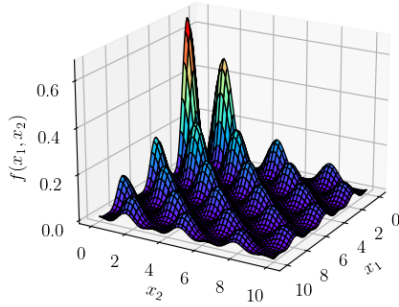# 4M17 Coursework #2
Optimisation Algorithm Performance Comparison
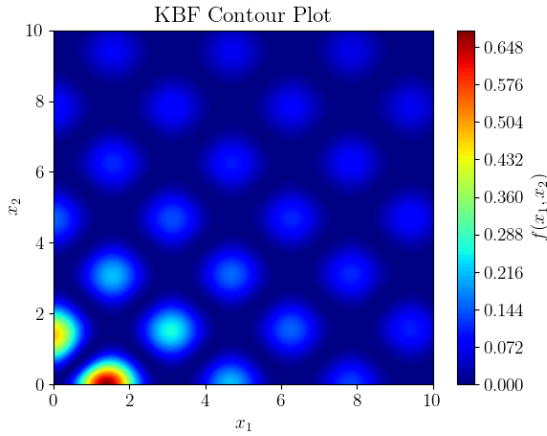
Candidate No: 5730E

December 15, 2023

KBF 3D Plot

(a) Surface plot.

KBF Contour Plot

(b) Contour plot.

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

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: Parallel Tempering, (PT).

2 Problem Setup: 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:

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

$$\text{subject to} \quad 0 \leq x_i \leq 10 \quad \forall i \in \{1, \ldots, n\}$$

$$\prod_{i=1}^{n} x_i > 0.75$$
$$\sum_{i=1}^{n} x_i < \frac{15n}{2} \quad (2)$$

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 constraint 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.

1

- 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. 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 [3].
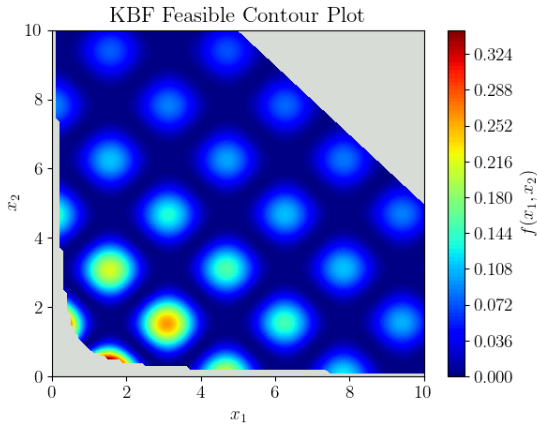

Figure 2: Feasible region carved out from the two-dimensional visualisation of the Keane's Bump Function, (KBF).

3 The Continuous Genetic Algorithm (CGA)

The discrete nature of the GA presented in [8] 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 [8] 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 [4]. 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 [11]. However, the algorithm presented in 3.1 is still classified as a GA in accordance with the differences presented in [5], 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.

3.1 Implementation

In accordance with the terminology presented in [8], 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:
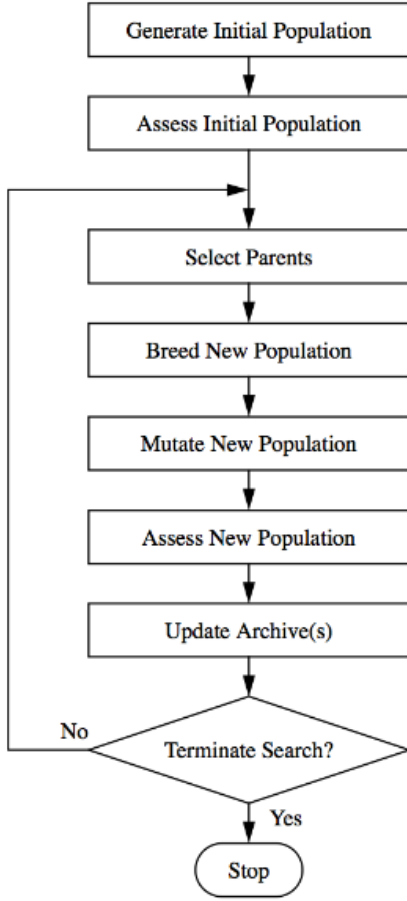
Figure 3: A flowchart depicting the CGA process, taken from [8].

