# The Robopol Refined Algorithm for the Traveling Salesman Problem

Author: Ing. Róbert Polák

Independent Researcher, Slovakia

robopol@gmail.com

Date: 28.05.2025

## Abstract

*The Traveling Salesman Problem (TSP) remains a cornerstone of combinatorial optimization, challenging researchers to find the shortest possible route that visits a given set of cities and returns to the origin city. This paper introduces the Robopol Refined algorithm, an advanced metaheuristic approach designed to efficiently find high-quality solutions for the TSP. The algorithm integrates an Iterated Local Search (ILS) framework with Beam Search, employing a sophisticated combination of perturbation mechanisms and local search operators. Key features include adaptive perturbation strategies based on solution stagnation, candidate list techniques inspired by Lin-Kernighan-Helsgaun (LKH), and a "Checkout Kick" mechanism to escape deep local optima. We discuss the architecture of the Robopol Refined algorithm, detailing its Numba-optimized components for enhanced performance on Euclidean TSP instances, and its parallel execution capabilities for multiple independent runs.*

**Keywords**

*Traveling Salesman Problem, TSP, Iterated Local Search, Beam Search, Metaheuristics, Combinatorial Optimization, Robopol Algorithm, Local Search, Perturbation, LKH.*

### 1. Introduction

The Traveling Salesperson Problem (TSP) is a classic NP-hard problem in combinatorial optimization. Given a set of cities and the distances between each pair of cities, the objective is to find the shortest possible tour that visits each city exactly once and returns to the starting city. Despite its simple formulation, finding an optimal solution becomes computationally infeasible for large instances, necessitating the use of heuristic and metaheuristic algorithms. Many successful approaches, such as the Lin-Kernighan-Helsgaun (LKH) algorithm \cite{Helsgaun2000}, have set high benchmarks in solving TSP instances.The Robopol Refined algorithm, presented in this paper, is a sophisticated metaheuristic developed to tackle the TSP. It builds upon the principles of Iterated Local Search (ILS), enhancing it with a Beam Search strategy to maintain a diverse set of high-quality solutions throughout the search process. The algorithm is implemented with a focus on performance, leveraging Numba for JIT compilation of computationally intensive parts (as seen in functions like _single_robopol_refined_run_njit and its parallel counterpart run_robopol_refined_parallel_numba), and supports parallel execution of multiple search trajectories.

### 2. The Robopol Refined Algorithm Architecture

The Robopol Refined algorithm operates through a series of iterated local search phases, refined by a beam search mechanism. Each run of the algorithm, potentially executed in parallel via

run_robopol_refined_parallel_numba, involves the following core components detailed within _single_robopol_refined_run_njit.

### 2.1. Initialization

An initial solution is typically generated using a constructive heuristic, such as the Optimized Nearest Neighbor algorithm (get_optimal_nearest_neighbor). This initial tour is then improved by applying local search operators like an efficient 2-opt (e.g., two_opt_njit_with_temp using temp_matrix which is a k-nearest-neighbor list) and an adaptive segment moving optimization (move_segment_optimization_adaptive). The resulting solution forms the starting point for the main ILS loop and initializes the beam. A candidate_set (generated by _calculate_candidate_set_njit) is also computed, storing promising edges for each node, a technique also central to powerful heuristics like LKH \cite{Helsgaun2000}, which aids in guiding certain heuristic operations.

### 2.2. Iterated Local Search (ILS)

CoreThe main loop of the algorithm iteratively perturbs existing solutions from the beam and applies an intensive local search (LLS) procedure to generate new candidate solutions.

#### 2.2.1. Perturbation

The perturb_stable_flexible_njit function implements an adaptive perturbation strategy. The choice and intensity of the perturbation (e.g., Double Bridge via perturb_double_bridge_njit, Segment Move via perturb_segment_move_njit, or Sequential 3-opt via perturb_sequential_3_opt_njit) are dynamically adjusted. This adaptiveness is influenced by iterations_since_last_global_improvement relative to stagnation_threshold_low and stagnation_threshold_high. A targeting_probability parameter controls the likelihood of using more focused perturbation strategies, which can leverage the pre-calculated candidate_set and distance_matrix. For instance, perturb_sequential_3_opt_njit has a use_targeted_version flag, and perturb_candidate_double_bridge_njit explicitly tries to utilize candidate edges. These targeted approaches draw inspiration from sophisticated move selection processes in algorithms like LKH \cite{Helsgaun2009}.

#### 2.2.2. Intensive Local Search (LLS)

The generate_locally_optimized_candidates_njit function takes a perturbed solution and applies a multi-stage LLS procedure. For each candidate generation:

1. An initial, primary perturbation is applied using perturb_stable_flexible_njit.
2. This is followed by a "large" random move, which can be a Sequential 3-opt, Double Bridge, or Segment Move.
3. Subsequently, a series of 2-opt optimizations (two_opt_njit_with_temp) is performed. This 2-opt phase is itself iterative, controlled by max_local_search_attempts and local_search_no_improvement_threshold, ensuring a thorough exploration of the local neighborhood.
4. Optionally, if the LLS yields significant improvement (e.g., exceeding a 2% threshold compared to the state after the initial perturbation), a further move_segment_optimization_adaptive step can be applied to the LLS solution.

