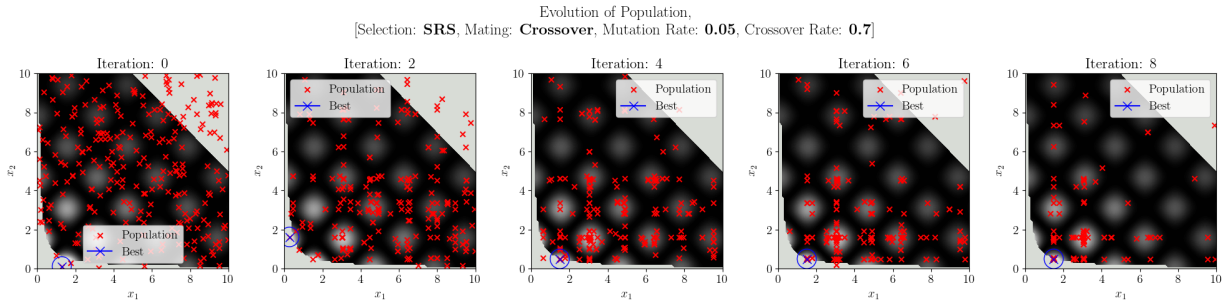
Here, N represents the population size, and p_i is elucidated above in the context of proportional selection. The anticipated number of duplicates is subsequently divided into an integer part, $I_i = \lfloor E_i \rfloor$, and a remainder, $R_i = E_i - I_i$.

The integer part is used for deterministic selection of individuals, while the remainder is employed for stochastic selection. The two parents are then randomly sampled from this augmented collection.

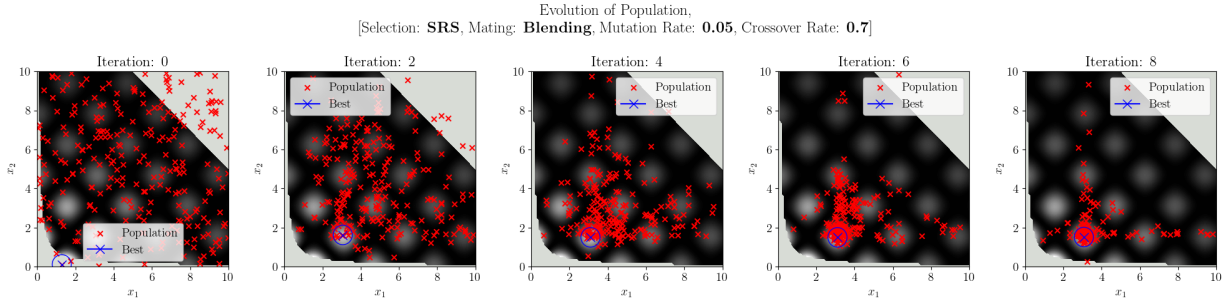
The discussion in [5] highlights that this approach appears to yield superior performance, ascribed to the inclusion of a degree of determinism in the selection criteria. Nevertheless, it remains worthwhile to evaluate the performance of the other two methods, as they might present distinct advantages within the framework of the KBF.

Selection Method	Mating Procedure	Iterations	Final Avg. Fitness	Final Min. Fitness
Proportional	Crossover	10	-0.2218	-0.2875
		100	-0.2385	-0.2835
	Blending	10	-0.2277	-0.2629
		100	-0.2341	-0.2628
Tournament	Crossover	10	-0.0215	-0.2157
		100	-0.0100	-0.2451
	Blending	10	-0.0516	-0.1818
		100	-0.0593	-0.1979
SRS	Crossover	10	-0.2197	-0.4333
		100	-0.2437	-0.2815
	Blending	10	-0.2295	-0.2629
		100	-0.2360	-0.2625

Table 1: Your Table Caption Here



(a) Mutation Procedure: Crossover



(b) Mutation Procedure: Blending

Figure 4: Evolution of the population over 10 iterations using Stochastic Remainder Selection without Replacement (SRS). SRS was found to be the most effective selection method, when compared to Proportional Selection and Tournament Selection, as seen in Figures 5 and 6, respectively.

3 Methodology

4 Results

5 Discussion

6 Conclusion

References

[1] M.A. El-Beltagy and A.J. Keane. A comparison of various optimization algorithms on a multilevel problem. *Engineering Applications of Artificial Intelligence*, 12(5):639–654, 1999.

[2] Randy L. Haupt, S. E. Haupt, and Randy L. Haupt. *Practical Genetic Algorithms*, chapter 3: The Continuous Genetic Algorithm, pages 51–66. John Wiley & Sons, Ltd, 2003.

