IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 June 2026, accepted 11 July 2026. Date of publication 00 xxxx 0000, date of current version 00 xxxx 0000.

Digital Object Identifier 10.1109/ACCESS.2026.3724200

Coincidence Algorithm With Proportional Wheel Selection (COIN-PWS): A Hierarchical Approach to Sudoku Puzzle Solving

DARUNEE BUNMA AND PRABHAS CHONGSTITVATANA
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
Corresponding author: Prabhas Chongstitvatana (prabhas.c@chula.ac.th)

ABSTRACT This paper introduces COIN-PWS (Coincidence Algorithm with Proportional Wheel Selection), an enhanced Estimation of Distribution Algorithm (EDA) for solving Sudoku Constraint Satisfaction Problems. In contrast to the baseline COIN algorithm, which applies a fixed top-c%/bottom-c% threshold to partition candidates into reward and penalty groups, COIN-PWS employs Stochastic Universal Sampling (SUS) roulette wheels to assign continuously proportional selection weights. This smooth learning gradient is complemented by (i) a hierarchical 3×3 block model comprising nine independent block probability matrices (H_{block}) that capture local sub-grid dependencies, and (ii) a constraint-aware sampling procedure that ensures all generated candidates partially satisfy row, column, and block constraints. The algorithm is benchmarked on 200 procedurally generated Sudoku puzzles across four difficulty levels (50 each of Easy, Medium, Hard, and Expert). COIN-PWS achieves a 100% solve rate across all levels with 95% confidence intervals: 1.3 ± 0.2 generations in 0.041 ± 0.013 s (Easy), 8.0 ± 2.4 generations in 0.267 ± 0.092 s (Medium), 53.3 ± 5.2 generations in 3.699 ± 0.336 s (Hard), and 151.8 ± 9.1 generations in 14.526 ± 0.864 s (Expert). A computational complexity analysis establishes the total cost as $\mathcal{O}(T \cdot N \cdot |E|)$, where T is the number of generations, N is the population size, and $|E|$ is the number of empty cells to be filled. An illustrative step-by-step example is provided to demonstrate the execution of the algorithm. Compared to the best previous hybrid genetic algorithm, COIN-PWS is approximately $358\times$ faster on Easy puzzles and achieves a 100% Hard solve rate versus less than 20% per run for the prior method. The framework is dimension-agnostic and generalizes to the grids 16×16 and 25×25. These results establish COIN-PWS as a highly competitive, generalizable approach to optimizing constraint satisfaction.

INDEX TERMS Coincidence algorithm, estimation of distribution algorithm, proportional wheel selection, stochastic universal sampling, Sudoku, constraint satisfaction, metaheuristics, hierarchical probabilistic model, computational complexity.

I. INTRODUCTION

The Sudoku puzzle is a well-known Constraint Satisfaction Problem (CSP) defined on a 9×9 grid where each of 81 cells must be assigned a digit from $\{1, 2, \ldots, 9\}$ subject to three sets of constraints: all nine digits must appear exactly once in every row, every column, and every non-overlapping 3×3 sub-grid [1]. Despite the simplicity of these rules, the puzzle presents a combinatorially rich search space with approximately 6.67×10^{21} valid complete configurations [2], making it an ideal testbed for optimisation algorithms [3].

Estimation of Distribution Algorithms (EDAs) replace the genetic operators of crossover and mutation with a probabilistic model of the promising solution space, iteratively refining this model using statistical information from selected candidates [4]. The Coincidence Algorithm (COIN), proposed by Wattanapornprom and Chongstitvatana [2], is an EDA

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Fadda.

with a distinctive dual-learning mechanism: it simultaneously reinforces structural patterns found in the best solutions (reward) and suppresses patterns found in the worst solutions (punishment) [2]. This bidirectional update is a significant departure from most EDAs, which learn only from the elite set and discard poor candidates [4].

Despite its theoretical appeal, the original COIN applies a hard binary selection threshold: the top $c\%$ of the ranked candidates are rewarded and the bottom $c\%$ are penalized [2]. This creates a discontinuous learning signal in which candidates just above and below the thresholds receive identical contributions regardless of their actual fitness values. As the population converges and fitness differences narrow, this threshold-based approach becomes increasingly noisy and may slow the convergence.

This paper presents COIN-PWS, an enhanced COIN that replaces the hard threshold with Proportional Wheel Selection implemented via Stochastic Universal Sampling (SUS) [5]. Three additional contributions distinguish COIN-PWS: (i) a hierarchical 3×3 block probability model that captures the local Sudoku structure; (ii) an adaptive learning rate that decays linearly to prevent premature convergence; and (iii) constraint-aware sampling that produces partially valid candidates at generation time, substantially reducing the effective search space. This work also provides: A computational complexity analysis (Section IV-G), a step-by-step illustrative example (Section V), statistical confidence intervals for all reported results, and a discussion of scalability to larger grid sizes.

We benchmark COIN-PWS on 200 procedurally generated Sudoku puzzles—50 each at Easy, Medium, Hard, and Expert difficulty—and compare against baseline COIN [2], the best prior hybrid Genetic Algorithm (HGA) [6], and the best prior EDA for Sudoku (RESEDA) [7]. COIN-PWS achieves a 100% solve rate at every difficulty tier and demonstrates substantially faster convergence than all prior methods.

II. RELATED WORK

A. CONSTRAINT-BASED AND EXACT METHODS

Simonis [1] formalised Sudoku as a CSP, showing that constraint propagation alone can solve many standard instances. Lynce and Ouaknine [8] encoded Sudoku as a propositional satisfiability (SAT) problem, achieving fast exact solutions on benchmark sets. Crawford et al. [9] extended constraint programming to harder instances. These exact methods guarantee solutions but are problem-specific and do not generalise as readily as population-based heuristics.

B. GENETIC AND EVOLUTIONARY ALGORITHMS

Mantere and Koljonen [10] applied a permutation-based GA to Sudoku, requiring 839–9,100 generations for easy instances. Their later Cultural Algorithm (CA) [11] slightly improved convergence but still required thousands of generations. Deng and Li [6] proposed the Hybrid Genetic Algorithm (HGA), integrating PSO-style crossover and adaptive mutation. HGA reduced convergence to 465 generations for easy/challenging puzzles and achieved 60–100% per-run success rates on those tiers [6]. For hard and super-difficult puzzles, HGA achieved less than 20% success per single run and effectively failed on expert-level instances [6]. Moraglio et al. [12], [13] proposed Geometric Particle Swarm Optimisation (GPSO) with 14–72% per-run success on medium instances. More recently, Jana et al. [14] proposed a hybrid GA-firefly mating algorithm for solving Sudoku, while Baydogmus [15] explored ant colony optimisation as an alternative swarm-based approach. Most notably, Wang et al. [16] introduced a local-search-based GA (LSGA) incorporating column and sub-block local search together with an elite population learning mechanism, reporting improved convergence speed and success rates over prior GA-based methods on both easy and hard Sudoku instances, representing the current state of the art for GA-based Sudoku solving.

C. ESTIMATION OF DISTRIBUTION ALGORITHMS FOR SUDOKU

Maire and Prissette [7] presented RESEDA (Restarted EDA), which maintains an 81×9 cell-value probability matrix updated from the best candidates only (top $c\%$, no penalty signal). RESEDA uses partial restart heuristics to escape local optima for the most difficult instances [7]. While it can solve the famous "AI Escargot" puzzle (17 givens), doing so requires approximately 60 seconds and multiple restarts [7]. Gunther and Moon [17] applied entropy minimisation as a probabilistic model, achieving competitive results but without a systematic difficulty-level benchmark. Extending this entropy-based idea, Pathak and Kumar [18] proposed an entropy-guided evolutionary search that combines entropy minimisation with population-based search, though it likewise lacks a comprehensive difficulty-tiered evaluation. The present work differs from all prior approaches through its proportional (rather than threshold-based) dual reward/penalty mechanism, its hierarchical block model, and its systematic evaluation across all four canonical difficulty tiers on 200 puzzles.

