From Hopfield Networks to Million-Net Routing: An Energy-Based Hybrid Router for Large-Scale Gate Array Global Routing

John Y. Lu
Lucid Systems
PO BOX 32193, Chicago, IL 60632
https://orcid.org/0009-0005-6525-2940

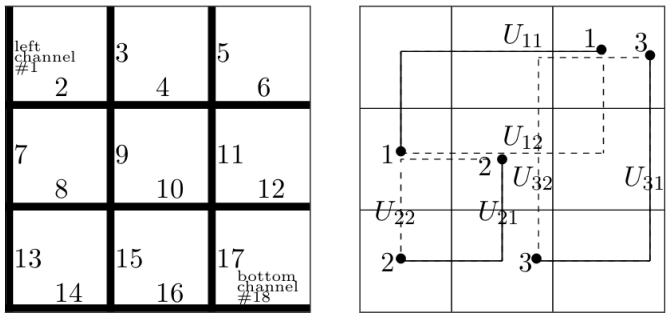
***Abstract*—This paper revisits the classical Hopfield approach to gate-array global routing and presents an evolutionary progression of three neural routing architectures: the Traditional Hopfield Router, the Legacy Router, and the Hybrid Router. The Traditional router—a faithful reproduction of Algorithm 1 from the seminal 1991 work by Lu and Thomborson [1], [2]; it follows the original Hopfield-Tank energy-minimization paradigm and provides a convergent Lyapunov-based routing framework. The Legacy router replaces classical neuron dynamics with a row-wise softmax update that can be interpreted as inspired by Exponentiated Gradient Descent (EGD) on the probability simplex, yielding great runtime improvements while preserving energy-guided optimization. The Hybrid router further integrates congestion-aware path generation, overflow-sensitive penalties, annealed softmax updates, and Rip-Up-and-Reroute (RRR) refinement to overcome congestion-induced local minima.**

Theoretical analysis shows that the Traditional router follows classical Hopfield Lyapunov descent, while the Legacy and Hybrid routers can be interpreted through the framework of Mirror Descent and Exponentiated Gradient optimization [3], [4], [5], [6]. All three are implemented in a unified simulation framework. Experimental evaluation on a Dell OptiPlex 7010 with an Intel® Core™ i5-3470 processor and 14 GB RAM demonstrates significant performance improvements over classical Hopfield routing. Across 100 independent trials per random generated benchmark, the Hybrid router achieved 100% valid routing solutions with zero violations on all benchmark instances from 10,240 to 25,000 nets. Additional scalability benchmarks from 100,000 to 500,000 nets achieved 100% Hybrid validity across 100 trials, and further extreme-scale experiments routed instances containing up to three million nets with 100% validity. The results demonstrate that modern energy-based optimization techniques can remarkably extend the practical scalability of Hopfield-inspired routing algorithms while preserving their fundamental optimization principles.

***Index Terms*—Hopfield networks, gate array global routing, energy-based optimization, neural optimization, exponentiated gradient descent, mirror descent, rip-up-and-reroute.**

I. Introduction

The physical design of Very Large Scale Integration (VLSI) circuits is a complex and multi-faceted process with a series of Electronic Design Automation (EDA) tasks. Central among these tasks is the routing phase, which involves establishing electrical connections between the various components of the circuit, such as gates and transistors, according to the netlist specifications. For specific VLSI architectures like gate arrays,



(a) (b)

Fig. 1: (a) A 3×3 gate array sample. (b) A 3-net 2-path gate array problem.

this problem presents unique challenges. An efficient and reliable global routing determines the approximate paths for all nets. Hence, the gate array global routing assigns one path per net in an $n \times n$ grid while respecting channel capacities and minimizing total wirelength. Each gate has a left and bottom channel, yielding $2n^2$ channels in total, each with a specified capacity (maximum number of tracks allowed). The routing problem is to select exactly one path for each net such that no channel exceeds its capacity. Consider a 3×3 gate array showing left channels(odd numbers) and bottom channels(even number)in Fig. 1(a) and Fig. 1(b) a 3-net 2-path gate array problem [7].

Earlier work by Lu and Thomborson [1] [2] introduced a neural network approach to gate array global routing that attracted attention from researchers in the field. The method represented each candidate path for a net as a neuron in a Hopfield–Tank network [3]. The associated energy function encoded:

- One-path-per-net constraints,
- Global activation constraints,
- Path length penalties,
- Channel congestion penalties.

While elegant, the original model was limited by static congestion modeling and difficulty escaping poor local minima. This paper revisits the neural formulation through a three-

tiered evolutionary framework:

1) **Traditional Hopfield Router**: faithful implementation of the 1991 algorithm.
2) **Legacy Router**: modern Hopfield enhancements with a revised energy function.
3) **Hybrid Router**: full integration of A*, vectorization, overflow penalties, and RRR.

By describing the algorithms, their energy formulations, and their implementation details, this paper provides a basis for comparing the three approaches and reproducing their results.

II. Background

A. Neural Networks for Optimization

Hopfield–Tank networks solve combinatorial optimization problems [3] by descending an energy landscape. Each neuron represents a decision variable, and problem constraints are encoded as energy terms. The system evolves via gradient descent on the energy function until convergence to a local minimum.

B. Neural Network Approach

The neural method treats routing as a simultaneous optimization over all nets. To the best of our knowledge, our 1991 work was the first to apply neural networks to gate array global routing. A key motivation for this study was to revisit that Hopfield-based framework and re-evaluate it on significantly larger benchmark instances than were previously feasible. This paper both documents that foundational approach and extends it using modern optimization techniques.

III. Traditional Hopfield Router

We present a complete, faithful implementation of Algorithm 8.1 from [1] which is identical to Algorithm 1 in [2].

A. Problem Representation

Given m nets and p candidate paths per net, define neuron matrix $U_{x,i}$ ($x = 1, \ldots, m$; $i = 1, \ldots, p$). Define indicator function:

$$G_{x,i,k} = \begin{cases} 1 & \text{if path } i \text{ of net } x \text{ uses channel } k, \\ 0 & \text{otherwise.} \end{cases}$$

B. Energy Function

Total energy: $E = E_1 + E_2 + E_3 + E_4$.

1) Term E_1: One Path Per Net:

$$E_1 = \frac{A}{2} \sum_x \sum_{i \neq j} U_{x,i} U_{x,j}$$

Zero iff at most one path per net is active.

