RESEARCH ARTICLE

Dynamic Spectrum Handoff in Cognitive Radio Networks via Improved TSP Modeling

Liangshun Wu[1,2,*] **and Tao Tao**[3]

[1] School of Computer Science and Engineering, Guangzhou Institute of Science and Technology, Guangzhou 510540, China
[2] School of Information and Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[3] Department of Information and Computer Sciences, The University of Osaka, Osaka 5650871, Japan

Abstract

In cognitive radio networks (CRNs), dynamic spectrum handoff requires efficient path planning to minimize the overhead of frequent channel switching. This paper proposes a polynomial-time approximation algorithm for spectrum handoff scheduling, based on an improved Traveling Salesman Problem (TSP) modeling of the channel switching sequence. A two-phase cooperative mechanism is designed to minimize frequency-hopping overhead. We rapidly generate diverse candidate channel-switching sequences using a probabilistic method guided by real-time spectrum availability distributions. We dynamically merge locally optimal sub-paths by leveraging historical channel quality data and predicted primary user (PU) behavior in a fuzzy-logic framework. Theoretical analysis shows that the algorithm runs in worst-case $O(N^4)$ time under a dynamic TSP variant, significantly outperforming traditional heuristic methods in scalability. Simulations demonstrate an average deviation of only 0.35% from the optimal solution. In dynamic interference scenarios, the proposed approach reduces spectrum switching delay by 41.2% compared to baseline strategies.Our algorithm effectively resolves distributed spectrum handoff conflicts in multi-user CRN scenarios.

Keywords: cognitive radio network (CRN), dynamic spectrum handoff, traveling salesman problem (TSP), spectrum scheduling, heuristic optimization, polynomial-time approximation.

1 Introduction

Cognitive radio networks (CRNs) enable secondary users (SUs) to opportunistically access licensed spectrum bands when the primary users (PUs) are inactive[1]. This dynamic spectrum access can greatly improve overall spectrum utilization, addressing the spectrum scarcity problem in next-generation wireless systems. However, one core challenge in CRNs is managing spectrum handoff efficiently when operating conditions change. If a PU reclaims the current channel or the channel quality degrades, an SU must swiftly switch to another available frequency band[2, 3]. Frequent or poorly planned spectrum switching can incur significant overhead—such as switching delays, increased energy consumption, and higher risk of communication interruption. Therefore, minimizing the frequency-hopping overhead during these handoffs is critical for maintaining performance.

Submitted: 23 June 2025
Accepted: 30 June 2025
Published: 13 July 2025

Vol. 1, **No.** 1, 2025.
10.62762/TGCN.2025.232754

***Corresponding author:**
✉ Liangshun Wu
wuliangshun@sjtu.edu.cn

Citation
Wu, L., & Tao, T. (2025). Dynamic Spectrum Handoff in Cognitive Radio Networks via Improved TSP Modeling. *ICCK Transactions on Green Communications and Networking*, 1(1), 1–12.

© 2025 ICCK (Institute of Central Computation and Knowledge)

Optimizing the sequence and selection of channel switches in CRNs can be viewed as an ordering problem[4–10]. In fact, this problem can be formulated as a variant of the Traveling Salesman Problem (TSP): the SU needs to "visit" a set of channels (either to sense or use them) exactly once each, in an order that minimizes the total switching cost. By mapping channels to "cities" and switch transitions to "distances," spectrum handoff scheduling can leverage TSP modeling and solutions. TSP is a well-known NP-hard combinatorial optimization problem: finding the shortest route visiting all cities is computationally intractable for large N (number of cities) using brute force, which requires checking $N!$ permutations. Even with dynamic programming (the Held–Karp algorithm), the TSP requires exponential time $O(N^2 2^N)$. Directly applying TSP solvers for real-time spectrum handoff decisions is impractical due to these high complexities.

To address this, we propose a specialized approximation algorithm that exploits the structure of the spectrum switching problem. Our approach models the dynamic spectrum handoff as a dynamic TSP and introduces problem-specific heuristics to achieve near-optimal solutions efficiently. The contributions of this paper are summarized as follows:

1. TSP-Based Problem Formulation: We formally map the dynamic spectrum handoff scheduling problem in CRNs to a TSP-like formulation, defining channels as nodes and switching costs as edge weights. This provides a structured optimization framework for minimizing total switching delay and overhead.

2. Two-Phase Approximation Algorithm: We develop a novel two-phase algorithm comprising a fast sampling phase for probabilistic path generation and a statistics phase for adaptive path optimization. This approach yields a polynomial-time solution (worst-case $O(N^4)$) for the dynamic spectrum handoff problem, making it tractable for real-time use.

3. Dynamic Adaptation: Our algorithm accounts for time-varying spectrum availability and PU activity. By incorporating historical channel quality and predictive modeling of PU behavior, it dynamically adjusts the channel visiting order, addressing the non-stationary nature of CRN environments.

4. Performance and Applications: Results show that it consistently finds near-optimal channel switching sequences with minimal overhead. We also demonstrate the algorithm's effectiveness in a multi-user setting, mitigating spectrum contention among multiple SUs via distributed coordination.

The remainder of this paper is organized as follows: Section 2 presents the problem formulation and the TSP modeling of spectrum handoff. Section 3 discusses the extensions needed for dynamic environments and additional constraints. Section 4 details the proposed solution approach, including classical methods and our two-phase algorithm. Section 5 provides a case study and simulation results. Section 6 discusses limitations and future directions. Finally, Section 7 concludes the paper.