The CGA process is outlined in Figure 3. Notably, three selection strategies and two mutation procedures were deliberately implemented to harness the flexibility inherent in the CGA, tailoring it to the optimisation challenges posed by the KBF. The subsequent section, 3.2, elucidates the selection method and mutation procedure choices that form the basis of the forthcoming comparison.

The CGA can be can be finely tuned for a specific objective function by modifying its fitness function, which assesses the quality of an individual. The management of constraints is woven into the selection process, as outlined below.

3.1.1   Initialisation

The population is initialised with random values uniformly distributed within the range of 0 to 10, which represents the bounds of the state space, encompassing both feasible and infeasible values. This popu-

lation comprises 250 individuals, as defined by the *POPULATION_SIZE* hyperparameter.

3.1.2   Parent Selection

Three selection methods were implemented: proportional selection, tournament selection, and stochastic remainder selection without replacement (SRS). The hyperparameter governing the quantity of parents chosen, denoted as *NUM_PARENTS*, is established at 25% of the population size, (rounded down to the nearest even number to facilitate the creation of parent pairs).

Another important aspect of the selection process is the handling of constraints. In particular, the selection process is repeated until only feasible individuals are selected. Infeasible individuals are simply not chosen as parents, a strategy advised by [8], which was deemed suitable considering that the feasible space is non-disjoint, and the constraints are of the inequality type.

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 [8].

As noted in [8], this approach is susceptible to high variance in individual selection, primarily because there is no assurance of choosing the optimal individual. Alternatively, the following two procedures show greater potential, incorporating a degree of determinism into the selection process.

However, it would be premature to disregard the proportional selection method. There is a chance that the determinism inherent in the other two methods could negatively impact the KBF optimisation process. A notable degree of stochasticity may improve the algorithm's effectiveness in exploring the search space, which may prove vital given the multi-modal nature of the KBF.

Tournament Selection

As outlined in [8], this strategy involves taking a small subset of the population, and selecting the top two individuals with the highest fitness. This is repeated until the required number of parents is achieved. 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 [7], however, these have been avoided to reduce the complexity of the algorithm's implementation.

Stochastic Remainder Selection without Replacement (SRS)

This strategy draws inspiration from [8]. 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. The ith indivdual is selected I_i times. Subsequently, the remained, R_i, is then used to stochastically augment the collection of individuals until the required number of parents is achieved. This is done by selecting the ith individual with a probability of R_i.

The discussion in [8] 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.

3.1.3 Mating Procedure

Similarly, two mating procedures have implemented: crossover and heuristic crossover. The entire population is replaced by offspring, bred from two randomly allocated parents from the pool of selected parents.

Crossover

The crossover procedure adheres to the principles detailed in [8]. Initially, a crossover point is randomly chosen. Genes from the first parent are incorporated into the offspring until this point, beyond which genes from the second parent take their place. Specifically, the sequencing of the parents is governed by the probability specified by the hyperparameter *CROSSOVER_PROB*.

Heuristic Crossover

Heuristic crossover is presented in [6]. By this variation, a random number β in the interval $[0, 1]$ is generated. The genes of the offspring are then determined as a blend of the original two parents, p_1 and p_2, as follows:

$$o_i = \beta(p_{1i} - p_{2i}) + p_{2i}$$

With this inspiration in mind, heuristic crossover was implemented in the CGA. Specifically, the sequence of parents in the above formula is determined by the *CROSSOVER_PROB* hyperparameter, aligning with the approach used previously in the original crossover procedure.

A crucial factor to bear in mind is that certain offspring may be produced outside the feasible region. This implementation deviates from the recommendation in [6] by not outright rejecting these offspring during the mating procedure. Rather, they are simply excluded from consideration as parents in the subsequent selection process.

The decision to incorporate this mutation procedure stems from the continuous nature of the state space. While the conventional crossover methods may be well-suited to binary representations, a more intuitive approach emerges when grappling with real-valued variables - using a blend of characteristics from both parents.

Moreover, the use of the heuristic crossover method aims to address previous limitations associated with standard crossover. Unlike conventional crossover, which confines offspring values to those of the parents, blending permits the generation of offspring beyond the parent values, allowing for potentially new information. This feature may prove particularly advantageous in the context of the KBF, enhancing the algorithm's capability to navigate the search space adeptly and thoroughly explore local optima.