2) Term E_2: Exactly m Active Neurons:

$$E_2 = \frac{B}{2} \left(\sum_{x,i} U_{x,i} - m \right)^2$$

3) Term E_3: Path Length Minimization: Let $L_{x,i}$ be path length, $L_{\min,x}$ the shortest for net x:

$$E_3 = \frac{C}{2} \sum_x \left(\sum_i U_{x,i} L_{x,i} - L_{\min,x} \right)^2$$

4) Term E_4: Channel Congestion Penalty: Let c_k be channel capacity, $c_{\max} = \max_k c_k$:

$$E_4 = \frac{D}{2} \sum_k \left(c_{\max} - c_k + \sum_{x,i} G_{x,i,k} U_{x,i} \right)^2$$

C. Connection Weights and Bias

Weight between neurons (x, i) and (y, j):

$$W_{x,i,y,j} = -A\delta_{x,y}(1-\delta_{i,j}) - B - C\delta_{x,y} L_{x,i} L_{y,j} - D \sum_k G_{x,i,k} G_{y,j,k}$$

where

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

External bias:

$$I_{x,i} = Bm + CL_{\min,x} L_{x,i} - D \sum_k (c_{\max} - c_k) G_{x,i,k}$$

D. Algorithm and Initialization

Algorithm 1 Traditional Hopfield Router (Algorithm 8.1 [1])

1: Initialize $W_{x,i,y,j}$ using weight formula above.
2: For each net x, randomly select one path i; set $U_{x,i} = 1.0 + 0.04 \cdot \text{rand}(0, 1)$; others to 0.
3: **repeat**
4: **for** each neuron (x, i) (random order) **do**
5: Compute input: $u_{x,i} = \sum_{y,j} W_{x,i,y,j} U_{y,j} + I_{x,i}$
6: Update output: $U_{x,i} \leftarrow \frac{1}{2} \left(1 + \tanh \left(\frac{u_{x,i}}{u_0} \right) \right)$
7: **end for**
8: **until** $|E^{(t)} - E^{(t-1)}| < \epsilon$
9: Round $U_{x,i}$ to $\{0, 1\}$; output selected paths.

E. Key Characteristics

The resulting dynamics admit a Lyapunov energy function whose monotonic descent guarantees convergence to a stable fixed point [3]. **Advantages**: Mathematical elegance, parallel processing, biological plausibility. **Limitations**: Slow convergence ($O(m^2 p^2)$), local minima, parameter sensitivity.

IV. Legacy Hopfield Router

The Legacy Hopfield Router represents an intermediate evolutionary step between the Traditional formulation and the full Hybrid model. While it retains the core neural architecture, it modernized update dynamics that improve convergence behavior and constraint satisfaction. Crucially, its energy function differs from the Traditional version in its implementation.

A. Energy Function

The Legacy router uses the same energy decomposition:

$$E = E_1 + E_2 + E_3 + E_4,$$

but uses the deviation of the energy function in the implementation compared to the Traditional router.

1) Term E_1: One Path Per Net :

$$E_1 = \frac{A}{2} \sum_x \sum_{i \neq j} U_{x,i} U_{x,j}$$

This term remains identical to the Traditional formulation, enforcing at most one active path per net.

2) Term E_2: Row-wise Normalization (Modified): Instead of enforcing a global sum of m, the Legacy router implicitly enforces one-hot selection per net through softmax sharpening (see Section IV-C). Consequently, the explicit E_2 term is often omitted or downweighted ($B \approx 0$) in practice, as the softmax layer handles the constraint more effectively. All experimental results reported in Section IX use the implementation of Algorithm 2 with the E_2 term retained. However, its contribution is substantially reduced because the row-wise softmax projection already enforces competition among paths belonging to the same net.

3) Term E_3: Direct Path Length Penalty: The Legacy router uses the same squared-deviation path length penalty as the Traditional router:

$$E_3 = \frac{C}{2} \sum_x \left(\sum_i U_{x,i} l_{x,i} - l_{min,x} \right)^2,$$

This penalizes the weighted active path length relative to the shortest path for each net, encouraging shorter route selection while remaining differentiable. The gradient with respect to $U_{x,i}$ is:

$$\frac{\partial E_3}{\partial U_{x,i}} = C(\sum_i U_{x,i} l_{x,i} - l_{min,x}) l_{x,i}$$

4) Term E_4: Congestion Penalty (Unchanged Form, Different Scaling):

$$E_4 = \frac{D}{2} \sum_k \left(\sum_{x,i} G_{x,i,k} U_{x,i} \right)^2.$$

Note that unlike the Traditional router—which includes $(c_{\max} - c_k)$ offsets—this version penalizes raw usage, making it more consistent with modern congestion models. The absence of capacity offsets simplifies gradient computation and improves numerical stability. The gradient with respect to $U_{x,i}$ is:

$$\frac{\partial E_4}{\partial U_{x,i}} = D \sum_k G_{x,i,k}, C_k$$

Where

$$C_k = \sum_{x,i} G_{x,i,k} U_{x,i}$$

B. Algorithm

The Legacy router proceeds as follows:

Algorithm 2 Legacy Hopfield Router

Require: Grid n, nets m, paths p, capacity $c_{\max}$, parameters A, B, C, D, β, max_iterations T, patience K

Generate p Manhattan paths per net; build $G_{x,i,k}$, $L_{x,i}$, $L_x^{\min} = \min_i L_{x,i}$.
Initialize U: for each net x, draw $i^* \sim \text{Uniform}\{1, \ldots, p\}$, set $U_{x,i^*} = 1 + 0.04 \cdot \text{rand}()$; all other entries $\leftarrow 0$.
$E_{\text{best}} \leftarrow E(U)$; $U_{\text{best}} \leftarrow U$; $n_\text{improve} \leftarrow 0$.
for $t = 1, 2, \ldots, T$ **do**
  Draw a uniformly random permutation π of all pairs (x, i).
  for each (x, i) in order π **do**
    (Asynchronous: U updated immediately after each neuron.)
    Recompute $\text{usage}_k \leftarrow \sum_{y,j} G_{y,j,k} U_{y,j}$ for all k (from current asynchronously-updated U).
    $e_1 = -A \left(\sum_j U_{x,j} - U_{x,i} \right)$
    $e_2 = -B \left(\sum_{y,j} U_{y,j} - m \right)$
    $e_3 = -C \left(\sum_j U_{x,j} L_{x,j} - L_x^{\min} \right) L_{x,i}$
    $e_4 = -D \sum_k G_{x,i,k} \cdot \text{usage}_k$
    $u_{x,i} = e_1 + e_2 + e_3 + e_4$
    $U_{x,i} \leftarrow \text{clip}(u_{x,i}, -10^6, 10^6)$
  end for
  Apply numerically stable row-wise softmax to raw gradient, for all x, i: $U_{x,i} \leftarrow \exp(\beta(U_{x,i} - \max_j U_{x,j})) \,/\, \sum_j \exp(\beta(U_{x,j} - \max_j U_{x,j}))$
  $E_{\text{curr}} \leftarrow E(U)$
  if $E_{\text{curr}} < E_{\text{best}}$ **then**
    $E_{\text{best}} \leftarrow E_{\text{curr}}$; $U_{\text{best}} \leftarrow U$; $n_\text{improve} \leftarrow 0$
  else
    $n_\text{improve} \leftarrow n_\text{improve} + 1$
  end if
  if $n_\text{improve} \geq K$ **then**
    break *(stagnation stop, analogous to SA termination)*
  end if
end for
Output: $\arg\max_i U_{x,i}^{\text{best}}$ for each net x *(return best routing found, not last)*

