

# QUASAR: An Evolutionary Algorithm to Accelerate High-Dimensional Optimization

Julian Soltes  
jsoltes@regis.edu  
Regis University  
Denver, Colorado, USA

## Abstract

High-dimensional numerical optimization presents a persistent challenge. This paper introduces Quasi-Adaptive Search with Asymptotic Reinitialization (QUASAR), an evolutionary algorithm to accelerate convergence in complex, non-differentiable problems afflicted by the curse of dimensionality.

Evaluated on the notoriously difficult CEC2017 benchmark suite of 29 functions, QUASAR achieved the lowest overall rank sum (150) using the Friedman test, significantly outperforming L-SHADE (229) and standard DE (305) in the dimension-variant trials. QUASAR also proves computationally efficient, with run times averaging  $1.4\times$  faster than DE and  $7.8\times$  faster than L-SHADE ( $p \ll 0.001$ ) in the population-variant trials.

Building upon Differential Evolution (DE), QUASAR introduces a highly stochastic architecture to dynamically balance exploration and exploitation. Inspired by the probabilistic behavior of quantum particles in a stellar core, the algorithm implements three primary components that augment standard DE mechanisms: 1) probabilistically selected mutation strategies and scaling factors; 2) rank-based crossover rates; 3) asymptotically decaying reinitialization that leverages a covariance matrix of the best solutions to introduce high-quality genetic diversity.

QUASAR's performance establishes it as an effective, user-friendly optimizer for complex high-dimensional problems.

## 1 Introduction

High-dimensional numerical optimization remains a persistent challenge across diverse fields, including machine learning, science, engineering, and finance [9]. This challenge stems from the curse of dimensionality; the exponential growth of the search space renders gradient-based and parametric search methods ineffective [6].

Consequently, optimization metaheuristics, particularly evolutionary algorithms, have become essential for locating optimal solutions in complex function landscapes.

Among these methods, Differential Evolution (DE) has established itself as a robust and highly effective optimizer due to its simplicity, efficiency, and effective mutation strategies [8]. However, standard DE variants can struggle to maintain an effective balance between global exploration and local exploitation, particularly in high-dimensional search spaces. This often leads to stagnation or premature convergence, requiring frequent restarts or extensive hyperparameter tuning; a significant practical drawback for real-world applications.

To address these limitations, this paper proposes Quasi-Adaptive Search with Asymptotic Reinitialization (QUASAR), an evolutionary algorithm designed to dynamically balance the exploration-exploitation trade-off and achieve superior performance in higher dimensions. This method is conceptually inspired by the search for stability in a stellar core, where quantum particles probabilistically explore configurations searching for the most stable state.

QUASAR extends the DE framework by incorporating a highly stochastic multi-strategy approach, including probabilistically selected mutation strategies and factors, as well as an adaptive crossover rate based on solution fitness. The algorithm's primary innovation is an asymptotically decaying reinitialization mechanism, utilizing the spatial covariance matrix of top-performing solutions to inject high-quality genetic diversity into the population.

The efficacy of QUASAR is demonstrated through rigorous testing against standard DE and its state-of-the-art variant L-SHADE across the complete CEC2017 benchmark suite. The results show that QUASAR achieves large and statistically significant performance gains, achieving an overall Friedman rank sum of 150, far below L-SHADE (229) or DE (305). These improvements are achieved using default parameters, highlighting QUASAR's ease of use.

The remainder of this paper is organized as follows: Section 2 details the QUASAR algorithm, including its mutation, crossover, selection, and reinitialization mechanisms. Section 3 defines the experimental design, and Section 4 discusses the experimental results compared to DE and L-SHADE. Section 5 details the ease of use and implementation of the algorithm. Section 6 concludes the work and Section 7 suggests future work.

## 2 QUASAR Algorithm

Quasi-Adaptive Search with Asymptotic Reinitialization (QUASAR) operates using a highly stochastic, multi-strategy approach.

### 2.1 Initial Population

The initial population is generated as a Sobol sequence by default, due to its low discrepancy and superior uniformity in high-dimensional spaces compared to other Quasi-Monte Carlo sampling methods [1]. The default population size,  $N$ , is set to  $10 \times D$  (where  $D$  is the number of dimensions).

Additional initialization methods are available, including uniform Latin Hypercube and Random distributions, as well as exploitative Hyperellipsoid Density sequences [7].

## 2.2 Mutation

Solutions are recombined with one another, analogous to particles sharing information via quantum entanglement. This 'spooky action at a distance' (A. Einstein, 1947), serves as the conceptual basis for the mutation mechanisms below.

QUASAR uses three distinct mutation strategies, each being a modified Differential Evolution variant. These strategies are selected dynamically based on the specified entanglement probability, *entangle rate*. This value defaults to 0.33, resulting in the three strategies being applied equally across the population.