[3] Frank Hoffmeister and Thomas Bäck. Genetic algorithms and evolution strategies: Similarities and differences. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 455–469, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

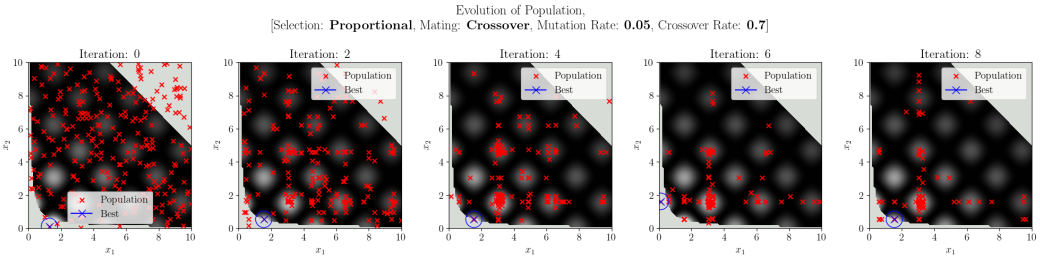
[4] Brad L. Miller and David E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, 9, 1995.

[5] Geoff Parks. Genetic algorithms: Lecture notes. Cambridge University, 4M17 Practical Optimisation Module, 2023. Unpublished.

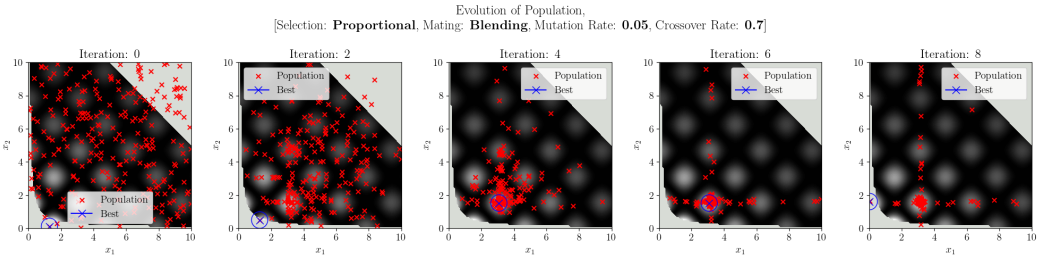
[6] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

7 Appendix

7.1 Supplementary Figures

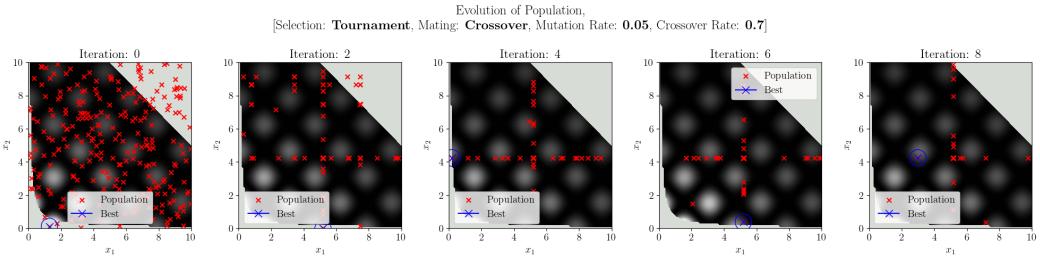


(a) Mutation Procedure: Crossover

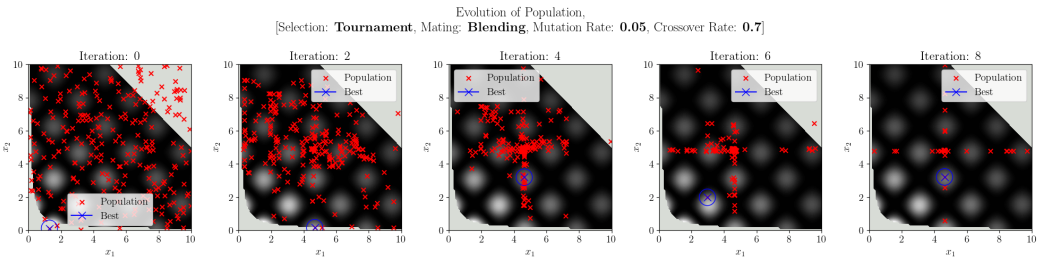


(b) Mutation Procedure: Blending

Figure 5: CGA evolution of the population over 10 iterations using proportional selection. Proportional selection proved to be an effective selection method. However, it was not chosen over SRS as the primary selection method for the reasons outlined in Section 2.2.2.



(a) Mutation Procedure: Crossover



(b) Mutation Procedure: Blending

Figure 6: CGA evolution of the population over 10 iterations using tournament selection. Using tournament selection, the algorithm was unable to converge to a solution. Even 100 iterations produced a suboptimal solution, showcasing convergences to local optima, rather than the global optimum. As such, it was disregarded as a viable selection method.