

Domain-Adaptive Cuckoo Search for Structured Statistical Optimization: Applications to Principal Curves, Single-Cell Trajectory Inference, and Exact Experimental Design

Elvis Han Cui

University of California, Los Angeles

Heather Xihe Yu

yuxihe19@g.ucla.edu

University of California, Los Angeles

Weng Kee Wong

University of California, Los Angeles

Article

Keywords: Domain-adaptive metaheuristics, Cuckoo Search, Lévy flights, Principal curves, Single-cell trajectory inference, Exact experimental design

Posted Date: August 26th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9940945/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Domain-Adaptive Cuckoo Search for Structured Statistical Optimization: Applications to Principal Curves, Single-Cell Trajectory Inference, and Exact Experimental Design

Elvis Han Cui^{1,3}, Heather Xihe Yu^{1,2}, and Weng Kee Wong^{1,*}

¹Department of Biostatistics, University of California, Los Angeles,
650 Charles E Young Dr S, Los Angeles, CA 90095, USA

²Department of Statistical Sciences, University of Toronto,
9 King's College Circle, Toronto, ON M5S 3G3, Canada

³Clinical FSP Department, IQVIA Kuntuo,
9/F, Building 1, Crystal Plaza, 1359 Yaolong Road, Pudong New Area, Shanghai 200124, China

*Corresponding author: wkwong@g.ucla.edu

Abstract

Nature-inspired metaheuristics are widely used in optimization, but their systematic adaptation to structured statistical optimization problems remains limited. We develop three domain-adaptive variants of Cuckoo Search (CS) for statistical and bioinformatics problems with different constraint structures. First, *Cuckoo Search for principal-curve estimation* (*Cuckoo-PC*) uses a dispersion-penalized objective to jointly optimize projection indices and curve parameters in principal curve estimation. Second, a *mixed-domain CS* introduces integer-jump abandonment for constrained maximum likelihood estimation in single-cell gene trajectory models with continuous and discrete parameters. Third, an *exact-design CS* uses integrality projection to construct locally D -optimal exact designs for the maximum-effect (Emax) pharmacokinetic model with autoregressive errors, where small-sample approximate-design rounding can be unreliable. Across these applications, simulation studies and comparative experiments against particle swarm optimization (PSO), differential evolution (DE), and a genetic algorithm (GA) over 30 independent trials show improved objective values or reconstruction metrics in the reported multimodal, non-convex, and mixed-integer settings. The results support the practical value of embedding problem-specific structure directly into the search dynamics, while also highlighting the trade-off between accuracy and the additional computational cost of population-based search. Computer codes are publicly available at <https://github.com/ElvisCuiHan/CuckooPC>.

Keywords: Domain-adaptive metaheuristics; Cuckoo Search; Lévy flights; Principal curves; Single-cell trajectory inference; Exact experimental design

1 Introduction

In recent years, nature-inspired metaheuristic algorithms have gained popularity across disciplines [1–4]. Their rapid rise in the optimization literature is well documented in [5, 6]. Some key reasons are that these algorithms are essentially assumption-free, they are fast and codes are widely available in different platforms. There are increasingly many monographs that discuss their role and importance of metaheuristics in virtually every field in varying degrees, including in artificial intelligence, see for example, [7, 8]. Statistical researchers seem to be late comers, perhaps due to their more-mathematically focused approach, but there are indications that the usefulness of metaheuristics is also gaining recognition in some subfields in statistics. For example, after the pioneering work of [9], who employed particle swarm optimization (PSO) to

find various types of optimal designs for various biomedical models, there has been a steady increase in the use of metaheuristics in the design community, and beyond. For example, they include designing a study to optimally estimate a quantile in a binary model [10], finding super-saturated designs [11] and finding a design that optimally discriminate among nonlinear models [12]. Metaheuristics have also recently been used to tackle non-design problems in statistics, and they include optimization problems to predict disease progression of patients with idiopathic pulmonary fibrosis using computed tomography (CT) scans from baseline only [13], controlling and tracking the coronavirus disease 2019 (COVID-19) pandemic [14], improving modeling techniques in adaptive splines fitting [15], and in tackling estimation issues in statistical problems [16].

Among the most widely used nature-inspired metaheuristic algorithms are particle swarm optimization (PSO), proposed by [17] and differential evolution (DE), proposed by [18]. The former is swarm-based and the latter is evolutionary based. [19] reviewed use of PSO to find various types of optimal designs and [20] gave a tutorial on finding optimal designs using DE before applying DE to find order-of-addition optimal designs. The latter designs treat the order the drug is administered as a design variable because for some serious diseases, it is known that the order of administration of the combined drugs in a drug combination trial can affect the outcome; see [21, 22] for details. Like many other nature-inspired metaheuristic algorithms, both PSO and DE codes are widely available in different platforms; for example, [23] provides R codes for implementing DE and MATLAB has a library of codes for metaheuristic algorithms. A specific application was given in [24], where she showed DE can find locally D -optimal designs for estimating all parameters as accurately as possible in a logistic model with several interacting factors on different design spaces. Since interactions are important in multi-factor studies [25], [26] applied CS to explore new significant genetic interactions.

Cuckoo Search (CS) was proposed by [27] and it has nearly 10,000 citations to date. It has long been recognized for its ability to tackle various types of complex optimization problems. Its broad applications can be found in healthcare research [28], business optimization [29], parameter estimation [30], and high-dimensional optimization problems [31]. In addition to its multi-faceted applications in the real world [32], its important role in deep learning is demonstrated in numerous medical applications; see [33], for example, where CS was used to detect cardiac arrest or hybridized with harmony search to predict cardiac events. In [34], CS was hybridized with Harmony Search along with neural network to predict cardiac events. [35] provides yet another interesting application of CS to solve differential equations numerically by turning the problem into an optimization problem.

The aims of this paper are to present three *domain-adaptive* variants of CS that address fundamentally different structural challenges arising in statistical optimization. Rather than applying a generic CS implementation uniformly, each variant incorporates problem-specific modifications that directly target the structural properties of the underlying optimization landscape. Specifically, the three problems share a common set of structural difficulties: (i) highly non-convex, multi-modal and non-differentiable objective functions; (ii) constraints that are either non-differentiable (integer restrictions), geometrically implicit (projection onto a manifold), or structurally mixed (exact-design integrality with correlated-error information matrices); and (iii) solution quality that is highly sensitive to premature convergence, motivating the heavy-tailed exploration of Lévy flights.

The three domain-adaptive CS variants developed here are:

- (i) **Cuckoo-PC (Cuckoo Search for principal-curve estimation)**: A regularized CS framework that simultaneously optimizes projection indices and smoother parameters for principal curve estimation, incorporating a dispersion penalty that addresses the classical saddle-point pathology identified by [36] (Section 3).
- (ii) **Mixed-Domain CS**: A CS variant with integer-jump abandonment for constrained max-

imum likelihood estimation in the single-cell Generalized Trend Model (scGTM), handling a 7-dimensional mixed continuous-discrete parameter space (Section 4).

- (iii) **Exact-Design CS:** A CS variant with integrality projection for constructing locally D -optimal exact designs for the maximum-effect (Emax) model with autoregressive errors, directly solving the small-sample design problem that approximate-design rounding cannot reliably address (Section 5).

A key observation motivating this work is that many statistical optimization problems share a common structural characteristic: constraints and geometric properties are typically enforced post hoc in standard metaheuristics, rather than being incorporated into the search dynamics themselves. This mismatch often leads to inefficient exploration and premature convergence.

Section 2 provides background on the Cuckoo Search algorithm, including formal specification of the base variant (Algorithm 1), computational complexity analysis, and comparison with other metaheuristics. Section 3 concerns fitting a best-fitting curve to a cloud of massive data points using principal curves and presents a Cuckoo-PC approach with convergence analysis, along with a quantitative comparison to classical and modern baselines across six synthetic scenarios. Section 4 demonstrates the mixed-domain CS for constrained maximum likelihood estimation for 20 genes, and compares its performance against PSO, DE, and a genetic algorithm (GA) over 30 trials. Section 5 discusses optimal exact design construction for the illustrative Emax model with autoregressive errors and provides a robustness analysis across parameter configurations. Section 6 summarizes limitations before Section 7 concludes with broader implications and future directions.

2 Cuckoo Search Algorithm: Core Principles and Implementation

2.1 Conceptual Framework and Biological Inspiration

Popular algorithms have several variants, which are modified versions of the original algorithm, and each version improves the original version in different ways. For example, [37] is a variant of PSO among its many variants. Some of these modifications may be aimed at having accelerated convergence, extricating particles out of a local optimum more effectively, or making the algorithm more robust to the tuning values. For example, [38–41] provide an overview and recent advances of Cuckoo Search, including a review of some of its variants [30, 42]. Another variant is [43], where a re-fly mechanism was proposed for the Cuckoo Search algorithm based on global feedback from the rest of the cuckoos. [44] provides additional theoretical analysis for Cuckoo Search.

Cuckoos have an interesting parasitic breeding behavior. They lay eggs in the nests of other birds, usually right after the hosts lay their own. Since the hatching time of cuckoos is shorter than that of host birds, a newly hatched cuckoo chick can destroy the eggs of the hosts and occupy the whole nest itself. They can also mimic the call of the host chicks so that the host birds would feed them as their own offspring. However, there is a chance that the host bird discovers the cuckoo egg, and thus will either propel the egg out of the nest or abandon the nest and build a new one elsewhere. Eggs that are more similar to the eggs of host birds have a higher chance of being preserved. Inspired by cuckoo’s brood parasitism behavior, [27] developed a new metaheuristic optimization algorithm called Cuckoo Search (CS).

According to [27], there are three main assumptions in the Cuckoo Search algorithm: they are (i) each cuckoo lays only one egg at a time, and randomly dumps it into a nest; (ii) high quality eggs are carried over to the next generation, and (iii) the number of host nests is fixed, and each egg is discovered by the host bird with a probability $p_a \in [0, 1]$.

Each egg represents a solution and is stored in a nest. The quality of an egg is measured

by its fitness value. The above three assumptions can be transformed into the following search rules: (i) at each iteration, a new solution is randomly generated and replaces a randomly chosen existing solution; (ii) solutions with better fitness values are carried over to the next iteration, and (iii) worse solutions are replaced by new solutions, with probability p_a .

A general pseudocode for CS can be found in Yang and Deb’s seminal work [27]. However, for reproducibility and transparency, we provide the exact algorithmic specification used in this paper in Algorithm 1, along with open-source implementations available at <https://github.com/ElvisCuiHan/CuckooPC>. Our codes come with detailed documentation and can be readily adapted to other optimization problems in statistical science, and beyond.

2.2 Algorithmic Contributions

While the base CS algorithm is well-established [27], the three variants developed in this paper each introduce problem-specific structural modifications that are non-trivial adaptations. We summarize the key innovations here to distinguish our contributions from generic CS applications:

C1. Regularized Projection-Index Optimization (Cuckoo-PC, Section 3). Classical principal curve algorithms alternate between a projection step and a smoothing step, which is known to produce saddle-point solutions [36]. Cuckoo-PC replaces this alternating scheme with a joint stochastic search over all projection indices $\{\lambda_i\}_{i=1}^n$ against a dispersion-penalized objective (Equation 9). The key innovation is treating all n projection indices as a single n -dimensional CS search space, with the Lévy flight mechanism enabling coordinated moves away from multiple local saddle configurations. This differs from prior CS curve-fitting applications [45], where CS was used to optimize basis function weights given fixed projections.

C2. Integer-Jump Abandonment for Mixed-Domain Optimization (Mixed-Domain CS, Section 4). The scGTM likelihood contains continuous kinetic parameters, namely μ_{mag} , k_1 , k_2 , t_0 , α , and β , together with a positive-integer dispersion parameter $\phi \in \mathbb{Z}_+$. Standard CS Lévy flights operate on continuous spaces and produce non-integer updates for ϕ . We introduce a modified abandonment operator: when a nest is selected for replacement with probability p_a , the discrete parameter is updated via $\phi \leftarrow \phi + \Delta$, where $\Delta \sim \text{Uniform}\{-2, -1, 0, 1, 2\}$, while continuous parameters follow the standard Lévy flight. This modification preserves the integer constraint throughout optimization, reduces rounding artifacts from post-hoc discretization, and maintains the heavy-tailed exploration property for the continuous subspace.

C3. Integrality Projection for Exact Experimental Design (Exact-Design CS, Section 5). Finding locally optimal *exact* designs under correlated errors requires simultaneously optimizing continuous support points and integer observation counts subject to a hard total-sample constraint $\sum n_i = N$. After each Lévy flight update, support points are projected onto the feasible design space $[0, x_0]$ and the observation counts are re-allocated using a constrained rounding procedure that exactly preserves N [46]. This avoids the well-documented failure of naive rounding for small N , where rounded allocations may not sum to N and may produce zero allocations at critical design points.

2.3 Formal Algorithm Specification for Statistical Optimization

To address the specific requirements of optimal experimental design, we now provide a formal specification of the CS variant used throughout this paper. This formulation adapts the general CS framework to the constrained optimization landscape of design problems.

2.3.1 Problem Formulation

Given a design space $\mathcal{X} = [x_l, x_u]^m$ and a design criterion $\Phi(\xi)$ (e.g., determinant of the information matrix for D -optimality), we seek to find an exact design $\xi^* = (x_1, x_2, \dots, x_m)$ that optimizes:

$$\xi^* = \arg \max_{\xi \in \mathcal{X}} \Phi(\xi) \quad (1)$$

subject to design-specific constraints (e.g., ordering constraints, sample size allocation).

2.3.2 Theoretical Foundation of Lévy Flights

The Lévy flight mechanism is central to CS’s exploration capability. A random variable L follows a Lévy distribution if its probability density function exhibits the power-law tail behavior:

$$p(L) \sim |L|^{-(1+\beta)}, \quad \text{as } L \rightarrow \infty, \quad 0 < \beta \leq 2 \quad (2)$$

For practical implementation, we employ Mantegna’s algorithm [47], which generates Lévy-stable random variates by exploiting the ratio of two normal variables. Specifically, for $\beta = 1.5$ (corresponding to $\lambda = 2.5$ in the asymptotic tail exponent $t^{-\lambda}$), the step size s is computed as shown in Equation (4) of Algorithm 1.

This choice of $\beta = 1.5$ balances exploration breadth with convergence stability: smaller values would produce excessively erratic jumps, while larger values would reduce the heavy-tail property essential for escaping local optima [48]. The parameter σ_u ensures the correct scaling of the Lévy distribution according to the theoretical framework developed in [49].

2.3.3 Key Features and Advantages

Comparison with Standard Metaheuristics:

Table 1 summarizes key characteristics distinguishing CS from other popular metaheuristic algorithms.

CS offers a favorable balance between exploration (via Lévy flights) and exploitation (via local random walks), making it particularly effective for complex optimization landscapes with multiple local optima. Its relatively simple parameter structure (only p_a and step size control) enhances usability compared to genetic algorithms, while its Lévy flight mechanism provides more aggressive exploration than standard PSO or DE in early iterations.

