Applied Computer Science, vol. 20, no. 4, pp. 117–137
doi: https://doi.org/10.35784/acs-2024-43

Submitted: 2024-09-07 | Revised: 2024-10-19 | Accepted: 2024-10-29

Keywords: artificial immune systems, fine-tuning, hybridization, traveling salesman problem

Peeraya THAPATSUWAN [0009-0007-4187-124X]*,
Warattapop THAPATSUWAN [0000-0001-7740-727X]*,
Chaichana KULWORATIT [0000-0001-9959-4264]**

ENHANCEMENT OF ARTIFICIAL IMMUNE SYSTEMS FOR THE TRAVELING SALESMAN PROBLEM THROUGH HYBRIDIZATION WITH NEIGHBORHOOD IMPROVEMENT AND PARAMETER FINE-TUNING

Abstract

This research investigates the enhancement of Artificial Immune Systems (AIS) for solving the Traveling Salesman Problem (TSP) through hybridization with Neighborhood Improvement (NI) and parameter fine-tuning. Two main experiments were conducted: Experiment A identified the optimal integration points for NI within AIS, revealing that position 2 (AIS+NIpos2) improved solution quality by an average of 27.78% compared to other positions. Experiment B benchmarked AIS performance with various enhancement techniques. Using symmetric and asymmetric TSP datasets, the results showed that integrating NI at strategic points and fine-tuning parameters boosted AIS performance by up to 46.27% in some cases. The hybrid and fine-tuned version of AIS (AIS-th) consistently provided the best solution quality, with up to a 50.36% improvement, though it required more computational time. These findings emphasize the importance of strategic combinations and fine-tuning for creating effective optimization algorithms.

1. INTRODUCTION

Real-world problems such as scheduling, packing, loading, and routing often require sophisticated approaches for larger instances. Over the past few decades, nature-inspired algorithms have emerged as robust solutions for complex optimization problems. These algorithms are categorized into three main groups (Engin & Döyen, 2004): socially-inspired algorithms like Tabu Search (Glover, 1989); physically-inspired algorithms such as Simulated Annealing (Kirkpatrick et al., 1983); and biologically-inspired algorithms including Genetic Algorithms (Goldberg, 1989), Particle Swarm Optimization (Eberhart et

* Kasetsart University Kamphaeng Saen Campus, Faculty of Liberal Arts and Science, Department of Computational Science and Digital Technology, peeraya.t@ku.ac.th, warattapop.t@ku.ac.th
** King Mongkut’s Institute of Technology Ladkrabang, School of Science, Department of Computer Science, chaichana.kul@kmitl.ac.th

al., 2001), Ant Colony Optimization (Dorigo & Stützle, 2004), and Artificial Immune Systems (AIS) (Hart & Timmis, 2008).

Among these, AIS has attracted significant attention due to its unique advantages. AIS employs a multi-directional search process using a diverse set of antibodies (candidate solutions), unlike conventional single-directional search methods (De Castro & Timmis, 2002). The cloning process in AIS selectively replicates high-affinity antibodies and employs a dual mutation mechanism, allowing antibodies that fail the first mutation a second chance. Despite these strengths, conventional AIS has faced criticism for its performance limitations in specific applications (Greensmith et al., 2010).

Enhancing AIS to find the global optimum consistently is a persistent challenge, particularly for large-scale combinatorial optimization problems. The stochastic nature of AIS's search process often leads to variable solution quality. Two primary techniques have been identified to enhance AIS performance: parameter fine-tuning and hybridization with other optimization methods (Sengupta et al., 2019).

Fine-tuning involves optimizing the algorithm's parameters through systematic experimentation, which is essential for metaheuristics relying on stochastic processes (Eiben & Smit, 2011; Huang et al., 2019). Research has shown that optimal parameter settings derived from well-designed experiments are critical to improving solution quality (Eiben & Smit, 2011; Adenso-Díaz & Laguna, 2006). Hybridization involves integrating AIS with other optimization techniques to enhance its performance. Methods such as local search heuristics, fuzzy logic, and genetic algorithms have successfully combined with optimization methods like AIS to solve complex problems (Akram & Habib, 2023; Ruan, 1997). The synergy of parameter tuning and hybridization often yields superior results compared to employing either technique alone (Yang, 2023; Joy et al., 2023).

