Optimising the Lunar Lander Using (1+1)-ES and Differential Evolution

Ace Abenroth
Sarah Lobis
Jonas Saathoff

Group number 9

Abstract. This study examines the optimisation of a neural-network controller for the Lunar Lander task in Gymnasium, a nonlinear and non-differentiable control problem that requires direct search methods. We implement and compare multiple variants of the (1+1)-Evolution Strategy (ES) and Differential Evolution (DE), including a fixed-parameter baseline, self-adaptive approaches, and a novel Adaptive Mixed DE, across both discrete and continuous control scenarios. The Adaptive Mixed DE consistently achieves the strongest performance, indicating that adaptive parameter control and dynamic strategy selection are highly effective for high-dimensional, stochastic control tasks.

1 Introduction

Evolutionary Computation (EC) algorithms are particularly useful for optimisation tasks where the objective function is nonlinear, noisy, or non-differentiable [8]. In continuous search spaces, direct search methods such as Evolution Strategies (ES) and Differential Evolution (DE) are often methods of choice [12]. In this project, we optimise the real-valued weights of a neural-network controller for the Lunar Lander environment in Gymnasium [14]. The goal is to find controller parameters that guide a spacecraft to a safe landing while maximizing cumulative rewards. We evaluated the (1+1)-ES [4] against multiple variants of DE [5]. We compared a fixed-parameter DE baseline with a Self-Adaptive DE and an Adaptive Mixed DE. Through extensive experiments supported by IOH benchmarking, we observed a clear hierarchy: the Adaptive Mixed DE achieved the highest stability, demonstrating that dynamic strategy selection offers a significant advantage. This aligns with recent findings in engineering domains such as photovoltaic parameter extraction [7].

2 Problem Description

In this assignment, our goal is to optimise a controller for the Lunar Lander so that the lander can reach the designated landing pad safely. The controller is defined by a set of real-valued weights that map the current state of the lander to a thruster configuration. We treat this as a black-box optimisation problem, meaning the only information available to us about solution quality is the final fitness score returned by the simulator. Since the simulator is dynamic and potentially noisy, especially when wind or turbulence are enabled, and fitness is shaped by many interacting components, including position error, velocity, tilt, fuel usage, and crash or landing bonuses, multiple runs of each algorithm and configuration are required. In our experiments, we therefore consider two environment configurations: a normal setting with default gravity $g = -10$, no wind, and no turbulence, and a deliberately noisy "Wind & Gravity" setting with increased gravity $g = -11.9$, wind power $= 12.0$ and turbulence $= 2.0$, which allows us to study the impact of these simulation parameters on optimiser robustness.

2.1 Understanding the task

The lunar lander has three thrusters, and its state at each simulation step is represented by eight values: position (x, y), velocity (v_x, v_y), angle, angular velocity, and two Boolean flags for leg contacts. To assess optimiser flexibility, we utilize both action space definitions provided by Gymnasium. In the *Continuous* variant, the controller maps the state to a vector $a \in [-1, 1]^2$, determining the continuous power of the main and lateral engines. In the *Discrete* variant, the controller outputs four real-valued logits, where an argmax operation selects one of four mutually exclusive actions: do nothing, fire left orientation engine, fire main engine, or fire right orientation engine. The quality of a candidate solution is expressed through a reward function accumulated over the entire landing attempt. Rewards increase when the lander moves steadily toward the landing pad and decrease when it becomes unstable, tilts too much, fires thrusters inefficiently, or crashes. The reward signal includes bonuses for leg contact and stable landing, with penalties for engine usage and crashing. The optimisation goal is therefore to maximise the total reward by choosing parameters that lead to consistently safe and smooth landings.

2.2 Approach

Since the Gymnasium environment acts as a black-box oracle, we rely on Evolutionary Computation (EC) direct search methods. Our strategy explores how increasingly sophisticated optimizers handle the multimodal and noisy characteristics of the fitness landscape.