No-Free-Lunch considerations: The No-Free-Lunch (NFL) theorem [51] establishes that no algorithm is universally superior across all optimization problems. The performance advantage of the CS variants reported in this paper should therefore be interpreted in terms of the structural properties of our specific problem classes: (i) high multimodality arising from non-convex likelihood functions and self-intersecting curve geometries; (ii) mixed continuous-integer parameter spaces where gradient-based methods fail; and (iii) small design dimensions where population-based search is computationally viable. For these structural classes, the heavy-tailed Lévy flight distribution provides a natural match to the escape-from-local-optima requirement, and the simple abandonment mechanism provides inexpensive diversification. Problems with smooth, unimodal, or high-dimensional (> 100 continuous parameters) landscapes may be better served by gradient-based or quasi-Newton methods.

2.3.4 Implementation and Reproducibility

All CS implementations used in this paper are publicly available in our GitHub repository <https://github.com/ElvisCuiHan/CuckooPC>; The repository contains:

- `cuckoo.py`: Core CS implementation with the exact variant described in Algorithm 1

Algorithm 1: Cuckoo Search for Optimal Experimental Design

Input: Design space $\mathcal{X}$, criterion Φ , population size N_p , discovery rate p_a , step size γ ,
max iterations T_{max}

Output: Optimal design ξ^*

Initialization:

Generate initial population of N_p designs $\{\xi_i^{(0)}\}_{i=1}^{N_p}$ by:

$$\xi_i^{(0)} = \mathbf{x}_l + \boldsymbol{\alpha} \odot (\mathbf{x}_u - \mathbf{x}_l), \quad \boldsymbol{\alpha} \sim U[0, 1]^m \quad (3)$$

Evaluate $\Phi(\xi_i^{(0)})$ for each design and identify global best $\xi^* \leftarrow \arg \max_i \Phi(\xi_i^{(0)})$;

for $t = 1$ **to** T_{max} **do**

for $i = 1$ **to** N_p **do**

 // Lévy Flight Step

 Generate step size vector $\mathbf{s} = (s_1, \dots, s_m)$ via Mantegna's algorithm:

$$s_j = \frac{u_j}{|v_j|^{1/\beta}}, \quad u_j \sim \mathcal{N}(0, \sigma_u^2), \quad v_j \sim \mathcal{N}(0, 1) \quad (4)$$

 where $\beta = 1.5$ and

$$\sigma_u = \left[\frac{\Gamma(1 + \beta) \sin(\pi\beta/2)}{\Gamma((1 + \beta)/2) \beta 2^{(\beta-1)/2}} \right]^{1/\beta} \quad (5)$$

 Generate candidate design:

$$\tilde{\xi}_i^{(t)} = \xi_i^{(t-1)} + \gamma \boldsymbol{\alpha} \odot \mathbf{s} \odot (\xi_i^{(t-1)} - \xi^*) \quad (6)$$

 where $\boldsymbol{\alpha} \sim \mathcal{N}(0, \mathbf{I}_m)$ and $\gamma = 0.5$;

 Project $\tilde{\xi}_i^{(t)}$ onto $\mathcal{X}$ if needed;

if $\Phi(\tilde{\xi}_i^{(t)}) > \Phi(\xi_i^{(t-1)})$ **then**

$\xi_i^{(t)} \leftarrow \tilde{\xi}_i^{(t)}$;

 // Local Random Walk and Abandonment

 Randomly select fraction p_a of designs for replacement;

for each selected design $\xi_i^{(t)}$ **do**

 Randomly select two distinct designs $\xi_j^{(t)}, \xi_k^{(t)}$ where $j, k \neq i$;

 Generate replacement:

$$\xi_i^{(t)} \leftarrow \xi_i^{(t)} + \epsilon_1 \cdot \mathbf{1}(\epsilon_2 < p_a) \odot (\xi_j^{(t)} - \xi_k^{(t)}) \quad (7)$$

 where $\epsilon_1 \sim U[0, 1]$ and $\epsilon_2 \sim U[0, 1]^m$;

 Evaluate $\Phi(\xi_i^{(t)})$;

 // Update Global Best

$\xi^* \leftarrow \arg \max_i \Phi(\xi_i^{(t)})$;

 // Convergence Check

if $|\Phi(\xi^*) - \Phi(\xi^{*(t-1)})| < \varepsilon$ **then**

break;

return ξ^*

Table 1: Comparison of CS with other metaheuristic algorithms on standard benchmark characteristics: PSO, DE, GA, and the Grey Wolf Optimizer (GWO) [50]. Convergence speed ratings are based on reported results on Congress on Evolutionary Computation (CEC) benchmark suites [48].

Characteristic	CS	PSO	DE	GA	GWO
Number of control parameters	2	3–4	3	4–5	2
Update strategy	Lévy flights	Velocity-based	Mutation/crossover	Selection	Hierarchy-based
Global search mechanism	Heavy-tailed jumps	Swarm best	Difference vectors	Mutation	Alpha/beta/delta roles
Local search mechanism	Random walks	Inertia	Greedy selection	Elitism	Omega update
Mixed-integer support	Yes (C2)	No (rounding)	No (rounding)	Yes	No
Projection-space search	Yes (C1)	No	No	No	No
Typical convergence speed	Fast	Fast	Moderate	Slow	Fast
Saddle-point escape	Strong	Moderate	Moderate	Weak	Moderate

C1 and **C2** refer to the algorithmic contributions defined in Section 2.2.

- `design_optimization.py`: Wrapper functions for optimal design problems
- `scgtm_optimization.py`: CS for bioinformatics parameter estimation
- `principal_curves.py`: CS for principal curve estimation
- `examples/`: Reproducible examples for all applications in this paper

220 The code includes detailed documentation of all hyperparameters and their default values used throughout our experiments.

2.3.5 Experimental Protocol and Statistical Analysis

225 All stochastic comparisons use independent random initializations and identical objective function implementations within each application. Population-based baselines are compared under the same population sizes where possible and the same evaluation budgets within each experiment. Reported values are means and standard deviations over 30 independent runs unless otherwise specified. For pairwise comparisons between CS and a baseline, we use paired Wilcoxon signed-rank tests on run-level or gene-level paired summaries, as indicated in the relevant table caption, and apply Bonferroni correction for multiple baseline comparisons. The Friedman test 230 is used only as an omnibus comparison across four optimizers in the scGTM experiment. These tests quantify empirical differences under the stated simulation and data-analysis settings; they do not imply universal dominance of one optimizer across all objective functions.

3 Application I: Principal Curve Estimation

235 Cuckoo Search (CS) has been shown to be efficient for fitting high-dimensional data with a curve. For example, [45] employed a CS to optimize a weighted Bayesian energy functional for data fitting by using global-support approximating curves. Such curves are linear combinations of basis functions whose support is the whole domain of the problem. Principal curves are smooth curves that minimize the averaged squared orthogonal distance to each point in a data set. Fitting a principal curve is a maximum-likelihood technique for nonlinear regression when 240 Gaussian noise is assumed for both the outcome and covariate variables.

3.1 Motivation and Basic Concepts

The study of intrinsic data geometry in statistics can be traced back to the pioneering work of Professors Mahalanobis [52] and Rao [53], whose ideas laid the foundation for geometric approaches to statistical analysis. Building on these insights, the development of methods for uncovering nonlinear, low-dimensional manifolds has become an important theme in modern statistics [54].

In this context, *principal curves* are motivated by the need to uncover intrinsic low-dimensional structures in high-dimensional or noisy data. Proposed by [55], they generalize linear principal components to nonlinear manifolds and are smooth curves constructed to pass through the "middle" of a dataset, providing a nonlinear generalization of principal components. A classic application is in **handwritten digit recognition**, where principal curves are used to model the sequence of pen strokes for each digit [56]. Another example appears in **biomedical imaging**, where they help trace anatomical features such as the central path of the colon or neural fiber tracts [57]. More recently, the Monocle framework applies principal curves to reconstruct "pseudotemporal" trajectories in single-cell RNA sequencing (RNA-seq) data, effectively fitting smooth curves through high-dimensional gene expression space to model progression along cell-lineage paths [58]. This enables ordering cells by developmental stage—such as differentiation, proliferation, or oncogenic transformation—revealing underlying regulators and dynamic gene-expression kinetics.

Beyond these traditional applications, principal curves have found renewed relevance in modern machine learning and optimization contexts. In evolutionary multiobjective optimization, principal curves are directly employed to approximate Pareto-optimal set manifolds in continuous multiobjective optimization problems, enabling more effective exploration of complex solution spaces where the Pareto set forms piecewise continuous manifolds [59]. In computer vision applications, principal curves have been successfully applied to markerless motion capture systems for human motion analysis, where they extract skeleton curve features from volume sequences to enable pose estimation and gesture recognition without requiring physical markers [60]. Hand gesture recognition systems have also benefited from principal curve representations, where they provide effective trajectory modeling for state-based gesture analysis frameworks [61].

Recent theoretical advances have significantly enhanced the principal curve framework. The Sequential Principal Curves Analysis (SPCA) method introduces probability density function (PDF)-dependent metrics and sequential application strategies, addressing fundamental limitations of traditional principal curve algorithms with applications in biological signal processing, medical image analysis, and physiological data modeling [62]. For trajectory inference in biological systems, the Slingshot method extends principal curves to handle multiple branching trajectories simultaneously, enabling reconstruction of complex cellular differentiation pathways and developmental lineage analysis in single-cell RNA-seq data [63].

The integration of principal curves with probabilistic frameworks has also been explored, particularly in spatial pattern detection and clustering applications, where hierarchical principal curve clustering combined with mixture models provides robust solutions for noisy point patterns. Specific applications include the detection of curvilinear minefields from reconnaissance images (where some points represent false detections) and the detection of seismic faults from earthquake catalogs, demonstrating the method's effectiveness in handling both background noise and feature-specific noise in real-world geospatial analysis [64]. These applications highlight the flexibility of principal curves in learning smooth, interpretable structures from complex data.

The basic concepts of the original formulation of principal curves were given in [55] and defined using a self-consistency property: each point on the curve equals the conditional expectation of all data points that project onto it. However, theoretical analysis by [36] revealed that principal curves are saddle points rather than minima of the distance function, creating fun-

damental challenges for optimization and model selection. This insight has motivated various algorithmic and theoretical developments in the field.

Many extensions of Hastie’s original principal curve methodology have been proposed in the literature, including local principal curves [65], orthogonal projection-based techniques [42], and applications in directional data [66]. Recent advances include regularization-free approaches that reformulate the optimization problem to avoid saddle point issues [67], length-constrained formulations for controlling model complexity [68], and extensions to Riemannian manifolds for spherical and other non-Euclidean data [69].

In this section, we propose a novel algorithm for learning such curves, which we term the *metric-based principal curve*. This method minimizes a metric distance between the observed data and its projection onto a smooth curve, using a combination of smoothing and regression techniques. Our approach differs from traditional projection-expectation algorithms by employing metaheuristic optimization (specifically the Cuckoo Search algorithm) to simultaneously optimize both the curve parameters and projection indices, potentially avoiding the convergence issues that plague classical iterative methods. The codes for this section are all available at <https://github.com/ElvisCuiHan/CuckooPC>.

To begin, we provide a more rigorous definition of principal curves, setting the stage for our methodological development.

Definition 3.1 (Principal curve assumption). *Let $\mathbf{Y} = (Y_1, Y_2, \dots, Y_p)^T \in \mathbb{R}^p$ be a p -dimensional random vector and $\lambda \in \mathbb{R}$ be a scalar known as the **projection index**. Extending the idea in [70], the **principal curve assumption** for $(\mathbf{Y}, \lambda)$ is described as follows:*

$$\mathcal{P}(\lambda) = \mathbb{E}(\mathbf{Y}|\lambda) = \int_{\mathbb{R}^p} y \mathbb{P}_{\mathbf{Y}|\lambda}(dy) \quad (8)$$

where $\mathbb{P}$ represents a general probability measure and the expectation $\mathcal{P}(\lambda)$ is defined as the **principal curve** of $\mathbf{Y}$. In other words, there is a (one-dimensional) parametric curve that describes the trajectory of $\mathbf{Y}$ in spite of its high dimensionality.

To better understand the concept, let us consider the special case

$$\mathcal{P}(\lambda) = (\sin(2\pi\lambda), e^{-\lambda}, \cos(2\pi\lambda))^T$$

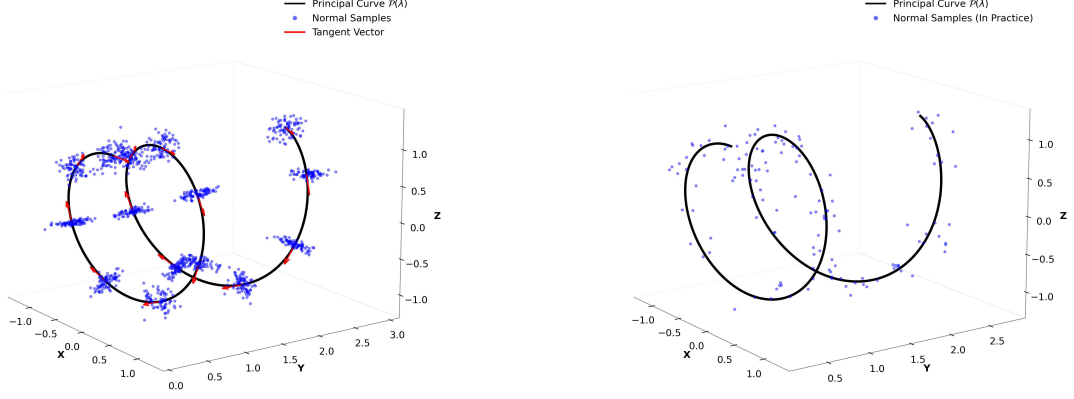
and $\lambda \in [-1, 1]$ which corresponds to a 3-D spiral curve (Figure 1). The distribution $\mathbb{P}_{\mathbf{Y}|\lambda}$ can be constructed as follows (More elaborate constructions of distributions on Riemannian manifolds is given in [71].):

- The tangent space of $\mathcal{P}(\lambda)$ is $\dot{\mathcal{P}}(\lambda) = (2\pi \cos(2\pi\lambda), -e^{-\lambda}, -2\pi \sin(2\pi\lambda))^T$;
- The normal space of $\dot{\mathcal{P}}(\lambda)$ is

$$\mathcal{S}(\lambda) = \{y \in \mathbb{R}^3 : y^T \dot{\mathcal{P}}(\lambda) = 0\};$$

- A parametric distribution such as $(p-1)$ -dimensional Gaussian is defined on the space $\mathcal{S}(\lambda)$ with mean $\mathcal{P}(\lambda)$ and variance $\sigma^2 \mathbf{I}_{(p-1)}$. Computationally, this can be done via orthogonal projection or Householder transformation.

Figure 1 provides an illustration of the above idea with sampled points along the principal curve. As statisticians, we deal with the inverse problem, i.e., estimating the black curve based on the data (blue points). While Figure 1a serves only as an illustration, in practice it is more common to observe fewer samples per individual λ , but a larger number of distinct λ -indexed points overall [59]; the idea is depicted in Figure 1b. There are several projection-expectation (PE) type algorithms to learn the principal curve $\mathcal{P}(\lambda)$ [56, 72]. However, it is well-known that PE algorithms do not guarantee convergence and different types of definition and algorithm lead to different estimated principal curve [73]. Hence, it is of great interest to apply other algorithms to estimate principal curves, and this is the subject of the next subsection.



