Journal of Applied Automation Technologies

Research Paper

Application of Genetic Algorithms in Hyperparameter Tuning of Large-Scale Machine Learning Models

Kamila Jaworska[1, *], Violetta Mikołajczyk[1] and Zofia Kukla[1]

[1] Faculty of Electrical Engineering, Automatics and Computer Science, Kielce University of Technology, Kielce, 25-314, Poland

*Corresponding author: kamila.j@tu.kielce.pl

Abstract. Hyperparameter tuning is a system problem that requires multiple accelerators to be used for an extended period to perform single model evaluation, and the search space includes continuous, discrete, and conditional choices. This paper presents a budget-aware genetic algorithm for large-scale machine learning models and integrates typed chromosome repair, asynchronous multi-fidelity evaluation, uncertainty-adjusted fitness, adaptive mutation, and stagnation-triggered diversity recovery. Random search, Bayesian optimization, Hyperband and a traditional genetic algorithm were used as baselines to compare the results obtained by gradient-boosted trees, a transformer text classifier and a graph neural network under the same compute budget. The mean normalised validation quality of the three workloads in the proposed method reached 0.913, compared with 0.889 for Bayesian optimization and 0.881 for Hyperband. Reduced accelerator-hour consumption by 31.6% compared with the conventional genetic algorithm and shortened wall-clock search time by 27.4% through asynchronous scheduling. The probability of an infeasible configuration after typed repair dropped to 1.7%, and the median quality loss caused by low-fidelity screening was still less than 0.6 percentage points. Based on ablation results, multi-fidelity promotion achieved the largest cost reduction, and adaptive diversity control prevented early convergence in conditional subspaces. Therefore, under the assumption that evolutionary operators, fidelity allocation and cluster scheduling are designed as a single optimization system, genetic search can still be applied to optimize expensive-model tuning.

Keywords: *Genetic Algorithms, Hyperparameter Tuning, Large-Scale Machine Learning, Multi-Fidelity Optimization, Asynchronous Evaluation, Resource Allocation, Distributed Computing*

Received on 04 October 2023, Accepted on 18 March 2024, Published on 28 March 2024

Introduction

The prediction behaviour of the current generation of machine learning systems is influenced by an increasing number of interacting hyperparameters, such as optimiser parameters, network width, regularization strength, data sampling rules and hardware-level batch controls. Manual adjustment is unreliable in the case of conditional choices and for full-trial distributed training. Therefore, although it is more convenient, we need to use automatic tuning for model construction [1]. Random and grid procedures are still commonly used because they are transparent, but their sampling efficiency decreases in high-dimensional mixed spaces [2]. Bayesian optimization is good for sequential decision-making, but its surrogate may be expensive or inaccurate when the observations are noisy and the variables are highly dependent [3]. Bandit methods reduce cost by early stopping, but the early learning curves do not always maintain the final ranking of large models [4]. Population-based methods have parallelism and can keep many promising regions at the same time [5]. The use of them has expanded with scalable training platforms [6]. However, at present, energy demand and accelerator scarcity have raised the cost of tuning to the level of a first-order research issue [7]. Reproducibility is also lacking in the reported protocols because scheduling, random seeds and fidelity decisions have not been disclosed [8].

Genetic algorithms are suitable for this problem because they do not require a differentiable objective function and can use real-valued, integer, categorical and inactive genes. Early evolutionary tuning studies showed good

global exploration [9]. Subsequent work introduced self-adaptive mutation and a rich population model [10]. Distributed evolutionary execution also showed a large amount of trial-level parallelism [11]. However, a typical genetic algorithm still assumes that the fitness values are comparable, complete, and obtained at roughly uniform cost. The above assumptions are not met when training jobs are asynchronous, low-fidelity estimates have unequal uncertainties, or memory limits are reached due to specific configurations [12]. A naive implementation will spend most of its budget evaluating weak offspring, repeatedly sampling infeasible regions, or over-selecting noisy low-cost trials [13]. Multi-fidelity allocation can solve the first problem [14]. Constraint-aware encoding can address the second [15]. However, the joint behaviour of them in an asynchronous evolutionary loop is not well understood for heterogeneous large-scale workloads.

This paper introduces genetic search as an integrated optimization and scheduling system rather than an independent selection operator. The proposed budget-aware genetic algorithm has typed conditional chromosomes, deterministic repair, uncertainty-adjusted multi-fidelity fitness, adaptive operator rates, and event-driven worker assignment. A promotion rule reserves expensive assessments for high-performing candidates who are also relatively certain, and a diversity controller can address a lack of candidates without discarding accumulated data. The three model families and their reports of quality, compute, wall-clock time, feasibility, convergence and robustness under matched budgets are shown in the table. The purpose of this Design is to provide more practical support for evolutionary search, rather than enabling free expansion of the population.

Related Work

Hyperparameter Optimization for Expensive Models

Hyperparameter optimization has developed in several ways. Model-free sampling is still a practical control method because it does not assume parallelism and is relatively simple, but it lacks inter-trial information transfer [16]. Sequential model-based optimization uses the observed configurations to build a response surface and selects new points according to an acquisition rule [17]. This way works well in small spaces; kernel or density estimators may not handle conditional architecture variables and a large number of accumulated observations [18]. Learning-curve extrapolation and bandit schedulers allocate additional epochs only to promising trials to achieve significant savings if the early validation behaviour is informative [19]. Population-based training combines weight learning and online mutation of hyperparameters, but its stateful exploitation mechanism differs from tuning independent final configurations [20].

The Engineering Boundary of these families is becoming more permeable. Bayesian procedures use parallel batches, evolutionary procedures employ learned surrogates, and bandit procedures embed population selection. The method needs to deal with the delay in observation, different costs, and invalid configurations reasonably. Synchronous batches in a large cluster are idle at certain times when a single trial falls behind. Aggressive early stopping reduces the volume of data used, but it may be too early and thus result in a low-quality model. Therefore, the practical optimizer should separate resource allocation from the raw observed score and retain enough evidence to compare candidates evaluated at different fidelities.

Evolutionary and Distributed Search

Evolutionary hyperparameter search is a configuration represented as a chromosome, and it modifies a population through selection, crossover and mutation. Real-coded operators are suitable for continuous schedules, and categorical and integer genes require type-preserving variation [21]. Multi-objective evolutionary algorithms have also been employed to trade off accuracy for lower latency, memory or energy [22]. Diversity preservation is required to prevent the population from being reduced to a small group near a noisy incumbent due to strong early selection [23]. Surrogate-assisted evolutionary algorithms reduce the number of exact evaluations, but an inaccurate surrogate can suppress unusual configurations before they are observed [24]. Distributed islands and asynchronous models improve utilisation by enabling some populations to evolve without a global barrier [25].

Three Gaps Motivate the Present Design. First, the mixed and conditional model spaces require explicit repair semantics; simple clipping cannot restore dependencies such as the attention width being divisible by the

number of heads. Second, the multi-fidelity scores are not assumed to be equally reliable observations. Third, asynchronous completion changes the selection pressure because the cheap candidates return more frequently. The above problems have been addressed by introducing a unified chromosome structure, calibrated uncertainty penalties, age-aware selection and budget accounting based on actual accelerator operation time.

