Hybrid quantum-classical metaheuristics for automated machine learning applications

Von Dollen, D.J.

Chapter 5

Quantum-enhanced evolution strategies

In this chapter we extend some of our previous work and examine QUBO-enhanced selection operations for evolution strategies for optimization over continuous and combined discrete and continuous domains. We address the problem of hyperparameter optimization of machine learning models as part of the larger AutoML process, and study the efficacy of our method to the application area of deep learning from tabular data sets. We also study this method when applied to classical reinforcement learning and neuroevolution of feedforward neural networks for learning control policies. This work is based on the following:

D. Von Dollen, R. Brasher, F. Neukart, and T. Bäck. Hyperparameter optimization and neuroevolution with binary quadratic meta-heuristics and evolution strategies. In *2023 7th IEEE Congress on Information Science and Technology (CiSt)*, pages 536–540, 2023. doi: 10.1109/CiSt56084.2023.10409984.

5.1 Introduction

In the field of artificial intelligence, there are many conceptual parallels between optimization and learning. Framing learning as an optimization problem allows for a data-driven approach. Machine learning (ML) is the process of systems learning from data and experience without explicit instruction. Often, machine learning models rely on parameters for model architecture and configuration known as hyperparameters. The

hyperparameters of machine learning models can themselves create high-dimensional configuration spaces whose optima are unknown. The problem for a practitioner can then become one of finding an optimal configuration of hyperparameters given a learning algorithm and data set.

Various approaches exist to automate the hyperparameter optimization routine, such as grid search, random search, gradient-based optimization, and Bayesian optimization [12, 203]. In this work, we examine the use of evolution strategies (ES) as a metaheuristic for this search. We propose a selection operator for ES and compare it to selection operators based on ranking over real-valued functions. We benchmark over common multimodal black-box functions with varying dimensionality and find that our selection operator shows improvement with respect to fitness given the same budget for a majority of the tasks in the benchmark suites. We then test our metaheuristic towards applications of hyperparameter optimization for deep learning models, where we optimize over both the model architecture and hyperparameters. We find that our approach shows improvement with respect to average fitness values, average validation accuracy, and average rank.

Neuroevolution is a technique for applying evolutionary algorithms towards improving the architecture or performance of artificial neural networks, which are machine learning models that are inspired by biological structures such as the brain. Evolution may be applied to the topology of the network or the weight space of the network [206]. In this study, we examine the evolution of the weight space for small feedforward neural networks using our method. We apply this to learning tasks, such as learning the XOR function and learning control policies for the cartpole environment of the OpenAI Gym [28]. Similar to results obtained towards quantum-enhanced genetic algorithms in chapter 4, we observe improvement in convergence rates and fitness using our method.

5.2 Methods

5.2.1 Extending the Quantum-Enhanced Selection Operator to Evolution Strategies

Evolution strategies (ES) are a type of evolutionary algorithm that may be applied to optimization problems with continuous objective functions. These algorithms date back to the 1960s [171]. We use a flavor of ES that uses recombination operators, which were introduced in the 1970s [190]. Our approach frames selection as a maximum di-

versity problem, which in itself is known to be NP-Hard [115] and modifies the operator of the heuristic to select parents for offspring using both fitness and distance. We hypothesize that by incorporating these fitness and distance metrics together within the metaheuristic, we may be able to balance the exploration/exploitation trade-off more effectively than by ranking alone.

As in previous chapters, we construct a QUBO to encode the selection operator. Let $\mathbf{Q}$ be the matrix that defines this QUBO. Let f be the fitness function. The diagonal entries of $\mathbf{Q}$ are given by $\mathbf{Q}_{ii} = f(\mathbf{a}_i)$ where $\mathbf{a}_i$ is a member of the population P indexed by i. The off-diagonal entries $\mathbf{Q}_{ij}$ are given by the correlation coefficients between $\mathbf{a}_i$ and $\mathbf{a}_j$ using the Pearson correlation coefficient (PCC). We chose this PCC over other metrics because we noticed an improved performance in the initial design phases, and also noticed that this measure works well with continuous variables as shown in Chapter 3. We then scaled this matrix by $\alpha = 10$ for diagonal entries and $\beta = 1000$ for off-diagonal entries to increase the ruggedness of the optimization landscape of the binary quadratic model. These scaling parameters were tuned on the basis of some preliminary studies on a subset of test functions(Ackley, Rastrigin) and findings from work in previous chapters. We observed that for multi-modal test functions, the setting $\alpha = 10$ and $\beta = 1000$ showed the best balance between the improvement in fitness and the diversity of the population.