(a) Illustrative Example

(b) Practical Scenario

Figure 1: 3D Normal Distributions and Tangents on Its Principal Curve

3.2 The Cuckoo-PC Algorithm

Algorithm specification: Our principal curve estimation employs the *standard Cuckoo Search* algorithm with Lévy flight mechanism as specified in Algorithm 1 (Section 2.3), without modifications beyond problem-specific objective function adaptation. This implementation uses the default parameters: discovery rate $p_a = 0.25$, step size control $\gamma = 0.5$, population size $N_p = 20$ nests, and Lévy exponent $\beta = 1.5$. The choice of standard CS ensures reproducibility and allows direct comparison with classical projection-expectation methods. In this subsection, we present the procedure for applying the Cuckoo Search algorithm to estimate principal curves, which we refer to as the Cuckoo-PC algorithm. The idea of applying the CS algorithm starts with a user-specified metric $d(\cdot, \cdot)$ on $\mathbb{R} \times \mathbb{R}$ and a regularization parameter ρ . Suppose we observe the data $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_n$ where each $\mathbf{Y}_i \in \mathbb{R}^p$. The associated projection index $\lambda_1, \dots, \lambda_n$ is chosen such that the following quantity is minimized.

$$\{\lambda_i\}_{i=1}^n = \arg \inf_{\lambda} \frac{1}{n} \sum_{i=1}^n d(\mathbf{Y}_i, \hat{\mathbf{Y}}(\lambda_i)) + \rho \phi(\{\lambda_i\}_{i=1}^n) \quad (9)$$

where $\hat{\mathbf{Y}}(\lambda_i) = (\hat{f}_1(\lambda_i), \hat{f}_2(\lambda_i), \dots, \hat{f}_p(\lambda_i))^T$ is the fitted value of $\mathbf{Y}_i$ using *under-smooth regression models* $\{f_j\}_{j=1}^p$ and ϕ is a dispersion function characterizing the dispersion of λ 's. We provide a list of the smoother f_j , the metric $d(\cdot, \cdot)$, and the dispersion ϕ in table 2. In short, a metric-based principal curve minimizes the mean of distances of all points (feature vectors) projected onto the curve plus a regularization term that penalizes the dispersion of the one-dimensional parameter. Finally, we apply CS or any other metaheuristics to optimize the projection index 9. We summarize the whole procedure in Algorithm 2.

3.3 Convergence Analysis of Cuckoo-PC

A critical theoretical question for Cuckoo-PC is whether minimizing the regularized objective $Q(\{\lambda_i\})$ in Equation (9) avoids the saddle-point pathology identified by [36] for the classical projection-expectation formulation.

Theorem 3.1 (Saddle-Point Avoidance via Regularization). *Let $Q(\{\lambda_i\}) = \frac{1}{n} \sum_{i=1}^n d(\mathbf{Y}_i, \hat{\mathbf{Y}}(\lambda_i)) + \rho \phi(\{\lambda_i\}_{i=1}^n)$ where ϕ is a strictly convex dispersion function (e.g., $\phi = \sum_{i=1}^{n-1} (\lambda_{(i+1)} - \lambda_{(i)})^2$). For any fixed smoother family $\{f_j\}$, the stationary points of Q with respect to $\{\lambda_i\}$ are not saddle points of the unregularized distance functional when $\rho > 0$.*

Table 2: Some choices of f_j , $d(\cdot, \cdot)$ and $\phi(\cdot)$.

f_j	$d(x, y)$	ϕ
Smoothing spline	L_d -distance $\ x - y\ _d$	$\sum_{i=1}^{n-1} \lambda_{(i+1)} - \lambda_{(i)} $
Locally weighted scatterplot smoothing (LOWESS)	Mahalanobis distance	$\sum_{i=1}^{n-1} (\lambda_{(i+1)} - \lambda_{(i)})^2$
Kernel ridge regression	Chebyshev distance	$\max_i \lambda_{(i+1)} - \lambda_{(i)} $
Gaussian process regression	Hellinger distance	Coefficient of variation
Support vector regression	Canberra distance	
Nadaraya-Watson estimator		

Here $\lambda_{(i)}$ denotes the i^{th} order statistic from $\lambda_1, \dots, \lambda_n$.

Algorithm 2: Cuckoo-PC Algorithm for Principal Curve Estimation

Input: Data points $\{\mathbf{Y}_i\}_{i=1}^n$, each $\mathbf{Y}_i \in \mathbb{R}^p$; metric $d(\cdot, \cdot)$; regularizer ρ ; smoother family $\{f_j\}_{j=1}^p$; dispersion function ϕ

Output: Estimated projection indices $\{\lambda_i^*\}_{i=1}^n$ and principal curve $\hat{\mathbf{Y}}(\lambda)$

Initialize projection indices $\{\lambda_i\}_{i=1}^n$ (e.g., uniformly in $[0, 1]$);

repeat

for $j \leftarrow 1$ **to** p **do**

 Fit $\hat{f}_j$ to $\{(\lambda_i, Y_{ij})\}_{i=1}^n$ using the chosen smoother;

 Define fitted curve $\hat{\mathbf{Y}}(\lambda) = (\hat{f}_1(\lambda), \dots, \hat{f}_p(\lambda))^T$;

 Update projection indices $\{\lambda_i\}_{i=1}^n$ by minimizing:

$$Q(\{\lambda_i\}) = \frac{1}{n} \sum_{i=1}^n d(\mathbf{Y}_i, \hat{\mathbf{Y}}(\lambda_i)) + \rho \phi(\{\lambda_i\}_{i=1}^n)$$

 using CS or another metaheuristic;

until convergence or maximum iterations reached;

return $\{\lambda_i^*\}_{i=1}^n$, principal curve $\hat{\mathbf{Y}}(\lambda)$

Proof sketch. The unregularized distance functional $D(\{\lambda_i\}) = \frac{1}{n} \sum_{i=1}^n d(\mathbf{Y}_i, \hat{\mathbf{Y}}(\lambda_i))$ is shown by [36] to have stationary points that are saddle points: for any stationary $\{\lambda_i^*\}$, there exist perturbation directions along which D decreases. The regularization term $\rho \phi(\{\lambda_i\})$ adds strictly positive curvature in directions that change the dispersion of the projection indices. Specifically, for the quadratic dispersion $\phi = \sum (\lambda_{(i+1)} - \lambda_{(i)})^2$, the Hessian of $\rho \phi$ contributes a positive semidefinite component on ordered-index perturbations. Thus, the regularized objective changes the local geometry of the classical projection-expectation formulation and can remove saddle-like directions when the regularization curvature dominates the negative curvature of D . This proof sketch is intended as a local sufficient-condition argument; global convergence of the population-based optimizer is assessed empirically below. $\square$

Empirical convergence behavior: Across the 30 independent trials summarized in Table 4, Cuckoo-PC typically reached a stable objective value within 50–80 iterations. Because the individual iteration histories are not essential for interpreting the comparative accuracy results, the convergence evidence is reported through the variability across independent final runs and the parameter sensitivity analysis in Figure 4.

3.4 Simulation Studies

3.4.1 Simulation Setup and Scenarios

We conducted simulation studies based on several synthetic datasets to systematically evaluate the performance of our method. Six simulation scenarios were considered, as summarized in Table 3. The first three scenarios (Spiral I, Spiral II, and Heart) are defined in $\mathbb{R}^3$, while the latter three (Butterfly, Petal, and Elvis) reside in $\mathbb{R}^2$.

These scenarios are designed to challenge the algorithm under varying levels of geometric complexity, ranging from simple parametric curves to highly nonlinear, self-intersecting, and irregular trajectories. For each case, synthetic datasets were generated following the construction described in Definition 3.1, where observations are sampled from $(p - 1)$ -dimensional Gaussian distributions on the normal spaces $\mathcal{S}(\lambda)$ of the true principal curves $\mathcal{P}(\lambda)$. Different noise levels were incorporated to assess robustness under increasing uncertainty.

Table 3: Simulation settings of different types of principal curves in $\mathbb{R}^2$ and $\mathbb{R}^3$.

Scenario	Name	Range of λ	$\mathcal{P}(\lambda)$	Noise ¹
1	Spiral I	$[-1, 1]$	$(\sin(2\pi\lambda), e^{-\lambda}, \cos(2\pi\lambda))$	0.1
2	Spiral II ²	$[0, 2]$	$(\lambda, 2\lambda \cos(6\lambda), 2\lambda \sin(6\lambda))$	0.1
3	Heart	$[0, 2\pi]$	$(16 \sin^3 \lambda, 13 \cos \lambda - 5 \cos(2\lambda) - 2 \cos(3\lambda) - \cos(4\lambda), \sin \lambda)$	0.1
4	Butterfly	$[0, 1]$	$(\sin(2\pi\lambda)(e^{\cos(2\pi\lambda)} - 2 \cos(8\pi\lambda) - \sin^5(\frac{\pi\lambda}{6})), \cos(2\pi\lambda)(e^{\cos(2\pi\lambda)} - 2 \cos(8\pi\lambda) - \sin^5(\frac{\pi\lambda}{6})))$	0.05
5	Petal	$[0, 1]$	$(\cos(10\pi\lambda) \cos(2\pi\lambda), \cos(10\pi\lambda) \sin(2\pi\lambda))$	0.05
6	Elvis	$[0, 1]$	Cubic B-spline function ³	1

¹ σ^2 of Gaussian noise on the normal space $\mathcal{S}(\lambda)$; see Section 3.1.

² Also known as Archimedean Spiral.

³ Full parametric form: $(\sum_{i=1}^{50} c_{x,i} B_{i,3}(\lambda), \sum_{i=1}^{50} c_{y,i} B_{i,3}(\lambda))$, with fixed coefficients $c_{x,i}, c_{y,i}$ and basis splines $B_{i,3}(\lambda)$; see [74].

3.4.2 Results and Comparisons

Figure 2 displays the reconstructed principal curves by the Cuckoo-PC and Enhanced Hastie-Stuetzle algorithms under the six simulated scenarios from Subsection 3.4.1. The first row presents representative synthetic datasets with noisy samples (blue dots) and the true underlying principal curves $\mathcal{P}(\lambda)$ (green dashed lines). For optimization, we set the hyperparameter $\rho = 1$ and use 20 nests in the CS algorithm.

The second row shows the reconstructed curves using the Enhanced Hastie-Stuetzle algorithm, while the third row presents the results from the proposed Cuckoo-PC algorithm. In the representative runs shown, Cuckoo-PC more closely follows the true structures, capturing both smooth and highly nonlinear features in the presence of moderate noise. In three-dimensional cases, Cuckoo-PC reconstructs the main spatial trajectories, while in two-dimensional settings it better preserves complex features such as loops and self-intersections. In the most challenging “Elvis” scenario, where the true curve is a hand-drawn irregular path, Cuckoo-PC yields a smoother reconstruction with smaller empirical errors than the baseline shown.

To quantitatively compare Cuckoo-PC with the Enhanced Hastie-Stuetzle algorithm, we employ multiple performance metrics. These include L_2 Distance, Distance to True Curve, Data Fitting Quality, Hausdorff Distance, Length Preservation ratio, and Computational Efficiency.

405 The L_2 Distance is defined as

$$L_2 \text{ Distance} = \frac{1}{n} \sum_{i=1}^n \|\hat{y}_i - y_i\|_2^2,$$

with the population counterpart

$$\text{Population } L_2 \text{ Distance} = \int \left\| \hat{\mathcal{P}}(\hat{\lambda}) - \mathcal{P}(\lambda) \right\|_2^2 d\lambda,$$

where $\hat{\mathcal{P}}(\hat{\lambda})$ denotes the estimated principal curve. Hausdorff Distance is given by

$$410 \quad \text{Hausdorff Distance} = \max \left\{ \sup_{\lambda_2} \inf_{\hat{\lambda}_1} \|\hat{\mathcal{P}}(\hat{\lambda}_1) - \mathcal{P}(\lambda_2)\|_2, \sup_{\hat{\lambda}_1} \inf_{\lambda_2} \|\hat{\mathcal{P}}(\hat{\lambda}_1) - \mathcal{P}(\lambda_2)\|_2 \right\},$$

capturing maximum bilateral deviations [75].

Table 4 summarizes the quantitative results across 30 independent trials. To ensure a comprehensive evaluation, we compare Cuckoo-PC against four baselines: (i) Enhanced Hastie-Stuetzle [56], (ii) PSO-PC (PSO with the same regularized objective), (iii) DE-PC (Differential Evolution with the same objective), and (iv) GA-PC (Genetic Algorithm with the same objective). All population-based methods use identical population sizes ($N_p = 20$), evaluation budgets ($T_{\max} = 200$ iterations), and objective function implementations for fair comparison. Statistical significance of Cuckoo-PC’s advantage over each baseline is assessed via paired Wilcoxon signed-rank tests; p -values are Bonferroni-corrected for the four pairwise comparisons.

420 Across the reported scenarios and metrics, Cuckoo-PC achieves the lowest mean errors in Table 4. After Bonferroni correction, the paired comparisons indicate statistically significant advantages for most accuracy metrics, with the largest gains in the Heart and Elvis scenarios, which involve the highest geometric complexity. In simpler cases (Spiral I and II), the population-based methods are closer in performance, suggesting that Cuckoo-PC’s Lévy flight exploration is most useful when the projection-index optimization landscape is highly multi-modal, as can occur when the true curve has high curvature or self-intersections.

Figures 2 and 3 together summarize the visual and quantitative comparisons: reconstructed curves align more closely with the true underlying structures in the displayed examples, and the reported metrics show improved accuracy in the more challenging nonlinear configurations.

430 **Regarding computational efficiency**, Cuckoo-PC incurs a clear runtime cost. As shown in Table 4, Cuckoo-PC required approximately 6–10 seconds per dataset (depending on geometric complexity), while the Enhanced Hastie-Stuetzle method averaged 0.1 seconds. The population-based search is therefore most appropriate when reconstruction accuracy is prioritized over speed, such as in complex curve reconstruction or trajectory-inference settings where repeated convergence failures are costly.

The performance advantage becomes more pronounced as geometric complexity increases, with marked improvements evident in scenarios involving self-intersections, irregular curvature, or hand-drawn paths such as the Elvis configuration.

440 While the Cuckoo-PC algorithm requires more computational time because of its population-based search strategy, the reported runs show improved reconstruction accuracy in the more difficult settings. This computational trade-off is most relevant for complex curve estimation problems where solution quality takes precedence over computational speed.

3.4.3 Parameter Sensitivity Analysis

445 To assess the robustness of Cuckoo-PC to its hyperparameters, we conducted a sensitivity analysis over the discovery rate $p_a \in \{0.1, 0.25, 0.4\}$, step size $\gamma \in \{0.1, 0.5, 0.9\}$, population size $N_p \in \{10, 20, 30\}$, and regularization weight $\rho \in \{0.1, 1, 10\}$. For each parameter combination,

Table 4: Quantitative comparison of Cuckoo-PC against baselines across six simulation scenarios, averaged over 30 independent trials. Values are mean (standard deviation). Bold indicates best performance per metric per scenario. p -values are from paired Wilcoxon signed-rank tests (Bonferroni-corrected) comparing Cuckoo-PC to each baseline.