This paper presents a novel heuristic technique combining a local search heuristic called Neighborhood Improvement (NI) with the AIS framework while incorporating parameter modification. Our objective is to investigate different approaches for incorporating NI into AIS and assess its influence on performance.

This study has two main objectives:

1. To investigate the optimal positions for integrating NI within AIS to enhance its performance.
2. To benchmark the performance of AIS enhanced through fine-tuning, hybridization with NI, and combining both techniques.

This research employs datasets from the Traveling Salesman Problem (TSP) library, precisely symmetric and asymmetric TSP instances (STSP & ATSP), to evaluate the proposed methods.

The following sections of this paper are structured as follows: Section 2 presents the mathematical model of the Traveling Salesman Problem (TSP) and the problem statement. Section 3 describes the Artificial Immune System (AIS) algorithm and its modifications for TSP. Section 4 focuses on the local search techniques, including the integration of Neighborhood Improvement (NI) within AIS. Section 5 details the experimental design and analysis aimed at benchmarking the performance of AIS with various enhancement techniques. Finally, Section 6 presents the conclusions and future research directions.

2. PROBLEM STATEMENT AND MATHEMATICAL MODEL

2.1. Mathematical formulation

The Traveling Salesman Problem (TSP) is a classic optimization problem where the objective is to find the shortest possible route that visits each city exactly once and returns to the starting city. The main challenge is the enormous number of possible tours, which grows factorially with the number of cities, making it a classic NP-hard problem (Larrañaga et al., 1999).

The TSP can be mathematically formulated as follows:

$$\text{Minimize} \qquad T = \sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij}x_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j = 1, 2, \ldots, m;\ i \neq j \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i = 1, 2, \ldots, m;\ i \neq j \tag{3}$$

$$u_i - u_j + m.x_{ij} \leq m - 1 \quad i, j = 2, 3, \ldots, m;\ i \neq j \tag{4}$$

$$u_i \geq 0 \ \ \forall i \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \tag{6}$$

Here, m represents the number of cities, d_{ij} denotes the distance between city i and city j, and x_{ij} is a binary variable indicating whether the route includes a direct path from city i to city j. The objective is to minimize the total travel distance. Constraints ensure that each city is visited exactly once and that no sub-tours exist. Variables u_i and u_j are used to prevent sub-tours.

2.2. Symmetric TSP (STSP)

The Symmetric Traveling Salesman Problem (STSP) is a classical combinatorial optimization problem where the goal is to find the shortest possible route that visits each city exactly once and returns to the starting point, with the distance between cities being the same in both directions. This variation has significant applications in real-world problems such as urban logistics, scheduling, and network design. Solutions to the STSP involve finding the minimum-cost route that visits each node exactly once and returns to the starting node, accounting for symmetrical travel costs (Lawler, 1985).

The STSP has been extensively studied for decades, with early approaches focusing on exact algorithms. These methods include cutting plane algorithms introduced by Laporte and Nobert, branch and bound techniques, and Lagrangian dual approaches (1980). Exact methods aim to guarantee finding the optimal solution, but they often face limitations in terms of computational feasibility, especially for large instances of the problem (Laporte, 1992). As the complexity of the STSP increased, researchers began utilizing heuristic and metaheuristic approaches to solve the problem. These approaches provide near-optimal solutions within a reasonable computation time, making them more practical for large-scale problems.

Genetic Algorithms (GA) have been widely used for solving STSP because of their ability to explore the solution space efficiently. Early work by Braun demonstrated the effectiveness of GA in finding high-quality solutions by emulating natural evolutionary processes (Braun, 1991). Similarly, Particle Swarm Optimization (PSO), inspired by the social behavior of birds flocking or fish schooling, has also been applied to STSP. Wang et al. introduced a PSO algorithm that incorporates a swap operator and mutation methods, yielding promising results (Kang-Ping et al., 2003), and subsequent improvements by Akhand et al. further enhanced PSO's performance (2014).