An additional noteworthy observation is that the introduction of β serves to bring the CGA into closer alignment with Evolutionary Strategies (ES). This resemblance becomes evident as the formulation above bears some similarity to the intermediate recombination strategy outlined in [11]. Nevertheless, it is essential to emphasise, as mentioned earlier, that mutation does not serve as the primary search mechanism in the CGA. Consequently, the CGA can still be appropriately categorised as a GA, as per the classification seen in [5].

3.1.4 Mutation

The mutation procedure proposed by [4] 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.

3.1.5 Evaluation

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.

3.1.6 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 5. The individual with the lowest fitness value is then returned as the optimal solution to the optimisation problem.

## 3.2 Tuning the CGA Hyperparameters

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

To choose a selection method and mutation procedure going forwards into the main comparison of this report, code was used as a platform for exploartion. Each selection method and mutation procedure was evaluated over 10 and 100 iterations, with a constant mutation rate of 0.05, and a crossover probability of 0.7. The number of parents was established at 25% of the population size, (rounded down to the nearest even number to facilitate the creation of parent pairs). The population size itself was set at 250, and the tournament size was similarly defined as 25% of this.

Additional results from the initial exploration of hyperparameter tuning are detailed in the Appendix. Specifically, the final fitness values are outlined in Table 1. Further insights into the initial evolution of the population during the first 8 iterations are illustrated in supplementary figures found in Figures 9, 8, and ??.
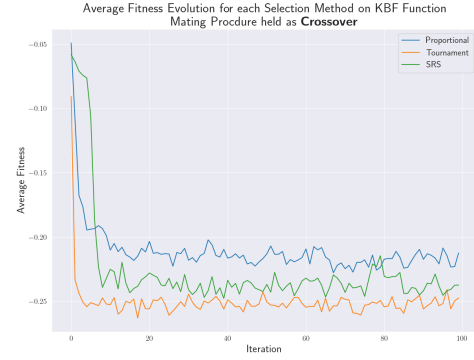
(a) Mutation Procedure: Crossover
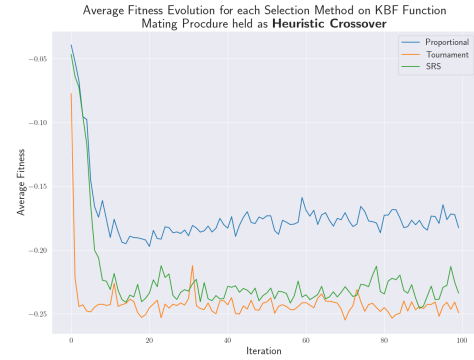
(b) Mutation Procedure: Heuristic Crossover

Figure 4: Evolution of the average fitness values of the CGA population over 100 iterations for each of the selection methods. The results are presented for both mutation procedures: crossover and heuristic crossover. Here, the mutation rate was set to 0.05, and the crossover probability was set to 0.7.

However, the most insightful conclusions can be drawn from the figures presented in 4, which illustrate the evolution of the average fitness values of the CGA population over 100 iterations for each of the selection methods. They illustrate that tournament selection seemed to outperform the other two selection methods when maximising the KBF, which is surprising given the favourable description in [8] regarding SRS.

The experiment was repeated several times, yielding

results that were admittedly variable, attributed to the inherent stochastic nature of the CGA. At times, the superiority of SRS over tournament selection was evident, yet proportional selection generally did not prove to perform better. Despite this variability, the outcomes exhibited sufficient consistency to justify the designation of tournament selection as the primary method for the upcoming comparison.

The observed differences in performance may be attributed to the distinct characteristics of the KBF. Tournament selection appears to demonstrate greater efficacy in navigating the search space compared to SRS, perhaps due to its less deterministic nature. This becomes particularly significant in light of the multi-modal nature of the KBF, where a certain level of stochasticity may prove essential for thorough exploration of the search space.

The findings additionally suggest that the heuristic crossover procedure consistently outperformed the standard crossover method, as evidenced by its more frequent attainment of the global maxima.

This may be attributed to the heuristic crossover's capacity to generate offspring beyond the existing genotypes present within a generation. This attribute is especially beneficial within the framework of the KBF, enhancing the algorithm's ability to navigate the search space adeptly and systematically explore the numerous local optima.

Having cemented the selection method and mutation procedue choices as tournament selection and heuristic crossover respectively, the CGA was further tuned to optimise its performance.

Specifically, choices for the mutation rate and crossover probability were explored. The results are presented in Figure 5, which showcases two heat maps of the final fitness values after 100 iterations for the CGA using the chosen procedures. The plots are presented for both the average final fitness, as well as the final fitness of the best individual.