To establish a baseline, we begin with trajectory-based search using the (1+1)-Evolution Strategy. This allows us to test whether simple hill-climbing is sufficient for this task. To examine the impact of mutation control, we compare both a fixed step-size version and a self-adaptive variant that autonomously adjusts its mutation strength during the search.

We then move to population-based optimisation with Differential Evolution (DE), which offers greater robustness against local optima than single-solution methods. Given that DE is highly sensitive to its control parameters, particularly the scaling factor and crossover rate, we include self-adaptive variants that allow the optimizer to tune its exploration-exploitation balance dynamically.

To ensure reliable comparisons under stochastic environmental conditions, all algorithms are benchmarked over multiple independent runs. Performance is then statistically analyzed using the IOHprofiler framework.

3 Optimisation tasks

3.1 Overview

In this work, we implemented and evaluated three variants of the (1+1)-Evolution Strategy (ES) to serve as baselines and diagnostic tools, alongside multiple Differential Evolution (DE) variants and a novel Adaptive Mixed DE algorithm that integrates strategy mixing with online parameter adaptation. All algorithms operate on flattened weight vectors (solutions) $x \in \mathbb{R}^D$, which are reshaped by the `GymProblem` wrapper into the controller matrices required by the environment. To ensure fair comparison, all runs utilize the same evaluation and IOH logging protocol as described in the Experimental Setup.

3.2 (1+1)-Evolution Strategy Variants

We describe three ES implementations included in our codebase. While our experiments primarily focus on the `one_plus_one_combo.py` variant due to its robustness, we also report results for the canonical `one_plus_one_es.py` and the self-adaptive `one_plus_one_self_adaptive.py` for comparative analysis.

Basic (1+1)-ES The `one_plus_one_es.py` variant implements the canonical (1+1)-ES as described in standard evolutionary computation literature [3,6]. It maintains a single parent x and a global scalar step-size $\sigma > 0$. In each iteration, an offspring is generated via $x' = x + \sigma \cdot \mathcal{N}(0, I)$. The offspring x' replaces the parent x only if $f(x') \geq f(x)$ (greedy selection). Step-size adaptation follows the classical $1/5$-success rule [6,10] implemented in a windowed format: every k iterations, the success rate is calculated. If the success rate exceeds $1/5$, σ is multiplied by a factor a; if it falls below, σ is divided by a. This variant serves as a minimal baseline for local search performance.

Self-adaptive (1+1)-ES The `one_plus_one_self_adaptive.py` variant utilizes continuous self-adaptation of the mutation strength, a method detailed by Schwefel [11]. The step-size is encoded within the genome and mutated alongside the solution via $\sigma' = \sigma \cdot \exp(\tau \cdot \mathcal{N}(0,1))$ followed by $x' = x + \sigma' \cdot \mathcal{N}(0, I)$. The learning rate is set to $\tau \approx 1/\sqrt{D}$. Successful offspring pass their mutated σ' to the next generation, allowing the algorithm to adapt the mutation magnitude implicitly without an external success counter.

Combined (1+1)-ES (Focus) The `one_plus_one_combo.py` variant integrates multiple heuristic improvements into a single robust routine. It is the primary ES baseline used in our results. Its key features include:

- **Mirrored Sampling:** To improve gradient estimation and convergence speed, the algorithm employs mirrored sampling ('mirrored=True'), generating antithetic pairs of perturbations [1].
- **Noise Handling:** The parent solution is periodically re-evaluated ('reevaluate_k=3') to prevent the algorithm from latching onto falsely optimistic fitness values caused by environmental stochasticity.
- **Mixed Mutation & Dual Adaptation:** The algorithm alternates between isotropic and anisotropic mutations, combining the $1/5$-success rule with log-normal updates for robustness.
- **Restart Mechanism:** To mitigate stagnation, a restart is triggered if the fitness plateaus for a configurable number of evaluations ('restart_no_improve=200') [2].

3.3 Differential Evolution Variants

All DE variants follow the standard population-based framework introduced by Storn and Price [13]. The framework includes a population of size NP (default 30), mutation to generate donor vectors, crossover with probability CR, and greedy selection.

