LLM-GUIDED EVOLUTIONARY PROGRAM SYNTHESIS FOR QUASI-MONTE CARLO DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-discrepancy point sets and digital sequences underpin quasi-Monte Carlo (QMC) methods for high-dimensional integration. We cast two long-standing QMC design problems as program synthesis and solve them with an LLM-guided evolutionary loop that mutates and selects code under task-specific fitness: (i) constructing finite 2D/3D point sets with low star discrepancy, and (ii) choosing Sobol' direction numbers that minimize randomized QMC error on downstream integrands. Our two-phase procedure combines constructive code proposals with iterative numerical refinement. On finite sets, we rediscover known optima in small 2D cases and set new best-known 2D benchmarks for $N \geq 40$, while matching most known 3D optima up to the proven frontier ($N \leq 8$) and reporting improved 3D benchmarks beyond. On digital sequences, evolving Sobol' parameters yields consistent reductions in randomized quasi-Monte Carlo (rQMC) mean-squared error for several 32-dimensional option-pricing tasks relative to widely used Joe–Kuo parameters, while preserving extensibility to any sample size and compatibility with standard randomizations. Taken together, the results demonstrate that LLM-driven evolutionary program synthesis can automate the discovery of high-quality QMC constructions, recovering classical designs where they are optimal and improving them where finite-N structure matters. Data and code are available at `anonymous`.

1 INTRODUCTION

Numerical integration in high dimensions is a cornerstone of modern science and engineering. While standard Monte Carlo (MC) methods offer a robust approach, their convergence rate, governed by the Central Limit Theorem, is often insufficient for applications requiring high precision (Glasserman, 2003). Quasi-Monte Carlo (QMC) methods provide a compelling alternative by replacing pseudorandom samples with deterministic, highly uniform point sets (Dick & Pillichshammer, 2010; Owen, 1995). The uniformity of these sets is quantified by their discrepancy, with lower values corresponding to more evenly distributed points. The Koksma–Hlawka inequality provides the theoretical underpinning for QMC, guaranteeing that the integration error is bounded by the product of the variation of the integrand and the star discrepancy of the point set (Koksma, 1964; Hlawka, 1961).

This relationship has fueled decades of research into the discovery and construction of low-discrepancy sets. A primary challenge in this field is the "*ab initio*" construction of a finite point set of N points in d dimensions that minimizes star discrepancy. This is a problem of combinatorial complexity, and while optimal solutions have been found in 2D for $N \leq 21$ and in 3D for $N \leq 8$ (Clément et al., 2024), the problem remains largely open for larger N and higher dimensions.

A related challenge is the construction of infinite low-discrepancy sequences, such as those by Halton, Hammersley, and Sobol'. Among these, Sobol' sequences are particularly prominent due to their excellent distribution properties and efficient generation (Sobol', 1967). However, their quality is highly dependent on a set of integer parameters known as direction numbers, and finding optimal parameters that ensure uniformity across all low-dimensional projections is a difficult combinatorial search problem (Joe & Kuo, 2008).

Recent breakthroughs in Large Language Models (LLMs) have demonstrated their remarkable capabilities in code generation, logical reasoning, and pattern recognition (Gemini Team et al., 2025). This has motivated the development of automated scientific discovery systems, such as AlphaEvolve, which leverage LLMs to navigate complex search spaces (Novikov et al., 2025). In this work, we utilize the OpenEvolve framework (Sharma, 2025), an open-source implementation based on the principles of AlphaEvolve, to tackle the aforementioned challenges in discrepancy theory. We treat the construction of these mathematical objects as a program synthesis problem within an evolutionary framework (Koza, 1994). The LLM acts as an intelligent mutation operator, iteratively modifying code that generates candidate solutions based on feedback from a fitness function.

This paper presents the successful application of this LLM-driven evolutionary approach to two fundamental problems:

1. Discovering finite 2D and 3D point sets with state-of-the-art low star discrepancy.
2. Searching for superior Sobol' direction numbers to minimize randomized quasi-Monte Carlo (rQMC) integration error in the high-dimensional context of pricing exotic financial options (Paskov & Traub, 1995).

Our results show that this methodology can match and, in several important cases, surpass the best known human-derived solutions, without any task-specific training of the LLM. This suggests that LLM-driven evolutionary search is a promising new paradigm for exploration and discovery in computational mathematics.

2 Theoretical Background

2.1 Star Discrepancy

Star discrepancy is the most common measure for quantifying the uniformity of a point set within the d-dimensional unit hypercube, $[0,1]^d$ (Owen, 1995). It captures the largest deviation between the volume of an axis-aligned "anchor box" and the fraction of points contained within it.