$$\mathbf{Q}_{ij} = \begin{cases} \alpha f(\mathbf{a}_i) & \text{if } i = j \\ -\beta |PCC(\mathbf{a}_i, \mathbf{a}_j)| & \text{if } i \neq j \end{cases} \tag{5.1}$$

The negative sign on $-\beta|PCC(\mathbf{a}_i, \mathbf{a}_j)|$ ensures that the correlation between selected individuals decreases energy $E(\mathbf{o})$, promoting diversity when minimizing energy. The ratio β/α controls the exploration-exploitation trade-off, with higher ratios favoring diversity and lower ratios emphasizing fitness. Note that for maximization problems, we first negate the objective function f to convert to minimization, preserving this diversity-promoting mechanism.

Then, using this matrix as input for our QUBO, we optimize according to (5.2) to find a $\min_{\mathbf{o}}(E)$ to obtain solutions with which to select μ parents to recombine to create offspring λ, where E is defined by:

$$E(\mathbf{o}) = \sum_{i \geq j} \mathbf{o}_i \mathbf{Q}_{ij} \mathbf{o}_j + \left(\sum_i \mathbf{o}_i - \mu \right)^2 \tag{5.2}$$

where $\mathbf{o} \in \{0, 1\}^n$. Similarly to Chapter 4, we select members of the population at the

Abbreviation	Definition
QA-ES plus	QUBO-enhanced Evolution Strategy with elitism
QA-ES comma	QUBO-enhanced Evolution Strategy without elitism
ES plus	Evolution Strategy with elitism
ES comma	Evolution Strategy without elitism

Table 5.1: Abbreviation for algorithms

indices i where $\mathbf{o}_i = 1$ for the components of the decision vector $\mathbf{o}$.

As observed in chapter 3 and chapter 4, quantum-enhanced selection methods outperform random and greedy selection. Our motivation for this work is to extend these selection operators to continuous domains and investigate whether or not these performance enhancements hold. Previous work [226] used a quantum annealing system for hybrid quantum-classical optimization. For this study, we used simulators from the dimod library provided by D-Wave [210] to sample QUBO solutions, as we did not have access to a QPU. We acknowledge that the term "quantum-enhanced" is somewhat aspirational in this context, as all experiments were conducted using classical simulation rather than actual quantum hardware; however, the QUBO formulation remains suitable for future implementation on quantum annealers as hardware capabilities improve.

There are four variations of ES that we benchmarked. We use plus to denote elitist variations $(\mu + \lambda)$ (where we track the best solution found so far and use it for recombination) and comma for non-elitist, (μ, λ) (where we do not track the best solution, and do not carry parents over into the next generation). The four combinations are summarized in Table 5.1.

We examined both elitist (QA-ES plus) and non-elitist (QA-ES comma) versions of the heuristic. Using ranking selection operators (ES-plus, ES-comma), in the elitist cases (QA-ES plus, ES plus), we preserved the best solution found in the following population, and in the non-elitist cases (QA-ES comma, ES comma) created a population of pure offspring. After performing recombination, we mutated the chromosomes by adding a small amount of random noise drawn from a normal distribution. We repeated this procedure in an iterative manner until we reached a given budget in a number of generations g. For our experiments, we set the number of selected parents to $\mu = 50$, and offspring to $\lambda = 100$ and $g = 100$, and averaged the results over 20 trials. We choose to set the ratio of $\mu/n = 1/2$ rather than $1/7$ to observe an increase in diversity in the tail of the distribution of selected parents vs. ranked approaches [7].

