

Automated model selection in population pharmacokinetics using metaheuristic algorithms

Zhonghui Huang

zhonghui.huang.20@ucl.ac.uk

University College London

Matthew Fidler

Novartis Pharmaceuticals Corporation

Joseph F Standing

University College London

Frank Kloprogge

University College London

Research Article

Keywords: population pharmacokinetics, automated modeling, metaheuristic algorithms, R package

Posted Date: July 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10420620/v1>

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

[Read Full License](#)

Additional Declarations: Competing interest reported. Matthew Fidler is an employee of Novartis. All other authors declared no competing interests for this work.

Abstract

Automating population pharmacokinetic (PopPK) modeling can reduce manual intervention, saving time and effort. Existing automated approaches within the nonlinear mixed-effects (NLME) framework have typically followed either modular stepwise strategies, in which model components are evaluated sequentially, or global search strategies, with genetic algorithms (GA) as a representative example. Stepwise approaches may become trapped in local optima, whereas GA-based approaches can be affected by epistasis arising from interactions among model components. This study aimed to develop a fully automated, end-to-end framework in R within the nlmixr2 ecosystem that integrates automated model building, initial estimate generation, parameter estimation, and model selection. The framework implemented a conventional stepwise model-building approach together with three metaheuristic algorithms: GA and two newly introduced algorithms, ant colony optimization (ACO) and tabu search (TS), and evaluated their performance against exhaustive search results as the ground truth across 144 simulated datasets. Results showed that metaheuristic algorithms consistently outperformed the stepwise approach for identifying ground-truth models. TS and ACO achieved the highest overall model retrieval rates (91%, 90%), followed by GA (87%), whereas the stepwise algorithm achieved approximately 60%. The proposed framework provides an open-source solution for fully automated PopPK model development and is available as a CRAN package “nlmixr2auto”. Our findings highlight the potential of ACO and TS as promising optimization strategies for automated NLME model selection.

Introduction

Nonlinear mixed-effects (NLME) modeling has become the core approach for population pharmacokinetics (PopPK) analysis in drug development and regulatory submission [1, 2]. The construction and selection of NLME models typically involve decisions across multiple dimensions, including the structural, statistical, and covariate models [3]. In current practice, these decisions are largely guided by expert judgment and stepwise model development procedures [4]. This process requires researchers to possess extensive knowledge of pharmacokinetic (PK) and statistical expertise, consumes considerable human and computational resources, and may lack standardization in decision-making. Automation offers a potential solution. In this context, a central technical challenge is achieving efficient and reliable automated model selection. In theory, exhaustive search is the most straightforward strategy, as it tests all possible model combinations within a predefined candidate space to identify the global optimum. However, as model dimensionality increases, the number of candidate models grows exponentially. For example, the number of model combinations required can reach the order of millions [5, 6], rendering exhaustive search practically infeasible.

Aside from exhaustive search, existing studies on fully automated modeling can be broadly divided into two approaches. The first follows the paradigm of traditional manual modeling through modular stepwise exploration, with PharmPy [7] as a representative tool. This approach decomposes an NLME model into multiple submodules and applies exhaustive strategies within each module, thereby substantially expanding the coverage of candidate models. While maintaining the interpretability of

stepwise model building, the process remains constrained by inter-module interactions [8], whereby a choice in one module depends on how the other modules are currently specified. In other words, an optimal choice within a given module is conditional on the current specifications of the other modules, which may not themselves be globally optimal, and therefore does not guarantee global optimality of the overall model. Additionally, the order dependence in model selection should be considered, as different sequences of module execution can yield divergent final models [7].

The second line of research is based on global search and machine learning algorithms that bypass modular stepwise exploration and instead optimize directly within the model space, with the recently developed pyDarwin platform [5, 9] as a notable example. Along this line, it has been studied and applied for nearly two decades, and has also been implemented in practical case studies [6, 10–14]. The GA may face epistasis [15], a situation in which one pair of genes masks or modifies the expression of another. In the context of population pharmacokinetics, this can be understood as the presence or absence of a three-compartment model influencing the assessment of inter-individual variability (IIV) on the second peripheral volume of distribution. Such interdependencies not only reduce the efficiency of the GA search but may even render certain problems unsolvable by GA [16]. In previous applications, issues have been reported such as discrepancies between the selected model and the original design [6], a tendency to favor overly simple models [13], and the need to combine the algorithm with additional local search procedures (e.g., a “two-downhill” strategy, which performs local neighborhood search by flipping two bits in the binary representation) to ensure accuracy [17].

In addition to the GA, other metaheuristic algorithms also hold potential value in NLME modeling. For example, Ant Colony Optimization (ACO) [18], inspired by the foraging behavior of ants, can recast the search for NLME model structures as a “path selection problem”, thereby enabling a more natural expression of multiple modeling alternatives. An ACO-based approach was introduced for automated PopPK model development, demonstrating the feasibility of applying ACO to this field [19]. Similar to feature selection tasks [20], ants traverse a feature graph, with each path consisting of the nodes (features) visited and thus defining the selected subset. The search for NLME models can be framed as ants exploring a graph of candidate model components, where a path directly encodes a specific model component rather than relying on binary encoding. Another alternative method is Tabu Search (TS) [21]. Based on local search, it systematically explores neighboring solutions, using a tabu list to avoid repetition and thus prevent cycling. Compared with traditional stepwise methods, it not only exploits the advantages of local search more efficiently but also allows the acceptance of suboptimal solutions to escape local optima, thereby combining flexibility with the potential for global optimization. To date, the application of algorithms such as ACO and TS to automated model selection in PopPK remains largely unexplored.

Apart from algorithm design, several research gaps remain. First, current tools such as PharmPy and pyDarwin have largely focused on their integration with NONMEM, a proprietary modeling environment, which may constrain their broader application within the open-source ecosystem. Second, methods for evaluating algorithm performance remain inadequate: most existing studies rely on decreases in the

fitness function of the selected model or on comparisons with the original models [6, 7, 14, 22], with only one study adopting exhaustive search as a ground truth [5]. As a result, it is difficult to determine whether the algorithms approach the true optimal solution. Third, most explorations are confined to individual cases or small-scale simulation studies, lacking systematic evaluation across diverse datasets and at a larger scale, leaving the robustness and generalizability of the methods insufficiently demonstrated. In summary, an open-source tool is needed to lower the barrier to practical application. The tool should be evaluated using appropriate performance assessment methods, such as an exhaustive search as a benchmark, while the effectiveness of its underlying algorithms should be systematically validated across larger, more diverse datasets rather than demonstrated only through isolated cases.

To address these gaps, this study aimed to establish and evaluate metaheuristic algorithms for automated PopPK model development. The specific objectives were: (1) to develop an algorithmic framework, including the design of a stepwise model-building algorithm that mimics the modeler's workflow, and three metaheuristic algorithms (GA, ACO, and TS); (2) to evaluate algorithm performance using exhaustive search as the ground truth under multiple pharmacokinetic scenarios, including different administration routes, simulated pharmacokinetic properties, and sampling designs, to ensure their robustness across a wide range of modeling contexts; (3) to develop a generalized modeling pipeline and R package by establishing an automated modeling workflow and implement it as an open-source R package, with the goal of promoting objectivity, standardization, and efficiency in the development of PopPK models.

Methods

Research framework

As illustrated in Fig. 1, this study established a systematic experimental procedure to evaluate the performance of model-search algorithms in PopPK modeling. A total of 144 simulated datasets were generated by combining 48 prespecified PK models, including 24 intravenous and 24 oral models, with three sampling designs: optimal, rich, and sparse sampling. For each dataset, exhaustive search was performed over the corresponding predefined candidate model space, comprising 711 intravenous (IV) models or 1,422 oral models, and the model with the best fitness value was treated as the dataset-specific reference model. All four model-search algorithms, including the genetic algorithm, ant colony optimization, tabu search, and the stepwise model-building algorithm, shared a candidate model space and a fitness function. The best models selected by these algorithms were compared with the corresponding exhaustive-search reference models to assess reference-model retrieval rates and computational efficiency, and additional stratified analyses were conducted according to administration route and sampling design. Detailed descriptions of dataset construction, candidate model-space construction, fitness function definition, algorithm implementation, and performance evaluation are provided in the followings.

Simulated datasets

The simulated datasets were constructed following the same design framework, covering 48 PK models with variations in administration routes (oral and intravenous administration), structural models (one-, two-, and three-compartment models), residual error types (additive, proportional, and combined residual error models), and five distinct IIV groups. The model structures and parameter settings were identical to those used in our previous study [23]. In addition to the optimal sampling design used in our previous study, the present study incorporated two additional designs, namely rich and sparse sampling, as specified below:

- **Rich design (single-dose rich data):** 100 individuals with intensive sampling at 0.25, 0.50, 1, 2, 3, 4, 6, 8, 10, 12, 14, and 24 h post-dose.
- **Single-dose sparse data:** 100 individuals allocated to three staggered sampling groups, with four samples per individual at 0.25, 0.50, 1, and 2 h ($n = 33$); 3, 4, 6, and 8 h ($n = 33$); or 10, 12, 14, and 24 h ($n = 34$) post-dose.

The detailed characteristics of all simulated datasets, including the optimal-design datasets (sim001-sim048), rich-sampling design datasets (sim049-sim096), and sparse-sampling design datasets (sim097-sim144), are summarized in **Supplemental Material 1**. The simulated datasets are provided in **Supplemental Material 2**.

Search Space

The candidate model search space was constructed by combining features across multiple dimensions:

- Structural models: one-, two-, or three-compartment model.
- Elimination mechanisms: linear or Michaelis-Menten.
- IIV: defined by the presence or absence of random effects on parameters including the Michaelis-Menten constant (K_m), central volume of distribution (V_c), peripheral volume of distribution (V_p), second peripheral volume of distribution (V_{p2}), intercompartmental clearance (Q), and second intercompartmental clearance (Q_2).
- IIV correlations: indicating whether correlations among random effects were considered.
- Residual error models: additive, proportional, or combined error.

Under this design, a total of 711 intravenous models and 1422 oral models were constructed.