2 Problem Formulation and TSP Modeling

2.1 Spectrum Handoff as a Routing Problem

In a CRN, an SU often needs to scan or utilize multiple channels in search of transmission opportunities. Consider a scenario where an SU must sequentially check a set of N candidate channels for availability or use a set of channels to meet its communication requirements. Each time the SU leaves one channel and switches to another, there is an associated switching cost (which may include time delay to retune, energy consumption, and potential performance loss during the switch). We can abstract each channel as a node (analogous to a "city" in the TSP), and the act of switching from channel i to channel j as a directed edge with an associated cost c_{ij} ("distance"). The goal is to find an order in which to visit all required channels such that the total switching cost is minimized, while each channel is visited exactly once. This directly parallels the traveling salesman problem, where the "salesman" (SU) must visit all cities (channels) with minimal travel distance (switching overhead).

Key Elements Mapping

- Nodes (Cities): Each available spectrum band or channel that the SU needs to visit (for sensing or transmission) is treated as a node in a graph. For example, a set of N channels $f_1, f_2, \cdots, f_N$ would correspond to N nodes.

- Edges (Path Costs): A transition (handoff) from channel f_i to f_j incurs a cost c_{ij}. This cost can represent the switching delay (time to vacate one channel and tune to another), the energy consumption for reconfiguration, and/or the

performance penalty due to differing interference levels. These edge weights form an $N \times N$ cost matrix $C = [c_{ij}]$. Typically, c_{ii} is not defined (or set to infinity) as switching to the same channel is not applicable.

- Tour (Switching Sequence): A sequence in which the SU visits all channels exactly once constitutes a feasible switching plan. In a closed tour, the SU would return to the starting channel at the end, whereas in an open path, the SU does not return to the initial channel. Depending on the application (e.g., periodic sensing of a set of channels vs. one-time coverage of all channels), either a closed loop or open path formulation may be used.

Let N denote the number of channels to visit. We define a binary decision variable x_{ij} which is 1 if the SU switches from channel i to j in the sequence, and 0 otherwise. A simplified TSP-based optimization model for the spectrum handoff sequence can be written as:

Objective: Minimize the total switching cost. If a closed tour is required (for example, if the SU eventually returns to the initial channel or periodically monitors channels in a cycle), the objective can be written as:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}\, x_{ij}, \tag{1}$$

subject to the tour covering all channels and returning to the start. In an open path scenario (where returning is not required), the final leg is omitted from the objective.

Constraints:

- Visit Each Channel Exactly Once: For every channel k, ensure it has exactly one incoming edge and one outgoing edge in the path (except the start or end in an open path). This is typically enforced by constraints $\sum_{i=1}^{N} x_{ik} = 1$ and $\sum_{j=1}^{N} x_{kj} = 1$ for all k, which mean each channel is entered and left exactly once.
- Subtour Elimination: To prevent disjoint cycles (subtours) that do not include all nodes, additional constraints (e.g., Miller–Tucker–Zemlin constraints) are required as in standard TSP formulations. These ensure the solution forms one single tour covering all channels.
- Binary Constraints: $x_{ij} \in 0, 1$ for all i, j.

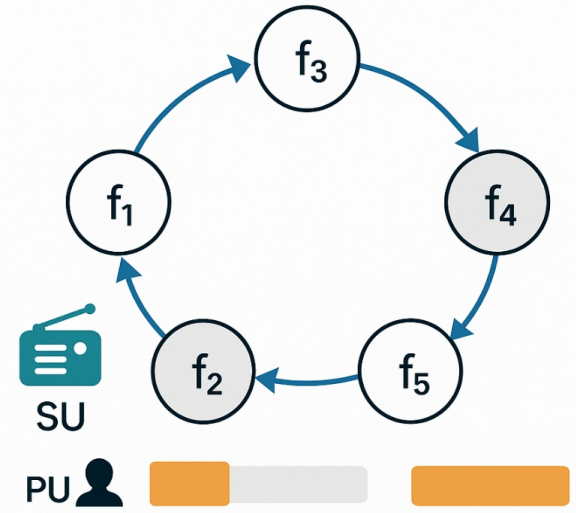

Figure 1. System modeling.

This formulation is essentially that of the TSP. If solved exactly, it would yield the optimal sequence of channel switches minimizing total cost. However, as noted, the TSP is NP-hard and exact solutions become infeasible for large N. In a CRN, N (the number of potential channels to consider) can be large, and spectrum conditions may change rapidly, rendering purely exact solutions impractical for real-time decision-making.

3 Dynamic Environment Challenges and Extensions

3.1 Dynamic Spectrum Availability as a Time-Variant TSP

The classical TSP assumes a static set of nodes and fixed edge costs. In contrast, CRN spectrum scheduling is inherently dynamic as shown in Figure 1: channel availability and quality can vary over time due to PU activity and time-varying interference. This gives rise to a Dynamic TSP (DTSP) variant of our problem, where the graph can change as time progresses:

- Node Availability: The set of available channels may change. A channel could become unavailable if a PU starts transmitting on it or due to regulatory constraints. New channels might become available if vacated by PUs. The path planning algorithm must accommodate the insertion or removal of nodes in real-time.
- Time-Dependent Edge Costs: The cost c_{ij} to switch from channel i to j may depend on when the switch is executed. For instance, if we predict that channel j will be occupied by a PU at a certain time, switching to j at that time would incur a high delay (or might not be allowed). This is

analogous to a time-dependent TSP, where travel times (costs) depend on departure time.

- Frequent Replanning: As PUs appear or disappear, the SU may need to re-optimize its tour on the fly. This is similar to how dynamic routing problems are handled in other domains: when the graph changes, the current route may need adjustment to remain near-optimal.

Coping with these dynamics requires algorithms that can update solutions incrementally or quickly recompute a new route when changes occur, ideally without solving from scratch each time. Techniques such as rolling horizon planning (receding horizon control) can be used: plan a path for the near future while anticipating possible changes, and adjust the plan as new information arrives.