**2.2.1 Local Mutation (Exploitation).** The local mutation strategy, '*Spooky-Best*', is chosen with probability *entangle rate*. This strategy is similar to the DE/best/1 format, exploiting the immediate search space around the current best solution  $\mathbf{X}_{\text{best},g}$ .

$$\mathbf{v}_{i,g} = \mathbf{X}_{\text{best},g} + F_{\text{local}}(\mathbf{X}_{i,g} - \mathbf{X}_{\text{rand},g}) \quad (\text{Spooky-Best})$$

The associated mutation factor  $F_{\text{local}}$  is sampled from a normal distribution  $\mathcal{N}(0, 0.33^2)$ , allowing for small bi-directional perturbations about the best solution.

**2.2.2 Global Mutations (Exploration).** Two global mutation strategies are applied to the remaining  $(1 - \text{entangle rate})$  fraction of the population. Promoting global exploration, these strategies, '*Spooky-Current*' and '*Spooky-Random*', each have a 50% probability of being selected:

$$\mathbf{v}_{i,g} = \mathbf{X}_{i,g} + F_{\text{global}}(\mathbf{X}_{\text{best},g} - \mathbf{X}_{\text{rand},g}) \quad (\text{Spooky-Current})$$

$$\mathbf{v}_{i,g} = \mathbf{X}_{\text{rand},g} + F_{\text{global}}(\mathbf{X}_{i,g} - \mathbf{X}_{\text{rand},g}) \quad (\text{Spooky-Random})$$

The mutation factor  $F_{\text{global}}$  is sampled from a bimodal distribution  $\mathcal{N}(0.5, 0.25^2) + \mathcal{N}(-0.5, 0.25^2)$  to drive non-local exploration. It is worth clarifying that the '*Spooky-Random*' strategy uses the same random vector  $\mathbf{X}_{\text{rand}}$  twice.

The terms used in the three mutation strategies are defined as:

- $\mathbf{v}_{i,g}$ : The mutant vector for the  $i$ -th individual at generation  $g$ .
- $\mathbf{X}_{i,g}$ : The current solution vector.
- $\mathbf{X}_{\text{best},g}$ : The best solution vector known in the population.
- $\mathbf{X}_{\text{rand},g}$ : A randomly chosen solution vector.
- $F_{\text{local}}$  and  $F_{\text{global}}$ : The corresponding mutation factors.

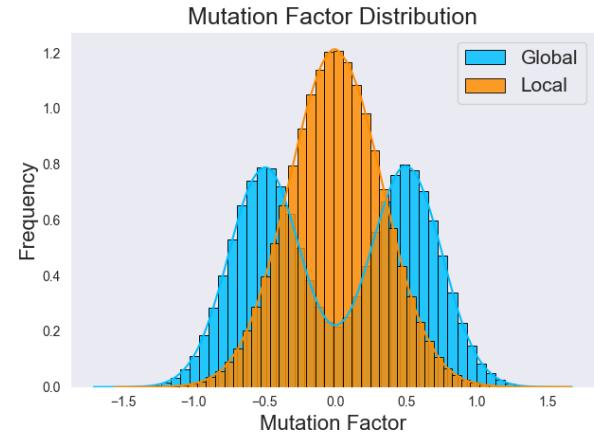
## 2.3 Crossover

QUASAR uses binomial crossover with a dynamic crossover rate ( $CR_i$ ), inversely proportional to the solution's fitness rank. This mechanism ensures that unstable solutions have a higher chance of inheriting components from the new mutant vector  $\mathbf{v}_i$ .

The raw crossover rate ( $CR_{\text{raw},i}$ ) for the  $i$ -th solution is calculated based on its rank ( $\text{rank}_i$ ), where  $\text{rank} = 1$  corresponds to the best solution (lowest fitness). The final dynamic crossover rate  $CR_i$  is capped at a minimum of 0.33:

$$CR_{\text{raw},i} = \frac{\max\_rank - \text{rank}_i}{\max\_rank} = \frac{N - 1 - \text{rank}_i}{N - 1} \quad (1)$$

$$CR_i = \max(CR_{\text{raw},i}, 0.33) \quad (2)$$



**Figure 1: Mutation factor distributions for global and local mutation strategies.**

The trial vector  $\mathbf{u}_i$  is then generated component-wise ( $n$  dimension) using binomial crossover, comparing a uniform random number  $\text{rand}(0, 1)$  against  $CR_i$ :

$$u_{i,n} = \begin{cases} v_{i,n}, & \text{if } \text{rand}(0, 1) \leq CR_i \\ X_{i,n}, & \text{if } \text{rand}(0, 1) > CR_i \end{cases} \quad (3)$$

## 2.4 Selection

A greedy elitism strategy is used to determine the next generation's population, where the target vector for generation  $g+1$  is the better of the two solutions between the current vector  $\mathbf{X}_{i,g}$  and the trial vector  $\mathbf{u}_{i,g}$ :

$$X_{i,(g+1)} = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) < f(\mathbf{X}_i) \\ \mathbf{X}_i, & \text{if } f(\mathbf{u}_i) \geq f(\mathbf{X}_i) \end{cases} \quad (4)$$

## 2.5 Asymptotic Reinitialization

The covariance-guided vector reinitialization applies to a fixed 33% of the worst-performing solutions. These solutions have a high chance in early generations of 'tunneling' to more promising regions.