Budget-Aware Genetic Optimization Framework

Problem Formulation and Conditional Encoding

Let a tuning task be defined over a conditional space containing continuous learning rates, integer dimensions, categorical optimizers, and activation flags for model-specific options. A candidate chromosome stores both gene values and a binary activity mask. The mask prevents an inactive parameter from affecting distance, crossover, or diversity. The chromosome is represented as

$$\boldsymbol{x} = (x_1, \dots, x_d; a_1, \dots, a_d), a_j \in \{0,1\}. \quad \text{Eq.(1)}$$

Each candidate is mapped through a deterministic repair operator before training. Repair clips bounded values, rounds integers, resolves categorical codes, enforces divisibility constraints, and reduces batch size if a memory estimator predicts an out-of-memory event. The repaired chromosome is

$$\tilde{\boldsymbol{x}} = R\left(\boldsymbol{x}; \mathcal{C}_{hard}, \widetilde{M}\right), \tilde{\boldsymbol{x}} \in \mathcal{X}_{feasible} \quad \text{Eq.(2)}$$

These rules make feasibility an explicit property of the search representation rather than an accidental consequence of failed jobs.

The optimization target combines predictive quality with measured computation. Predictive quality is normalized within each workload so that metrics with different directions, such as error and F1 score, share a consistent maximization orientation:

$$q(\boldsymbol{x}) = \frac{m(\boldsymbol{x}) - m_{min}}{m_{max} - m_{min} + \varepsilon} \quad \text{Eq.(3)}$$

Computation is recorded as accelerator-hours rather than nominal epochs because model size, sequence length, and batch size produce materially different costs. For resource level r, the measured cost is

$$c(\boldsymbol{x}, r) = \frac{1}{3600} \sum_{k=1}^{K} n_k(\boldsymbol{x}, r) t_k \quad \text{Eq.(4)}$$

where n_k denotes the number of active devices of type k, and t_k is their measured active time in seconds. A userselected coefficient controls the acceptable quality-cost trade-off:

$$J(\boldsymbol{x}, r) = q(\boldsymbol{x}, r) - \lambda log[1 + c(\boldsymbol{x}, r)] \quad \text{Eq.(5)}$$

The scalar objective is used for evolutionary selection, while its quality and cost components remain available for Pareto analysis.

Conditional crossover is performed only for genes that are active in one or both parents. Simulated binary crossover is employed for real-valued attributes, and uniform exchange is used for integers and categories. Crossover is performed, and then the activity mask is recomputed based on the child configuration and repaired once. Set mutation probabilities for all types of genes. Log-scaled variables, such as the learning rate and weight decay, are perturbed in logarithmic coordinates to avoid the upward bias caused by additive noise. As shown in Figure 1, typed encoding, feasibility repair, evolutionary variation, multi-fidelity evaluation, uncertainty calibration, archive maintenance and diversity feedback form a closed optimization architecture.

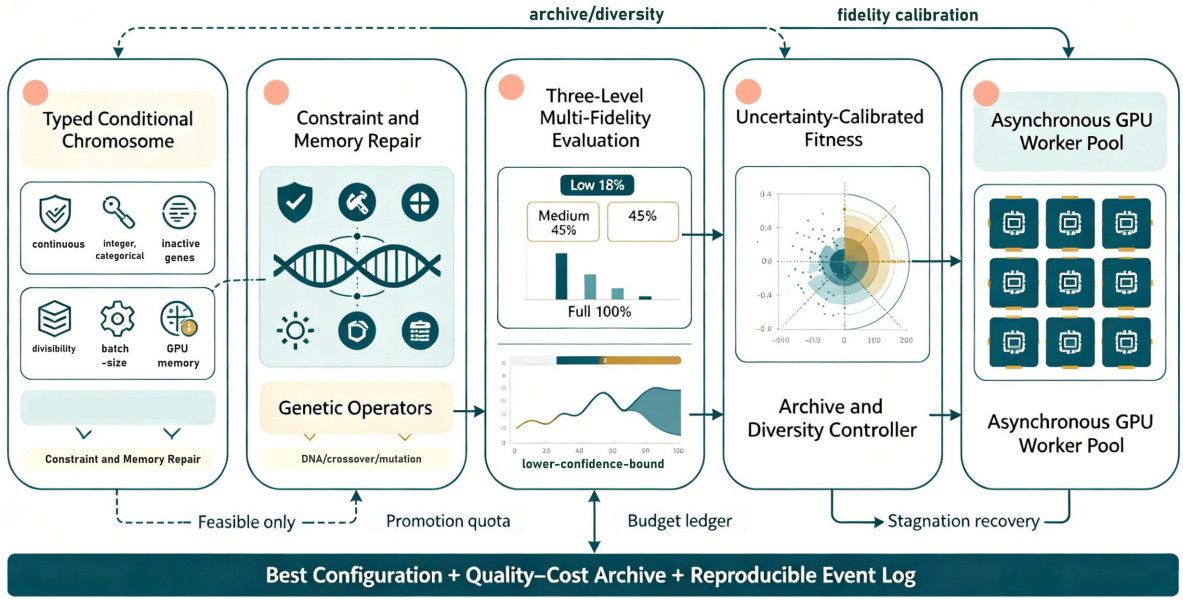

Figure 1. Architecture of the budget-aware genetic hyperparameter optimization framework.

Multi-Fidelity Fitness and Evolutionary Control

Every repaired candidate begins at a low resource level defined by a fraction of the complete training schedule and, where appropriate, a reduced data subset. A candidate can subsequently be promoted through three fidelity levels. Scores are corrected with a workload-specific learning-curve calibration fitted only from completed promotions. The optimizer retains both the observed score and its estimated standard error rather than replacing low-fidelity observations with point predictions.

Selection uses a conservative lower confidence bound:

$$LCB(\boldsymbol{x}, r) = \hat{\mu}_j(\boldsymbol{x}, r) - \beta_r \hat{\sigma}_j(\boldsymbol{x}, r) \quad \text{Eq.(6)}$$

This formulation prevents uncertain, inexpensive trials from dominating configurations supported by stronger fullfidelity evidence.

Parent selection combines tournament rank, evaluation age, and novelty. Age compensation limits completion-time bias in asynchronous execution, while novelty is calculated with a mixed-type distance over active genes:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{j=1}^{d} a_j(\boldsymbol{x}) a_j(\boldsymbol{y}) \delta_j(x_j, y_j)}{\sum_{j=1}^{d} a_j(\boldsymbol{x}) a_j(\boldsymbol{y}) + \varepsilon} \quad \text{Eq.(7)}$$

Here, δ_j is selected according to gene type: normalized absolute distance for numerical genes and Hamming distance for categorical genes. Candidate novelty is the mean distance to its k nearest neighbors:

$$N(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{y} \in NN_k(\boldsymbol{x})} d(\boldsymbol{x}, \boldsymbol{y}) \quad \text{Eq.(8)}$$

Elitism transfers a compact archive of nondominated quality-cost candidates into the next logical population.

Remaining slots are filled as workers become available. This steady-state design avoids a global generation barrier and allows long transformer trials to coexist with shorter tree-model evaluations.