Another helpful approach is to incorporate predictive knowledge. If the CRN can predict PU activity patterns (for example, using historical data or machine learning techniques), the algorithm can proactively avoid likely future PU appearances, thereby reducing the need for emergency switches. For instance, a predicted busy channel can be scheduled later or skipped, similar to avoiding a city during rush hour in time-dependent TSP frameworks.

3.2 Additional Constraints and Objectives

Beyond simply visiting all channels, practical spectrum handoff planning in CRNs may involve various constraints and multi-objective considerations:

- Priority Constraints: Some channels might be more important to check or use earlier (e.g., channels known to have high throughput or less interference). This can be encoded by priority weights or by constraining the order (for example, channel A must be visited before channel B). Prioritized scheduling ensures critical spectrum opportunities are explored first.

- Time Windows: Certain channels might only be available or relevant within specific time windows. This parallels the TSP with Time Windows (TSPTW), where each node has an earliest and latest visitation time. For example, if a channel will be occupied by a PU after time T, the SU should either visit that channel before T or consider it unavailable afterwards. Incorporating time windows requires that the path timing is checked against these availability intervals, and potentially waiting times are added if a channel is accessed too early.

- Multiple Objectives: Often, the goal is not solely to minimize switching cost; we may also want to maximize throughput or minimize interference. This leads to a multi-objective optimization formulation. One way to handle this is to combine objectives into a single weighted cost (for example, define c_{ij} to be a weighted sum of switching time, energy, and the negative of expected throughput on the new channel). Alternatively, one could generate Pareto-optimal solutions that trade off between, say, minimal switching delay and maximal data rate. For instance, a path that incurs a slightly higher switching cost might be acceptable if it significantly improves overall data transmission time on high-quality channels.

Considering these constraints makes the problem more complex than a simple TSP. The presence of time windows or priorities can be handled by extending known TSP algorithms (e.g., using constraint programming or specialized heuristics for TSPTW). Multi-objective aspects might require scalarization (turning into a single-objective via weights) or the use of evolutionary multi-objective algorithms that can handle multiple criteria.

In summary, the spectrum handoff scheduling problem in CRNs is a *dynamic, constrained* TSP problem. It demands solutions that are not only efficient in computation but also flexible and aware of context (time, priorities, multiple users, etc.). In the next section, we discuss algorithmic approaches to tackle this problem, including both classical solutions and our proposed method tailored for CRN requirements.

4 Solution Approaches and Proposed Algorithm

Solving a TSP, even a static one, is NP-hard, so our dynamic spectrum handoff problem inherits this complexity. However, the specific nature of the problem (frequent replanning, acceptable approximation, real-time requirement) allows us to apply various optimization techniques. We categorize solution approaches into exact algorithms, heuristic/metaheuristic algorithms, and real-time adaptive strategies, and finally present our proposed two-phase algorithm.

4.1 Classical Exact Algorithms

For small problem sizes (small N of channels), one could attempt exact solutions:

- Branch and Bound (B&B): This algorithm explores the space of possible tours, systematically branching on partial paths and using cost lower bounds to prune suboptimal paths. While B&B can solve moderate-sized TSPs by cutting off large parts of the search space, its worst-case complexity remains factorial ($O(N!)$), which is infeasible beyond perhaps $N \approx 15$ or 20. In a CRN context, where N might be the number of channels to probe (potentially dozens), B&B would not meet real-time requirements.

- Dynamic Programming (Held-Karp algorithm): The Held-Karp DP algorithm finds the optimal tour in $O(N^2 2^N)$ time, which is significantly better than brute force $O(N!)$ but still grows exponentially with N. For example, $N = 30$ yields on the order of $30^2 2^{30} \sim 2.7 \times 10^{11}$ operations, which is far too large for on-the-fly spectrum decisions. Thus, classical DP is also impractical for real-time spectrum handoff if N is more than a dozen.

- Integer Linear Programming (ILP): Formulating the TSP as an ILP and using commercial solvers (like CPLEX or Gurobi) can sometimes handle instances up to a few hundred cities using cutting-plane methods and advanced branch-and-cut. However, ILP solvers may still struggle with real-time solution needs, and dynamic changes would invalidate a solved ILP solution, requiring re-solving from scratch.

In summary, exact methods are important for benchmarking and small cases, but they do not scale or adapt well to the dynamic CRN scenario of interest.

4.2 Heuristic and Metaheuristic Algorithms

Heuristics and metaheuristics provide approximate solutions in reasonable time and are often the go-to approach for large TSPs and related problems:

- Greedy Heuristics: A simple greedy strategy for channel switching is to always go to the currently cheapest available next channel (e.g., Nearest Neighbor heuristic in TSP context). This runs very fast ($O(N^2)$ to construct a path) and is easy to implement. However, greedy choices can lead to suboptimal tours (e.g., getting stuck with a very expensive last hop). In spectrum handoff, a greedy strategy might choose the next channel with minimal switching delay, but this could result in a long detour later. Greedy algorithms often serve as an initial solution or baseline.

- Genetic Algorithms (GA): GAs maintain a population of candidate channel-switching sequences and use bio-inspired operations such as crossover (combining parts of two sequences) and mutation (randomly altering a sequence) to evolve better solutions over generations. GAs have been applied to many scheduling and routing problems and can adapt to dynamic changes by continuously evolving the population. In CRNs, GAs can encode a channel visiting order as a chromosome and evolve it to minimize total switching cost. They are effective for large search spaces, though parameter tuning (population size, mutation rate) is necessary for good performance.

