1 Black-Box Optimization

This section presents a comparative study of two black-box optimization algorithms: Genetic Algorithm (GA) and Simulated Annealing (SA). The analysis is conducted on two different 3D functions, shown in Figure 1: `@rastriginsfcn`, highly multimodal with numerous local minima, and the smoother `@ps_example`. For GA, the parameter to be optimized is the crossover fraction f_{cross}, defined as the proportion of non-elite individuals in the new generation generated through crossover rather than mutation, favoring exploitation when close to 1 and exploration when close to 0. For SA, the considered parameter is the initial temperature T_0, which defines the initial randomized state of the algorithm: high values promote exploration via uphill moves, low values favor exploitation through refinement of current solutions.

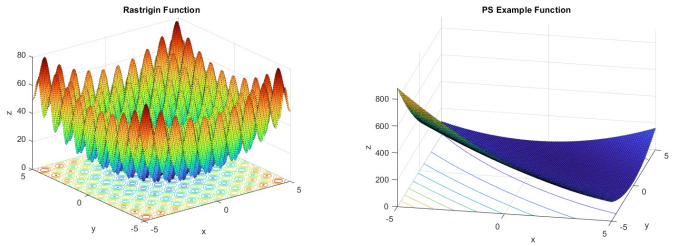

Figure 1: `@rastriginsfcn` (left) and `@ps_example` (right)

Figure 2 shows the population evolution of the Genetic Algorithm with a high crossover fraction, where darker points indicate individuals from later generations. The population tends to concentrate near the minimum from the beginning. Since `@rastriginsfcn` is multimodal, the population is much more spread out, as the algorithm struggles to converge to a single optimum, whereas rapid convergence is more easily achieved for `@ps_example`. With $f_{cross} = 0.2$, as shown in Figure 3, exploration is favored. While `@rastriginsfcn` shows widespread scattering, `@ps_example` still exhibits a significantly higher concentration of individuals near the global minimum, as the population is naturally guided by the convexity of the function.

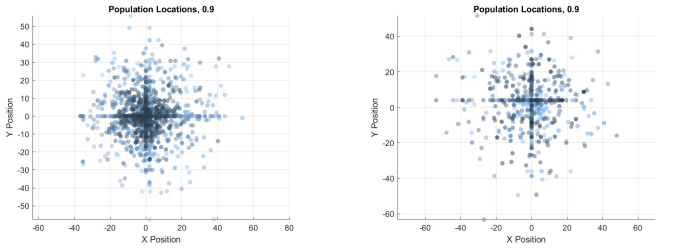

Figure 2: Population locations of `@rastriginsfcn` (left) and `@ps_example` (right) for $f_{cross} = 0.9$

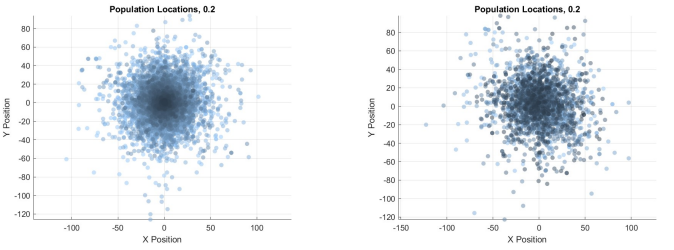

Figure 3: Population locations of `@rastriginsfcn` (left) and `@ps_example` (right) for $f_{cross} = 0.2$

Figures 4 and 5 show that, as the initial temperature increases, Simulated Annealing explores a broader area. However, while `@ps_example` easily brings the algorithm towards the global minimum, with lower and more consistent function values, `@rastriginsfcn` results in a more scattered and variable distribution.