Crossover Mechanisms We implemented and compared two distinct multiple crossover variants :

- **Binomial Crossover (Bin):** The standard method where each parameter is independently swapped from the mutant vector with probability CR. This assumes little correlation between adjacent parameters.
- **Exponential Crossover (Exp):** A method that acts as a "splicing" operator, copying a contiguous block of parameters from the mutant vector. This preserves potential linkages between neighboring weights in the genome.

Mutation Strategies We evaluated the following mutation strategies:

- **rand/1** and **rand/2**: Standard exploratory strategies (using 1 or 2 difference vectors).
- **best/1** and **best/2**: Donors are formed using the current best individual, biasing the search toward exploitation.
- **rand-to-best/2**: Combines a random base with a vector pointing toward the best solution.
- **current-to-best/1**: A mutation strategy that moves the current individual toward the best solution, often providing faster convergence.

We employed the jDE self-adaptation scheme [5] initialized with $F = 0.8$ and $CR = 0.9$. These standard high values were chosen to ensure aggressive exploration in the early generations, effectively preventing premature convergence in the complex landscape before the adaptation mechanism fine-tunes the search parameters.

3.4 Adaptive Mixed DE

The `adaptive_mixed_de` variant extends the jDE framework by introducing online strategy selection, inspired by the SaDE algorithm [9]. Each individual is assigned a strategy index s_i from a pool $S = \{\text{rand/1, rand/2, rand-to-best/2}\}$.

We maintain an Exponential Moving Average (EMA) of success rates for each strategy. Strategies are sampled for offspring generation based on probabilities derived from these EMA weights. When a strategy s successfully produces a superior offspring, its weight is increased:

$$\text{EMA}_s \leftarrow (1 - \alpha) \cdot \text{EMA}_s + \alpha \cdot \mathbb{I}(\text{success}) \tag{1}$$

This mechanism ensures that the algorithm dynamically allocates resources to the mutation operators that are most effective at the current stage of optimisation.

3.5 Parameter Summary

Experimental parameters were set as follows: **(1+1)-ES (Combo)** uses $\sigma_0 = 0.1$, mirrored sampling, re-evaluation frequency $k = 3$, and restart threshold 200. **Basic (1+1)-ES** uses $\sigma_0 = 0.1$ with 1/5-rule adaptation [6]. **Self-adaptive ES** applies log-normal σ updates with $\tau = 1/\sqrt{D}$ [11]. All **DE Variants** use $NP = 30$, initial $F = 0.8, CR = 0.9$ with jDE [5]. The **Adaptive Mixed DE** selects from $\{\text{rand/1, rand/2, rand-to-best/2}\}$ with adaptation rate $\alpha = 0.05$ [9].

Table 1: Performance on **Continuous Control (Normal)**. Sorted by mean fitness.

Algorithm	Mean	Median	Best	Worst
Adaptive Mixed DE	**313.03**	313.96	328.04	291.79
Standard DE	312.45	312.67	326.88	291.94
Adaptive DE	310.67	311.67	325.35	290.08
Current-to-Best/1	310.90	311.07	328.58	281.69
Best/2	309.15	311.98	323.59	282.16
Best/1	306.16	313.41	**333.73**	228.53
Rand-to-Best/2	304.96	303.49	328.22	286.06
Rand/2	304.15	304.46	324.94	268.97
(1+1)-ES Combo	288.48	285.57	320.38	245.53
(1+1)-ES Plain	120.36	157.89	315.70	-269.07

Table 2: Performance on **Discrete Control** (Normal vs. Wind). Note the stability of Adaptive Mixed DE versus the risk of Best/1.

