*Proceedings of Machine Learning Research* **318** (2026)
The 39th Canadian Conference on Artificial Intelligence

Canadian AI 2026
25–29 May 2026, Vancouver, BC

# Sequence-Based Evolutionary and Neural Strategies for Reducing Zone Crossings in Toolpaths

Yasamin Aali[†,*], Ansh Shah[†], Mohammad Istiaq Uddin[†], Kazi Nishat Anwar[†], Sheridan Houghten[†], Rahnuma Islam Nishat[†]
[†] Computer Science, Brock University, St. Catharines, ON, Canada

**Abstract**

Excessive transitions between material zones in 3D printing reduce both efficiency and integrity. We introduce a hybrid optimization framework to minimize these zone crossings in Hamiltonian toolpaths. Our approach combines the local search capabilities of Simulated Annealing (SA) with the global exploration of a sequence-based Genetic Algorithm (GA). Furthermore, we propose a hybrid neural network that models the learned optimization behavior and predicts efficient sequences of operations. Experiments show that our method significantly reduces zone crossings across various complex patterns, and proves its effectiveness and scalability for efficient multi-material additive manufacturing.

**Keywords:** 3D Printing, Reconfiguration, Artificial Intelligence

## 1. Introduction

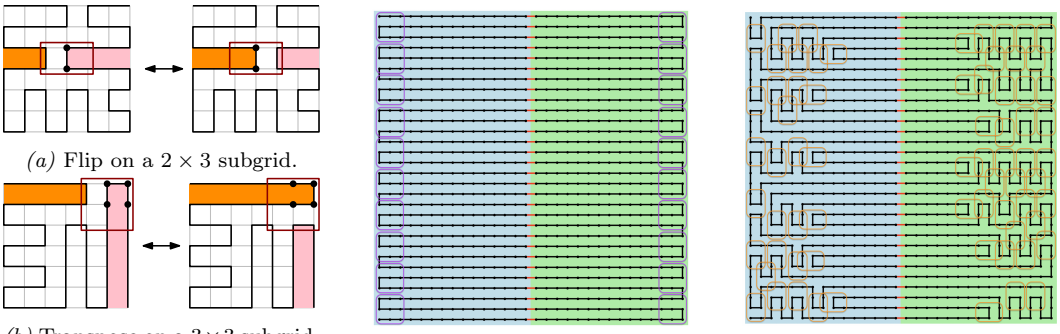
Additive manufacturing (AM) is the process of manufacturing an object by depositing materials (i.e., plastic, metal, wood etc.) layer by layer; 3D printing is one of the most common types of AM [1]. As opposed to subtractive manufacturing, 3D printing is popular in many applications, such as aerospace [2], biomedical engineering [3], and automotive applications [4], due to its ability to manufacture complex geometries with less material waste and rapid customization. Toolpaths are the routes that a 3D printer's head follows to deposit material and create a printed object. These paths play a critical role in determining the quality, speed, and strength of the final product. Therefore, many studies aim to model better toolpath designs for different objectives, such as improving geometric accuracy or enhancing mechanical performance. Toolpath strategies are generally divided into contour-parallel [5] and direction-parallel paths. Several works focus on optimizing printing parameters and improving print quality using learning-based models [6–11].

In practical 3D printing, objects may consist of multiple functional *zones*, such as areas made from different materials [12] or colors. Reducing the number of transitions between these zones is therefore critical, as each nozzle crossing increases printing time and repositioning overhead, disrupts material flow, and affects thermal behavior, which in turn reduces print quality. In other words, minimizing *zone crossing* would be desirable, where zone crossing refers to the toolpath crossing the boundary of two zones. In this paper, we study the problem of optimizing (minimizing) the zone crossing of a toolpath using reconfiguration of Hamiltonian paths and cycles in grid graphs [10].

Since 3D printing is inherently a layer-by-layer process, we optimize the toolpath independently for each layer and allow the method to scale to complex three-dimensional objects. Each layer is modeled as a two-dimensional grid, and we model the nozzle path as a Hamiltonian path over a grid graph, where each vertex represents a printable unit, and the path defines the nozzle's visit to each node. A Hamiltonian path is the best approach for this problem because it visits every node exactly once, ensuring a complete visit without revisiting previously printed parts and without lifting the nozzle.