Fitness function

The fitness function in this study was defined as the Bayesian information criterion (BIC) with penalty terms applied when any diagnostic metric (relative standard error (RSE), shrinkage, between-subject

variability (BSV), residual error variances, or IIV correlations) violated the pre-specified acceptance criteria:

$$\begin{aligned}
 \text{Fitness} = \text{BIC} &+ \sum_{i=1}^{n_{\theta}} \left(10 \cdot 1_{\{20\% < \text{RSE}_i \leq 40\%\}} + 10000 \cdot 1_{\{\text{RSE}_i > 40\%\}} \right) \\
 &+ \sum_{j=1}^{n_{\omega}} \left(10 \cdot 1_{\{30\% < \text{Shrinkage}_j \leq 50\%\}} + 10000 \cdot 1_{\{\text{Shrinkage}_j > 50\%\}} \right) \\
 &+ \sum_{j=1}^{n_{\omega}} \left(10000 \cdot 1_{\{\text{BSV}_j < 10\%\}} \right) + \left(10 \cdot 1_{\{0 < \sigma_{\text{add}} < 0.001\}} + 10 \cdot 1_{\{0 < \sigma_{\text{prop}} < 0.001\}} \right) \\
 &+ \sum_{i < j} \left(10 \cdot 1_{\{0.05 \leq \rho_k < 0.1 \vee 0.8 < \rho_k \leq 0.95\}} \right) \\
 &+ \sum_{i=1}^{n_{\theta}} 10000 \cdot 1_{\{\theta_i > \theta_i^{\text{bound}}\}} + 10000 \cdot 1_{\{\text{covariance step failed}\}}
 \end{aligned}$$

where $1_{\{\cdot\}}$ denotes the indicator function (1 if the condition holds, 0 otherwise), RSE_i is the relative standard error of θ_i , Shrinkage_j and BSV_j denote the shrinkage and between-subject variability (CV%) of the j_{th} IIV parameter, respectively, σ_{add} and σ_{prop} are additive and proportional residual error terms; ρ_k is the k th correlation estimate between two IIV parameters; and θ_i^{bound} is the pre-specified boundary value for θ_i . This fitness function was the same as that explored in our study [23].

Stepwise algorithm

Algorithm steps. The stepwise model-building algorithm (hereafter, the stepwise algorithm) developed in this study was divided into four modules (see **Supplemental Material 3**, Fig. 1): the structural model (number of compartments and elimination mechanism), the IIV structure, the IIV correlation, and the RV model.

The baseline model was specified with IIV only on CL, assuming first-order elimination and a combined residual error model. Based on this foundation, the following optimizations were performed sequentially: first, different compartment models (one-, two-, and three-compartment) were compared to identify the best-fitting structure; second, linear elimination was evaluated against Michaelis-Menten (M-M) elimination within the candidate structural models; third, additional IIV terms were progressively introduced for other parameters and retained only if they improved the fit; fourth, model performance was compared with and without IIV correlations to determine whether correlation terms were warranted; and finally, among additive, proportional, and combined residual error models, the best-performing structure was selected.

At each stage, the optimal model was carried forward as the starting point for the subsequent step until the final optimal model was obtained. If no improvement was achieved at a given stage, that step was skipped, and the process advanced to the next phase.

Genetic algorithms

Problem construction. A GA with a chromosome-encoding scheme was employed to represent candidate PopPK models as binary strings (Fig. 2), and the full encoding map is given in **Supplemental Material 3**, Table 1. Each gene position specified a particular model feature, covering the number of compartments, elimination type, IIV, correlation among random effects, and residual error structure. Binary features with two alternatives (IIV presence and elimination type) were encoded using a single gene. Categorical features with three possible options (number of compartments and residual error) were jointly encoded by two genes, with two-bit patterns mapping to the same category.

Algorithm steps. The overall process of the genetic algorithm consists of initialization, model fitting and fitness evaluation, selection, crossover and mutation, elitism, local search, and stopping criteria. A flowchart illustrating the genetic algorithm procedure is provided in **Supplemental Material 3**, Fig. 2.

- **Initialization.** Once the encoding scheme was defined, the genetic algorithm began by generating an initial population via random sampling, with each individual representing a candidate model. The initial population was thus formed according to the predefined encoding rules.
- **Model fitting and fitness evaluation.** In each generation, once the model corresponding to an individual had been fitted, its performance was assessed using a unified, predefined fitness function.
- **Selection.** Tournament selection (**Supplemental Material 3**, Fig. 3) was adopted in this study to choose individuals as parents for the next generation. The procedure started with the first individual in the population, who was taken as the candidate. Another distinct individual was then randomly selected from the population to serve as its competitor. Their fitness values were compared, and the individual with the lower fitness value was retained as a parent. This procedure was repeated sequentially for each individual in the population until the parent pool was completed.
- **Crossover and mutation.** The selected individuals underwent crossover operations (exchange of partial gene segments) and random mutations to generate a new generation of models, as shown in (**Supplemental Material 3**, Figs. 4 and 5)
- **Local search.** A hybrid GA was employed by incorporating local search at fixed intervals during generation process. Specifically, a 1-bit local search was applied, in which the current best solution was refined by systematically flipping each bit of its chromosome in turn, thereby exploring all possible one-bit neighborhood solutions.
- **Elitism.** In this study, an elitism strategy was adopted, in which the best solution in each generation (i.e., the chromosome with the lowest fitness value) was directly preserved for the next generation.

- **Stopping criteria.** Iterations were carried out until the algorithm reached the predefined maximum number of generations.

Parameter settings. A baseline setting for the sensitivity analysis was established, consisting of a population size of 20, a crossover probability of 1, a mutation probability of 0.05, a maximum of 20 iterations, and five local search operations. The values for population size and maximum iterations follow the baseline model configuration reported by Sibieude et al. [14], while the settings for crossover and mutation probabilities are consistent with GA applications in PopPK modeling described by Ismail et al. [6]. The number of local searches was chosen with reference to the reported range of 2–15 in the two aforementioned works.

For the sensitivity analysis, each parameter was perturbed individually within a predefined range using the one-at-a-time method, in which only one parameter was altered at a time while all others were held constant. The ranges in Table 1 were derived from the parameter settings reported by Sibieude et al. [14] and Ismail et al. [6], with adjustments made to reflect the search space scale of this study. Specifically, the maximum number of iterations was set to 10–50, reflecting the empirical range of 20–50 reported in their work, while higher values such as 200 iterations were reserved for substantially larger search spaces with more than a million models [6]. Mutation probability was defined within the reported lower range of 0.01–0.05 and extended upward to 0.3 to evaluate the effect of higher levels. The number of local search operations was set to 2–10, with the initial population size set to 10.

Table 1
Baseline settings and candidate ranges of GA parameters

Parameter	Baseline	Candidate range
Population size	20	10, 20, 30, 40, 50
Crossover probability	1	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Mutation probability	0.05	0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3
Maximum iterations	20	10, 20, 30, 40, 50
Number of local searches	5	2, 3, 4, 5, 6, 7, 8, 9, 10

Ant colony optimization

Problem construction. The path-construction graph of the ACO is shown in Fig. 3. The fundamental elements of this graph comprise nodes, which represent decision points corresponding to model features; edges, which denote specific PopPK components available at each node and store pheromone information; and paths, which describe the sequence of decisions made by an ant from the starting point through all reachable nodes until the terminal node.

The construction begins at the number of compartments node, followed by a series of IIV nodes, and then proceeds to the Michaelis-Menten node to determine whether linear or nonlinear elimination is used. From the IIV on V_c node onward, the process enters a common region that all paths must traverse, including the IIV correlation node and the residual error model node. For oral administration, an additional node was included to decide whether to introduce IIV on K_a , with two possible paths (yes/no). Each complete path specifies a unique PopPK model structure.

Algorithm steps. The proposed method was based on the Simple Ant Colony Optimization (SACO) algorithm [24], and combined it with a rank-based pheromone-updating strategy. The overall procedure consists of initialization, solution construction and state transition, model fitting and fitness evaluation, pheromone updating, strategy enhancements, and stopping criteria. A flowchart illustrating the ant colony optimization procedure is provided in **Supplemental Material 3**, Fig. 6.

- **Initialization.** At the beginning of the algorithm, the pheromone levels on all edges were initialized to the same value, after which the first generation of ants was generated. In this study, the first generation consisted of two types of ants: fixed-route ants and random-route ants. The fixed-route ants were designed to explore predefined models to ensure that essential paths were adequately covered. Given that the number of ants is typically similar to the problem size [25, 26], and that the present study involves 10 model features, 6 fixed-route ants were introduced to prevent the entire population from being restricted to fixed routes. These six ants were designated to cover the combinations of structural models (one-, two-, and three-compartment) with residual error models (additive and proportional), resulting in six fixed routes. The remaining random-route ants were designed to construct their solutions by randomly selecting paths according to a uniform-probability state transition rule, thereby generating the initial candidate solution set.
- **Solution construction and state transition rule.** In each travel, every ant started from the initial node and traversed all decision nodes corresponding to model features, selecting paths according to the state transition rule. For an ant k located at node i , the probability of choosing edge j as the next step was determined by the following expression:

$$P_{ij}^k = \frac{[\tau_{ij}]^\alpha}{\sum_{l \in \Omega_i^k} [\tau_{il}]^\alpha}$$

where τ denotes the pheromone intensity on edge ij , α serves as the control parameter that determines the importance of pheromone trails, and Ω_i^k denotes the set of feasible edges available to ant k when located at node i .

- **Model fitting and fitness evaluation.** After an ant completed constructing a candidate solution by selecting its path, the model was fitted, and parameters were estimated. The performance of each solution was then quantified using the predefined fitness function. A historical ranking of the fitness function was subsequently performed, in which candidate solutions of the current generation were

ranked together with previously obtained solutions. Based on this ranking, the pheromone increment for each path was defined as follows:

$$\Delta \tau_{ij}^k = \begin{cases} \frac{1}{f(S^k)}, & (i, j) \in S^k \\ 0, & \text{otherwise} \end{cases}$$

where ij denotes an edge on the path, S^k represents the solution constructed by ant k , and $f(S^k)$ is the predefined fitness function, the value of which is determined by the historical ranking; $\Delta \tau_{ij}^k$ is the amount of pheromone deposited by ant k on edge (i, j) . Solutions with higher ranks correspond to smaller values of $f(S^k)$, thereby receiving larger pheromone increments during the update process.

- **Pheromone update.** The pheromone update process consists of two mechanisms: pheromone reinforcement and evaporation. Reinforcement refers to the weighted update of the pheromone trails on the edges traversed by all ants. When an edge ij was visited by ant k , its contribution was defined as:

$$\Delta \tau_{ij}(t) = \sum_{k=1}^m \Delta \tau_{ij}^k(t)$$

Here, $\Delta \tau_{ij}(t)$ denotes the total pheromone deposited on edge ij at iteration t , and $\Delta \tau_{ij}^k(t)$ represents the contribution from ant k . Pheromone evaporation was applied at the end of each travel to all edges and the process was expressed as:

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \Delta \tau_{ij}(t), 0 < \rho \leq 1$$

where $\tau_{ij}(t)$ represents the pheromone intensity on edge ij at iteration t , while $\tau_{ij}(t+1)$ is the updated pheromone level. The parameter ρ is the evaporation rate, ensuring that pheromone levels do not grow without bound.