Promotion is determined using conservative fitness, novelty, and the incremental cost of the next fidelity:

$$S_p(\boldsymbol{x}, r) = LCB(\boldsymbol{x}, r) + \eta N(\boldsymbol{x}) - \kappa \Delta c(\boldsymbol{x}, r) \quad \text{Eq.(9)}$$

A fixed promotion quota will not cover all of the low-cost completions within the full-fidelity budget. Candidates who violate the hard constraints will not be advanced, and configurations close to the soft memory threshold will incur a cost penalty. As shown in Figure 2, the event-driven scheduler connects offspring creation, low-fidelity screening, confidence-aware promotion, worker allocation, archive updates and stagnation recovery without a global synchronization barrier.

https://doi.org/10.64972/jaat.2024v2.267p12e:164-177

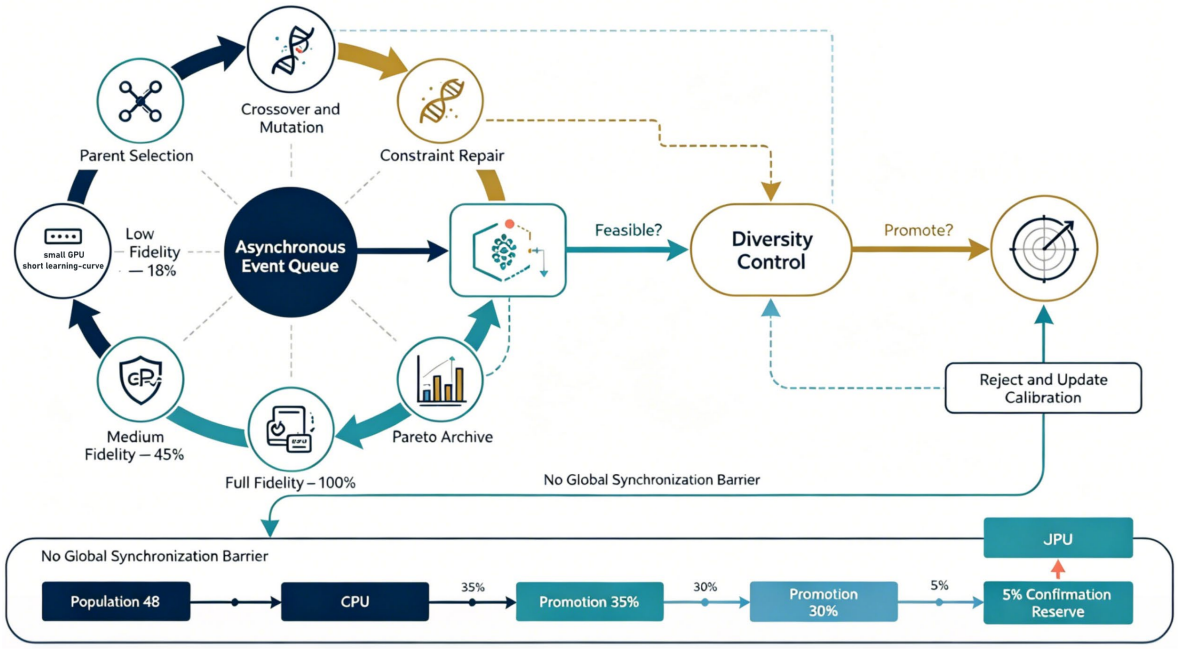

Figure 2. Event-driven workflow for asynchronous offspring creation.

Operator rates adapt to population diversity. When normalized diversity falls below its target corridor, mutation increases and tournament size decreases:

$$p_m(t) = clip\left[p_0 exp\left(\gamma(D^* - D_t)\right), p_{min}, p_{max}\right] \quad \text{Eq.(10)}$$

Population diversity is measured as the mean pairwise mixed-type distance:

$$D_t = \frac{2}{P(P-1)} \sum_{1 \le i < j \le P} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{Eq.(11)}$$

where P is the population size. When diversity is high and progress remains stable, exploitation is strengthened. A separate stagnation trigger reinitializes the least novel fraction of the population around sparsely sampled archive regions. Unlike a complete restart, this mechanism preserves elite configurations and the fitted fidelity calibration.

Distributed Execution and Complexity

The coordinator maintains the population, Pareto archive, fidelity model, and resource ledger. Workers receive immutable trial specifications containing the repaired chromosome, random seed, data-split identifier, fidelity level, and execution limit. Heartbeats distinguish slow progress from failed execution. An infrastructural failure is retried once, whereas deterministic configuration failures receive an infeasible label. This distinction prevents cluster instability from becoming artificial selection pressure.

For real-valued genes, offspring generation uses a type-preserving crossover expression:

$$z_j' = \frac{1}{2}\left[(1+\xi)z_j^{(1)} + (1-\xi)z_j^{(2)}\right] \quad \text{Eq.(12)}$$

where $z_j^{(1)}$ and $z_j^{(2)}$ are parental values and ξ is sampled by the crossover operator. Integer and categorical offspring are subsequently converted through their corresponding typed operators and the deterministic repair function.

Reproducibility is supported at three levels. Search randomness is controlled by a master seed, each trial uses a derived seed, and scheduling events are written to an append-only log. Because asynchronous completion order can still alter the search trajectory, experiments are repeated with ten master seeds and compared through distributions rather than a single best run. Checkpoints contain population state, archive state, calibration parameters, pending-trial metadata, and consumed budget.

The stagnation gap at event t is recorded as

$$E_t = \max_{\tau \le t} J_\tau^* - J_t^* \quad \text{Eq.(13)}$$

where J_t^* is the best confirmed fitness available at the current event. A sustained nonzero gap, together with low population diversity, activates archive-guided recovery.

Budget accounting occurs when a worker emits telemetry rather than when a trial is submitted. Preemption, dataloading stalls, and elastic allocation make nominal reservation time a poor approximation of consumed computation. The ledger records device-active seconds, host preprocessing time, checkpoint traffic, and termination reason. Only device-active time enters the primary optimization budget, while the remaining fields support operational diagnosis. Five percent of the total budget is withheld from ordinary offspring and released for final confirmation of archive candidates.

Worker utilization is measured as

$$U = 1 - \frac{\sum_{w=1}^{W} T_w^{idle}}{WT} \quad \text{Eq.(14)}$$

where W is the number of workers, T is elapsed search time, and T_w^{idle} is the idle duration of worker w. Concurrency changes the effective population dynamics. If every completion immediately produced an offspring from the newest population, fast configurations would contribute descendants more frequently than slow configurations even at identical fitness. Age compensation corrects part of this bias, and the scheduler limits each lineage to two pending descendants. Completed evidence is versioned so that an offspring remains linked to the parent states used at creation. This event history permits reconstruction of selection decisions and controlled replay under alternative promotion policies.

The coordinator cost is modest relative to model training. Mixed-distance computation grows with population size and the number of active genes, while archive comparison grows with archive size. In the evaluated regime, these operations remain below one percent of wall-clock time. Communication primarily consists of small trial specifications and metric records, while model checkpoints remain in shared object storage. The architecture therefore scales with the number of available evaluators until queue starvation or shared-storage contention becomes dominant.

Experimental Evaluation and Analysis

Experimental Setup and Comparative Performance

Three Workloads were chosen for the experiment to show different search-space structures. GBT-100M is a workload for training gradient-boosted trees on 100 million anonymised tabular records with 18 active or conditional hyperparameters. Transformer-320M fine-tuned a 320-million-parameter encoder for six-class technical document classification using 24 hyperparameters. GNN-48M workload trained a 48-million-parameter graph attention network on a 32-million-edge interaction graph with 21 hyperparameters. Search spaces include log-continuous schedules, bounded integers, optimizer categories, augmentation switches and architecture dependencies. Each method had 480 accelerator-hours per run and ten independent runs. The final candidates were retrained at full fidelity using three seeds on the held-out validation set.