We begin with a simple Hamiltonian path that we call a *zigzag* path ("c" in Figure 1). Although the zigzag path is commonly used as an initial toolpath [13], it is not efficient

[*] yaali@brocku.ca



*(a)* Flip on a $2 \times 3$ subgrid.

*(b)* Transpose on a $3 \times 3$ subgrid.

*(c)* Transpose candidates.

*(d)* Flip candidates.

*Figure 1.* Reconfiguration operations on Hamiltonian paths in a $7 \times 7$ grid. (a) Flip and (b) transpose operations. (c) Purple subgrids show transpose candidates, (d) orange subgrids show flip candidates.

on its own and typically results in a large number of zone crossings. To safely modify complex structures while preserving their key properties, we use reconfiguration strategies. *Reconfiguration* is the process of transforming one feasible solution of a problem into another through a sequence of small, valid steps, where each intermediate state remains a feasible solution of the original problem. To iteratively improve the path, we apply a series of local reconfiguration operations: *flips* and *transposes*, which are shown in figure 1. A *flip* operation is applied on a $3 \times 2$ or $2 \times 3$ subgrid where the two middle vertices (shown as black circles in Figure 1(a)) in that subgrid detach from one part (shown in pink) of the Hamiltonian path/cycle and attach to another part (shown in orange), giving a new Hamiltonian path/cycle. A *transpose* operation is applied on a $3 \times 3$ subgrid where the four vertices on one of the four cells in the subgrid (shown in black circles in Figure 1(b)) detach from one part of the Hamiltonian path/cycle and attach to another part.

Our framework (Figure 2) proceeds in four stages: (1) initialization of a baseline zigzag Hamiltonian path, (2) evolutionary refinement using SA and GA to generate optimized trajectories and a training dataset, (3) a hybrid CNN–RNN neural network that predicts effective operation sequences using spatial and temporal features fused via FiLM, and (4) reconstruction of an optimized toolpath with reduced zone crossings.

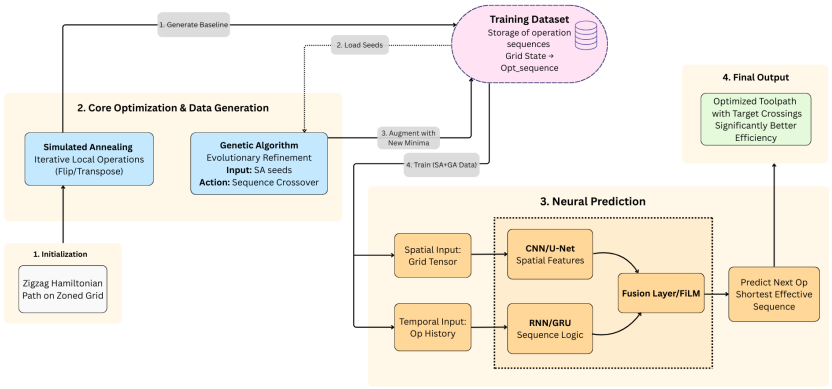



*Figure 2.* An overview of our hybrid optimization framework.

Our primary contributions can be summarized as follows: We propose a SA-based optimization framework for minimizing zone crossings in Hamiltonian toolpaths, and explore a GA over operation sequences, further improving minimization (Section 2.1). We propose a neural network–guided optimization strategy that learns to predict short sequences of effective reconfiguration operations (Section 2.2). We validate the effectiveness of all three methods through experiments across multiple zone patterns and grid sizes (Section 3). We show the effectiveness of our SA, GA and neural network algorithms by providing experimental results and corresponding analysis (Section 4).

Source code is publicly available at https://github.com/yasaminaali/3D-SGAHNN.git.

## 2. Methodology and algorithms

### 2.1. Heuristic Optimization Pipeline

We apply SA to optimize the Hamiltonian toolpath (i.e., minimize crossings between zones) using flips and transpose operations. Starting from an initial Hamiltonian toolpath (the zigzag path from Figure 1(c) is an example), at each iteration, SA iteratively applies local reconfiguration moves. To guide the search, we use a layered approach, optimizing from the outer boundary toward the center. A dynamic move pool maintains only valid operations and is refreshed as the path evolves, improving efficiency. Notably, we prioritize transpose moves during the initialization phase (first 60% of iterations) (shown in purple in Figure 1(c)), to break the rigid zigzag structure and create the irregular geometry needed for flip moves (shown in orange in Figure 1(d)). Valid mutations produce a candidate path $P'$, whose quality is evaluated by the change in crossings, $\Delta = \text{cross}(P') - \text{cross}(P)$. If the candidate improves the objective ($\Delta < 0$), it is accepted unconditionally. Otherwise, it is accepted with probability using the Metropolis acceptance rule $\exp(-\Delta/T)$, where $T$ is the current temperature. The algorithm reheats the temperature when no improvement is observed for several iterations.