Ant Colony Optimization (ACO), inspired by the foraging behavior of ants, has been effectively applied to the STSP. Dorigo and Gambardella introduced ACO for the Traveling Salesman Problem (TSP), demonstrating its capability in finding optimal routes by mimicking the pheromone trail-laying behavior of ants (1997). This method has been continuously improved and combined with other techniques to enhance solution quality. Similarly, the Artificial Bee Colony (ABC) algorithm, which is based on the foraging behavior of bees, has been adapted to STSP. Karaboga and Gorkemli developed the Combinatorial ABC (CABC) algorithm to address combinatorial optimization problems like STSP (2011).

The Firefly Algorithm (FA), inspired by the flashing behavior of fireflies, has been applied to STSP. The work by Li et al. highlighted FA's effectiveness in finding optimal solutions through attraction mechanisms based on firefly brightness (Li et al., 2015). Additionally, Panwar and Deep introduced the Discrete Grey Wolf Optimizer (D-GWO), which incorporates the 2-opt local search method to enhance solution quality. Their study demonstrated that D-GWO outperformed other metaheuristic algorithms in solving TSP instances (2021).

Several hybrid approaches have been developed to combine the strengths of different metaheuristics. Deng et al. proposed a hybrid approach that combines Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) to achieve better results for STSP (2012). Chen and Chien developed a hybrid method combining genetic simulated annealing, ant colony system, and particle swarm optimization (GSA-ACS-PSOT) to solve TSPs (2011). Additionally, Khan and Maiti used a combination of 2-opt and 3-opt with the Artificial Bee Colony (ABC) algorithm, which led to significant improvements in solution quality (2019).

The STSP has been the subject of extensive research, leading to the development of various exact, heuristic, and metaheuristic approaches. While exact methods guarantee optimal solutions, they are often impractical for large instances due to computational constraints. Metaheuristics, on the other hand, provide a balance between solution quality and computational efficiency, making them more suitable for real-world applications (Blum & Roli, 2003). The continuous innovation in hybrid and advanced metaheuristic algorithms ensures ongoing improvements in solving the STSP, addressing the ever-growing complexity of this classic optimization problem (Krishna et al., 2021).

In conclusion, the Symmetric Traveling Salesman Problem has seen significant advancements through the development of various algorithms. Researchers have continually sought to improve the efficiency and effectiveness of solving this problem, from the foundational exact methods to the more recent hybrid metaheuristics. Combining different metaheuristic strategies based on natural processes has led to strong solutions that can be used in the real world. This shows that STSP research is still relevant and important in combinatorial optimization.

2.3. Asymmetric TSP (ATSP)

The Asymmetric Traveling Salesman Problem (ATSP) is a well-studied variation of the TSP where the cost or distance between two nodes is direction-dependent. This variation is crucial for real-world problems like urban logistics, scheduling, and network design, where bidirectional travel costs differ. Solutions to the ATSP involve finding the minimum-cost route that visits each node exactly once and returns to the starting node, considering these asymmetrical travel costs.

ATSP has been extensively researched, leading to numerous methods to solve this complex optimization problem. Over the years, various innovative approaches have significantly advanced the field. The foundational work on ATSP began with the Miller-Tucker-Zemlin (MTZ) model introduced in 1960 (Miller et al., 1960), which provided a compact mathematical formulation with subtour elimination constraints (SECs). However, the MTZ model faced computational efficiency challenges, especially for large-scale problems. In the 1990s, Desrochers and Laporte (1991) improved the MTZ model by introducing lifted SECs, enhancing linear programming relaxation, and improving computational efficiency. Concurrently, Gouveia and Pires (1999) proposed disaggregation techniques for the MTZ model, further boosting the efficiency and effectiveness of solving ATSP by offering stronger SECs.

As exact algorithms struggled with large and complex ATSP instances, researchers shifted to heuristic and metaheuristic methods. In 2012, Nagata and Soler (2012) introduced a genetic algorithm (GA) for ATSP, focusing on effective crossover and mutation operations, which showed superior performance in exploring the solution space and finding near-optimal solutions. In 2019, Boryczka and Szwarc (2019) developed an improved harmony search algorithm, demonstrating its effectiveness in dynamically adjusting parameter values for complex ATSP problems. These advancements highlighted the potential of nature-inspired algorithms in addressing ATSP's directional cost variations.