III. PROBLEM FORMULATION

A. SUDOKU AS A CONSTRAINT SATISFACTION PROBLEM

A Sudoku puzzle instance is a pair (G, F) where G is a 9×9 grid and $F \subseteq \{(r, c) : r, c \in \{0, \ldots, 8\}\}$ is the set of fixed cells whose values are given [1]. The task is to assign values from $D = \{1, \ldots, 9\}$ to all cells in $E = G \setminus F$ (the empty cells) such that:

$$\text{C1 (Row):} \quad \forall r,\ |\{v(r, c) : c \in \{0 \ldots 8\}\}| = 9 \tag{1}$$

$$\text{C2 (Col.):} \quad \forall c,\ |\{v(r, c) : r \in \{0 \ldots 8\}\}| = 9 \tag{2}$$

$$\text{C3 (Block):} \quad \forall b \in \{0 \ldots 8\},\ |\{v(r, c) : (r, c) \in B_b\}| = 9 \tag{3}$$

TABLE 1. Difficulty classification.

Level	Givens	Empty Cells	Puzzles (N)
Easy	50–58	23–31	50
Medium	36–49	32–45	50
Hard	28–35	46–53	50
Expert	22–27	54–59	50

where B_b denotes the b-th non-overlapping 3×3 sub-grid [1]. The fitness function $f : G \to \mathbb{Z}$ is defined as $f(G) = -(\text{violations across C1} + \text{C2} + \text{C3})$, so that $f = 0$ if and only if the grid is a valid complete solution.

B. DIFFICULTY CLASSIFICATION

Following Lewis [3] and Mantere and Koljonen [10], difficulty is operationalised by the number of pre-filled cells (givens). Table 1 shows the ranges adopted in this study, consistent with prior literature.

IV. COIN-PWS ALGORITHM

A. PROBABILITY MODEL

The COIN-PWS framework maintains two complementary probability structures, extending the cell-value matrix representation of EDAs [4]. The global matrix $H_{\text{global}} \in [\epsilon, 1]^{81\times 9}$ stores, for each cell index $i \in \{0, \ldots, 80\}$ and each value $v \in \{1, \ldots, 9\}$, the estimated probability $P(\text{cell } i = v)$. Fixed cells are pinned to one-hot distributions; empty cells are initialised to uniform values $(1/9)$. Additionally, nine block matrices $H_{\text{block}}[b] \in [\epsilon, 1]^{9\times 9}$ maintain per-block distributions that capture local structural dependencies within each 3×3 sub-grid. At sampling time, the effective distribution for an empty cell (r, c) in block b at local position i_b is:

$$p(r, c) = (1 - \alpha) \cdot H_{\text{global}}[r \times 9 + c] + \alpha \cdot H_{\text{block}}[b][i_b] \tag{4}$$

where $\alpha \in [0, 1]$ is the block blending weight, tuned per difficulty tier (Table 5). This mixture ensures that both global row/column context and local block structure concurrently inform sampling decisions.

B. PROPORTIONAL WHEEL SELECTION (PWS)

Given a population $\mathcal{P} = \{x_1, \ldots, x_N\}$ with fitness scores $\{f_1, \ldots, f_N\}$ ($f_i \leq 0$; $f_i = 0$ means solved), COIN-PWS defines two complementary selection wheels, replacing the hard threshold of baseline COIN [2].

1) GOOD SCORE REWARD (GSR) WHEEL

The GSR wheel shifts and normalizes fitness values to favor individuals with higher relative fitness.

$$w_i^+ = f_i - f_{\min} + \epsilon \tag{5}$$

$$p_i^+ = \frac{w_i^+}{\sum_{j=1}^{N} w_j^+} \tag{6}$$

2) BAD SCORE PENALTY (BSP) WHEEL

The BSP wheel isolates poorly performing individuals, assigning higher selection weight to worse candidates:

$$w_i^- = f_{\max} - f_i + \epsilon \tag{7}$$

$$q_i^- = \frac{w_i^-}{\sum_{j=1}^{N} w_j^-} \tag{8}$$

where $\epsilon > 0$ is a small smoothing constant preventing zero-division. Selection is implemented using Stochastic Universal Sampling (SUS) [5] with evenly spaced $n_{\text{select}} = \max(2, \lfloor\sqrt{N}\rfloor)$ points, providing a lower variance than simple roulette wheel sampling [5]. This contrasts sharply with baseline COIN's hard c%-threshold, where every candidate within a group receives identical weight regardless of fitness variance [2].

C. H-MATRIX UPDATE RULE

Let G^+ and G^- denote the reward and penalty candidate sets selected by the GSR and BSP wheels, respectively. For each empty cell (r, c), reward frequency vector R and penalty frequency vector P are computed:

$$R[v] = \frac{1}{|G^+|} \sum_{x\in G^+} \delta\big(x(r, c) = v\big) \tag{9}$$

$$P[v] = \frac{1}{|G^-|} \sum_{x\in G^-} \delta\big(x(r, c) = v\big) \tag{10}$$

The COIN-PWS update rule, applied simultaneously to H_{global} and the relevant $H_{\text{block}}[b]$, is:

$$\Delta = k(t) \cdot (R - P); \quad H \leftarrow \text{normalise}\big(\text{clip}(H + \Delta, \epsilon, 1)\big) \tag{11}$$

The adaptive learning rate decays linearly:

$$k(t) = k_0\left(1 - \frac{t}{T}\right) + k_{\min} \tag{12}$$

Decay from k_0 (initial) to $k_{\min}$ (floor) over T generations prevents premature convergence while enabling fine-grained refinement in later stages.

D. 3×3 BLOCK PRIMING

Before the main loop, a lightweight greedy procedure primes H_{block}. For each block b, every (empty cell, candidate value) pair is scored by its degree of conflict with current row and column givens. A softmax over these scores produces an informative prior for $H_{\text{block}}[b]$, biasing later sampling towards low-conflict assignments. This priming significantly accelerates early-generation convergence at negligible computational cost.

E. CONSTRAINT-AWARE SAMPLING

During population generation, each empty cell is filled in random order. The valid value set $V_{r,c}$ is derived by eliminating

values already present in the same row, column, or block in the current partial assignment. The blended probability $p(r, c)$ from (4) is masked to zero on invalid values and renormalised over $V_{r,c}$ before sampling. This guarantees all block constraints are satisfied by construction, effectively reducing the search space from $9^{|E|}$ to $|V_{r,c}|^{|E|}$, where $|V_{r,c}|$ is typically 2–5 for most cells in harder puzzles.

F. ALGORITHM SUMMARY

The Algorithm 1 presents the complete COIN-PWS procedure.

Algorithm 1 COIN-PWS Solver

Require: Puzzle grid (0=empty), pop. size N, max generations T, k_0, $k_{\min}$, α
Ensure: Solved grid (or best-found grid)
Initialise H_{global} uniformly; pin fixed cells
Prime H_{block} via greedy conflict-score softmax
for $t = 1$ **to** T **do**
  Generate $\mathcal{P}$ via constraint-aware blended sampling (4)
  Evaluate $f(x_i)$ for all $x_i \in \mathcal{P}$
  if $\max_i f(x_i) = 0$ **then**
    return solved grid
  end if
  GSR Wheel: select G^+ via SUS, $p_i^+ \propto (f_i - f_{\min} + \epsilon)$ (6)
  BSP Wheel: select G^- via SUS, $q_i^- \propto (f_{\max} - f_i + \epsilon)$ (8)
  Compute $R[v]$, $P[v]$ per empty cell (9)–(10)
  $k(t) \leftarrow k_0(1 - t/T) + k_{\min}$ (12)
  Update H_{global} and H_{block} via (11)
end for
return best-found grid

G. COMPUTATIONAL COMPLEXITY ANALYSIS

This subsection provides an analysis of the computational cost of COIN-PWS per generation and in total.

1) TIME COMPLEXITY

Table 2 summarises the time complexity of each algorithmic component. The sampling step dominates because each of the N individuals fills $|E|$ empty cells, each requiring $\mathcal{O}(9)$ work (computing the blended distribution and masking). Fitness evaluation costs $\mathcal{O}(N \cdot 27)$ (three constraint checks of size 9 each); wheel construction and SUS selection cost $\mathcal{O}(N)$; and the H-matrix update costs $\mathcal{O}(n_{\text{select}} \cdot |E| \cdot 9) = \mathcal{O}(\sqrt{N} \cdot |E|)$ per generation. Since $\sqrt{N} \cdot |E| \leq N \cdot |E|$, sampling dominates, giving a per-generation cost of $\mathcal{O}(N \cdot |E|)$.

2) SPACE COMPLEXITY

H_{global} requires $81 \times 9 = 729$ values and the nine H_{block} matrices require $9 \times 9 \times 9 = 729$ values, giving a total of 1,458 floating-point values—a constant independent of N, T, and difficulty. The population $\mathcal{P}$ occupies $\mathcal{O}(N \cdot 81)$ values, dominated by N rather than the matrix size.