The probability of reinitialization  $P_{\text{reinit}}(g)$  decays asymptotically from  $P_{\text{reinit}}(0) = 1$  toward a target final probability, calculated using the current generation  $g$ :

$$P_{\text{reinit}}(g) = \exp\left(\frac{\ln(P_{\text{final}})}{g_{\text{final}} g_{\text{max}}} \cdot g\right) \quad (5)$$

where  $g_{\text{max}}$  is the maximum generation count,  $P_{\text{final}} = 0.33$  is the target final probability, and  $g_{\text{final}} = 0.33$  defines the fraction of generations at which  $P_{\text{reinit}}$  reaches  $P_{\text{final}}$ . This ensures the probability drops to  $P_{\text{final}}$  at  $g = g_{\text{max}} \cdot g_{\text{final}}$ .

When a reinitialization occurs, new positions are sampled, informed by the covariance of top-performing individuals. The best  $M$  solutions (where  $M = 25\%$  of the population) are used to calculate the mean vector  $\mu$  and the covariance matrix  $\Sigma$ :

$$\mu = \frac{1}{M} \sum_{i=1}^M \mathbf{x}_i \quad (6)$$

$$\Sigma = \frac{1}{M-1} \sum_{i=1}^M (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T + \epsilon \mathbf{I} \quad (7)$$

where  $\epsilon$  is a small scalar ( $10^{-12}$ ) and  $\mathbf{I}$  is the identity matrix, added to prevent  $\Sigma$  from being singular.

New vectors  $\mathbf{y}$  are generated by sampling from a multivariate normal distribution, with noise added to further promote exploration.

$$\mathbf{y} \sim \mathcal{N}(\mu, \Sigma) + \delta \quad (8)$$

The noise vector is  $\delta$  is sampled from a multivariate normal distribution, with respect to the search bounds:

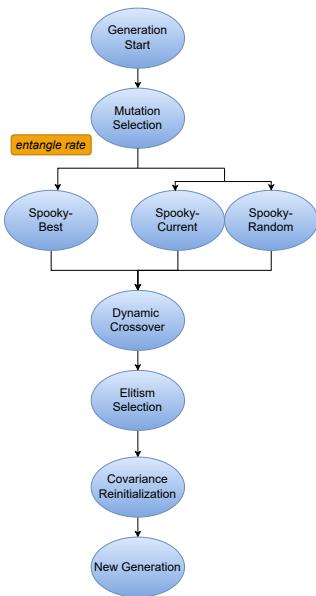
$$\delta \sim \mathcal{N}\left(0, \left(\frac{\mathbf{b}_{\text{high},n} - \mathbf{b}_{\text{low},n}}{20}\right)^2 \mathbf{I}\right) \quad (9)$$

Finally, all components of the new solution  $\mathbf{X}_{\text{new}}$  are clipped to ensure they remain within the defined search bounds  $\mathbf{B} = [\mathbf{b}_{\text{low}}, \mathbf{b}_{\text{high}}]$ .

$$\mathbf{X}_{\text{new},n} = \text{clip}(\mathbf{y}_n, \mathbf{B}_n) \quad (10)$$

These new solutions completely replace the reinitialized solutions and are not subjected to crossover mechanisms.

Due to the  $O(N)$  scaling of the covariance matrix calculations, along with decaying probability of occurrence, the additional overhead becomes negligible compared to the objective function evaluations.



**Figure 2: QUASAR workflow for evolving each generation.**

### 3 Experimental Design

#### 3.1 Comparison Optimizers & Test Functions

The performance of QUASAR was benchmarked against modern implementations of Differential Evolution (DE) and L-SHADE, specifically SciPy's DE and MealPy's L-SHADE.

This choice is justified by the fact that the QUASAR algorithm is built on the DE framework, as well as DE's standing as a robust and effective optimizer [5]. L-SHADE was specifically chosen due to its success in CEC optimization competitions [4].

The trials were conducted using all 29 functions of the CEC2017 numerical optimization benchmark suite.

#### 3.2 Experimental Setup

To ensure a fair comparison, each optimizer was run using their default hyperparameters. The initial population generation methods were also left to defaults; the tested sample sizes not being powers of 2 diminishes the uniformity of QUASAR's initial Sobol sequence [2], providing a theoretical initial handicap.