Definition 1 (Star Discrepancy). Let $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of N points in $[0,1]^d$. An anchor box $[\mathbf{0}, \mathbf{q})$ for any $\mathbf{q} = (q_1, \ldots, q_d) \in [0,1]^d$ is the hyperrectangle $[0, q_1) \times \cdots \times [0, q_d)$. The star discrepancy D_N^* of the set P is defined as:

$$D_N^*(P) = \sup_{\mathbf{q} \in [0,1]^d} \left| \frac{\#(P \cap [\mathbf{0}, \mathbf{q}))}{N} - \text{Vol}([\mathbf{0}, \mathbf{q})) \right| \tag{1}$$

Here, the supremum is taken over all possible anchor boxes. A small D_N^* value implies that for any anchor box, the fraction of points falling within it is a good approximation of its volume, indicating high uniformity. The practical importance of star discrepancy for QMC is cemented by the Koksma–Hlawka inequality (Koksma, 1964; Hlawka, 1961), which bounds the error of QMC integration:

$$\left| \int_{[0,1]^d} f(\mathbf{u})\, d\mathbf{u} - \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \right| \leq V(f) \cdot D_N^*(P) \tag{2}$$

where $V(f)$ is the total variation of the function f in the sense of Hardy and Krause (Niederreiter, 1992; Dick & Pillichshammer, 2010). This inequality guarantees that point sets with lower star discrepancy lead to smaller integration error bounds.

2.2 Sobol' Sequences and Direction Numbers

Sobol' sequences are a class of low-discrepancy sequences that are particularly effective for QMC integration (Sobol', 1967; Dick & Pillichshammer, 2010). They are constructed using the properties of primitive polynomials over the finite field of two elements, $\mathbb{F}_2$.

Definition 2 (Sobol' Sequence Construction). For each dimension $j \geq 1$, a primitive polynomial over $\mathbb{F}_2$ of degree s_j is chosen (Dick & Pillichshammer, 2010):

$$P_j(z) = z^{s_j} + a_{1,j} z^{s_j - 1} + \cdots + a_{s_j - 1, j} z + 1 \tag{3}$$

where the coefficients $a_{k,j}$ are either 0 or 1. From this polynomial, a sequence of positive, odd integers called direction numbers $m_{k,j}$ (for $k = 1, \ldots, s_j$) are chosen freely.

For $k > s_j$, subsequent direction numbers (expressed as fixed-point binary fractions $v_{k,j} = m_{k,j}/2^k$) are generated by the canonical Sobol' recurrence (Joe & Kuo, 2008; Dick & Pillichshammer, 2010):

$$v_{k,j} \;=\; a_{1,j}v_{k-1,j} \;\oplus\; a_{2,j}v_{k-2,j} \;\oplus\; \cdots \;\oplus\; a_{s_j-1,j}v_{k-s_j+1,j} \;\oplus\; \big(v_{k-s_j,j} \gg s_j\big) \tag{4}$$

where $\oplus$ denotes bitwise XOR on the binary fixed-point representation and $\gg s_j$ is a right bit-shift by s_j places. For $j = 1$, we set $v_{k,1} = 2^{-k}$ (van der Corput base-2).

The j-th coordinate of the i-th point in the sequence, $x_{i,j}$, is then generated using Gray-code bits:

$$x_{i,j} \;=\; \bigoplus_{k\geq 1} g_k\, v_{k,j}, \qquad \text{with } i = \sum_{k\geq 1} i_k 2^{k-1}, \;\; g \;=\; i \oplus (i \gg 1) \;=\; \sum_{k\geq 1} g_k 2^{k-1}. \tag{5}$$

The quality of the Sobol' sequence, particularly the uniformity of its low-dimensional projections, is critically dependent on the choice of the primitive polynomials and the initial direction numbers $(m_1, \ldots, m_s)$. The work of Joe & Kuo (2008) provides a widely used set of these parameters that serve as a strong baseline. In tables we encode the polynomial coefficients $a_{k,j}$ as an integer A_j (binary bitmask).

3 Related Work

Classical approaches for generating low-discrepancy sets are primarily number-theoretic. Foundational methods include Halton, Hammersley, and Sobol' sequences, which are designed to achieve superior asymptotic uniformity compared to random sampling. While powerful, these classical constructions are not always optimal for a finite number of points N. Mathematical programming has been used to find provably optimal sets, though these approaches are computationally intensive and limited to small instances (Clément et al., 2024). Heuristic methods, such as genetic algorithms and threshold accepting, have been applied to tackle larger instances by searching the space of point configurations (Clément et al., 2023).

More recently, machine learning techniques have been introduced to this domain. Message-Passing Monte Carlo (MPMC) leverages Graph Neural Networks (GNNs) to transform random initial points into low-discrepancy configurations, achieving state-of-the-art results by directly optimizing point coordinates (Rusch et al., 2024). Our work differs by framing the task as a program synthesis problem rather than direct coordinate optimization.

A related line of research focuses on optimizing the parameters of Sobol' sequences. The quality of a Sobol' sequence is critically dependent on a set of initialization parameters known as direction numbers. The direction numbers published by Joe & Kuo (2008) are a widely-used standard, derived from an extensive computational search to find parameters that ensure high uniformity in two-dimensional projections by minimizing a quality measure known as the t-value. Subsequent work has focused on further improving these parameters or guaranteeing quality for specific projection properties crucial for applications such as computer graphics (Bonneel et al., 2025). LatNetBuilder (L'ecuyer & Munger, 2016) generalizes these ideas by searching the parameters of lattice rules and digital nets to minimize user-specified criteria, such as discrepancy bounds or projection-dependent t-values.