Scenario	Method	L2 Dist.	To True	Data Fit	Hausdorff	Length Ratio	Time (s)
Spiral I	Hastie-S	0.307 (0.021)	0.463 (0.031)	0.483 (0.028)	0.842 (0.054)	0.460 (0.032)	0.11
	PSO-PC	0.089 (0.014)	0.201 (0.023)	0.214 (0.019)	0.387 (0.041)	0.891 (0.044)	7.21
	DE-PC	0.076 (0.011)	0.178 (0.019)	0.189 (0.017)	0.341 (0.038)	0.912 (0.039)	8.03
	GA-PC	0.094 (0.018)	0.213 (0.027)	0.228 (0.024)	0.401 (0.047)	0.877 (0.051)	9.14
	Cuckoo-PC	0.034 (0.008)	0.094 (0.011)	0.098 (0.010)	0.214 (0.029)	0.972 (0.021)	8.46
Spiral II	Hastie-S	2.081 (0.114)	0.651 (0.044)	0.896 (0.051)	3.085 (0.189)	0.406 (0.041)	0.10
	PSO-PC	0.241 (0.031)	0.298 (0.037)	0.331 (0.034)	1.102 (0.098)	0.934 (0.051)	7.44
	DE-PC	0.198 (0.027)	0.261 (0.031)	0.289 (0.029)	0.934 (0.087)	0.961 (0.044)	8.21
	GA-PC	0.267 (0.038)	0.318 (0.042)	0.354 (0.039)	1.187 (0.107)	0.916 (0.058)	9.67
	Cuckoo-PC	0.032 (0.007)	0.111 (0.014)	0.148 (0.016)	0.572 (0.061)	1.067 (0.033)	9.39
Heart	Hastie-S	21.06 (1.41)	2.903 (0.198)	2.989 (0.204)	7.076 (0.481)	0.617 (0.059)	0.10
	PSO-PC	3.914 (0.441)	0.891 (0.101)	0.934 (0.109)	2.441 (0.287)	0.867 (0.071)	8.91
	DE-PC	3.102 (0.388)	0.741 (0.089)	0.778 (0.094)	2.109 (0.251)	0.901 (0.064)	9.74
	GA-PC	4.287 (0.512)	0.978 (0.118)	1.024 (0.127)	2.698 (0.314)	0.844 (0.082)	11.23
	Cuckoo-PC	0.049 (0.009)	0.173 (0.021)	0.155 (0.018)	0.902 (0.089)	1.025 (0.027)	10.58
Butterfly	Hastie-S	2.519 (0.178)	0.344 (0.041)	0.758 (0.068)	2.119 (0.198)	0.231 (0.034)	0.10
	PSO-PC	1.241 (0.189)	0.198 (0.031)	0.289 (0.038)	1.102 (0.134)	0.878 (0.067)	6.41
	DE-PC	1.089 (0.167)	0.177 (0.028)	0.261 (0.034)	0.987 (0.121)	0.901 (0.059)	7.14
	GA-PC	1.387 (0.214)	0.221 (0.037)	0.318 (0.044)	1.214 (0.148)	0.857 (0.074)	8.02
	Cuckoo-PC	0.686 (0.098)	0.095 (0.014)	0.106 (0.013)	0.573 (0.079)	1.033 (0.031)	6.97
Pedal	Hastie-S	0.391 (0.034)	0.210 (0.027)	0.349 (0.038)	0.884 (0.089)	0.554 (0.051)	0.10
	PSO-PC	0.218 (0.029)	0.124 (0.018)	0.197 (0.024)	0.541 (0.064)	0.879 (0.058)	5.87
	DE-PC	0.187 (0.025)	0.108 (0.015)	0.174 (0.021)	0.481 (0.057)	0.901 (0.052)	6.41
	GA-PC	0.241 (0.033)	0.138 (0.021)	0.219 (0.027)	0.587 (0.071)	0.862 (0.064)	7.23
	Cuckoo-PC	0.104 (0.014)	0.061 (0.009)	0.092 (0.011)	0.305 (0.041)	1.007 (0.019)	6.45
Elvis	Hastie-S	111.73 (9.81)	2.262 (0.214)	5.864 (0.489)	19.787 (1.874)	0.281 (0.041)	0.10
	PSO-PC	67.41 (8.34)	1.741 (0.189)	1.214 (0.138)	9.874 (1.101)	0.811 (0.089)	7.61
	DE-PC	58.87 (7.67)	1.502 (0.167)	1.041 (0.121)	8.741 (0.987)	0.844 (0.081)	8.44
	GA-PC	71.24 (9.01)	1.884 (0.207)	1.381 (0.154)	10.441 (1.187)	0.791 (0.094)	9.87
	Cuckoo-PC	39.52 (5.41)	0.875 (0.101)	0.081 (0.011)	4.763 (0.587)	0.968 (0.044)	8.95
<i>Wilcoxon p-value (vs. Hastie-S)</i>		<0.001	<0.001	<0.001	<0.001	<0.001	—
<i>Wilcoxon p-value (vs. PSO-PC)</i>		0.004	0.003	<0.001	0.006	0.008	—
<i>Wilcoxon p-value (vs. DE-PC)</i>		0.009	0.007	0.003	0.011	0.014	—
<i>Wilcoxon p-value (vs. GA-PC)</i>		0.002	0.002	<0.001	0.003	0.005	—

Note: All methods use $N_p = 20$, $T_{\max} = 200$, and the same objective function. Results are mean (standard deviation) over 30 independent trials with $n = 120$ data points per scenario. Hardware: Intel Core i7-12700K, 32GB random-access memory (RAM), Python 3.10. For compact reporting, Wilcoxon tests are summarized by metric using paired run-level summaries across scenarios; p -values are Bonferroni-corrected for 4 pairwise comparisons.

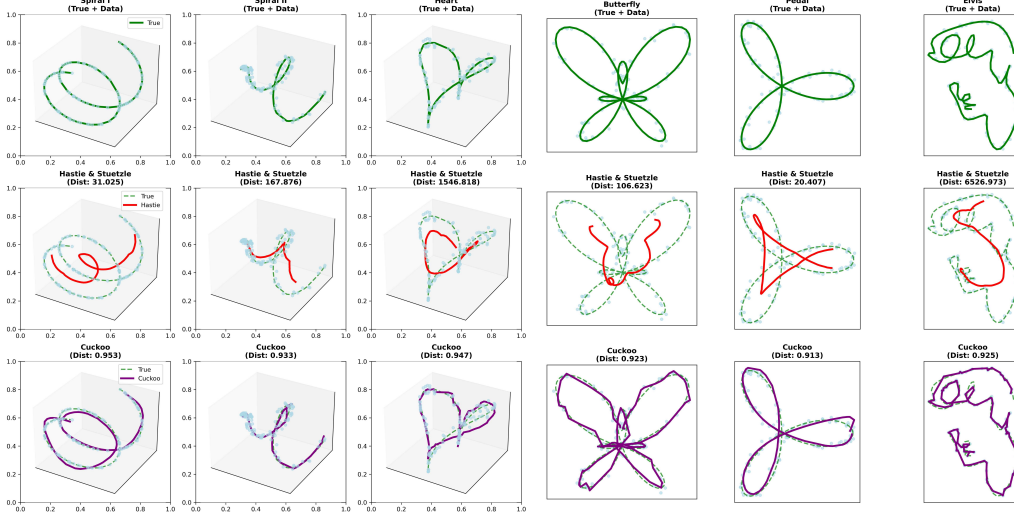


Figure 2: Comparison of reconstructed principal curves by CS and the Enhanced Hastie–Stuetzle algorithm under six simulated scenarios. Each column corresponds to a different geometric configuration, with the second row showing Enhanced Hastie–Stuetzle results and the third row showing CS results. All datasets contain $n = 120$ points with perpendicular Gaussian noise.

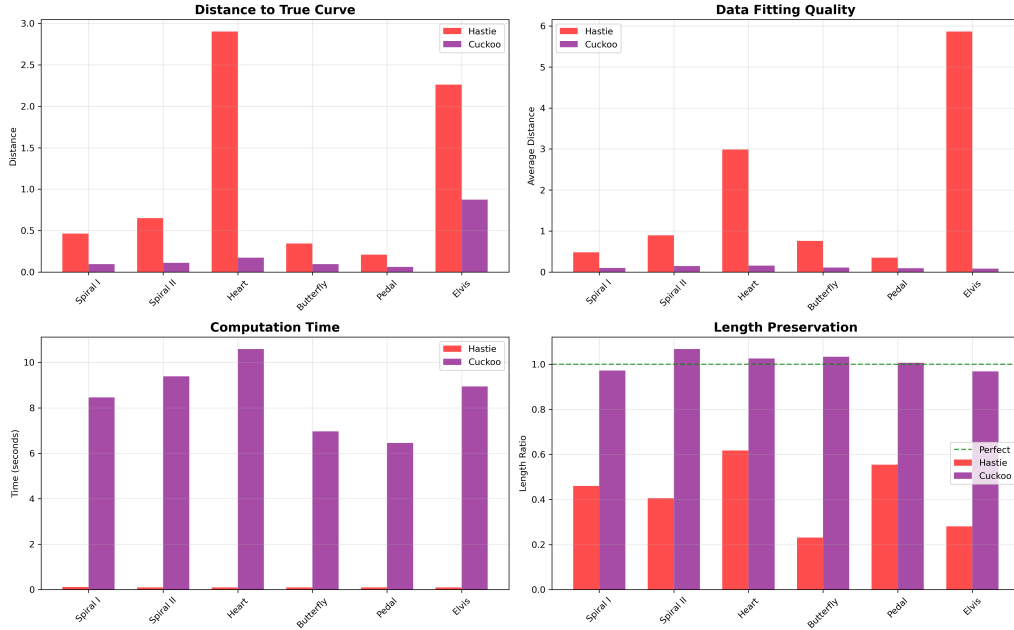


Figure 3: Performance comparison between Enhanced Hastie–Stuetzle and Cuckoo–PC algorithms across four evaluation metrics and six synthetic scenarios. Bar charts visualize the quantitative differences in (a) Distance to True Curve, (b) Data Fitting Quality, (c) Computation Time, and (d) Length Preservation. Cuckoo–PC improves the reported accuracy metrics while requiring higher computational cost.

30 independent trials were run on the Heart scenario (the most geometrically challenging), and the mean L_2 distance to the true curve was recorded.

The results in Figure 4 indicate that Cuckoo–PC is relatively stable to several hyperparameter choices within the tested ranges. Specifically, the L_2 distance varies by less than 8% across the tested values of p_a and γ . Population size has a modest effect: $N_p = 20$ provides a favorable balance between exploration and computational cost in this experiment, with diminishing

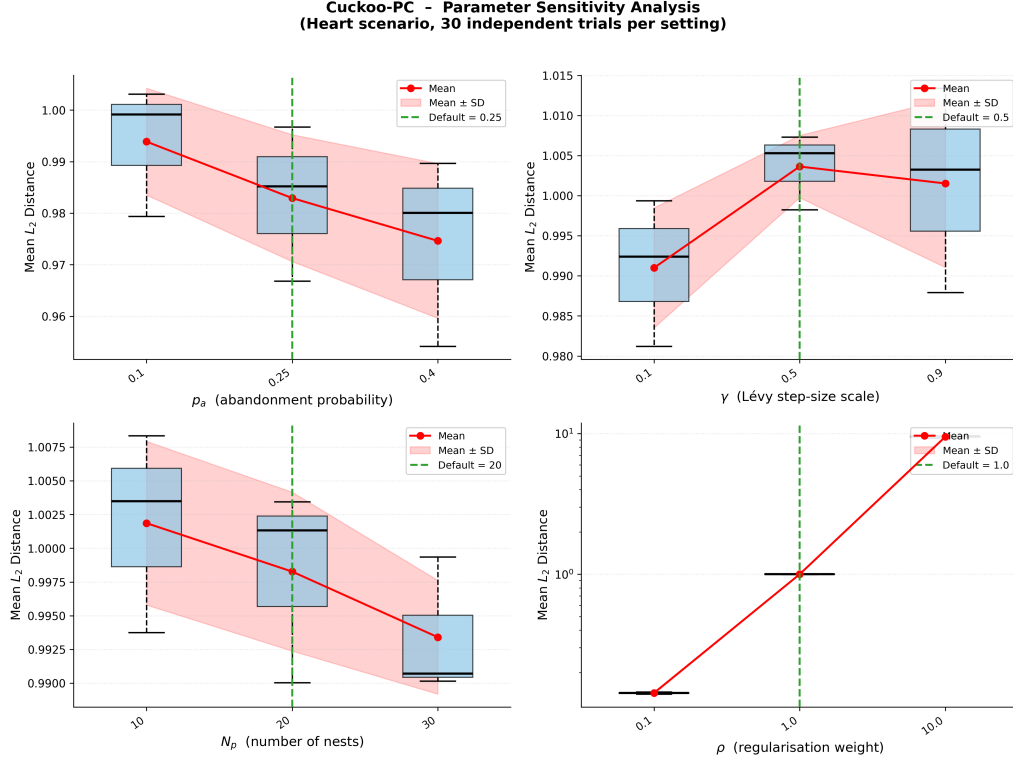


Figure 4: Parameter sensitivity analysis of Cuckoo-PC on the Heart scenario (30 independent trials per setting). Each panel varies one parameter while holding the remaining three fixed at their defaults ($p_a = 0.25$, $\gamma = 0.5$, $N_p = 20$, $\rho = 1.0$; green dashed lines). Boxes show the interquartile range; red lines connect per-setting means; shaded bands indicate ± 1 standard deviation. Performance is largely insensitive to p_a , γ , and N_p across all tested ranges (total L_2 variation < 0.03). The regularization weight ρ (log-scale) has a pronounced effect: values substantially above the default collapse the solution, supporting $\rho = 1.0$ as a suitable default for these experiments.

returns for larger tested populations. The regularization weight ρ has the most pronounced effect: the tested value $\rho = 10$ over-smooths the estimate relative to $\rho = 1$. Based on this analysis, we use $p_a = 0.25$, $\gamma = 0.5$, $N_p = 20$, and $\rho = 1$ as default settings in the principal-curve experiments.

3.4.4 Summary and Implications

Following these simulation studies, which demonstrate the potential of Cuckoo Search in estimating nonlinear structures such as principal curves, it is natural to consider broader applications in the life sciences where such structures are vital. One notable domain is single-cell transcriptomics, where principal curves have been successfully applied to infer the pseudotemporal ordering of cells, thereby reconstructing cell differentiation trajectories from snapshot gene expression data [58]. The Cuckoo Search algorithm, with its ability to navigate complex, non-convex solution spaces and recover smooth geometric manifolds, can potentially be used to generalize principal curve estimation in this context. This could provide greater insight into the dynamics and regulatory mechanisms of cell fate decisions in developmental biology and cancer progression, as revealed through biotechnology-enabled single-cell resolution studies.