	Normal Environment		Wind & Gravity	
Algorithm	Mean	Worst	Mean	Worst
Adaptive Mixed DE	**316.40**	290.20	**317.24**	276.92
Adaptive DE	315.75	287.69	313.79	288.13
Standard DE	314.32	283.24	312.67	282.54
Rand/2	311.84	293.11	311.36	290.72
Best/2	308.97	281.87	308.40	260.79
Best/1	300.14	92.99	298.93	83.69
(1+1)-ES Combo	286.82	248.61	296.83	273.86
(1+1)-ES Plain	78.02	-298.52	21.17	-290.91

4 Results

In this section, we analyze the performance of the proposed algorithms across three scenarios: Continuous Control (Normal), Discrete Control (Normal), and Discrete Control with Wind/Turbulence. All results are based on 30 independent runs per algorithm with a budget of 20,000 evaluations.

4.1 Performance Statistics

Table 1 and Table 2 summarize the statistical performance. We report the Mean Reached fitness (average final score), the Best Reached (peak performance), and the Worst Reached (indicating risk of crashing).

4.2 Convergence and Stability

Differences in learning dynamics are illustrated in Fig. 1. The population-based methods (DE) show a steep learning curve, reaching high fitness values within the first 5,000 evaluations. The "Combo" ES, while an improvement over the plain ES, converges more slowly and plateaus at a lower fitness level.

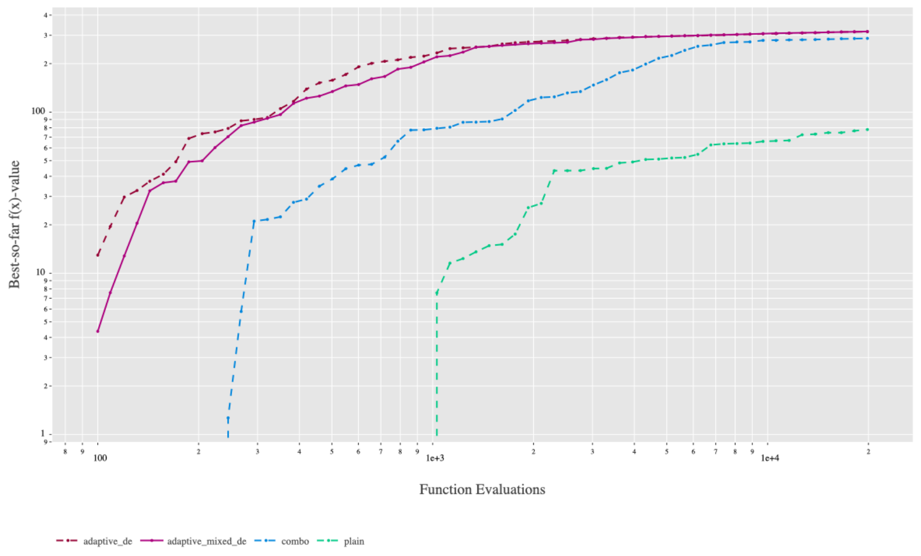

Fig. 1: Mean best-so-far fitness over evaluations (Continuous Normal). DE variants show rapid convergence compared to ES baselines.

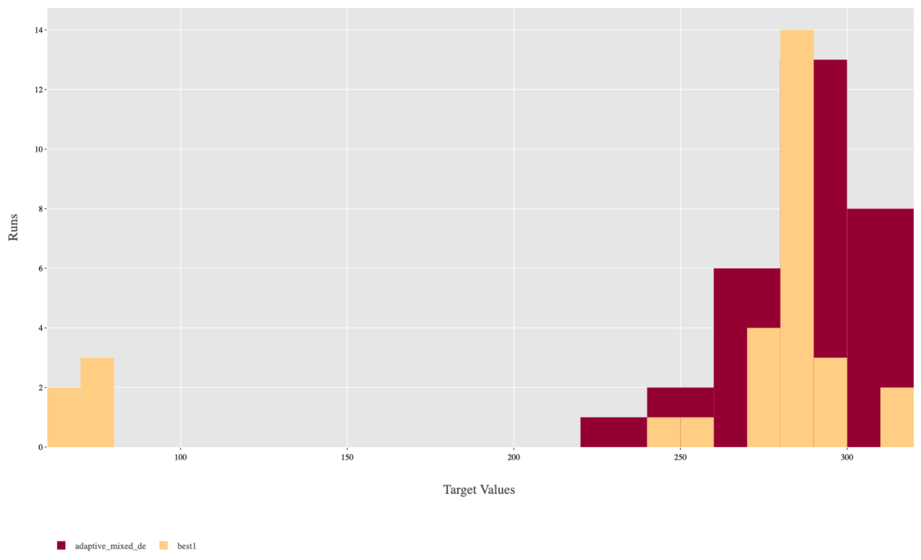

Fig. 2: Distribution of final fitness values in the Discrete Wind/Gravity scenario. Note the high variance in 'best1' compared to the consistency of 'Adaptive Mixed DE'.