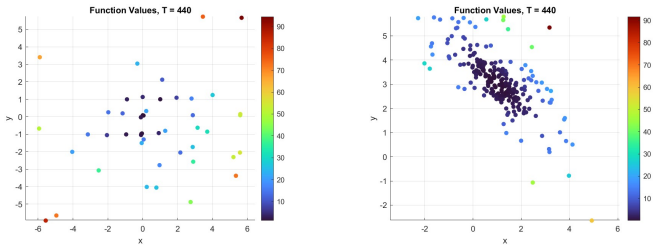

Figure 4: Function values of `@rastriginsfcn` (left) and `@ps_example` (right) for $T_0 = 440$

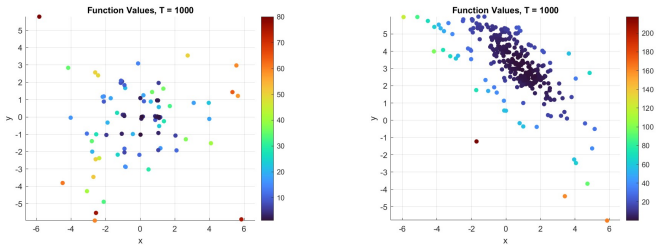

Figure 5: Function values of `@rastriginsfcn` (left) and `@ps_example` (right) for $T_0 = 1000$

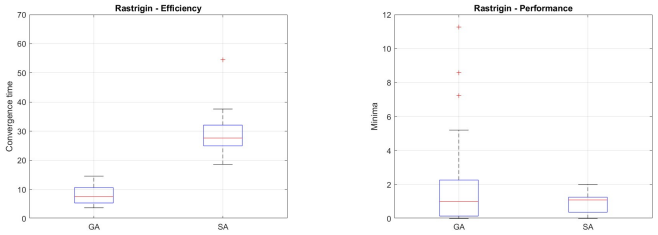

Figure 6: Efficiency and performance of the two algorithms for `@rastriginsfcn`

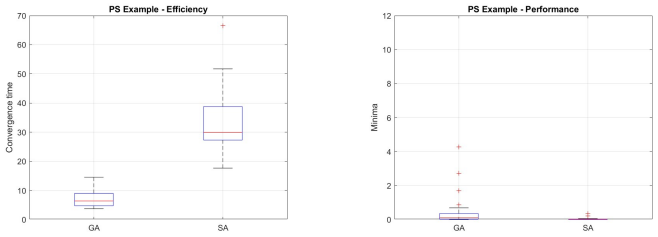

Figure 7: Efficiency and performance of the two algorithms for `@ps_example`

A quantitative analysis of performance (fitness function) and efficiency (convergence time) is presented. Figure 6 compares the two algorithms on `@rastriginsfcn`, showing that *SA* requires significantly more time to converge than *GA*, due to its sequential search strategy as opposed to the population-based approach of *GA*. The performances are comparable, though slightly better for *GA*, as both algorithms

struggle to escape local minima. However, the Genetic Algorithm exhibits greater variability. Figure 7 compares them on `@ps_example`, with *SA* converging more slowly and with greater variability in convergence time, yet achieving better and more stable performance on this simpler function.

Being `@ps_example` unimodal, the performance of both *GA* and *SA* is significantly better compared to the multimodal function. However, note that the boxplots aggregate results across all tested parameters, capturing overall algorithm behavior rather than variability under fixed conditions. Detailed trends are shown instead in Figures 8 and 9.