Notbaly, as depicted in Fig. 5a, it is evident that an increased stochasticity of the CGA, indicated by higher crossover probabilities, leads to improved average final fitness. This aligns with expectations, as a broader and more randomly distributed population enhances the likelihood of individuals being situated in the proximity of local maxima. Consequently, a greater number of individuals within the population receive lower fitness values, a trend reflected in the observed average final fitness.

Fig. 5b offers deeper insights. It is evident that the

crossover probability stands out as the most influential hyperparameter, while the fine-tuning of the mutation rate plays less of a role in shaping the performance of the CGA.
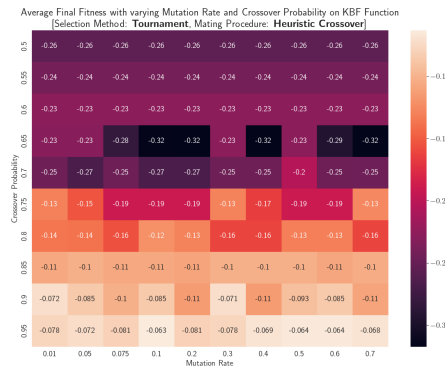
(a) Average final fitness across entire population.
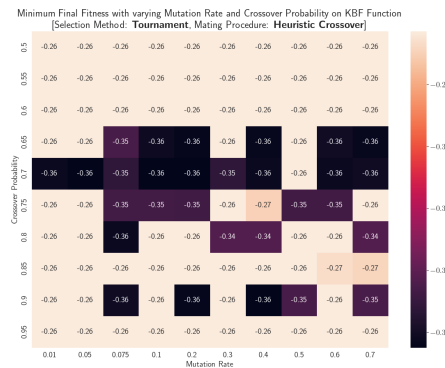
(b) Final (minimum) fitness of best individual.

Figure 5: Heat maps of the final fitness values after 100 iterations for the CGA using tournament selection and heuristic crossover. Here, the mutation rates and crossover probabilities were varied to assess their impact on performance.
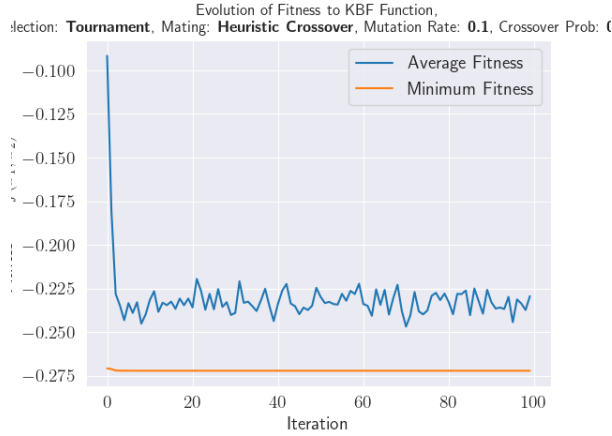
Figure 6: Evolution of the fitness values of the CGA population over 100 iterations for optimally-tuned CGA.

Overall, a clear optimal set of hyperparameters emerges. Moving forward, the mutation rate is fixed at 0.1, and the crossover probability is set to 0.65. These values have proven to yield a consistently optimal performance for the algorithm, comfortably lying within the darker regions of the heat maps.

To reiterate and summarise this section of the report, the CGA is now constructed as follows:

- **Selection Method:** Tournament Selection

- **Mutation Procedure:** Heuristic Crossover

- **Mutation Rate:** 0.1

- **Crossover Probability:** 0.65

- **Tournament Size:** 62

- **Number of Parents:** 62

- **Population Size:** 250

Fig. 6 depicts the evolution of the fitness values over 100 iterations using the optimal hyperparameters delineated above. Note that the (minimum) fitness attributed to the best individual remains constant, because an individual within the population is stochastically initialised near the maxima that the algorithm converges to. This is less likely to occur in 8-dimensional space, where the search space is significantly larger, and the fitness of the best individual is expected to evolve more noticeably.

Fig. 7 depicts the convergence of the population to the second largest maxima over 4 and 80 iterations using the optimally-tuned CGA. Regrettably, global maximum convergence remained elusive even after 100 iterations. However, there does seem to be a discernible trend of the algorithm converging towards the second-largest maxima in Fig. 7, offering a promising indication of progress.