Since SciPy's DE implementation internally scales the input population size by dimensionality ( $\text{popsize} \times D$  when all dimensions have equal bounds), a correction factor of  $1/D$  is applied to its initial sample size.

The computational budget for metaheuristic comparison is often defined by the maximum number of objective function evaluations ( $FES_{max}$ ) [3]. Due to L-SHADE's linear population reduction, the  $FES_{max}$  becomes a difficult comparison metric to compare against. As such, the primary comparisons are performed on final solution fitness errors and overall optimization times, given the same number of generations, using modern implementations via SciPy's DE and MealPy's L-SHADE.

By default, the L-SHADE implementation features tolerance parameters for early convergence, and the QUASAR and DE implementations further include a final polishing step using L-BFGS-B minimization. These were all disabled for the trials to ensure the optimizers fully utilize their computational budget, terminating only when reaching the maximum number of generations  $g_{max}$ .

#### 3.3 Experimental Trials:

The trials were structured into two distinct phases to assess QUASAR's behavior under varying dimensionalities and population sizes.

The results are compared based on the overall aggregated rank sums, geometric mean error factors ( $GMERF$ ), and optimization run times.

*Dimension-Variant Trials:* These trials assessed QUASAR's scalability by varying the problem dimension  $D$  with a fixed population size ( $N = 1000$ ).

The dimension-variant experiments were run with 30 independent trials for each function/dimension combination to ensure statistical significance.

The dimensions were varied between  $[10, 30, 50, 100]$ , resulting in a total of 10,400 full optimization runs; 3,480 for each algorithm. The computational budget was kept constant, with maximum number of iterations  $g_{max} = 100$  and sample size  $N = 1000$ .

*Sample-Variant Trials:* These trials assessed the impact of population size on performance in high dimensions (a fixed  $D = 100$ ).

**Table 1: Dimension-Variant Test Parameters**

Parameter	Test Values
Dimensions ( $D$ )	[10, 30, 50, 100]
Population Size ( $N$ )	1000
Generations ( $g_{\max}$ )	100

The computational budget is varied between population sizes of [100, 250, 500, 1000, 5000], with 30 independent trials performed for each sample size / test function combination using maximum number of iterations  $g_{\max} = 100$ .

The  $N = 5000$  trials, however, were only run for 10 trials to ensure statistical significance while mitigating the high computational overhead, resulting in a total of 11,310 full optimization runs; 3770 for each algorithm.

**Table 2: Sample-Variant Test Parameters**

Parameter	Test Values
Dimensions ( $D$ )	100
Population Size ( $N$ )	[100, 250, 500, 1000, 5000]
Generations ( $g_{\max}$ )	100

### 3.4 Statistical Analysis:

*Overall Rank:* To draw the overall statistical comparison between the optimizers across all function/dimension and function/sample size combinations, the Friedman test was used.

For each unique combination, the median final error value ( $f$ ) across all independent trials was calculated for each optimizer. Optimizers were ranked based on these median errors, with the best performance (lowest error) receiving a rank of 1.

The ranks were summed across all scenarios to produce the aggregated rank sum for each optimizer. A lower sum indicates superior overall performance.

*Solution Quality Improvements:* The primary comparison metric used to quantify the magnitude of performance difference is the Geometric Mean Error Reduction Factor  $GMERF$ , which provides an unbiased measure of relative performance across all  $N_{\text{trials}}$  independent runs for a given scenario:

$$GMERF = \left( \prod_{i=1}^{N_{\text{trials}}} \frac{f_{\text{comparison},i}}{f_{\text{QUASAR},i}} \right)^{\frac{1}{N_{\text{trials}}}}$$

Here,  $f_{\text{comparison},i}$  and  $f_{\text{QUASAR},i}$  are the final error values achieved by the competitor algorithm (e.g., DE or L-SHADE) and QUASAR, respectively, in the  $i$ -th trial.  $GMERF > 1$  indicates that QUASAR performed better than the comparison optimizer.

For the final aggregated results, the Overall  $GMERF$  is calculated as the geometric mean of the  $GMERF$  values obtained for each function/dimension combination.

$$GMERF_{\text{Overall}} = \left( \prod_{j=1}^{N_{\text{combinations}}} GMERF_j \right)^{\frac{1}{N_{\text{combinations}}}}$$

*Computational Efficiency:* The computational efficiency of the optimizers is quantified by comparing their average optimization run times.

