Human Interaction and Emerging Technologies (IHIET 2026), Vol. 236, 2026, 419–429
https://doi.org/10.54941/ahfe1008117

AI Guided Simulated Annealing for Automated Gene Editing Design

Dai Duong Nguyen[1], Ryan Shaw[1], and Kenneth Y. T. Lim[2]

[1]Independent Scholar, Singapore
[2]National Institute of Education, 1 Nanyang Walk, 637616, Singapore

ABSTRACT

Designing effective gene and mRNA sequences is challenging because possible nucleotide combinations grow exponentially with sequence length. Traditional methods like simulated annealing explore large search spaces effectively but depend on scoring functions to evaluate candidates. Hand-crafted scoring rules are computationally slow and inflexible across biological contexts or patient-specific constraints. This project presents an AI-guided simulated annealing framework for automated gene sequence design using two approaches. The first replaces fixed rule-based scoring with an adaptive model evaluating candidates using biological reference data and patient-specific information. By adjusting biological trait importance based on age, disease background, and treatment goals, the scoring model dynamically changes sequence evaluation without modifying the optimization algorithm. The second approach employs Gradient Boosting Regression on CRISPR guide RNA sequences with extracted biological features including GC content, positional nucleotides, and sequence complexity metrics. This model learns from validated literature guides, providing interpretable, deterministic scoring while maintaining adaptability. The framework supports long-running searches with minimal human intervention. Sequence evaluation is decoupled from the optimization engine, allowing scoring models and reference databases to update as new data emerges, enabling pipeline reuse across vaccine design, cancer gene targets, and personalized therapies. By combining a fast native optimization core with adaptive, context-aware evaluation, this work demonstrates flexible large-scale gene sequence optimization. The system shows how AI-driven scoring improves heuristic search practicality and advances sequence design toward personalized, data-driven biomedical applications.

Keywords: Gene sequence optimization, Simulated annealing, AI-guided optimization, Adaptive scoring, Personalized gene design, Computational genomics

INTRODUCTION

Gene editing enables targeted DNA changes and has shifted many therapies toward addressing disease causes rather than symptoms, with CRISPR Cas9 supporting advances in engineered immune cell therapies for cancer, improved vaccine and antiviral strategies, and correction of inherited mutations in clinical settings. Across these applications, success depends on designing nucleic acid sequences that satisfy strict functional constraints while remaining effective in complex biological environments, yet this design process faces a major computational bottleneck because the number

Received June 10, 2026; Revised July 5, 2026; Accepted July 23, 2026; Available online August 25, 2026

of possible nucleotide sequences grows explosively with length. This work therefore pursues two objectives: first, to replace slow hand crafted scoring routines with an AI based evaluator that estimates sequence fitness rapidly and in a context aware manner, with outputs that can be updated as new experimental data become available, and second, to automate and scale sequence optimisation using simulated annealing through long runs and parallel restarts distributed across available compute so that more of the design space can be explored with minimal manual intervention and to support future closed loop workflows where experimental results refine the scorer and guide subsequent searches.

LITERATURE REVIEW

Current Methods for Integrating AI, Simulated Annealing and Automation

Zhang et al. (2023) introduced LinearDesign, an algorithmic framework that applies lattice parsing to optimize mRNA sequences for stability and codon usage in polynomial time. Using this approach, they identified an optimal mRNA encoding the 1,273–amino-acid SARS-CoV-2 spike protein in roughly 11 minutes, despite the astronomically large number of possible sequences for a protein of that length. Complementing such principled algorithmic methods, deep-learning models like DeepCRISPR, CRISTA, and DeepHF have been developed to predict high-performing CRISPR guide RNAs by incorporating genomic context, protein features, and off-target risk into their scoring functions (Dixit et al., 2024). More recently, integrated AI agents such as CRISPR-GPT have pushed this automation further: rather than only scoring candidates, they can plan and execute end-to-end CRISPR workflows by selecting suitable CRISPR systems, designing guides, recommending delivery methods, drafting protocols, and assisting with data analysis.