- **Strategy enhancements.** The ACO algorithm in this study incorporated an elitist strategy. In each travel, an elitist strategy was introduced, whereby the best individual was preserved in the next generation, ensuring that the ant colony always retained the global best solution. An extended form of this strategy was also examined, in which the preservation mechanism was expanded from a single individual to a proportion of $p\%$ of the next-generation population, which is expressed as:

$$Ants^{t+1} = \underbrace{S^{best}, \dots, S^{best}}_{p/100 * m \text{ ants}} \cup \underbrace{\text{sampled by transition rule}}_{(1-p/100) * m \text{ ants}}$$

where $Ants^{t+1}$ denotes the set of ants in travel $t+1$. The term S^{best} represents the globally best solution found so far, and $p/100 * m$ indicates that a fraction $p\%$ of the m ants in the next travel were directly assigned as copies of S^{best} . The remaining $(1-p/100) * m$ ants were generated according to the transition

rule. In addition to the elitist strategy, this study also introduced a minimum-probability path strategy. In this approach, a lower bound $p_{\min}$ was imposed on the exploration probability of each edge, ensuring that even suboptimal or “inferior” paths retained a non-zero chance of being selected.

- **Stopping criteria.** The stopping condition of the ant colony algorithm was defined as reaching the maximum number of iterations, at which point the algorithm stopped and output the final optimal model.

Parameter settings. All key hyperparameters of the ACO algorithm, including the number of ants, maximum iterations, pheromone importance (α), evaporation coefficient (ρ), initial pheromone level (ϕ), elitism percentage, and minimum edge probability ($p_{\min}$), were included in a local sensitivity analysis.

As shown in Table 2, the number of ants was set to 15, consistent with the problem size (10–11 features, 22–24 paths) and the principle of scale comparability [25, 26], with a sensitivity range from 10 to five times the problem size, although a previous study typically used up to three times [27]. The baseline for maximum iterations was set to 10, referencing the 20–50 generations in GA and considering the faster convergence of ACO [28], with a test range of 10–50. The pheromone importance α was fixed at 1, a common empirical default [25], with a test range of 0.25–5 to encompass empirical recommendations [26]. The evaporation coefficient ρ was set to 0.2, consistent with the typical 0.1–0.5 range [27], and its test range was extended to the full theoretical domain (0–1). The initial pheromone level ϕ was set to 1, with an upper bound of 200, following the large-scale test settings of a previous study [29]. The elitism percentage p was set to 0.1, corresponding to one additional copy of the best solution among 15 ants, with a test range of 0–0.9; $p = 0$ disables the proportional mechanism while still retaining the best solution. Finally, the minimum selection probability was set to 0.1 to preserve path diversity and prevent premature convergence, with a sensitivity range of 0–0.2. The upper bound was set considering that, for a node with three paths, the theoretical maximum is (1/3), near which path selection becomes nearly uniform and impedes convergence.

Table 2
Baseline settings and candidate ranges of ACO parameters

Parameter	Baseline	Candidate range
Number of ants	15	10, 20, 30, 40, 50
Maximum iterations	10	5, 10, 15, 20, 25, 30, 35, 40, 45, 50
Pheromone importance (α)	1	0.25, 0.5, 1, 2, 3, 4, 5
Evaporation coefficient (ρ)	0.2	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
Initial pheromone level (ϕ)	1	1, 2, 5, 10, 15, 20, 25, 30, 40, 60, 80, 100, 200
Elitism percentage ($p\%$)	10	0, 10, 20, 30, 40, 50, 60, 70, 80, 90
Minimum probability ($p_{\min}$)	0.1	0.0, 0.05, 0.1, 0.2

Tabu search

Model representation. Tabu search was designed as a neighborhood search-based optimization algorithm augmented with a tabu list and aspiration criteria, enabling the joint selection of PopPK structural and statistical models (see Fig. 4). Within this framework, all models were encoded as integer vectors, with elements representing the number of compartments, elimination types, IIV on structural parameters, IIV correlations, and residual error models. The following figure provides a schematic illustration of tabu search execution.

Algorithm steps. The implementation of tabu search followed the basic framework proposed by Glover [21, 30, 31]. The algorithm consisted of the following key elements: initialization, neighborhood search with candidate list strategy, a tabu list with tabu tenure, an aspiration criterion, a move selection rule, and stopping criteria. A flowchart illustrating the tabu search procedure is provided in **Supplemental Material 3**, Fig. 7.

- **Initialization.** The search process began with an initial model serving as the starting point. By default, a two-compartment model with IIV on CL and a combined residual error model was adopted as the initial solution.
- **Neighborhood search and candidate list strategy.** Neighborhood search generated candidate solutions by locally modifying model components of the current best model. The neighborhood was defined as all solutions obtained by altering a single variable, i.e., flipping one model feature in the model encoding. A full 1-bit neighborhood search strategy was employed, enumerating and evaluating all neighboring solutions at each iteration.
- **Tabu list and tabu tenure.** Each move was defined by the modification applied to a specific feature in the model encoding. Once such a modification was executed, it was added to the tabu list together with an associated tabu tenure. Both directions of the same modification were prohibited during the tabu tenure. For example, if the algorithm moved from a two-compartment to a three-compartment model, then both the transition from two to three compartments and the reverse transition from three to two compartments were tabu.
- **Aspiration criterion.** The aspiration criterion was not applied, and tabu-restricted models were not evaluated to reduce computational burden.
- **Move selection rule.** A greedy selection rule was adopted, in which the candidate solution with the best fitness value (as determined by the previously described evaluation) was selected as the new current solution. If no feasible neighbor was identified in a given iteration, the current solution was retained, or a perturbation mechanism was triggered.
- **Perturbation mechanism.** When no improving neighbor could be found in a given iteration, the algorithm triggered a perturbation mechanism. Specifically, two variables in the model encoding were randomly flipped to generate a new candidate solution.
- **Stopping criteria.** A fixed number of iterations was specified as the stopping criterion.

Parameter settings. The maximum number of iterations was limited to 20, taking into account the high computational cost of evaluating numerous models with a 1-bit neighborhood and aligning with the commonly used search budgets of 20–50 generations in genetic algorithms given the available computational resources. The tabu tenure was set to 3. Since the experimental dataset contained only 10 features in total, and under the single-compartment condition only four features were available for selection, a tabu tenure longer than 3 could have resulted in nearly all features being marked as tabu, thereby constraining the search.

Algorithm evaluation

An exhaustive search was employed as the ground truth. Specifically, within the defined search space, all possible valid model combinations were evaluated, and the corresponding screening results were recorded. These served as the reference for assessing the performance of the newly developed heuristic algorithms. Algorithm performance was evaluated from two perspectives: retrieval accuracy and computational efficiency.

With respect to accuracy, five core metrics were defined and calculated: the structural model, IIV, residual error model, correlation between IIV parameters, and overall concordance. For each metric, retrieval rate was defined as the proportion of correctly identified components among all cases. In terms of computational efficiency, performance was quantified by actual runtime and by the number of unique models visited during execution for each case or dataset, and runtime was reported as the cumulative time under a 4-core CPU allocation.

After determining the final parameter settings, accuracy and efficiency analyses were performed. For each simulated dataset, candidate models identified by the metaheuristic algorithms were compared with exhaustive search results on a case-by-case basis, and retrieval rates were calculated for each metric. In addition, both the GA and ACO were run five times under the final parameter settings, and the results were averaged. Furthermore, the robustness of each algorithm across different routes of administration and data types was evaluated.

Software

All algorithmic analyses were performed in R [32], and model parameter estimation was carried out using the *nlmixr2* package [33], with parameter initialization facilitated by the *nlmixr2autoinit* function developed previously [34]. All model runs were executed on the UCL Myriad High Performance Computing Facility (Myriad@UCL), with each job allocated 4 CPU cores and 2 GB of memory. The automated modeling framework *nlmixr2auto* introduced in this paper is provided as an open-source R package [35].

Results

Sensitivity analysis of GA parameters

Figure 5 shows that the GA parameters, crossover probability, mutation probability, and the frequency of local search had significant effects on the retrieval rate of the full model (i.e., matching the ground truth model identified by exhaustive search). As crossover probability increased, the retrieval rate improved progressively and reached an optimal range of 0.5–0.7, where the full model stabilized around 88–89%, and the structural, IIV, correlation, and residual components achieved approximately 90–93%, 95–98%, 95–96%, and 96%, respectively. Mutation probability performed best at 0.1, with the full model reaching 91% and component-specific retrieval rates ranging from 94% to 97%, whereas increasing it to 0.2 slightly reduced full model performance to about 89%. For local search, more frequent execution (i.e., smaller step intervals of 2–3 instead of 5) consistently improved retrieval rates, with the full model increasing from 86% at five steps to 87–88% at two to three steps, and the components similarly rising to 94–97%.

Moreover, as population size and the maximum number of iterations increased, all models exhibited a consistent upward trend in retrieval rate. In particular, while larger population sizes continued to improve performance by expanding the search space, increasing the number of iterations showed clear convergence. Beyond roughly 20 iterations, the retrieval rate stabilized at a high level, with the number of unique models reaching about 80 (over 10% of the search space), and no substantial further gains were observed. From the perspective of the number of unique models, when the population size reached approximately 20, the number of unique models increased to 75, which accounted for more than 10% of the entire search space. Therefore, considering the balance between computational cost and performance, the parameters in this study were ultimately set to a population size of 20 and 20 iterations. The final parameter configuration of the genetic algorithm was as follows: a population size of 20, a crossover probability of 0.7, a mutation probability of 0.1, a maximum of 20 iterations, and local search every 3 steps.

Sensitivity analysis of ACO parameters

The parameters alpha value, evaporation rate, elitism strategy, initial pheromone level, and minimum probability were found to have significant effects on the retrieval rate of the full model (see Fig. 6). Increasing α from 1 to 2 raised the proportion of corrected recovered full models from 72% to 76%. Over the same range, the retrieval rates for the structural component, IIV, and residual error changed from 91% to 88%, 82% to 91%, and 86% to 88%, respectively, while that for the correlation remained stable at 94%. The evaporation rate exhibited a clear optimum at 0.5, with the full-model retrieval rate peaking at 79%, while the corresponding values for the four model dimensions ranged from 88% to 94%. With respect to elitism, a percentage-based mechanism ($p = 0.1$) produced a full-model retrieval rate of 76%, whereas retaining only one best solution per iteration ($p = 0$) increased the retrieval rate to 82%. Retrieval performance was optimal when the initial pheromone ϕ was set to 1 or 2, yielding full-model retrieval rates of 74% and 76%, respectively. Beyond $\phi = 2$, performance declined steadily. The minimum probability parameter showed a modest effect, with the proportion of successfully identified full models increasing from 76% at a $p_{\min}$ of 0.1 to 80% at 0.2.