Our approach is most closely related to the emerging paradigm of using Large Language Models (LLMs) for automated scientific discovery. Novikov et al. (2025) introduces AlphaEvolve, a framework that combines LLMs with an evolutionary search, treating algorithm discovery as a program evolution problem where the LLM functions as an intelligent mutation operator and receives feedback from a fitness function. This method has successfully discovered novel algorithms for fundamental problems, from matrix multiplication (Fawzi et al., 2022) to open mathematical conjectures (Romera-Paredes et al., 2024). Our work is directly inspired by these principles, applying a similar evolutionary loop to the specific mathematical challenges of discovering low-discrepancy sets and optimizing Sobol' sequences.

4 METHODOLOGY

Our approach is based on the OpenEvolve framework, an open-source implementation of the principles demonstrated by AlphaEvolve (Novikov et al., 2025). It frames the search for novel mathematical constructs as an evolutionary search over a population of programs that generate them (Koza, 1994). The LLM serves as a sophisticated mutation operator, guided by a fitness function.

The evolutionary loop (Romera-Paredes et al., 2024) proceeds as follows:

1. **Initialization:** The process begins with a population of "parent" programs. These programs are code snippets in Python that generate a candidate solution. The initial population can be seeded with simple heuristics or well-known constructions.
2. **Evaluation:** Each program in the population is executed, and its output is evaluated by a fitness function. The fitness function returns a scalar score quantifying the quality of the solution (e.g., lower rQMC MSE or star discrepancy is better).
3. **Selection and Prompting:** High-performing programs are selected to serve as parents. A detailed prompt is then constructed for the LLM, including the parent program's source code, its fitness score, and an instruction tasking the LLM with generating a variation that will improve upon the score. The prompt also includes code from other high-performing "inspirations" to encourage crossover as well as guidance from the user (Appendix A).
4. **Generation (Mutation):** The LLM receives the prompt and generates a new, modified program. This is the core "mutation" step. The LLM's ability to understand code syntax and semantics allows for complex and intelligent modifications.
5. **Loop:** The newly generated program is evaluated, its fitness is scored, and it is added to the population. The process then repeats, iteratively refining the population toward better solutions.

4.1 EXPERIMENTAL SETUP

We used an LLM-guided evolutionary search over a population of Python programs (population 60, four islands with occasional migration). Parents were chosen by fitness with occasional archive sampling; the LLM rewrote core functions each generation. We used Google's Gemini 2.0 Flash as the mutation operator with temperature 0.7 and top-$p = 0.95$. We did not train or fine-tune the LLM in any way; it is used as an off-the-shelf code rewriter. Each evolutionary run used roughly 2000 LLM calls and a fixed budget for fitness evaluations. Full settings, including population structure, archive, migration, and program validation checks, are given in Appendix A.

4.2 DISCOVERY OF LOW-DISCREPANCY POINT SETS

A two-phase strategy was employed to balance broad exploration with fine-tuned optimization.

- **Phase I: Direct Construction:** The LLM was prompted to generate Python code that directly constructs an N-point set in a d-dimensional space. The initial parent program in 2D implemented a simple shifted Fibonacci lattice (Appendix A, Listing 1) and in 3D implemented a scrambled Sobol' sequence (Appendix A, Listing 2). This phase encouraged the LLM to explore a wide range of constructive heuristics.
- **Phase II: Iterative Optimization:** After a sufficient number of iterations, the LLM was prompted to generate Python code that uses iterative optimization routines (e.g., `scipy.optimize.minimize`) to refine an initial guess. This shifted the search from finding explicit constructions to a direct optimization of the point coordinates.
- **Fitness Function:** The fitness score was $\frac{1}{1+D_N^*}$ where D_N^* is the star discrepancy of the generated point set.

We compute D_N^* exactly by scanning extremal anchor boxes on the coordinate-induced grid (overall $\mathcal{O}(N^{d/2+1})$. Implementation details are in Appendix A.

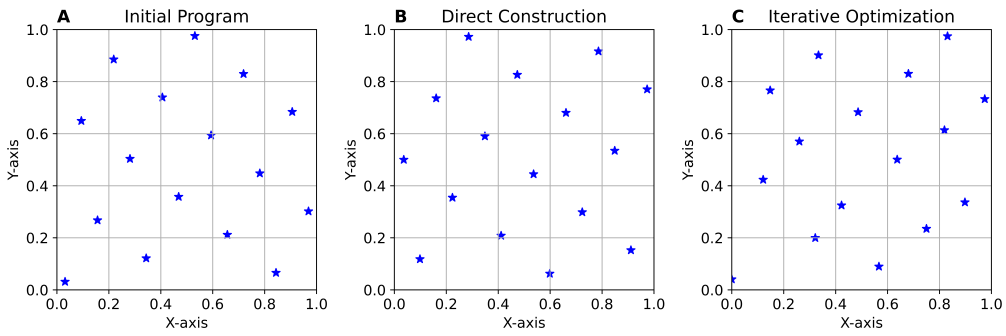

Figure 1: Visualization of $N = 16$ point set generation in two dimensions. (A) Initial shifted Fibonacci lattice (Discrepancy: 0.0962). (B) Best direct construction found in Phase I (Discrepancy: 0.0924). (C) Final optimized point set from Phase II (Discrepancy: 0.0744), which is within 0.68% of the known optimal value of 0.0739.