The motivation for setting these parameters at these values was rationalized in that we found the gap between the convergence velocities to be the largest at $\mu/n = 1/2$, as shown in chapter 4. This selection rate is more relaxed than the typical 1/7 ratio recommended for standard evolution strategies, in our tests we found this rate to work more effectively with the diversity-promoting QUBO selection mechanism.

For maximization objectives (e.g., validation accuracy in hyperparameter optimization experiments), we transform the problem to minimization by negating the objective function: $f'(\mathbf{a}) = -f(\mathbf{a})$. This transformation is also applied when constructing the QUBO matrix $\mathbf{Q}$ in Equation 5.1, resulting in the negated diagonal entries mentioned in this chapter. We show the outline of the main algorithm in Algorithm 2.

The construction of the QUBO matrix $\mathbf{Q}$ requires the computation of pairwise correlations for a population dominated by $O(n^2)$, which we acknowledge introduces some overhead. Solving QUBO using classical simulators adds further cost. We highlight that for problems with expensive objective function evaluations, such as hyperparameter optimization, where each evaluation requires training a neural network, this overhead may be negligible compared to the evaluation budget. We recognize that this overhead could also be substantially reduced with quantum annealing hardware, which has the potential to find high-quality QUBO solutions in microseconds [136]. Since we lacked access to a QPU, this work uses classical simulation as a proof-of-concept to investigate the selection mechanism's efficacy rather than assess quantum hardware advantages.

5.3 Experiments

5.3.1 Black box optimization Benchmarking functions

For our benchmarks, we first examined black-box multi-modal real-valued functions from the Python optproblems library [234]. The functions we selected included the *Ackley*, *Rastrigin*, *Griewank*, *Vincent*, *Lunacek*, and *Weierstrass* functions as referenced in Table 5.2 and Table 5.9. All of these functions are real-valued and multimodal as shown in Figure 5.1. For this suite of experiments, we examined varying the size of the problem d, looking at the dimensions of the problem $d \in \{3, 20, 50\}$.

5.3.2 HPO Benchmarking suite

We also wanted to explore the application area of hyperparameter optimization for machine learning models as part of this work. For benchmarking, we chose the Yahpo-

Algorithm 2 Quantum-Enhanced Evolution Strategy (*QA-ES-plus, QA-ES-comma*)

Require: Population size n
Require: Chromosome size d
Require: Number of parents μ
Require: Number of offspring λ
Require: Maximum generations g
Require: Mutation strength σ
Require: QUBO scaling parameters α, β
Require: Objective function f (to be minimized)
Require: Boolean flag *elitist*
Require: Boolean flag *quantum-enhanced*
Initialize $P(0) \leftarrow [\mathbf{a}_1, \ldots, \mathbf{a}_n]$, where $\mathbf{a}_i \in \mathbb{R}^d$, uniformly distributed
Initialize $\mathbf{a}^+ \leftarrow \mathbf{a}_1$
for generation $t = 1$ to g **do**
    for $\mathbf{a}_i \in P(t-1)$ **do**
        if $f(\mathbf{a}_i) < f(\mathbf{a}^+)$ **then**
            $\mathbf{a}^+ \leftarrow \mathbf{a}_i$
        end if
    end for
    if *quantum-enhanced* **then**
        Use $P(t-1)$ to construct $\mathbf{Q}$ for QUBO as per 5.1
        Sample solution vector minimizing $E(\mathbf{o})$ from QUBO as per 5.2
        Select $P_\mu(t)$ from $P(t-1)$ where $o_i = 1$ for each o_i in $\mathbf{o}$
    else
        Select μ parents from $P(t-1)$ by ranking to form $P_\mu(t)$
    end if
    if *elitist* **then**
        $P_\mu(t) \leftarrow P_\mu(t) \cup \{\mathbf{a}^+\}$ ▷ Include best solution in parent pool
    end if
    Initialize $P_\lambda(t) \leftarrow \emptyset$
    for $k = 1$ to λ **do**
        Randomly select $\mathbf{p}_1, \mathbf{p}_2 \in P_\mu(t)$
        $\mathbf{c} \leftarrow \frac{\mathbf{p}_1 + \mathbf{p}_2}{2}$ ▷ Intermediate recombination
        $\mathbf{c}' \leftarrow \mathbf{c} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ ▷ Gaussian mutation
        $P_\lambda(t) \leftarrow P_\lambda(t) \cup \{\mathbf{c}'\}$
    end for
    $P(t) \leftarrow$ best n individuals from $P_\lambda(t)$
end for
return $\mathbf{a}^+$