The ratios of run times  $R_t$  for QUASAR vs. DE & L-SHADE are shown in Table 7 & Table 8, by dimension and sample size respectively.  $R_t$  is calculated by dividing the average run time of the comparison method (Mean Sec<sub>comparison</sub>) by the average run time of QUASAR (Mean Sec<sub>QUASAR</sub>) across all trials for that specific dimension.

$$R_t = \frac{\text{Mean Run Time}_{\text{comparison},D}}{\text{Mean Run Time}_{\text{QUASAR},D}} \quad (11)$$

The overall efficiency metrics are the overall run time ratios,  $R_{t,\text{overall}}$ , calculated as the arithmetic mean of each paired ratio  $R_t$ .

$$R_{t,\text{overall}} = \frac{1}{|D|} \sum_{D \in D} \left( \frac{1}{|R_{t,D}|} \sum_{i=1}^{|R_{t,D}|} \frac{t_{\text{comparison},i,D}}{t_{\text{QUASAR},i,D}} \right) \quad (12)$$

*Statistical Significance:* For direct comparison within each dimension or sample size, the paired Wilcoxon signed-rank test is used. This non-parametric test was chosen as the final error distributions are on different scales for each function and are assumed not to be normal, and to ensure consistency with the non-parametric ranking methodology.

The Wilcoxon test was conducted on the final error values ( $f$ ) and run times to generate the p-values for each function/dimension combination.

## 4 Results

The primary solution quality metrics are the Friedman aggregated rank sum and the geometric mean error reduction factor  $GMERF$ , quantifying QUASAR's improvements in final solution fitness. The computational efficiency is also measured by comparing full optimization run times.

Since lower errors and run times indicate better performance, a ratio  $> 1$ , for both the  $GMERF$  and run times ratios, indicates that QUASAR demonstrated superior performance.

Similarly, all plots shown are of raw fitness errors and run times; for all plots, lower values indicate higher performance.

### 4.1 Solution Quality

The ranks and geometric means of fitness error  $GMERF$  of the final solution are calculated for both the dimension-variant and sample-variant trials.

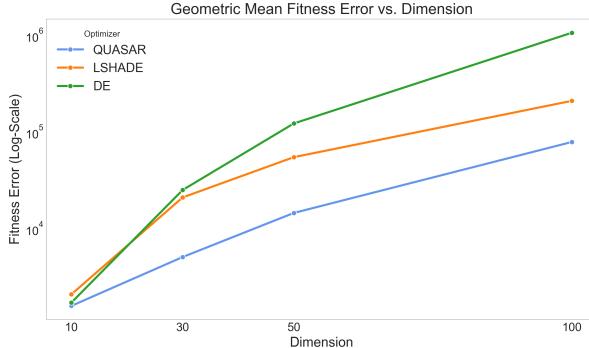
The 95% confidence intervals show consistent performance improvements in almost every scenario tested.

**4.1.1 Dimension-Variant Trials:** The dimension-variant analysis assessed QUASAR's convergence against DE and L-SHADE across four discrete dimensions (10D, 30D, 50D, and 100D), using a fixed sample size  $N = 1000$  and maximum generations  $g_{\max} = 100$ .

The geometric mean error reduction factors ( $GMERF$ ) were consistently high across all dimensions (Fig 3). The 95% confidence intervals validate its consistent improvements; the intervals do not fall below  $1.0 \times GMERF$ .

The largest improvements were seen against DE in 100D, achieving a  $GMERF$  of  $13.52 \times$  with a CI upper bound of  $16.68 \times$ . The

comparison against L-SHADE showed similar improvements; the full set of GMERF values are shown in Table 5.



**Figure 3: Geometric mean error (log-scale) by dimension for each optimizer, with population size of 1000.**

The Friedman test was used on the entire set of unique function/dimension combinations to determine the overall performance hierarchy of the optimizers. The test confirmed a statistically significant improvement in performance ( $p \ll 0.001$ ), shown in Table 3.

**Table 3: Dimension-Variant Rank Sums**

Method	Rank Sum
QUASAR	150
L-SHADE	229
DE	305

**4.1.2 Sample Size-Variant Trials:** To assess the impact of varying population sizes, QUASAR and DE were compared across a fixed  $100D$  and 100 maximum generations  $g_{max}$ , using various sample sizes ( $N$ ).

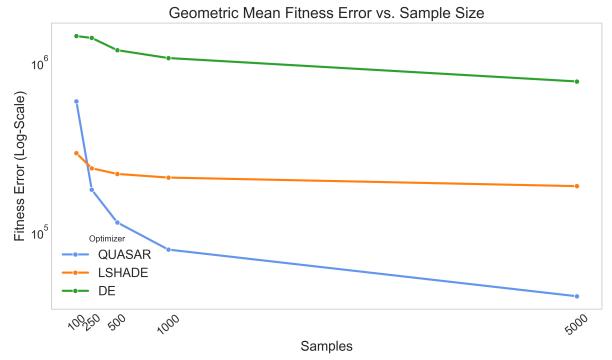
The fitness improvement factors saw the highest gains with higher population sizes; in the  $N = 5000$  case, QUASAR reached a geometric mean error improvement factor GMERF of  $18.5\times$  compared to DE, and  $4.47\times$  compared to L-SHADE.