While SA is effective at improving a single Hamiltonian toolpath at a time, a GA works with a population of solutions and explores multiple candidates simultaneously. Instead of starting from random toolpaths, we use good SA-generated solutions to initialize the GA population, allowing the GA to focus on refinement and recombination. Our algorithm was inspired by the idea of Dubé et al [14]. We recombine sequences of valid local operations that preserve Hamiltonicity when applied. To make this representation precise, each individual in the GA population is defined as a fixed-length sequence of local reconfiguration operations, $\mathcal{S} = (o_1, o_2, \ldots, o_L)$, where each gene $o_i = (\text{kind}, x, y, \text{variant})$ encodes a local reconfiguration operation. Tournament selection is used to choose parents, and one-point crossover combines their operation sequences. To preserve solution quality, a child is retained only if its crossing count satisfies $Crossings_{\text{Child}} \leq min(Crossings)_{\text{Parents}} + \epsilon$, which ($\epsilon$) is a small tolerance.

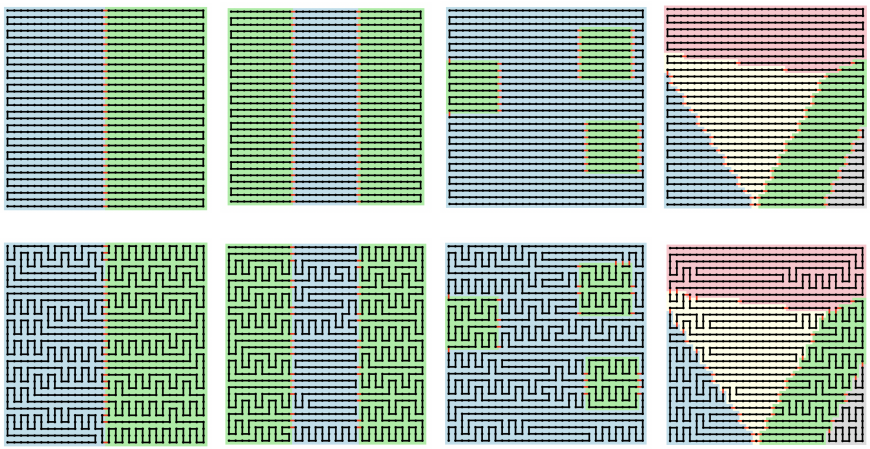
### 2.2. Hybrid Neural Architecture

We use FusionNet, a hybrid CNN–RNN architecture trained end-to-end to predict optimal reconfiguration operations. FusionNet formulates toolpath as a pixel-wise classification and ranking problem, which allows it to handle the discrete and multimodal nature of topological optimization. The spatial encoder is a 4-level Residual U-Net that processes a multi-channel grid representation of the toolpath, progressively reducing resolution to capture both local and global structural patterns. Group Normalization keeps training stable across varying grid sizes. At the smallest (8×8) level, Multi-Head Self-Attention lets the network reason about distant relationships. We employ a 2-layer GRU with 192 hidden units to track the sequence of recent operations. It compresses the history of recent operations into a context vector that summarizes the optimization trajectory so far. Feature-wise Linear

Modulation (FiLM) combines the spatial features with the temporal context. The GRU's context vector $z$ generates scale ($\gamma$) and shift ($\beta$) parameters that rescale each spatial decoder layer: $\text{FiLM}(F|z) = \gamma(z) \odot F + \beta(z)$. This way the model conditions its spatial predictions on the optimization history. Two output heads handle prediction jointly. The *position head* produces a $128 \times 128$ score map indicating the best location to act on. The *action head* produces a 12-channel volume ($12 \times 128 \times 128$) classifying the best operation (among 8 Transpose and 4 Flip variants) at each location. During inference, the model alternates between neural predictions and lightweight SA perturbations to escape local minima and explore new regions of the solution space.