Osaba et al. (2016; 2018) significantly contributed to solving both symmetric and asymmetric TSPs with their discrete bat algorithm (DBA) and discrete water cycle algorithm (DWCA). These bio-inspired algorithms leveraged natural behaviors to optimize the search process, showing robustness and superior performance in various scenarios. Building on the MTZ model, Campuzano et al. (2020) enhanced its computational performance by generating valid inequalities from fractional solutions, significantly reducing the number of nodes in the branch-and-bound tree, accelerating the convergence of MTZ-type formulations, and extending to solve the multiple ATSP (mATSP).

In 2023, Zhang et al. (2023) introduced the discrete mayfly algorithm (DMA) for the spherical ATSP, integrating inver-over, crossover, and 3-opt operators to enhance the search process in discrete space. This algorithm demonstrated superiority over other metaheuristics, especially in high-dimensional ATSP instances. This progression from foundational mathematical formulations like the MTZ model to advanced metaheuristic algorithms highlights continuous efforts to improve computational efficiency and solution quality.

Burke et al. demonstrated the significant advantages of employing a hybrid approach for solving the ATSP (2001). By combining the HyperOpt heuristic, which embeds an exact algorithm within a local search framework, with the well-established 3-opt heuristic, they were able to leverage the strengths of both methods. This hybrid approach allowed for the exhaustive exploration of promising regions in the solution space, leading to better tour optimization and overcoming the limitations of using a single heuristic. Integrating this hybrid into the Variable Neighborhood Search (VNS) framework further enhanced its effectiveness, enabling the algorithm to escape local optima and achieve high-quality solutions. The introduction of a "guided shake" within VNS, as opposed to random shakes, also contributed to the efficiency of the heuristic. Overall, the hybrid approach proved highly capable of handling real-world constraints and provided a powerful tool for solving complex ATSP problems in industrial settings.

3. ARTIFICIAL IMMUNE SYSTEM

The Artificial Immune System (AIS) is a class of computational algorithms inspired by the principles and processes of the biological immune system. It has been effectively applied to various complex problem-solving scenarios, including combinatorial optimization. The natural immune system's ability to recognize, learn, and remember foreign pathogens provides a robust metaphor for developing algorithms that can adapt and optimize solutions in dynamic and complex environments (Greensmith et al., 2010).

3.1. Natural clonal selection

In the natural immune system, clonal selection refers to the mechanism by which immune cells (B cells) that recognize specific antigens are selected for proliferation and differentiation. When a B cell binds to an antigen, it becomes activated and undergoes clonal expansion, producing many clones. These clones then undergo somatic hypermutation, which introduces random mutations into their receptors, increasing the diversity and improving the immune system's ability to recognize and neutralize the antigen (Burnet, 1959).

1. Antigen Recognition: Each B cell has unique receptors on its surface that can bind to specific antigens. When a B cell encounters an antigen that matches its receptors, it binds to the antigen.
2. Activation and Proliferation: Once the antigen is bound, the B cell becomes activated and undergoes clonal expansion, rapidly dividing to produce many identical copies (clones) of itself.
3. Differentiation: The cloned B cells differentiate into plasma and memory B cells. Plasma cells produce and secrete large quantities of antibodies explicitly targeting the antigen. Memory B cells remain in the body to provide a faster response if the same antigen is reencountered in the future.
4. Affinity Maturation: During the proliferation process, somatic hypermutation occurs, introducing small mutations in the receptors of the cloned B cells. This process

increases the diversity of the receptors, allowing for higher affinity binding to the antigen. B cells with higher affinity receptors are selectively expanded.

5. Elimination of Self-reactive Cells: B cells that bind strongly to self-antigens (molecules ordinarily present in the body) are typically eliminated through negative selection to prevent autoimmune responses.

3.2. AIS clonal selection

Clonal selection is a cornerstone of the Artificial Immune System (AIS) and is inspired by the biological process of clonal selection observed in the natural immune system. In the context of AIS, clonal selection involves the processes of selection, cloning, and mutation of candidate solutions, which are referred to as antibodies (De Castro & Von Zuben, 2001). This process allows the algorithm to iteratively improve the quality of solutions and explore the solution space effectively.