These performance increases scale super-linearly with population size (Fig. 4, Table 6). The only scenario where QUASAR did not outperform L-SHADE was the  $N = 100$  (low-population) case, where L-SHADE achieved a GMERF of  $0.49\times$ .

The Friedman test is run on the set of all function/population size combinations, shown in Table 4.

**Table 4: Sample-Variant Rank Sums**

Method	Rank Sum
QUASAR	217
L-SHADE	223
DE	430



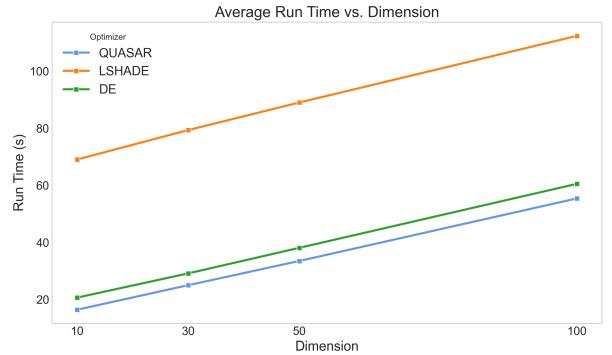
**Figure 4: Geometric mean error (log-scale) by population size for each optimizer, in 100 dimensions.**

## 4.2 Computational Efficiency

The computational efficiency of QUASAR was measured using the overall ratio of run time  $R_{t_{overall}}$  (seconds) relative to DE and L-SHADE.

**4.2.1 Dimension-Variant Trials:** In the dimension-variant trials, QUASAR demonstrated consistently faster run times compared to DE and L-SHADE.

The overall average run time improvement ratios  $R_{t_{overall}}$  were  $1.42\times$  and  $6.29\times$  respectively. The average paired run time ratios  $R_t$  are shown in Table 7, and optimization run times in seconds are in Table 7 and Fig. 5.



**Figure 5: Average optimization run times for each optimizer, by dimension.**

**4.2.2 Sample Size-Variant Trials:** In the sample-variant trials, run times were consistently faster than DE and L-SHADE, averaging  $7.78\times$  and  $1.39\times$  speed increases respectively across all sample-variant trials (Fig. 6, Table 8).

The  $N = 250$  trials, however, demonstrated a ratio  $R_t$  vs DE of  $0.84\times$ , indicating that DE was approximately 16% faster.

**Table 5: Dimension-Variant GMERF Results: QUASAR vs L-SHADE and DE**

D	vs L-SHADE		vs DE		p-value	p-value	QUASAR	L-SHADE	DE
	GMERF	95% CI	GMERF	95% CI	(vs L-SHADE)	(vs DE)	(GM Error)	(GM Error)	(GM Error)
10	1.30	(1.20, 1.40)	1.06	(1.01, 1.12)	$7 \times 10^{-16}$	$7 \times 10^{-3}$	$1.59 \times 10^3$	$2.10 \times 10^3$	$1.72 \times 10^3$
30	4.14	(3.62, 4.74)	4.94	(4.24, 5.77)	$5 \times 10^{-91}$	$3 \times 10^{-108}$	$5.10 \times 10^3$	$2.11 \times 10^4$	$2.52 \times 10^4$
50	3.78	(3.32, 4.31)	8.44	(7.09, 10.04)	$9 \times 10^{-79}$	$5 \times 10^{-134}$	$1.46 \times 10^4$	$5.51 \times 10^4$	$1.23 \times 10^5$
100	2.67	(2.32, 3.06)	13.52	(10.95, 16.68)	$2 \times 10^{-36}$	$2 \times 10^{-141}$	$7.917 \times 10^4$	$2.11 \times 10^5$	$1.07 \times 10^6$