Baselines included random search, Gaussian-process or density-based Bayesian optimization according to the dimension of the space, Hyperband and a traditional synchronous genetic algorithm. Population size was 48 for both genetic methods. The proposed way had fidelity fractions of 0.18, 0.45, and 1.0, and promotion quotas of 35% and 30%. Quality is normalised to the best full-fidelity validation result obtained among all methods for a given workload. Infrastructure included 32 A100-class accelerators and a shared object store. Energy was taken from device telemetry every 10 seconds. The protocol now reports the uncertainty and resource consumption of the optimizer comparison, as suggested by [26].

Figure 3(a) shows that the proposed method reached normalized quality of 0.921, 0.907, and 0.911 on GBT-100M, Transformer-320M, and GNN-48M, respectively; the corresponding Bayesian values were 0.902, 0.884, and 0.881. In Figure 3(b), median accelerator uses to reach 98% of each run's final quality was 231, 318, and 287 hours, compared with 344, 461, and 408 hours for the conventional genetic algorithm. Figure 3(c) combines wall-clock completion with run-level quality-compute position: completion times were 9.6, 13.8, and 12.4 hours,

https://doi.org/10.64972/jaat.2024v2.267p12e:164-177

and 25 of 30 proposed-method runs lay on the empirical nondominated envelope. These observations are consistent with the view that optimizer ranking should be conditioned on budget rather than only terminal score [27].

Figure 3. Overall comparative performance: (a) grouped bars of normalized validation quality across three workloads and five optimizers; (b) run-level box distributions of accelerator-hours required to reach 98% of terminal quality; (c) quality-compute Pareto scatter with nondominated envelope and budget reference.

The Transformer workload produced the largest absolute compute saving because the early low-fidelity evaluations were expensive enough to distinguish between weak learning-rate and sequence-length combinations but still significantly cheaper than full fine-tuning. GBT-100M had a smaller quality difference between methods but the highest trial throughput. GNN-48M produced more noisy rankings in the first fidelity, and thus the promotion controller kept a larger uncertainty margin. The mean final quality across workloads for the proposed optimizer, Bayesian optimisation, Hyperband, conventional genetic algorithm and random search was 0.913, 0.889, 0.881, 0.872 and 0.851, respectively. No comparison uses a single best run and avoids the instability reported in large-scale search studies [28].

Per-workload dispersion indicates that the average advantage was not achieved by a single favourable task. The interquartile ranges of the proposed methods were 0.007 for GBT-100M, 0.011 for Transformer-320M and 0.014 for GNN-48M. The corresponding ranges of the traditional genetic algorithm were 0.012, 0.026 and 0.023. Seed variation was reduced most significantly in the transformer task, and the traditional population often prematurely committed to an optimizer category. The archive saved configurations of the three optimizer families in more than 70% of the budget for the eight proposed-method runs compared to the two runs without fixed mutation. Terminal robustness was therefore associated with preserved categorical diversity rather than with extra random exploration.

Trial accounting separates useful training from avoidable expenditure. Across all workloads, 61.3% of conventional genetic-algorithm compute was spent on configurations finishing below the median final quality. The proposed method reduced this share to 34.7%, while 17.8% of its budget supported broad low-fidelity screening. Failed jobs consumed 2.1% of the budget, compared with 8.9% without repair. Data loading and final validation contributed 4.3%. Remaining compute supported promoted candidates that supplied either an archive member or calibration evidence. Terminal score alone is thus an incomplete systems metric: two optimizers can return similar incumbents while imposing very different amounts of nonproductive training.

https://doi.org/10.64972/jaat.2024v2.267p12e:164-177

A separate confirmation stage retrained the five top configurations from each run using three data-order seeds. The proposed method kept its within-run winner in 24 out of 30 runs; in four runs, the second archived member was better, and in two runs, the third member was the best. In 18 trials, the original best solution of Bayesian optimization was still the best. Mean optimism is defined as search-time quality minus confirmation quality, and these values for the proposed method and Bayesian optimization were 0.0038 and 0.0089, respectively. The difference is due to both repeated noisy observations and a wide archive. It can also report a confirmed set and, in this case, does not need to present the single maximum validation observation as a stable model choice.

Search-space occupancy was measured for discretised marginal bins and conditional paths. Random search covered 82% of the bins but allocated trials almost uniformly in the weak and strong regions. The old genetic algorithm accounted for 47% and was used excessively after the first quarter of the budget. The proposed method covered 66%, and 73% of its full-fidelity evaluations were in the best two deciles of retrospectively estimated quality. Coverage was relatively high for the categorical optimizer and architecture decisions compared to fine-grained continuous schedules. This pattern matches the intended division of labour: crossover preserves good combinations, adaptive mutation reopens collapsed coordinates, and fidelity allocation avoids paying the full training cost for wide exploration.

Convergence, Fidelity Behavior, and Ablation

Figure 4(a) traces median best-so-far quality against accelerator-hours. The proposed curve crossed 0.88 after 146 hours, whereas Bayesian optimization, Hyperband, and the conventional genetic algorithm required 209, 188, and 254 hours. Figure 4(b) presents candidate attrition as a waterfall: the initial 1,440 low-fidelity trials lost 954 candidates at the first boundary, 344 after medium fidelity, and 51 after full-fidelity confirmation, leaving 91 confirmed top-decile configurations. Figure 4(c) displays run-level distributions of low-to-full-fidelity rank correlation across budget quartiles; the median increased from 0.61 in the first quartile to 0.79 in the final quartile, while dispersion narrowed as calibration evidence accumulated. The uncertainty penalty prevented the broad early distribution from being treated as exact ordering. These patterns align with evidence that fidelity selection must reflect both cost and ranking reliability [29].

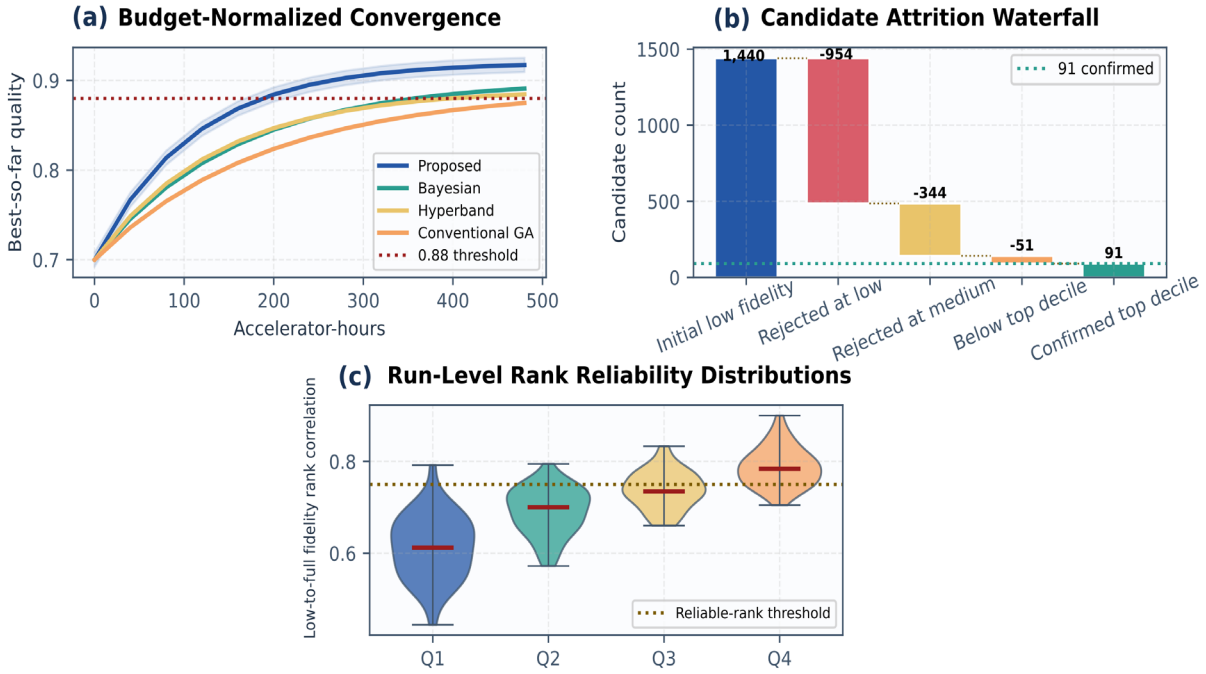

Figure 4. Convergence and fidelity dynamics: (a) best-so-far normalized quality versus accelerator-hours; (b) waterfall decomposition of candidate attrition across fidelity boundaries; (c) violin distributions of low-to-full-fidelity rank correlation over budget quartiles.