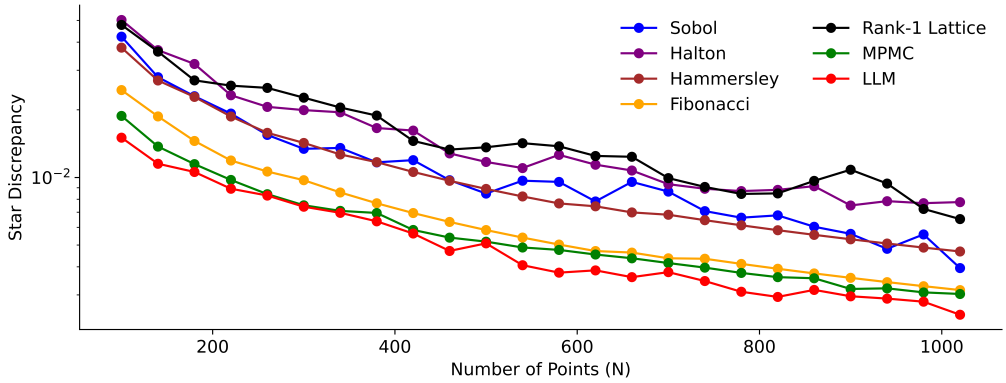

Figure 2: The Star Discrepancy D_N^* of Sobol', Halton, Hammersley, Fibonacci, Rank-1-Lattice, MPMC (message passing Monte Carlo), and LLM-evolved sets for increasing number of points $N = 100 \ldots 1020$ in 2D.

4.3 Discovery of Sobol' Direction Numbers

- **Program Representation:** The evolved programs are Python functions that return a list of dictionaries. Each dictionary contains the Sobol' parameters (`s`, `a`, `m_i`) for a single dimension.

- **Initialization:** The initial population contains the following implementation of the direction numbers (Joe & Kuo, 2008).

- **Fitness Function:** The primary fitness metric is $\frac{1}{1+\text{MSE}}$ where MSE is the mean squared error of an rQMC estimate for a 32-dimensional Asian option price. The MSE is calculated for $N = 8192$ points and averaged over 1000 consistent randomizations (left matrix scramble followed by a Cranley–Patterson random shift, i.e., LMS+shift) to ensure robustness and reproducibility. All rQMC comparisons are paired by using identical randomization seeds per method and N. The diffusion paths are constructed from $[0, 1]^d$ via standard time discretization of geometric Brownian motion with equal timesteps; we report this mapping to clarify effective-dimension effects.

5 Experiments and Results

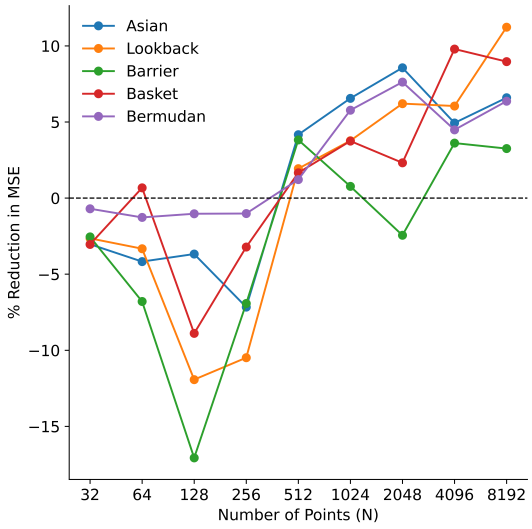

Figure 3: The % reduction in MSE (rQMC integration over 10000 random scrambles and shifts) using Sobol' direction numbers found via LLM evolutionary search vs. those of Joe & Kuo (2008). The % reduction in MSE are averaged across all scenarios of that particular option (Appendix C).

Listing 1 Directly Constructed 16 Point Set ($N = 16$)

```
def construct_star():
    A = np.zeros((N, 2))
    phi=(math.sqrt(5)-1)/2
    for i in range(N):
        A[i, 0]=(i+(1/math.sqrt(3)))/N
        A[i, 1]=((0.5+(i*phi)%1))%1
    return A
```

5.1 Discovery of Low-Discrepancy Point Sets

We first applied our two-phase strategy to the canonical problem of discovering N-point sets in 2D and 3D with minimal star discrepancy. We illustrate the discovery process for a 2D 16-point set (Fig. 1). The initial program, a Fibonacci lattice, had a discrepancy of 0.0962. After 243 iterations in the direct construction phase, a new construction was found with a discrepancy of 0.0924 (Listing 1), which consisted of fine-tuning optimal shifts to the Fibonacci lattice. After switching to the iterative optimization phase, the framework further refined the point set, achieving a final discrepancy of 0.0744, which is within 0.68% of the known optimal value of 0.0739. The final program creates an initial guess, consisting of a randomly jittered Fibonacci lattice, followed by a sequential least squares programming (SLSQP) optimization loop with stochastic restarts (Listing 2).