METHODOLOGY / MATERIALS

OVERVIEW

OpenAI Approach

This approach is organised into three interacting components. The first component is a simulated annealing optimiser that proposes small changes to a candidate DNA or mRNA sequence and accepts or rejects each proposal using a temperature-controlled criterion. The second component is a summariser that converts each raw nucleotide sequence into a compact set of predefined biological descriptors, which standardises what information is presented for scoring and reduces evaluation cost during repeated optimisation. The third component is a grader that consumes the summary together with a structured context and returns a single scalar fitness value to the optimiser. Using the OpenAI API in the grader makes the scoring layer easy to update as new experimental outcomes and clinical priorities emerge, since the same interface can support prompt level adaptation through context inputs as well as model adaptation through supervised fine tuning on curated examples of inputs and desired outputs.

Traditional Machine Learning Approach

This approach focuses on a machine learning-based framework for CRISPR guide RNA design using traditional supervised learning combined with simulated annealing optimization as a tractable proof-of-concept to validate the core methodology before scaling to more complex sequence optimization problems. The approach replaced computationally expensive physics-based scoring functions with a trained gradient boosting model capable of rapid sequence evaluation. The complete pipeline consisted of three major components: biological feature extraction from nucleotide sequences, model training using gradient boosting regression, and sequence optimization via simulated annealing with the trained model as the scoring function.

OPENAI MODEL

Reference Sources, Location, Storage, and Use

The summariser and grader use public bioinformatics references such as NCBI RefSeq, Ensembl, UniProt, Kazusa, REBASE, and ViennaRNA to obtain sequence records, annotations, codon usage data, restriction sites, and RNA structure information. These references support consistent feature extraction and constraint checking during optimisation.

In the system, this information is stored locally in a reference database or structured files so it can be accessed quickly during long simulated annealing runs. This avoids repeated online lookups, reduces processing time, and ensures that all candidate sequences are evaluated using the same biological definitions.

The reference store can also be extended with patient-specific or experimental data, such as expression results, stability measurements, immune response data, and clinical constraints. This allows the AI grader to score sequences based not only on general biological priors, but also on real outcomes and personalised design goals.

Sequence Proposal and Simulated Annealing Search Procedure

The simulated annealing component represents each candidate solution as a nucleotide sequence together with its current score and the best score observed so far. A run begins from an initial sequence and proceeds by iteratively proposing a neighbour sequence using a bounded perturbation operator. For protein coding designs, neighbour proposals are commonly implemented as synonymous codon substitutions so that the amino acid sequence remains unchanged while the nucleotide realisation explores alternative codon choices. Additional proposal operators can include local resampling within a short window, targeted edits intended to remove a prohibited motif, and constraint repair steps that resample until a valid neighbour is obtained. Each proposed neighbour is then evaluated by the scoring pipeline and either accepted as the new state or rejected according to the acceptance criterion, while the best observed candidate is retained for output at the end of the run.

Mathematical Formulation of the Acceptance Rule and Cooling Schedule

Let F(s) denote the fitness assigned to a sequence s by the grader, where larger values indicate better candidates. At iteration k, the algorithm proposes a neighbour s′ and computes the fitness change ΔF defined as F(s′) minus F(s). If ΔF is at least zero, the proposal is accepted. If ΔF is below zero, the proposal is accepted with probability exp of ΔF divided by T_k, where T_k is the temperature at iteration k. Early in the run, higher temperature increases the chance of accepting worse proposals, which supports exploration of rugged objective landscapes. Temperature is then reduced according to a cooling schedule, often geometric cooling written as T_{k+1} equals α times T_k with α between zero and one, which makes acceptance progressively more selective and encourages convergence toward high scoring regions.

Operation of the Grader and its Interface to Model-Based Scoring

The grader serves as the evaluation interface for simulated annealing by converting each candidate into a single numeric fitness score that can be consumed by the acceptance rule. For every proposal, the grader receives the structured summary produced by the summariser together with a context object that encodes the current design setting, then submits a request to the OpenAI platform using the Responses API. The grader enforces a strict output contract that requires one decimal number between 0 and 100 with exactly nine digits after the decimal point, and it validates every model response against this constraint before returning a score to the optimiser. When a response does not conform, the grader issues a corrective prompt and retries until a valid score is obtained. The model used is gpt 5.2 with the default reasoning behaviour.