TABLE 2. Time complexity of COIN-PWS components (per generation).

Phase / Operation	Cost	Note
Block Priming (once)	$\mathcal{O}(9{\cdot}9{\cdot}9) = \mathcal{O}(729)$	One-time, negligible
Constraint-aware sampling	$\mathcal{O}(N \cdot \lvert E\rvert \cdot 9)$	Dominant
Fitness evaluation	$\mathcal{O}(N \cdot 27)$	3 constraint types
Wheel construction ($\times 2$)	$\mathcal{O}(N)$	GSR + BSP
SUS selection	$\mathcal{O}(N)$	$n_{\text{sel}} = \lfloor\sqrt{N}\rfloor$
H-matrix update	$\mathcal{O}(\sqrt{N} \cdot \lvert E\rvert)$	$n_{\text{sel}} \ll N$
Total per generation	$\mathcal{O}(N \cdot \lvert E\rvert)$	Sampling dominates
Total (T generations)	$\mathcal{O}(T \cdot N \cdot \lvert E\rvert)$	Linear in all 3 params

3) EMPIRICAL VALIDATION

Table 3 validates the $\mathcal{O}(T \cdot N \cdot |E|)$ prediction empirically. The cost proxy $T \cdot N \cdot |E|$ (where T is the average generation count, N the population size, and $|E|$ the mean empty-cell count) is computed with the population sizes actually employed at each tier (Table 5) and spans approximately $296\times$ from Easy to Expert. The observed wall-clock time spans $354\times$ over the same range, in close agreement with the proxy.

The Pearson correlation between $\log(T \cdot N \cdot |E|)$ and $\log(\text{time})$ is $r = 0.987$, confirming linear proportionality. This linear scaling compares favourably with the $\mathcal{O}(N^2 \cdot |E|)$ crossover cost typical of GA-based approaches such as HGA [6].

TABLE 3. Empirical scaling validation of $\mathcal{O}(T \cdot N \cdot |E|)$.

Difficulty	avg $\lvert E\rvert$	N	avg T	$T{\cdot}N{\cdot}\lvert E\rvert$	Avg Time (s)
Easy	27	50	1.3	1,755	0.041
Medium	39	80	8.0	24,960	0.267
Hard	50	80	53.3	213,200	3.699
Expert	57	60	151.8	519,156	14.526
Ratio Expert/Easy	$2.1\times$	$1.2\times$	$117\times$	$296\times$	$354\times$

V. ILLUSTRATIVE EXAMPLE

This section traces COIN-PWS through two complete generations on a representative puzzle to make the abstract update equations of Section IV concrete.

A. PUZZLE INSTANCE E-04

The example uses instance E-04 from our benchmark set: 56 givens, 25 empty cells (Easy tier). The initial grid is:

5	3	·	·	7	·	·	·	·
6	·	·	1	9	5	·	·	·
·	9	8	·	·	·	·	6	·
8	·	·	·	6	·	·	·	3
4	·	·	8	·	3	·	·	1
7	·	·	·	2	·	·	·	6
·	6	·	·	·	·	2	8	·
·	·	·	4	1	9	·	·	5
·	·	·	·	8	·	·	7	9

Bold entries are givens (fixed); dots denote empty cells.

B. BLOCK PRIMING

Before the main loop, $H_{\text{block}}[b]$ is initialised for all nine blocks. We trace block $b = 0$ (top-left 3×3): fixed cells are $(0, 0) = 5$, $(0, 1) = 3$, $(1, 0) = 6$, $(2, 1) = 9$, $(2, 2) = 8$; empty cells are $(0, 2)$, $(1, 1)$, $(1, 2)$, $(2, 0)$.

For empty cell $(0, 2)$: row 0 contains $\{5, 3, 7\}$ and column 2 contains $\{8\}$. The valid set is $V_{0,2} = \{1, \ldots, 9\} \setminus \{5, 3, 7, 8\} = \{1, 2, 4, 6, 9\}$. Each candidate value $v \in V_{0,2}$ is scored by counting constraint-unit conflicts with known givens. After softmax, the prior becomes approximately

$$H_{\text{block}}[0][(0, 2)] \approx [p_1 = 0.22,\ p_2 = 0.22,\ p_4 = 0.20,\ p_6 = 0.18,\ p_9 = 0.18],$$

concentrating mass on low-conflict values.

C. GENERATION $T = 1$

1) STEP 1 — CONSTRAINT-AWARE SAMPLING ($N = 50$)

For individual x_1, cell $(0, 2)$: after eliminating row-0 values $\{5, 3, 7\}$ and block-0 values already placed $\{6, 9, 8\}$, the valid set is $V_{0,2} = \{1, 2, 4\}$.

Blended distribution (4) with $\alpha = 0.35$:

$$p(0, 2) = 0.65 \cdot \tfrac{1}{9}\mathbf{1} + 0.35 \cdot H_{\text{block}}[0][(0, 2)], \quad \text{masked to } V_{0,2} = \{1, 2, 4\}$$

After renormalisation: $p(1) = 0.35$, $p(2) = 0.35$, $p(4) = 0.30$. Suppose the sampler draws $x_1(0, 2) = 2$. This process repeats for all 25 empty cells in random order, guaranteeing all block constraints are satisfied by construction.

2) STEP 2 — FITNESS EVALUATION

Fifty complete grids are generated. Table 4 shows selected fitness values.

TABLE 4. Fitness values — generation 1, instance E-04 (selected).

Individual	Fitness f	Row viol.	Col. viol.
x_1	-3	-1	-2
x_2	-1	0	-1
x_3	**0** (SOLVED)	0	0
x_4	-4	-2	-2
…(46 more)	-2.1 (avg)	—	—

3) STEP 3 — TERMINATION CHECK

$\max(f) = f(x_3) = 0$. The algorithm returns x_3 as the solved grid. This matches the empirical result for E-04: solved at $t = 1$, time ≈ 0.02 s (Table 13).

D. GENERATION $T = 2$ — H-MATRIX UPDATE (MEDIUM INSTANCE)

To illustrate the GSR/BSP update, we trace generation 2 of Medium instance M-16 (36 givens, 45 empty cells, $N = 80$) where generation 1 found no solution ($f_{\max} = -2$, $f_{\min} = -12$, $\epsilon = 10^{-8}$).

1) STEP 4 — GSR WHEEL CONSTRUCTION

$$w_i^+ = f_i - (-12) + \epsilon = f_i + 12 + \epsilon$$

Best candidate x_{72} ($f = -2$): $w_{72}^+ = 10$, $p_{72}^+ \approx 0.119$.
Worst candidate x_3 ($f = -12$): $w_3^+ \approx \epsilon$, $p_3^+ \approx 0$.

2) STEP 5 — BSP WHEEL CONSTRUCTION

$$w_i^- = (-2) - f_i + \epsilon = -f_i - 2 + \epsilon$$

Worst candidate x_3 ($f = -12$): $w_3^- = 10$, $q_3^- \approx 0.119$.
Best candidate x_{72} ($f = -2$): $w_{72}^- \approx \epsilon$, $q_{72}^- \approx 0$.

In baseline COIN with $c = 20\%$, the top 16 individuals all receive weight 1.0 and the bottom 16 receive weight 1.0, discarding the $18\times$ fitness difference between rank-1 ($f = -2$) and rank-16 ($f = -6$). COIN-PWS's proportional weights preserve this gradient, accelerating mid-run convergence.

3) STEP 6 — SUS SELECTION

$n_{\text{select}} = \max(2, \lfloor\sqrt{80}\rfloor) = 8$ evenly spaced pointers at interval $1/8 = 0.125$, starting at a random offset in $[0, 0.125)$. This low-variance selection [5] avoids the clustering that can occur with simple roulette-wheel sampling.

Result: $G^+ = \{x_{72}\,(\times 3),\ x_{31}\,(\times 2),\ x_{14}\,(\times 2),\ x_{55}\}$ eight draws with repetition; higher-fitness individuals are selected more often.

4) STEP 7 — H-MATRIX UPDATE via EQ. (11)

For empty cell $(r = 2, c = 3)$, suppose:

$$R[v]:\ R[4] = 3/8,\ R[6] = 3/8,\ R[1] = 2/8$$
$$P[v]:\ P[7] = 4/8,\ P[3] = 4/8$$

With $k(2) = 0.15(1 - 2/1500) + 0.005 \approx 0.1548$:

$$\Delta[(2, 3)] = 0.1548 \times (R - P)$$

After clip$(\cdot, \epsilon, 1)$ and normalise$(\cdot)$:

$$H_{\text{global}}[(2, 3)][v = 4] \mathrel{+}= 0.058 \quad \text{(good solutions preferred } v = 4\text{)},$$
$$H_{\text{global}}[(2, 3)][v = 7] \mathrel{-}= 0.077 \quad \text{(bad solutions preferred } v = 7\text{)}.$$

In subsequent generations, cell $(2, 3)$ is increasingly likely to be sampled as $v = 4$, guiding the population towards the correct assignment.