We then benchmarked our method against Fibonacci, MPMC (Rusch et al., 2024), and known optimal point sets (Clément et al., 2024) for $N = 1 \ldots 100$ (Table 1). Our method successfully rediscovers the known optimal point sets for $N \leq 10$ and remains highly competitive for larger N. The most significant results came from searching for larger point sets where optimal solutions are not known. For 2D point sets with $N > 30$, LLM evolutionary search discovered new configurations with lower star discrepancy than the best-known values from the literature. For instance, for $N = 100$, our method found a point set with a discrepancy of 0.0150, a substantial improvement over the previous best of 0.0188. For $N = 100$ we further compare the full evolutionary loop against a single one-shot prompt and a multi-turn prompting baseline without evolution, as well as against a smaller Gemini Flash-Lite model. Over 16 seeds, evolutionary search with either LLM achieves consistently lower discrepancy with much smaller variance than the prompting-only baselines (Appendix B), showing that population-based evolution to be more robust than repeated prompting.

We generated 2D point sets up to $N = 1020$ with lower star discrepancy than previously reported (Appendix B). In 3D, our method matched the known optimal point sets for $N = 1, 2, 3, 5, 6, 7$

N	Fibonacci	MPMC	LLM	Clément et al.
1	1.0000	**0.6180**	**0.6180**	**0.6180**
2	0.6909	**0.3660**	**0.3660**	**0.3660**
3	0.5880	**0.2847**	**0.2847**	**0.2847**
4	0.4910	**0.2500**	**0.2500**	**0.2500**
5	0.3528	**0.2000**	**0.2000**	**0.2000**
6	0.3183	0.1692	**0.1667**	**0.1667**
7	0.2728	0.1508	**0.1500**	**0.1500**
8	0.2553	0.1354	**0.1328**	**0.1328**
9	0.2270	0.1240	**0.1235**	**0.1235**
10	0.2042	0.1124	**0.1111**	**0.1111**
11	0.1857	0.1058	0.1039	**0.1030**
12	0.1702	0.0975	0.0960	**0.0952**
13	0.1571	0.0908	0.0892	**0.0889**
14	0.1459	0.0853	0.0844	**0.0837**
15	0.1390	0.0794	0.0791	**0.0782**
16	0.1486	0.0768	0.0745	**0.0739**
17	0.1398	0.0731	0.0712	**0.0699**
18	0.1320	0.0699	0.0676	**0.0666**
19	0.1251	0.0668	0.0654	**0.0634**
20	0.1188	0.0640	0.0611	**0.0604**
30	0.0792	N/A	0.0438	**0.0424**
40	0.0638	N/A	**0.0331**	0.0332
50	0.0531	N/A	**0.0278**	0.0280
60	0.0442	0.0273	**0.0234**	0.0244
100	0.0275	0.0188	**0.0150**	0.0193

Table 1: 2D Star Discrepancy Comparison for $N = 1 \ldots 100$ between Fibonacci, MPMC (Message-Passing Monte Carlo), LLM evolutionary search, and Clément et al. (provably optimal for $N \leq 20$). Best values are **bolded**.

Listing 2 Iteratively Optimized 2D Point Set ($N = 16$)

```
def construct_star():
    x = np.zeros((N, 2))
    for i in range(N):
        x[i, 0] = (i + np.random.rand()) / N
        x[i, 1] = ((i * 0.38196601125) % 1) + np.random.rand()/(2*N)
    def discrepancy_wrapper(x):
        points = x.reshape(N, 2)
        return star_discrepancy(points)
    x0 = x.flatten()
    bounds = [(0.0, 1.0)] * (N * 2)
    best_result = None
    best_discrepancy = float('inf')
    for _ in range(25):
        x0_restart = x.flatten() + np.random.normal(0, 0.01, N * 2)
        x0_restart = np.clip(x0_restart, 0.0, 1.0)
        result = minimize(discrepancy_wrapper, x0_restart, method='
            SLSQP', bounds=bounds, options={'maxiter': 30000, 'ftol':
            1e-15, 'iprint': 0})
        discrepancy = discrepancy_wrapper(result.x)
        if discrepancy < best_discrepancy:
            best_discrepancy = discrepancy
            best_result = result
    optimized_points = best_result.x.reshape(N, 2)
    return optimized_points
```

(Table 2) and provided explicit constructions that set new best-known star-discrepancy benchmarks beyond the proven-optimal range for $N > 8$ (Appendix B).