In the noisy "Wind and Gravity" scenario, stability becomes the primary concern. Fig. 2 reveals the risk of aggressive strategies. While `best/1` occasionally finds the global optimum (peak 333.73), its distribution is wide, with worst-case outliers dropping to 83.69. In contrast, `Adaptive Mixed DE` maintains a compact distribution, reliably producing safe landing controllers (> 275 fitness) even in adverse conditions.

4.3 Impact of Crossover Strategy

We compared Binomial and Exponential crossover using Adaptive Mixed DE on the Discrete Control scenario (30 runs). Binomial crossover achieved superior performance (Mean: 316.80, Median: 321.94) compared to Exponential (Mean: 309.34, Median: 307.69). This advantage stems from Binomial's independent parameter mixing, which swaps genes with probability CR regardless of position. This decorrelation better suits the unstructured neural network weight space, whereas Exponential's block-based copying assumes a positional linkage that does not exist in our flattened weight vectors [13].

4.4 Discussion

DE vs. ES Baselines The results unequivocally show that Differential Evolution outperforms the (1+1)-ES baselines in this high-dimensional control task ($D \approx 300$). The plain ES frequently fails to solve the environment (mean fitness < 100 in discrete cases), likely due to its inability to adapt step-sizes effectively across all dimensions simultaneously. The "Combo" ES bridges the gap using restarts and mirrored sampling, but the population-based diversity of DE proves superior for avoiding local optima.

Adaptive Strategy Selection The `Adaptive Mixed DE` consistently achieves the highest mean fitness across all three scenarios (313.03, 316.40, and 317.24). This validates the hypothesis that online strategy selection allows the algorithm to dynamically switch between exploration (e.g., `rand/2`) and exploitation (e.g., `rand-to-best/2`) as needed. This adaptability is particularly valuable in the windy scenario, where it outperformed standard `rand/2` and `best/2`.

Robustness to Environmental Noise In the "Wind and Gravity" scenario (Table 2), performance degradation was minimal for the robust algorithms. Remarkably, the `Adaptive Mixed DE` maintained a higher mean fitness (317.24) compared to the normal discrete scenario (316.40), likely because the environmental noise prevented the population from stagnating in shallow local optima. However, the greedy `best/1` strategy suffered significantly, highlighting the risk of over-exploitation in stochastic environments.

5 Application of DE in Practice

Our experiments on the Lunar Lander demonstrated that standard DE can struggle without dynamic parameter control. This observation is consistent with recent findings in engineering domains, particularly in the study *"Parameter's extraction of solar photovoltaic models using an improved differential evolution algorithm"* by Kharchouf et al. [7].

This study addresses the nonlinear parameter estimation of solar photovoltaic (PV) cell models, specifically the single-diode model (SDM) and double-diode model (DDM). Accurate extraction of these electrical parameters is critical for simulating the I–V characteristics that determine solar efficiency. Similar to the Lunar Lander controller, this problem defines a complex, nonlinear search space where the objective is to minimize the Root Mean Squared Error (RMSE) between experimentally measured data and predicted IV curves. Performance is reported in terms of convergence curves and final RMSE values for the different DE variants.

Depending on the model, the problem is five-dimensional (SDM) or seven-dimensional (DDM). The authors implemented a DE/best/1/bin strategy but, crucially, addressed the sensitivity of DE control parameters—a challenge we also encountered. In this configuration, "/bin" denotes binomial crossover with a fixed crossover rate CR, and no alternative crossover operators are explored. While our approach utilized online adaptation (Adaptive Mixed DE), Kharchouf et al. introduced a meta-heuristic DE (MSDE) variant that pre-optimizes the mutation factor (F) and crossover rate (CR) for each dataset before the main optimisation.