The primary challenge lies in the fact that both the first and second largest maxima are situated along the non-linear constraint boundary of the optimisation problem, posing a difficult navigational challenge.
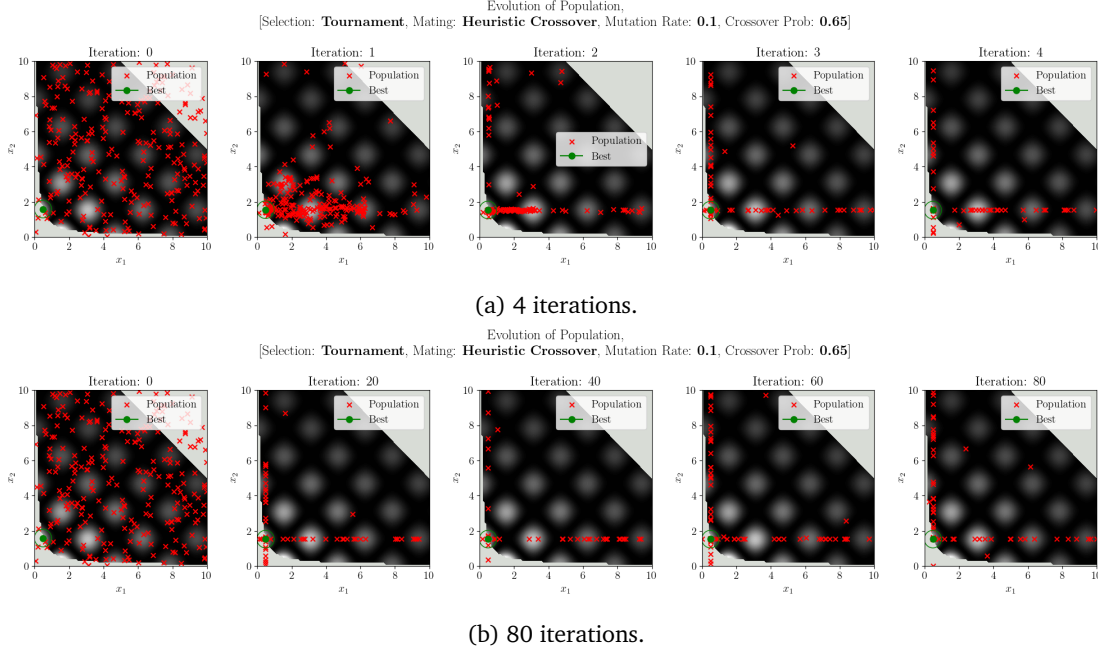
Evolution of Population,
[Selection: **Tournament**, Mating: **Heuristic Crossover**, Mutation Rate: **0.1**, Crossover Prob: **0.65**]

(a) 4 iterations.

Evolution of Population,
[Selection: **Tournament**, Mating: **Heuristic Crossover**, Mutation Rate: **0.1**, Crossover Prob: **0.65**]

(b) 80 iterations.

Figure 7: Evolution of the CGA population over 4 and 80 iterations using the optimal hyperparameters delineated at the end of Section 3.2.

4 Parallel Tempering (PT)

Having obsererved the CGA's performance and its difficulty in identifying global maxima effectively, an informed decision was made to introduce Parallel Tempering (PT) as the second algorithm for comparison.

This is a Markvo Chain Monte Carlo (MCMC) optimisation approach presented in [2]. Similar to simulated annealing, (SA), it revolves around the idea of smoothening the search space by introducing a temperature parameter, t.

By incorporating a Metropolis-Hasting acceptance probability that is proportional to $\exp \frac{1}{t}$, exploration of the solution space is encouraged at high temperatures, while exploitation of the local nature of the function is facilitated at low temperatures. At high temperatures, the acceptance distribution is is smoothened out by this proportionalilty, and the algorithm is then able to explore the search space more thoroughly. As the temperature diminishes, the acceptance distribution becomes more dependent on the fitness of the proposed solution to the KBF, enabling the algorithm to explore the local nature of the function.

However, PT incorporates the novel concept of Replica Exchange Monte Carlo. Unlike SA, which operates along a single tragectory, starting with a high temperature that gradually decreases, PT maintains multiple replicas of the system at different temperatures simultaneously. Each replica explores its own solution space, and exchanges of the solutions can occur between replicas at adjacent temperature levels. This exchange mechanism enhances global exploration by facilitating the movement of solutions between replicas operating at high and low temperature.

This parallel exploration across temperatures improves the algorithm's ability to escape local optima, providing a valuable alternative to traditional single-trajectory methods like SA. Consequently, PT could emerge as a well-suited approach for maximising the highly multi-modal KBF.

