Applying the RA-NN Formulation to PCI Optimization

Correct Formulation Makes the Network Learnable

ABSTRACT

PCI optimization in LTE and NR networks is widely formulated as graph coloring: minimize collisions; minimize colors used. This formulation is wrong in three ways: the wrong graph, the wrong objective, and the wrong evaluation. The actual operational problem is PCI confusion, a 2-hop property where a serving cell cannot distinguish between neighbors sharing the same PCI when processing UE measurement reports, causing handover failures. Minimum graph coloring does not merely ignore confusion; it guarantees it, packing 1,500 cells into 14–17 colors when 504 are available and guaranteeing that every node is confused. We formulate the network as a sparse interaction graph built from UE measurement reports — the RA-NN representation. When the network is formulated correctly, the information needed to resolve every confusion is already present in the graph's 2-hop neighborhood structure. The resolution procedure extracts it directly through message passing: propagate PCI counts through 2-hop neighborhoods, identify violations, reassign to safe values. The procedure converges monotonically, executes in under 0.1 seconds on a 1,500-cell network, and changes only the cells that need to change. The simplicity is the correct formulation.

Michael Chiaramonte
Cognitive Network Solutions
March 6, 2026

1. INTRODUCTION

Every cell in an LTE or NR network broadcasts a Physical Cell Identity (PCI), a numeric identifier that UEs use to distinguish cells during measurement, handover, and connection setup. LTE defines a pool of K=504 PCIs; NR extends this to K=1008. When PCIs are poorly assigned, two problems arise.

PCI Collision (1-hop property): Two adjacent cells share the same PCI. A UE at the boundary cannot distinguish between them.

PCI Confusion (2-hop property): A cell has two or more neighbors that share the same PCI with each other. When a UE camped on that cell reports measurements referencing that PCI, the serving cell cannot determine which neighbor the report refers to. As a result, it cannot distinguish which cell the UE should hand over to. This will cause many handover failures and dropped calls in the area with the confusion. Formally, node v is confused when $\gamma(v, c) \geq 2$ for any PCI value c, where $\gamma(v, c)$ counts how many of v's neighbors use PCI c. This γ notation follows the standard formulation also used in [1].

Collision is worse for the individual UE at the boundary, but in cellular networks (which are locally dense by design), any collision between two cells causes confusion at every shared neighbor. Adjacent cells in a well-planned network always share common neighbors; if two cells are close enough to interfere, the cells between them see both. Resolving all confusions resolves all collisions as a byproduct. The optimization target is zero confusions.

Despite this, the literature overwhelmingly formulates PCI optimization as graph coloring: minimizing the chromatic number or eliminating edge conflicts, which addresses only collisions. Farghaly et al. [1] propose DSATUR and MP-BRKGA, explicitly claiming to handle "PCI collision and confusion constraints" while encoding only collision constraints. This mismatch is not merely an omission; it is adversarial: minimum graph coloring guarantees confusion when the PCI pool greatly exceeds the chromatic number.

This paper demonstrates that the correct formulation makes this optimization task trivial. We apply the RA-NN (Radio Access Neural Network) formulation, a neural systems approach that models the radio access network as a physical interaction graph built from UE measurement reports [2], to PCI optimization. When the network is represented correctly and the objective targets the actual problem, the solution is a straightforward message passing on the network graph. No heuristic search, no genetic algorithms. The information needed to resolve every confusion is already present in the graph's 2-hop neighborhood structure; the procedure extracts it directly rather than searching for it.

2. THE CONFUSION PROBLEM

The γ matrix is the central object in PCI optimization. For each node v and PCI value c, $\gamma(v, c)$ equals the number of v's neighbors using PCI c. This is computable in $O(E)$ from the edge list. The key components to extract from γ become:

- **PCI Collision** is $\gamma\big(v, a(v)\big) \geq 1$: a neighbor shares v's own PCI.
- **PCI Confusion** is $\gamma(v, c) \geq 2$ for any c: v has two or more neighbors sharing the same PCI, regardless of whether it matches v's own.