The results parallel our findings: the "improved" MSDE variant improved convergence speed by approximately 50% compared to standard DE and consistently achieved lower RMSE values. This external study validates our core conclusion that while DE is a powerful global optimizer, its application to real-world nonlinear tasks relies heavily on mechanisms—whether online adaptation or heuristic pre-tuning—that mitigate the rigidity of fixed control parameters.

In our view, the main limitation of their approach is that it relies on an offline pre-optimisation of F and CR for each dataset, which is less flexible and more expensive than online adaptation and would not transfer well to strongly stochastic settings like our Lunar Lander experiments. In addition, they only consider a single mutation/crossover combination (DE/best/1/bin) and mainly compare against standard DE. A natural extension would be to test adaptive strategies such as our Adaptive Mixed DE on the same PV datasets, include alternative mutation and crossover operators, and assess robustness under controlled measurement noise.

6 Conclusion

In this study, we benchmarked evolutionary strategies and differential evolution variants on the high-dimensional Lunar Lander control task. Our results demonstrate a clear hierarchy: population-based methods significantly outperform trajectory-based (1+1)-ES baselines, which struggled to escape local optima in the 300-dimensional search space.

Most notably, the Adaptive Mixed DE emerged as the most robust algorithm. By dynamically selecting mutation strategies based on their online success rates, it achieved the highest stability in stochastic (windy) environments, mitigating the risk of catastrophic failures observed in aggressive strategies like best/1. This confirms that for black-box control problems with environmental noise, adaptive mechanisms are superior to static parameter settings. These findings mirror recent advancements in engineering domains, such as photovoltaic parameter extraction, where enhanced DE variants similarly outperform standard baselines.

References

1. Auger, A., Brockhoff, D., Hansen, N.: Mirrored sampling and sequential selection for evolution strategies. In: Parallel Problem Solving from Nature (PPSN XI). pp. 11–21. Springer (2010)

2. Auger, A., Hansen, N.: A restart cma evolution strategy with increasing population size. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation. vol. 2, pp. 1769–1776. IEEE (2005). `https://doi.org/10.1109/CEC.2005.1554902`
3. Beyer, H.G.: The Theory of Evolution Strategies. Springer-Verlag, Berlin Heidelberg (2001)
4. Beyer, H.G., Schwefel, H.P.: Evolution strategies: A comprehensive introduction. Natural Computing **1**(1), 3–52 (2002)
5. Brest, J., Greiner, S., Boškovic, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation **10**(6), 646–657 (2006). `https://doi.org/10.1109/TEVC.2006.872133`
6. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin Heidelberg, 2nd edn. (2015)
7. Kharchouf, Y., Herbazi, R., Chahboun, A.: Parameter's extraction of solar photovoltaic models using an improved differential evolution algorithm. Energy Conversion and Management **251**, 114972 (2022)
8. Mitchell, M., Taylor, C.E.: Evolutionary computation: An overview. Annual Review of Ecology and Systematics **30**, 593–616 (1999)
9. Qin, A.K., Suganthan, P.N., Zhang, Q.: Self-adaptive differential evolution algorithm for global optimization problems. IEEE Transactions on Evolutionary Computation **13**(2), 371–391 (2009)
10. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
11. Schwefel, H.P.: Evolution and Optimum Seeking. Wiley-Interscience, New York (1995)
12. Storn, R., Price, K.: Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-012, International Computer Science Institute, Berkeley, CA (1995), `https://icsi.berkeley.edu/`
13. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4), 341–359 (1997)
14. Towers, M., Kwiatkowski, A., Terry, J., Balis, J.U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Tan, H., Younis, O.G.: Gymnasium: A standardized interface for reinforcement learning environments. arXiv preprint arXiv:2407.17032 (2025), `https://arxiv.org/abs/2407.17032`