All models exhibited a consistent improvement in retrieval rate with increasing maximum number of iterations and the number of ants, while also showing a trend toward convergence. When the maximum number of iterations increased from 10 to 20, the retrieval rate of the full model rose from approximately 72% to about 85%, and the number of unique models increased from around 56 to 75. However, with further increases in iterations (30, 40, 50), the retrieval rate continued to rise to roughly 94%, and the number of unique models grew to about 107, but the marginal improvement became significantly smaller. Given the search space of 711 IV models, this corresponded to approximately 15% coverage. In contrast, increasing the number of ants had a more pronounced effect on retrieval performance: when the ant population size increased from 15 to 50, the retrieval rate of the full model improved from about 72% to roughly 92%, while the number of unique models nearly doubled, increasing from around 56 to about 120. Therefore, by balancing computational cost with accuracy, the final parameter configuration of the ACO algorithm was set as follows: the α value was set to 1, the evaporation rate ρ was fixed at 0.5, the maximum number of iterations was limited to 20, and an elitist strategy was applied by retaining one optimal solution in each iteration. In addition, the initial pheromone was set to 2, the minimum selection probability to 0.2, and the ant population size to 15.

General comparative performance of algorithms

At the overall model level, TS and ACO exhibited the best performance, with retrieval rates approaching or exceeding 90% (see Fig. 7). This corresponded to approximately 130 cases (out of 144) that were consistent with those identified through exhaustive search. The GA performed slightly less well, with a retrieval rate of around 87%, whereas the stepwise algorithm showed substantially lower performance, at approximately 60%. With respect to model-specific components (i.e., the structural model, IIV, IIV correlation, and residual error), GA, ACO, and TS achieved very similar outcomes, with retrieval rates consistently ranging from 92–98%. In contrast, the stepwise algorithm performed substantially worse on these components, with retrieval rates ranging from 73% to 92%.

In Fig. 7, the average number of unique models evaluated per dataset was approximately 119 for ACO, 110 for GA, and 109 for TS, indicating that these three algorithms were of the same order of magnitude in terms of search diversity. This corresponded to coverage of about 15–17% of the search space when considering 711 candidate models, or about 7–8% when considering 1422 candidate models. In contrast, the stepwise algorithm covered on average only 1–2% of the intravenous search space and less than 1% of the oral search space. In terms of computation time, the average runtime per dataset was similar for ACO, GA, and TS, at 4–4.6 hours, whereas the stepwise algorithm required the shortest runtime (less than half an hour), owing to its evaluation of far fewer candidate models.

Scenario-specific comparative performance of algorithms

Figure 9 showed that across both optima-design and rich-data conditions, retrieval rates for the individual model components (structural parameters, IIV, IIV correlation, and residual error) were typically in the 90–100% range for all three metaheuristic algorithms (GA, ACO, and TS). Based on full-model retrieval (optima-design and rich-data settings only), IV administration produced uniformly high

performance (93–97%) across GA, ACO, and TS, whereas oral administration reduced full-model retrieval to 79–88%. For optima-design data, oral administration yielded 82–86%, whereas for rich-data it yielded 79–88%; both were lower than the corresponding IV ranges (93–96% and 96–97%, respectively).

Under sparse-data conditions, IV and oral administration differed substantially. For IV scenarios, TS achieved 100% retrieval across all model components, while GA and ACO were slightly lower overall, with full-model retrieval of 90% and 97%, respectively. By contrast, under oral administration, full-model retrieval was 88% for ACO and TS but only 72% for GA.

For the stepwise algorithm, retrieval rates were lower and more variable than those of the metaheuristic approaches, ranging from 42–83% across scenarios. Structural model identification was poorer for oral than IV data, with accuracies of 62% versus 88% under optimal-design conditions and 58% versus 75% under rich-data conditions. Under sparse data, structural identification similarly remained low at 58% under oral administration, whereas it increased to 96% under intravenous administration.

Discussion

This study, based on the nlmixr2 platform, developed an automated modeling framework that flexibly integrates search algorithms, including the stepwise algorithm and three metaheuristic algorithms. Through systematic validation on simulated datasets, using exhaustive search results as ground-truth models, the study evaluated the performance of GA, ACO, and TS compared with the stepwise algorithm for automated PopPK model selection. The results demonstrated that all three metaheuristic algorithms exhibited higher model retrieval accuracy and robustness, with overall performance markedly superior to that of the stepwise algorithm.

Overall evaluation results showed that TS and ACO achieved the highest retrieval rates, greater than 90% and slightly outperforming the GA, which achieved approximately 87%. All three metaheuristic algorithms clearly outperformed the stepwise algorithm, which lagged behind at around 60%. This indicates that metaheuristic algorithms are more effective for variable selection, and relying solely on stepwise procedures may risk missing global optimal solutions [12]. At the level of specific model components, all three metaheuristic algorithms clearly outperformed the stepwise algorithm. In particular, the stepwise algorithm achieved no greater than a 75% retrieval rate in structural model identification.

Furthermore, different levels of data richness and study design were shown to have a critical impact on algorithm performance. Under optimal-design and rich-data conditions, component-level retrieval rates for all three metaheuristic algorithms were generally at least 90%, with the only exception being structural-model retrieval in the rich-data oral scenario, which was 88%. However, under sparse-data conditions, the differences became more pronounced. TS maintained high accuracy (88–100%), followed by ACO (88–97%), whereas GA achieved 72–90%. Compared with the rich data scenario (86–96%), this corresponds to a decrease of 6–14 percentage points. This is largely attributable to the fact that, in this study design, ACO and TS could incorporate predefined starting points set to relatively

simple models. In low-information scenarios where the true model lacked IIV in its parameters, this feature enabled the algorithms to converge quickly to reasonable structures while maintaining high stability. In contrast, GA typically initializes its first generation randomly and may produce individuals with poor fitness, which can reduce the likelihood of quickly reaching an optimal solution and increase convergence time [36].

Moreover, prior research has indicated that GAs often require two-bit neighborhood search to enhance their ability [5]. However, the present implementation employed only a one-bit neighborhood search, which may have further weakened performance under sparse-data conditions. Notably, even with more complex models, the overall performance of ACO and TS was not significantly impaired, indicating strong generalizability. This suggests that in future studies, if domain knowledge or prior information can be incorporated to more rationally define initial models, ACO and TS with their capacity for customized initialization will exhibit even greater advantages.

This study also has certain limitations. First, the evaluation was based on simulated datasets generated from predefined parameter sets. While these are representative, real clinical data are typically more complex. Second, the study primarily focused on first-order absorption. Future work should extend to more complex pharmacokinetic scenarios, such as nonlinear absorption kinetics. Third, computational considerations remain important. In this study, GA, ACO, and TS were evaluated over 8 to 15% of the total model search space. This implies a potential risk that the observed high accuracy may be due, at least in part, to extensive testing rather than the intrinsic superiority of the algorithms. To maintain performance in larger model spaces, future research should focus on enhancing the algorithms' search strategies and ensuring robust model selection without relying solely on testing many candidate models. For instance, introducing two-bit neighborhood search and niching strategies into GA to enhance global search, exploring the max-min ant system [37] or pheromone reinitialization mechanisms in ACO to prevent premature convergence, and adopting multi-start strategies in TS to further strengthen robustness and search diversity.

In summary, this study demonstrates that metaheuristic algorithms, particularly TS and ACO, are clearly superior to the traditional stepwise algorithm in automated PopPK model selection. They achieve higher accuracy, greater robustness under sparse data conditions, and broader coverage of the model search space. These findings provide strong support for integrating metaheuristic algorithms into next generation pharmacometric toolchains and highlight directions for future algorithmic refinement and clinical validation.

Declarations

Competing Interests

Matthew Fidler is an employee of Novartis. All other authors declared no competing interests for this work.

Author Contribution

ZHH, FK and JS: design and conceptualization; ZHH: algorithm programming and data analysis; ZHH and MF: data interpretation and methodological best practices; ZHH: first draft of manuscript; All authors contributed to writing, editing and reviewing the manuscript.

Acknowledgement

F.K. is recipient of a Sir Henry Dale Fellowship jointly funded by the Wellcome Trust and the Royal Society (Grant Number 220587/Z/20/Z).

Data Availability

Simulated datasets are provided within the supplementary material.

References

1. Committee for Medicinal Products for Human Use (CHMP) (2007) Guideline on Reporting the Results of Population Pharmacokinetic Analyses. London, UK
2. U.S. Food and Drug Administration (2022) Population Pharmacokinetics: Guidance for Industry. Silver Spring, MD
3. Bonate PL (2011) Nonlinear Mixed Effects Models: Theory. In: Pharmacokinetic-Pharmacodynamic Modeling and Simulation, 2nd ed. Springer US, Boston, MA, pp 233–301
4. Byon W, Smith M, Chan P, et al (2013) Establishing Best Practices and Guidance in Population Modeling: An Experience With an Internal Population Pharmacokinetic Analysis Guidance. CPT: Pharmacometrics & Systems Pharmacology 2:51. <https://doi.org/10.1038/psp.2013.26>
5. Li X, Sale M, Nieforth K, et al (2024) pyDarwin machine learning algorithms application and comparison in nonlinear mixed-effect model selection and optimization. J Pharmacokinet Pharmacodyn 51:785–796. <https://doi.org/10.1007/s10928-024-09932-9>
6. Ismail M, Sale M, Yu Y, et al (2022) Development of a genetic algorithm and NONMEM workbench for automating and improving population pharmacokinetic/pharmacodynamic model selection. J Pharmacokinet Pharmacodyn 49:243–256. <https://doi.org/10.1007/s10928-021-09782-9>
7. Chen X, Nordgren R, Belin S, et al (2024) A fully automatic tool for development of population pharmacokinetic models. CPT: Pharmacometrics & Systems Pharmacology 13:1784–1797. <https://doi.org/10.1002/psp4.13222>
8. Wade JR, Beal SL, Sambol NC (1994) Interaction between structural, statistical, and covariate models in population pharmacokinetic analysis. Journal of Pharmacokinetics and Biopharmaceutics 22:165–177. <https://doi.org/10.1007/BF02353542>