C. Modern Activation: Softmax Sharpening

The most significant departure from the Traditional router is the replacement of the tanh-based update with a row-wise softmax competition [8]:

$$U_{x,i} \leftarrow \frac{\exp(\beta U_{x,i})}{\sum_j \exp(\beta U_{x,j})},$$

where β is a temperature parameter (typically $\beta = 8.0$). This ensures:

- Exactly one path per net (hard constraint satisfaction),

- Stronger inter-path competition,
- Faster convergence (5.0 iterations vs. 7.4–30.6 in Traditional).

The softmax is applied after each full update cycle, acting as a projection onto the simplex of valid routing decisions. The resulting dynamics bear similarities to Exponentiated Gradient optimization [4].

This combination of a simplified energy landscape and discrete projection via softmax yields significantly improved performance over the Traditional router, as validated in Section IX EXPERIMENTAL RESULTS.

V. Hybrid Router

The Hybrid router introduces four key improvements beyond the Legacy model.

A. Congestion-Aware A* Path Generation

Generate p diverse paths per net using A* with dynamic cost:

$$\text{cost}(k) = 1 + \alpha \cdot \text{usage}[k], \quad \alpha \in [0.5, 1.0] \text{ for } i > 0$$

First path ($i = 0$) uses $\alpha = 0$ (shortest path).

B. Vectorized Energy Evaluation

Compute usage and congestion via tensor operations:

$$\text{usage} = \text{tensordot}(U, G, \text{axes} = ([0,1],[0,1]))$$
$$\text{congestion_cost} = \text{tensordot}(G, \text{usage}, \text{axes} = ([2],[0]))$$

Reduces complexity from $O(m^2p^2)$ to $O(mpn^2)$.

C. Overflow-Sensitive Penalty

Add new energy term:

$$E_5 = \lambda \sum_k \max(0, \text{usage}[k] - c_{\max})^2$$

Gradient of E_5 Let $O_k = \max(0, C_k - c_{\max})$. Then:

$$\frac{\partial E_5}{\partial U_{x,i}} = 2\lambda \sum_{k: C_k > c_{\max}} G_{x,i,k} O_k$$

Gradient w.r.t. $U_{x,i}$:

$$\frac{\partial E_5}{\partial U_{x,i}} = 2\lambda \sum_{k:\ \text{usage}[k] > c_{\max}} G_{x,i,k} (\text{usage}[k] - c_{\max})$$

Algorithm 4 implements this gradient as

$$e_5 = -\lambda_s \, ovflgrad_{x,i},$$

where $ovflgrad_{x,i}$ corresponds to the quantity in Eq. (13).

Combined Input to Neuron (x, i)

$$u_{x,i} = -\left(\frac{\partial E_1}{\partial U_{x,i}} + \cdots + \frac{\partial E_5}{\partial U_{x,i}} \right)$$

with each partial derivative as derived in Sections III, IV, and above.

D. Rip-Up-and-Reroute (RRR) Refinement Loop

Rip-Up-and-Reroute (RRR) refinement, a technique widely used in global and channel routing algorithms to repair congestion violations [9] [10]. The Rip-Up-and-Reroute Procedure proceeds as Algorithms 3:

Algorithm 3 Rip-Up-and-Reroute Procedure

Require: Current state U, indicator tensor G, path lengths L, base penalty λ_0, outer index s, inner iterations T_{in}, reroute fraction ρ, overflow threshold τ, capacity $c_{\max}$

1: Scale overflow penalty: $\lambda_s \leftarrow \lambda_0 \cdot (1 + 0.5 \cdot s)$
2: Run T_{in} steps of inner Hopfield dynamics with penalty λ_s (Algorithm 4, Steps 6–12), producing updated continuous state U.
3: Discretise: for each net x, set $U^{\text{disc}}_{x,i} = 1$ if $i = \arg\max_j U_{x,j}$, else 0.
4: $\text{usage}_k \leftarrow \sum_{x,i} G_{x,i,k} U^{\text{disc}}_{x,i}$ for all k; $W \leftarrow \sum_{x,i} U^{\text{disc}}_{x,i} L_{x,i}$
5: Compute total overflow: $\Omega \leftarrow \sum_k \max(0, \text{usage}_k - c_{\max})$
6: Log $\{s, \lambda_s, \Omega, W\}$ to stats.
7: **if** $\Omega \le \tau$ **then**
8: **return** U *(feasible—stop early)*
9: **end if**
10: Identify over-capacity channels: $\mathcal{K} \leftarrow \{k : \text{usage}_k > c_{\max}\}$
11: **for** each net $x = 1, \ldots, m$ **do**
12: $c_x \leftarrow \left|\{k \in \mathcal{K} : \exists\, i,\ G_{x,i,k}{>}0 \text{ and } U^{\text{disc}}_{x,i}{>}0\}\right|$
13: **end for**
14: $\mathcal{R} \leftarrow$ top $\max(1, \lceil \rho m \rceil)$ nets sorted by c_x descending
15: **for** each $x \in \mathcal{R}$ **do**
16: $U_{x,:} \leftarrow \mathbf{0}$; draw $i^* \sim \text{Uniform}\{1, \ldots, p\}$; $U_{x,i^*} \leftarrow 1 + 0.04 \cdot \text{rand}()$
17: **end for**
18: **return** U

VI. Relation to Energy-Based Models and Contrastive Divergence

Energy-based neural models often present an important changllege: balancing theoretical convergence guarantees and practical computational efficiency. A classical example is the Boltzmann Machine, whose stochastic dynamics provably converge to a stationary distribution defined by its energy function. However, exact learning requires computing intractable expectations over the model's equilibrium distribution. In practice, this led to the development of Contrastive Divergence (CD), which replaces exact gradient computation with a short, approximate Markov chain. CD sacrifices strict theoretical guarantees but dramatically improves tractability and empirical performance. A similar pattern appears in the evolutionary progression of our three routers.