## 3. Experiments

**Different Zone Patterns & Grid Sizes.** We evaluate the framework across multiple zone patterns and grid sizes. To test robustness across different zoning complexities, we use four zone layouts: *Left–Right* as a structured baseline, *Stripes* to represent repeated parallel regions, *Islands* to show many small disconnected regions, and *Voronoi* to generate irregular boundaries representative of realistic multi-color printing scenarios. Figure 3 shows examples of each layout with the initial zigzag path and the SA-optimized result. To assess scalability, we run experiments on square grids from $30 \times 30$ to $100 \times 100$ and a high-aspect-ratio $30 \times 100$ configuration, as the search space grows significantly with grid size.



*(a)* Left-Right *(b)* Stripes *(c)* Islands *(d)* Voronoi

*Figure 3.* Top row: Initial zig-zag Hamiltonian paths for different zoning patterns, with zone crossings highlighted in red. Bottom row: Optimized paths after applying SA.

## 4. Comparison and Results

**Minimization of Zone Crossings and Scalability.** Both SA and GA significantly outperform the standard zigzag toolpath baseline. As the primary optimization algorithm, SA demonstrated strong performance and significantly reduced crossings across all zone patterns. Table 3 summarizes the best crossing reductions achieved by SA compared to the standard zigzag toolpath. The results indicate that SA is most effective on structured zone patterns such as Left-Right and Stripes, consistently achieving reductions in the range of 50.0% to 60.0%. The GA achieved consistent results, maintaining a reduction between 50.0% and 60.0% across all square grids. In the Voronoi pattern, the GA reached a high

reduction of 58.3% on the $30 \times 30$ grid. Similarly, for the Islands ($8 \times 8$) pattern, the GA performed well on larger grids, achieving 50.0% reduction for both $80 \times 80$ and $100 \times 100$.

*Table 1.* Comparison between SA and GA models by the best percentage reduction (Red.) across different grid sizes and zone patterns.

| Grid Size/ | Left-Right | | Stripes ($k = 3$) | | Islands ($k = 3$) (8×8) | | Voronoi ($k = 3$) | |
|---|---|---|---|---|---|---|---|---|
| Zone Patterns | SA Red.(%) | GA Red.(%) | SA Red.(%) | GA Red.(%) | SA Red.(%) | GA Red.(%) | SA Red.(%) | GA Red.(%) |
| $30 \times 30$ | 60.0% | 60.0% | 50.0% | **56.3**% | 33.3% | **52.1**% | 43.3% | **58.3**% |
| $50 \times 50$ | 56.0% | 56.0% | 54.0% | 52.0% | 45.8% | 41.1% | 40.4% | 11.7% |
| $60 \times 60$ | 56.7% | 56.7% | 53.3% | 50.0% | 41.7% | 40.0% | 14.0% | **34.2**% |
| $80 \times 80$ | 50.0% | **55.0**% | 42.5% | **48.8**% | 33.3% | **50.0**% | 19.7% | 15.0% |
| $100 \times 100$ | 54.0% | 50.0% | 46.0% | 45.0% | 45.8% | **50.0**% | 23.4% | 20.0% |
| $30 \times 100$ | 56.0% | 53.0% | 54.0% | 53.0% | 45.8% | 38.5% | 37.3% | 20.2% |

**Sequence Efficiency and Neural Prediction.** To ensure a fair comparison, we established a fixed crossing count target for the NN based on the optimal solutions found by SA in the dataset generation phase (shown in Table 2). Across all tested patterns, the model consistently required fewer operations to achieve the optimization target compared to the stochastic search of SA in Islands and Voronoi zone patterns. For the $80 \times 80$ grid, the model converged in just 23 operations, whereas SA required 588 operations, a reduction of 96%. This shows that the model has effectively learned to identify and exploit the sparse, localized features of the Islands pattern without the need for extensive random exploration. While the Voronoi pattern presents a harder challenge, the model still reduces the operation count by approximately 30–67% of SA (e.g., 233 vs. 705 operations on $60 \times 60$ grid).