Role of the Summariser in the Scoring Pipeline

The summariser transforms each candidate nucleotide sequence into a compact, standardised representation that can be scored quickly and consistently during optimisation. Rather than presenting the grader with the full raw sequence, the summariser extracts a fixed set of biological descriptors and returns them in a structured format such as JSON. This reduces prompt size, limits variation in how information is presented across candidates, and improves repeatability across long runs and restarts. The summariser also acts as the main point where biological reference information is applied. It uses the local reference store containing codon mappings, motif definitions, and parameter defaults so that the same definitions of motifs, windows, and thresholds are used throughout the optimisation process. The result is that the optimiser can treat scoring as a stable numerical function, while the biological meaning of each candidate is expressed explicitly in the summary object.

Biologically Relevant Descriptors and Rationale

The summariser extracts a compact set of biological descriptors so that the AI grader can score candidate sequences efficiently without processing the full raw sequence each time. These descriptors include sequence length,

global and local GC content, codon usage patterns, rare codon indicators, dinucleotide and short k-mer composition, and basic motif checks. Together, they provide useful signals about expression potential, RNA stability, manufacturability, and safety.

For coding sequences, codon-related features help estimate whether a design is likely to translate efficiently in the intended host. For RNA designs, CpG, UpA, k-mer, and structure-related features help represent stability and possible immune-sensing risks. Motif screening also flags practical problems such as restriction sites, long homopolymer runs, repeats, internal stop codons, or frame-disrupting patterns. These descriptors are not the main contribution of the project; rather, they serve as a structured biological input layer that allows the AI-based scoring model to make faster and more consistent evaluations during simulated annealing.

Use of the OpenAI API for Adaptation to Patient Conditions and for Model Improvement

The scoring layer is designed so that different biological or clinical contexts can be expressed as explicit inputs to the grader rather than as changes to the optimiser. Patient and application context can be provided as structured fields, such as age group, disease background, target tissue, delivery route constraints, and the design objective, and these fields influence how the model weighs trade-offs when producing the final score. This approach allows the same simulated annealing implementation to be reused across tasks while the evaluation criteria change in a controlled way. Structured Outputs are used to keep the scoring contract stable as contexts vary, since optimisation requires a reliable numeric response at every evaluation. As experimental outcomes accumulate, the same interface supports data driven improvement through supervised fine tuning, in which example inputs and known good outputs are used to train a customised model that produces more consistent task specific scoring behaviour.

TRADITIONAL MACHINE LEARNING MODEL

Feature Engineering

A feature extraction system converted 20-nucleotide guide RNA sequences into 52 numerical features capturing biologically relevant properties (Gasiunas et al., 2018). Global composition features included GC content, purine content, and individual nucleotide counts. Position-specific features encoded nucleotides at critical locations, specifically positions 0 and 19, and the PAM-proximal region spanning positions 17–19, which Doench et al. (2016) identified as significantly influencing guide activity. Dinucleotide features counted consecutive base pairs (GG, CC, AA, TT, CG, GC) affecting RNA structure.

Homopolymer detection implemented binary flags for runs of identical nucleotides, which are TTTT, TTT, AAAA, GGGG, and CCCC, alongside a calculated maximum run length, as poly-T stretches are known to act as transcription terminators (Doench et al., 2016). Sequence complexity metrics

quantified diversity through unique k-mer counts and Shannon entropy, while self-complementarity was assessed by comparing each sequence to its reverse complement, providing a normalised off-target binding risk score. Seed region analysis focused on the critical 12 nucleotides at the 3' end, local GC variation was measured via a 5-nucleotide sliding window, and motif detection flagged beneficial (GG at start or end) and detrimental sequences (TTTT and polyA signals).

Training Data Generation

A stratified dataset of 2,000 sequences was constructed, comprising 15 validated guides drawn from published literature (Hart et al., 2015), targeting loci including EMX1, VEGFA, FANCF, HBB, CCR5, PCSK9, IL2RG, CD19, and KRAS, and 1,985 synthetic sequences distributed across three quality tiers: 30% high-quality (balanced composition, no homopolymers), 40% medium-quality (random nucleotides), and 30% low-quality (poly-T runs, extreme GC content, repetitive patterns).