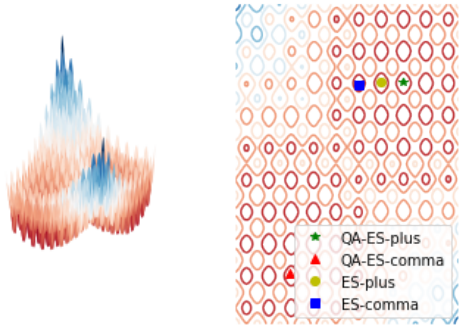

Figure 5.1: Best results for optimization algorithms after 1 run over Lunacek function.

gym library [163], specifically the *lcbench* test suite, which is in turn based on work from [249]. This suite contains 34 separate instances of ResNet-based surrogate models [89] in a variety of tabular datasets. The datasets are compiled from the OpenML repository and are collected from real-world scenarios and problem settings, including forest cover type prediction, handwritten digit recognition, 3D object recognition, and sound type classification. For a complete list of OpenML datasets with short descriptions, see Table 5.10. The configuration space for the hyperparameter optimization routine includes a combination of common hyperparameters for deep neural networks such as *learning rate*, *momentum* and *dropout rate*, and network architecture hyperparameters such as *number of layers* and *max number of units* as shown in 5.2. We chose this benchmark because all the parameters for the configuration space were of a mixture of real and integer values, which in turn resulted in a useful encoding for our matrix $\mathbf{Q}$ for our binary quadratic selection operator for the evolution strategy metaheuristic. Since the search consisted of maximizing the validation accuracy, we made a small change and negated the diagonal entries in the QUBO for the selection operator and set $\alpha = 1000$ and $\beta = 1$. This reverses the emphasis compared to black-box optimization, as preliminary studies showed the mixed discrete-continuous hyperparameter space benefits more from fitness-driven selection and exploitation around high-performing configurations.

5.3.3 Neuroevolution and RL agents

For our third benchmark, we chose neuroevolution of weights for neural networks to learn the XOR function and policies for the cartpole environment from the OpenAI Gym reinforcement learning library [28]. For XOR, the network architecture consisted

```
{'OpenML_task_id': '3945',
 'batch_size': 20,
 'epoch': 49,
 'learning_rate': 0.07069550588647724,
 'max_dropout': 0.11804066212348294,
 'max_units': 73,
 'num_layers': 4,
 'weight_decay': 0.019944111259510827}
```

Figure 5.2: Example configuration of hyperparameters for the lcbench benchmark suite.

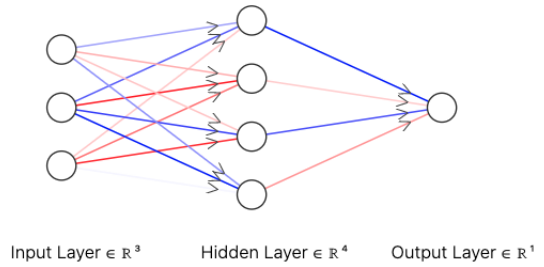

Figure 5.3: Feedforward neural network architecture for learning XOR.