### 2.3. Beam Search

Instead of focusing on a single incumbent solution, Robopol Refined utilizes a beam search. From each solution in the current beam (beam_paths), multiple new candidate solutions are generated using the perturbation and LLS steps described above. The _select_best_candidates_njit function then selects the beam_width best solutions from all generated candidates (based on all_candidate_distances_after_lls_list) to form the beam for the next iteration. This helps maintain solution diversity and explore different regions of the search space.

### 2.4. Acceptance Criterion and Stagnation Handling

A new global best solution (global_best_path, global_best_distance) is accepted if it significantly improves upon the current global best, considering an acceptance_tolerance_base. The number of iterations without global improvement (iterations_without_improvement) is tracked.

- Adaptive Perturbation: As mentioned, perturbation strength adapts to iterations_without_improvement.
- Checkout Kick Mechanism: If checkout_kick_active is true and the search stagnates for a prolonged period (defined by ils_no_improvement_limit * checkout_kick_threshold_factor), a strong "kick" is applied to the current global_best_path. This involves applying perturb_double_bridge_njit multiple times (controlled by checkout_kick_strength), optionally followed by a 2-opt optimization (checkout_kick_2opt_iterations). The resulting solution can then re-seed the candidate pool for the next iteration, aiming to escape a deep local optimum.
- Termination: The ILS loop for a single run terminates if a maximum number of iterations (ils_iterations) is reached, or if iterations_without_improvement exceeds ils_no_improvement_limit, or if the beam itself stagnates (i.e., beam_stagnation_counter comparing beam_paths and previous_beam_paths reaches BEAM_STAGNATION_LIMIT).

### 2.5. Parallel Execution

The run_robopol_refined_parallel_numba function allows for num_refined_runs independent instances of the _single_robopol_refined_run_njit procedure to be executed in parallel using Numba's prange. The best solution found across all parallel runs is then returned as the final result.

## 3. Key Algorithmic Components (Numba-Optimized)

Several Numba-optimized functions (@njit) form the building blocks of the Robopol Refined algorithm, ensuring efficient execution:

- Distance Calculation: calculate_distance_matrix_1 (parallelized for creating the full distance matrix) and compute_path_distance (for calculating total tour length).
- Candidate Set Generation: _calculate_candidate_set_njit creates a list of promising neighboring nodes for each node.
- Local Search Operators:
    - two_opt_njit_with_temp: An efficient 2-opt implementation that utilizes the temp_matrix (k-nearest neighbors list) to guide its search for improving moves.
    - move_segment_optimization_adaptive: An adaptive procedure for relocating segments of the tour, which also uses an active-node list (active_node_ids_to_check) to focus search efforts. It considers moving segments of varying max_segment_size to positions near max_nearest_neighbors. The move_segment_in_path and move_segment_in_path_1 helper functions handle the actual segment relocation.
- Perturbation Operators:
    - perturb_double_bridge_njit: Performs the classic 4-opt move known as the double bridge.
    - perturb_segment_move_njit: Relocates a random segment (up to max_segment_pct of tour length) of the tour to a random new position.
    - perturb_sequential_3_opt_njit: Applies a series of 3-opt moves (_apply_3_opt_move_njit). It has a use_targeted_version flag to enable a more strategic selection of 3-opt moves based on potential gain and candidate edges, trying max_attempts_target to find such a move before resorting to random 3-opts if necessary.
    - perturb_candidate_double_bridge_njit: A targeted Double Bridge variant that attempts to create at least one candidate edge from the candidate_set while maximizing gain, with max_attempts_candidate_db for finding suitable cuts.

- Helper Functions:
    - get_optimal_nearest_neighbor and get_optimal_nearest_neighbor_fixed: Used for initial solution construction.
    - weighted_random_choice: A Numba-compatible function for making weighted random selections, used in perturb_stable_flexible_njit.

**4. Conclusion**

The Robopol Refined algorithm presents a robust and efficient metaheuristic approach for solving the Traveling Salesperson Problem. By strategically combining an Iterated Local Search framework with Beam Search, adaptive perturbation strategies (dynamically chosen from Double Bridge, Segment Move, and Sequential 3-opt), candidate-list techniques inspired by effective heuristics like LKH \cite{Lin1973, Helsgaun2000}, and specialized mechanisms like the Checkout Kick, it aims to achieve a strong balance between intensification of search in promising regions and diversification to explore the broader solution space. The extensive use of Numba for JIT compilation of core components and the support for parallel execution of multiple runs further enhance its practical applicability to computationally challenging TSP instances. Future work could involve more extensive benchmarking against established TSP solvers like LKH \cite{Helsgaun2000, Helsgaun2009} and further tuning of its adaptive parameters and heuristic selection logic.5.

## REFERENCES

[1] Lin, S., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research, 21(2).

[2] Helsgaun, K. (2000). An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal of Operational Research, 126(1).

[3] Helsgaun, K. (2009). General k-opt submoves for the Lin-Kernighan TSP heuristic. Mathematical Programming Computation, 1(2-3).

[4] Johnson, D. S., & McGeoch, L. A. (1997). The Traveling Salesman Problem: A Case Study in Local Optimization. In E. H. L. Aarts & J. K. Lenstra (Eds.), Local Search in Combinatorial Optimization. John Wiley and Sons.(You may consider adding more references related to Iterated Local Search, Beam Search, or other specific techniques mentioned if applicable.)