Unfortunately, this introduces an elevated computational cost, due to the management of multiple replicas and the periodic exchange of solutions. However, the potential advantages in terms of global exploration of the KBF outweigh the added computational expense. This is especially true, considering the highly parallelisable nature of PT, which may allow for more efficient utilisation of parallel computing resources than the CGA.

4.1 Implementation

PT originates from statistical physics, as extensively discussed in [2]. Therefore the following implemen-

tation draws inspiration from literature focused on sampling complex probability distributions.

The discourse presented in [2] emphasises a delicate trade-off between optimising MCMC sampling outcomes and minimising computational efforts. Hence, the PT implementation presented here is deliberately crafted for flexibility, offering the ability to fine-tune the algorithm, as shown in Section 4.2.

As per the guidance in [9], constraints are handled by simply rejecting any proposed changes that violate the constraints. This is accomplished within the Metropolis-Hastings criterion regarding the acceptance or rejection of a proposed change. This was determined as a suitable approach, given that the feasible space is non-disjoint, and the constraints are of the inequality type.

The approach can additionally be characterised as population-based, incorporating 10 replicas in accordance with the *NUM_REPLICAS* hyperparameter. Within each replica, there are 25 chains of solutions determined by the *NUM_CHAINS* hyperparameter. Consequently, the total number of solutions being evaluated at any given time is 250, which is equivalent to the population size of the CGA.

4.1.1 Initialisation

The chains are initialised within the range of 0 to 1, as advised by the guidance provided in [10], specifically under the SA-derived solution generation framework employed in this implementation.

This range persists throughout the algorithm, until function evaluations are necessitated, at which point the *scale_up* lambda function is called to return the solution in the original state space.

Mirroring the CGA implementation, solutions were initialised without consideration for their feasibility. Whilst initialising within the feasible solution space was observed to enhance performance, this approach was deliberately omitted to ensure fair comparisons between the PT and CGA. However, the performance was notably compromised, given that solutions initialised deep within the infeasible zones of the search space faced challenges in escaping under the Metropolis-Hastings acceptance criterion.

4.1.2 Temperature Scheduling

The primary avenue for flexibility in PT lies in setting and adapting the temperatures of the replicas, as discussed in [2]. Drawing insights from the previous work of [1] on an unrelated, yet similar problem, this was taken advantage of by parameterising the temperature schedule as follows:

$$T_i = \left(\frac{i}{N_{\text{replicas}}} \right)^p \tag{3}$$

Here, T_i denotes the temperature of the ith replica, and N_{replicas} is the total number of replicas. The exponent p is a hyperparameter referred to as *POWER_TERM* that can be tuned to optimise the performance of the algorithm.

4.2 Tuning the PT

Additionally, other sources of flexibility include the manipulation of the number of replicas running simultaneously within the system and the frequency at which solutions are exchanged.

5 Comparison

6 Results

7 Discussion

8 Conclusion

References

[1] Ben Calderhead and Mark Girolami. Estimating bayes factors via thermodynamic integration and population mcmc. *Computational Statistics & Data Analysis*, 53(12):4028–4045, 2009.

[2] David J. Earl and Michael W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910, 2005.

[3] 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.

[4] 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.

[5] 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.

[6] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer, 2011.

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

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

[9] Geoff Parks. Simulated annealing: Lecture notes. Cambridge University, 4M17 Practical Optimisation Module, 2023. Unpublished.

[10] Geoffrey Thomas Parks. An intelligent stochastic optimization routine for nuclear fuel cycle design. *Nuclear Technology*, 89(2):233–246, 1990.

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

9 Appendix

9.1 Supplementary Results regarding CGA Tuning

Selection Method	Mating Procedure	Iterations	Final Avg. Fitness	Final Min. Fitness
Proportional	Crossover	10	-0.19890798	-0.247328018
		100	-0.212384457	-0.24867806
	Heuristic Crossover	10	-0.199296791	-0.258198948
		100	-0.22656183	-0.250977443
Tournament	Crossover	10	-0.309572335	-0.331996331
		100	-0.309235401	-0.342745604
	Heuristic Crossover	10	-0.241261775	-0.262876797
		100	-0.247574635	-0.262876811
SRS	Crossover	10	-0.189986008	-0.208434151
		100	-0.288766718	-0.319667279
	Heuristic Crossover	10	-0.177779432	-0.207148488
		100	-0.182101833	-0.338823558

Table 1: Raw results from an initial exploration of the selection method and mutation procedure hyperparameters within the CGA. Presented as the final fitness values of the CGA population after 10 and 100 iterations. Here, minimum fitness refers to the fitness of the best (feasible) individual within the population. Additionally, the mutation rate was set to 0.05, and the crossover probability was set to 0.7.