A. Traditional Hopfield Router: Convergent but Computationally Expensive

The Traditional router is a faithful implementation of the Hopfield–Tank model. Its dynamics satisfy the classical conditions for a Lyapunov function: symmetric weights, asynchronous updates, and monotone activation. As a result, the total routing energy $E = E_1 + E_2 + E_3 + E_4$ decreases monotonically under the update rule, guaranteeing convergence to a fixed point. This router plays the same conceptual role as the full Boltzmann Machine: theoretically clean, provably convergent, but computationally slow.

B. Legacy Router: Practical Acceleration via Non-Lyapunov Updates

The Legacy router retains the same energy formulation but replaces the tanh-based Hopfield update with a row-wise softmax projection. This projection enforces one-path-per-net constraints more aggressively and yields an order-of-magnitude speedup. However, the softmax step is not derived from the classical Hopfield ODE and does not preserve strict monotonicity of the Lyapunov energy. The resulting dynamics resemble Contrastive Divergence:

- guided by the true energy landscape,
- still descending the energy before projection,
- but no longer guaranteed to decrease energy at every iteration.

Despite relaxing the formal convergence guarantees, the Legacy router consistently converges in practice and produces higher-quality solutions than the Traditional model.

C. Hybrid Router: Energy-Based Core with Heuristic Refinement

The Hybrid router extends the Legacy model with an overflow-sensitive penalty term and a Rip-Up-and-Reroute (RRR) outer loop. The inner loop remains an energy-based Hopfield descent with softmax projection, while the outer loop deliberately perturbs the state to escape poor minima. This mirrors the evolution from Boltzmann Machines to tempered transitions, persistent CD, and hybrid Monte Carlo—methods that preserve the spirit of energy-based optimization while introducing heuristic components that improve exploration and solution quality.

VII. Exponentiated Gradient Descent (EGD)-Inspired Interpretation of the Legacy and Hybrid Routers

The Legacy and Hybrid routers replace the classical Hopfield-Tank neuron activation mechanism with a row-wise softmax competition among candidate routing paths. Although these dynamics are not identical to classical Exponentiated Gradient Descent (EGD) [4], they exhibit several properties associated with EGD and entropic Mirror Descent [5]. This section develops an EGD-inspired interpretation that is fully consistent with the implemented Algorithms 2 and 4.

A. Routing Activations and Simplex Constraints

For each net x, the router maintains an activation vector

$$U_x = (U_{x,1}, U_{x,2}, \ldots, U_{x,p}), \tag{1}$$

where $U_{x,i}$ denotes the activation associated with candidate path i. Following softmax normalization, the activations satisfy

$$U_{x,i} \geq 0, \tag{2}$$

and

$$\sum_{i=1}^{p} U_{x,i} = 1. \tag{3}$$

Therefore each routing row lies on the probability simplex

$$\Delta_p = \left\{ U_x : U_{x,i} \geq 0,\ \sum_i U_{x,i} = 1 \right\}. \tag{4}$$

This simplex formulation naturally represents the competition among mutually exclusive routing paths.

B. Legacy Router Update

The Legacy router computes four incremental energy-driven terms. The one-path-per-net competition term is

$$e_1 = -A \left(\sum_j U_{x,j} - U_{x,i} \right). \tag{5}$$

The global activation term is

$$e_2 = -B \left(\sum_{x,i} U_{x,i} - m \right), \tag{6}$$

where m denotes the number of nets. The path-length term is

$$e_3 = -C \left(\sum_j U_{x,j} L_{x,j} - L_{\min,x} \right) L_{x,i}. \tag{7}$$

The congestion term is

$$e_4 = -D \sum_{k \in P_{x,i}} \text{usage}_k, \tag{8}$$

where

$$\text{usage}_k = \sum_{x,i} G_{x,i,k} U_{x,i}. \tag{9}$$

The combined score used by the implementation is

$$u_{x,i} = e_1 + e_2 + e_3 + e_4. \tag{10}$$

Softmax normalization then produces

$$U_{x,i} = \frac{\exp(\beta(u_{x,i} - M_x))}{\sum_j \exp(\beta(u_{x,j} - M_x))}, \tag{11}$$

with

$$M_x = \max_j u_{x,j}. \tag{12}$$

C. Hybrid Router Update

The Hybrid router retains the Legacy terms and introduces an additional overflow-sensitive penalty. The overflow contribution is

$$e_5 = -2L_{\text{over}} \sum_{k \in P_{x,i}} \max\left(0, \text{usage}_k - C_{\max}\right). \tag{13}$$

The resulting Hybrid score becomes

$$u_{x,i} = e_1 + e_2 + e_3 + e_4 + e_5. \tag{14}$$

Softmax normalization is then applied using an annealed temperature schedule. In practice, e_2 has little influence because the row-wise softmax already enforces strong competition among candidate paths.

D. Relationship to Exponentiated Gradient Descent

Classical Exponentiated Gradient Descent updates simplex-constrained variables according to [4]

$$U_{x,i}^{t+1} = \frac{U_{x,i}^t \exp(-\eta g_{x,i})}{\sum_j U_{x,j}^t \exp(-\eta g_{x,j})}. \tag{15}$$

Algorithms 2 and 4 do not implement (15) directly. Instead, they compute additive energy-driven scores (10) and (14) followed by the simplex-preserving softmax projection of (11). Consequently, the Legacy and Hybrid routers should not be interpreted as exact implementations of EGD. Nevertheless, the implemented dynamics share several important characteristics with EGD:

- simplex preservation;
- exponential normalization;
- multiplicative preference amplification;
- competition among candidate paths;
- rapid empirical convergence.

For this reason, the algorithms may be viewed as EGD-inspired optimization procedures.

E. Connection to Mirror Descent

Mirror Descent generalizes Euclidean optimization by employing a geometry adapted to the feasible set. For simplex-constrained optimization, a common mirror map is the negative-entropy function [5]

$$\psi(U_x) = \sum_i U_{x,i} \log U_{x,i}. \tag{16}$$

Because the routing activations remain on a simplex through softmax normalization, the behavior of the Legacy and Hybrid routers resembles entropic Mirror Descent. This relationship should be regarded as an interpretive framework rather than a strict equivalence. The routing energy is highly nonconvex, and the Hybrid router additionally incorporates overflow penalties, temperature annealing, congestion-aware path generation, and Rip-Up-and-Reroute refinement. Therefore the classical convergence guarantees of convex Mirror Descent do not directly apply [6].