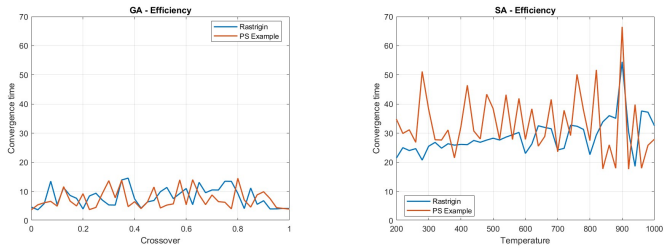

Figure 8: Efficiency of *GA* and *SA* over f_{cross} and T_0

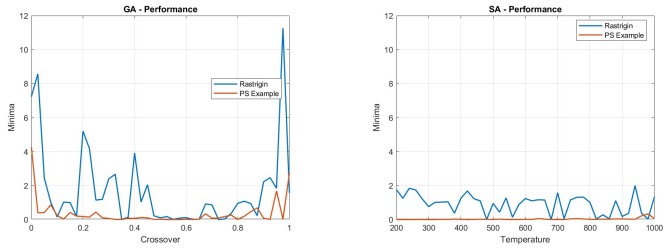

Figure 9: Performance of *GA* and *SA* over f_{cross} and T_0

Convergence time of *GA* does not show a clear monotonic trend across values of the crossover fraction, but presents some peaks and local minima for both functions. In terms of performance, the algorithm is highly sensitive to f_{cross}, particularly for the multimodal `@rastriginsfcn` function, where performance improves sharply between $f_{cross} = 0$ and 0.4, forming a plateau in the interval $[0.4, 0.6]$, before degrading again due to excessive crossover and thus insufficient exploration. For this function, the optimal trade-off is found at $f_{cross} \approx 0.5$, corresponding to a minimum in convergence time and a very low function value. Since the function `@ps_example` is smoother, the algorithm yields outstanding performance over a broader range, with the worst cases occurring at extreme crossover values. The optimal trade-off here is found at $f_{cross} \approx 0.7$, where both low convergence time and very good performance are observed. This reflects that higher crossover is more beneficial in problems where the landscape is convex or quasi-convex, allowing rapid convergence without being misled by (absent) local minima.

In contrast, *SA* shows much worse efficiency for both functions compared to *GA*, and significantly better performance on `@ps_example`, while struggling more to find the global optimum of `@rastriginsfcn`. For the latter, efficiency remains relatively stable across all temperatures but worsens slightly at higher values, as a symptom of over-exploration. On the other hand, performance is poorer below $T_0 \approx 400$, and improves overall with increasing temperature. The best trade-off in this case appears to be at $T_0 \approx 800$, allowing high - but not excessive - exploration. For `@ps_example`, efficiency oscillates considerably, with greater variability at higher temperatures. Performance, by contrast, remains consistently very good, though the minimum value found increases slightly with T_0. The best trade-off appears to be at $T_0 \approx 400$, where the estimated optimum is still very low and convergence time exhibits a local minimum. However, it should be noted that these results are stochastic and heavily influenced by random evaluations.

In conclusion, *GA* achieves better overall efficiency on both `@rastriginsfcn` and `@ps_example`, and, when f_{cross} is appropriately set, ensures better performance on the former than *SA*, which, in turn, performs considerably better on the latter. Since *GA* leverages a population-based approach, it can explore a large space of potential solutions and effectively find the global optimum even for complex multimodal functions. On the other hand, *SA* is highly effective at converging to the global optimum for functions with few local minima, such as `@ps_example`.

2 PCA on data-set

This section presents a PCA analysis of the *Iris* dataset, which contains four features (sepal length, sepal width, petal length, and petal width) for 150 plants belonging to three different species (*setosa*, *versicolor*, *virginica*). The ground truth, with flower classes represented in both the sepal and petal feature spaces, as well as the PCA score plot using the first two principal components, is shown in Figure 10.

Setosa is clearly separated from the others, both in the first and especially in the second plot, while *versicolor* and *virginica* overlap significantly. This separation is well captured by the first principal component, while the second *PC* partly distinguishes the overlapping features.

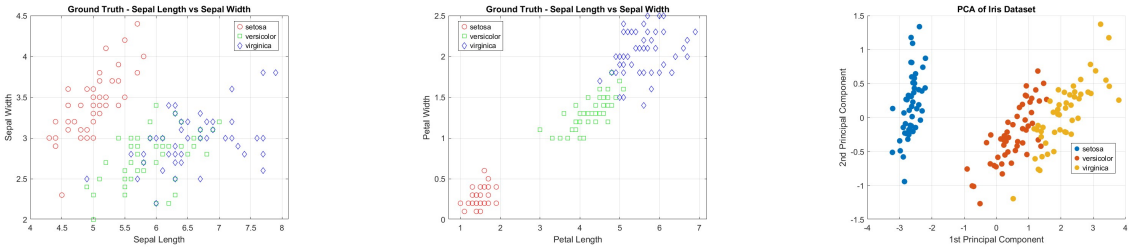

Figure 10: Species distribution by sepal and petal dimensions, and principal components PC_1, PC_2