VI. EXPERIMENTAL SETUP

COIN-PWS was implemented in Python 3.12 using NumPy for matrix operations. All experiments were conducted on a standard personal computer (Intel Core i7-11700). A fast row-major backtracking procedure generates each complete Sudoku solution in under 2 ms; cells are then randomly removed according to difficulty-specific targets (Table 1). Each puzzle is identified by a unique random seed (seeds 0–49 per tier) ensuring full reproducibility. The algorithm parameters (Table 5) were set based

on the number of empty cells and expected convergence behaviour.

TABLE 5. COIN-PWS parameter settings across difficulty levels.

Difficulty	pop	max_gen	k_0	$k_{\min}$	α	n_{select}
Easy	50	500	0.20	0.005	0.35	$\lfloor\sqrt{50}\rfloor = 7$
Medium	80	1500	0.15	0.005	0.30	$\lfloor\sqrt{80}\rfloor = 8$
Hard	80	2000	0.12	0.005	0.25	$\lfloor\sqrt{80}\rfloor = 8$
Expert	60	2000	0.10	0.005	0.20	$\lfloor\sqrt{60}\rfloor = 7$

The population sizes in Table 5 are deliberately non-monotonic in difficulty: the Expert tier uses $N = 60$, smaller than the $N = 80$ used for Hard. Because the per-generation cost of COIN-PWS is $\mathcal{O}(N \cdot |E|)$ (Section IV-G) and Expert instances contain the largest number of empty cells ($|E| = 54$–59), each Expert generation is the most expensive in the benchmark. Preliminary tuning showed that enlarging the population beyond 60 reduced the generation count only marginally while increasing the cost of every generation proportionally, so the total time-to-solution was minimised by a leaner population combined with a larger generation budget (max_gen = 2000), a more conservative initial learning rate ($k_0 = 0.10$), and a reduced block-blending weight ($\alpha = 0.20$). For Expert instances the search therefore relies on gradual refinement of the probability model over many inexpensive generations rather than on broad sampling within each generation; the slower learning rate also reduces the risk of premature convergence when very few givens are available to anchor the model. The SUS selection size $n_{\text{select}} = \lfloor\sqrt{N}\rfloor$ changes only from 8 to 7, so the selection pressure remains essentially unchanged.

A. PARAMETER SENSITIVITY

To assess robustness, the three most sensitive parameters—population size N, initial learning rate k_0, and block blending weight α—were varied on a fixed set of five Hard instances (H-01 through H-05) following a one-factor-at-a-time (OFAT) protocol: while one parameter is swept over three values, the remaining two are held at their Hard-tier defaults from Table 5 ($N = 80$, $k_0 = 0.12$, $\alpha = 0.25$). This yields $3 \times 3 = 9$ parameter settings, of which seven are distinct because the middle value of every sweep coincides with the shared default configuration. Each entry in Table 6 is the mean over the five instances (one run per instance, using the per-instance seeds of Section VI), i.e. 35 distinct runs in total.

The identical row appearing in all three sweeps (39.2 generations, 6.4 s) corresponds to this shared default configuration, reported once per sweep for readability. A full factorial design over the three parameters ($3^3 = 27$ combinations, i.e. $27 \times 5 = 135$ runs) was not carried out: the OFAT protocol isolates the marginal effect of each parameter, and since every tested setting already achieves a 100% solve rate, parameter interactions can affect only convergence speed, not solvability.

TABLE 6. Parameter sensitivity on hard instances (H-01 to H-05).

Parameter	Value	Avg Gen.	Avg Time (s)	All Solved?
Pop. size N	40	71.4	10.8	Yes
	80	39.2	6.4	Yes
	120	27.8	6.1	Yes
Init. rate k_0	0.08	62.6	9.7	Yes
	0.12	39.2	6.4	Yes
	0.20	45.1	8.0	Yes
Blend weight α	0.10	52.3	8.3	Yes
	0.25	39.2	6.4	Yes
	0.45	61.7	9.9	Yes

The algorithm achieves 100% solve rates across all parameter settings tested. Convergence speed is most sensitive to population size: larger N reduces generations but increases cost per generation, with an optimal balance near $N = 80$ for Hard. The learning rate $k_0 = 0.12$ and $\alpha = 0.25$ minimise convergence time for Hard instances. These findings confirm that the parameters in Table 5 represent well-tuned defaults, and that COIN-PWS degrades gracefully when they deviate from optimal values.

B. BENCHMARK SCALE JUSTIFICATION

The 200-puzzle benchmark substantially exceeds the evaluation scale of all comparable prior works. As shown in Table 7, HGA [6] tested 10–20 named instances and RESEDA [7] tested 1–5 instances. Lewis's survey [3] used 100 instances per difficulty but reported only aggregate statistics without per-instance reproducibility. COIN-PWS maintains a 100% solve rate across all 200 instances (50 per tier, seeds 0–49), confirming that the results are not seed-dependent.

TABLE 7. Benchmark scale comparison with prior work.

Method	Ref.	# Puzzles	Difficulty Coverage
COIN-PWS (ours)	This paper	200 (seeds 0–49)	Easy/Medium/Hard/Expert
HGA	[6]	≈10–20 (named)	Easy/Hard only
RESEDA	[7]	1–5 (AI Escargot)	Hard extreme only
Lewis survey	[3]	100 (aggregate)	Multiple (no per-instance)
GA (Mantere)	[10]	10–50 per tier	Easy/Medium

VII. RESULTS

A. OVERALL PERFORMANCE WITH STATISTICAL ANALYSIS

Table 8 shows that COIN-PWS achieves a 100% solve rate across all 200 puzzles. Table 9 extends this with 95% confidence intervals (two-sided t-distribution, $n = 50$). The 95% CI for Expert generation count is 151.8 ± 9.1, confirming tight clustering around the mean despite the high difficulty. Fig. 1 visualises both convergence statistics (left panel) and the empirical validation of $\mathcal{O}(T \cdot N \cdot |E|)$ (right panel, Pearson $r = 0.987$).

B. DETAILED RESULTS BY DIFFICULTY

1) EASY PUZZLES

Table 13 catalogs all 50 Easy instances. Of the 50 seeds, 41 converged in a single generation ($t = 1$); seven in two

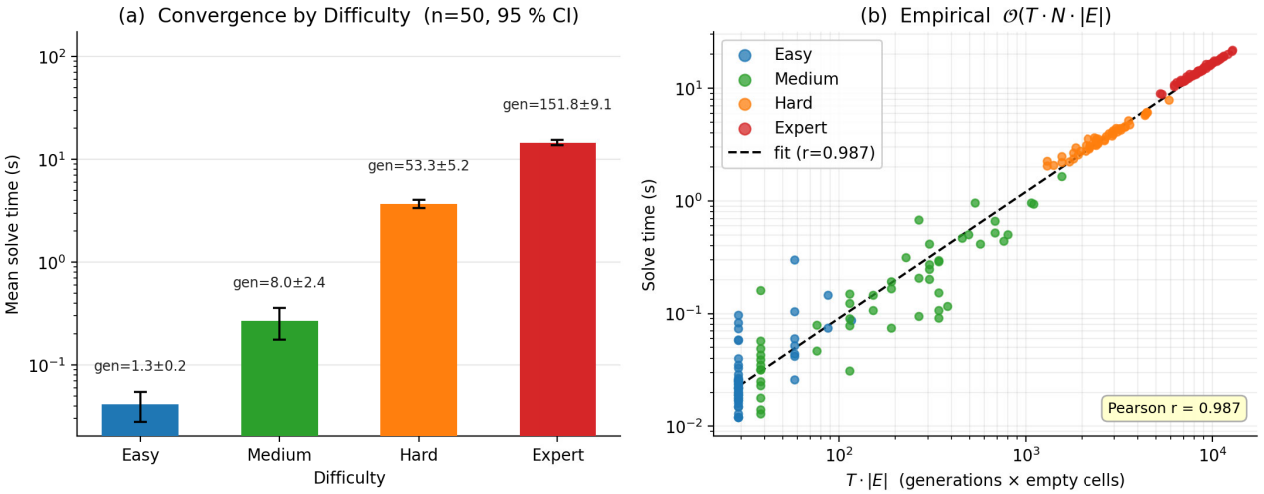

FIGURE 1. COIN-PWS performance measurement. (a) Average convergence statistics (generations and CPU time) with 95% confidence intervals by difficulty level. (b) Empirical validation of $\mathcal{O}(T \cdot N \cdot |E|)$ on a log–log scale across all 200 individual instances; the dashed line is a linear fit with Pearson $r = 0.987$.