Computing γ from the edge list is a single pass over the directed edges. For each directed edge (u, v) in the graph, increment γ[v, PCI(u)] by 1. This produces the complete N × K matrix in O(|E|) time. A node v is confused if any entry γ(v, c) ≥ 2. Total network confusion is the count of nodes where $\max_2$ γ(v, c) ≥ 2, a single reduction over the matrix. The entire confusion state of the network is captured in this one data structure, and updating it after a PCI change requires touching only the changed node's edges.

The objective is an assignment where $\gamma(v, c) < 2$ everywhere. When this holds, collisions are zero as a byproduct.

Graph coloring produces a collision-free assignment: no edge has the same color on both endpoints. But collision-free says nothing about confusion. A proper coloring freely assigns the same PCI to non-adjacent nodes, including non-adjacent neighbors of the same node, because no edge constraint prevents it. In networks where the PCI pool K far exceeds the graph's chromatic number $\chi(G)$, proper coloring uses far fewer colors than available. The unused color slots guarantee massive PCI reuse among non-adjacent neighbors.

3. WHY GRAPH COLORING FAILS

Farghaly et al. [1] define both collision and confusion correctly (page 5, Fig. 1) and claim to "adapt BRKGA to handle PCI collision and confusion constraints" (page 3). None of the proposed algorithms encode confusion constraints:

Algorithm	Constraint encoded	What it actually solves
DSATUR	Minimum proper graph coloring	Collision only: greedy color assignment avoiding edge conflicts
MP-BRKGA	Neighboring cells cannot share the same PCI	Collision only: fitness based on edge conflict cost

No algorithm checks whether $\gamma(v, c) \geq 2$ for any node. Their ILP formulation has an objective function of "Minimize: 0" (page 7); it searches for any feasible coloring satisfying the constraints, not an optimal one. The paper describes Greedy as the baseline that DSATUR improves upon (page 5) but never benchmarks Greedy. As our results show, Greedy achieves the same outcome (zero collisions, maximum confusion) in 0.033 seconds, making the entire DSATUR/BRKGA comparison moot.

The consequences of this mismatch are not subtle. Consider a production LTE network: $K = 504$, maximum degree $\Delta = 56$. DSATUR produces a valid coloring using only ~15 colors, the chromatic number $\chi(G)$ of this graph (confirmed by our benchmark). A node with 56 neighbors colored from 15 colors has on average $\lceil \frac{56}{15} \rceil \approx 4$ neighbors per color. $\gamma(v, c) \geq 2$ for nearly every color c at every dense node. Every node in the network is confused. The optimal strategy for confusion is the opposite of minimum coloring: spread assignments across as many of the 504 available colors as possible:

Strategy	Colors used	Expected γ per color	Result
Minimum coloring (DSATUR)	15 of 504	~4	Every node confused
RA-NN formulation	All 504	0	**Zero confusion**

PCI optimization is not a chromatic number problem.

There is one scenario where collision-free implies confusion-free: when local neighborhoods are cliques. In a clique of size d, proper coloring requires d distinct colors for all neighbors, so $\gamma(v, c) \leq 1$ for all c, meaning no confusion. But the chromatic number disproves this. If neighborhoods were cliques, $\chi(G)$ would be $\Delta + 1 = 57$. DSATUR achieving $\chi(G) = 15$ proves neighborhoods are extremely sparse, far from cliques. That sparsity is exactly why graph coloring freely reuses colors among non-adjacent neighbors, creating confusion. The assumption is self-contradictory: the fewer colors minimum coloring needs, the sparser the neighborhoods, and the more confusion it creates.

Finally, Farghaly et al. evaluate on SINR distribution via NS-3 simulation (pages 12–24) and never count collisions or confusions directly. A good PCI plan is an interference mitigation plan; it directly affects SINR by eliminating PCI-induced interference (collisions, confusions, mod-3/mod-6 reference signal conflicts). SINR is a valid way to observe the effect. But there is no reason to measure the effect when you can solve the cause directly. A PCI plan with zero confusions that satisfies mod constraints has already extracted the maximum possible SINR gain from PCI planning. Every source of PCI-induced interference is eliminated. There is nothing left for PCI optimization to improve; any remaining SINR degradation comes from non-PCI sources (propagation, fading, load, scheduling) that require different solutions entirely. Measuring SINR to evaluate a PCI plan is evaluating a proxy for a quantity you can compute exactly.