Figure 11 shows the biplot of *PCA* scores, the variance explained by each component, and the cumulative curve. Vectors in the biplot indicate the projection of the original variables into the reduced space. $PC1$, which captures nearly all variability, is mainly influenced by the petal features, though sepal length also contributes. As aforementioned, it primarily separates *setosa* from the other species. Notably, the petal width and petal length vectors are almost aligned, reflecting the fact that the direction of maximum variability in the second plot of Figure 10 is at 45°. $PC2$, associated with a small portion of the variance, is more related to sepal features.

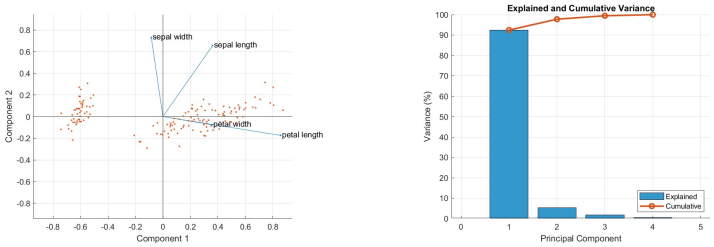

Figure 11: *PCA* scores (biplot) and explained variance

This is evident in the explained variance plot: PC_1 alone accounts for over 90%, and the first two components together explain almost the entire variability of the dataset. Depending on the desired trade-off, one or two components may be retained, although keeping both ensures high reconstruction accuracy with reduced dimensionality.

3 Perform k-means and identify the optimal number of clusters

This section presents a k-means clustering of the *Iris* dataset, based on the two features sepal width and sepal length. K-means is a simple algorithm that groups nearby points according to a predefined number of clusters, which in this case is selected based on the *silhouette* score, a metric that measures how well each point fits within its cluster compared to other clusters.

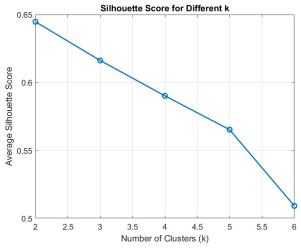

Figure 12: Silhouette scores for different k

Figure 12 plots silhouette scores as a function of the number of clusters: this metric decreases with increasing k, but $k = 3$ (the ground truth value) yields a result very close to $k = 2$. As a consequence, k-means is performed in both cases and the results are compared.

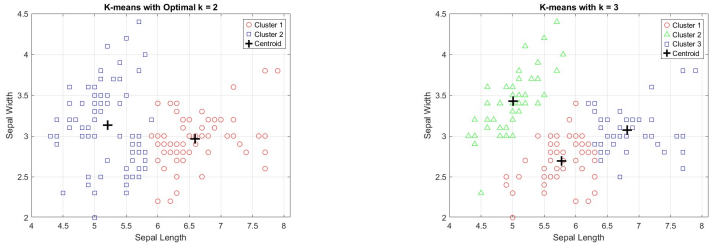

Figure 13: k-means clustering results for $k = 2$ and $k = 3$

Clustering results, along with centroids, are shown in Figure 13. Compared to the ground truth, $k = 2$ isolates *setosa* in the second cluster, while it splits the other two classes across both; $k = 3$, instead, perfectly identifies *setosa* in the second cluster and approximately separates *virginica* and *versicolor*, although it cannot capture their overlap.

Confusion matrices in Figure 14 show that the misclassified portion of the dataset with two clusters is 20.7%, while $k = 3$ yields better results with 18% of misclassified points, all between *versicolor* and *virginica* (first and third clusters).