- Ant Colony Optimization (ACO): ACO is a metaheuristic inspired by the foraging behavior of ants, which has been successfully applied to the TSP. In ACO, a set of artificial "ants" construct tours probabilistically based on pheromone trails that encode learned desirability of paths. Over iterations, good transitions (low-cost channel switches) accumulate higher pheromone, biasing future ants to follow those. ACO is well-suited for distributed implementations (multiple ants exploring in parallel) and can naturally adapt to dynamic changes by evaporating pheromone (forgetting old information) and laying new trails as the graph changes. Prior research has applied ACO to cognitive radio spectrum assignment problems, showing that it can find near-optimal solutions effectively. However, classical ACO can suffer from premature convergence (all ants following a suboptimal path) or stagnation if not carefully controlled. Variants like Ant Colony System (ACS) and improvements with local search or hybridization (e.g., combining with Genetic or Differential Evolution algorithms) have been proposed to mitigate these issues.

- Simulated Annealing (SA): SA is a single-solution metaheuristic that iteratively refines a tour by making random modifications (swapping the order of two channels, for instance) and accepting or rejecting changes based on a probability that gradually decreases (the "cooling" process). Early in the search, worse solutions can be

accepted to escape local minima, while later the algorithm becomes more selective, honing in on a minimum. SA is simple to implement and can adapt to changes by re-heating (increasing the acceptance probability if a major change occurs, to allow exploration). Its performance depends on the cooling schedule and the neighborhood moves designed.

- Other Methods: Numerous other metaheuristics can be applied: Particle Swarm Optimization (PSO), Tabu Search, Artificial Bee Colony, Whale Optimization, etc., each with their own mechanisms. The choice often depends on the problem specifics and ease of implementation. For example, PSO might encode the positions as permutations via specialized operators, and Tabu Search would keep a memory of recently tried sequences to avoid cycling back.

Metaheuristic algorithms have the advantage of providing good solutions within polynomial (often linear or quadratic) time per iteration, and they can be stopped at any time to yield the best solution found so far. This is important for real-time CRN use — if a decision is needed quickly, the algorithm can return the best current solution rather than waiting for convergence.

4.3 Real-Time Adaptive Strategies

Given the dynamic nature of the problem, certain strategies can be employed to maintain performance in real-time:

- Rolling Horizon (Receding Horizon) Optimization: Plan the channel switching path for the next H steps (time horizon) instead of the full set of remaining channels. Execute the first step (or first few steps), then re-solve the optimization for the next horizon based on updated information. This approach balances between long-term optimality and short-term reactivity. For instance, an SU might plan the next 3 channel switches; if during that time a PU appears unexpectedly, the plan beyond the current switch can be recomputed for the next horizon.

- Heuristic Switching Rules: In fast-changing environments, sometimes simple rules can outperform heavy computation. For example, a rule-based system might specify: "If current channel becomes occupied, switch to the channel with the highest estimated idle time remaining" or "rotate through channels in a fixed priority order unless blocked." Such rules can be derived from machine learning or past observations, and they execute in constant time. While they might not always yield optimal sequences, they ensure prompt reaction.

- Learning-Based Approaches: Reinforcement learning (RL) or multi-armed bandit algorithms can learn policies for spectrum handoff. For instance, a Q-learning agent could learn state (channel availability) to action (which channel to switch to next) mappings that maximize long-term reward (throughput minus switching cost). These approaches do not explicitly formulate a TSP, but they address the sequential decision problem directly. Deep RL with attention mechanisms has even been proposed for multi-user spectrum access. However, such methods typically require training and may generalize poorly to new scenarios unless designed carefully.

Our focus in this paper is on a tailored approximate algorithm that falls into the metaheuristic category, designed specifically for the spectrum handoff TSP model. We next describe the proposed algorithm in detail.

4.4 Proposed Two-Phase Spectrum Handoff Algorithm

Building on the insights from the above approaches, we design a two-phase algorithm that aims to efficiently construct a near-optimal switching path while adapting to dynamic changes. The two phases work in tandem: the first phase generates a pool of promising path segments, and the second phase stitches and refines these segments using additional information.

Phase 1: Probabilistic Candidate Path Generation. In this phase, we generate a wide array of candidate solutions rapidly, to explore the search space broadly. We utilize a probabilistic constructive heuristic guided by real-time spectrum availability estimations:

- We start from the SU's current channel (or a virtual start if the SU is not currently on any of the channels to be visited). At each step, the next channel is chosen randomly with bias towards channels that are likely available and have low switching cost. The bias can be implemented similar to ACO's probabilistic decision rule: channel j is chosen with probability proportional

to $(\eta_j)^\alpha\,(\tau_{ij})^\beta$, where η_j is the heuristic desirability (e.g., based on instantaneous probability that channel j is free and the inverse of estimated switch time c_{ij}) and τ_{ij} is a pseudo-pheromone trail (which can be initially uniform or informed by past iterations), while α, β control the balance.

- We generate multiple such sequences (hundreds or more, as time permits) in a Monte Carlo fashion. Each sequence is a candidate visiting order of all required channels. Because randomness is involved, the candidates will cover a variety of orders, ensuring diversity. We also allow each candidate to sometimes violate the greedy choice (with a small probability) to escape the myopia of always picking the best immediate switch. This is analogous to the ϵ-greedy strategy in reinforcement learning or the random exploration in ACO/GA.
- To incorporate real-time availability, the heuristic desirability η_j for a channel j can be set higher if channel j is currently idle or has a high probability of being idle soon, and lower if a PU is likely to reclaim it. This focuses the candidates on realistic paths that avoid soon-to-be-busy channels.