N	1	2	3	4	5	6	7	8
MPMC	0.6833	**0.4239**	0.3491	0.3071	0.2669	0.2371	0.2158	0.1993
LLM	**0.6823**	**0.4239**	**0.3445**	0.3042	**0.2618**	**0.2326**	**0.2090**	0.1937
Clément et al.	**0.6823**	**0.4239**	**0.3445**	**0.3038**	**0.2618**	**0.2326**	**0.2090**	**0.1875**

Table 2: 3D Star Discrepancy for $N = 1 \dots 8$. Lower is better; best per N is **bolded**.

	Training Example			Out-of-the-Money			At-the-Money		
N	Sobol	LLM	p-value	Sobol	LLM	p-value	Sobol	LLM	p-value
32	**0.2484**	0.2523	0.9757	**0.1920**	0.1979	0.9757	**0.3246**	0.3346	0.9757
64	**0.0642**	0.0665	0.9980	**0.0605**	0.0631	0.9980	**0.0873**	0.0908	0.9980
128	**0.0183**	0.0192	0.9999	**0.0220**	0.0231	0.9999	**0.0277**	0.0282	0.9999
256	**0.0056**	0.0061	1.0000	**0.0086**	0.0089	1.0000	**0.0092**	0.0098	1.0000
512	0.001646	**0.001614**	0.2841	0.003469	**0.003311**	**0.0208**	0.003248	**0.003106**	**0.0208**
1024	0.000542	**0.000524**	0.0548	0.001457	**0.001313**	**3.45e-08**	0.001228	**0.001168**	**0.0304**
2048	0.000233	**0.000225**	**0.0199**	0.000619	**0.000535**	**1.45e-14**	0.000550	**0.000521**	**0.0131**
4096	9.876e-05	**9.346e-05**	**0.0058**	0.000258	**0.000243**	**4.51e-04**	0.000240	**0.000226**	**0.0019**
8192	4.523e-05	**4.104e-05**	**7.03e-06**	0.000117	**0.000107**	**1.06e-06**	0.000102	**9.523e-05**	**0.0100**

	In-the-Money			High Volatility			Low Volatility		
N	Sobol	LLM	p-value	Sobol	LLM	p-value	Sobol	LLM	p-value
32	**0.1398**	0.1428	0.9757	**2.0816**	2.1493	0.9757	**0.0238**	0.0246	0.9757
64	**0.0367**	0.0382	0.9980	**0.5714**	0.5958	0.9980	**0.0066**	0.0068	0.9980
128	**0.0106**	0.0111	0.9999	**0.1811**	0.1877	0.9999	**0.002174**	0.002205	0.9999
256	**0.003083**	0.003373	1.0000	**0.0598**	0.0643	1.0000	**0.000756**	0.000803	1.0000
512	0.000868	**0.000859**	0.2841	0.0205	**0.0196**	**0.0073**	0.000285	**0.000275**	**0.0208**
1024	0.000261	**0.000260**	0.2581	0.007565	**0.007060**	**3.38e-04**	0.000113	**0.000109**	0.0717
2048	0.000101	**0.000100**	0.3878	0.003145	**0.002867**	**1.75e-06**	5.214e-05	**4.961e-05**	**0.0056**
4096	4.037e-05	**3.949e-05**	0.2180	0.001293	**0.001233**	**0.0022**	2.323e-05	**2.196e-05**	**5.68e-04**
8192	1.766e-05	**1.683e-05**	**0.0047**	0.000566	**0.000532**	**0.0156**	1.008e-05	**9.300e-06**	**1.27e-04**

Table 3: Mean Squared Error (MSE) and p-values for Asian option scenarios. The table compares the standard Sobol' sequence against a sequence found via LLM evolutionary search (LLM). P-values are from a one-sided Wilcoxon signed-rank test conducted over the 10000 randomizations with pairing by identical randomization seeds and are false discovery rate (FDR) corrected. P-values below 0.05 are **bolded**.

One natural concern is that Phase II might simply be rediscovering standard coordinate-wise local optimization, with the LLM contributing little beyond calling `scipy.optimize.minimize`. To test this, we ran baselines that apply SLSQP directly to Sobol' point sets, Fibonacci lattice, and best Phase I sets, all with the same objective and similar iteration budgets. These SLSQP baselines substantially improve over their respective seeds, but the LLM-evolved Phase II programs still achieve lower star discrepancy in 22 of the 24 tested values of N with reductions of up to 15% relative to the best SLSQP-only baseline (Appendix B). This suggests the gains are not due to SLSQP alone, but to the LLM discovering nontrivial initializations and restarts that make SLSQP more effective.

5.2 Improved Direction Numbers

Having demonstrated the framework's ability to construct point sets, we next applied it to the discrete optimization problem of discovering improved Sobol' direction numbers. After several hundred evolutionary iterations, our LLM evolutionary search routine discovered a more performant set of parameters, focusing its modifications on the early dimensions, which are known to explain the vast majority of the variance in Asian option pricing. Specifically, the parameters for dimensions 4, 5, and 6 were updated (Appendix C). All other dimensions (up to 32) remained identical to the Joe & Kuo (2008) baseline.