AIS clonal selection relies on two key principles (Garrett, 2005): cloning and affinity maturation. The steps involved in AIS clonal selection are as follows:

1. Initialization: An initial population of antibodies (candidate solutions) is generated, often randomly or using heuristic methods.
2. Evaluation: Each antibody's fitness is evaluated using an affinity function, which measures how well the candidate solution solves the given problem. The affinity function is analogous to the objective function in optimization problems.
3. Selection: A subset of high-affinity antibodies is selected based on their fitness values. These selected antibodies are deemed the most promising solutions and are chosen for cloning. The selection process ensures that only the best-performing solutions are proliferated.
4. Cloning: The selected antibodies undergo clonal expansion, where each antibody produces a number of clones proportional to its affinity. This process amplifies the high-quality solutions, increasing their representation in the population.
5. Mutation: The cloned antibodies are subjected to hypermutation, which introduces random mutations into the clones. The mutation rate is typically inversely proportional to the antibody's affinity, meaning that high-affinity antibodies undergo fewer mutations, preserving their quality while exploring new potential solutions.
6. Affinity Maturation: The mutated clones are re-evaluated for their fitness. This step ensures that only the clones with improved or high affinity are retained in the population. The selection of these high-affinity clones mimics the natural immune system's ability to adapt and improve its response to antigens.
7. Replacement: Low-affinity antibodies in the original population are replaced with the high-affinity clones from the clonal expansion and mutation process. This replacement ensures that the population's overall quality improves over successive iterations.
8. Termination: The process of evaluation, selection, cloning, mutation, and replacement is repeated until a termination criterion is met. This criterion could be a predefined number of generations, a satisfactory fitness level, or population convergence.

Freitas and Timmis (2003) noted that AIS algorithms often neglect inductive and positional bias within representation and affinity measures. To address this, we introduced a modified AIS incorporating the Neighborhood Improvement (NI) heuristic, enhancing standard AIS performance.

4. LOCAL SEARCH

Local search algorithms are critical in combinatorial optimization, particularly for solving the Traveling Salesman Problem (TSP) and its asymmetric variant (ATSP). These algorithms iteratively improve a candidate solution by exploring its neighborhood, which consists of solutions that are "close" in some sense. The goal is to find a locally optimal solution that approximates the global optimum.

4.1. Neighborhood Improvement (NI)

This research introduces a novel local search heuristic called Neighborhood Improvement (NI). This heuristic is designed to enhance the performance of the Artificial Immune System (AIS) framework by systematically exploring and improving candidate solutions. Specifically tailored for the Traveling Salesman Problem (TSP) and its asymmetric variant (ATSP), NI focuses on identifying and relocating the worst-performing pairs of cities within a tour to generate improved solutions. The heuristic iteratively refines the solution until no further improvements can be found.

Algorithm 1. The procedure of Neighborbood Improvement

```
1: Initialize the tour
2: Initialize improvement as True
3: while improvement do
4:    improvement ← False
5:    best_tour ← tour
6:    (city1, city2) ← FindWorstPair(tour)
7:    new_tour ← RelocateCities(tour, city1, city2)
8:    if CalculateDistance(new_tour) < CalculateDistance(best_tour) then
9:        best_tour ← new_tour
10:       improvement ← True
11:   end if
12:   tour ← best_tour
13: end while
14: return tour
```

The procedure of NI (illustrated in Figure 1 and detailed in the pseudocode provided in Algorithm 1) can be described as follows:

1. Selection of Candidate Solution: Start with an initial tour generated by the optimization algorithm.
2. Identification of Worst Pair: Determine the pair of cities in the tour that contribute the most to the total distance.
3. Relocation of Cities: Explore the relocation of these cities within the tour and evaluate the resulting distance.
4. Evaluation and Replacement: If the new tour resulting from the relocation is shorter than the current tour, update the tour with this new configuration.