9. Li X, Sale M, Nieforth K, et al (2024) pyDarwin: A Machine Learning Enhanced Automated Nonlinear Mixed-Effect Model Selection Toolbox. *Clinical Pharmacology & Therapeutics* 115:758–773. <https://doi.org/10.1002/cpt.3114>
10. Bies RR, Muldoon MF, Pollock BG, et al (2006) A Genetic Algorithm-Based, Hybrid Machine Learning Approach to Model Selection. *J Pharmacokinet Pharmacodyn* 33:195–221. <https://doi.org/10.1007/s10928-006-9004-6>
11. Chaturvedula A, Sale ME, Lee H (2014) Genetic algorithm guided population pharmacokinetic model development for simvastatin, concurrently or non-concurrently co-administered with amlodipine. *The Journal of Clinical Pharmacology* 54:141–149. <https://doi.org/10.1002/jcph.176>
12. Sale M, Sherer EA (2015) A genetic algorithm based global search strategy for population pharmacokinetic/pharmacodynamic model selection. *British Journal of Clinical Pharmacology* 79:28–39. <https://doi.org/10.1111/bcp.12179>
13. Sherer EA, Sale ME, Pollock BG, et al (2012) Application of a single-objective, hybrid genetic algorithm approach to pharmacokinetic model building. *J Pharmacokinet Pharmacodyn* 39:393–414. <https://doi.org/10.1007/s10928-012-9258-0>
14. Sibieude E, Khandelwal A, Girard P, et al (2022) Population pharmacokinetic model selection assisted by machine learning. *J Pharmacokinet Pharmacodyn* 49:257–270. <https://doi.org/10.1007/s10928-021-09793-6>
15. Davidor Y (1990) Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems* 4:369–383
16. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. *Multimed Tools Appl* 80:8091–8126. <https://doi.org/10.1007/s11042-020-10139-6>
17. Li X, Sale M, Nieforth K, et al (2024) pyDarwin: A Machine Learning Enhanced Automated Nonlinear Mixed-Effect Model Selection Toolbox. *Clinical Pharmacology & Therapeutics* 115:758–773. <https://doi.org/10.1002/cpt.3114>
18. Dorigo M, Birattari M, Stutzle T (2007) Ant colony optimization. *IEEE computational intelligence magazine* 1:28–39
19. Huang Z, Standing JF, Klopogge F (2023) Development and Exploration of Exhaustive, Stepwise and Heuristic Algorithms for Automated Population Pharmacokinetic Modelling. In: Abstracts of the 31st Population Approach Group in Europe (PAGE) Meeting. A Coruña, Spain
20. Sivagaminathan RK, Ramakrishnan S (2007) A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert Systems with Applications* 33:49–60. <https://doi.org/10.1016/j.eswa.2006.04.010>
21. Glover F (1990) Tabu Search: A Tutorial. *Interfaces* 20:74–94. <https://doi.org/10.1287/inte.20.4.74>
22. Duvnjak Z, Schaedeli Stark F, Cosson V, et al (2024) Simulation-based evaluation of the PharmPy Automatic Model Development tool for population pharmacokinetic modeling in early clinical drug development. *CPT: Pharmacometrics & Systems Pharmacology* 13:1707–1721. <https://doi.org/10.1002/psp4.13213>

23. Huang Z, Fidler M, Standing JF, Kloprogge F (2026) Fitness Functions for Population Pharmacokinetic Modeling. *Journal of Pharmacokinetics and Pharmacodynamics*
24. Dorigo M, Stützle T (2001) An experimental study of the simple ant colony optimization algorithm. In: 2001 WSES International Conference on Evolutionary Computation (EC'01). pp 253–258
25. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26:29–41. <https://doi.org/10.1109/3477.484436>
26. Prakasam A, Savarimuthu N (2016) Metaheuristic algorithms and probabilistic behaviour: a comprehensive analysis of Ant Colony Optimization and its variants. *Artif Intell Rev* 45:97–130. <https://doi.org/10.1007/s10462-015-9441-y>
27. Vasko FJ, Bobeck JD, Governale MA, et al (2011) A statistical analysis of parameter values for the rank-based ant colony optimization algorithm for the traveling salesperson problem. *Journal of the Operational Research Society* 62:1169–1176. <https://doi.org/10.1057/jors.2010.74>
28. Sariff NB, Buniyamin N (2010) Genetic algorithm versus ant colony optimization algorithm. *Universiti Teknologi MARA* 132:
29. Zhang X, Chen X, He Z (2010) An ACO-based algorithm for parameter optimization of support vector machines. *Expert Systems with Applications* 37:6618–6628. <https://doi.org/10.1016/j.eswa.2010.03.067>
30. Glover F, Taillard E, Taillard E (1993) A user's guide to tabu search. *Ann Oper Res* 41:1–28. <https://doi.org/10.1007/BF02078647>
31. Glover F (1989) Tabu Search—Part I. *ORSA Journal on Computing*. <https://doi.org/10.1287/ijoc.1.3.190>
32. R Core Team (2026) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria
33. Fidler M, Wilkins JJ, Hooijmaijers R, et al (2019) Nonlinear Mixed-Effects Model Development and Simulation Using nlmixr and Related R Open-Source Packages. *CPT: Pharmacometrics & Systems Pharmacology* 8:621–633. <https://doi.org/10.1002/psp4.12445>
34. Huang Z, Fidler M, Lan M, et al (2025) An automated pipeline to generate initial estimates for population Pharmacokinetic base models. *J Pharmacokinet Pharmacodyn* 52:60. <https://doi.org/10.1007/s10928-025-10000-z>
35. Huang Z, Standing J, Fidler M, Kloprogge F (2026) nlmixr2auto: Automated Population Pharmacokinetic Modeling
36. An Improved Genetic Algorithm with a New Initialization Mechanism Based on Regression Techniques. <https://www.mdpi.com/2078-2489/9/7/167>. Accessed 19 Sept 2025
37. Stützle T, Hoos HH (2000) MAX-MIN Ant System. *Future Generation Computer Systems* 16:889–914. [https://doi.org/10.1016/S0167-739X\(00\)00043-1](https://doi.org/10.1016/S0167-739X(00)00043-1)

Figures

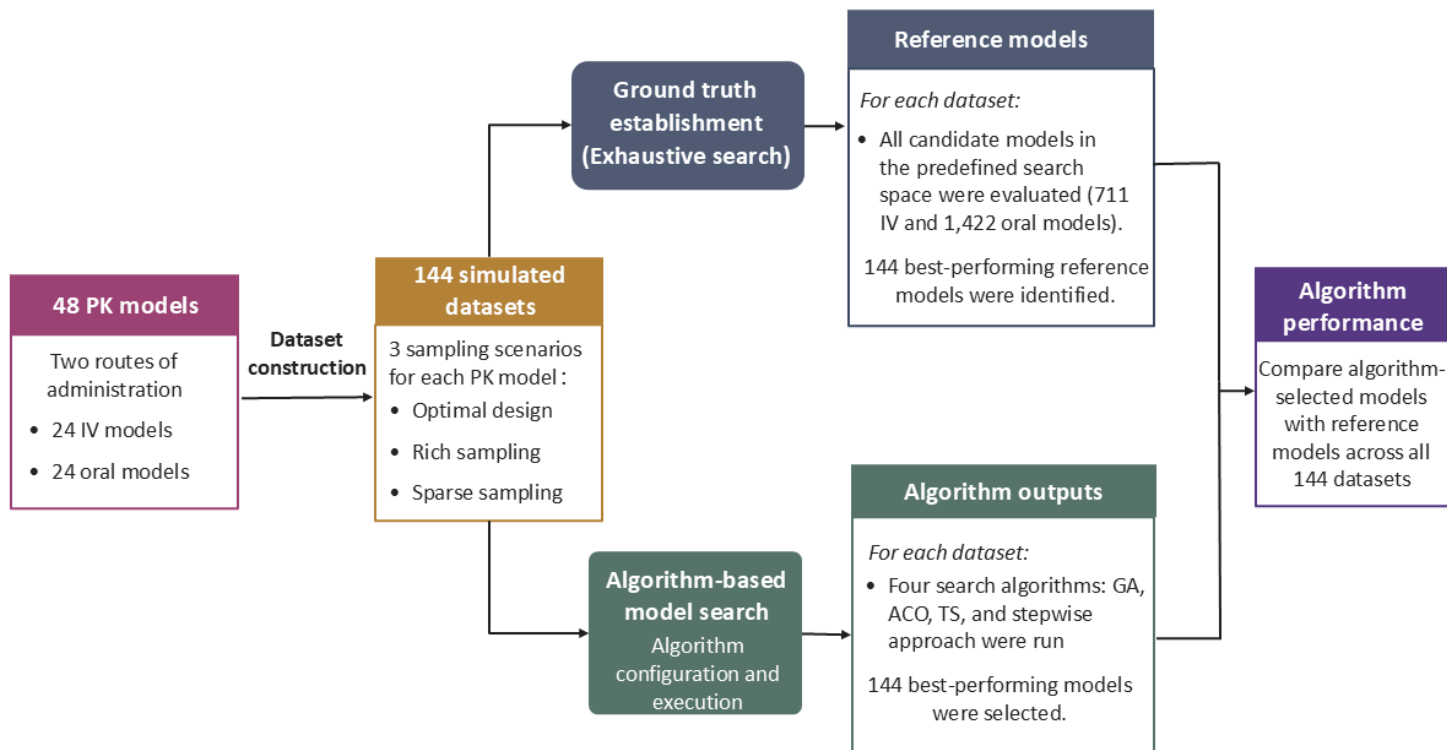


Figure 1

Study workflow for evaluating search algorithms against reference models using simulated pharmacokinetic datasets