Ablation experiments removed one component at a time while preserving the 480-hour budget. Removing multi-fidelity promotion increased accelerator consumption at the 0.90 quality threshold by 28.9%. Replacing uncertainty-adjusted fitness with raw low-fidelity score reduced final quality by 1.8 percentage points on GNN-

https://doi.org/10.64972/jaat.2024v2.267p12e:164-177

48M. Disabling typed repair increased infeasible trials from 1.7% to 12.8%. A fixed mutation rate caused three of ten transformer runs to converge to the same narrow optimizer-and-width region. Asynchronous scheduling alone reduced idle worker time, but its benefit was smaller when promotion was absent because too many long, weak trials entered full training.

Figure 5(a) quantifies these effects through the change in final quality: removing uncertainty calibration produced the largest loss, 1.8 points, followed by diversity adaptation at 1.4 points. Figure 5(b) shows compute inflation of 28.9%, 9.7%, 7.6%, and 4.1% for the no-fidelity, no-repair, fixed-mutation, and synchronous variants. Figure 5(c) presents the interaction between mutation rate and normalized diversity; successful runs occupied a corridor from 0.22 to 0.38 diversity with mutation probabilities between 0.08 and 0.19. The nonlinear response supports adaptive rather than globally fixed operator settings [30].

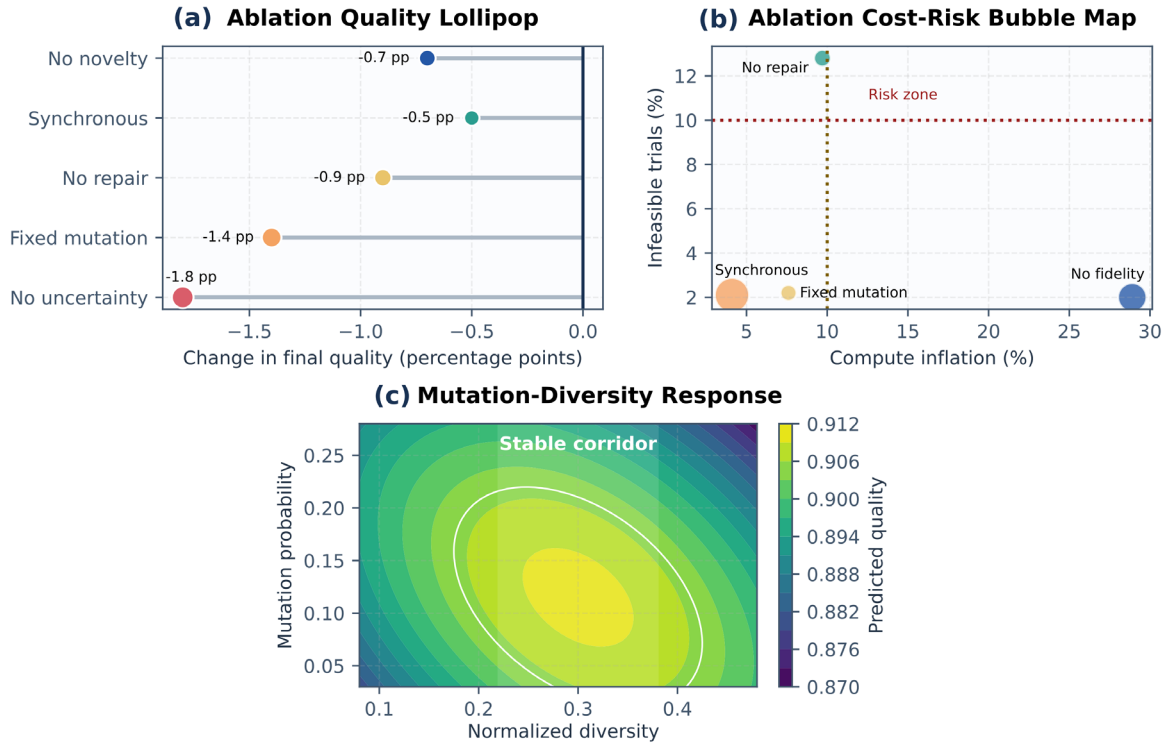

Figure 5. Component ablation and operator behavior: (a) lollipop ranking of final-quality loss for five ablations; (b) bubble map relating compute inflation, infeasible-trial rate, and wall-clock penalty; (c) mutation-diversity response contour with the stable operating corridor.

Promotion errors were examined by retrospectively evaluating a stratified sample of 180 rejected candidates at full fidelity. Twelve would have entered the top decile, giving a false-rejection rate of 6.7%. Without uncertainty adjustment, the same rate was 14.4%. The calibrated method therefore discarded some late-blooming configurations, but the loss was limited relative to the compute saved. Increasing the first promotion quota from 35% to 50% reduced false rejection to 4.9% while consuming 13.2% more compute. This tunable relationship is preferable to claiming that any low-fidelity rule is universally safe [31].

Learning-curve calibration differed by model family. Median absolute prediction error at medium fidelity was 0.006 for GBT-100M, 0.011 for Transformer-320M, and 0.015 for GNN-48M. Coverage of the nominal 90% interval was 91.4%, 88.7%, and 87.9%, respectively. The slightly low graph-model coverage explains why a larger conservative penalty improved its final ranking. Calibration was refitted after every 24 promoted observations; shortening this interval to 12 changed terminal quality by less than 0.2 points but increased coordinator work by 38%. Extending it to 48 observations produced visible lag during the first half of the budget. The selected interval represents an engineering compromise rather than a universal statistical constant.

Diversity recovery was triggered 2.1 times per transformer run and 1.3 times per graph run, but only 0.6 times per tree run. Following a trigger, median nearest-neighbor distance increased from 0.14 to 0.27 within 18 completed trials. Best-so-far quality improved within the next 60 accelerator-hours in 79% of trigger events.

https://doi.org/10.64972/jaat.2024v2.267p12e:164-177

Uniform random injection achieved a larger immediate distance of 0.34, yet only 52% of events improved within the same budget because many samples entered weak categorical regions. Archive-guided recovery was less disruptive and made better use of conditional structure already identified by the search.

The Advantage of conservative ranking changed over time. In the first budget quarter, 37 per cent of the promoted transformer candidates were selected partly because their confidence intervals included the current promotion threshold. The proportion was 18 per cent in the previous quarter due to a shorter calibration period. Removing the uncertainty term resulted in faster visible progress for a short time, but six early leaders later dropped out of the full-fidelity median group. Only two of these reversals included an uncertainty term. Therefore, the optimizer accepted a relatively small early ambiguity in exchange for a lower probability of spending the expensive final stage on overestimated candidates.