5. Iteration: Repeat the process until no further improvements can be made, ensuring that the tour is locally optimized.

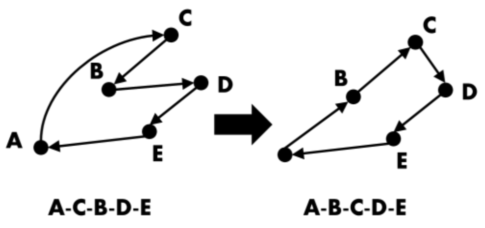

Fig. 1. The Neighborhood Improvement (NI)

4.2. Integration of NI with AIS

The NI heuristic is integrated into the AIS framework at three strategic positions, as shown in Figure 2 and explained in the pseudocode of Algorithm 2. These positions (NI_{pos1}, NI_{pos2}, and NI_{pos3}) are chosen based on their potential to enhance the overall optimization process by improving solution quality at different stages of the algorithm.

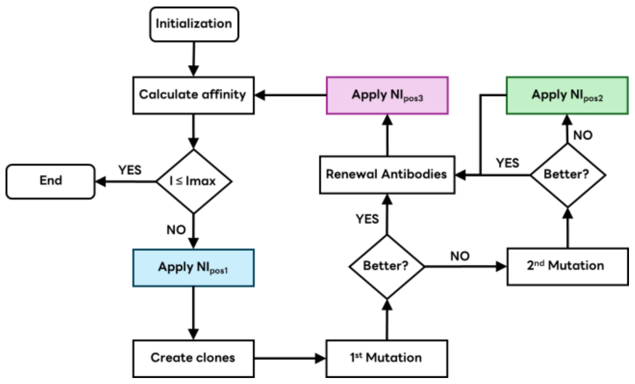

Fig. 2. Flowchart of the NI integration with AIS

1. NI_{pos1} (Before Cloning and Mutation): Applying NI at this position helps refine the best antibodies before cloning and mutation. This approach ensures that high-quality solutions are further improved, providing a stronger foundation for generating clones. By optimizing the best antibody initially, the subsequent clones are more likely to explore promising regions of the solution space.
2. NI_{pos2} (After Mutation): Integrating NI after the mutation provides an opportunity to correct or enhance solutions that were not significantly improved by the initial and second mutations. This intermediate application of NI helps to steer the search process towards more optimal solutions by addressing suboptimal configurations introduced during mutation.
3. NI_{pos3} (After Elimination and Replacement): Applying NI at the final stage, after eliminating the worst antibodies and creating new ones, ensures that the best solutions are refined before the next iteration. This final improvement step helps maintain a

high level of solution quality and prevents the algorithm from stagnating in local optima.

Algorithm 2. The integrating NI into AIS

```
1: P ← InitializeAntibodySize()
2: Imax ← InitializeMaxIterations()
3: %B ← InitializePercentageElimination()
4: population ← GeneratePopulation(P)
5: for each antibody in population do
6:     antibody.affinity ← CalculateAffinity(antibody)
7: end for
8: I ← 1
9: while I ≤ Imax do
10:    for each antibody in population do
11:        if hybrid in position 1 then
12:            best_antibody ← FindBestAntibody(population)
13:            best_antibody ← NeighborhoodImprovement(best_antibody)
14:        end if
15:        NC ← CalculateNumberOfClones(antibody)
16:        clones ← CloneAntibody(antibody, NC)
17:        for each clone in clones do
18:            mutated_clone ← ApplyFirstMutation(clone)
19:            mutated_clone.affinity ← CalculateAffinity(mutated_clone)
20:            if mutated_clone.affinity > clone.affinity then
21:                clone ← mutated_clone
22:            else
23:                mutated_clone ← ApplySecondMutation(clone)
24:                mutated_clone.affinity ← CalculateAffinity(mutated_clone)
25:                if mutated_clone.affinity > clone.affinity then
26:                    clone ← mutated_clone
27:                else
28:                    if hybrid in position 2 then
29:                        clone ← NeighborhoodImprovement(clone)
30:                    end if
31:                end if
32:            end if
33:        end for
34:        antibody ← clone
35:    end for
36:    population ← EliminateWorstAntibodies(population, %B)
37:    new_antibodies ← GenerateNewAntibodies(%B)
38:    population.extend(new_antibodies)
39:    if hybrid in position 3 then
40:        best_antibody ← FindBestAntibody(population)
41:        best_antibody ← NeighborhoodImprovement(best_antibody)
42:    end if
43:    I ← I + 1
44: end while
45: best_solution ← FindBestAntibody(population)
46: return best_solution
```