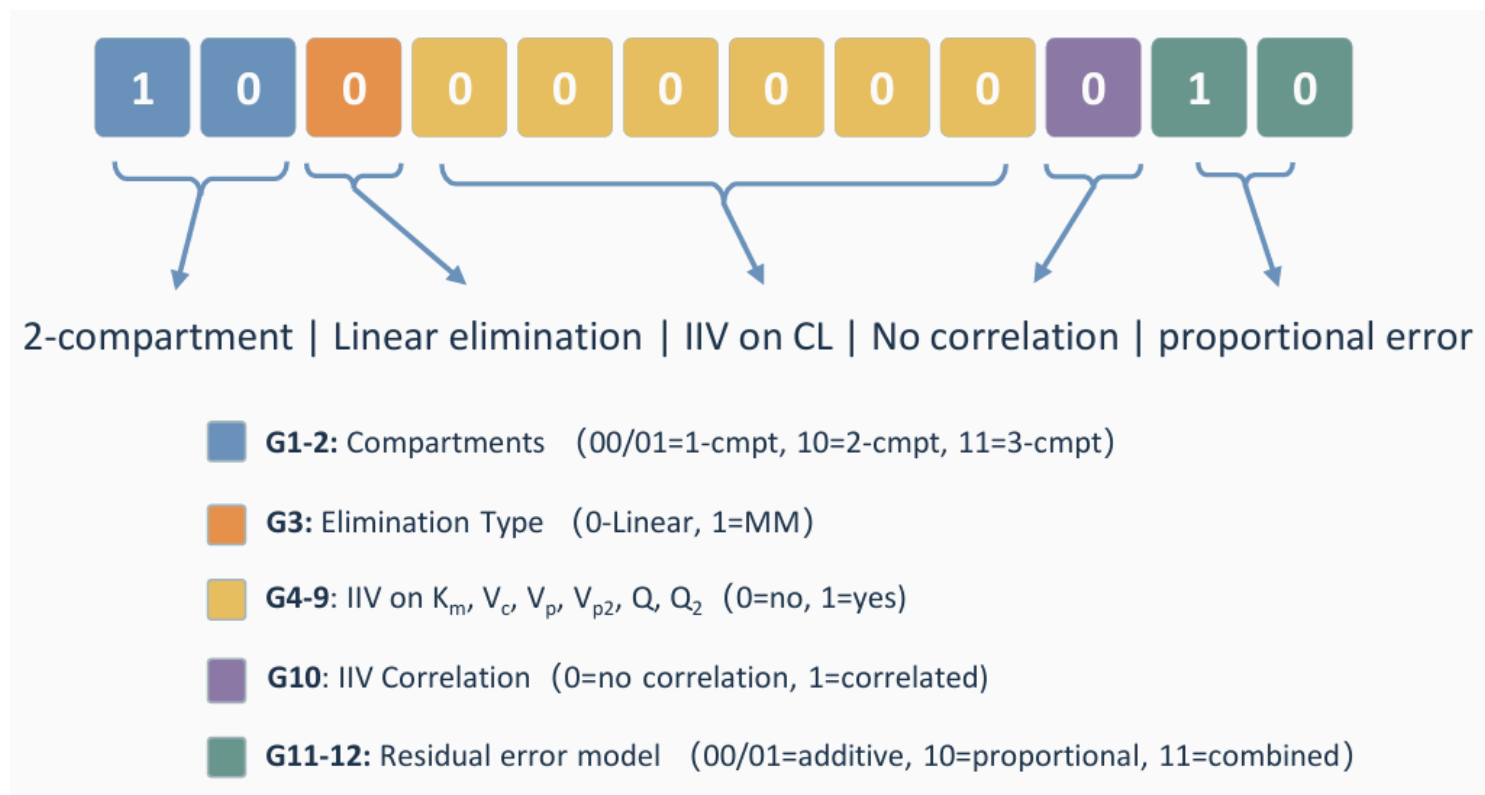


Figure 2

Schematic representation of a GA chromosome encoding for a PopPK model

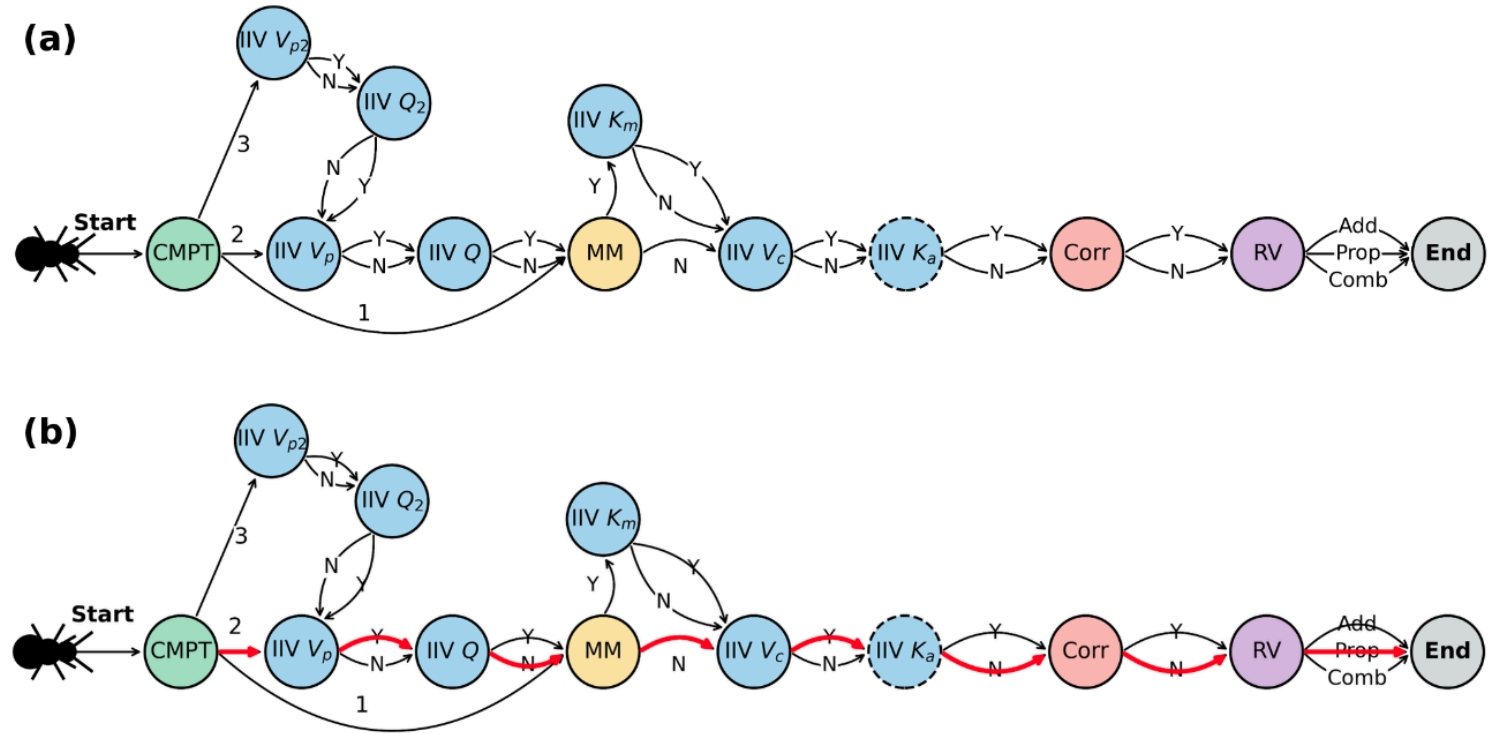


Figure 3

(a) Construction graph used in the ACO for PopPK model selection. The graph starts at the number of compartments [CMPT] node, where one-, two-, or three-compartment structures are selected. Subsequent nodes determine the inclusion of IIV terms on V_p , V_{p2} , Q_2 , K_m , V_c , and K_a (blue circles). The MM node (yellow) specifies whether nonlinear Michaelis-Menten kinetics is applied. After merging, all paths pass through the IIV V_c node, the IIV correlations (Corr) node (red), and the RV node (purple) to select among additive, proportional, or combined residual error models. Each complete path uniquely defines a PopPK model. (b) Example of model building in the ACO graph. Black edges show all available options and red edges highlight the final path selected by the ant. In this example, the chosen path corresponds to a two-compartment model with linear elimination, IIV on V_c and V_p , IIV correlations excluded, and a proportional residual error model

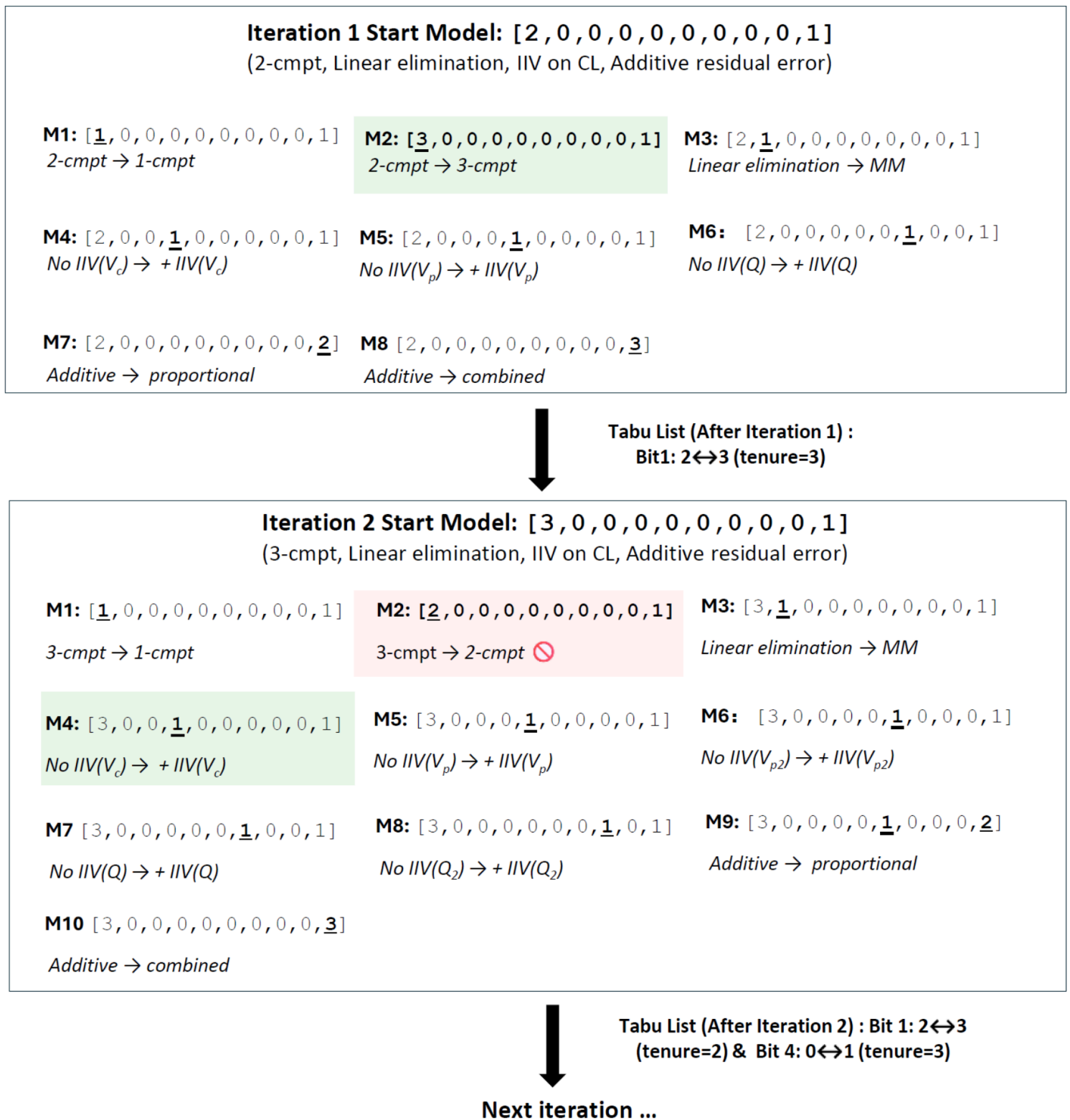


Figure 4

Tabu search example with a one-bit-flip neighborhood. One-step (1-bit) neighboring models are evaluated (gray), and the best admissible neighbor is selected (green). Candidates restricted by the tabu list are shown in red