of 3 input units, 4 hidden units, and 1 output unit with *sigmoid* activations between layers and cross-entropy loss (see Figure 5.3). For the cartpole RL environment, we use an architecture of 4 units for the input layer, 8 units for the hidden layer, and 2 units for the output layer with *relu* activation between layers and *softmax* output. We took the argmax over the output layer to prescribe an action from the action space for the given input state. For these neuroevolution experiments, we conducted a hyperparameter search and found that using the Euclidean norm as the distance metric in the QUBO formulation of the selection operator gave performance improvements over *PCC* for these tests. Unlike continuous optimization problems where PCC effectively captures the correlation structure, the weight spaces of the neural network benefited from the more direct distance measure provided by the Euclidean norm. We tuned the α and β parameters using some preliminary studies for the feedforward neural network, and found that the setting of $\beta = 100$ and $\alpha = 1000$ yielded faster convergence than purely fitness-driven selection while maintaining sufficient diversity in the neural network weight space.

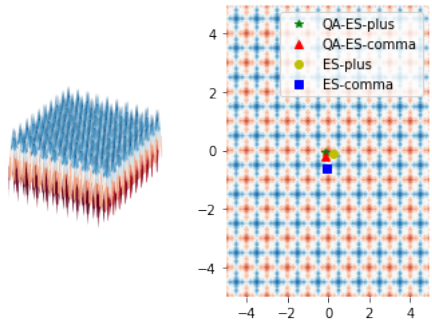

Figure 5.4: Best results for optimization algorithms after 1 run over Weierstrass function.

5.4 Results

For the first suite of benchmarks, we examined the variation of the problem dimension size in Tables 5.2, 5.3, and 5.4. For 94.4% of these tests, our method showed performance gains with respect to average fitness. Some of these improvements were very large; for example, the *Ackley* case from Table 5.2 showed an improvement by a factor of 15. This could be attributed to the increase in convergence velocity in the higher settings of α within the QUBO formulation. To assess the statistical significance of performance differences between quantum-enhanced and classical variants, we performed two-sample t-tests comparing the mean fitness values across 20 independent trials. We report p-values in the results tables, with the significance threshold set at $p < 0.1$. All tests are two-tailed, as we test for differences in either direction.

Interestingly, our method did not show improvements in the single case of the Weierstrass function of size $50d$ as shown in Table 5.4. This could be due to the structure and fractal-like nature of the Weierstrass function itself, with many optima spread over a large neighborhood, as shown in Figure 5.4. For our hyperparameter optimization tests, we averaged all 33 *lcbench* test scenarios and found that our method obtained the highest ranking of all other methods as shown in Table 5.6. We explain this acceleration in convergence velocity by carefully tuning α and β in the QUBO selection mechanism to balance fitness and diversity within the population. When evaluating for validation accuracy in Table 5.5, we observed that our method achieved better performance over both the elitist and the non-elitist versions, although for the elitist cases (QA-ES-plus, ES-plus) the significance level was not high enough to reject the null hypothesis after testing for normality. However, in the non-elitist case (QA-ES-comma, ES-comma), our method outperformed the baseline with a significance

level to reject the null hypothesis (p-value=0.0011). For some cases such as trial 126025 from the lcbench suite, the difference was substantial between our method vs. others, especially in the initial generations of the optimization routine as shown in Figure 5.5.

We tabulated the results of the trials for applying our method toward neuroevolution for learning XOR and reinforcement learning for control policies for the cartpole environment. As shown in Figure 5.6, our method showed better average best values but did not achieve strong enough results to reject the null in the elitist case (p-value=0.21). However, for the cartpole environment, our non-elitist method significantly (p-value = 2.9e-5) outperformed baselines.

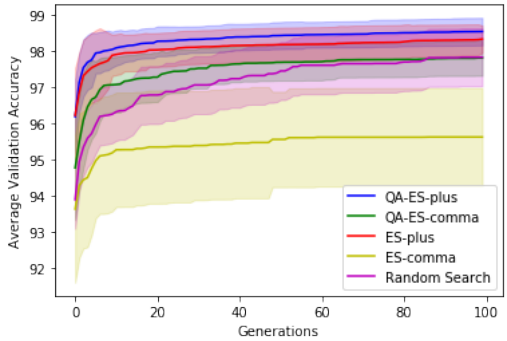

Figure 5.5: Validation Accuracy curves for lcbench benchmark 126025, 20 trials, 100 generations.