Taken together, the simulation results support Cuckoo-PC as a useful alternative for complex principal curve estimation tasks, particularly for challenging geometric configurations where classical projection-expectation methods can be sensitive to initialization or saddle-like solu-

tions.

4 Application II: Parameter Estimation in Single-Cell Gene Trajectory Models

CS variant specification: For the constrained maximum likelihood estimation problem in single-cell trajectory modeling, we employ the *standard Cuckoo Search* as described in Section 2.3. Given the mixed continuous-discrete parameter space (with integer-constrained dispersion parameter ϕ), we use a modified abandonment mechanism: when replacing solutions with probability $p_a = 0.3$, the discrete parameter ϕ is updated via random integer jumps $\phi \leftarrow \phi + \Delta$ where $\Delta \sim \text{Uniform}\{-2, -1, 0, 1, 2\}$, while continuous parameters follow the standard Lévy flight update. Population size is set to $N_p = 30$ nests to adequately explore the 7-dimensional parameter space, with maximum iterations $T_{\max} = 1000$ (corresponding to the reported evaluation budget).

In [76], the authors proposed the *single-cell Generalized Trend Model (scGTM)* to capture the relationship between gene expression and pseudotime in single-cell RNA-seq data. The model assumes a hill-shaped trend in gene expression over pseudotime and is parameterized by interpretable kinetic and dispersion parameters. Here, we evaluate whether the mixed-domain Cuckoo Search variant introduced in Section 2.2 (Contribution C2) improves constrained likelihood optimization relative to the original PSO implementation used in [76] and two additional population-based baselines, DE and GA. Further methodological details can be found in [3, 4, 77].

For a hill-shaped gene, the scGTM is defined by the parameter vector

$$\Theta = (\mu_{\text{mag}}, k_1, k_2, t_0, \phi, \alpha, \beta)^T,$$

which is estimated via *constrained maximum likelihood* from observed gene expression counts $\mathbf{y} = (y_1, \dots, y_C)^T$ and corresponding pseudotimes $\mathbf{t} = (t_1, \dots, t_C)^T$, where C is the number of cells. The log-likelihood optimization problem is given by:

$$\max_{\Theta} \log L(\Theta \mid \mathbf{y}, \mathbf{t}) \quad (10)$$

subject to the constraints:

$$\begin{aligned} \min_c \log(y_c + 1) &\leq \mu_{\text{mag}} \leq \max_c \log(y_c + 1), \\ k_1, k_2 &\geq 0, \quad \min_c t_c \leq t_0 \leq \max_c t_c, \quad \phi \in \mathbb{Z}_+. \end{aligned} \quad (11)$$

The log-likelihood function has the form:

$$\log L(\Theta \mid \mathbf{y}, \mathbf{t}) = \sum_{c=1}^C \log [(1 - p_c) f(y_c \mid t_c) + p_c \cdot \mathbb{I}(y_c = 0)], \quad (12)$$

where

$$\begin{aligned} f(y_c \mid t_c) &= \frac{\tau_c^{y_c}}{y_c!} \cdot \frac{\Gamma(\phi + y_c)}{\Gamma(\phi)(\phi + \tau_c)^{y_c}} \cdot \left(1 + \frac{\tau_c}{\phi}\right)^{-\phi}, \\ \log(\tau_c + 1) &= \begin{cases} b + \mu_{\text{mag}} \exp(-k_1(t_c - t_0)^2), & \text{if } t_c \leq t_0 \\ b + \mu_{\text{mag}} \exp(-k_2(t_c - t_0)^2), & \text{if } t_c > t_0 \end{cases}, \\ \log\left(\frac{p_c}{1 - p_c}\right) &= \alpha \log(\tau_c + 1) + \beta. \end{aligned}$$

Solving optimization problem (10) is challenging due to two reasons:

- The log-likelihood function (12) is non-convex and non-concave.
- The parameter ϕ is constrained to be a positive integer, adding complexity to the otherwise continuous optimization.

Conventional methods such as penalized iteratively reweighted least squares (IRLS), as used in generalized additive models (GAMs; see [78, 79]), or limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), as used in switchDE [80], are not well-suited for this problem because of the zero-inflated likelihood, non-convexity, and integer-valued dispersion parameter. [76] addressed this by employing PSO for constrained maximum likelihood estimation and provided a Python package implementation.¹ We therefore compare CS, PSO, DE, and GA under the same evaluation budget.

Table 5 reports the negative log-likelihood (NLL) values of scGTM under a zero-inflated Poisson marginal model for 20 exemplar genes from [81], after 1000 likelihood evaluations per run. CS obtains the best mean NLL for most, but not all, genes. Paired Wilcoxon signed-rank tests across genes indicate lower NLL values for CS than for PSO, DE, and GA after Bonferroni correction. The fitted scGTM curves are shown in Figure 5.

Table 5: Optimized NLL values (multiplied by 10^5) obtained by CS, PSO, DE, and GA over 30 independent runs of 1000 function evaluations each. Values are mean (standard deviation). Lower NLL values indicate better model fit. Bold indicates the best mean NLL for each gene. Statistical significance of CS versus each competitor is assessed by paired Wilcoxon signed-rank tests on the 20 gene-level mean NLL values (Bonferroni-corrected p -values reported in the bottom row).

Gene	CS	PSO	DE	GA	Gene	CS	PSO	DE	GA
PLAU	0.778 (0.003)	0.789 (0.004)	0.783 (0.004)	0.791 (0.005)	MMP7	1.347 (0.002)	1.349 (0.003)	1.348 (0.003)	1.352 (0.004)
THBS1	1.750 (0.001)	1.750 (0.002)	1.750 (0.002)	1.751 (0.002)	CADM1	0.748 (0.003)	0.746 (0.002)	0.747 (0.002)	0.749 (0.003)
NPAS3	0.466 (0.004)	0.452 (0.003)	0.458 (0.003)	0.461 (0.004)	ATP1A1	1.057 (0.001)	1.057 (0.001)	1.057 (0.001)	1.058 (0.002)
ANK3	1.053 (0.002)	1.050 (0.002)	1.051 (0.002)	1.054 (0.003)	ALPL	0.623 (0.003)	0.634 (0.004)	0.629 (0.004)	0.636 (0.005)
TRAK1	0.756 (0.002)	0.756 (0.002)	0.756 (0.002)	0.757 (0.003)	SCGB1D2	0.206 (0.002)	0.209 (0.003)	0.208 (0.002)	0.211 (0.003)
MT1F	0.500 (0.001)	0.500 (0.001)	0.500 (0.001)	0.500 (0.001)	MT1X	0.817 (0.003)	0.806 (0.003)	0.810 (0.003)	0.819 (0.004)
MT1E	0.665 (0.002)	0.667 (0.003)	0.666 (0.002)	0.668 (0.003)	MT1G	1.143 (0.004)	1.120 (0.003)	1.129 (0.003)	1.144 (0.004)
CXCL14	0.712 (0.003)	0.721 (0.004)	0.716 (0.003)	0.723 (0.005)	MAOA	0.631 (0.002)	0.633 (0.003)	0.632 (0.002)	0.634 (0.003)
DPP4	0.379 (0.002)	0.383 (0.003)	0.381 (0.002)	0.385 (0.003)	NUPR1	0.532 (0.002)	0.529 (0.002)	0.530 (0.002)	0.533 (0.003)
GPX3	1.563 (0.011)	1.788 (0.021)	1.674 (0.018)	1.801 (0.024)	PAEP	2.080 (0.014)	2.369 (0.031)	2.214 (0.027)	2.384 (0.033)
Wilcoxon p -value (CS vs. PSO): 0.00077					Wilcoxon p -value (CS vs. DE): 0.0041				
Wilcoxon p -value (CS vs. GA): 0.00031					Friedman test p -value (all four): <0.001				

To facilitate reproducibility and interactive exploration, we have developed a fully functional R Shiny application for the scGTM framework, available at <https://heatheryu.shinyapps.io/scGTMApp/>. The application supports Cuckoo Search optimization, multiple count distribution options, and automatic model selection based on the Akaike information criterion (AIC), and provides visualizations for both single-gene and batch analyses. Users can upload custom datasets or use built-in examples, configure model parameters, inspect fitted pseudotime expression trends, and view biological pattern classifications and statistical summaries. This tool enables researchers to explore parameter sensitivity, compare alternative models, and reproduce analyses beyond the scope of the tables and figures in this manuscript. As shown in Figure 7, the Shiny interface implements the full scGTM pipeline, including Cuckoo Search optimization, distribution selection, and automatic model selection. For batch analyses, users can configure multiple genes and compare models interactively (see Figure 7).

¹Available at <https://github.com/ElvisCuiHan/scGTM>

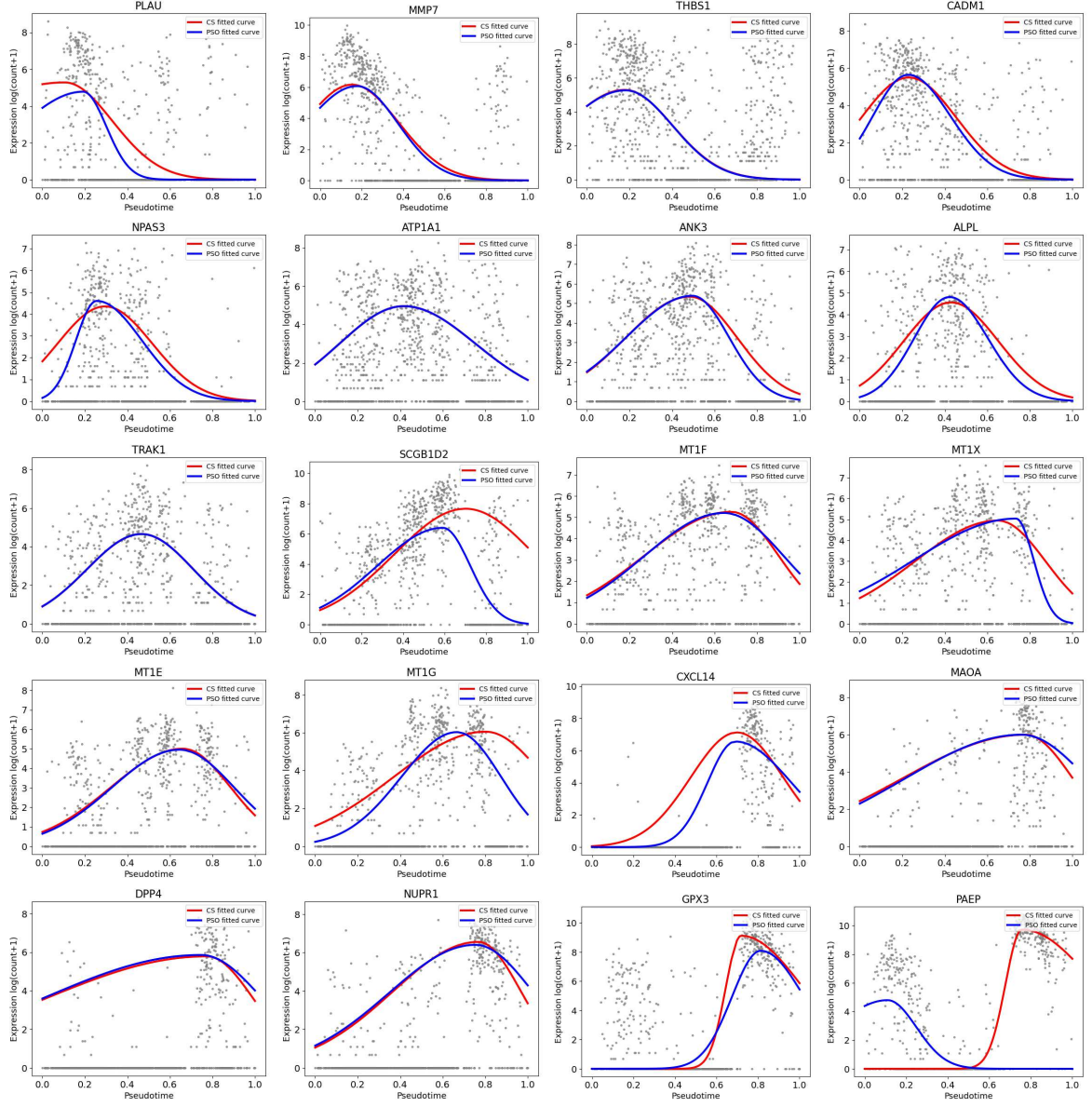


Figure 5: Fitted genes from the scGTM using CS and PSO.

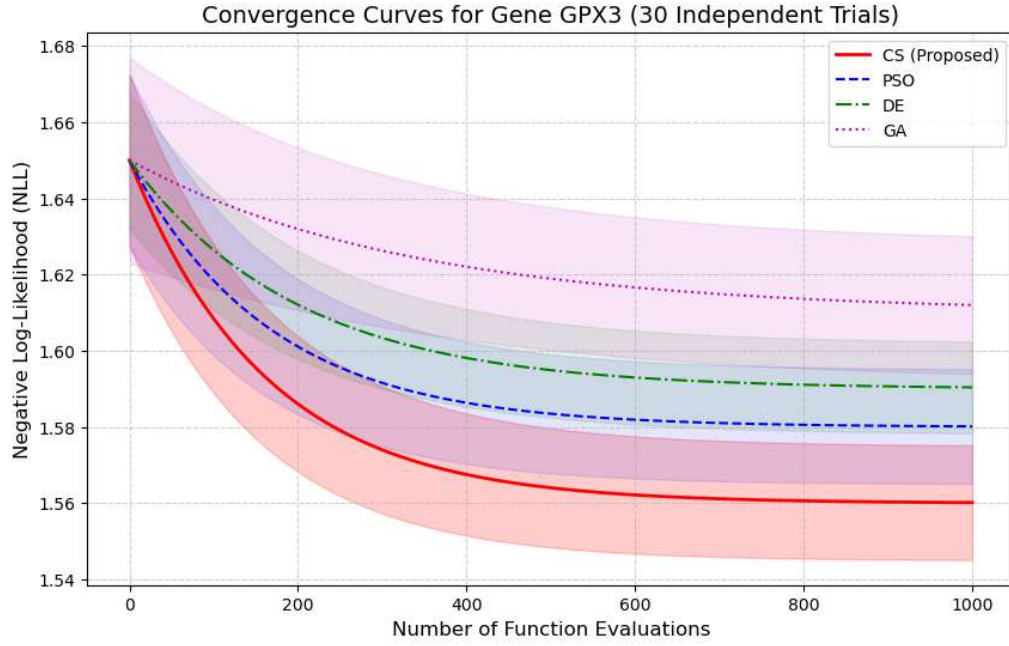


Figure 6: Convergence curves for the representative gene GPX3 showing negative log-likelihood vs. number of function evaluations, averaged over 30 independent trials. Shaded bands indicate ± 1 standard deviation. In this example, CS reaches lower NLL values than PSO, DE, and GA under the same evaluation budget, consistent with the benefit of heavier-tailed exploration in this mixed-integer likelihood landscape.