4. THE RA-NN FORMULATION

RA-NN (Radio Access Neural Network) is a neural systems formulation that models the radio access network as a sparse interaction graph built from UE measurement reports [2]. Rather than treating the network as a collection of independent cells with theoretical propagation models, RA-NN constructs the graph from what UEs report, the ground truth of how the network is experienced by the devices it serves. The full formulation is described in [2]; we summarize the elements relevant to PCI optimization here.

The system ingests UE measurement reports (RSRP, RSRQ, and other measurements reported back to the network), aggregates them through a GPU-accelerated processing pipeline, and produces a sparse interaction graph where nodes are cells carrying their current configuration, and edges represent observed interference: two cells are neighbors if UEs report measurable signal from both. Edge weights reflect the strength and frequency of co-observation.

The graph source matters. Farghaly et al. build their interference graph from NS-3’s Automatic Neighbor Relations (ANR) with a fixed RSRQ threshold (page 12). Fixed-threshold ANR systematically fails in two ways. Over-shooting cells, misconfigured tilts, and multipath reflections create neighbor relations between geographically distant cells; a loose threshold injects long-range edges into local clusters. Conversely, ANR only discovers a relation when a UE performs a handover across that boundary, so low-traffic boundaries generate no entry even though interference exists; a snapshot misses intermittent interference.

Farghaly et al. also introduce “top-neighbor” pruning as a contribution, keeping only the most influential neighbors per cell to improve scalability (page 3). This deliberately removes edges from the graph. Any pruned edge could participate in a confusion that the algorithm never sees. Pruning the neighbor list for algorithmic convenience is the opposite of what PCI planning requires: the full interference picture. If the algorithm cannot scale to the complete neighbor list, the algorithm is the problem, not the graph.

The RA-NN interaction graph captures actual interference geometry as experienced by users: real coverage footprints and time-varying conditions. The only filtering applied is a physical one: UE measurements below −124dBm are excluded, matching the lowest practical Qrxlevmin setting, the threshold below which the UE itself does not report or act on the signal. This is not an arbitrary tuning parameter; it is the physical boundary of what the network can observe.

When the network is correctly formulated as a sparse interaction graph from UE measurements, and the objective correctly targets confusion, the resolution procedure is straightforward message passing. The concrete steps are given in Algorithm 1.

Algorithm 1: PCI Confusion Resolution

```
Input: graph edges E, current PCI assignment a, PCI pool size K
Output: updated assignment a with zero confusions

1: repeat until no confused nodes remain:
    2: Build $\gamma$ ($N \times K$):
        3: for each edge (u, v):
            4: $\gamma[v, a[u]] \leftarrow \gamma[v, a[u]] + 1$
    5: Build 2-hop PCI visibility ($N \times K$):
        6: for each edge (u, v):
            7: $\text{visibility}_2\text{hop}[v] \leftarrow \text{visibility}_2\text{hop}[v] + (\gamma[u] > 0)$
        visibility₂hop[v, c] counts how many of v's neighbors have at least one neighbor using PCI c
    8: Identify nodes to change:
        A node is an issue if its current PCI causes confusion at one or more neighbors,
        contributing to $\gamma \geq 2$ at an adjacent node
    9: for each issue node v:
        10: candidate_set $\leftarrow \{c \in 0..K-1 : \text{visibility}_2\text{hop}[v, c] = 0\}$
        11: a[v] $\leftarrow$ select from candidate_set
        Any PCI absent from the entire 2-hop neighborhood is guaranteed safe:
        assigning it can increment $\gamma$ at direct neighbors from 0 to 1, never reaching 2

Typical convergence: 5–15 iterations
```

The information needed to resolve every confusion is already present in the graph. The 2-hop neighborhood structure tells you exactly which nodes are confused, exactly which PCIs cause it, and exactly which replacements are safe. There is no solution space to search; the graph encodes the answer, and the procedure extracts it.

The result converges monotonically, requires no random initialization, no population management, no tabu tenure, no fitness function. It runs in 103ms on a 1,500-cell network.

This is the central argument: graph coloring approaches require complex heuristic search (DSATUR, BRKGA, TabuCol) and still fail, not because they lack algorithmic sophistication, but because they start from the wrong formulation. Wrong graph, wrong objective, no visibility into the 2-hop structure that confusion requires. No amount of algorithmic cleverness compensates for formulating the problem incorrectly. When the formulation is correct, the solution does not require searching; it requires reading.