Ground truth scores were assigned using a simplified Doench scoring function (Doench et al., 2016). Each sequence received a base score of 0.6, with a bonus of up to 0.2 awarded for GC content between 40-60% (optimised at 50%). Positional bonuses were applied for G at position 0 (+0.15), G at position 19 (+0.20), and a GG dinucleotide at the start (+0.10). Penalties were applied for TTTT (–0.3), TTT (–0.1), and any other 4-base homopolymer in A, C, or G (–0.2 each). All scores were clipped to the range [0.0, 1.0].

Model Architecture and Training

A Gradient Boosting Regressor was trained on standardised features, reflecting broader trends in applying machine learning to guide RNA efficacy prediction (Dixit et al., 2024; Qu et al., 2025). All inputs were scaled to zero mean and unit variance using a StandardScaler. The model was configured with 200 estimators, a learning rate of 0.05, maximum tree depth of 5, minimum samples per split of 10, minimum samples per leaf of 4, a subsample ratio of 0.8, and a random state of 42. Performance was assessed via 5-fold cross-validation with Mean Absolute Error as the evaluation metric.

Sequence Optimization via Simulated Annealing

The trained model served as the energy function for a simulated annealing optimisation procedure (Kirkpatrick, Gelatt and Vecchi, 1983). Each candidate sequence was initialised with G fixed at positions 0 and 19 and randomly sampled nucleotides at the remaining positions. At each iteration, a single randomly selected position was mutated to a randomly chosen nucleotide. The Metropolis acceptance criterion was then applied: improvements were always accepted, while worse solutions were accepted with probability $\exp(\Delta/T)$, where Δ is the change in score and T is the current temperature. Temperature cooled exponentially from an initial value of 100.0, multiplying by a factor of 0.95 at each step, with termination after 10,000 iterations or

when the temperature fell below 0.01, allowing broad exploration early and focused exploitation later (Alba and Tomassini, 2002).

Validation Protocol

Model performance was evaluated on 11 validated guides using Mean Absolute Error (average prediction deviation), Pearson correlation (linear agreement between predictions and ground truth), and prediction standard deviation (variance check against mode collapse).

Software and Libraries

Implementation used Python 3.8+ with NumPy 1.21+ (numerical operations), scikit-learn 1.0+ (GradientBoostingRegressor, StandardScaler, cross_val_score), and pickle (model serialization). The system can be ported to C++ when scaling to longer sequences and distributed computing systems. Random seeds were set to 42 for reproducibility.

RESULTS / DISCUSSION

OPENAI MODEL

Lone Model Speed

As α moves closer to 1, the temperature decreases more slowly, so the optimiser spends many more steps in the higher-temperature regime. That raises the time required to reach $T < 1$ from about 15 seconds at $\alpha = 0.900$ to about 1513 seconds at $\alpha = 0.999$. This pattern is expected for geometric cooling, where small changes in α near 1 cause large changes in the number of temperature updates. In practical terms, larger α enables more exploration and usually a better chance of escaping local optima, but it also increases wall time and the number of scoring calls, which directly increases cost and reduces throughput.

The average end to end evaluation time observed by the optimiser is 1.659 seconds. In these runs, the value is computed from the elapsed wall clock time printed by the program, divided by the number of completed optimisation steps. Each step includes writing the candidate sequence to a temporary file, launching the external grader process, waiting for the API response, reading and parsing the returned score, and then performing the acceptance update. As a result, this metric reflects the practical per evaluation latency seen by simulated annealing, rather than only the model inference time in isolation.

Parallel Model Speed

It is observed that this same end to end evaluation time changes under concurrency. As the number of parallel simulated annealing runs increases, the average time per evaluation rises from the single run baseline to substantially larger values at high concurrency. This pattern is consistent with shared bottlenecks, including request queueing, rate limits, network contention, and local process overhead from many simultaneous grader invocations. The results indicate a clear knee in performance, where modest parallelism adds

little overhead, while larger parallelism increases waiting time sharply. In practice, this suggests selecting a concurrency level near the knee for efficient throughput, then scaling further by distributing runs across machines or introducing scheduling, caching, or batching to reduce contention.