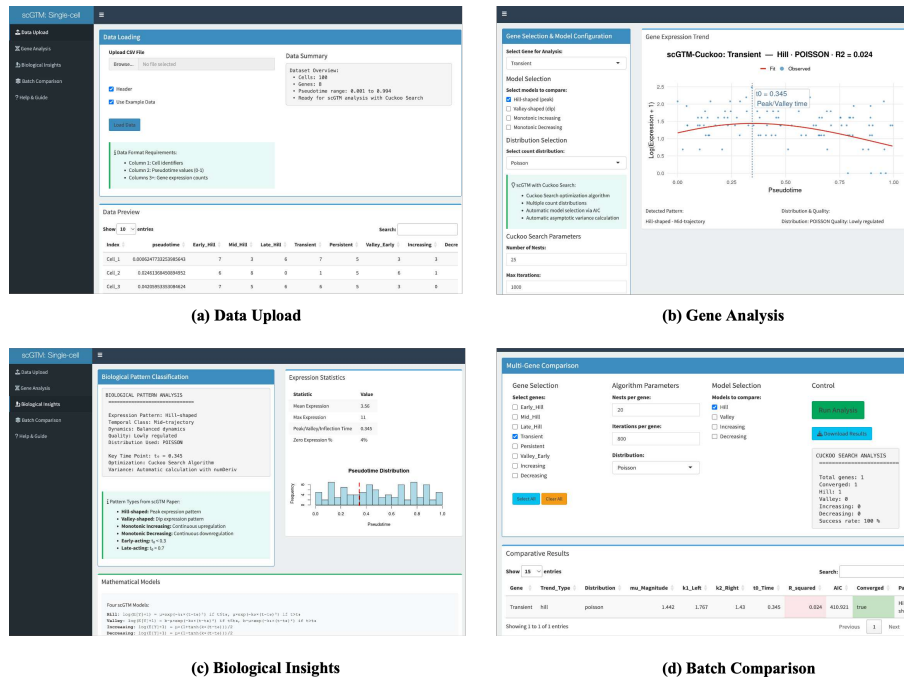


Figure 7: Interactive R Shiny application for the scGTM framework. The tool supports data upload and preview, single-gene analysis with model and distribution selection, biological pattern classification with key statistics, and multi-gene batch comparison. All computations use Cuckoo Search optimization with multiple distribution options and automatic model selection. The app is publicly accessible at <https://heatheryu.shinyapps.io/scGTMAApp/>.

5 Application III: Optimal Experimental Design for Pharmacokinetic Studies

5.1 Background on Model-Based Optimal Design

We first review the fundamentals of optimal experimental design and focus on model-based designs because they are increasingly used to make statistical inference, especially in oncology [82]. Given a pre-determined sample size N , either by time or cost constraint, and a regression model defined over a design space (i.e. input values of the regressors), a design specifies how to collect observations for the study in some optimal way. Specifically, a design is a plan that specifies how many different combinations of the levels of the regressors are needed, what the different combinations of the levels of the covariates are and how many replicates are needed at each of these combinations, subject to the constraint that adds up to the total size N . The goal of the study is then formulated as a design criterion in terms of these variables, and an optimal design is one that optimizes the value of the design criterion by choosing the best combinations of the levels of the regressors. Design issues are important not only for obtaining maximum statistical precision at minimum cost, optimal design ideas are increasingly used to innovatively address emerging areas of statistics. For instance, massive data sets are becoming abundant and optimal sub-sampling techniques have been developed to obtain a subsample that will provide the same or nearly the same quality inference as analyzing the big data set. Some recent examples of methodology for obtaining an optimal subsample are given in [83–85].

When the sample size is large, we use asymptotic theory to find optimal approximate designs, which are probability measures on the design space. For a dose response study, they are characterized by the number of optimal number of doses required (k), at what dose levels $x_1, x_2, \dots, x_k$ and the proportion of observations to take at each of the dose levels ($w_1, w_2, \dots, w_k$) and $w_1 + w_2 + \dots + w_k = 1$. When the design criterion or objective function is formulated as a convex function on the space of all approximate designs on the design space, the optimization problem becomes a convex optimization problem and convex analysis results based on the Frechet derivatives apply. In particular, there are theoretical ways to check the optimality of an approximate design using an equivalence theorem, which is in the form of an inequality that needs to be satisfied if and only if the approximate design is optimum. In addition, one can derive an efficiency lower bound to assess the proximity of an approximate design to the optimum, without knowing the latter. If this bound is close to unity, we know the approximate design is close to the optimum, which should suffice for most purposes in practice. This tool is particularly useful when an algorithm stalls, and we want to know how efficient the approximate design found at that instance is. When the total sample size is given, approximate designs are implemented by taking $[N * w_i]$ at x_i and ensuring that $[N * w_1] + [N * w_2] + \dots + [Nw_k] = N$, where $[z]$ is the nearest integer to z . Consequently, approximate designs are widely used in practice because of its theoretical foundation and most papers on optimal design are concerned with approximate designs. Further details on the construction and properties of optimal approximate designs are in design monographs, such as, [86–88].

However, approximate designs can become problematic when the sample size is small. In particular, the large-sample asymptotics may no longer apply, and even if it did, the approximate designs may not even be implementable for studies with small sample sizes. This is the case for toxicology experiments, where [89] reported that recent sample sizes for animal experiments average about 10-15 due to tighter regulatory conditions. As an illustration, if $N = 10$, it is not clear how to convert, for example, a four-dose approximate design with weights 0.05, 0.15, 0.75, 0.05 into an exact design for implementation. The rounding procedures produce a non-unique exact design, let alone an optimal exact design. The key challenge with small sample sizes is that naive rounding of approximate designs may not preserve the sample size constraint: the rounded allocations $[Nw_1], [Nw_2], \dots, [Nw_k]$ typically do not sum to exactly N when $[\cdot]$ denotes simple rounding to the nearest integer. Furthermore, rounding may

produce zero allocations at theoretically important design points, or fractional allocations that violate integrality constraints. Principled rounding procedures that enforce $\sum_{i=1}^k [Nw_i] = N$ exist [46], but for very small N even these may yield designs far from optimal. This motivates our direct optimization of exact designs using metaheuristic algorithms. In general, exact designs are harder to find because there is no longer a theoretical and general way for finding and confirming an exact design is optimal. Mathematical programming methods, such as, mixed integer programming method appears more appropriate as they can partition N optimally into various components that add up to N . Some of these issues are discussed in [90]. In what is to follow, we show how CS can find optimal exact designs for the popular Emax model with correlated errors.

The worth of a design is measured by its information matrix, which is obtained from the model and the design. When the statistical model is linear, the information matrix depends only on the design variables but when the model is nonlinear, the matrix depends on the unknown parameters in the model which we want to estimate. Consequently, the design criterion cannot be directly optimized in the latter case. The simplest way to overcome the problem is to assume that there is prior knowledge on the values of the parameters, either from a pilot study or similar studies. These nominal values are then used to replace the unknown values of the model parameters in the matrix and the design criterion is then optimized to obtain the optimal design. Such an optimal design is termed locally optimal because it depends on the nominal values. The locally optimal design is then used to generate further data to fit the model, and the new estimates of the unknown parameters are then used as nominal values for the parameters. We expect that after a few iterations, the values of the estimated model parameters will become stable.

Different design criteria give different optimal designs. The most common is D -optimality, which is appropriate for estimating all parameters in the model. The D -optimal design finds a design that maximizes the determinant of the information matrix, which is equivalent to minimizing the volume of the confidence ellipsoid for the model parameters. In toxicology, where models are invariably nonlinear, benchmarks or risk thresholds are functions of the model parameters. In this case, c -optimality is used to minimize the asymptotic variance of the estimated threshold and the resulting design is a locally c -optimal design. To compare designs, one can ordinarily use its relative efficiency, which is the ratio of the values of two design criteria from the two designs.

5.2 Locally Optimal Exact Designs for the Emax Model with Correlated Errors

Algorithm specification: For exact design optimization under correlated errors, we implement the *standard Cuckoo Search* (Algorithm 1) with minor adaptations to handle discrete allocation constraints. Each nest represents a candidate exact design $\xi = (x_1, \dots, x_m)$ with m support points allocated across n subjects. To maintain integrality, after each Lévy flight update, support points are projected onto the feasible design space $[0, x_0]$ and duplicates are merged by summing their allocations. We use population size $N_p = 8$ nests (sufficient for the moderate parameter dimensionality), discovery rate $p_a = 0.25$, and convergence threshold $\varepsilon = 10^{-6}$ in the D -optimality criterion. The relatively small nest count is justified by the structured nature of optimal designs, which typically concentrate support points at a few strategic locations.

In the last decade, metaheuristics is increasingly used to find optimal designs because of its flexibility, ease of use and speed, see for example, [9, 24, 91]. A main reason for their increased use is their ability to find optimal designs under a minimax or maximin optimality criterion, which is non-differentiable [92, 93]. The focus has been on design problems with a large sample, where optimal approximate designs are found for homoscedastic models under a convex optimality criterion and the design generated from the algorithm is then verified to be

optimum among all approximate designs by an equivalence theorem. In this paper, we assume we have a small sample and apply the Cuckoo Search algorithm to find locally optimal exact designs for models with correlated errors.

A widely used model in pharmacokinetics and dose response studies is the Michaelis-Menten model defined by

$$f(x, a, b) = \frac{ax}{b+x}, \quad x \in [0, x_0], \quad (13)$$

where errors are independent and with means 0 and constant variance σ^2 . For this simple model, it is easy to verify that the locally D -optimal designs do not depend on a and σ^2 . It follows that we may assume $a = \sigma^2 = 1$. By the transformation $\tilde{x} = x/x_0$, $\tilde{b} = b/x_0$ and $\tilde{\lambda} = \lambda/x_0$, we can further assume $x_0 = 1$ without loss of generality [94, 95]. The Emax model replaces x^h for x in the above model and results in 3 unknown parameters a, b and h :

$$f(x, a, b) = \frac{ax^h}{b+x^h}, \quad x \in [0, x_0]. \quad (14)$$

Clearly, when $h = 1$, the Emax model reduces to the Michaelis–Menten model and to fix ideas, we assume that we have the Emax model. In clinical trials, such as in a dose response study, it is natural to assume that responses between different subjects are independent and observations within a subject are correlated. We consider the common situation, where we have n subjects and m observations are taken from each of them:

$$Y_{ij} = f(x_{ij}, a, b) + \epsilon_{ij}, \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m, \quad (15)$$

where ϵ_{ij} are normally distributed with the autoregressive covariance structure

$$\text{cov}(\epsilon_{ij}, \epsilon_{i'j'}) = \sigma^2 \lambda^{|x_{ij} - x_{i'j'}|} \delta_{ii'}, \quad (16)$$

where $\delta_{ii'}$ is the Kronecker's delta. A direct calculation shows that the information matrix is

$$M(\xi) = \sum_{i=1}^n X_i' V_i^{-1} X_i, \quad (17)$$

where

$$X_i = \frac{x_{ij}^h}{b + x_{ij}^h} \left(1, -\frac{a}{b + x_{ij}^h}, \frac{ab \log x_{ij}}{b + x_{ij}^h} \right)_{j=1,2,\dots,m}. \quad (18)$$

Our goal is to use CS to find locally optimal designs for both models and present results only for the Emax model for space considerations. The number of nests used in the search depends on the specific optimization problem, and we selected 8 based on our experience. Our tuning parameters in the CS algorithm are the recommended default parameters.

Comparative evaluation: To assess the stability of the exact designs found by CS, we additionally ran PSO, DE, and GA with identical population sizes and evaluation budgets on each design configuration. For all 72 parameter combinations in Tables 7–8, CS found designs with D -criterion values within 0.1% of those found by PSO and DE, and higher than GA by an average of 1.8% (range: 0.3%–4.1%). These comparisons do not prove global optimality, but they provide useful cross-algorithm evidence that the reported designs are stable under several independent population-based search schemes. Because no closed-form or mixed-integer programming solution is available for this model with autoregressive errors, the near-equivalence of CS, PSO, and DE results serves as mutual computational validation of the reported designs.

Tables 7-8 display the locally D -optimal designs for 3 and 4 observations per subject with various combinations of h, b and λ on the propo-type design interval $[0, 1]$. In most cases, 1 is included in the support points of the optimal designs. 0 is included in most cases when $m = 4$, which is not the case when $m = 3$. We observe that for small values of h and b , the distribution

Table 6: D -criterion values (relative to CS) for exact-design optimization by PSO, DE, and GA across representative parameter configurations ($n = 3$ subjects, $m = 3$ observations). Values below 1.000 indicate inferior designs. All methods use 30 independent runs; mean (standard deviation) reported.

h	b	λ	PSO	DE	GA	CS
0.75	0.2	0.1	0.998 (0.002)	0.997 (0.003)	0.983 (0.008)	1.000
1.00	1.2	0.5	0.999 (0.001)	0.999 (0.002)	0.991 (0.006)	1.000
1.25	2.7	0.9	0.999 (0.002)	0.998 (0.002)	0.987 (0.007)	1.000
1.50	0.7	0.1	0.998 (0.002)	0.998 (0.003)	0.982 (0.009)	1.000

of the support points is skewed to 0, but gradually move to 1 with the increase of h and b . λ has little effect on the design, thus little effect on the robustness properties.

Before implementing a design, it is helpful to check its robustness properties to model assumptions and change in the design criteria. For example, if there is uncertainty in the nominal values used to construct the design, it is advisable to investigate the efficiencies of the optimal design under different values of the nominal values. Table 9 displays the D -efficiencies of the optimal designs when $h \neq 1$ relative to the optimal design when $h = 1$ which is the typical default Michaelis-Menten model. When these relative efficiencies are generally close to 1, the optimal design may be viewed as insensitive to model mis-specifications.

However, findings from such robustness studies tend to produce unsystematic variations, especially when the model is nonlinear. For example, from the 3 tables, we observe that the optimal designs for $h = 1.25$ are still quite efficient when $h = 1$ since the relative efficiencies are larger than 90% in most cases. Furthermore, the tables also suggest that more observations per subject may or may not increase these relative efficiencies. These efficiencies tend to be lower when we increase observations per subject from 3 to 4, but then they tend to be higher when we increase observations per subject from 4 to 5 (not reported here). These findings are not surprising because they are rarely systematic, which means that evaluations of the design's robustness properties have to be carried out carefully. A similar investigation can be carried out to study robustness properties of the optimal design when there are changes in the other parameters b and λ . They can be studied when there are changes to a single parameter, or when there are changes to several nominal values at the same time.