5. PROPERTIES

5.1 CONVERGENCE

The procedure's correctness rests on one property: any PCI absent from a node's 2-hop neighborhood is a safe replacement. Assigning such a PCI can only increment γ at direct neighbors from 0 to 1, below the confusion threshold of 2. No single reassignment to a 2-hop-absent PCI can create new confusion. Combined with throttling that prevents multiple simultaneous changes from interacting, each iteration strictly reduces the total confusion count. Since confusion count is a non-negative integer that strictly decreases, the procedure terminates.

On the experimental scenarios, convergence is rapid. Scenario A (7 confusions) converges in 3 iterations. Scenario B (815 confusions) converges in 8 iterations. Each iteration reduces the confusion count strictly: no iteration leaves the count unchanged or increases it. The monotonic decrease is not a statistical tendency; it is a structural guarantee of the 2-hop safe selection criterion.

The procedure was evaluated on 100 independently generated starting assignments on the same network graph, using four generation strategies: random uniform PCI assignment, limited PCI pool (10–50 colors), balanced shuffle, and perturbation of the production assignment. Initial confusion counts ranged from 283 to 1,500 (every node in the network confused). All 100 converged to zero confusions. The hardest scenario required 199 iterations and 265ms. Scenarios A and B presented in Section 7 are representative of this range.

The procedure's convergence can be formally established through a potential function argument.

Define the potential function as:

$$\Phi = \Sigma v \in V \ \ \Sigma c \in [K] \ \ \max(0, \gamma(v, c) - 1) \quad (1)$$

the total excess of γ beyond the confusion threshold, summed over all nodes and PCI values. Φ is a non-negative integer. Each term is either zero ($\gamma(v, c) \leq 1$, no excess) or a positive integer (the amount by which γ exceeds the threshold). When $\Phi = 0$, every $\gamma(v, c) \leq 1$, meaning zero confusions and zero collisions.

Theorem 1: Monotonic Convergence

Theorem. Each iteration of Algorithm 1 strictly decreases Φ by at least 1. The procedure terminates with $\Phi = 0$ in at most Φ_0 iterations, where Φ_0 is the initial value of Φ.

Proof. Four properties establish the claim.

(1) Progress is always possible. Suppose node u is confused: $\gamma(u, c) \geq 2$ for some PCI value c. Then u has at least two neighbors with PCI c. Let v be any such neighbor. Since v has PCI c, every neighbor w of v satisfies $\gamma(w, c) \geq 1$ (v itself contributes). The 2-hop visibility of PCI c at v is

$$agg_2(v, c) = \Sigma_{w \in N(v)}\ \gamma(w, c) \quad (2)$$

Since $u \in N(v)$ and $\gamma(u, c) \geq 2$, while all other neighbors contribute at least 1:

$$agg_2(v, c) \geq (deg(v) - 1) \cdot 1 + 1 \cdot 2 = deg(v) + 1 > deg(v) \quad (3)$$

This exceeds the issue detection threshold. Therefore, whenever confusions exist, the procedure identifies at least one issue node and makes at least one change per iteration.

(2) Safe replacement cannot create confusion. When node v changes to a PCI q absent from its entire 2-hop neighborhood ($agg_2(v, q) = 0$), every direct neighbor w of v has $\gamma(w, q) = 0$ before the change. Afterward, $\gamma(w, q)$ increases from 0 to 1. Since $1 < 2$, no neighbor becomes confused on PCI q. No term in Φ increases.

(3) Each change reduces Φ by at least 1. From (1), every changed node v satisfies $agg_2(v, \text{old PCI}) > deg(v)$, so at least one neighbor u has $\gamma(u, \text{old PCI}) \geq 2$. When v drops its old PCI, $\gamma(u, \text{old PCI})$ decreases by 1. The contribution $\max(0, \gamma(u, \text{old PCI}) - 1)$ decreases by at least 1. Combined with (2), the net effect on Φ is at most -1 per reassignment.