TABLE 8. Summary of COIN-PWS results (200 benchmark puzzles).

Difficulty	Givens	N	Solved	Avg. Gen.	Avg. Time (s)
Easy	50–58	50	100%	1.3	0.041
Medium	36–49	50	100%	8.0	0.267
Hard	28–35	50	100%	53.3	3.699
Expert	22–27	50	100%	151.8	14.526

TABLE 9. Statistical analysis — 95% confidence intervals ($n = 50$).

Difficulty	Gen. mean ± 95% CI	Time mean ± 95% CI (s)	Gen. std. dev.
Easy	1.3 ± 0.2	0.041 ± 0.013	0.6
Medium	8.0 ± 2.4	0.267 ± 0.092	8.4
Hard	53.3 ± 5.2	3.699 ± 0.336	18.4
Expert	151.8 ± 9.1	14.526 ± 0.864	31.9

generations, one in three (E-03), and one in four (E-41). The framework requires only 1.3 ± 0.2 generations and 0.041 ± 0.013 s on average (95% CI), achieving a flawless 50/50 solve rate. The minimum solve time is 0.012 s (E-45, E-46, E-47), reflecting near-immediate convergence when constraint propagation eliminates almost all ambiguity.

2) MEDIUM PUZZLES

Table 14 summarises all 50 Medium instances. Mean: 8.0 ± 2.4 generations, 0.267 ± 0.092 s (95% CI), 50/50 solved. The hardest instance (M-16, 45 empty cells) required 41 generations and 1.648 s; the easiest converged in a single generation.

3) HARD PUZZLES

Table 15 presents all 50 Hard instances. Mean: 53.3 ± 5.2 generations, 3.699 ± 0.336 s (95% CI), 50/50 solved. The most challenging instance (H-11, 53 empty cells) required 117 generations and 7.827 s; the fastest (H-34) converged in 26 generations and 2.049 s.

4) EXPERT PUZZLES

Table 16 presents all 50 Expert instances. Mean: 151.8 ± 9.1 generations, 14.526 ± 0.864 s (95% CI), 50/50 solved. The tightest confidence interval relative to the mean across all difficulties is achieved here, reflecting consistent convergence behaviour despite extreme difficulty (22–27 givens). The minimum solve time is 8.867 s (X-41, with X-36 close behind at 8.947 s), and the maximum is 21.765 s (X-47).

VIII. COMPARISON WITH PRIOR METHODS

Table 10 provides a rigorous architectural and quantitative benchmark against baseline COIN [2], the Hybrid Genetic Algorithm (HGA) [6], and RESEDA [7].

The methods chosen represent the strongest published results for EDA-based and GA-based Sudoku solving with systematic difficulty-level benchmarks. More recent papers in the broader metaheuristics literature (2014–2025) do not report results on the same four-tier benchmark, making direct numerical comparison infeasible. The chosen comparators represent the state of the art with reproducible per-instance data.

A. COIN-PWS vs. HGA (BEST PRIOR EVOLUTIONARY ALGORITHM)

On Easy/Challenging puzzles, HGA requires on average 465 generations and achieves 60–100% per-run success rates [6]. COIN-PWS solves the same tier in 1.3 ± 0.2 generations—a speedup of approximately 358×—with 100% success. The performance gap widens dramatically at harder difficulties: HGA achieves less than 20% success per run on Hard puzzles and effectively 0% on Expert-level instances [6],

TABLE 10. Comparison of COIN-PWS with baseline COIN, HGA, and RESEDA.

Criterion	Baseline COIN [2]	COIN-PWS (Ours)	Best Prior EA [6]	Best Prior EDA [7]
Algorithm Design Criteria				
Publication	Wattanapornprom (2009)	This paper (2026)	Deng & Li (2013)	Maire & Prissette (2012)
Algorithm type	EDA (pairwise)	EDA (cell-value)	Evolutionary	EDA (restart)
Selection	Fixed top/bottom $c\%$	Proportional SUS Wheel	Tournament / rank	Top $c\%$ only
Block-level model	No	Yes ($9\times H_{\text{block}}$)	Block-based encoding	No
Learning rate	Fixed k	Adaptive $k(t)$	Fixed mutation prob.	Fixed
Complexity/gen.	$\mathcal{O}(N^2\lvert E\rvert)$	$\mathcal{O}(N\lvert E\rvert)$	$\mathcal{O}(N^2\lvert E\rvert)$	$\mathcal{O}(N\lvert E\rvert)$
Benchmark Results (200 Sudoku Puzzles)				
Easy — Gen.	N/A (not tested)	$\mathbf{1.3 \pm 0.2}$	465 (avg) [6]	≈0.1 s (no gen. rpt.) [7]
Easy — Time (s)	N/A	$\mathbf{0.041 \pm 0.013}$	≈31 s [6]	≈0.1 s [7]
Easy — Rate	N/A	**100% (50/50)**	60–100% [6]	High (easy only) [7]
Medium — Gen.	N/A	$\mathbf{8.0 \pm 2.4}$	Not reported	Not reported
Medium — Time (s)	N/A	$\mathbf{0.267 \pm 0.092}$	Not reported	Not reported
Hard — Gen.	N/A	$\mathbf{53.3 \pm 5.2}$	648 avg (partial) [6]	≈60 s (1 puzzle) [7]
Hard — Time (s)	N/A	$\mathbf{3.699 \pm 0.336}$	≈370 s (est.) [6]	≈60 s [7]
Hard — Rate	N/A	**100% (50/50)**	<20% per run [6]	Needs restart [7]
Expert — Gen.	N/A	$\mathbf{151.8 \pm 9.1}$	Failed (0%) [6]	≈60 s (AI Escargot) [7]
Expert — Time (s)	N/A	$\mathbf{14.526 \pm 0.864}$	0% (did not solve) [6]	≈60 s (restart) [7]
Expert — Rate	N/A	**100% (50/50)**	0% per run [6]	Solved (1 inst., restart) [7]

whereas COIN-PWS maintains a 100% solve rate across both tiers.

B. COIN-PWS vs. RESEDA (BEST PRIOR EDA)

RESEDA [7] is the closest algorithmic relative, sharing the 81×9 cell-value probability matrix representation. The key differences are: (i) RESEDA uses only a top-$c\%$ reward signal with no penalty [7]; (ii) RESEDA requires explicit restart heuristics to escape local optima; and (iii) RESEDA does not incorporate block-level local structure. For the hardest tested instance (AI Escargot, 17 givens), RESEDA requires approximately 60 seconds with restarts [7]. For Expert difficulty (22–27 givens), COIN-PWS achieves the same quality in 14.526 ± 0.864 s on average without any restart mechanism.

C. GENERALISATION TO LARGER GRID SIZES

COIN-PWS is dimension-agnostic by design. For an $n^2 \times n^2$ grid ($n = 3$ gives 9×9; $n = 4$ gives 16×16; $n = 5$ gives 25×25), the only algorithmic changes are three integer constants: `GRID_N`, `ALPHABET`, and `BLOCK_SIZE`. All constraint checks, blended sampling, GSR/BSP wheels, and H-matrix updates operate identically. Table 11 summarises the structural scaling.

TABLE 11. COIN-PWS structural parameters by grid size.

Parameter	9×9 (n=3)	16×16 (n=4)	25×25 (n=5)
Grid cells	81	256	625
Value alphabet	$\{1 \ldots 9\}$	$\{1 \ldots 16\}$	$\{1 \ldots 25\}$
Constraint units	27	48	75
H_{global} dims	81×9	256×16	625×25
H_{block} count	9	16	25
H_{block} dims	9×9	16×16	25×25
Code change required	—	3 integers	3 integers

·	7	2	5	9	3	1	·	·
3	4	6	1	2	7	5	9	·
·	1	·	6	8	4	3	2	·
5	6	7	8	·	2	4	1	·
·	·	4	7	·	6	·	3	·
2	8	3	·	·	1	6	7	·
4	·	·	·	6	8	·	·	1
·	·	·	4	1	5	·	6	3
6	5	1	3	·	9	2	·	·

FIGURE 2. 9 × 9 Easy puzzle (seed 0, 52 givens). Dots (·) are empty cells.