5. EXPERIMENTAL DESIGN AND ANALYSIS

In this work, the instant datasets of symmetric TSP (STSP) and asymmetric TSP (ATSP) provided in the TSP library (Reinelt, 1991) were used in all experiments. The tours in these datasets are measured using Euclidean distance, which is an abstract, dimensionless unit representing the total length of the paths traveled between cities. The experiments were conducted on a machine with an Intel Core i7-12700H processor, 16 GB of RAM, and an NVIDIA RTX 3050 GPU, running Windows 11. The proposed methods were developed using Python 3.9 with the NumPy and SciPy libraries for numerical computations.

The study consisted of two main experiments: Experiment A, which investigated the appropriate technique for hybridizing AIS with NI, and Experiment B, designed to benchmark the original AIS and improved AIS.

5.1. Experiment A

The objective of Experiment A was to determine the optimal positions for integrating Neighborhood Improvement (NI) within the Artificial Immune System (AIS) to enhance its performance. To achieve this, the total number of searches, determined by the number of candidate solutions multiplied by the number of iterations, was fixed at 5,000 generated solutions.

Tab. 1. Results of experiment A for STSP instances

Problem sizes	Hybrid AIS Types	Quality of solutions (tours) obtained				Time (s)
		SD	Shortest	Longest	Average	
eil51	AIS+NI$_{pos1}$	7.74	438	472	452.47	8.47
	AIS+NI$_{pos2}$	8.54	425	464	427.51	19.13
	AIS+NI$_{pos3}$	8.45	436	470	450.20	8.47
berlin52	AIS+NI$_{pos1}$	205.8	7787	8640	8276.67	8.67
	AIS+NI$_{pos2}$	294.62	7542	8148	7676.40	18.13
	AIS+NI$_{pos3}$	188.98	7891	8712	8256.03	8.70
pr76	AIS+NI$_{pos1}$	4365.12	124071	143657	132119.33	34.08
	AIS+NI$_{pos2}$	4173.78	121069	139274	130034.27	28.47
	AIS+NI$_{pos3}$	4187.74	123977	140303	130886.17	10.97
kroA100	AIS+NI$_{pos1}$	1168.93	29530	34520	32106.20	13.67
	AIS+NI$_{pos2}$	1207.85	28867	33704	30900.77	40.80
	AIS+NI$_{pos3}$	957.09	29932	33274	31351.47	13.53
eil101	AIS+NI$_{pos1}$	24.76	790	886	845.97	13.37
	AIS+NI$_{pos2}$	24.88	784	873	834.17	41.00
	AIS+NI$_{pos3}$	24.37	792	878	838.27	13.30
bier127	AIS+NI$_{pos1}$	5255.51	170032	193897	181716.97	37.53
	AIS+NI$_{pos2}$	5135.55	165274	186768	179777.73	100.53
	AIS+NI$_{pos3}$	5759.61	170559	195591	182738.73	16.40
rat195	AIS+NI$_{pos1}$	189.54	4944	5569	5303.60	81.17
	AIS+NI$_{pos2}$	196.19	4767	5448	5075.37	320.37
	AIS+NI$_{pos3}$	231.54	4897	5598	5264.20	79.13
a280	AIS+NI$_{pos1}$	376.263	8046	9744	8681.00	90.23
	AIS+NI$_{pos2}$	268.47	7668	8972	8163.80	343.67
	AIS+NI$_{pos3}$	319.32	7669	9126	8487.17	37.00

Higher values of generated solutions increase the likelihood of finding reasonable solutions but require longer computational times. The AIS parameters used in this experiment were as follows: the number of antibodies (P) was set to 10, the number of iterations (Imax) was set to 500, and the percentage of antibody elimination (%B) was set to 10%.