(4) Simultaneous changes do not interact. Throttling selects at most one node per old PCI value; deduplication selects at most one node per new PCI value. Changes therefore operate on disjoint columns of the γ matrix on both the removal and addition sides. Cross-column interference is prevented by the 2-hop safety condition: if node A holds PCI p within 2 hops of node B, then $agg_2(B, p) > 0$, so p cannot appear in B's safe candidate set. No change adds a PCI that another change simultaneously removes within the same neighborhood. All changes in an iteration are independent, and their Φ reductions sum.

Combining (1)–(4): Φ is a non-negative integer that decreases by at least 1 each iteration. The procedure terminates with $\Phi = 0$ — zero confusions — in at most Φ_0 iterations. ■

The bound Φ_0 is conservative. Each change reduces γ at all neighbors of the changed node (not just one), and multiple independent changes occur per iteration. On the experimental graph, initial Φ ranged from 312 to 30,502 across the 100 test scenarios, yet convergence required only 6 to 199 iterations.

The procedure requires that at least one PCI is absent from each node's 2-hop neighborhood, meaning that K exceeds the number of distinct PCIs reachable within 2 hops. In any properly designed cellular network this condition is trivially satisfied: $K = 504$ (LTE) or $K = 1008$ (NR) far exceeds any realistic 2-hop neighborhood size. If a network were so dense that a node's 2-hop neighborhood consumed all 504 PCIs, the problem would not be PCI planning; it would be a fundamental coverage design failure where the physical network has more overlapping cells than the PCI space can support, requiring physical intervention (tilt adjustment, power reduction) before any PCI algorithm can help.

5.2 MODULO CONSTRAINTS (MOD-3, MOD-6, MOD-30)

PCI planning involves additional constraints beyond collision and confusion. PCI mod-3 determines the Primary Synchronization Signal (PSS), which must differ between co-sited sectors to maintain reference signal orthogonality. PCI mod-6 determines the Secondary Synchronization Signal (SSS) mapping, relevant in single-port and mixed MIMO/SISO configurations. PCI mod-30 affects uplink reference signal patterns, where adjacent cells sharing the same mod-30 value may face uplink decoding challenges.

Of these, mod-3 is the most operationally critical and the most straightforward to enforce. A 3-sector site has sectors 0, 1, and 2. Partition the network by sector ID into three independent subproblems, each operating on its own slice of the PCI pool: 168 PCIs per sector for LTE (504/3), 336 for NR (1008/3). The confusion resolution procedure runs identically on each partition; the mod-3 constraint is satisfied by construction since no two co-sited sectors draw from the same pool. This is a data partitioning step, not an algorithmic change.

Mod-6 and mod-30 are additional constraints that further subdivide the PCI space. PCFICH (Physical Control Format Indicator Channel) collisions are a related concern: neighboring cells whose PCIs differ by a multiple of 50 (for 20 MHz channels) use the same PCFICH resource mapping, potentially causing control channel decoding failures. Farghaly et al. [1] acknowledge these constraints (page 5) but do not enforce them in their algorithms either; their ILP formulation includes a mod-3 constraint (Eq. 8) but comments out the SSS constraint as infeasible (Eq. 9, page 7). All of these constraints are enforceable through the same mechanism: restrict each node’s candidate PCI pool to values satisfying the required modulo residues and PCFICH separation from its neighbors. The confusion resolution procedure remains identical; only the candidate set narrows.

5.3 COMPLEXITY

Variable	Meaning
N	Cells (nodes) in the graph
E	Directed edges (neighbor relations)
K	PCI pool size (504 LTE, 1008 NR)
T	Iterations to convergence (5–15 in practice)
G, P	BRKGA generations and population size

Farghaly et al. also evaluate an Integer Linear Programming (ILP) formulation as an optimal benchmark [1, Table 1]. ILP exhibits exponential complexity in the number of cells, $O(3^n)$, and the authors themselves note it is impractical for networks exceeding 100 cells. This graph has 1,500 cells. We do not benchmark ILP because it cannot execute at this scale; its inclusion would demonstrate only that exact methods are infeasible for production networks, which is already established.