To validate these new direction numbers, they were benchmarked against the standard Joe & Kuo (2008) parameters across a suite of six Asian option pricing scenarios with varying parameters (Appendix C). The true option price was computed by taking the average of 1000 randomly scrambled

Sobol sequences with $N = 2^{21}$ points each. The rQMC MSE was evaluated over 10000 random seeds for $N = 32 \dots 8192$ points. The direction numbers discovered by LLM evolutionary search produced a significantly lower integration MSE for larger sample sizes $N \geq 512$ under one-sided Wilcoxon signed-rank test and false discovery rate correction (Table 3).

To disentangle the role of initialization from that of the LLM-guided search, we repeated the Sobol' experiment under three variants: (i) our main setup, warm-started from the Joe–Kuo parameters with Gemini Flash; (ii) the same pipeline with the smaller Gemini Flash-Lite model; and (iii) the same pipeline but initialized from random direction numbers rather than Joe–Kuo. Across six 32D Asian-style payoffs and sample sizes N, both Flash and Flash-Lite achieve similar MSE reductions over Joe–Kuo for $N \gtrsim 2048$, whereas the randomly initialized variant remains worse than Joe–Kuo for most N (Appendix C). This suggests that the evolutionary search is leveraging a strong domain-specific prior given by Joe–Kuo, and refines it, rather than solving Sobol' design entirely from scratch.

We also compare our evolved direction numbers against Sobol' digital nets constructed by LatNetBuilder (L'ecuyer & Munger, 2016), using its recommended random-CBC search and projection-dependent t-value criterion in $d = 32$. For all six Asian-style payoffs and $N \leq 1024$, our direction numbers yield substantially lower rQMC MSE than the LatNetBuilder nets, often by factors of 2–10, with differences vanishing only for the largest N where all designs converge (Appendix C).

To ensure the evolved parameters were not merely overfitted to the Asian option's specific payoff structure, we tested their generalizability on a diverse suite of high-dimensional exotic options, including Lookback, Barrier, Basket, and Bermudan options (Appendices C, D). The evolved direction numbers demonstrated strong, generalizable performance, achieving significantly lower integration error across this wider range of financial instruments for larger sample sizes ($N \geq 512$) with the sole exception of Barrier options. This suggests that the evolutionary search discovered a Sobol' sequence with fundamentally more robust and broadly applicable projection properties.

6 Discussion

Using an LLM evolutionary framework, we generate 2D and 3D point sets with state-of-the-art star discrepancy and discover Sobol' direction numbers that lower rQMC integration error for high-dimensional exotic financial option pricing.

Recent state-of-the-art methods, such as the mathematical programming approach of Clément et al. (2024) and the GNN-based MPMC framework of Rusch et al. (2024), are designed to construct point sets for a fixed number of points N and dimensions d. In contrast, our approach generates the direction numbers that define a Sobol' sequence, offering several key advantages:

1. **Extensibility to any N:** Unlike fixed approaches where a new, computationally expensive optimization must be performed from scratch for each value of N, a single, compact set of discovered direction numbers can be used to generate a high-quality point set for any desired number of points. This makes the solution immediately applicable to a wide range of practical problems without requiring any re-computation.
2. **Support for Progressive Integration:** An integration can be performed with N points, and if more accuracy is needed, the next N points from the sequence can be added to refine the estimate while reusing all previous calculations. This is a fundamental capability that static point set generation methods inherently lack.
3. **Easy Randomization:** Unlike highly optimized, deterministic point sets, the LLM optimized Sobol' sequence can make use of standard randomization techniques, such as Owen scrambling (Owen, 1995; 1998), to obtain unbiased error estimates of rQMC.
4. **High-Dimensional Applicability:** The Sobol' framework is designed from the ground up for high-dimensional integration. By optimizing the parameters within this framework, our approach is directly applicable to problems such as the 32-dimensional option pricing benchmarks used in our tests, a domain where direct coordinate optimization for a large N would be computationally intractable.

Other more recent systems, such as ShinkaEvolve (Lange et al., 2025), extend the same underlying AlphaEvolve loop with additional engineering features tailored to long-horizon code benchmarks,

such as richer parent sampling, novelty filtering, and advanced orchestration of program executions. In our setting, each candidate's fitness is a cheap, deterministic quantity (star discrepancy or rQMC MSE) computed by running a short Python script, so the core evolutionary mechanism provided by OpenEvolve is sufficient to explore the search space effectively. We deliberately avoided any RL-style training or fine-tuning of the LLM: all improvements come from search in program space driven by an off-the-shelf model and the numerical fitness signal.

Taken together with the broader suite of 32D lookback, basket, and Bermudan payoffs in Appendix D, these results suggest that our evolved direction numbers are genuinely useful beyond the specific Asian option used in the fitness function: they systematically reduce MSE for a range of smooth, path-dependent integrands, but do not help (and may slightly hurt) for highly discontinuous integrands such as barrier options. We view this as an instance of problem-dependent Sobol' design: the same pipeline could in principle be re-run with an objective that mixes several payoff types to trade off specialization and robustness.