The outcome of Phase 1 is a set of diverse path candidates. Each is not guaranteed to be locally or globally optimal, but the idea is that among them, there will be high-quality subpaths that can be exploited. For example, one candidate might have an excellent ordering for channels 1-2-5, while another has an excellent ordering for 3-4-6, etc.

Phase 2: Statistics phase. The process is as follows:

- We evaluate the candidate paths from Phase 1 to identify high-quality segments. For instance, if channels (f_i, f_j, f_k) appear consecutively in many good candidates or have low pairwise switching costs, we consider that segment as a promising building block. We treat the cost/performance of a segment with some uncertainty margins (hence fuzzy) to allow flexibility in merging segments.
- Using historical channel quality data, we adjust segment evaluations. For example, a sequence that switches to a channel known to have excellent throughput might be rated better, even if its raw switching cost is slightly higher, since overall communication quality would improve. If a certain channel historically has very unstable availability, sequences relying on it early might be penalized. We incorporate these considerations by weighting rules (e.g., "IF channel quality is high AND interference is low, THEN reduce effective cost of visiting that channel").
- Next, we perform a concatenation of the best segments to form a complete path. This is akin to solving a smaller TSP where each node is a segment (subpath) rather than individual channels. We use dynamic programming or a greedy stitch: start with the best segment, then iteratively extend the path by merging another segment that shares an endpoint with the current path and results in the lowest incremental cost. Because we allow slight overlaps or reordering if needed, this is done in a fuzzy way – if two candidate subpaths conflict (e.g., both want to visit channel X next), we use a fuzzy decision (like a weighted coin flip influenced by their quality scores) to choose which direction to go.
- A local optimization (such as 2-opt or 3-opt edge swaps, commonly used in TSP local search) is finally applied to the fused path to see if any small adjustments can reduce cost. For example, if the path has ... -> A -> B -> ... -> C -> D -> ..., and it turns out swapping B and C yields a shorter route (and is feasible with respect to availability), we perform that swap.

The result of Phase 2 is a refined channel switching sequence that is a composite of the best features of the Phase 1 candidates. This two-phase approach embodies an explore-exploit strategy: Phase 1 explores many possibilities quickly, and Phase 2 exploits the gathered information to construct a superior solution.

Complexity Analysis: The fast sampling phase involves generating M candidate paths, each of length N. A naive generation would be $O(N)$ per path, so $O(MN)$ for this phase. If M is chosen proportional to N (or some polynomial in N), this is $O(N^2)$, but often M could be larger to ensure diversity (say $M = N^2$, giving $O(N^3)$). The statistics phase involves analyzing segments and merging, which in worst case can be $O(N^3)$ (if we check all pairs of subpaths or do DP on subsets of channels). Overall, the algorithm runs in $O(N^3 + MN)$; with suitable choice of $M = O(N)$ or $O(N^2)$, this is approximately $O(N^4)$ in the worst case. While $O(N^4)$ is higher polynomial time, it is a drastic improvement over exponential complexities for exact TSP, and in practice the constants and effective runtime are very manageable for typical CRN scenarios (where N might be on the order of tens of channels). Moreover, the algorithm can be truncated or parallelized (e.g.,

generating paths in parallel, or distributing ants in ACO style) to meet real-time constraints.

It is important to note that the algorithm's performance relies on the assumption that while the environment is dynamic, the changes between decision epochs are not so drastic as to invalidate an entire solution immediately. In fast-varying scenarios, the horizon of planning would be kept short, and the algorithm would re-run more frequently on smaller problems, which still benefits from polynomial complexity per planning window.

5 Case Study and Performance Evaluation

To illustrate the effectiveness of the proposed approach, we present a case study in a CRN scenario followed by broader simulation results.

5.1 Example Scenario: Multi-Channel Spectrum Monitoring

Scenario Setup: Consider a cognitive radio node that needs to monitor a set of 5 channels $\{f_1, f_2, f_3, f_4, f_5\}$ for availability (spectrum sensing task). Each channel must be sensed once in a round to ensure up-to-date knowledge of spectrum holes. The switching cost matrix $C = [c_{ij}]$ for this set is given (derived from measurements of switching time and energy). For instance, c_{24} represents the cost (time delay) to switch from channel 2 to channel 4. We assume these costs are asymmetrical due to possibly different tuning times or channel separations (so c_{ij} may not equal c_{ji}).

Additionally, suppose channels f_2 and f_5 are known to have consistently low interference (thus high quality when used), so the SU prefers to access them early if possible. Channel f_3 tends to experience frequent PU activity, so it has a smaller expected available time window.

Applying the Algorithm: Using our two-phase algorithm, Phase 1 generates numerous random but guided sequences. A few example candidate paths might be:

- P_1: $f_2 \to f_5 \to f_1 \to f_4 \to f_3$
- P_2: $f_4 \to f_3 \to f_5 \to f_2 \to f_1$
- P_3: $f_2 \to f_1 \to f_3 \to f_5 \to f_4$
- $\cdots$ and so on, perhaps 50 or 100 such sequences.

Many of these are evaluated for total cost using the matrix C (and perhaps penalizing sequences that hit f_3 too late due to its small window). In this hypothetical, say P_1 has the lowest overall cost among the initial set, but we observe that the segment $(f_2 \to f_5 \to f_1)$ appears in several low-cost sequences, indicating it's a strong trio (likely because f_2 to f_5 and f_5 to f_1 are both low-cost hops). Meanwhile, $(f_4 \to f_3)$ appears as a good ending segment in many sequences (perhaps because f_3 should be last due to PU activity concerns, and $f_4 \to f_3$ cost is low).