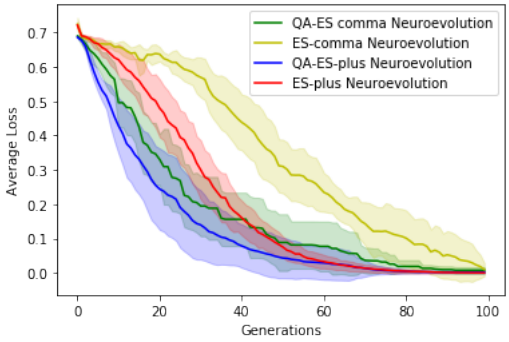

Figure 5.6: Average cross-entropy loss over generations for learning XOR: 10 trials, 100 generations.

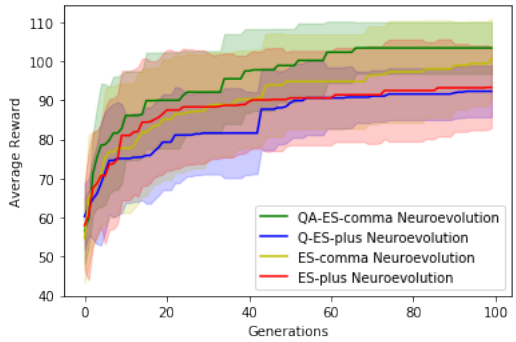

Figure 5.7: Average Reward over generations for the cartpole environment: 10 trials, 100 generations.

5.5 Conclusions

We applied our method across a variety of problems with objective functions that were real-valued or a mix of real-valued and integer-valued, to build on previous work which only examined pseudo-Boolean domains. For multimodal objective functions, our method performed well across varying problem sizes. The results show even stronger performance by the selection operator over multi-modal and continuous domains in comparison to results over pseudo-Boolean functions in chapter 4.

We applied our method as a metaheuristic with ES for applications of hyperparameter optimization for machine learning models. For these applications, we found that our method did well in the non-elitist case, but only slightly better in the elitist cases, both in terms of rank and validation accuracy. However, non-elitist versions significantly outperformed baselines.

In cases of neuroevolution, we saw increased convergence rates as shown in Figure 5.6 for the XOR problem. It was interesting that non-elitist showed better performance for neuroevolution of RL agents. This suggests that by incorporating a diversity metric as a metaheuristic within the optimization routine, the exploitation/exploration trade-off may be more effectively balanced than by a purely rank-based approach.

Future work could examine applying this method to neural architecture search, where network topologies are evolved, as well as the weight space for the network. Although we limited the population size for this study, future work could examine larger population sizes with problems of higher dimensionality. Multi-objective optimization for hyperparameter optimization is another area that could be explored in future works. Future areas of research could examine using different types of quan-

tum computing for QUBO sampling, such as the approximate quantum optimization algorithm on a gate model quantum computer [63].

In pursuing our research questions, we show with these results positive cases for RQ3, and confirm that this selection operator may be useful as a metaheuristic in hybridization with classical optimization routines. In the case of optimization over the suite of multi-modal black box functions, these results make a stronger case in comparison to chapter 4, with more conclusive results across the black-box optimization benchmarks. As the size and coherence of quantum processing units increase, these results may prove to be even more beneficial by being able to accommodate larger population sizes with larger chips with higher degrees of connectivity.