(a) Mutation Procedure: Crossover

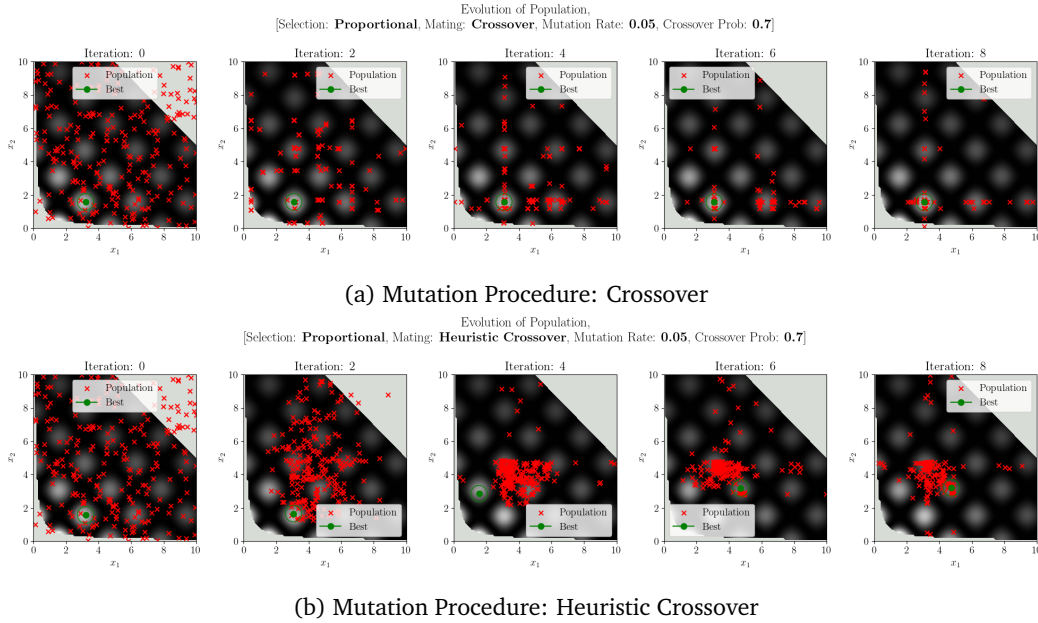(b) Mutation Procedure: Heuristic Crossover

Figure 8: Evolution of the CGA population over 8 iterations using proportional selection. Proportional selection proved to be a somewhat effective selection method. However, it was not chosen over tournament selection as the primary selection method for the reasons outlined in Section 3.2.