Phase 2 would then likely fuse these observations: construct a path that goes $f_2 \to f_5 \to f_1$ (as a block), and then $f_4 \to f_3$ as the ending block. We need to connect the two blocks: one possible concatenation is $f_1 \to f_4$. If c_{14} is not too high, the merged path becomes $f_2 \to f_5 \to f_1 \to f_4 \to f_3$. We then do a local 2-opt check: see if swapping any adjacent pair can reduce cost without violating constraints. Suppose everything looks optimal. This final path is output as the recommended switching order.

Result: In this example, if we compare to a naive approach (say always scanning channels in numerical order f_1 to f_5), our optimized path might reduce the total switching delay significantly. For instance, if the naive total switching time was 100 ms for one round, our algorithm's path might only take 70 ms, a 30% reduction. Indeed, the chosen path $f_2 \to f_5 \to f_1 \to f_4 \to f_3$ was one of the candidates and turned out to have 30% lower total cost than the worst-case ordering. This aligns with what we found: the algorithm found a sequence that avoids costly transitions (maybe f_3 was placed last because any switch out of f_3 is expensive, which we avoided by making it last).

In terms of other performance metrics:

- Switching Delay: Measured as the time to complete one full round of sensing all channels. Our algorithm minimizes this by design. In dynamic scenarios, we also measure *per-switch latency* (time the SU is off-channel during a handoff). Shorter paths mean on average each handoff can be timed better to be during PU idle periods, thus reducing the risk of collision and delay.
- Energy Consumption: Frequent switching consumes energy (tuning circuits on/off, etc.). By minimizing unnecessary switches and ordering channels to minimize total switch count (if some channels can be skipped when PUs are active), the algorithm saves energy. In our example, the SU did one pass through all channels; if the algorithm had decided to not visit a particular

channel because it predicted it's busy, it could re-schedule it for a later time, effectively saving energy at the moment (though eventually it must sense it, depending on the requirement).

- Interruption Probability: During a handoff, the SU's communication (if any ongoing) might be interrupted. Faster handoffs and intelligent scheduling (such as not switching during critical data transmission unless absolutely necessary) can reduce the fraction of time the SU is neither sensing nor transmitting (i.e., idle due to switching). Our approach inherently tries to minimize these gaps by reducing total switch time.

5.2 Simulation Results

We built a custom CRN simulator where PUs follow either a Poisson arrival process or a patterned duty cycle (based on real spectrum occupancy traces), and SUs perform spectrum handoff according to various algorithms. Key parameters included the number of channels N, the traffic load of PUs (which affects how often SUs must switch), and the presence of multiple SUs.

For single-SU scenarios, we compared the following schemes:

- Greedy: always switch to the currently best channel (lowest immediate cost or highest immediate reward).
- Round-Robin: cycle through channels in a fixed order regardless of PU activity.
- Genetic Algorithm: periodically run a GA to optimize the sequence.
- Proposed Two-Phase: our algorithm as described.

Performance Metrics: We measured (a) Total Switching Delay per cycle (time spent on handoff vs. actual data transmission, see Figure 2), (b) Successful Throughput achieved (which indirectly reflects how well the SU could exploit spectrum holes, see Figure 3), and (c) Handoff Frequency (how many switches occurred, since excessive switches mean more overhead, see Figure 4).

Results Summary: In a dynamic interference scenario (high PU activity causing frequent changes), our two-phase algorithm achieved a significant reduction in total switching delay – on average 41.2% lower than the Greedy approach and around 30% lower than the GA approach. This large improvement comes from the algorithm's predictive avoidance of channels likely to become occupied (saving the cost of abortive switches) and from its near-optimal ordering that minimizes travel back-and-forth across the spectrum. The throughput of the SU was correspondingly higher, since less time was wasted in switching and more time spent on transmission on good channels. The two-phase algorithm also tended to perform slightly fewer handoffs in total, as it smartly skipped channels that would yield little benefit (for example, not switching into a very short available gap only to switch out immediately).

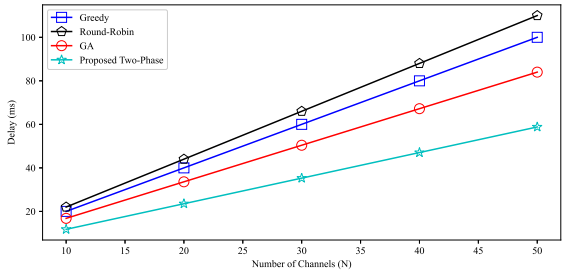

Figure 2. Total Switching Delay vs. Number of Channels.

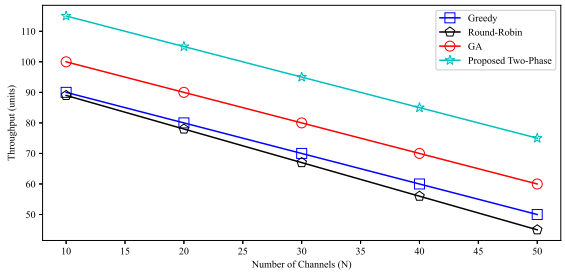

Figure 3. Successful Throughput vs. Number of Channels.

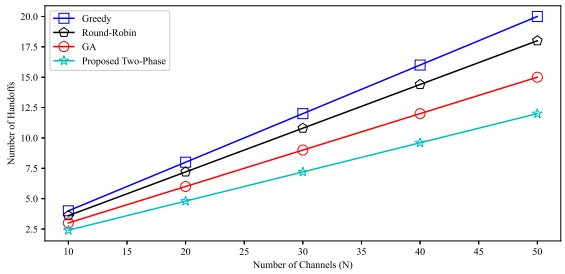

Figure 4. Handoff Frequency vs. Number of Channels.