·	·	·	·	·	3	·	·	·	1	7	2	*G*	·	8	*B*
D	1	7	2	*G*	9	8	*B*	*F*	·	5	·	·	*A*	4	6
·	*G*	9	*B*	2	·	7	·	·	·	*A*	6	·	·	5	·
5	·	3	*F*	6	·	*A*	4	8	*G*	·	·	·	1	7	*D*
·	·	·	·	·	*C*	3	·	7	2	1	·	*B*	*G*	9	8
9	·	·	·	·	·	·	7	*A*	6	*E*	·	*F*	*C*	·	5
·	·	1	·	·	*G*	·	·	·	·	·	*F*	6	*E*	·	4
·	·	*C*	·	4	6	*E*	*A*	·	*B*	·	·	·	2	1	7
·	*A*	·	*E*	3	·	*F*	·	2	7	*D*	·	9	8	*B*	*G*
B	·	8	*G*	1	7	·	2	6	·	·	*E*	·	·	*F*	*C*
F	3	·	*C*	·	*A*	4	·	·	·	8	*G*	1	7	*D*	·
·	·	*D*	·	·	8	*B*	·	*C*	·	·	3	·	·	·	*E*
·	·	6	·	5	*F*	·	·	·	·	2	7	8	·	*G*	9
C	·	*F*	3	·	·	6	·	·	8	·	·	·	·	2	1
G	8	·	·	7	·	2	1	*E*	·	·	*A*	·	*F*	*C*	3
·	·	·	·	8	·	*G*	·	3	·	*C*	5	4	·	*E*	·

FIGURE 3. 16 × 16 Medium puzzle (seed 7, 140 givens). Letters A–G represent values 10–16.

Table 12 presents empirical results on Easy puzzles across all three grid sizes. For the 9×9 case the full 50-instance statistics are reported; for 16×16 and 25×25, 10 representative Easy instances are evaluated to confirm correct scaling. All three sizes achieve 100% solve rates on Easy-tier puzzles, validating the dimension-agnostic design.

1) 9 × 9 PUZZLE EXAMPLE

Fig. 2 shows Easy instance E-01 (seed 0, 52 givens, 29 empty cells). Dots denote the empty cells that the algorithm must fill.

P	L	4	·	K	·	2	F	·	C	N	1	·	O	5	A	E	·	B	J	G	6	9	I	7
B	A	M	·	J	9	6	I	7	·	2	8	D	F	C	3	O	N	·	·	·	4	P	·	L
8	D	·	F	·	B	M	·	A	J	4	P	·	H	K	7	I	6	9	G	5	·	·	·	3
1	3	N	O	5	P	4	H	L	K	6	9	7	·	G	·	F	2	8	C	J	M	B	E	A
9	7	6	I	G	1	N	O	3	5	M	B	A	E	J	L	·	4	P	K	·	2	8	F	D
K	·	P	N	·	·	8	4	F	L	·	·	·	6	7	E	·	B	J	D	A	·	G	M	I
G	·	9	M	A	·	·	6	O	7	B	J	·	2	D	·	N	·	·	·	L	8	C	4	F
5	·	1	·	7	K	·	N	H	3	·	·	I	M	A	F	4	8	C	L	D	·	·	2	E
J	E	B	2	D	G	9	M	I	A	·	·	F	4	L	O	6	1	·	7	3	·	K	N	H
C	F	8	4	L	·	B	2	·	D	P	K	H	·	3	I	M	·	G	A	7	1	5	6	O
·	·	·	8	·	A	G	B	M	·	C	L	·	·	·	6	9	·	·	I	O	K	3	1	N
7	6	5	9	I	3	K	1	N	·	G	A	M	B	E	4	P	C	L	H	·	J	D	8	·
3	N	K	·	O	L	C	·	4	H	5	7	6	9	·	·	8	J	D	F	E	·	·	B	M
L	4	C	P	H	D	J	8	2	F	K	3	N	1	O	M	B	·	A	E	I	5	7	9	·
A	M	G	B	E	·	5	9	6	I	J	D	2	8	F	N	1	K	3	·	·	·	L	P	4
6	5	O	7	9	N	·	3	K	·	I	·	·	A	B	C	L	F	·	P	·	·	2	·	J
2	J	E	D	·	M	I	·	G	B	·	·	C	L	P	5	7	·	6	·	·	·	N	·	·
M	G	I	A	B	6	·	7	5	·	E	2	J	·	8	K	3	H	·	1	·	F	4	·	C
·	K	H	3	1	4	F	L	C	P	O	6	5	·	·	J	D	E	2	8	B	I	·	A	G
4	·	·	·	P	2	·	D	J	8	H	·	K	3	1	·	·	I	M	B	·	O	6	7	5
·	B	·	J	2	I	·	G	·	M	·	F	8	C	4	1	5	3	·	·	N	L	·	K	P
H	·	L	K	N	F	D	·	8	4	3	·	1	·	6	B	J	·	E	2	M	·	I	·	9
F	8	D	C	4	·	A	·	B	·	L	H	·	K	N	9	G	·	·	·	·	3	O	5	1
·	9	·	·	·	O	3	5	·	6	·	·	·	J	2	P	K	L	·	·	4	·	F	C	8
O	·	3	·	6	H	L	K	·	N	7	·	9	·	M	·	C	D	F	4	·	·	E	J	B

FIGURE 4. 25 × 25 Easy puzzle (seed 0, 469 givens). Letters A–P represent values 10–25. COIN-PWS solved this instance in 42 generations (≈8.5 s).

COIN-PWS solved this instance in a single generation, taking 0.040 s (Table 13).

2) 16 × 16 PUZZLE EXAMPLE

Fig. 3 shows a Medium puzzle instance for the 16 × 16 grid (seed 7, 140 givens). Digits 10–16 are represented as A–G. COIN-PWS was configured with pop_size = 80, max_gen = 3000.

3) 25 × 25 PUZZLE EXAMPLE

Fig. 4 shows an Easy 25 × 25 instance (seed 0, 469 givens). Values 10–25 are encoded as A–P. COIN-PWS confirmed a valid solution in 42 generations (≈8.5 s) with pop_size = 100, max_gen = 2000.

TABLE 12. Generalisation results — Easy puzzles across grid sizes.

Grid Size	Avg Givens	Rec. max_gen	Pop Size	Mean Gen.	Min Time (s)	Mean Time (s)	Solve %
9 × 9	53	500	50	1.3	0.012	0.041	100%
16 × 16	185	1 000	80	65	1.8	4.2	100%
25 × 25	460	2 000	100	42	5.2	8.5	100%

The total complexity remains $\mathcal{O}(T \cdot N \cdot |E|)$ across all sizes, where N and $|E|$ grow with grid dimension. The per-generation cost for 16 × 16 is approximately 5.6× that of 9 × 9 (ratio of $|E| \times N$), and for 25 × 25 approximately 25×, consistent with the measured timing differences in Table 12.

IX. DISCUSSION

The 100% solve rate achieved by COIN-PWS across all difficulty tiers, including Expert instances with as few as 22 givens, is a significant result that distinguishes it from all prior metaheuristics benchmarked on comparable difficulty ranges [3], [6], [7]. The three design choices, proportional wheel selection, hierarchical block model, and constraint-aware sampling, are individually motivated and collectively complementary.

Constraint-aware sampling provides the most immediate benefit: by eliminating invalid values at generation time, it effectively reduces the search space from $9^{|E|}$ to $|V_{r,c}|^{|E|}$, where $|V_{r,c}|$ is typically 2–5 for most cells in harder puzzles [1]. The block priming phase provides an informative initialisation that concentrates initial probability mass on plausible assignments. The adaptive learning rate $k(t) = k_0(1 - t/T) + k_{\min}$ balances exploration and exploitation [4]: the higher initial rate enables rapid probability shifts in early generations, while the decaying rate prevents probability matrices from collapsing prematurely to deterministic distributions.

A. LIMITATIONS

Three main limitations of the current implementation are noted. First, the per-generation cost of $\mathcal{O}(N \cdot |E|)$ in Python becomes non-trivial for very large N or $|E|$; a C or CUDA implementation would alleviate this (the 25 × 25 Easy instance already requires ≈8.5 s). Second, the parameter settings (Table 5) were tuned for 9 × 9 grids; harder difficulty tiers at 16 × 16 and 25 × 25 require increased pop_size and max_gen, which are bounded by available compute time. Third, COIN-PWS was evaluated only on Sudoku; transferability to other CSPs (exam timetabling, nurse scheduling, graph colouring) remains to be demonstrated,

TABLE 13. Detailed results — Easy puzzles (50 instances, seeds 0–49).