(a) Mutation Procedure: Crossover
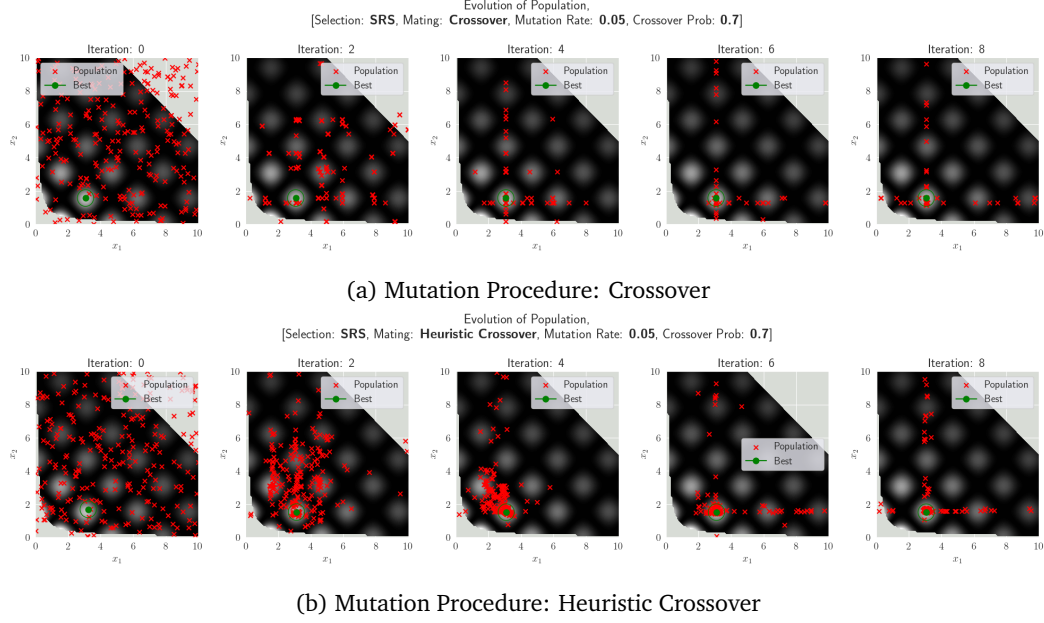

(b) Mutation Procedure: Heuristic Crossover

Figure 9: Evolution of the CGA population over 8 iterations using stochastic remainder selection without replacement (SRS). SRS proved to be the second most effective selection method, when compared to Proportional Selection and Tournament Selection, as discussed in 3.2.

9.2 Supplementary Results regarding PT Tuning

Exchange Procedure	Exchange Parameter	Power Term	Final Avg. Fitness	Final Min. Fitness
Periodic	0.1	1	-0.195696325	-0.271773819
		3	-0.195696325	-0.271773819
		5	-0.190261962	-0.325543757
	0.3	1	-0.182369994	-0.346578069
		3	-0.182369994	-0.346578069
		5	-0.183774281	-0.341275246
	0.5	1	-0.158484931	-0.267351197
		3	-0.158484931	-0.267351197
		5	-0.171493053	-0.341275246
Stochastic	0.1	1	-0.181921769	-0.33399822
		3	-0.181921769	-0.33399822
		5	-0.192349346	-0.33399822
	0.3	1	-0.209998474	-0.347740414
		3	-0.209998474	-0.347740414
		5	-0.202770259	-0.345528224
	0.5	1	-0.206847914	-0.357649754
		3	-0.206847914	-0.357649754
		5	-0.202650445	-0.354243287

Table 2: Raw results from an initial exploration of the exchange procedure, exchange parameter, and power term hyperparameters within PT. Presented as the final fitness values of the PT population after 100 iterations. Here, minimum fitness refers to the fitness of the best (feasible) solution across all chains.
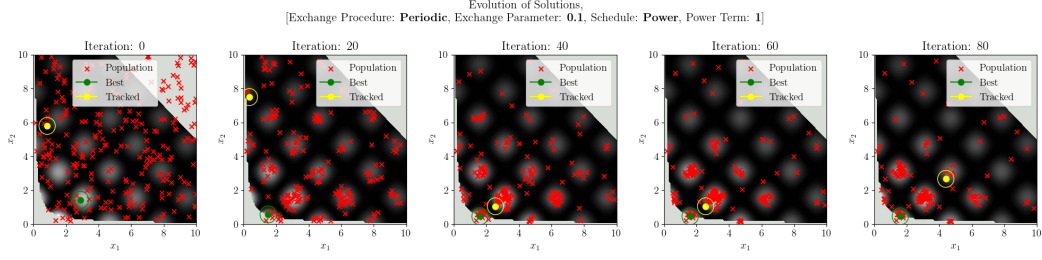
Figure 10: Evolution of the solutions across all chains over 100 iterations using periodic replica exchange and a uniform temperature scheduling, (power term of 1). Here, replica exchange occurs every 10% of the time, (exchange parameter set to 0.1).
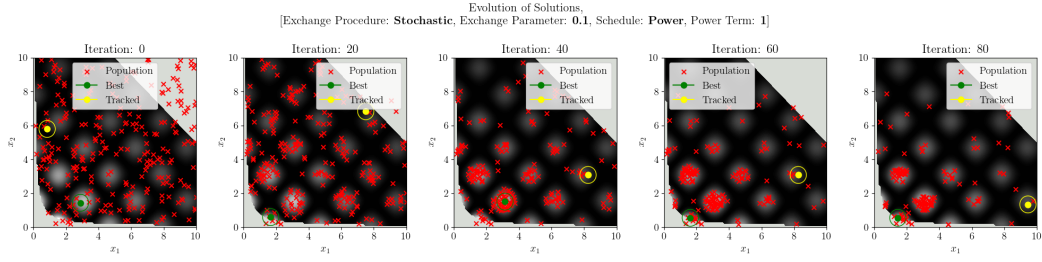

Figure 11: Evolution of the solutions across all chains over 100 iterations using stochastic replica exchange and a uniform temperature scheduling, (power term of 1). Here, replica exchange occurs with a probability of 10%.