Objective Function	QA-ES-plus	QA-ES-comma	ES-plus	ES-comma	p-value
Ackley	**0.353** +/- 0.15	1.04 +/- 0.41	5.268 +/- 0.84	1.008 +/- 0.59	4.6e-17
Rastrigin	**1.418** +/- 0.55	2.787 +/- 0.92	12.257 +/- 3.35	2.647 +/- 1.05	4.5e-12
Griewank	**0.008** +/- 0.0	0.01 +/- 0.0	0.029 +/- 0.01	0.01 +/- 0.01	5.4e-07
Vincent	**-0.997** +/- 0.0	-0.997 +/- 0.0	-0.993 +/- 0.01	-0.995 +/- 0.0	2.5e-03
Lunacek	**1.041** +/- 0.59	2.156 +/- 0.74	6.068 +/- 2.03	1.948 +/- 0.99	2.8e-10
Weierstrass	1.04 +/- 0.24	**1.0** +/- 0.22	1.049 +/- 0.23	1.033 +/- 0.23	4.6e-01

Table 5.2: Average fitness values over 20 trials over black-box function benchmarks problem dimension=3, population size=100, 100 generations. Values shown as mean +/- standard deviation over trials.

Objective Function	QA-ES-plus	QA-ES-comma	ES-plus	ES-comma	p-value
Ackley	**4.2** +/- 0.2	6.2 +/- 0.3	7.7 +/- 0.3	6.2 +/- 0.5	1.8e-32
Rastrigin	**150.5** +/- 14.4	166.4 +/- 13.4	206.1 +/- 13.9	170.2 +/- 10.4	6.5e-15
Griewank	**0.5** +/- 0.1	0.9 +/- 0.0	0.9 +/- 0.1	0.9 +/- 0.0	9.1e-14
Vincent	**-0.7** +/- 0.0	-0.6 +/- 0.0	-0.6 +/- 0.0	-0.6 +/- 0.0	8.2e-11
Lunacek	**142.8** +/- 14.4	205.8 +/- 12.9	258.1 +/- 17.6	209.4 +/- 15.0	5.0e-23
Weierstrass	**25.8** +/- 1.1	25.8 +/- 1.4	26.3 +/- 0.9	26.1 +/- 1.1	0.07

Table 5.3: Average fitness values over 20 trials over black-box function benchmarks problem dimension=20, population size=100, 100 generations. Values shown as mean +/- standard deviation over trials.

Objective Function	QA-ES-plus	QA-ES-comma	ES-plus	ES-comma	p-value
Ackley	**5.94** +/- 0.22	7.56 +/- 0.22	8.29 +/- 0.19	7.64 +/- 0.17	3.50e-30
Rastrigin	**524.62** +/- 26.62	559.25 +/- 17.89	611.44 +/- 23.80	555.20 +/- 22.77	3.80e-13
Griewank	**0.86** +/- 0.04	1.04 +/- 0.01	1.05 +/- 0.00	1.04 +/- 0.01	5.10e-14
Vincent	**-0.48** +/- 0.03	-0.43 +/- 0.03	-0.40 +/- 0.03	-0.43 +/- 0.02	1.10e-09
Lunacek	**586.08** +/- 53.55	705.02 +/- 26.99	787.87 +/- 28.60	702.80 +/- 30.46	4.00e-15
Weierstrass	77.95 +/- 1.44	78.98 +/- 1.49	**77.80** +/- 1.73	78.41 +/- 1.45	0.61

Table 5.4: Average fitness values over 20 trials over black-box function benchmarks problem dimension=50 population size=100, 100 generations. Values shown as mean +/- standard deviation over trials.

QA-ES-plus	QA-ES-comma	ES-plus	ES-comma	Random Search	p-value
98.49 +/- 0.08	-	98.46 +/- 0.10	-	97.76 +/- 0.35	0.09
-	**97.8 +/- 0.14**	-	96.03 +/- 0.25	97.76 +/- 0.35	1.1E-45

Table 5.5: Average Validation Accuracy results over 33 lcbench hyperparameter optimization benchmarks, 20 trials per benchmark. Values shown as mean +/- standard deviation over trials.

QA-ES-plus	QA-ES-comma	ES-plus	ES-comma	Random Search
4.529412	2.588235	4.411765	1.0	2.47

Table 5.6: Average Rank results over 33 lcbench hyperparameter optimization benchmarks, 20 trials per benchmark, (higher is better). Values shown as mean +/- standard deviation over trials.