Pairwise component interactions were non-additive. Both repair and adaptive mutation reduced the transformer quality by 3.7 points; the sum of their individual losses was exceeded by 0.9 points because invalid offspring concentrated the mutation in the remaining active coordinates. Removing asynchronous execution and multi-fidelity promotion increased the wall-clock time by 51.6%, compared with the isolated increases of 27.4% and 18.8%. The scheduler benefits more from asynchrony when the trial durations are not uniform at all fidelity levels. The above interactions are sufficient to consider the framework a system; however, controlled ablations of its components have also been conducted.

Scalability, Robustness, and Resource Analysis

Scaling tests varied the worker count from 4 to 64 while holding the search budget and population policy constant. Figure 6(a) shows throughput rising from 1.8 to 19.7 completed trial-equivalents per hour between 4 and 32 workers, then reaching 27.1 at 64 workers. Figure 6(b) decomposes worker time into active execution, queue starvation, storage waiting, and retry or other overhead; active time declined from 94.0% at four workers to 82.3% at 64 workers, while queue-idle and storage-wait shares rose to 7.8% and 6.9%. Figure 6(c) presents the full scheduling-latency distributions rather than only two summary curves. Median latency increased from 38 ms to 214 ms and the 95th percentile from 71 ms to 603 ms, with visibly broader right tails at 32 and 64 workers. Similar saturation behavior has been observed in distributed optimization services [32].

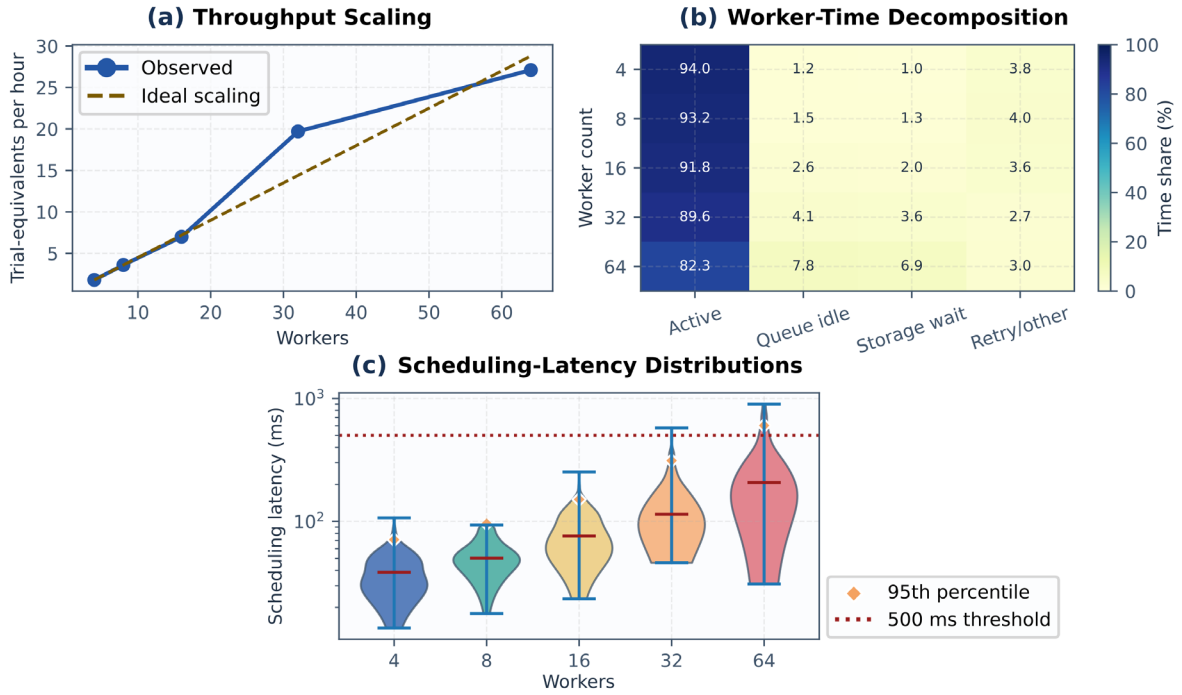

Figure 6. Distributed scalability: (a) completed trial-equivalents per hour and ideal-scaling reference; (b) heatmap decomposition of active, queue-idle, storage-wait, and retry or other worker time; (c) violin distributions of scheduling latency with 95th-percentile markers across worker counts.

https://doi.org/10.64972/jaat.2024v2.267p12e:164-177

Robustness was tested underscore noise, worker failure, and deliberate model mismatch in the fidelity calibration. With Gaussian score noise of standard deviation 0.01, terminal quality declined by 0.4 points; at 0.03 noise, the decline was 1.5 points. A 5% transient worker-failure rate increased wall-clock time by 3.2% because retried specifications retained their evaluation history. When calibration was transferred from Transformer-320M to GNN-48M without refitting, low-fidelity error increased sharply and final quality dropped by 2.3 points. Online refitting recovered 1.7 of those points after 120 completed observations. This result cautions against assuming that learning-curve priors transfer unchanged across model families [33].

Figure 7(a) presents the sensitivity of terminal quality to population size and promotion quota; the broad plateau between populations of 40-56 and quotas of 30%-40% indicates moderate tuning robustness. Figure 7(b) compares normalized resource-efficiency profiles. The proposed method scored 82 for useful training, 74 for promotion efficiency, 98 for feasibility, 90 for worker utilization, and 86 for energy efficiency; the conventional genetic algorithm scored 61, 48, 87, 72, and 60 on the same 0-100 scales. Figure 7(c) shows energy per successful top-decile configuration falling from 42.8 kWh for the conventional genetic algorithm to 29.6 kWh. The reduction is relevant because tuning energy can exceed the cost of one final training run [34].