*Table 2.* Comparison of FusionNet (M) and SA in terms of average runtime (seconds) and number of operations for Voronoi and Islands patterns.

| | Voronoi | | | | Islands | | | |
|---|---|---|---|---|---|---|---|---|
| **Grid Size** | **M Time** | **SA Time** | **M Ops** | **SA Ops** | **M Time** | **SA Time** | **M Ops** | **SA Ops** |
| $30 \times 30$ | 8.5 | 23 | 246 | 343 | 7.1 | 21 | 228 | 346 |
| $50 \times 50$ | 18.1 | 50 | 287 | 556 | 18.8 | 48 | 222 | 338 |
| $30 \times 100$ | 22.9 | 143 | 300 | 533 | 6.5 | 146 | 100 | 115 |
| $60 \times 60$ | 24.8 | 166 | 233 | 705 | 19.8 | 179 | 156 | 564 |
| $80 \times 80$ | 64.0 | 6 | 332 | 712 | 32.2 | 424 | 23 | 588 |
| $100 \times 100$ | 153.6 | 8 | 487 | 748 | 75.2 | 9 | 81 | 528 |

The model significantly outperformed SA in terms of speed. On the $60 \times 60$ Voronoi grid, the model completed the task in 24.8 seconds, roughly 6.7 times faster than the SA baseline (166 seconds). Similarly, for the $30 \times 100$ Islands grid, the model finished in 6.5 seconds compared to 146 seconds for SA. It is worth noting that SA runtimes are shorter in some cases because we restrict it to short configurations.

## 5. Conclusion and Future Work

We present a hybrid AI framework to minimize zone crossings in multi-material 3D printing toolpaths. By applying local reconfiguration operations, our SA framework successfully reduced zone crossings, achieving peak reductions of approximately 45–56% across most structured zone patterns. We further apply a GA that evolves sequences of operations, achieving up to 58.3% reduction in crossing counts for Voronoi and Islands patterns. For

complex Voronoi and Islands patterns, FusionNet significantly outperformed SA by averaging 225 operations in 37.6 seconds, compared to SA's 506 operations in 101.9 seconds.

The neural model is most effective on complex patterns such as Islands and Voronoi, where heuristic methods struggle, but offers limited benefit on simpler layouts where constructive approaches are already near-optimal. Performance on larger grids is constrained by training data coverage. In future work, we plan to extend the framework to arbitrary polygon shapes to validate robustness on non-rectangular geometries, explore two-point crossover for improved GA search, and implement layer-aware optimization for full 3D objects, where each layer's path is incrementally adjusted from the previous layer rather than generated from scratch.

**References**


[1] A. Jadhav and V. Jadhav. "A review on 3D printing: An additive manufacturing technology". In: *Materials Today: Proceedings* 62 (2022), pp. 2094–2099.

[2] M. S. Karkun and S. Dharmalingam. "3D printing technology in aerospace industry–a review". In: *International Journal of Aviation, Aeronautics, and Aerospace* 9.2 (2022), p. 4.

[3] P. Agarwal, G. Arora, A. Panwar, V. Mathur, V. Srinivasan, D. Pandita, and K. S. Vasanthan. "Diverse applications of three-dimensional printing in biomedical engineering: a review". In: *3D Printing and Additive Manufacturing* 10.5 (2023), pp. 1140–1163.

[4] B. Choudhuri, M. Uddin, and R. Sen. "Review on 3D printing technology in automotive industry and electric vehicle". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* (2025), p. 09544070251337273.

[5] H. Zhao, F. Gu, Q.-X. Huang, J. Garcia, Y. Chen, C. Tu, B. Benes, H. Zhang, D. Cohen-Or, and B. Chen. "Connected fermat spirals for layered fabrication". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–10.

[6] O. Ulkir and G. Akgun. "Predicting and optimising the surface roughness of additive manufactured parts using an artificial neural network model and genetic algorithm". In: *Science and Technology of Welding and Joining* 28.7 (2023), pp. 548–557.

[7] S. Deswal, A. Kaushik, R. K. Garg, R. K. Sahdev, and D. Chhabra. "Optimization of fused deposition modelling printing parameters using hybrid GA-fuzzy evolutionary algorithm". In: *Sādhanā* 49.4 (2024), p. 257.

[8] S. Sundaram and A. Zeid. "Artificial intelligence-based smart quality inspection for manufacturing". In: *Micromachines* 14.3 (2023), p. 570.

[9] F. Talaat and E. Hassan. "Artificial intelligence in 3D printing". In: *Enabling Machine Learning Applications in Data Science: Proceedings of Arab Conference for Emerging Technologies 2020*. Springer. 2021, pp. 77–88.

[10] A. Bedel, Y. Coudert-Osmont, J. Martínez, R. Nishat, S. Whitesides, and S. Lefebvre. "Closed space-filling curves with controlled orientation for 3D printing". In: *Computer Graphics Forum*. Vol. 41. 2. Wiley Online Library. 2022, pp. 473–492.