QA-ES-plus	QA-ES-comma	ES-plus	ES-comma	p-value
0.0007 +/- (0.001)	0.0101 +/- (0.02)	0.0040 +/- (0.01)	0.0164 +/- (0.02)	0.21

Table 5.7: Average best fitness over XOR learning problem. Values shown as mean +/- standard deviation over trials.

QA-ES-plus	QA-ES-comma	ES-plus	ES-comma	p-value
92.3 +/- (6.6)	**103.4 +/- (6.6)**	93.3 +/- (10.3)	93.3 +/- (10.6)	2.9e-05

Table 5.8: Average Reward for Cartpole RL environment. Values shown as mean +/- standard deviation over trials.

Name	**Objective Function**	**Optima**
Ackley	$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	Multiple Local
Rastrigin	$f(\mathbf{x}) = 10d + \sum_{i=1}^{d}\left(x_i^2 - 10\cos(2\pi x_i)\right)$	Multiple Local
Griewank	$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	Multiple Local
Vincent	$f(\mathbf{x}) = -\sum_{i=1}^{d}\sin(10\log(x_i))$	Multiple Global
Lunacek	$f(\mathbf{x}) = \min\left(\sum_{i=1}^{d}(x_i-\mu_1)^2, d\cdot s^2 + \sum_{i=1}^{d}(x_i-\mu_2)^2\right) + 10\left(d - \sum_{i=1}^{d}\cos(2\pi(x_i-\mu_1))\right)$	Multiple Local
Weierstrass	$f(\mathbf{x}) = \sum_{i=1}^{d}\left(\sum_{k=0}^{k_{\max}} a^k\cos(2\pi b^k(x_i+0.5))\right)$	Fractal-like

Table 5.9: Black-box objective Functions and optima used to benchmark optimization algorithms.

5.6 Summary

In this chapter, we extended the quantum-enhanced selection operator to evolution strategies and optimization over continuous objective functions. We tested our method

over a variety of multi-modal black-box test functions, and applied the method to the problem of hyperparameter optimization over tabular data. This was done in response to RQ3. We also investigated the application area of neuroevolution of reinforcement learning agents for control problems. These problems are central to the AutoML process for hyperparameter optimization and may be of benefit to the machine learning practitioner seeking to effectively search configuration spaces for machine learning models.

OpenML Task ID	OpenML Dataset Name	Description
3945	KDDCup09_appetency	CRM strategies prediction
7593	covertype	Forest cover type prediction
34539	Amazon employee access	Employee access prediction
126025	adult	Income over 50K prediction
126026	nomao	Location data deduplication
126029	bank-marketing	Term deposit subscription prediction
146212	shuttle	Shuttle simulator data analysis
167104	Australian	Credit card application outcomes
167149	kr-vs-kp	Chess end-game prediction
167152	mfeat-factors	Handwritten numeral recognition
167161	credit-g	Credit risk assessment
167168	vehicle	3D object recognition
167181	kc1	Software defect dataset
167184	blood-transfusion-service-center	Blood donation prediction
167185	cnae-9	Business description classification
167190	phoneme	Sound type classification
167200	higgs	Higgs boson detection
167201	connect-4	Connect-4 game outcomes
168329	helena	Anonymized ML classification challenge
168330	jannis	Anonymized ML classification challenge
168331	volkert	Anonymized ML challenge
168335	MiniBooNE	Neutrino type classification
168868	APSFailure	Component failure prediction
168908	christine	Anonymized ML classification challenge
168910	fabert	Anonymized ML classification challenge
189354	airlines	Flight delay classification challenge
189862	jasmine	Anonymized ML classification challenge
189865	sylvine	Anonymized ML classification challenge
189866	albert	Anonymized ML classification challenge
189873	dionis	Anonymized ML classification challenge
189905	car	Car model predictions
189906	segment	Image segmentation task
189908	Fashion-MNIST	Clothing image classification
189909	jungle_chess_2pcs_raw_endgame_complete	Jungle Chess game analysis

Table 5.10: *lcbench* real-world datasets for classification challenges.