Algorithm	Complexity	Solves
ILP	$O(3^n)$ — exponential [1]	Collision only (infeasible at $N > 100$)
DSATUR	$O(N^2 \times K)$ [1]	Collision only
BRKGA	$O(G \times P \times N^2)$ [1]	Collision only
RA-NN	$O(T \times E \times K)$	Confusion (and collision)

RA-NN scales linearly with graph size. In sparse graphs, which cellular networks are by physical locality, edge count E grows linearly with node count N. A 10x larger network (15,000 cells, ~480K directed edges) runs in approximately 1 second. DSATUR's quadratic scaling would require approximately 1,650 seconds (~27 minutes) at the same scale. BRKGA scales linearly in N but generations and population constants may need to increase for convergence on larger graphs.

6. EXPERIMENTAL SETUP

6.1 NETWORK GRAPH

All experiments use a 1,500-cell LTE network (500 three-sector sites) simulated via Sionna at production scale, representative of a real 500-site cluster. The simulation generates per-UE measurement reports (RSRP per detected cell) for 50,000 users, formatted to match 3GPP measurement report structure. The interaction graph is constructed from these simulated reports using the same RA-NN pipeline that would process real network data: if a UE reports measurable signal from two cells, those cells are neighbors. This produces a graph grounded in physical signal propagation, not arbitrary thresholds.

The scale difference is substantial: 1,500 cells vs. 96 in the largest scenario of [1], over 15× more nodes. Each eNodeB in [1] is a single cell on a rectangular grid; this network uses 500 three-sector sites. But the real gap is in edge count. PCI algorithm complexity is driven by the interference neighborhood structure, and this graph has 48,320 directed edges, orders of magnitude beyond what a 96-node grid topology produces. Farghaly et al.'s scalability analysis already shows diminishing or negative returns at 96 eNBs (ILP degrades SINR by 5.5%).

Dense urban networks represent the most challenging case for PCI optimization due to high node degree and neighborhood overlap. Suburban and rural topologies have strictly lower degree, making the 2-hop safe candidate pool larger and the problem easier. Demonstrating convergence on the hardest topology class generalizes to all others.

Property	Value
Cells (Nodes)	1,500

Neighbor Relations (Edges)	48,320
Average degree	32.2
Maximum degree	56
PCI pool	504

6.2 SCENARIOS

Two scenarios test the procedure across operating conditions. **Scenario A (Production)** starts from a near-optimal PCI assignment: 7 confusions, 0 collisions, the operational case of a well-planned network with residual confusions. **Scenario B (Stress Test)** starts from a deliberately degraded assignment: 815 confusions, 46 collisions, testing robustness under severe conditions.

Both use the same retune protocol: start from the given assignment and improve it while minimizing cells changed, since each PCI change usually requires a temporary service interruption. Greenfield planning (initial deployment) is the same problem: start from any assignment, including random, and retune. The distinction is operational, not algorithmic.

6.3 ALGORITHMS

We benchmark RA-NN against five established graph coloring algorithms on identical graph topologies with identical starting assignments. The baselines span the algorithmic spectrum: Greedy (sequential first-available-color, deterministic baseline), DSATUR (saturation-degree ordering, Python, 10 random orderings), DSATUR (C++) (same algorithm, native compiled), TabuCol (tabu search, 50,000 max iterations), and BRKGA (multi-population genetic algorithm, 200 generations, population 100). Deterministic algorithms run once; stochastic algorithms are run on 10 independent seeds, reporting mean ± standard deviation.

7. RESULTS

7.1 SCENARIO A: PRODUCTION NETWORK (7 CONFUSIONS, 0 COLLISIONS)

Algorithm	Time	Confusions	Collisions	Changes	Colors
RA-NN	**0.090s**	**7 → 0**	**0 → 0**	**7 (0.5%)**	**504**
TabuCol	0.012s	7 → 7	0 → 0	0 (0.0%)	504
Greedy	0.033s	7 → 1,500	0 → 0	1,465 (97.7%)	17
DSATUR (C++)	0.103s	7 → 1,500	0 → 0	1,470 (98.0%)	15
BRKGA	1.75s	7 → 608	0 → 0	1,130 (75.3%)	477
DSATUR	16.5s	7 → 1,500	0 → 0	1,470 (98.0%)	14

7.2 SCENARIO B: STRESS TEST (815 CONFUSIONS, 46 COLLISIONS)

Algorithm	Time	Confusions	Collisions	Changes	Colors
RA-NN	**0.103s**	**815 → 0**	**46 → 0**	**204 (13.6%)**	**481**
Greedy	0.033s	815 → 1,500	46 → 0	1,460 (97.3%)	17
DSATUR (C++)	0.102s	815 → 1,500	46 → 0	1,460 (97.3%)	15
TabuCol	0.342s	815 → 703	46 → 0	42 (2.8%)	480
BRKGA	1.77s	815 → 608	46 → 0	1,155 (77.0%)	477
DSATUR	16.7s	815 → 1,500	46 → 0	1,457 (97.1%)	14

7.3 INTERPRETATION

The results are separated into three categories: algorithms that solve collisions and destroy the network, an algorithm that solves collisions and ignores confusion, and the RA-NN formulation that solves the actual problem.