Future Work

These results show that throughput is dominated by end-to-end scoring latency rather than by the simulated annealing updates. Future work will therefore focus on reducing evaluation cost while preserving biological usefulness, for example through multistage scoring that filters poor candidates using inexpensive heuristics and cached lookups before invoking model-based grading, and through memoisation that hashes sequence summaries to reuse scores for identical or near identical candidates. In addition, the sharp rise in evaluation time at high parallelism motivates work on concurrency control, including request scheduling, client-side rate limiting, and backoff strategies to avoid queueing collapse. Efficiency is also expected to improve by replacing per call process spawning with a persistent grading service and by distributing independent annealing chains across machines when scaling beyond a single host.

TRADITIONAL MACHINE LEARNING MODEL

Model Training Performance

The gradient boosting model achieved exceptional accuracy on both training and validation data.

Training MAE of 0.0024 indicates predictions within ±0.0024 of true scores. Cross-validation MAE of 0.0090 (3.75× training MAE) suggests modest overfitting but excellent generalization. The near-perfect correlation ($r = 0.9999$) confirms the model captured the scoring function accurately. Prediction standard deviation of 0.2906 demonstrates diverse outputs across the score range, avoiding mode collapse where the model would predict similar scores for all sequences.

Feature Importance Analysis

The gradient boosting algorithm inherently calculates feature importance based on how frequently each feature is used for decision tree splits and how much each feature contributes to reducing prediction error. This analysis revealed which sequence properties the model identified as most predictive of guide RNA quality.

Maximum homopolymer length dominated (47.3%), reflecting the severe impact of nucleotide runs, particularly poly-T stretches, on transcriptional expression (Doench et al., 2016). GC-related features collectively contributed 28.8%, validating balanced composition requirements established in sgRNA design literature (Doench et al., 2016; Gasiunas et al., 2018). Position-specific features (pos19_is_G, pos0_is_G) and poly-T detection confirmed terminal nucleotides and transcription terminator avoidance as critical factors

(Doench et al., 2016). This hierarchy aligns with established guide RNA design principles (Gasiunas et al., 2018), confirming biologically meaningful learning.

Sequence Optimization Results

Simulated annealing converged rapidly from initial sequence GGTAGGTAAGCGGGGTATTG (score 0.8789) to optimised sequence TTACGCTACTCCAGAAGGTG (score 0.9938) in only 30 iterations, requiring just five accepted beneficial mutations.

The optimised sequence achieved perfect GC content (50%), avoided all homopolymer runs (max length 2), and ended with G. The absence of G at position 0 reveals the model prioritised GC balance over terminal position rules, with rapid convergence (16.7% improvement acceptance rate) confirming that continuous ML predictions provide effective search guidance, consistent with related sequence optimisation frameworks (Zhang et al., 2023; Xie et al., 2022).

Validation on Literature Guides

Validation yielded MAE of 0.0030, correlation of 0.9999, and prediction standard deviation of 0.2328. All guides achieved errors below 0.01, with maximum error only 0.0089. The model accurately predicted both high-quality guides (scores 0.9-1.0) and correctly identified low-quality VEGFA guides (score 0.4) containing problematic homopolymer runs, consistent with the known activity profiles of these guides in published screening datasets (Hart et al., 2015; Doench et al., 2016). Validation MAE closely matched training MAE, confirming the model learned the scoring function rather than memorising sequences.

Computational Efficiency

The ML approach achieved approximately 50–200× speedup over traditional methods. Feature extraction and prediction required ~0.5-1.0 ms per sequence (1,000–2,000 sequences/second), optimisation converged in under one second, and the full pipeline completed in ~30 seconds. This efficiency enables previously impractical strategies such as long annealing runs and massively parallel searches, supporting the development of distributed optimisation frameworks for large-scale guide design.