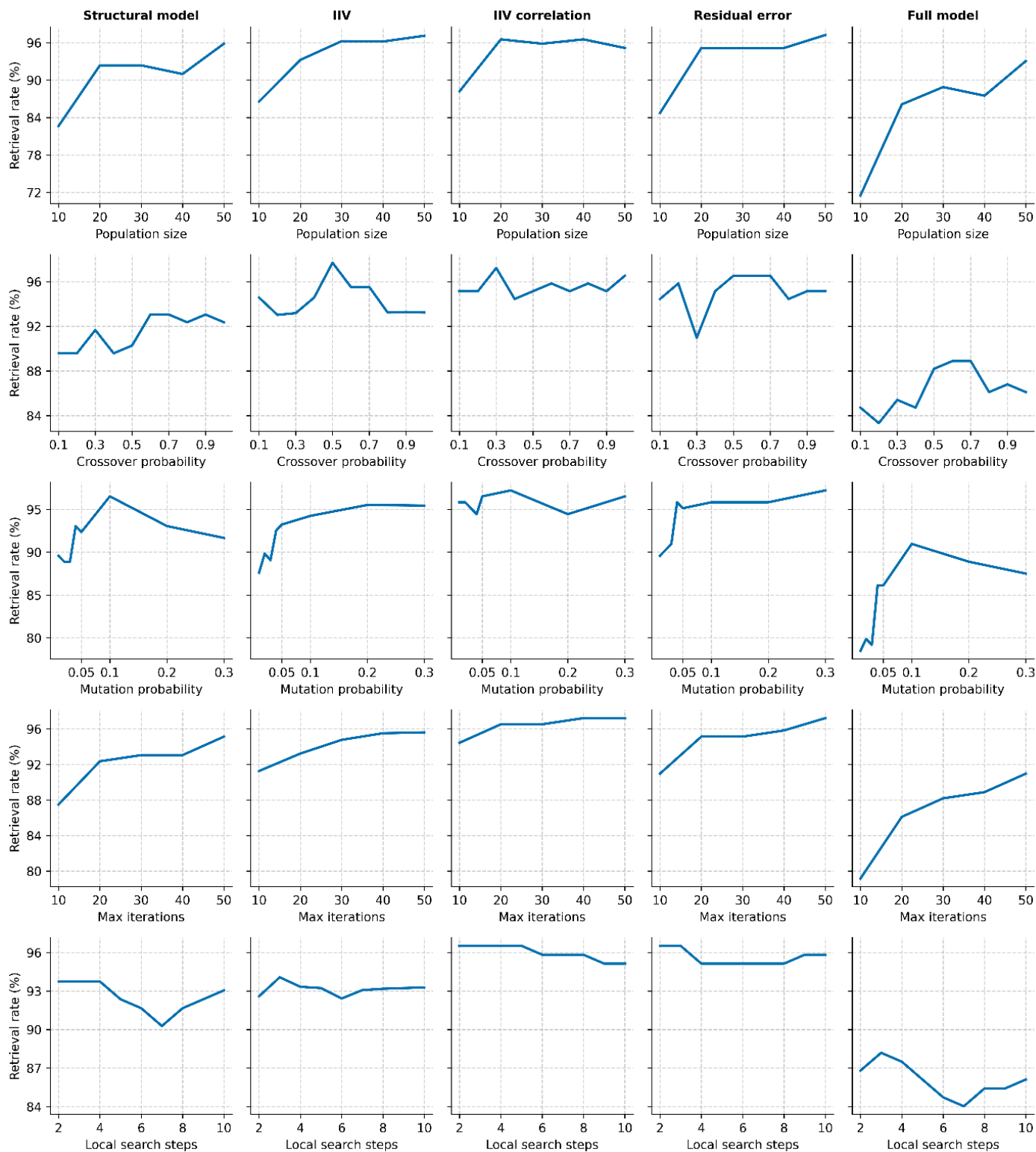


Figure 5

Effect of GA parameters on model selection performance for PopPK models. Retrieval rate was assessed under varying (a) population size, (b) crossover probability, (c) mutation probability, (d) maximum iterations, and (e) local search steps. Each panel reports results separately for the structural model, IIV, IIV correlation, residual error, and the full model, illustrating how parameter choices affect retrieval performance across model components

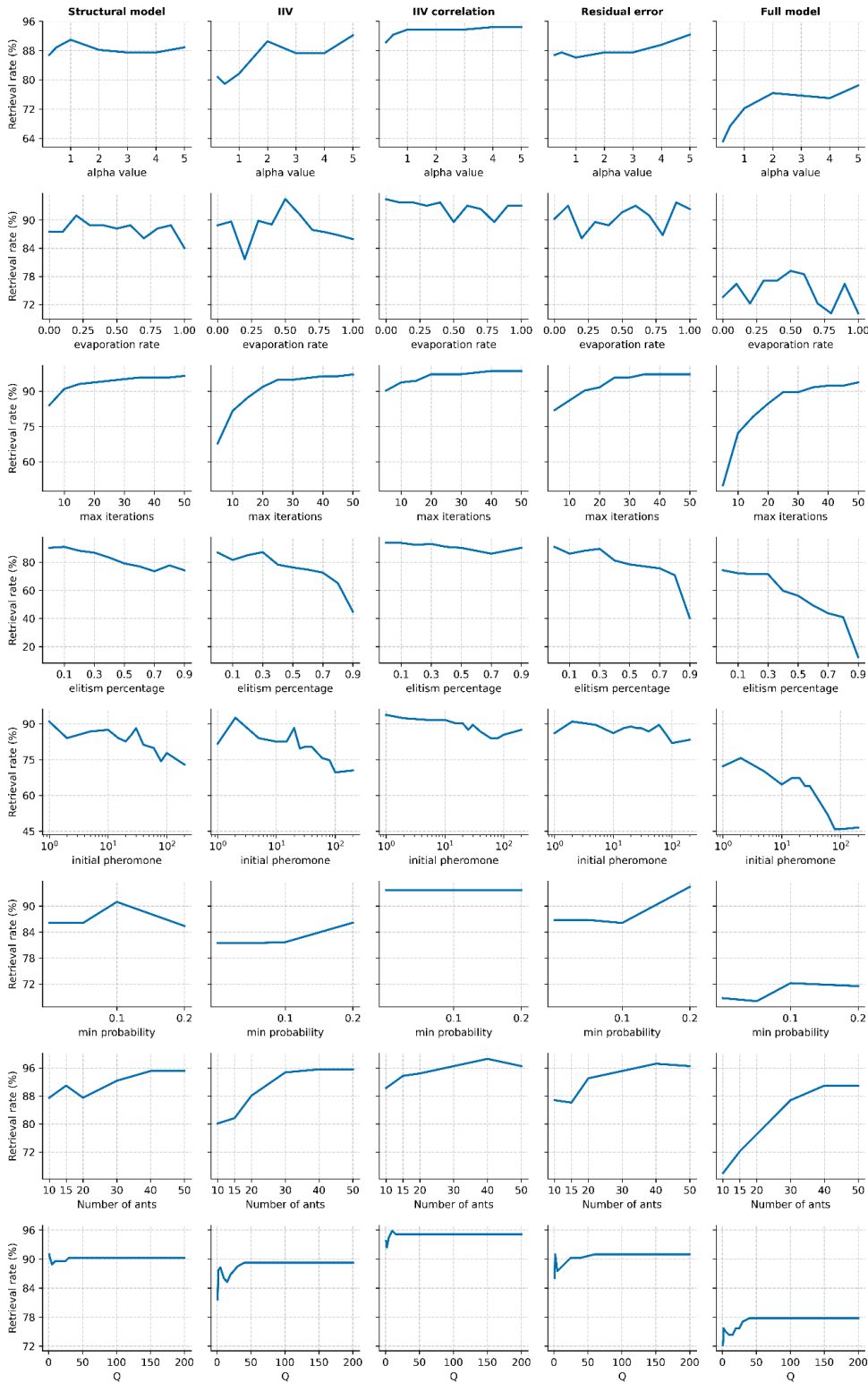


Figure 6

Effect of ACO parameters on model selection performance for PopPK models. Retrieval rates were assessed across varying (a) pheromone importance (α), (b) evaporation rate (ρ), (c) maximum number of iterations, (d) elitism proportion (p), (e) initial pheromone level (ϕ), (f) minimum edge-selection probability ($p_{\min}$), (g) number of ants, and (h) pheromone-update parameter (Q). Columns show the

retrieval rates for the structural model, inter-IIV structure, IIV correlation, residual error model, and full model