[11] B. Aljabali, J. Shelton, and S. Desai. "Genetic Algorithm-Based Data-Driven Process Selection System for Additive Manufacturing in Industry 4.0". In: *Materials* 17.18 (2024), p. 4544.

[12] P. Patpatiya, K. Chaudhary, A. Shastri, and S. Sharma. "A review on polyjet 3D printing of polymers and multi-material structures". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 236.14 (2022), pp. 7899–7926.

[13] Y.-a. Jin, Y. He, G.-h. Xue, and J.-z. Fu. "A parallel-based path generation method for fused deposition modeling". In: *Int. J. Adv. Manuf. Technol* 77.5 (2015), pp. 927–937.

[14] M. Dubé, S. Houghten, and D. Ashlock. "Pandemic: A Graph Evolution Story". In: *2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. 2019, pp. 1–8.

## Appendix

**Evaluation Metrics.** We evaluate solution quality using the number of zone crossings, reporting both initial counts and the reduction achieved. For the GA, we additionally track per-generation convergence and population diversity. For the neural model, we measure the number of operations required and runtime performance compared to SA.

**Dataset Construction and Preprocessing.** We generate a dataset of 22,111 trajectories by running SA with varying grid sizes, zone patterns, and random seeds, recording the full sequence of operations and crossing counts for each run. This dataset is extended with GA-generated solutions, where each individual's operation sequence and resulting path are evaluated using the same crossing metric.

**Implementation details.** Hyperparameter settings are provided in Table 4. SA uses dynamic reheating (factor 1.5, cap $T = 600$), a move pool refreshed every 250 iterations, and a border-to-inner search heuristic. The GA population of 200 individuals is initialized from top SA solutions, with 60% carryover, top-$K$=6 elitism, and one-point crossover at probability 0.90 with a 60/40 split ratio and tolerance $\epsilon$=2. The neural network is trained on a stratified 80/10/10 split for up to 200 epochs using AdamW (learning rate $4 \times 10^{-4}$, batch size 64), cosine annealing with linear warmup, and early stopping with patience of 40 epochs.

*Table 3.* Comparing the best performance of the SA algorithm across different grid sizes and zone patterns: initial crossing count (Init.) and the percentage reduction (Red.).

| **Grid Size/** | **Left-Right** | | **Stripes** ($k = 3$) | | **Islands** ($k = 3$) (**8**×8) | | **Voronoi** ($k = 3$) | |
|---|---|---|---|---|---|---|---|---|
| **Zone Patterns** | **Init.** | **Red.(%)** | **Init.** | **Red.(%)** | **Init.** | **Red.(%)** | **Init.** | **Red.(%)** |
| $30 \times 30$ | 30 | 60.0% | 60 | 50.0% | 48 | 33.3% | 60 | 43.3% |
| $50 \times 50$ | 50 | 56.0% | 100 | 54.0% | 48 | 45.8% | 99 | 40.4% |
| $60 \times 60$ | 60 | 56.7% | 120 | 53.3% | 48 | 41.7% | 50 | 14.0% |
| $80 \times 80$ | 80 | 50.0% | 160 | 42.5% | 48 | 33.3% | 142 | 19.7% |
| $100 \times 100$ | 100 | 54.0% | 200 | 46.0% | 48 | 45.8% | 197 | 23.4% |
| $30 \times 100$ | 100 | 56.0% | 200 | 54.0% | 48 | 45.8% | 27 | 37.3% |

*Table 4.* Hyperparameter settings for the heuristic algorithms (SA, GA) and the proposed FusionNet architecture.

| **SA** | | **GA** | | **FusionNet** | |
|---|---|---|---|---|---|
| **Parameter** | **Value** | **Parameter** | **Value** | **Parameter** | **Value** |
| Max Iterations | 5000 | Population Size | 200 | Input Resolution | $128 \times 128$ |
| Init. Temp ($T_{max}$) | 80.0 | Max Generations | 500 | Encoder Depth | 4 Levels |
| Final Temp ($T_{min}$) | 0.5 | Crossover Rate | 0.9 | RNN Hidden Dim | 192 |
| Move Pool Size | 5000 | Selection | Tourn. ($k = 3$) | RNN Layers | 2 |
| Pool Refresh | 250 iter | Crossover Type | 1-Point | History Len ($K$) | 32 |
| Reheat Patience | 3000 iter | Elite Count | 6 | Batch Size | 64 |
| Reheat Factor | 1.5 | Retention | 60% | Learning Rate | $4 \times 10^{-4}$ |
| Transpose Ratio | 0.6 | | | Margin ($\lambda_m$) | 1.0 |

**Detailed Analysis.** Figure 4 provides a heat map visualization. Table 5 shows the average crossing reduction across multiple runs.