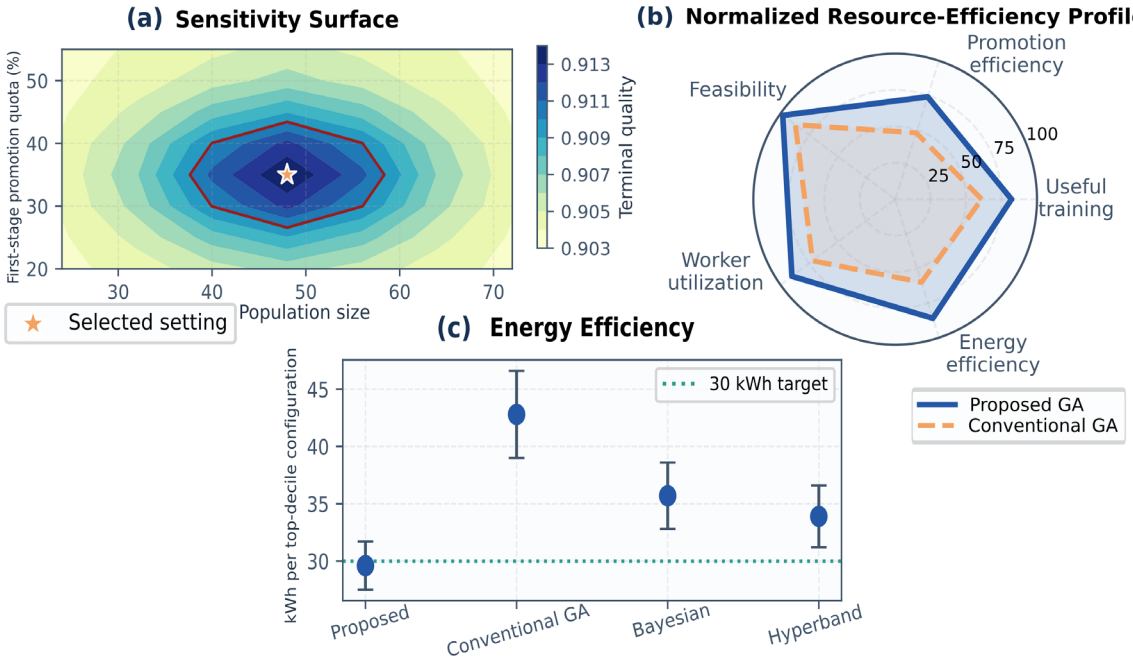

Figure 7. Robustness and resource profile: (a) terminal-quality contour over population size and first-stage promotion quota; (b) radar comparison of normalized resource-efficiency dimensions for the proposed and conventional genetic algorithms; (c) energy per top-decile configuration with uncertainty intervals and a 30-kWh target.

The proposed method does not dominate every regime. Bayesian optimization was competitive during the first 80 accelerator-hours on GBT-100M, where the response surface was comparatively smooth. Hyperband achieved similar early progress when learning-curve rank correlation exceeded 0.8. The genetic method became more favorable as conditionality, categorical decisions, and evaluation parallelism increased. Statistical comparison over ten runs used paired bootstrap intervals and a Friedman test followed by corrected pairwise comparisons. The proposed method's quality advantage over random search, Hyperband, and the conventional genetic algorithm was significant at the 0.05 level; its 2.4-point advantage over Bayesian optimization had a 95% interval of 1.1-3.6 points. Reporting effect sizes alongside significance avoids overinterpreting large experimental logs [35].

Threats to validity include the use of three workloads, one accelerator class, and a scalar quality-cost coefficient during selection. The measured gains may change for autoregressive models with much longer trials or for spaces dominated by continuous variables. Energy estimates use device telemetry and exclude embodied hardware cost. The repair rules also encode domain knowledge, so part of the gain reflects better problem

representation rather than the evolutionary loop alone. These boundaries are made explicit because reproducible optimizer evaluation depends on transparent workloads, budgets, and failure policies [36].

Fairness of Resources was investigated because Asynchronous Schedulers may inadvertently favour small trials. Based on the group proportions for the dominant-resource shares of accelerator memory, active compute time and checkpoint bandwidth, these have been classified. The proposed scheduler reduced the 90th-to-10th percentile ratio of accumulated dominant-resource share to 1.34 from 1.71 in the unrestricted first-come dispatch. Strictly equal shares reduced throughput by 8.6 per cent because large-memory transformer jobs could not use temporary gaps. The selected soft fairness penalty reduced lineage and model-family imbalance without requiring equal distribution. Over the course of ten runs, no model family was under 27% of full-fidelity capacity, and the highest proportion was 39%. It is necessary to balance them in a joint service to tune several workloads that have different completion speeds.

Finally, sensitivity to the quality-cost coefficient was measured at 0.00, 0.02, 0.05, 0.08, and 0.12. A zero-coefficient maximized terminal quality to 0.918 but consumed the entire 480-hour allowance in every run. The selected value of 0.05 produced quality of 0.913 while reaching its confirmed incumbent after 329 hours on average. At 0.12, consumption fell to 244 hours but quality declined to 0.899 because promotion became overly conservative for the transformer workload. The empirical knee occurred between 0.04 and 0.07 across all workloads. This interval provides a practical default, while deployments facing hard energy or deadline constraints can choose a larger coefficient and inspect the retained Pareto archive instead of rerunning the search.

Queue inspection explains the utilization curve. At 32 workers, the median ready queue contained 11 low-fidelity candidates and four promotions, enough work to absorb normal duration variance. At 64 workers, the median queue fell to six items and was empty for 7.8% of scheduler observations. Increasing offspring prefetch removed most starvation but weakened selection because children were created from stale population states. A prefetch limit equal to one quarter of the population preserved quality within 0.3 points and raised 64-worker utilization from 82.3% to 86.1%. Larger limits improved utilization marginally while increasing average parent-state age from 7.4 to 19.6 completion events.

Energy sensitivity used three grid-intensity scenarios rather than a single carbon claim. At 0.20, 0.40, and 0.60 kg CO2-equivalent per kWh, median emissions associated with discovering a top-decile transformer configuration were 5.9, 11.8, and 17.8 kg for the proposed method. Conventional genetic search produced 8.6, 17.1, and 25.7 kg under the same assumptions. The proportional reduction remains stable because both methods used identical hardware, although absolute values depend on location and time. Checkpoint traffic was 18% lower because fewer weak candidates reached stages that emit large optimizer states. This secondary saving was excluded from the primary device-energy comparison.

Inject coordinator interruptions at 25%, 50% and 75% of the budget to assess operational resilience. Restoring the latest checkpoint took 18-41 seconds and lost no more than a single population update; thus, workers continued because the trial specifications were inflexible. Trial and fidelity identifiers detected the duplicate completion message. Replaying ten interrupted runs increased the mean of normalized terminal quality by 0.0008 over uninterrupted event logs. Reliability is relatively inexpensive but may be less credible. An optimizer that performs well only under failure-free scheduling may offer no benefit in a shared-infrastructure environment.

Conclusion

Formulate the large-scale hyperparameter tuning as a combined search, evidence allocation and scheduling problem in this paper. The proposed budget-aware genetic algorithm combined conditional typed chromosomes, deterministic feasibility repair, uncertainty-aware multi-fidelity fitness, adaptive diversity control and asynchronous execution. For the Tree, Transformer and Graph workloads, the integrated design improved terminal validation quality and reduced accelerator usage and wall-clock time compared with traditional evolutionary search and popular non-evolutionary baselines. Based on the experiments, the reasons for the improvement in efficiency were determined to be calibrated promotion and feasibility-aware representation; diversity control helps prevent early convergence.

The number of workloads, a fixed three-level fidelity ladder, and manual specification of repair constraints are the current limitations of the study. Its calibration model will reject a small number of late-improving candidates, and the scalar quality-cost fitness does not cover all deployment requirements. Hardware diversity, communication-intensive training, and very large autoregressive models may be introducing bottlenecks that are not reflected in the current cluster and task set.

Future work will learn repair operators from execution traces, adapt fidelity levels continuously, and add latency, memory and energy as explicit objectives. A decentralized island deployment can enhance the fault tolerance of heterogeneous clusters, and calibrated priors can be transferred among related workloads to reduce cold-start overhead. Periodically analyze asynchronous selection bias and carry out an all-weather assessment of the foundation model training pipeline to determine when evolutionary tuning is more advantageous in practice.