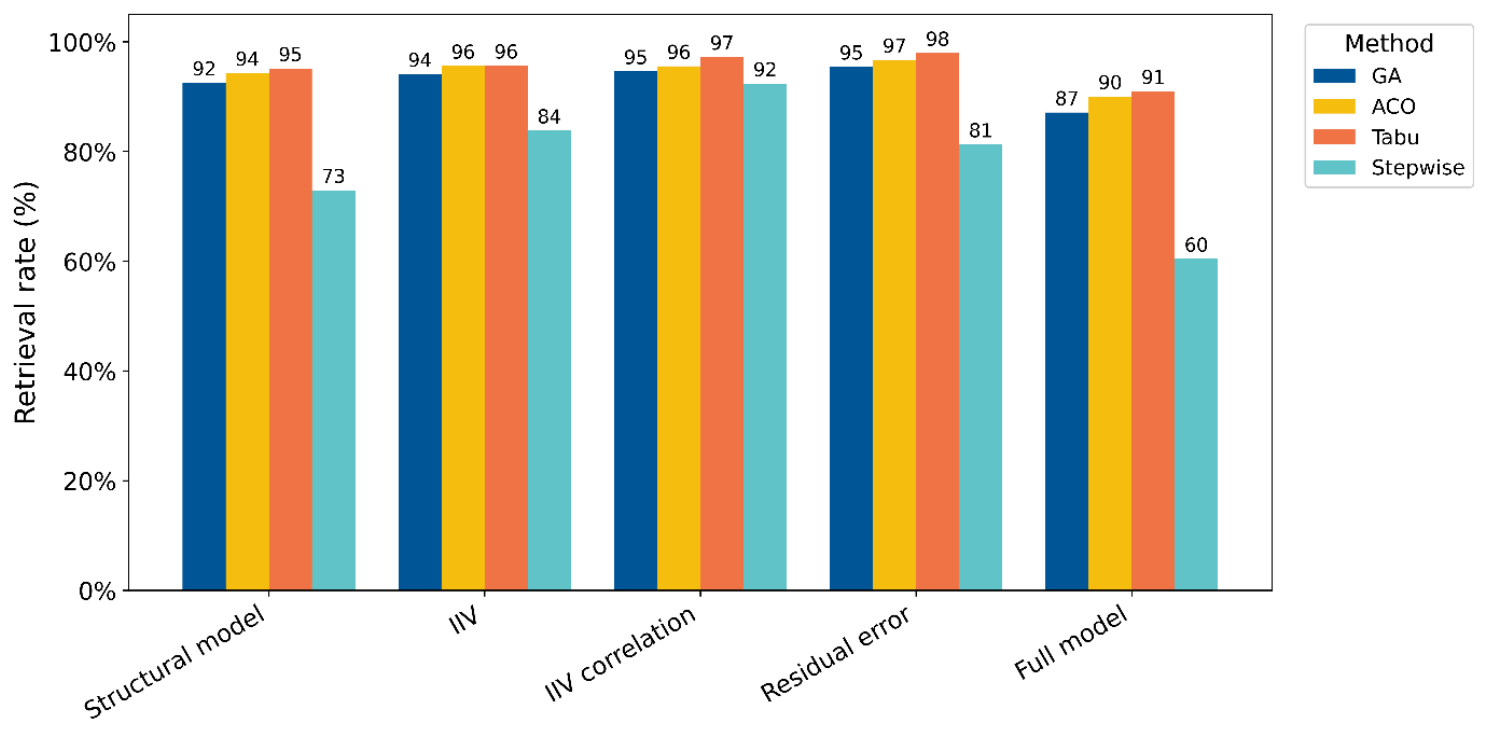


Figure 7

Comparison of model selection performance using exhaustive search as ground truth across four algorithms (GA, ACO, TS, and Stepwise) on five evaluation metrics: Structural model, IIV, IIV correlation, residual error, and overall model (“Full model”). Bars represent retrieval rates, where ACO and GA values correspond to the mean across five independent runs, while TS and stepwise algorithms are shown from single-run results. Values are displayed above each bar. Color coding: GA (dark blue), ACO (yellow), TS (orange), and stepwise (cyan) algorithms

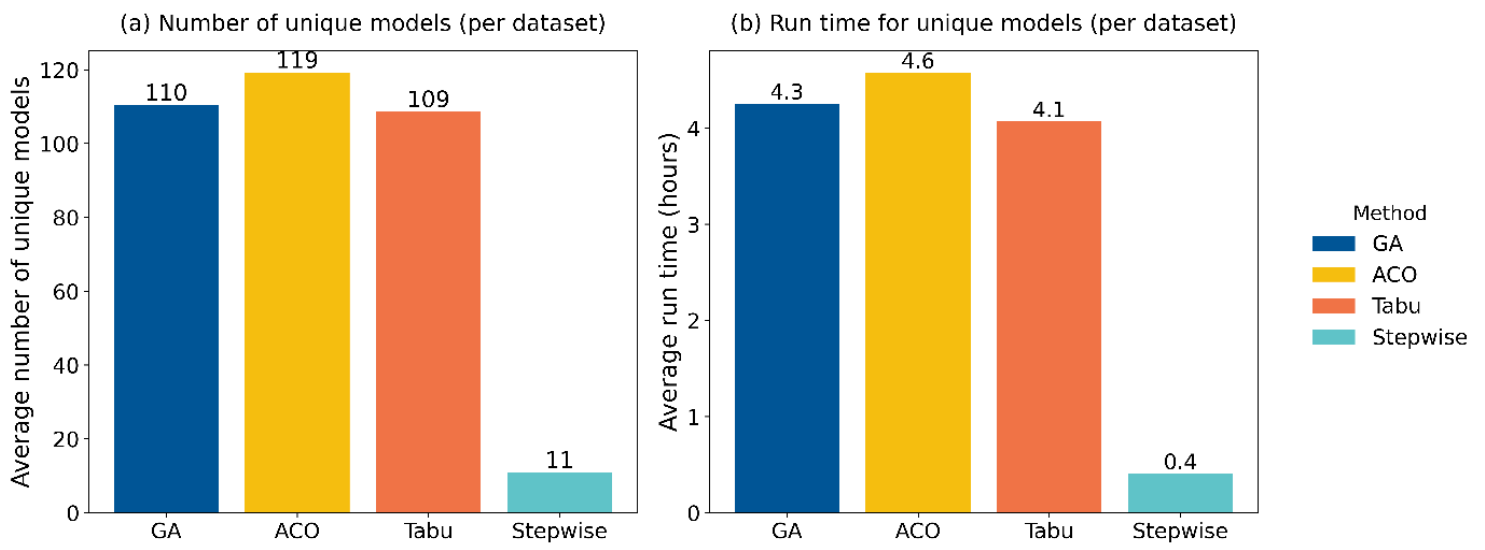


Figure 8

Comparison of algorithm performance in terms of (a) the average number of unique models and (b) the average run time for unique models across 144 datasets. Bars represent the mean values per dataset, with ACO and GA results averaging over five independent runs, while TS and Stepwise are based on single runs. Color coding: GA (dark blue), ACO (yellow), TS (orange), and Stepwise (cyan). The reported runtime refers to the cumulative time on a 4-core CPU cluster.

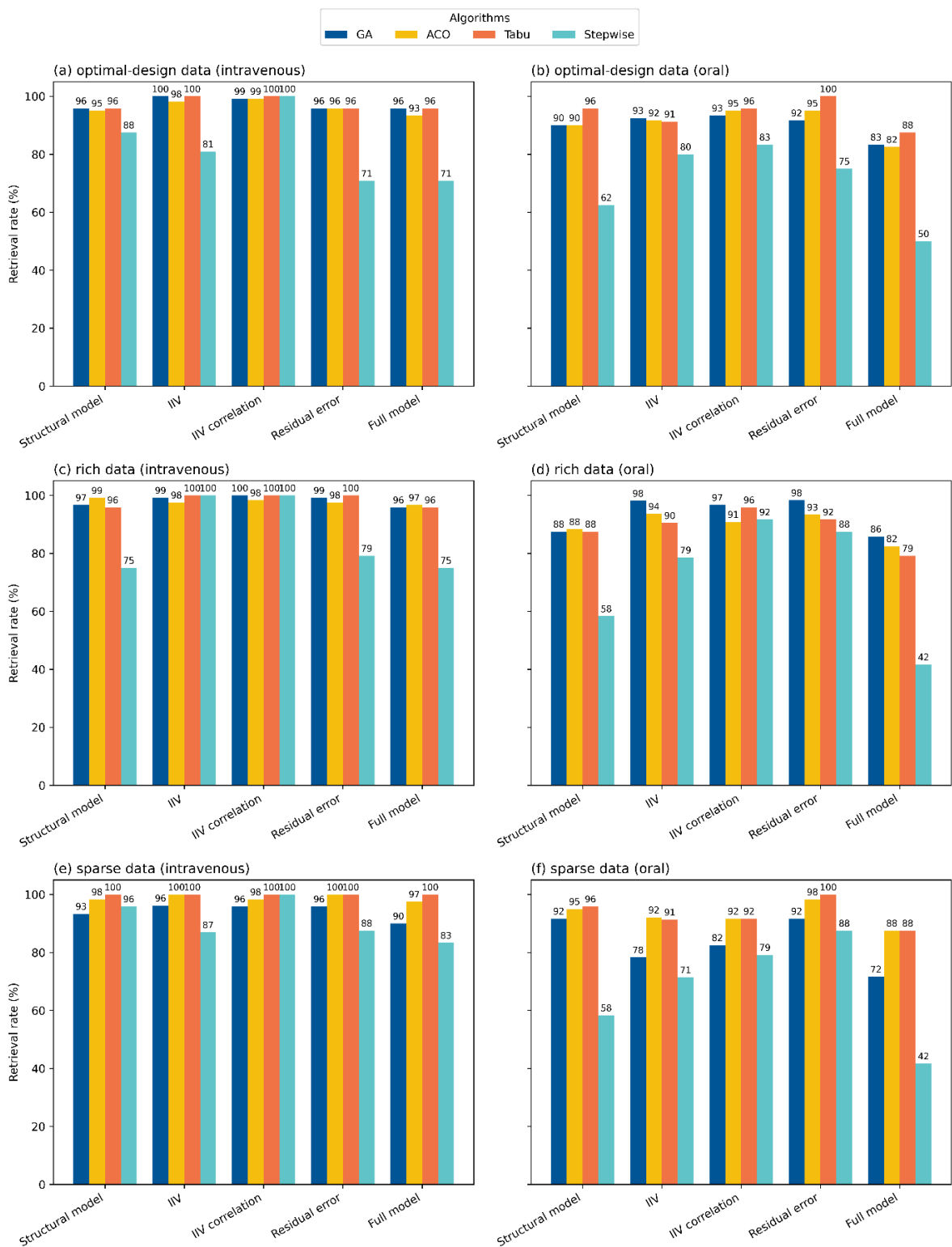


Figure 9

Comparative performance of GA, ACO, TS, and stepwise algorithms across data conditions and administration routes: (a) optimal-design, intravenous; (b) optimal-design, oral; (c) rich data, intravenous; (d) rich data, oral; (e) sparse data, intravenous; (f) sparse data, oral. Colors represent different algorithms: dark blue = GA, yellow = ACO, orange = TS, cyan = Stepwise. Retrieval rates are shown for structural models, IIV, IIV correlations, residual error models, and full models

Supplementary Files

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

- [ESM1.csv](#)
- [ESM2.csv](#)
- [ESM3.docx](#)