F. Summary

The Legacy and Hybrid routers are not exact implementations of classical Exponentiated Gradient Descent. Instead, they employ energy-based scoring functions followed by simplex-preserving softmax normalization. These dynamics are closely related to EGD and entropic Mirror Descent while remaining fully consistent with Algorithms 2 and 4. The complete Hybrid router further augments this optimization core with overflow-aware penalties and Rip-Up-and-Reroute refinement, yielding an energy-based metaheuristic [11] that has successfully routed benchmark instances up to 3 million nets with 100% validity.

Algorithm 4 Hybrid Router with Rip-Up-and-Reroute

Require: Grid n, nets m, paths p, capacity $c_{\max}$, parameters A, B, C, D, λ_0, β_{start}=2, β_{end}=12, S=3 outer iters, T_{in}=10 inner iters, reroute fraction ρ=0.3, overflow threshold τ=0

Generate p congestion-aware Manhattan paths per net; build $G_{x,i,k}$, $L_{x,i}$, $L_x^{\min} = \min_i L_{x,i}$.
Initialize U: for each net x, draw $i^* \sim \text{Uniform}\{1, \ldots, p\}$, set $U_{x,i^*} \leftarrow 1 + 0.04 \cdot \text{rand}()$; all other entries $\leftarrow 0$.
Save base penalty $\lambda_0 \leftarrow \lambda$.
for $s = 0, 1, \ldots, S-1$ **do**
  $\lambda_s \leftarrow \lambda_0 \cdot (1 + 0.5 \cdot s)$ *(penalty grows each outer loop)*
  for $t = 0, 1, \ldots, T_{\text{in}} - 1$ **do**
    Anneal: $\beta_t \leftarrow \beta_{\text{start}} + \frac{t}{T_{\text{in}}-1}(\beta_{\text{end}} - \beta_{\text{start}})$
    $\text{usage}_k \leftarrow \sum_{x,i} G_{x,i,k} U_{x,i}$; $\text{cong}_{x,i} \leftarrow \sum_k G_{x,i,k} \cdot \text{usage}_k$
    $\text{ovfl}_k \leftarrow \max(0, \text{usage}_k - c_{\max})$; $\text{ovflgrad}_{x,i} \leftarrow 2\sum_k G_{x,i,k} \cdot \text{ovfl}_k$
    $e_1 \leftarrow -A(\sum_j U_{x,j} - U_{x,i})$; $e_2 \leftarrow -B(\sum_{x,i} U_{x,i} - m)$
    $e_3 \leftarrow -C(\sum_j U_{x,j} L_{x,j} - L_x^{\min}) L_{x,i}$; $e_4 \leftarrow -D \cdot \text{cong}_{x,i}$; $e_5 \leftarrow -\lambda_s \cdot \text{ovflgrad}_{x,i}$
    $u_{x,i} \leftarrow e_1 + e_2 + e_3 + e_4 + e_5$ for all (x, i) simultaneously
    Numerically stable softmax (no tanh) for all (x, i): $U_{x,i} \leftarrow \exp(\beta_t(u_{x,i} - \max_j u_{x,j})) \,/\, \sum_j \exp(\beta_t(u_{x,j} - \max_j u_{x,j}))$
  end for
  Run Algorithm 3 (RRR) with current U, λ_s, s; receive updated U.
  if Algorithm 3 returned early (feasible, $\Omega \le \tau$) **then**
    break
  end if
end for
Restore $\lambda \leftarrow \lambda_0$.
Output: $\arg\max_i U_{x,i}$ for each net x.

VIII. Implementation Details

A. Data Structures

- G: 3D array $[m, p, 2n^2]$, binary channel usage.

- U: 2D array $[m, p]$, neuron activations.
- Paths: list of (cell, edge) tuples.

B. Channel Numbering

For cell (i, j) in $n \times n$ grid:

- Left channel ID: $2(i-1)n + 2(j-1) + 1$ (odd)
- Bottom channel ID: $2(i-1)n + 2(j-1) + 2$ (even)

C. Computational Complexity

For a problem of n grid with m nets and p paths per net,

- **Traditional**: $O(\text{iter} \cdot m^2 p^2)$
- **Legacy**: $O(\text{iter} \cdot mpn^2)$ The complexity is dominated by channel-usage accumulation over $O(n^2)$ channels for each path candidate.
- **Hybrid**: $O(\text{outer} \cdot \text{inner} \cdot mpn^2 + pmn^2 \log n)$

IX. EXPERIMENTAL RESULTS

The Traditional Hopfield Router, Legacy Router, and Hybrid Router were evaluated on a broad collection of random generated benchmarks ranging from 8 nets to 3,000,000 nets. The primary benchmark suite up to 25,000-net and additional extreme-scale experiments from 100,000 to 3,000,000 nets to explore scalability were executed on a Dell OptiPlex 7010 with an Intel® Core™ i5-3470 processor and 14 GB RAM.

For each benchmark, 100 independent trials were executed using randomized net ordering and randomized initial neuron states. A routing solution is considered *valid* when all channel-capacity constraints are satisfied and no overflowed channels remain.

The Traditional router terminates when the routing energy converges, the Legacy router uses a five-iteration stagnation criterion (maximum 100 iterations), and the Hybrid router reports the number of Rip-Up-and-Reroute (RRR) outer loops required to obtain a feasible solution.

A. Small-Scale Benchmarks

Table I summarizes representative results for benchmarks containing fewer than 100 nets.

TABLE I: Representative Small-Scale Benchmark Results

Benchmark	Router	Valid (%)	Runtime (s)
8 Nets	Traditional	87	0.0336
4x4 Grids	Legacy	91	0.0015
	Hybrid	**100**	**0.0016**
50 Nets	Traditional	100	1.1974
10x10 Grids	Legacy	100	0.0047
	Hybrid	**100**	**0.0025**
80 Nets	Traditional	96	3.4175
15x15 Grids	Legacy	87	0.0083
	Hybrid	**100**	**0.0035**

For lightly congested routing problems all three routers achieve high routing quality. However, the Legacy and Hybrid routers exhibit significantly lower runtime due to their softmax-based optimization strategy. The Hybrid router consistently achieves either matching or superior validity while retaining the fastest execution time.

B. Medium-Scale Benchmarks

As routing density increases, the differences between the three neural architectures become more pronounced.