**Table 6: Sample Size-Variant GMERF Results: QUASAR vs L-SHADE & DE**

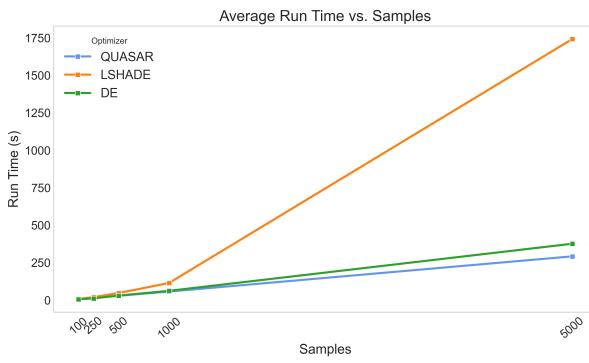
S	vs L-SHADE		vs DE		p-value	p-value	QUASAR	L-SHADE	DE
	GMERF	95% CI	GMERF	95% CI	(vs L-SHADE)	(vs DE)	(GM Error)	(GM Error)	(GM Error)
100	0.49	(0.44, 0.56)	2.43	(2.04, 2.89)	$7 \times 10^{-86}$	$3 \times 10^{-46}$	$5.94 \times 10^5$	$2.94 \times 10^5$	$1.44 \times 10^6$
250	1.34	(1.19, 1.50)	7.86	(6.51, 9.49)	$3.3 \times 10^{-2}$	$3 \times 10^{-141}$	$1.79 \times 10^5$	$2.39 \times 10^5$	$1.41 \times 10^6$
500	1.93	(1.70, 2.20)	10.40	(8.54, 12.67)	$2 \times 10^{-8}$	$2 \times 10^{-140}$	$1.15 \times 10^5$	$2.21 \times 10^5$	$1.19 \times 10^6$
1000	2.66	(2.32, 3.06)	13.52	(10.95, 16.68)	$5 \times 10^{-34}$	$2 \times 10^{-141}$	$7.92 \times 10^4$	$2.11 \times 10^5$	$1.07 \times 10^6$
5000	4.47	(3.41, 5.86)	18.54	(12.49, 27.53)	$3 \times 10^{-37}$	$3 \times 10^{-48}$	$4.20 \times 10^4$	$1.88 \times 10^5$	$7.78 \times 10^5$

**Table 7: Dimension-Variant Run Times: QUASAR vs L-SHADE & DE**

D	R <sub>t</sub>	R <sub>t</sub>	p-value	p-value	QUASAR	L-SHADE	DE
	vs L-SHADE	vs DE	(vs L-SHADE)	(vs DE)	(sec)	(sec)	(sec)
10	4.21	1.26	$5 \times 10^{-144}$	$7.3 \times 10^{-141}$	16.4	69.1	20.7
30	3.17	1.17	$5 \times 10^{-144}$	$1.1 \times 10^{-139}$	25.0	79.4	29.2
50	2.66	1.14	$5 \times 10^{-144}$	$1.6 \times 10^{-137}$	33.5	89.0	38.1
100	2.03	1.09	$5 \times 10^{-144}$	$3 \times 10^{-137}$	55.4	112.4	60.5

**Table 8: Sample Size-Variant Run Times: QUASAR vs L-SHADE & DE**

S	R <sub>t</sub>	R <sub>t</sub>	p-value	p-value	QUASAR	L-SHADE	DE
	vs L-SHADE	vs DE	(vs L-SHADE)	(vs DE)	(sec)	(sec)	(sec)
100	1.37	1.05	$1 \times 10^{-142}$	$1 \times 10^{-84}$	5.8	7.9	6.0
250	1.47	0.84	$1 \times 10^{-142}$	$5 \times 10^{-58}$	13.7	20.1	11.4
500	1.65	1.06	$1 \times 10^{-142}$	$2 \times 10^{-123}$	28.7	47.2	30.5
1000	2.00	1.08	$1 \times 10^{-142}$	$9 \times 10^{-121}$	57.3	114.4	62.0
5000	5.96	1.29	$3 \times 10^{-49}$	$3 \times 10^{-48}$	292.0	1741.7	376.1



**Figure 6: Average optimization run times for each optimizer, by population size.**

## 5 Ease of Use and Implementation

QUASAR is designed to be robust with minimal parameter dependence, making it effective for most numerical optimization problems.

### 5.1 Ease of Use

A key feature of QUASAR is its ability to operate effectively with its default *entangle rate*, bypassing the hyperparameter tuning often required by other metaheuristics such as DE and L-SHADE.

The algorithm relies on its quasi-adaptive mechanisms to dynamically balance exploration and exploitation. For typical use cases, the only inputs needed are the objective function, search bounds, and the target number of generations ( $g_{\max}$ ).

## 5.2 Implementation

QUASAR is implemented as part of a streamlined open-source Python package, designed for seamless integration into existing workflows. The syntax follows the structure of SciPy's optimization module, making it nearly one-to-one interchangeable with its Differential Evolution.

The source code requires minimal dependencies, relying only on NumPy and SciPy. The open-source package containing QUASAR, `hdim-opt`, is available via the Python Package Index (PyPI) using '`pip install hdim_opt`', or at [github.com/jgsoltes/hdim-opt](https://github.com/jgsoltes/hdim-opt) which includes the experimental notebooks for full reproducibility.

## 6 Conclusion

This study introduced Quasi-Adaptive Search with Asymptotic Reinitialization (QUASAR), a quantum-inspired evolutionary algorithm designed to efficiently explore and exploit high-dimensional spaces.