Methodologically, our work does not introduce new evolutionary operators; our contribution is empirical and conceptual: we show that a generic LLM-guided program search, applied almost out-of-the-box, is capable of rediscovering and subtly improving long-studied QMC constructions, and we analyze when these gains go beyond what can be achieved with standard local optimization alone. Empirically, our evolved Sobol' parameters work best for smooth, low-to-moderate variation integrands, and do not improve performance for highly discontinuous functions such as the payoff of barrier options.

One practical limitation of our approach is computational cost. For example, the full Sobol' search requires roughly 2000 LLM calls and on the order of $\approx 1.6\times10^{10}$ 32-dimensional QMC samples and payoff evaluations (Appendix A). On our CPU workstation this corresponds to about 96 wall-clock hours per Sobol' experiment. However, this cost is incurred once, offline: the resulting direction numbers can then be reused across many downstream Monte Carlo tasks.

We quantify significance using paired signed-rank tests and report all seeds; however, due to compute limits we performed one evolutionary run per problem, which we treat as a limitation. Future work could explore alternative prompting techniques, optimization of evolutionary programming hyperparameters, and meta-model methods that generate optimal direction numbers for any given dimension d or point sets for any given dimension d or number of points N.

7 Conclusion

We have demonstrated the application of an LLM-driven evolutionary framework to tackle complex discovery problems in the field of low-discrepancy sets. Our method has discovered new 2D and 3D point sets with star discrepancy values lower than any previously published, setting new benchmarks in a field of long-standing mathematical interest. Furthermore, it has produced novel Sobol' direction numbers that improve the accuracy of rQMC integration for a variety of 32-dimensional financial derivatives. This work strengthens the case for using LLMs as core components in an automated scientific discovery process, capable of generating novel and valuable mathematical knowledge.

References

Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, and Victor Ostromoukhov. Sobol' sequences with guaranteed-quality 2d projections. *ACM Transactions on Graphics*, 44, 2025. doi: 10.1145/3730821.

François Clément, Diederick Vermetten, Jacob de Nobel, Alexandre D. Jesus, Luís Paquete, and Carola Doerr. Computing star discrepancies with numerical black-box optimization algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1330–1338. ACM, 2023. doi: 10.1145/3583131.3590456.

François Clément, Carola Doerr, Kathrin Klamroth, and Luís Paquete. Constructing optimal L_∞ star discrepancy sets. arXiv preprint arXiv:2311.17463v2, 2024.

Josef Dick and Friedrich Pillichshammer. *Digital Nets and Sequences*. Cambridge University Press, 2010. doi: 10.1017/CBO9780511761188.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 2022. doi: 10.1038/s41586-022-05172-4.

Gemini Team et al. Gemini: A family of highly capable multimodal models. Google Technical Report, 2025.

Paul Glasserman. *Monte Carlo Methods in Financial Engineering*, volume 53. Springer New York, 2003. doi: 10.1007/978-0-387-21617-1.

Edmund Hlawka. Funktionen von beschränkter variation in der theorie der gleichverteilung. *Annali di Matematica Pura ed Applicata*, 54:325–333, 1961. doi: 10.1007/BF02415361.

Stephen Joe and Frances Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30:2635–2654, 2008. doi: 10.1137/070709359.

J. F. Koksma. The theory of asymptotic distribution modulo one. *Compositio Mathematica*, 16: 1–22, 1964.

John R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4:87–112, 1994. doi: 10.1007/BF00175355.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

Pierre L'ecuyer and David Munger. Algorithm 958: Lattice builder: A general software tool for constructing rank-1 lattice rules. *ACM Trans. Math. Softw.*, 42(2), May 2016. ISSN 0098-3500. doi: 10.1145/2754929. URL `https://doi.org/10.1145/2754929`.

Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics, USA, 1992. ISBN 0898712955.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv preprint arXiv:2406.13131, 2025.

Art B. Owen. Randomly permuted (t,m,s)-nets and (t, s)-sequences. In Harald Niederreiter and Peter J.-S. Shiue (eds.), *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume 106 of *Lecture Notes in Statistics*, pp. 299–317. Springer, 1995. doi: 10.1007/978-1-4612-2554-9_24.

Art B. Owen. Latin supercube sampling for very high-dimensional simulations. *The Annals of Statistics*, 26:619–640, 1998. doi: 10.1214/aos/1028144853.

Spassimir H. Paskov and Joseph F. Traub. Faster valuation of financial derivatives. *The Journal of Portfolio Management*, 22:113–123, 1995. doi: 10.3905/jpm.1995.409541.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

T. Konstantin Rusch, Nathan Kirk, Michael M. Bronstein, Christiane Lemieux, and Daniela Rus. Message-passing monte carlo: Generating low-discrepancy point sets via graph neural networks. *Proceedings of the National Academy of Sciences*, 121, 2024. doi: 10.1073/pnas.2409913121.

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent, 2025. URL `https://github.com/codelion/openevolve`.

I. M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7:86–112, 1967. doi: 10.1016/0041-5553(67)90144-9.