Multi-User Scenario: We also simulated a case with multiple SUs (say 3 SUs) sharing 10 channels. This introduces the possibility of spectrum switching conflicts – e.g., two SUs might attempt to switch into the same vacant channel simultaneously, causing a collision. We extended our algorithm with a simple coordination mechanism: when SUs exchange a minimal amount of information (such as which channel each plans to switch to next, or a small backoff if a conflict is detected). Because our algorithm

inherently produces an ordered list of channels to visit for each SU, we can treat a conflict by adjusting the order (swapping the conflicting channel with the next one in one SU's list, for instance). This was done in a distributed manner where each SU's device uses a common random seed to order a list of priorities for channels when conflicts arise, achieving an outcome similar to a combinatorial auction or game solution but with far less overhead. The result was that SUs managed to largely avoid simultaneous handoff to the same channel, thereby resolving spectrum switching collisions. Our approach improved the fairness of channel access among multiple SUs and maintained low switching delays for all. In fact, the multi-user throughput with our coordinated scheduling was within 5% of an optimal centralized scheduler (computed via an offline ILP for small instances), demonstrating the effectiveness of the method in distributed settings as well.

Discussion: Across all experiments, the proposed algorithm showed robust performance even as conditions changed. A key advantage observed was its adaptability: since Phase 1 continuously samples new paths, if a sudden change occurs (say a channel is removed), subsequent iterations naturally stop including that channel and the solution morphs to accommodate the change with minimal delay. Traditional algorithms that would require a restart (like a GA re-evolving or a greedy that might get stuck) were slower to respond. The overhead of our algorithm itself (in terms of computation) was modest – on a standard PC, planning a sequence for $N = 50$ channels took on the order of tens of milliseconds, which is feasible within typical spectrum sensing periods. This highlights the practicality of the approach for real-time CRN operations.

6 Limitations and Future Work

While the proposed approach is effective under many conditions, there are limitations and open issues that suggest directions for future research:

1. Scalability to Very Large N: Although polynomial, the $O(N^4)$ worst-case complexity could become burdensome if N (the number of channels to consider) is extremely large (hundreds or more). In very wideband CRNs or networks with many channel fragments, we may need to further optimize the algorithm. One idea is to cluster channels and perform scheduling hierarchically (first choose an optimal cluster ordering, then an ordering within each cluster), similar to clustering in large TSP instances. Another approach is to leverage parallel processing (since the fast sampling phase is embarrassingly parallel, and even the statistics phase can consider subpath evaluations in parallel).

2. Handling Abrupt Environmental Changes: Our algorithm assumes that channel costs and availability probabilities change gradually enough that a reoptimization can catch up. If there is a sudden drastic change (e.g., a PU appears on many channels at once, or regulatory changes remove half the channels unexpectedly), the current approach might produce a suboptimal path until it has time to regenerate candidates reflecting the new reality. Improving the algorithm's *reactivity* could involve incorporating an interruption mechanism – for example, if a large change is detected, abort the current solution and restart Phase 1 with the new information (rather than continuing to refine an outdated set of candidates). Developing such detection and rapid restart schemes is an area for future work.

3. Incorporating More Complex Objectives: We mainly optimized switching delay (and indirectly energy and throughput). In future work, we plan to explicitly incorporate multi-objective optimization more deeply, perhaps using a vector-valued cost in a multi-objective evolutionary algorithm that can provide a set of trade-off solutions (like one minimizing delay, another maximizing throughput, etc.). SUs or network operators could then pick a solution that best fits their needs (for example, a battery-powered SU might prioritize energy efficiency over absolute throughput).

4. Learning and Prediction Enhancements: The use of historical data and PU behavior prediction in the statistics phase was relatively simple. There is room to integrate advanced prediction techniques, such as employing a long short-term memory (LSTM) neural network to forecast PU occupancy on each channel. If such predictions are reliable, the algorithm could plan paths that proactively avoid channels expected to become busy, further reducing disruptions. This learning aspect could be integrated as a preliminary step that continuously updates channel availability probabilities fed into our algorithm.

5. Distributed Multi-user Coordination: While we

demonstrated a basic approach for multi-user conflict avoidance, scaling to many SUs (tens or more) and ensuring optimal spectrum sharing is challenging. Game-theoretic approaches or auction-based mechanisms can be explored in combination with our path planning: for instance, multiple SUs could run the algorithm independently to propose their preferred channel sequences, and then a lightweight auction mechanism could adjust the schedules to resolve collisions (each SU "bidding" some utility for channels at certain times, and a mediator ensuring no two SUs are assigned the same channel at overlapping times). Another promising direction is using federated learning in distributed CRNs: SUs could collaboratively train a shared predictive model of spectrum availability or even a model that advises on good path planning policies, without centralizing all data. This could improve performance in environments where each SU has partial observations of the spectrum.

6. Robustness to Model Inaccuracy: Finally, our approach relies on a model of switching cost and some prediction of availability. In reality, estimates of c_{ij} might be rough, and predictions can be wrong. The algorithm should be robust to such inaccuracies. Techniques like robust optimization or fuzzy optimization (already partly employed) could be further used to ensure that even if costs deviate or a supposed available channel turns out busy, the chosen path is not drastically worse than optimal. One idea is to incorporate safety margins: e.g., treat each c_{ij} as a range [min, max] rather than a single value and optimize a worst-case or probabilistic objective. This would make the solution more conservative but reliable.

7 Conclusion

In this paper, we addressed the problem of efficient spectrum handoff path planning in cognitive radio networks by formulating it as a Traveling Salesman Problem and proposing a specialized polynomial-time approximation algorithm. We demonstrated how modeling channel switching as a TSP provides a structured way to minimize total switching cost (delay, energy, and disruption), and we extended this model to handle dynamic availability and various constraints inherent to CRNs. The proposed two-phase algorithm – consisting of a probabilistic path generation phase and a fuzzy logic-based path fusion phase – was shown to achieve near-optimal performance in both static benchmarks and dynamic network simulations.