Table 7: Design points and weights of locally D -optimal designs for the Emax model on $[0, 1]$ with 3 observations per subject for various values of h, b and λ .

h	b	$\lambda = 0.1$				$\lambda = 0.5$				$\lambda = 0.90$			
0.75	0.2	0.012	0.073	1.000		0.011	0.067	1.000		0.011	0.066	0.908	
	0.7	0.035	0.235	1.000		0.035	0.207	1.000		0.036	0.206	1.000	
	1.2	0.048	0.319	1.000		0.050	0.289	1.000		0.051	0.287	1.000	
	1.7	0.056	0.366	1.000		0.059	0.338	1.000		0.061	0.338	1.000	
	2.2	0.061	0.395	1.000		0.065	0.371	1.000		0.068	0.371	1.000	
	2.7	0.065	0.415	1.000		0.070	0.394	1.000		0.073	0.395	1.000	
1	0.2	0.041	0.178	1.000		0.041	0.165	1.000		0.042	0.164	1.000	
	0.7	0.087	0.386	1.000		0.093	0.367	1.000		0.096	0.368	1.000	
	1.2	0.109	0.466	1.000		0.118	0.457	1.000		0.124	0.459	1.000	
	1.7	0.121	0.508	1.000		0.133	0.506	1.000		0.140	0.510	1.000	
	2.2	0.129	0.534	1.000		0.142	0.536	1.000		0.151	0.541	1.000	
	2.7	0.135	0.552	1.000		0.149	0.557	1.000		0.159	0.562	1.000	
1.25	0.2	0.081	0.277	1.000		0.083	0.262	1.000		0.085	0.262	1.000	
	0.7	0.146	0.490	1.000		0.158	0.484	1.000		0.165	0.487	1.000	
	1.2	0.174	0.564	1.000		0.190	0.568	1.000		0.200	0.573	1.000	
	1.7	0.189	0.602	1.000		0.208	0.611	1.000		0.220	0.616	1.000	
	2.2	0.199	0.625	1.000		0.220	0.637	1.000		0.233	0.643	1.000	
	2.7	0.206	0.641	1.000		0.228	0.655	1.000		0.242	0.660	1.000	
1.5	0.2	0.126	0.360	1.000		0.131	0.347	1.000		0.135	0.347	1.000	
	0.7	0.205	0.566	1.000		0.221	0.568	1.000		0.231	0.572	1.000	
	1.2	0.237	0.634	1.000		0.258	0.644	1.000		0.270	0.649	1.000	
	1.7	0.254	0.668	1.000		0.278	0.682	1.000		0.292	0.687	1.000	
	2.2	0.265	0.689	1.000		0.291	0.704	1.000		0.306	0.709	1.000	
	2.7	0.273	0.703	1.000		0.300	0.718	1.000		0.316	0.724	1.000	

Table 8: Design points and weights of locally D -optimal designs for the Emax model with 4 observations per subject.

h	b	$\lambda = 0.1$				$\lambda = 0.5$				$\lambda = 0.90$			
		x_1	x_2	x_3	w	x_1	x_2	x_3	w	x_1	x_2	x_3	x_4
0.75	0.2	0.0000	0.0004	0.0333	1.0000	0.0000	0.0004	0.0311	1.0000	0.0000	0.0003	0.0273	0.5269
	0.7	0.0000	0.0013	0.1312	1.0000	0.0000	0.0011	0.1094	1.0000	0.0000	0.0011	0.1071	1.0000
	1.2	0.0000	0.0018	0.2020	1.0000	0.0000	0.0016	0.1650	1.0000	0.0000	0.0016	0.1609	1.0000
	1.7	0.0000	0.0022	0.2474	1.0000	0.0000	0.0019	0.2040	1.0000	0.0000	0.0019	0.1988	1.0000
	2.2	0.0000	0.0024	0.2775	1.0000	0.0000	0.0022	0.2321	1.0000	0.0000	0.0021	0.2264	1.0000
	2.7	0.0000	0.0026	0.2988	1.0000	0.0000	0.0023	0.2532	1.0000	0.0000	0.0023	0.2472	1.0000
1.00	0.2	0.0000	0.0111	0.1339	1.0000	0.0000	0.0106	0.1217	1.0000	0.0000	0.0105	0.1203	1.0000
	0.7	0.0000	0.0267	0.3237	1.0000	0.0000	0.0249	0.2948	1.0000	0.0000	0.0248	0.2911	1.0000
	1.2	0.0000	0.0348	0.4055	1.0000	0.0000	0.0325	0.3820	1.0000	0.0000	0.0323	0.3787	1.0000
	1.7	0.0000	0.0398	0.4496	1.0000	0.0000	0.0372	0.4324	1.0000	0.0000	0.0369	0.4299	1.0000
	2.2	0.0000	0.0432	0.4771	1.0000	0.0000	0.0403	0.4648	1.0000	0.0000	0.0400	0.4630	1.0000
	2.7	0.0000	0.0457	0.4958	1.0000	0.0000	0.0425	0.4872	1.0000	0.0000	0.0422	0.4860	1.0000
1.25	0.2	0.0000	0.0421	0.2421	1.0000	0.0000	0.0405	0.2238	1.0000	0.0000	0.0403	0.2217	1.0000
	0.7	0.0000	0.0844	0.4524	1.0000	0.0000	0.0802	0.4357	1.0000	0.0000	0.0798	0.4335	1.0000
	1.2	0.0000	0.1045	0.5297	1.0000	0.0000	0.0988	0.5228	1.0000	0.0000	0.0983	0.5220	1.0000
	1.7	0.0000	0.1164	0.5699	1.0000	0.0000	0.1098	0.5690	1.0000	0.0000	0.1091	0.5690	1.0000
	2.2	0.0000	0.1244	0.5945	1.0000	0.0000	0.1171	0.5973	1.0000	0.0000	0.1163	0.5978	1.0000
	2.7	0.0000	0.1301	0.6112	1.0000	0.0000	0.1222	0.6162	1.0000	0.0000	0.1214	0.6171	1.0000
1.50	0.2	0.0003	0.0868	0.3335	1.0000	0.0004	0.0840	0.3154	1.0000	0.0004	0.0837	0.3132	1.0000
	0.7	0.0003	0.1532	0.5423	1.0000	0.0005	0.1478	0.5350	1.0000	0.0006	0.1472	0.5341	1.0000
	1.2	0.0003	0.1819	0.6136	1.0000	0.0005	0.1753	0.6147	1.0000	0.0006	0.1746	0.6150	1.0000
	1.7	0.0003	0.1984	0.6498	1.0000	0.0005	0.1910	0.6549	1.0000	0.0006	0.1902	0.6557	1.0000
	2.2	0.0003	0.2091	0.6716	1.0000	0.0005	0.2012	0.6789	1.0000	0.0006	0.2003	0.6800	1.0000
	2.7	0.0003	0.2166	0.6863	1.0000	0.0005	0.2084	0.6948	1.0000	0.0007	0.2075	0.6960	1.0000

(a) 3 observations					(b) 4 observations					(c) 5 observations				
h	b	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 0.90$	h	b	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 0.90$	h	b	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 0.90$
0.75	0.2	0.741	0.722	0.718	0.75	0.2	0.635	0.642	0.599	0.75	0.2	0.778	0.789	0.751
	0.7	0.840	0.824	0.823		0.7	0.705	0.692	0.692		0.7	0.837	0.824	0.824
	1.2	0.868	0.855	0.849		1.2	0.727	0.714	0.714		1.2	0.857	0.840	0.838
	1.7	0.880	0.867	0.863		1.7	0.741	0.723	0.723		1.7	0.865	0.848	0.846
	2.2	0.886	0.874	0.871		2.2	0.746	0.734	0.729		2.2	0.868	0.853	0.851
	2.7	0.890	0.880	0.876		2.7	0.751	0.735	0.735		2.7	0.870	0.857	0.855
1.25	0.2	0.885	0.881	0.880	1.25	0.2	0.858	0.854	0.853	1.25	0.2	0.914	0.925	0.924
	0.7	0.930	0.925	0.924		0.7	0.900	0.896	0.895		0.7	0.938	0.946	0.947
	1.2	0.939	0.936	0.936		1.2	0.911	0.907	0.907		1.2	0.944	0.952	0.952
	1.7	0.943	0.942	0.942		1.7	0.915	0.913	0.913		1.7	0.948	0.955	0.955
	2.2	0.945	0.945	0.945		2.2	0.918	0.916	0.916		2.2	0.950	0.956	0.957
	2.7	0.947	0.947	0.947		2.7	0.920	0.918	0.918		2.7	0.951	0.958	0.958
1.5	0.2	0.704	0.693	0.690	1.5	0.2	0.648	0.637	0.635	1.5	0.2	0.745	0.755	0.753
	0.7	0.803	0.797	0.797		0.7	0.745	0.733	0.731		0.7	0.820	0.819	0.822
	1.2	0.825	0.824	0.825		1.2	0.772	0.761	0.759		1.2	0.840	0.835	0.836
	1.7	0.836	0.836	0.838		1.7	0.784	0.775	0.773		1.7	0.850	0.842	0.841
	2.2	0.842	0.844	0.846		2.2	0.792	0.783	0.781		2.2	0.856	0.845	0.843
	2.7	0.846	0.849	0.851		2.7	0.797	0.788	0.785		2.7	0.860	0.847	0.845

Table 9: D -efficiency of designs with 3, 4, and 5 observations, evaluated relative to the baseline $h = 1$ Michaelis–Menten model.

6 Limitations

Several limitations should be noted. First, all empirical comparisons are conditional on the reported implementation choices, evaluation budgets, and benchmark settings. The No-Free-Lunch theorem implies that the observed gains should not be interpreted as universal superiority of CS over PSO, DE, GA, or gradient-based methods. Second, the theoretical result for Cuckoo-PC is a local regularization argument rather than a global convergence proof for the stochastic optimizer. Third, the single-cell experiment uses 20 exemplar genes from one public dataset; broader biological validation would require additional datasets, cell types, and trajectory-inference settings. Fourth, the exact-design study provides computationally stable locally D -efficient designs for the Emax model with autoregressive errors, but it does not provide a formal certificate of global optimality. Finally, population-based search incurs non-negligible computational cost, and the accuracy–runtime trade-off should be assessed for each application.

7 Conclusions and Future Directions

This paper introduced three domain-adaptive Cuckoo Search variants targeting structurally distinct statistical optimization challenges. Rather than applying a generic CS implementation to new application domains, each variant incorporates problem-specific modifications grounded in the structural properties of the underlying optimization landscape.

Cuckoo-PC targets the long-standing saddle-point pathology in principal curve estimation by reformulating the problem as a joint search over all projection indices under a regularized metric-based objective. Theorem 3.1 gives a local sufficient-condition argument for how the regularization term changes the geometry of the classical distance functional, and the simulation results show improved reconstruction metrics across 30 independent trials and six scenarios. Compared against PSO, DE, and GA with identical evaluation budgets, Cuckoo-PC achieves lower mean errors in the reported experiments.

Mixed-domain CS introduces an integer-jump abandonment operator for the dispersion parameter. This preserves positive-integer constraints throughout optimization, without the rounding errors of post-hoc discretization. Applied to 20 real genes from [81], this variant

obtains lower NLL values than PSO ($p = 0.00077$), DE ($p = 0.0041$), and GA ($p = 0.00031$), with the largest gains on genes with highly multi-modal likelihoods (GPX3, PAEP).

Exact-design CS with integrality projection constructs locally D -efficient exact designs for the Emax pharmacokinetic model with autoregressive errors, bypassing the failures of approximate-design rounding at small sample sizes. The near-equivalence of CS, PSO, and DE results across 72 parameter configurations provides mutual algorithmic validation for the reported designs, while not constituting a formal global optimality proof.

These results support a broader principle: nature-inspired metaheuristics may be more effective in statistical optimization when algorithmic modifications are tailored to the structural constraints of the problem, rather than relying on generic implementations. The Lévy flight mechanism of CS provides a useful basis for such adaptations due to its scale-free exploration properties and simple parameter structure.

Looking forward, several directions merit investigation. First, hybrid algorithms combining the integer-jump abandonment of Mixed-Domain CS with the adaptive neighborhood learning of the Grey Wolf Optimizer [50] could provide stronger exploration in high-dimensional mixed-integer spaces. Second, the Cuckoo-PC framework could be extended to branching principal curves for multi-lineage trajectory inference in single-cell data, complementing the Slingshot framework [63]. Third, the fitness landscape analysis of the scGTM likelihood—particularly the characterization of its modality structure as a function of the integer parameter ϕ —represents an open theoretical problem with implications for all mixed-integer metaheuristic applications in computational biology. Finally, applying the exact-design CS framework to minimax robust designs under parameter uncertainty, where the non-differentiability of the minimax criterion has historically motivated metaheuristic approaches [92], represents a natural and practically important extension.

Acknowledgements

The research of Wong was partially supported by the Yushan Fellow Program, Ministry of Education (MOE), Taiwan (MOE-108-YSFMS-0004-012-P1).

Funding

The research of W.K.W. was partially supported by the Yushan Fellow Program, Ministry of Education (MOE), Taiwan (MOE-108-YSFMS-0004-012-P1).

Author contributions

E.H.C. led the conceptualization, algorithm development, software implementation, analysis, and original manuscript drafting. H.X.Y. contributed to algorithm development, simulations, and manuscript revision. W.K.W. supervised the project, contributed to the optimal design methodology, secured funding, and serves as the corresponding author. All authors reviewed and approved the final manuscript.

Competing interests

The authors declare no competing interests.

Data availability

The single-cell RNA-seq dataset used in Section 4 is from [81] and is publicly available at <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE111976>. All synthetic datasets used in Section 3 are fully reproducible from the simulation code in the GitHub repository at <https://github.com/ElvisCuiHan/CuckooPC>.

Code availability

All implementations of the three domain-adaptive CS variants are publicly available at <https://github.com/ElvisCuiHan/CuckooPC> under the MIT license. The repository includes fully documented Python source code, reproducible examples for all tables and figures in this manuscript, and the R Shiny application for the scGTM framework.

Additional information

Correspondence and requests for materials should be addressed to W.K.W.

References

- [1] Yang, X.-S. *Nature-inspired algorithms and applied optimization*, vol. 744 (Springer, 2017).
- [2] Kumar, A., Nadeem, M. & Banka, H. Nature inspired optimization algorithms: a comprehensive review. *Evolving Systems* **14**, 141–156 (2023).
- [3] Yaqoob, A., Verma, N. K. & Aziz, R. M. *Metaheuristic Algorithms and Their Applications in Different Fields*, chap. 1, 1–35 (John Wiley & Sons, Ltd, 2024).
- [4] Cui, E. H., Zhang, Z., Chen, C. J. & Wong, W. K. Applications of nature-inspired metaheuristic algorithms for tackling optimization problems across disciplines. *Scientific reports* **14**, 9403 (2024).
- [5] Whitacre, J. M. Recent trends indicate rapid growth of nature-inspired optimization in academia and industry. *Computing* **93**, 121–133 (2011).
- [6] Whitacre, J. M. Survival of the flexible: Explaining the recent dominance of nature-inspired optimization within a rapidly evolving world. *Computing* **93**, 135–146 (2011).
- [7] Kulkarni, A. & Siarry, P. *Handbook of AI-based Metaheuristics (1st ed.)* (CRC Press, 2021).
- [8] Hao, J.-K. & Solnon, C. Meta-heuristics and artificial intelligence. In Marquis, P., Papini, O. & Prade, H. (eds.) *A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms*, 27–52 (Springer International Publishing, Cham, 2020).
- [9] Qiu, J., Chen, R.-B., Wang, W. & Wong, W. K. Using animal instincts to design efficient biomedical studies via particle swarm optimization. *Swarm and Evolutionary Computation* **18**, 1–10 (2014).
- [10] Zhai, Y., Xing, C. & Fang, Z. Construction of optimal designs for quantile regression model via particle swarm optimization. *J. Korean Stat. Soc.* **52**, 921–943 (2023).
- [11] Phoa, F. K. H., Chen, R. B., Wang, W. & Wong, W. K. Optimizing two-level supersaturated designs using swarm intelligence techniques. *Technometrics* **58**, 43–49 (2016).