The algorithm utilizes a highly stochastic architecture, including probabilistic mutation mechanisms, rank-based crossover rates, and asymptotically decaying covariance reinitializations.

Experimental trials across the CEC2017 benchmark suite confirm that QUASAR delivers exceptional performance compared to Differential Evolution (DE) and L-SHADE, consistently ranking best via the Friedman test.

These ranks are supported by the geometric mean error reduction factors (*GMERF*), showing consistently improved performance across all dimensions, with *GMERF* values ranging from  $(1.01, 16.68)\times$ . Analysis by population size shows super-linear improvements for all population sizes  $N > 100$ , reaching up to  $27.53\times$  *GMERF* improvements over DE.

The computational efficiency of the algorithm is a key benefit, with run times consistently lower than the comparison optimizers ( $R_t > 1$ ), scaling positively with population size and dimensionality.

The empirical evidence validates QUASAR as a powerful evolutionary algorithm, ready for the next generation of high-dimensional optimization problems.

## 7 Future Works

### 7.1 Enhancements:

QUASAR's quasi-adaptive mechanisms are highly effective; future improvements will explore fully self-adaptive strategies, similar to those found in other modern DE variants.

The current covariance matrix ( $\Sigma$ ) treats all top solutions equally; future work will investigate applying weights to the very best solutions. New methods for defining the probability of occurrence may be explored, such as sinusoidal or sigmoid activation behavior.

Crossover rate is currently defined by fitness rank, and may benefit from a probability-based approach, similar to the other QUASAR components, via soft-max or similar methods.

### 7.2 Verification:

QUASAR's highly stochastic framework makes it a practical choice for non-differentiable and non-parametric problems. The algorithms' robustness would be further validated by rigorously evaluating its performance on more constrained, non-parametric problems.

## Acknowledgments

The author would like to thank Dr. Kellen Sorauf, Dr. Ksenia Polson, Dr. Mike Busch, and Monet Morris for their invaluable feedback and support throughout this study. It would not have been possible without the insights gained from each of our discussions.

Dr. Sorauf's recommendation to include L-SHADE in the experiment significantly improved the rigor of the study; Dr. Polson's lectures laid the framework for the statistical analysis; Dr. Busch pushed me to pursue the project to begin with. Monet remains a source of inspiration, as always.

This research was conducted independently and was not part of a formal project.

## References

- [1] Nicolas Bonneel, David Coeurjolly, Jean-Charles Iehl, and Victor Ostromoukhov. 2025. Sobol' Sequences with Guaranteed-Quality 2D Projections. *ACM Transactions on Graphics* 44 (2025), 1–16. Issue 4. doi:10.1145/3730821
- [2] Stephen Joe and F. Y. Kuo. 2008. Constructing Sobol Sequences with Better Two-Dimensional Projections. *SIAM Journal on Scientific Computing* 30 (2008), 2635–2654. Issue 5. doi:10.1137/070709359
- [3] Alena Kazikova, Marek Pluhacek, and Roman Senkerik. 2021. How Does the Number of Objective Function Evaluations Impact Our Understanding of Metaheuristics Behavior? *IEEE Access* 9 (2021), 44032–44048. doi:10.1109/ACCESS.2021.3066135
- [4] Z. W. Mo, H. L. Liu, B. D. Weng, and G. M. Lei. 2019. Efficient surrogate-assisted evolutionary optimization algorithm for expensive high-dimensional problems. *Information Sciences* 476 (2019), 42–56. doi:10.1016/j.ins.2018.10.052
- [5] Kenneth V. Price. 1996. Differential evolution: a fast and simple numerical optimizer. In *Proceedings of North American Fuzzy Information Processing*. Berkeley, CA, USA, 524–527. doi:10.1109/NAFIPS.1996.534790
- [6] Shih-Hsien Shan and G. Gary Wang. 2008. Survey of Modeling and Optimization Strategies for High-Dimensional Design Problems. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. doi:10.2514/6.2008-5842
- [7] J. G. Soltes. 2025. Hyperellipsoid Density Sampling: Exploitative Sequences to Accelerate High-Dimensional Optimization. *arXiv preprint arXiv:2511.07836* (2025). arXiv:2511.07836
- [8] Rainer Storn. 1996. On the usage of differential evolution for function optimization. In *Proceedings of North American Fuzzy Information Processing*. Berkeley, CA, USA, 519–523. doi:10.1109/NAFIPS.1996.534789
- [9] Ming Zhou, Man Cui, Dong Xu, Sen Zhu, Zheng Zhao, and Abdullah Abusorrah. 2024. Evolutionary Optimization Methods for High-Dimensional Expensive Problems: A Survey. *IEEE/CAA Journal of Automatica Sinica* 11 (May 2024), 1092–1105. Issue 5. doi:10.1109/JAS.2024.124320