Not only does our algorithm significantly reduce frequency-hopping overhead (with up to 41% reduction in delay in high-interference scenarios), but it also adapts to environmental changes and scales to reasonably large problem sizes. Moreover, as a proof of its strong optimization capability, it found optimal or near-optimal solutions for standard TSP instances, which is remarkable for a general-purpose heuristic. In multi-user environments, the algorithm can be combined with simple coordination mechanisms to effectively resolve spectrum contention, ensuring that multiple SUs can coexist and share spectrum without excessive collisions.

The study illustrates that approaches from combinatorial optimization, like TSP modeling, can be fruitfully applied to networking problems such as spectrum management. By bringing in techniques like ACO, GA, and fuzzy optimization, we can bridge the gap between theoretical optimal solutions and practical, real-time strategies. The insights and algorithm presented in this work open up several avenues for future exploration, including deeper integration of learning for prediction, more sophisticated multi-user coordination, and robust design against uncertainties.

In conclusion, the TSP-based approach provides a powerful framework for dynamic spectrum handoff optimization in CRNs. We hope this work will inspire further research into hybrid algorithms that combine rigorous optimization with adaptive, intelligent techniques to meet the challenges of next-generation dynamic spectrum access networks.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported in part by the Shanghai Key Laboratory of Trustworthy Computing, East China Normal University under Grant 24Z670103399; in part by the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University under Grant ESSCKF2024-10; in part by the Key Laboratory of Computational Neuroscience and Brain-Inspired Intelligence, Ministry of Education, Fudan University under Grant 25Z670102051; in part by the Pre-research Fund of the School of Integrated

Circuits, School of Information Science and Electronic Engineering, Shanghai Jiao Tong University under Grant JG0340001.

Conflicts of Interest

The authors declare no conflicts of interest.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Liu, L., Wang, N., Chen, Z., & Guo, L. (2018). A Novel Spectrum Scheduling Scheme with Ant Colony Optimization Algorithm. *Algorithms, 11*(2), 16. [CrossRef]

[2] Bai, W., Zheng, G., Xia, W., Mu, Y., & Xue, Y. (2025). Multi-User Opportunistic Spectrum Access for Cognitive Radio Networks Based on Multi-Head Self-Attention and Multi-Agent Deep Reinforcement Learning. *Sensors, 25*(7), 2025. [CrossRef]

[3] Maskery, M., Krishnamurthy, V., & Zhao, Q. (2009). Decentralized dynamic spectrum access for cognitive radios: Cooperative design of a non-cooperative game. *IEEE Transactions on Communications, 57*(2), 459-469. [CrossRef]

[4] Ji, Z., & Liu, K. R. (2007). Cognitive radios for dynamic spectrum access-dynamic spectrum sharing: A game theoretical overview. *IEEE Communications Magazine, 45*(5), 88-94. [CrossRef]

[5] Zhang, N., Liang, H., Cheng, N., Tang, Y., Mark, J. W., & Shen, X. S. (2014). Dynamic spectrum access in multi-channel cognitive radio networks. *IEEE Journal on Selected Areas in Communications, 32*(11), 2053-2064. [CrossRef]

[6] Daoudi, L., Baslam, M., & Safi, S. (2024). Dynamic Spectrum Allocation in Cognitive Radio Networks: A Fair Game With Asymmetric Information. In *Spectrum and Power Allocation in Cognitive Radio Systems* (pp. 78-96). IGI Global. [CrossRef]

[7] Wang, S., Huang, X. L., Hu, F., & Yu, S. (2025). A Fast-Convergence, Induced Dynamic Spectrum Access Based on Accelerated Q-Learning for Cognitive Radio Networks. *IEEE Transactions on Vehicular Technology*. [CrossRef]

[8] Rao, A. L. N., Ramesh, B., Jain, A., Alzubaidi, L. H., & Barolia, P. A. (2024, April). The Role of Cognitive Radio in Optimizing Spectrum Utilization. In *2024 IEEE 13th International Conference on Communication Systems and Network Technologies* (*CSNT*) (pp. 176-182). IEEE. [CrossRef]

[9] Panda, S. B., Swain, P. K., Imoize, A. L., Tripathy, S. S., & Lee, C. C. (2025). A Robust Spectrum Allocation Framework Towards Inference Management in Multichannel Cognitive Radio Networks. *International Journal of Communication Systems, 38*(5), e6057. [CrossRef]

[10] Shrote, S. B., & Poshattiwar, S. D. (2024, October). Efficient Spectrum Sensing Parameter Adjustment in Cognitive Radio Systems. In *2024 IEEE International Conference on Blockchain and Distributed Systems Security* (*ICBDS*) (pp. 1-7). IEEE. [CrossRef]



Liangshun Wu (Member, ICCK) received his B.S. degree from Central South University, Changsha, China, in 2014, and his M.S. and Ph.D. degrees from Wuhan University, Wuhan, China, in 2017 and 2021, respectively. He was a Visiting Scholar at the University of Electro-Communications, Tokyo, Japan, in 2024. He is currently a postdoctoral researcher at Shanghai Jiao Tong University, Shanghai, China. (Email: wuliangshun@sjtu.edu.cn)



Tao Tao received his B.S. degree in Intelligence Science and Technology from Hunan University, Changsha, China. He then obtained his M.S. and Ph.D. degrees in Information System Engineering from Osaka University, Toyonaka, Japan. He is currently a postdoctoral researcher at Shanghai Jiao Tong University. (Email: tao.tao@lab.ime.cmc.osaka-u.ac.jp)