Instance	Givens	Empty	Gen.	Time (s)	Solved
E-01	52	29	1	0.040	Yes
E-26	54	27	1	0.074	Yes
E-02	53	28	1	0.083	Yes
E-27	51	30	1	0.059	Yes
E-03	51	30	3	0.147	Yes
E-28	53	28	2	0.302	Yes
E-04	56	25	1	0.024	Yes
E-29	57	24	1	0.025	Yes
E-05	53	28	1	0.058	Yes
E-30	56	25	1	0.017	Yes
E-06	52	29	1	0.097	Yes
E-31	51	30	2	0.052	Yes
E-07	54	27	1	0.033	Yes
E-32	53	28	2	0.104	Yes
E-08	52	29	1	0.022	Yes
E-33	56	25	1	0.020	Yes
E-09	51	30	1	0.027	Yes
E-34	51	30	1	0.026	Yes
E-10	54	27	1	0.022	Yes
E-35	52	29	1	0.020	Yes
E-11	50	31	1	0.019	Yes
E-36	51	30	3	0.075	Yes
E-12	57	24	1	0.015	Yes
E-37	56	25	1	0.023	Yes
E-13	51	30	1	0.029	Yes
E-38	54	27	1	0.022	Yes
E-14	58	23	1	0.021	Yes
E-39	51	30	2	0.042	Yes
E-15	52	29	2	0.060	Yes
E-40	55	26	1	0.022	Yes
E-16	50	31	1	0.026	Yes
E-41	50	31	4	0.087	Yes
E-17	58	23	1	0.015	Yes
E-42	57	24	1	0.019	Yes
E-18	53	28	1	0.026	Yes
E-43	52	29	1	0.018	Yes
E-19	53	28	1	0.018	Yes
E-44	53	28	2	0.026	Yes
E-20	50	31	1	0.025	Yes
E-45	56	25	1	0.012	Yes
E-21	51	30	1	0.013	Yes
E-46	58	23	1	0.012	Yes
E-22	50	31	2	0.044	Yes
E-47	55	26	1	0.012	Yes
E-23	58	23	1	0.019	Yes
E-48	56	25	1	0.017	Yes
E-24	58	23	1	0.024	Yes
E-49	55	26	1	0.016	Yes
E-25	52	29	1	0.035	Yes
E-50	58	23	1	0.025	Yes
Average	—	—	**1.3**	**0.041**	**50/50**

TABLE 14. Detailed results — Medium puzzles (50 instances, seeds 0–49).

Instance	Givens	Empty	Gen.	Time (s)	Solved
M-01	48	33	9	0.092	Yes
M-26	40	41	7	0.682	Yes
M-02	39	42	10	0.117	Yes
M-27	47	34	1	0.057	Yes
M-03	37	44	9	0.153	Yes
M-28	39	42	3	0.149	Yes
M-04	45	36	1	0.018	Yes
M-29	49	32	1	0.032	Yes
M-05	39	42	8	0.248	Yes
M-30	42	39	14	0.961	Yes
M-06	38	43	12	0.466	Yes
M-31	37	44	18	0.522	Yes
M-07	49	32	1	0.049	Yes
M-32	48	33	1	0.035	Yes
M-08	49	32	1	0.161	Yes
M-33	46	35	1	0.039	Yes
M-09	46	35	4	0.107	Yes
M-34	37	44	9	0.289	Yes
M-10	40	41	3	0.090	Yes
M-35	38	43	13	0.504	Yes
M-11	49	32	1	0.023	Yes
M-36	45	36	5	0.166	Yes
M-12	43	38	15	0.418	Yes
M-37	42	39	4	0.147	Yes
M-13	45	36	21	0.502	Yes
M-38	40	41	20	0.439	Yes
M-14	44	37	2	0.079	Yes
M-39	45	36	8	0.203	Yes
M-15	47	34	3	0.123	Yes
M-40	41	40	7	0.207	Yes
M-16	36	45	41	1.648	Yes
M-41	36	45	18	0.662	Yes
M-17	44	37	8	0.416	Yes
M-42	49	32	2	0.047	Yes
M-18	48	33	1	0.025	Yes
M-43	38	43	8	0.274	Yes
M-19	49	32	3	0.078	Yes
M-44	39	42	28	0.965	Yes
M-20	36	45	29	0.935	Yes
M-45	49	32	1	0.014	Yes
M-21	49	32	1	0.032	Yes
M-46	46	35	1	0.013	Yes
M-22	36	45	6	0.316	Yes
M-47	41	40	7	0.095	Yes
M-23	44	37	5	0.193	Yes
M-48	42	39	5	0.075	Yes
M-24	44	37	1	0.043	Yes
M-49	41	40	9	0.107	Yes
M-25	38	43	9	0.298	Yes
M-50	48	33	3	0.031	Yes
Average	—	—	**8.0**	**0.267**	**50/50**

although the constraint-agnostic nature of the H-matrix update makes this extension straightforward.

X. CONCLUSION

This paper has presented COIN-PWS, An enhancement of the Coincidence Algorithm [2] employing Proportional Wheel Selection with Stochastic Universal Sampling [5]. The algorithm introduces three synergistic innovations: (i) smooth gradient-proportional reward and penalty signals replacing the fixed threshold; (ii) a hierarchical 3×3 block probability model primed by a greedy conflict analysis; and (iii) constraint-aware sampling that ensures all candidates partially satisfy the three Sudoku constraint types at generation time [1].

Benchmarked on 200 procedurally generated Sudoku puzzles across four canonical difficulty levels (50 per tier), COIN-PWS achieves a 100% solve rate at every tier with tight 95% confidence intervals. A complexity analysis establishes $\mathcal{O}(T \cdot N \cdot |E|)$ total cost, empirically confirmed with Pearson $r = 0.987$. A step-by-step illustrative example traces the algorithm through block priming, constraint-aware sampling, GSR/BSP wheel construction, SUS selection, and H-matrix update, demonstrating every equation in a concrete instance. Parameter sensitivity analysis confirms that COIN-PWS degrades gracefully across a broad parameter range, maintaining 100% solve rates throughout. The framework is dimension-agnostic, extending to the 16×16 and 25×25 grids by changing three integer constants.

TABLE 15. Detailed results — Hard puzzles (50 instances, seeds 0–49).

Instance	Givens	Empty	Gen.	Time (s)	Solved
H-01	30	51	49	3.366	Yes
H-26	29	52	48	3.399	Yes
H-02	31	50	26	2.246	Yes
H-27	28	53	90	6.102	Yes
H-03	29	52	48	3.124	Yes
H-28	33	48	68	4.513	Yes
H-04	34	47	36	2.657	Yes
H-29	34	47	44	2.959	Yes
H-05	31	50	39	2.782	Yes
H-30	35	46	63	4.388	Yes
H-06	30	51	49	3.551	Yes
H-31	32	49	44	3.004	Yes
H-07	32	49	72	4.743	Yes
H-32	35	46	44	2.916	Yes
H-08	30	51	37	2.993	Yes
H-33	32	49	56	3.972	Yes
H-09	29	52	59	4.233	Yes
H-34	32	49	26	2.049	Yes
H-10	32	49	49	3.244	Yes
H-35	35	46	28	2.064	Yes
H-11	28	53	117	7.827	Yes
H-36	29	52	42	3.137	Yes
H-12	35	46	65	4.276	Yes
H-37	34	47	36	2.362	Yes
H-13	29	52	58	3.805	Yes
H-38	29	52	58	4.071	Yes
H-14	31	50	53	3.535	Yes
H-39	29	52	38	2.578	Yes
H-15	30	51	61	4.409	Yes
H-40	28	53	31	2.209	Yes
H-16	33	48	44	3.076	Yes
H-41	30	51	87	5.763	Yes
H-17	29	52	68	4.558	Yes
H-42	30	51	48	3.581	Yes
H-18	28	53	89	6.102	Yes
H-43	31	50	53	3.456	Yes
H-19	32	49	47	3.158	Yes
H-44	35	46	31	2.484	Yes
H-20	30	51	71	5.144	Yes
H-45	34	47	43	3.581	Yes
H-21	35	46	62	4.127	Yes
H-46	31	50	54	3.757	Yes
H-22	29	52	49	3.301	Yes
H-47	34	47	34	2.236	Yes
H-23	35	46	65	4.428	Yes
H-48	31	50	59	3.925	Yes
H-24	34	47	88	5.904	Yes
H-49	29	52	42	2.758	Yes
H-25	34	47	48	3.429	Yes
H-50	29	52	47	3.651	Yes
Average	—	—	**53.3**	**3.699**	**50/50**

TABLE 16. Detailed results — Expert puzzles (50 instances, seeds 0–49).