Future Work

Current results rely on synthetic Doench-style labels rather than experimental measurements, so the near-perfect metrics (r = 0.9999, MAE = 0.0030) reflect function approximation; retraining on experimental data is expected to yield realistic correlations of 0.7–0.9. The modular architecture supports expansion to mRNA vaccines and CAR-T constructs via extended feature extraction and context-specific models capturing chromatin accessibility, expression patterns, and mutation databases. Computational scaling will move from

a single-CPU chain to distributed parallel searches via Ray or Dask, and a closed-loop synthesis-measure-retrain workflow with active learning will confirm predictions against genuine editing efficiency. Benchmarking against DeepCRISPR, CRISTA, DeepHF, LinearDesign, and CRISPR-GPT on shared datasets will evaluate predictive accuracy and computational efficiency.

CONCLUSION

This work demonstrates that scaling simulated annealing for sequence design is practical when the scoring function is decoupled as a modular, independent component. Experiments indicate that the primary computational bottleneck is candidate evaluation, governed tightly by the cooling parameter, α, and shared service limits during high concurrency where per-step latency rises from 1.659 to 30.995 seconds at 200 parallel runs. To bypass these constraints, a complementary machine learning pathway using a supervised regressor provides rapid, repeatable scoring. This justifies a hybrid architecture: fast predictors screen massive candidate volumes, while adaptive, model-based grading is reserved to refine top-tier regions. Ultimately, heuristic optimization becomes viable for gene and mRNA design when the evaluation layer is learnable and operationally robust, allowing the system to seamlessly integrate updated biological priors and patient-specific data over long, parallel runs.

ACKNOWLEDGMENT

We are deeply grateful to Dr. Kenneth Lim for his continued mentorship and support throughout the year. In addition to his invaluable research guidance, he also generously provided financial support for the use of OpenAI models and the purchase of required computing credits, which made this work possible.

REFERENCES

Adli, M. (2018) 'The CRISPR tool kit for genome editing and beyond', Nature Communications, 10(1). doi: 10.1038/s41467-018-02842-2.

Alba, E. and Tomassini, M. (2002) 'Parallelism and evolutionary algorithms', IEEE Transactions on Evolutionary Computation, 6(5), pp. 443–462. doi: 10.1109/TEVC.2002.800880.

Dixit, S. et al. (2024) 'Advancing genome editing with artificial intelligence: opportunities, challenges, and future directions', Frontiers in Bioengineering and Biotechnology, 11. doi: 10.3389/fbioe.2023.1335901.

Doench, J. et al. (2016) 'Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9', Nature Biotechnology, 34, pp. 184–191. doi: 10.1038/nbt.3437.

Frangoul, H. et al. (2023) 'CRISPR-Cas9 gene editing for sickle cell disease and β-thalassemia', New England Journal of Medicine, 389(8), pp. 753–764. doi: 10.1056/NEJMoa2309194.

Gasiunas, G. et al. (2018) 'Optimizing CRISPR-Cas9 guide RNA design for genome editing', Genes, 9(12), p. 586. doi: 10.3390/genes9120586.

Hart, T. et al. (2015) 'High-resolution CRISPR screens reveal fitness genes and genotype-specific cancer liabilities', Cell, 163(6), pp. 1515–1526. doi: 10.1016/j.cell.2015.11.015.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) 'Optimization by simulated annealing', Science, 220(4598), pp. 671–680. doi: 10.1126/science.220.4598.671.

Miao, L. et al. (2021) 'mRNA-based CRISPR-Cas9 for gene editing', Journal of Controlled Release, 338, pp. 608–616. doi: 10.1016/j.jconrel.2021.08.026.

Qu, Y. et al. (2025) 'CRISPR-GPT for agentic automation of gene-editing experiments', Nature Biomedical Engineering, pp. 1–14. doi: 10.1038/s41551-025-01463-z.

Sadelain, M. et al. (2013) 'The basic principles of chimeric antigen receptor (CAR) gene therapy', Cancer Discovery, 3(4), pp. 388–398. doi: 10.1158/2159-8290.CD-12-0548.

Xie, N.G. et al. (2022) 'Designing highly multiplex PCR primer sets with simulated annealing design using dimer likelihood estimation (SADDLE)', Nature Communications, 13(1). doi: 10.1038/s41467-022-29500-4.

Zhang, H. et al. (2023) 'Algorithm for optimized mRNA design improves stability and immunogenicity', Nature, pp. 1–3. doi: 10.1038/s41586-023-06127-z.