Greedy already solves the paper's problem: zero collisions in 0.033 seconds. DSATUR, DSATUR (C++), and BRKGA are slower paths to the same collision-free outcome. The entire comparison in [1] is moot.

Every minimum-coloring algorithm makes confusion catastrophically worse. Greedy, DSATUR, and DSATUR (C++) replan the entire network into 14–17 colors (changing 97–98% of cells) and produce exactly 1,500 confusions, every node in the network, on both scenarios. 1,500 cells packed into 15 colors when 504 are available. This is not a failure of implementation. It is the pigeonhole principle operating exactly as predicted: $\lceil \frac{56}{15} \rceil \approx 4$ neighbors per color at every dense node.

BRKGA uses 477 colors and still fails. Its genetic search spreads PCIs more than DSATUR, but without a confusion objective it produces 608 confusions. Undirected color spreading is not enough; confusion resolution requires 2-hop awareness that only the correct formulation provides.

TabuCol illustrates the collision-confusion gap precisely. On Scenario A (0 collisions), TabuCol detected no collision violations, made zero changes, and terminated. The 7 confusions were invisible to it; they are not part of its objective function. A collision-free network can still have confusions, and a collision-targeted algorithm has no signal to act on them. On the stress test, it removes all 46 collisions but confusions only drop from 815 to 703 as a byproduct.

The RA-NN solves both scenarios to completion with every confusion and collision eliminated. Scenario A solves in 90ms with 7 PCI changes (0.5%), while Scenario B solves in 103ms with 204 PCI changes (13.6%). The execution times are nearly identical: 0.090s vs. 0.103s. A 100x harder problem takes the same time because the procedure reads the same graph; only the number of iterations changes slightly.

The execution time is dominated by the graph traversal in the γ construction and 2-hop visibility steps, not by the number of confused nodes. Scenario A requires 3 iterations over the same graph; Scenario B requires 8. The per-iteration cost is identical because the graph structure is unchanged; only the number of passes differs, which accounts for the near-identical wall times (0.090s vs. 0.103s). A 100× increase in confusion count adds five iterations of the same O(|E|) traversal.

Most traditional graph coloring algorithms will perform a full replan, changing 97–98% of all cells. If the network is already deployed, a PCI change is a service-impacting event. This is because changing a cell's PCI requires taking the cell out of service, applying the new identity, and bringing it back online. Changing 98% of cells means systematically taking down all cells in the network to implement this plan. RA-NN changes 0.5% in production (7 cells) and 13.6% under stress (204 cells). That is the difference between a surgical correction and a network-wide service interruption.

8. DISCUSSION

DSATUR and BRKGA are well-studied, competently engineered methods of graph coloring. However, they fail here because they are solving the wrong problem on the wrong graph.

When the RA-NN formulation provides the correct interaction graph (from UE measurements, not ANR), targets the correct objective (confusion, not collision), and evaluates the correct metric (confusion count), the ideal optimization procedure emerges naturally. The graph's 2-hop neighborhood structure already contains all the information needed: which nodes are confused, which PCIs cause it, and which replacements are safe.

This is a general property of the RA-NN approach: by internalizing the full picture of the network (real interference geometry, actual coverage footprints, the complete neighborhood structure) into the interaction graph, downstream applications operate with simple, fast procedures instead of brute-force heuristic search. The computational work moves from the algorithm to the formulation.