TABLE II: Medium-Scale Routing Benchmarks

Benchmark	Router	Valid (%)	Runtime (s)
160 Nets	Traditional	82	16.4832
20x20 Grids	Legacy	75	0.0161
	Hybrid	**100**	0.0064
320 Nets	Traditional	65	88.2785
30x30 Grids	Legacy	68	0.0309
	Hybrid	**94**	0.0119
1,000 Nets	Traditional	90	1910.58
50x50 Grids	Legacy	89	0.0983
	Hybrid	**100**	**0.0341**

The Traditional router becomes computationally impractical beyond approximately 1,000 nets. The Traditional and Legacy routers begin to deteriorate rapidly under congestion. In contrast, the Hybrid router maintains near-perfect routing validity through the combined effect of congestion-aware path generation, overflow penalties, and RRR-based route repair.

C. Large-Scale Benchmarks

Table III summarizes the large-scale results obtained on the platform.

TABLE III: Large-Scale Benchmark Results

Benchmark	Router	Valid (%)	Avg. Viols.	Runtime (s)
10,240 Nets	Legacy	75	0.31	2.3873
100x100 Grids	Hybrid	**100**	**0.00**	**0.6637**
13,000 Nets	Legacy	39	0.94	4.0252
100x100 Grids	Hybrid	**100**	**0.00**	**2.1144**
15,000 Nets	Legacy	53	0.64	4.7783
100x100 Grids	Hybrid	**100**	**0.00**	**3.0335**
20,000 Nets	Legacy	66	0.41	6.7473
120x120 Grids	Hybrid	**100**	**0.00**	**3.5650**
25,000 Nets	Legacy	24	1.31	8.9021
130x130 Grids	Hybrid	**100**	**0.00**	**5.4159**

Although the Legacy router remains computationally efficient, its validity decreases substantially as congestion increases. At 25,000 nets, for example, the Legacy router achieves only 24% valid routing solutions.

In contrast, the Hybrid router maintains 100% validity and zero average violations across every benchmark from 10,240 through 25,000 nets. These results demonstrate the effectiveness of combining Exponentiated Gradient optimization with overflow-aware penalties and iterative congestion repair.

D. Extreme-Scale and Million Nets Scalability Experiment

To further evaluate Legacy and Hybrid scalability, additional benchmark containing from 100,000 nets to 500,000 nets was executed with channel capacity equal to 9 in 100 trials .

The Legacy and Hybrid Scalability experiment reveals the fundamental distinction between the Legacy and Hybrid architectures. The Legacy router suffers severe congestion-induced degradation, producing valid solutions down to only 4% of the 500,000 nets. The Hybrid router, however, maintains 100% validity while eliminating all routing violations and completing execution approximately 53% faster than the Legacy approach.

TABLE IV: Legacy and Hybrid Scalability Benchmark

Benchmark	Router	Valid (%)	Avg. Viols.	Runtime (s)
100,000 Nets	Legacy	51	0.63	40.8859
260x260 Grids	Hybrid	**100**	**0.00**	22.8859
120,000 Nets	Legacy	61	0.48	47.9119
290x290 Grids	Hybrid	**100**	**0.00**	**25.4956**
150,000 Nets	Legacy	36	0.87	62.4121
320x320 Grids	Hybrid	**100**	**0.00**	**33.6118**
200,000 Nets	Legacy	36	1.15	85.2400
370x370 Grids	Hybrid	**100**	**0.00**	**44.5911**
350,000 Nets	Legacy	13	1.90	142.8966
490x490 Grids	Hybrid	**100**	**0.00**	**76.4977**
500,000 Nets	Legacy	4	3.47	225.2427
580x580 Grids	Hybrid	**100**	**0.00**	**121.3073**

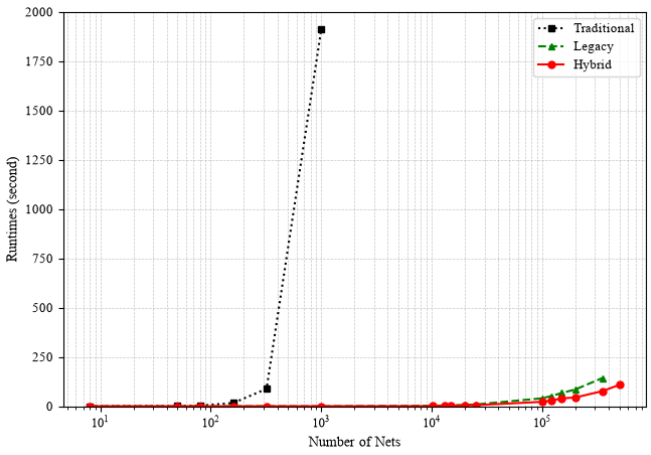


Fig. 2: Runtime versus benchmark size. The Traditional router exhibits rapidly increasing execution time, whereas the Legacy and Hybrid routers maintain practical runtimes across large-scale benchmarks.

Table V shows Hybrid extreme scale results in 100 trials. To test the extreme scalability of Hybrid router an additional benchmark containing up to 3,000,000 nets was also executed with path per net equal to 4 and channel capacity equal to 9 in 100 trials. Most importantly, these results indicate that the scalability limit of the Hybrid router has not yet been reached. The successful routing of a 3 million-net benchmark suggests that the combination of Exponentiated Gradient optimization, overflow-sensitive penalties, congestion-aware path generation, and Rip-Up-and-Reroute refinement remains effective far beyond the problem sizes traditionally associated with Hopfield-based routing systems.

TABLE V: Extreme-Scale Hybrid Benchmark Results

Nets	Grids	Valid (%)	Iterations	Runtime (s)
1,000,000	820x820	**100**	1.1	236.6999
1,500,000	1100x1100	**100**	1.0	265.2877
2,000,000	1300x1300	**100**	1.0	305.6015
2,500,000	1300x1300	**100**	1.2	630.8614
3,000,000	1500x1500	**100**	1.0	818.0054

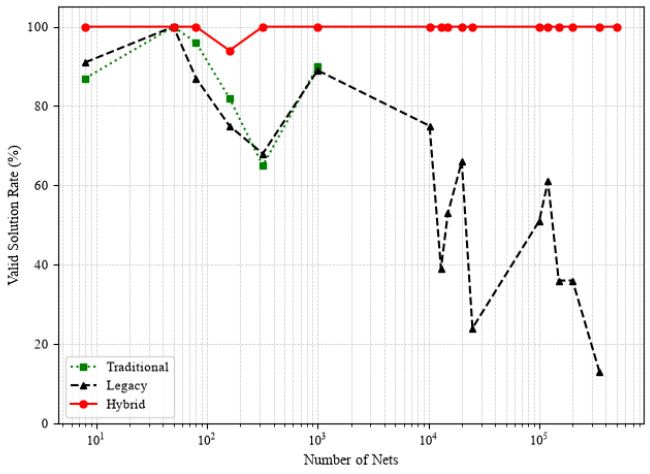


Fig. 3: Valid solution rate versus benchmark size. The Traditional router deteriorates rapidly under increasing congestion, while the Legacy router remains effective only for moderate-scale problems. The Hybrid router maintains near-perfect validity across all tested benchmark sizes, achieving 100% validity through 3,000,000 nets.