For instance, in the Left-Right pattern, SA decreases crossings from 100 to 46 on the $100 \times 100$ grid (54.0%). This behavior is also reflected in the average results, where reductions decline from 47.6% ($30 \times 30$) to 8.2% ($100 \times 100$) as the grid size increases. For

*Table 5.* Average crossing reduction ratio achieved by SA across different zone patterns and grid sizes.

| **Zone Patterns/Grid Size** | **30×30** | **50×50** | **60×60** | **80×80** | **100×100** | **30×100** |
|---|---|---|---|---|---|---|
| Left-Right | 47.6% | 33.1% | 33.7% | 24.1% | 8.2% | 44.3% |
| Stripes | 46.6% | 41.8% | 40.0% | 30.8% | 21.1% | 46.2% |
| Islands | 23.4% | 20.4% | 19.0% | 16.7% | 16.1% | 20.7% |
| Voronoi | 17.0% | 13.0% | 10.7% | 7.7% | 5.8% | 9.8% |

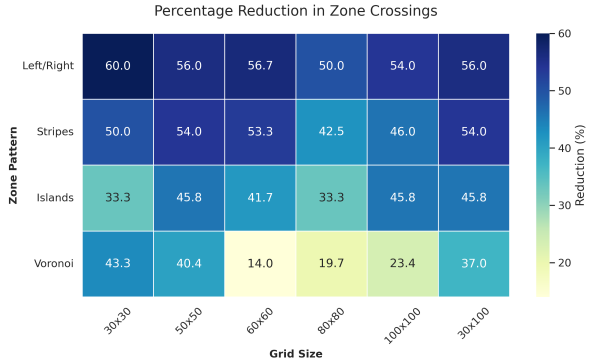
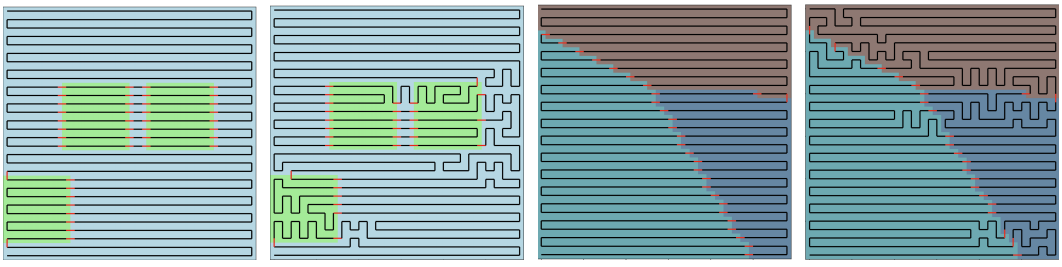



*Figure 4.* Heatmap illustrating the efficiency of the SA algorithm. Values represent the best percentage reduction in zone crossings.

more complex patterns, the performance naturally decreases but remains significant. In the Islands pattern, best-case reductions range between 33.3% and 45.8%, with average reductions of approximately 23.4% (30×30) and 16.1% (100×100). The Voronoi pattern presents greater difficulty, with the best reductions between 14.0% and 43.3%, and lower averages such as 17.0% (30 × 30) and 5.8% (100 × 100), highlighting the challenges introduced by irregular zone boundaries. In several cases, the GA clearly outperformed SA. For example, in the Islands pattern on the 80 × 80 grid, the GA improved the reduction from 33.3% to 50.0%. A similar improvement is observed in the 30 × 30 Voronoi case, where the GA achieved 58.3% compared to SA's 43.3%. The GA generally showed better scalability. On larger grids such as 80 × 80, the GA outperformed SA in most patterns.

The examples in Figure 5 show that in the Islands pattern, the model reduces crossings from 42 to 32 while applying 216 operations. For the Voronoi pattern, crossings are reduced from 29 to 23 using 169 operations.



*(a)* Init:42 *(b)* Opt:32 - M Ops:216 *(c)* Init:29 *(d)* Opt:23 - M Ops:169

*Figure 5.* Comparison between the initial zigzag path and the model-optimized path on Islands and Voronoi patterns.