Author Contributions

Kamila Jaworska contributes to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision. Violetta Mikołajczyk and Zofia Kukla contribute to methodology, software, validation, analysis, investigation. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

References

[1] Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061

[2] Hertel, L., Collado, J., Sadowski, P., Ott, J., & Baldi, P. (2020). Sherpa: Robust hyperparameter optimization for machine learning. SoftwareX, 12, 100591. https://doi.org/10.1016/j.softx.2020.100591

[3] Liu, Y., & Wang, H. (2020). Efficient hyperparameter optimization through model-based reinforcement learning. Neurocomputing, 409, 381–393. https://doi.org/10.1016/j.neucom.2020.06.064

[4] Liu, Y., & Wang, H. (2020). Fast hyperparameter tuning using Bayesian optimization with directional derivatives. Knowledge-Based Systems, 205, 106247. https://doi.org/10.1016/j.knosys.2020.106247

[5] Yao, Q., Wang, M., Chen, Y., Dai, W., Li, Y.-F., Tu, W.-W., Yang, Q., & Yu, Y. (2020). Taking human out of learning applications: A survey on automated machine learning. Artificial Intelligence Review, 53, 4407–4446. https://doi.org/10.1007/s10462-019-09770-9

[6] Hamdia, K. M., Zhuang, X., & Rabczuk, T. (2021). An efficient optimization approach for designing machine learning models based on genetic algorithm. Neural Computing and Applications, 33(6), 1923–1933. https://doi.org/10.1007/s00521-020-05035-x

[7] Demo, N., Tezzele, M., & Rozza, G. (2021). A supervised learning approach involving active subspaces for an efficient genetic algorithm in high-dimensional optimization problems. SIAM Journal on Scientific Computing, 43(3), B831–B853. https://doi.org/10.1137/20M1345219

[8] El-Mihoub, T. A., Hopgood, A. A., & Nolle, L. (2021). Self-adaptive learning for hybrid genetic algorithms. Evolutionary Intelligence, 14, 1565–1579. https://doi.org/10.1007/s12065-020-00425-5

[9] Moran, A., & Blei, D. M. (2020). HY-POP: Hyperparameter optimization of machine learning models through parametric programming. Computers & Chemical Engineering, 139, 106902. https://doi.org/10.1016/j.compchemeng.2020.106902

[10] Wang, Y., Franzon, P. D., Smart, D., & Swahn, B. (2020). Multi-fidelity surrogate-based optimization for electromagnetic simulation acceleration. ACM Transactions on Design Automation of Electronic Systems, 25(5), 1–21. https://doi.org/10.1145/3398268

[11] Tenne, Y. (2022). An analysis of the RBF hyperparameter impact on surrogate-assisted evolutionary optimization. Scientific Programming, 2022, 5175941. https://doi.org/10.1155/2022/5175941

[12] van Rijn, S., Schmitt, S., van Leeuwen, M., & Bäck, T. (2022). Finding efficient trade-offs in multi-fidelity response surface modelling. Engineering Optimization, 55(6), 946–963. https://doi.org/10.1080/0305215X.2022.2052286

[13] Lei, B., Kirk, T. Q., Bhattacharya, A., Pati, D., Qian, X., Arroyave, R., & Mallick, B. K. (2021). Bayesian optimization with adaptive surrogate models for automated experimental design. npj Computational Materials, 7, 194. https://doi.org/10.1038/s41524-021-00662-x

[14] Hassan, S., Hemeida, A. M., Alkhalaf, S., Mohamed, A. A., & Senjyu, T. (2020). Multi-variant differential evolution algorithm for feature selection. Scientific Reports, 10, 17261. https://doi.org/10.1038/s41598-020-74228-0

[15] Albadr, M. A., Tiun, S., Ayob, M., & Al-Dhief, F. (2020). Genetic algorithm based on natural selection theory for optimization problems. Symmetry, 12(11), 1758. https://doi.org/10.3390/sym12111758

[16] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8, 53. https://doi.org/10.1186/s40537-021-00444-8

[17] Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green AI. Communications of the ACM, 63(12), 54–63. https://doi.org/10.1145/3381831

[18] Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D. R., Texier, M., & Dean, J. (2021). Carbon emissions and large neural network training. Computer, 55(7), 22–31. https://doi.org/10.1109/MC.2022.3148714

[19] Borysenko, O., & Byshkin, M. (2021). CoolMomentum: A method for stochastic optimization by Langevin dynamics with simulated annealing. Scientific Reports, 11, 10705. https://doi.org/10.1038/s41598-021-90144-3

[20] Rai, A. (2020). Explainable AI: From black box to glass box. Journal of the Academy of Marketing Science, 48, 137–141. https://doi.org/10.1007/s11747-019-00710-5

[21] He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622. https://doi.org/10.1016/j.knosys.2020.106622

[22] Zöller, M.-A., & Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. Journal of Artificial Intelligence Research, 70, 409–472. https://doi.org/10.1613/jair.1.11854

[23] Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: A systematic review. Archives of Computational Methods in Engineering, 29, 2531–2561. https://doi.org/10.1007/s11831-021-09694-4

[24] Slowik, A., & Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications, 32, 12363–12379. https://doi.org/10.1007/s00521-020-04832-8

[25] Agushaka, J. O., Ezugwu, A. E., & Abualigah, L. (2022). Dwarf mongoose optimization algorithm. Computer Methods in Applied Mechanics and Engineering, 391, 114570. https://doi.org/10.1016/j.cma.2022.114570

[26] Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The arithmetic optimization algorithm. Computer Methods in Applied Mechanics and Engineering, 376, 113609. https://doi.org/10.1016/j.cma.2020.113609

[27] Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. Knowledge-Based Systems, 191, 105190. https://doi.org/10.1016/j.knosys.2019.105190

[28] Hashim, F. A., Hussain, K., Houssein, E. H., Mabrouk, M. S., & Al-Atabany, W. (2022). Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. Applied Intelligence, 51, 1531–1551. https://doi.org/10.1007/s10489-020-01893-z

[29] Li, J.-Y., Zhan, Z.-H., Wang, C., Jin, H., & Zhang, J. (2020). Boosting data-driven evolutionary algorithm with localized data generation. IEEE Transactions on Evolutionary Computation, 24(5), 923–937. https://doi.org/10.1109/TEVC.2020.2979740

[30] Liu, R., Li, J., Fan, J., Mu, C., & Jiao, L. (2021). A coevolutionary technique based on multi-swarm particle swarm optimization for dynamic multi-objective optimization. European Journal of Operational Research, 291(1), 317–333. https://doi.org/10.1016/j.ejor.2020.09.032

[31] Zhang, Y. (2022). Logistics distribution scheduling model of supply chain based on genetic algorithm. Journal of Industrial and Production Engineering, 39(2), 83–88. https://doi.org/10.1080/21681015.2021.1958938

[32] Du, Y., Wang, J., & Lei, D. (2021). Dynamic decision support framework for production scheduling using a combined genetic algorithm and multiagent model. Expert Systems, 38(1), e12533. https://doi.org/10.1111/exsy.12533

[33] Dhaenens, C., & Jourdan, L. (2020). Metaheuristics for big data. European Journal of Operational Research, 283(2), 404–419. https://doi.org/10.1016/j.ejor.2019.11.048

[34] Pietrenko-Dabrowska, A., & Koziel, S. (2022). Rapid yield optimization of miniaturized microwave passives by response features and variable-fidelity EM simulations. Scientific Reports, 12, 22440. https://doi.org/10.1038/s41598-022-26562-8

[35] Brownlee, A. E. I., Wright, J. A., He, M., Lee, T., & McMenemy, P. (2022). A multi-objective optimization algorithm based on a multi-agent blackboard system. Journal of Computational Design and Engineering, 9(2), 480–506. https://doi.org/10.1093/jcde/qwac009

[36] Pietikäinen, M., & Silvén, O. (2022). Challenges of artificial intelligence—from machine learning and computer vision to emotional intelligence. Journal of Imaging, 8(12), 331. https://doi.org/10.3390/jimaging8120331