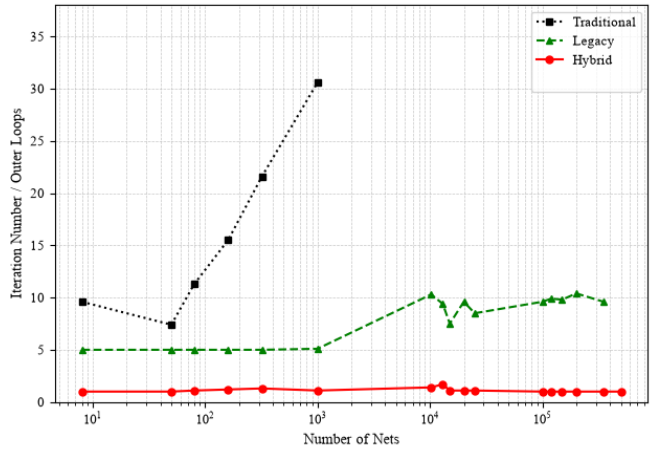


Fig. 4: Iterations versus Benchmark Size. Iterations note: Traditional is variable and stops when $E(t)-E(t-1) < 1e-6$. Legacy best-energy tracking is applied and it stops after 5 non-improving iterations. Hybrid RRR outer loops is actually used and it stops early if overflow is 0.

E. Discussion

The experimental results reveal a clear evolutionary progression among the three routing architectures.

The Traditional router represents classical Hopfield energy minimization and provides a convergent Lyapunov-based baseline. The Legacy router replaces classical neuron dynamics with softmax-based Exponentiated Gradient updates, producing dramatic runtime improvements but remaining vulnerable to congestion-induced local minima. Finally, the Hybrid router augments the EGD-Inspired framework with overflow-aware

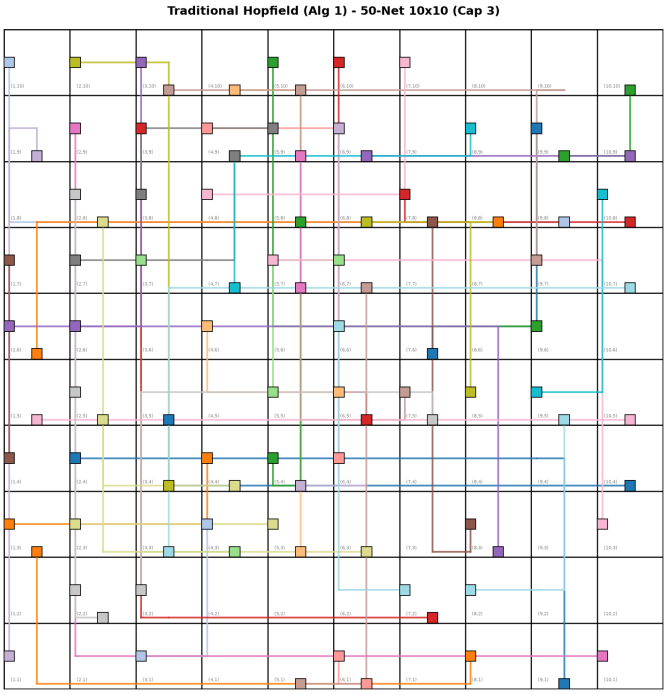


Fig. 5: A 50-net 3-path Solution by Traditional Router.

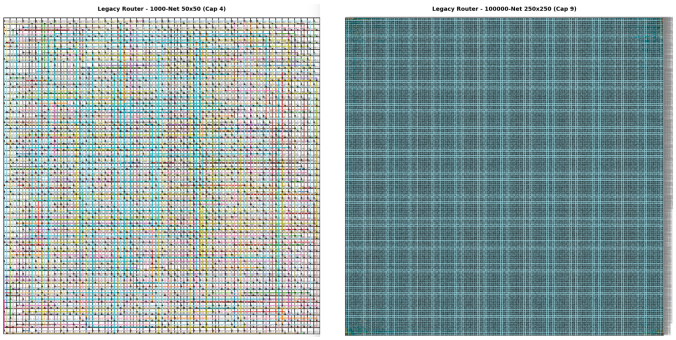


Fig. 6: Representative large-scale routing solutions generated by the Legacy and Hybrid routers.

energy terms and iterative route repair, yielding both high scalability and consistently feasible routing solutions.

These observations support the theoretical interpretation developed in earlier sections: the Legacy and Hybrid routers may be viewed as modern energy-based optimization systems operating on the probability simplex, with the Hybrid router further incorporating metaheuristic [11], [12] refinement mechanisms to overcome the limitations of pure energy descent.

X. Conclusion

This paper revisited the classical Hopfield approach to gate-array global routing and demonstrated how modern optimization techniques can transform a mathematically elegant but computationally limited neural routing model into a scalable, high-performance routing framework. Building upon the original Lu-Thomborson formulation for routing optimization [2], three generations of neural network routers were investigated: the Traditional Hopfield Router, the Legacy Router, and the Hybrid Router.

The Traditional Router serves as an exact implementation of the 1991 foundation [2] based on classical Hopfield-Tank model [3]. Its dynamics are controled by a Lyapunov energy function whose monotonic descent provides convergence to a stable fixed local minima. While mathematically appealing, the computational cost of asynchronous Hopfield updates increases rapidly with problem size, limiting practical scalability.

The Legacy Router modernizes the Hopfield framework with a revised energy function and by replacing tanh-based neuron dynamics with row-wise softmax competition. We showed that this update can be interpreted as Exponentiated Gradient Descent (EGD)-inspired [4] or, equivalently, Mirror Descent with Kullback-Leibler divergence [5]. This reformulation produces striking runtime improvements while preserving the energy-based character of the optimization process. However, the absence of explicit congestion-repair mechanisms causes routing quality to deteriorate under severe congestion as benchmark sizes increase.

The Hybrid Router extends the Legacy model through the introduction of congestion-aware path generation, overflow-sensitive energy penalties, annealed softmax updates, vectorized congestion evaluation, and a Rip-Up-and-Reroute (RRR) refinement loop. From an optimization standpoint, the Hybrid Router can be interpreted as an energy-based metaheuristic in which an EGD-inspired neural optimization core is embedded within a congestion-correction framework. This combination preserves the efficiency of simplex-constrained optimization while providing a mechanism for escaping poor local minima.