The relationship between chromatic number and confusion creates a fundamental paradox for graph coloring approaches. If $\chi(G)$ approaches $\Delta + 1$, neighborhoods are dense and proper coloring assigns distinct colors to most neighbors, minimizing confusion. If $\chi(G)$ is much less than $\Delta + 1$, neighborhoods are sparse, proper coloring reuses colors heavily, guaranteeing confusion. Real cellular networks are the second case. This simulated deployment had $\Delta = 56$ but $\chi(G) = 15$. The sparser the neighborhoods, the more "efficient" minimum coloring is at reusing colors, and the more confusion it creates. The fewer colors minimum coloring needs, the worse it is for PCI optimization. This is not fixable with better heuristics. It is a fundamental incompatibility between traditional graph coloring and PCI planning.

Three operational requirements apply when optimizing a deployed network, regardless of use case. Speed: deployed networks change continuously, and an optimizer must execute fast enough to respond to real conditions. Minimal disruption: configuration changes can sometimes cause service disruptions, so the optimizer must change only what needs changing; replanning from scratch is not an option when the network is live. Convergence: the optimizer must converge toward the best achievable solution given all constraints, not leave residual violations that require manual cleanup. RA-NN satisfies all three: sub-second execution, minimal configuration changes, and zero confusions and collisions.

PCI optimization requires no learned parameters because the interaction graph alone is sufficient. This is by design: the RA-NN formulation establishes a representation in which the network's physical interaction structure is correct before any learning begins. Downstream tasks that require generalization — mobility optimization, interference management, capacity planning — operate on the same substrate with learned components. PCI demonstrates that the substrate itself is sound.

PCI planning is inherently a per-frequency-layer problem. Interference is intra-frequency: two cells on different carriers do not interfere via PCI. A production network running multiple LTE bands and NR carriers has one independent PCI planning problem per frequency layer. The RA-NN interaction graph naturally captures this: edges for PCI optimization can be scoped for cells in the same frequency layer to run the same procedure independently on each layer’s intra-frequency graph simultaneously. Proving the approach on one frequency graph proves it for all of them.

Production networks are not static. Cells are deployed, decommissioned, and reconfigured. Coverage footprints shift with seasonal foliage, construction, and equipment upgrades. The RA-NN formulation is built on a dynamic coverage substrate; the interaction graph is continuously reconstructed from live UE measurement reports. When the graph structure changes, the confusion resolution procedure can re-execute in sub-second time. This is by design: the formulation separates the slow, continuous process of building an accurate interaction graph from the fast process of extracting the solution from it. The graph absorbs the complexity of the real world; the algorithm remains trivial regardless of what changes.

9. CONCLUSION

The PCI optimization problem has been misformulated as graph coloring. Three errors compound:

- **The wrong graph:** ANR with fixed thresholds systematically over- and under-reports neighbors. The RA-NN formulation constructs the interaction graph from UE measurements, the ground truth of how the network is experienced.
- **The wrong objective:** Graph coloring minimizes collisions (1-hop). The operational problem is confusion (2-hop). Minimum coloring is adversarial: $\chi(G) = 15$ colors for $K = 504$ on $\Delta = 56$ guarantees $\gamma \approx 4$ at dense nodes. Every node confused.
- **The wrong evaluation:** SINR measures the effect of PCI-induced interference, but a zero-confusion plan has already maximized the SINR gain from PCI planning. Solve the cause directly; confusion count is exact and computable in $O(E)$.

When all three are corrected, the information needed to resolve every confusion is already present in the graph. The 2-hop neighborhood structure tells you which nodes are confused, which PCIs cause it, and which replacements are safe; the resolution procedure simply extracts this through message passing. It converges monotonically, executes in under 0.1 seconds, and changes only the cells that violate constraints. The algorithm is trivial because the formulation is correct. The comparison in [1] is moot even on its own terms: Greedy, the baseline they mention but never benchmark, achieves zero collisions in 0.033 seconds. DSATUR C++ (0.102s) and BRKGA (1.75s) are slower paths to the same outcome. And that outcome (zero collisions, 1,500 confusions) is worse than the starting state.

REFERENCES

[1] Farghaly, S.I., Khayal, H.M., Algohary, I.M., et al. “Enhancement of LTE and NR systems through efficient physical cell identity allocation.” Scientific Reports 16, 5626 (2026). https://doi.org/10.1038/s41598-026-36608-w

[2] Cognitive Network Solutions. “RA-NN: A Neural Systems Formulation for Radio Access Networks.” Technical whitepaper, 2026.