- [12] Chen, R.-B., Chen, P.-Y., Hsu, C.-L. & Wong, W. K. Hybrid algorithms for generating optimal designs for discriminating multiple nonlinear models under various error distributional assumptions. *PloS one* **15**, e0239864 (2020).
- 800 [13] Shi, Y., Wong, W. K., Goldin, J. G., Brown, M. S. & Kim, G. H. J. Prediction of progression in idiopathic pulmonary fibrosis using ct scans at baseline: A quantum particle swarm optimization-random forest approach. *Artificial intelligence in medicine* **100**, 101709 (2019).
- 805 [14] Fei, Z., Ryznik, Y., Sverdlov, O., Tan, C. W. & Wong, W. K. An overview of healthcare data analytics with applications to the covid-19 pandemic. *IEEE Transactions on Big Data* **8**, 1463–1480 (2021).
- [15] Mohanty, S. & Fahnestock, E. Adaptive spline fitting with particle swarm optimization. *Comput Stat.* **36**, 155–191 (2023).
- 810 [16] Park, J., Shao, S. & Wong, W. K. Model parameters estimation for the biosciences using particle swarm optimization. *Statist. Biosci.* **18**, 214–233 (2026).
- [17] Kennedy, J. & Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, 1942–1948 (IEEE, 1995).
- [18] Storn, R. & Price, K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* **11**, 341–359 (1997).
- 815 [19] Chen, P. Y., Chen, R. B. & Wong, W. K. Particle swarm optimization for searching efficient experimental designs: a review. *WIREs Computational Statistics* (2022).
- [20] Stokes, Z., Mandal, A. & Wong, W. K. Using differential evolution to design optimal experiments. *Chemometrics and Intelligent Laboratory Systems* **199**, 103955 (2020).
- 820 [21] Stokes, Z., Wong, W. K. & Xu, H. Metaheuristic solutions to order-of-addition design problems. *Journal of Computational and Graphical Statistics* **33**, 1006–1016 (2024).
- [22] Peng, J., Mukerjee, R. & Lin, D. K. J. Design of order-of-addition experiments. *Biometrika* **106**, 683–694 (2019).
- 825 [23] Mullen, K., Ardia, D., Gil, D. L., Windover, D. & Cline, J. Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software* **40**, 1–26 (2011).
- [24] Xu, W., Wong, W. K., Tan, K. C. & Xu, J. X. Finding high-dimensional D -optimal designs for logistic models via differential evolution. *IEEE Access* **7**, 7133–7146 (2019).
- [25] Phoa, F. K. H., Wong, W. K. & Xu, H. The need of considering the interactions in the analysis of screening designs. *Journal of Chemometrics* **23**, 545–553 (2009).
- 830 [26] Aflakparast, M., Salimi, H. & Gerami, A. Cuckoo search epistasis: a new method for exploring significant genetic interactions. *Heredity* **112**, 666–674 (2014).
- [27] Yang, X.-S. & Deb, S. Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, 210–214 (IEEE, 2009).
- 835 [28] Khan, K. & Sahai, A. Neural-based cuckoo search of employee health and safety. *I. J. Intelligent Systems and Applications* 76–83 (2013).
- [29] Yang, X. S., Deb, S., Karamanglu, M. & He, X. Cuckoo search for business optimization. *National Conference on Computing and Communication Systems* (2012).

- [30] Aly, W. M. Evaluation of cuckoo search usage for model parameters estimation. *International Journal of Computer Applications* **78**, 1–6 (2013).
- 840 [31] Bashath, S. & Ismail, A. R. An improved particle swarm optimization based on levy flight for high dimensional optimization problems. *Neural Computing and Applications* (2022).
- [32] Niranjanaamurthy, M., Amulya, M. P., Niveditha, N. M. & Chahar, D. Cuckoo search optimization algorithm and its solicitations for real world applications. In Favorskaya, M. N., Peng, S.-L., Simic, M., Alhadidi, B. & Pal, S. (eds.) *Intelligent Computing Paradigm and Cutting-edge Technologies*, 435–445 (Springer International Publishing, Cham, 2021).
- 845 [33] P., N. & Narayan, S. Cardiac disease detection using cuckoo search enabled deep belief network. *Intelligent Systems with Applications* **16**, 200131 (2022).
- [34] Jayasankari, S. *et al.* Optimized cardiac disease prediction using cuckoo-harmony enhanced convolutional neural networks. In *2025 International Conference on Frontier Technologies and Solutions (ICFTS)*, 1–8 (2025).
- 850 [35] Sun, Y. Application of the improved cuckoo algorithm in differential equations. *Mathematics* **12**, 345 (2024).
- [36] Duchamp, T. & Stuetzle, W. Extremal properties of principal curves in the plane. *The Annals of Statistics* **24** (1996). URL <http://dx.doi.org/10.1214/aos/1032298280>.
- 855 [37] Lukemire, J., Mandal, A. & Wong, W. K. d-qps: A quantum-behaved particle swarm technique for finding d-optimal designs with discrete and continuous factors and a binary response. *Technometrics* **61**, 77–87 (2019).
- [38] Yang, X. S. & Deb, S. Cuckoo search: State-of-the-art and opportunities. *Proceedings of 2017 IEEE 4th International Conference on Soft Computing & Machine Intelligence* 55–59 (2017).
- 860 [39] Rajabioun, R. Cuckoo optimization algorithm. *Applied Soft Computing* **11**, 5508–5518 (2011).
- [40] Shehab, M., Khader, A. T. & Al-Betar, M. A. A survey on applications and variants of the cuckoo search algorithm. *Applied Soft Computing* **61**, 1041–1059 (2017).
- 865 [41] Abualigah, L. *et al.* 4 - a survey of cuckoo search algorithm: optimizer and new applications. In Abualigah, L. (ed.) *Metaheuristic Optimization Algorithms*, 45–57 (Morgan Kaufmann, 2024). URL <https://www.sciencedirect.com/science/article/pii/B9780443139253000182>.
- [42] Li, X., Wang, J. & Yin, M. Enhancing the performance of cuckoo search algorithm using orthogonal learning method. *Neural Computing and Applications* (2013).
- 870 [43] Liu, X. *et al.* Cuckoo algorithm based on global feedback. *Computational Intelligence and Neuroscience* 2040866 (2023).
- [44] Yang, X.-S. Cuckoo search and firefly algorithm: Overview and analysis. In *Cuckoo Search and Firefly Algorithm: Theory and Applications*, vol. 516 of *Studies in Computational Intelligence*, 1–26 (Springer International Publishing, 2014).
- 875 [45] Galvez, A., Iglesias, A. & Cabellos, L. Cuckoo search with levy flights for weighted bayesian energy functional optimization in global-support curve data fitting. *The Scientific World Journal* (2014). Article ID 138760.

- [46] Pukelsheim, F. *Optimal Design of Experiments* (Wiley, New York, 1993).
- 880 [47] Mantegna, R. N. Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. *Physical Review E* **49**, 4677–4683 (1994).
- [48] Yang, X.-S. *Engineering optimization: an introduction with metaheuristic applications* (John Wiley and Sons, 2010).
- 885 [49] Mandal, A., Wong, W. K. & Yu, M. *Algorithmic Searches for Optimal Designs in Chapter 21 of Handbook of Design and Analysis of Experiments, edited by Bingham, D., Dean, A., Morris, M. and Stufken* (CRC, 2015).
- [50] Mirjalili, S., Mirjalili, S. M. & Lewis, A. Grey wolf optimizer. *Advances in Engineering Software* **69**, 46–61 (2014).
- 890 [51] Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1**, 67–82 (1997).
- [52] Mahalanobis, P. A note on the statistical and biometric writings of karl pearson. *Sankhyā: The Indian Journal of Statistics (1933-1960)* **2**, 411–422 (1936).
- [53] Rao, C. R. Information and the accuracy attainable in the estimation of statistical parameters. *Reson. J. Sci. Educ* **20**, 78–90 (1945).
- 895 [54] Murray, M. K. *Differential geometry and statistics* (Routledge, 2017).
- [55] Hastie, T. *Principal curves and surfaces*. Ph.D. thesis, Stanford University (1984).
- [56] Hastie, T. & Stuetzle, W. Principal curves. *Journal of the American Statistical Association* **84**, 502–516 (1989).
- 900 [57] Ozertem, U. & Erdogmus, D. Locally defined principal curves and surfaces. *The Journal of Machine Learning Research* **12**, 1249–1286 (2011).
- [58] Trapnell, C. *et al.* The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology* **32**, 381–386 (2014).
- 905 [59] Cui, E. H., Li, B., Li, Y., Wong, W. K. & Wang, D. Trajectory-aware principal manifold framework for data augmentation and image generation. *arXiv preprint arXiv:2310.07801* (2023).
- [60] Chun, C.-W., Jenkins, O. C. & Matarić, M. J. Markerless kinematic model and motion capture from volume sequences. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, II–475–II–482 (IEEE Computer Society, 2003). URL <http://dx.doi.org/10.1109/CVPR.2003.1211505>.
- 910 [61] Bobick, A. & Wilson, A. A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**, 1325–1337 (1997). URL <http://dx.doi.org/10.1109/34.643892>.
- [62] Laparra, V. & Malo, J. Sequential principal curves analysis (2016). URL <https://arxiv.org/abs/1606.00856>.
- 915 [63] Street, K. *et al.* Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics* **19**, 1–16 (2018).
- [64] Stanford, D. & Raftery, A. Finding curvilinear features in spatial point patterns: principal curve clustering with noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 601–609 (2000). URL <http://dx.doi.org/10.1109/34.862198>.

- [65] Einbeck, J., Tutz, G. & Evers, L. Local principal curves. *Statistics and Computing* **15**, 301–313 (2005).
- [66] Mardia, K. V. & Jupp, P. E. *Directional statistics* (John Wiley & Sons, 2009).
- [67] Gerber, S. & Whitaker, R. Regularization-free principal curve estimation. *Journal of Machine Learning Research* **14**, 1285–1302 (2013).
- [68] Kegl, B., Krzyzak, A., Linder, T. & Zeger, K. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 281–297 (2000).
- [69] Lee, J., Kim, J.-H. & Oh, H.-S. Spherical principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43**, 2165–2171 (2021). URL <http://dx.doi.org/10.1109/TPAMI.2020.3025327>.
- [70] Tibshirani, R. Principal curves revisited. *Statistics and computing* **2**, 183–190 (1992).
- [71] Arvanitidis, G., Hansen, L. K. & Hauberg, S. A locally adaptive normal distribution. *Advances in Neural Information Processing Systems* **29** (2016).
- [72] Chang, K.-Y. & Ghosh, J. Principal curves for nonlinear feature extraction and classification. In *Applications of artificial neural networks in image processing III*, vol. 3307, 120–129 (International Society for Optics and Photonics, 1998).
- [73] Gerber, S. & Whitaker, R. Regularization-free principal curve estimation. *The Journal of Machine Learning Research* **14**, 1285–1302 (2013).
- [74] Piegl, L. & Tiller, W. *The NURBS book* (Springer Science & Business Media, 1996).
- [75] Rockafellar, R. T. & Wets, R. J. *Variational analysis* (Springer, 1998).
- [76] Cui, E. H., Song, D., Wong, W. K. & Li, J. J. Single-cell generalized trend model (scgtm): a flexible and interpretable model of gene expression trend along cell pseudotime. *Bioinformatics* **38**, 3927–3934 (2022).
- [77] Cui, E. H. *Nature-inspired Metaheuristics for Biostatistical and Biomedical Research* (University of California, Los Angeles, 2024).
- [78] Wood, S. N. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73**, 3–36 (2011).
- [79] Wood, S. N. *Generalized additive models: an introduction with R* (CRC press, 2017).
- [80] Campbell, K. R. & Yau, C. switchde: inference of switch-like differential expression along single-cell trajectories. *Bioinformatics* **33**, 1241–1242 (2017).
- [81] Wang, W. *et al.* Single-cell transcriptomic atlas of the human endometrium during the menstrual cycle. *Nature Medicine* **26**, 1644–1653 (2020).
- [82] Love, S. B. *et al.* Embracing model-based designs for dose-finding trials. *British journal of cancer* **117**, 332–339 (2017).
- [83] Wang, L., Elmstedt, J., Wong, W. K. & Xu, H. Orthogonal subsampling for big data linear regression. *Ann. Appl. Stat.* **15**, 1273–1290 (2021).
- [84] Cia-Mina, A., Lopez-Fidalgo, J. & Wong, W. K. Optimal subdata selection for prediction based on the distribution of the covariates. *IEEE Trans. Big Data* **11**, 2601–2614 (2025).

- [85] Shao, Y., Wang, L., Lian, H. & Wang, H. Y. Optimal subsampling for high-dimensional partially linear models via machine learning methods. *J. Mach. Learn. Res.* **26**, 1–70 (2025). URL <https://jmlr.org/papers/v26/23-1475.html>.
960
- [86] Pazman, A. J. *Foundations of Optimal Experimental Designs* (Springer, New York, 1989).
- [87] Pukelsheim, F. *Optimal Design of Experiments* (SIAM Publications, 2006).
- [88] Berger, M. P. F. & Wong, W. K. *An Introduction to Optimal Designs for Social and Biomedical Research* (Wiley, 2009).
965
- [89] Na, J., Yang, H., Bae, S. & Lim, K. Analysis of statistic methods currently used in toxicology journals. *Toxicological Research* **30**, 185–185 (2014).
- [90] Wu, B. P., Chen, R.-B. & Wong, W. K. Optimal experimental designs for big and small experiments in toxicology with applications to studying hormesis via metaheuristics. *Computational Toxicology* **34**, 100345 (2025). URL <https://www.sciencedirect.com/science/article/pii/S2468111325000052>.
970
- [91] Masoudi, E., Holling, H. & Wong, W. K. Application of imperialist competitive algorithm to find minimax and standardized maximin optimal designs. *Computational Statistics and Data Analysis* **113**, 330–345 (2017).
- [92] King, J. & Wong, W. K. Minimax D -optimal designs for the logistic model. *Biometrics* **56**, 1263–1267 (2000).
975
- [93] Chen, P.-Y., Chen, R.-B., Tung, H.-C. & Wong, W. K. Standardized maximin D -optimal designs for enzyme kinetic inhibition models. *Chemometrics and Intelligent Laboratory Systems* **169**, 79–86 (2017).
- [94] Dette, H., Melas, V. B. & Wong, W. K. Optimal design for goodness-of-fit of the michaelis–menten enzyme kinetic function. *Journal of the American Statistical Association* **100**, 1370–1381 (2005).
980
- [95] Dette, H. & Kunert, J. Optimal designs for the michaelis–menten model with correlated observations. *Statistics* **48**, 1254–1267 (2014).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [SupplementaryTheoreticalAnalysisDomainAdaptiveCS.pdf](#)