The primary experimental evaluation was run on a Dell OptiPlex 7010 in Python. Across 100 independent trials per benchmark, the Hybrid Router consistently outperformed both Traditional and Legacy approaches. Within the reported benchmark suite, the Hybrid Router achieved 100% valid routing solutions with zero violation on every benchmark from 10,240 nets through 25,000 nets, whereas the Legacy Router exhibited significant degradation as congestion increased, reaching only 24% validity on the 25,000-net benchmark.

Additional extrem-scale experiments from 100,000 to 3,000,000 nets further demonstrated the scalability of the Hybrid architecture.

More broadly, the results reveal a clear evolutionary progression in neural routing methodology:

• Traditional Router: classical Hopfield energy minimization and Lyapunov descent [3].

• Legacy Router: Exponentiated Gradient Descent inspired and Mirror Descent on the probability simplex [4], [5].

• Hybrid Router: energy-based optimization augmented by congestion-aware penalties and metaheuristic refinement.

This progression mirrors the historical development of other energy-based learning systems, where theoretical convergence guarantees are complemented by practical mechanisms for improving scalability and escaping poor local optima [4]- [6].

The state-of-the-art performance of Hybrid router in successful routing of benchmark instances containing up to

3,000,000 nets with 100% validity in experimental trials demonstrates that Hopfield-inspired neural optimization remains highly relevant for large-scale gate array global routing in modern VLSI design automation when combined with modern optimization and search techniques. By integrating energy-based modeling, Exponentiated Gradient updates, simplex-constrained optimization, congestion-aware penalties, and iterative refinement, the proposed Hybrid Router extends the practical range of Hopfield-based routing by several orders of magnitude while preserving the fundamental energy-minimization philosophy established by Hopfield and Tank [3].

These results provide evidence that classical neural optimization frameworks, when reinterpreted through the lens of modern optimization theory, can remain competitive for contemporary large-scale routing problems and may form the basis of future energy-based routing systems. More broadly, the results demonstrate how classical Hopfield-style energy minimization can be extended through simplex-preserving softmax competition and iterative repair mechanisms to remain effective on optimization problems several orders of magnitude larger than those originally considered.

TABLE VI: Evolution of Three Routers

Router	Optimization Paradigm	Convergence Interpretation
Traditional	Hopfield-Tank Energy	Lyapunov Descent
Legacy	EGD-inspired	Mirror Descent on the Simplex
Hybrid	EGD-inspired + Overflow Penalty + RRR	Energy-Based Metaheuristic with EGD-Inspired Core

XI. LIMITATIONS

Although the Hybrid Router demonstrates strong performance across all evaluated benchmarks, several limitations remain.

A. Prototype Implementation

The current implementation is written in Python that runs on a CPython interpreter. Unlike compiled C/C++ implementations, Python executes interpreted bytecode and incurs substantial runtime overhead. Consequently, the reported execution times should be viewed as measurements of a research prototype rather than the maximum achievable performance of the underlying algorithm. The observed scalability therefore reflects both algorithmic complexity and interpreter overhead. Significant additional performance improvements are expected through implementation in a compiled language, just-in-time compilation, or GPU acceleration.

B. Hardware Dependence of Benchmark Results

The primary benchmark suite reported in this paper was executed on a Dell OptiPlex 7010 equipped with an Intel® Core™ i5-3470 processor and 14 GB of RAM. The reported runtimes therefore reflect the capabilities of this specific hardware platform, which relies on a third-generation processor released in early 2012. Importantly, subsequent experiments of Legacy and Hybrid router successfully routed benchmark instances containing up to 350,000 nets with 100% Hybrid validity. Additional Hybrid router experiments successfully routed larger instances containing up to 3,000,000 nets, also with 100% validity. These further tests suggest that the scalability limit of the Hybrid Router has not yet been reached, and that the 3,000,000-net benchmark reported herein should not be interpreted as an upper bound on algorithmic capability. The largest experiments were limited primarily by available runtime and memory budgets rather than by observed algorithmic failure.

C. Two-Dimensional Routing Model

The present formulation focuses on two-dimensional gate-array routing. Modern VLSI designs frequently require multi-layer routing, via modeling, timing constraints, and manufacturability considerations that are not explicitly incorporated into the current energy function.

D. Lack of Global Optimality Guarantees

The Traditional router possesses Lyapunov convergence properties inherited from classical Hopfield networks. The Legacy and Hybrid routers, however, employ softmax-based Exponentiated Gradient updates and Rip-Up-and-Reroute refinements. While these methods exhibit consistent empirical convergence, they do not guarantee convergence to a globally optimal routing solution.

ACKNOWLEDGEMENTS

To Clark D. Thomborson for his essential contribution to "Gate Array Global Routing Using A Neural Network" [2]. This work builds directly on those ideas.

REFERENCES

[1] Y. Lu, "Solving combinatorial optimization problems by simulated annealing, genetic algorithms, and neural networks," *University of Minnesota*, September 1991.

[2] Y. Lu and C. D. Thomborson, "Gate array global routing using a neural network," *Intelligent Engineering Systems Through Artificial Neural Networks*, pp. 895–900, ASME Press, 1991.

[3] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[4] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors,"*Information and Computation*, vol. 132, no. 1, pp. 1-63, 1997.

[5] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*,vol. 31, no. 3, pp. 167-175, 2003.

[6] N. Cesa-Bianchi and G. Lugosi, "Prediction, Learning, and Games", Cambridge University Press, 2006.

[7] R. M. Karp, F. Leighton, R. Rivest, C. D. Thompson, U. Vazirani, and V. Vazirani, "Global wire routing in two dimensional arrays," in *Proc. Ann. Symp. Foundations of Computer Science*, 1983, pp. 453–459.

[8] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, and others "Hopfield Networks is All You Need," *arXiv*, preprint arXiv:2008.02217,2020.

[9] A. Dees Jr. and Patrick J. Karger, "Automated Rip-Up and Reroute Techniques," *IEEE 19th Design Automation Conference*, 1982.

[10] T. Yoshimura and E. S. Kuh,"Efficient algorithms for channel routing," *IEEE Trans. on Computer-Aided Design*, 1982.

[11] H. H. Hoos and T. Stützle, "Stochastic Local Search: Foundations and Applications," *Morgan Kaufmann,*, 2004.

[12] D. P. Bertsekas, "Constrained Optimization and Lagrange Multiplier Methods," *Academic Press,*1982.