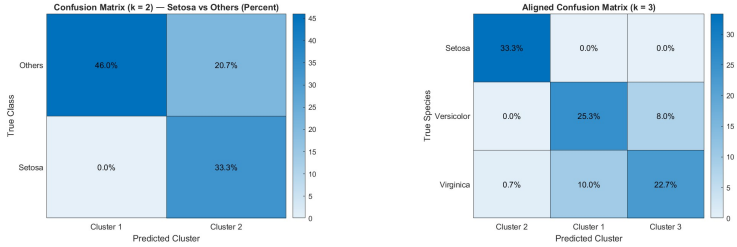

Figure 14: Confusion matrices for $k = 2$ and $k = 3$

4 Review of Data-Driven Design Papers

Batch Bayesian Optimization in Robotic Cooking

Junge et al. [1] address subjective food quality optimization using Batch Bayesian Optimization (BO) in contrast to Sequential Optimization. With a robotic platform capable of consistently preparing omelettes, the study investigates how control parameters (salt, pepper, mixing, whisking, and cooking time) affect flavor, appearance, and texture as rated by human users. A sensitivity analysis was used to split the multi-objective problem into two subgroups: flavor (influenced by salt, pepper, and mixing) and appearance/texture (influenced by mixing, whisking, and cooking time), and the corresponding optimization tasks were solved separately. Results showed that the strategy is effective in optimizing subjective data, particularly with Batch BO, due to improved feedback quality and broader exploration of the parameter space.

Batch Bayesian Optimization was chosen as it suits expensive, low-dimensional, stochastic functions, and in this context evaluations are limited by cooking cost, appetite, and fading memory of participants. Unlike Sequential BO, it allows parallel evaluations and re-assessment of feedback, reducing memory biases. It supports bulk optimization across users and operates with a fixed number of samples, without relying on a stopping criterion.

The method was tested in six experiments with a total of 73 omelettes. Compared to Sequential BO, Batch BO enabled better exploration of the input space, exposing human assessors to greater variability in the samples. Performance was measured using the Relative Improvement (RI) metric, comparing post-optimization scores to the mean of earlier trials, scaled by input variance. Higher exploration and the possibility of re-evaluation then led to improved results for Batch BO.

Future extensions could reduce subjectivity by integrating multi-modal sensory inputs (aroma, tactile, vision) or incorporating nutritional or dietary constraints. Preference models learned from user history could further personalize the process. Algorithmically, multi-objective Bayesian Optimization could better capture trade-offs, while scaling the system may enable generalization across diverse cultural preferences.

Multi-Objective Genetic Algorithms in Smart Kitchens

Dziurzanski et al. [2] present a data-driven framework for optimizing cooking processes in commercial kitchens. The authors adapt an IoT-based factory system to planning and scheduling food preparation, applying a digital twin of the smart kitchen to evaluate functions based on real-time data acquired from smart appliances. A Pareto front for three conflicting objectives (makespan, energy, and food quality, based on user feedback) is reconstructed using chromosomes that encode the recipe to be executed and its priority, subject to resource constraints. Experiments were conducted in two scenarios differing in order volume and resource availability, highlighting trade-offs such as makespan versus energy, or makespan versus quality, which can be mitigated by executing recipes in parallel.

The MOEA/D Genetic Algorithm was selected because the optimization problem is inherently multi-objective, involving trade-offs. This algorithm is specifically designed to decompose and solve multiple scalar subproblems in parallel. Moreover, evolutionary algorithms are well-suited to encoding scheduling problems, where each solution can be represented as a chromosome.

The method proved successful in generating high-quality solutions across all objectives. It was benchmarked in two scenarios: one with a standard workload and another with a scaled-up case involving four times the amount of ordered food. The generated schedules effectively exploited parallelism, handled resource constraints, and adapted to the dynamic unavailability of appliances.

Future extensions could integrate real-time feedback loops into the process, such as sensor-based evaluation of food quality. Additionally, incorporating adaptive learning mechanisms could improve optimization efficiency over time. Finally, enhancing the digital twin with probabilistic models to capture high uncertainty in appliance availability or cooking durations would increase the robustness of the method.

References

[1] Kai Junge et al. "Improving robotic cooking using batch Bayesian optimization". In: *IEEE Robotics and Automation Letters* (2020).

[2] Piotr Dziurzanski, Shuai Zhao, and Leandro Soares Indrusiak. *Integrated process planning and scheduling in commercial smart kitchens*. 2019. arXiv: `1910.03322 [cs.AI]`.