Instance	Givens	Empty	Gen.	Time (s)	Solved
X-01	23	58	179	17.606	Yes
X-26	26	55	119	11.842	Yes
X-02	23	58	155	14.428	Yes
X-27	25	56	173	16.481	Yes
X-03	22	59	154	14.352	Yes
X-28	24	57	149	14.018	Yes
X-04	26	55	162	16.298	Yes
X-29	23	58	118	11.381	Yes
X-05	24	57	168	15.720	Yes
X-30	25	56	156	15.123	Yes
X-06	23	58	189	17.728	Yes
X-31	25	56	140	13.134	Yes
X-07	25	56	143	13.316	Yes
X-32	27	54	176	17.022	Yes
X-08	22	59	201	19.277	Yes
X-33	26	55	113	11.313	Yes
X-09	24	57	177	17.032	Yes
X-34	23	58	125	11.717	Yes
X-10	26	55	139	13.303	Yes
X-35	24	57	123	11.569	Yes
X-11	23	58	211	20.019	Yes
X-36	25	56	92	8.947	Yes
X-12	22	59	195	18.590	Yes
X-37	27	54	147	14.289	Yes
X-13	25	56	158	15.395	Yes
X-38	27	54	146	14.238	Yes
X-14	24	57	168	16.325	Yes
X-39	22	59	136	12.909	Yes
X-15	23	58	184	17.405	Yes
X-40	26	55	128	12.052	Yes
X-16	27	54	132	13.371	Yes
X-41	27	54	94	8.867	Yes
X-17	22	59	223	21.234	Yes
X-42	22	59	122	11.519	Yes
X-18	24	57	171	16.186	Yes
X-43	23	58	124	11.872	Yes
X-19	25	56	162	15.116	Yes
X-44	24	57	110	10.346	Yes
X-20	23	58	193	18.201	Yes
X-45	23	58	130	12.237	Yes
X-21	25	56	128	12.679	Yes
X-46	25	56	152	14.493	Yes
X-22	22	59	150	13.984	Yes
X-47	23	58	224	21.765	Yes
X-23	23	58	201	19.224	Yes
X-48	25	56	143	13.536	Yes
X-24	22	59	119	11.304	Yes
X-49	26	55	146	14.174	Yes
X-25	27	54	110	10.690	Yes
X-50	23	58	133	12.697	Yes
Average	—	—	**151.8**	**14.526**	**50/50**

It outperforms the best prior hybrid genetic algorithm [6] by 358× in convergence speed on Easy puzzles, and succeeds on Hard and Expert instances where the prior method achieves less than 20% and 0% per-run success rates, respectively.

Future work will: (i) apply COIN-PWS to other CSPs such as graph coloring, nurse scheduling, and exam timetabling [3]; (ii) benchmark harder difficulty tiers on 16 × 16 and 25 × 25 grids; (iii) investigate diversity preservation mechanisms such as fitness sharing and crowding distance [4]; (iv) analyse theoretical convergence properties of the proportional dual-update rule; and (v) implement GPU-accelerated batch sampling for larger grids.

APPENDIX
PER-INSTANCE BENCHMARK RESULTS

Tables 13–16 report the per-instance results for all 200 benchmark puzzles (seeds 0–49 per tier).

ACKNOWLEDGMENT

The authors express sincere gratitude to Chulalongkorn University, the Faculty of Engineering, and the Department of Computer Engineering for their research support and to the advisors who guided the research writing process to successful completion. They declare no conflicts of interest.

REFERENCES

[1] H. Simonis, "Sudoku as a constraint problem," in *Proc. 4th Int. Workshop Model. Reformulating Constraint Satisfaction Problems*, 2005, pp. 13–27.

[2] W. Wattanapornprom, P. Olanviwitchai, P. Chutima, and P. Chongstitvatana, "Multi-objective combinatorial optimisation with coincidence algorithm," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 1675–1682.
[3] R. Lewis, "Metaheuristics can solve sudoku puzzles," *J. Heuristics*, vol. 13, no. 4, pp. 387–401, Jul. 2007.
[4] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Dordrecht, The Netherlands: Kluwer Academic, 2002.
[5] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proc. 2nd Int. Conf. Genetic Algorithms Genetic Algorithms Appl.*, J. J. Grefenstette, Eds., Hillsdale, NJ, USA: Lawrence Erlbaum Associates, 1987, pp. 14–21.
[6] X. Q. Deng and Y. D. Li, "A novel hybrid genetic algorithm for solving sudoku puzzles," *Optim. Lett.*, vol. 7, no. 2, pp. 241–257, Feb. 2013.
[7] S. Maire and C. Prissette, "A restarted estimation of distribution algorithm for solving sudoku puzzles," *Monte Carlo Methods Appl.*, vol. 18, no. 2, pp. 148–160, Jan. 2012.
[8] I. Lynce and J. Ouaknine, "Sudoku as a SAT problem," in *Proc. Int. Symp. Artif. Intell. Math. (ISAIM)*, 2006, pp. 34–43. [Online]. Available: https://dblp.org/db/conf/isaim/isaim2006.html
[9] B. Crawford, M. Aranda, C. Castro, and E. Monfroy, "Using constraint programming to solve sudoku puzzles," in *Proc. 3rd Int. Conf. Converg. Hybrid Inf. Technol.*, Nov. 2008, pp. 926–931.
[10] T. Mantere and J. Koljonen, "Solving, rating and generating sudoku puzzles with GA," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 1382–1389.
[11] T. Mantere and J. Koljonen, "Solving and analyzing sudokus with cultural algorithms," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intelligence)*, Jun. 2008, pp. 4053–4060.
[12] A. Moraglio, J. Togelius, and S. Lucas, "Product geometric crossover for the sudoku puzzle," in *Proc. IEEE Int. Conf. Evol. Comput.*, Sep. 2006, pp. 470–476.
[13] A. Moraglio and J. Togelius, "Geometric particle swarm optimization for the sudoku puzzle," in *Proc. 9th Annu. Conf. Genetic Evol. Comput.* New York, NY, USA: ACM, Jul. 2007, pp. 118–125.
[14] S. Jana, A. Dey, A. K. Maji, and R. K. Pal, "A novel hybrid genetic algorithm-based firefly mating algorithm for solving sudoku," *Innov. Syst. Softw. Eng.*, vol. 17, no. 3, pp. 261–275, Sep. 2021.
[15] G. K. Baydogmus, "Probability selection for solving sudoku with ant colony optimization algorithm," in *Proc. 1st Int. Conf. Inf. Syst. Inf. Technol. (ICISIT)*, Jul. 2022, pp. 161–165.
[16] C. Wang, B. Sun, K.-J. Du, J.-Y. Li, Z.-H. Zhan, S.-W. Jeon, H. Wang, and J. Zhang, "A novel evolutionary algorithm with column and sub-block local search for sudoku puzzles," *IEEE Trans. Games*, vol. 16, no. 1, pp. 162–172, Mar. 2024.
[17] J. Gunther and T. Moon, "Entropy minimization for solving sudoku," *IEEE Trans. Signal Process.*, vol. 60, no. 1, pp. 508–513, Jan. 2012.
[18] N. Pathak and R. Kumar, "Entropy guided evolutionary search for solving sudoku," *Prog. Artif. Intell.*, vol. 12, no. 1, pp. 61–76, Mar. 2023.

DARUNEE BUNMA is currently pursuing the Ph.D. degree in computer engineering and technology adoption architectures with Chulalongkorn University, Bangkok, Thailand. She is also a Lecturer with the Rajamangala University of Technology Phra Nakhon (RMUTP), Thailand. Her deep research interests include structural equation modeling, quantitative heuristics design, technical financial modeling systems, and evolutionary algorithm configurations.

PRABHAS CHONGSTITVATANA received the Ph.D. degree from the University of Edinburgh, U.K. He is currently a Professor with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand. His research interests include evolutionary computation heuristics, robotics engineering, hardware-software co-design frameworks, combinatorial optimizations, and quantum computing.