Each algorithm's computational runs were repeated 30 times with different random seed numbers for each TSP dataset. The experimental results were analyzed in terms of standard deviation (SD), the shortest, longest, and average distance of tours obtained from each method, and execution time (in seconds), as shown in Tables 1 and 2 for STSP and ATSP, respectively.

Table 1 shows the results for solving STSP. It can be seen that the quality of the traveled tour in terms of average solutions (obtained from AIS+NI$_{pos2}$) was considerably better than the results for AIS+NI$_{pos1}$ and AIS+NI$_{pos3}$ for almost all problem sizes except the smallest STSP instance (eil51) for which AIS+NI$_{pos3}$ found the shortest average tour. Moreover, the experimental results for ATSP instances shown in Table 2 were similar to the findings for STSP. The shortest tour of the smallest instance (br17) was obtained using all methods. For the remaining instances, AIS+NI$_{pos2}$ outperformed AIS+NI$_{pos1}$ and AIS+NI$_{pos3}$ in terms of the average solution. However, higher performance usually requires extended computational time, as Tables 1 and 2 show.

Tab. 2. Results of experiment A for ATSP instances

Problem sizes	**Hybrid AIS Types**	**Quality of solutions (tours) obtained**				**Time (s)**
		SD	**Shortest**	**Longest**	**Average**	
br17	AIS+NI$_{pos1}$	0	39	39	39.00	11.47
	AIS+NI$_{pos2}$	0	39	39	39.00	11.57
	AIS+NI$_{pos3}$	0	39	39	39.00	11.47
ftv35	AIS+NI$_{pos1}$	94.21	1595	2054	1808.63	16.17
	AIS+NI$_{pos2}$	106.14	1513	1998	1804.03	31.80
	AIS+NI$_{pos3}$	96.83	1644	2057	1821.5	16.17
ft53	AIS+NI$_{pos1}$	685.64	8347	11786	10033.3	17.37
	AIS+NI$_{pos2}$	663.01	8268	11446	9828.27	44.20
	AIS+NI$_{pos3}$	553.45	8985	11424	9958.87	17.27
ftv70	AIS+NI$_{pos1}$	176.52	3280	3960	3565.43	25.40
	AIS+NI$_{pos2}$	220.94	2969	3904	3404.33	87.00
	AIS+NI$_{pos3}$	175.65	3126	3872	3522.07	25.37
kro124p	AIS+NI$_{pos1}$	2760.56	50935	58005.8	58005.80	34.70
	AIS+NI$_{pos2}$	1853.27	49777	61364	57116.50	46.47
	AIS+NI$_{pos3}$	3238.56	50912	63796	57737.50	36.00
ftv170	AIS+NI$_{pos1}$	476.76	9196	11001	10294.90	58.60
	AIS+NI$_{pos2}$	439.14	8937	10597	9765.97	202.07
	AIS+NI$_{pos3}$	540.19	9152	11559	10223.40	58.53
rbg323	AIS+NI$_{pos1}$	98.52	2868	3313	3081.43	124.57
	AIS+NI$_{pos2}$	89.56	2790	3111	2946.43	378.03
	AIS+NI$_{pos3}$	84.66	2909	3293	3081.93	115.83
rbg443	AIS+NI$_{pos1}$	130.78	4699	5211	4936.47	175.17
	AIS+NI$_{pos2}$	142.972	4503	5133	4757.30	592.13
	AIS+NI$_{pos3}$	128.98	4611	4872.77	4872.77	173.23

The results indicate that integrating NI at position 2 within the AIS (AIS+NI$_{pos2}$) consistently yielded better average solutions than integrating NI at positions 1 and 3. This improvement can be attributed to several factors:

1. Target of Improvement: The first and third positions for integrating NI target the best antibody directly, aiming to refine the top-performing solution at the beginning and end of the iterations. In contrast, the second position applies NI to clones, meaning the entire antibody population undergoes improvement, potentially leading to a more widespread enhancement of solutions.
2. Diversity and Quality: Applying NI to the clones at position 2 improves the overall quality of the antibody population. This approach ensures that the diversity of solutions is maintained while simultaneously enhancing the quality of multiple solutions rather than focusing solely on the best one. This broader application can prevent premature convergence and promote better solution space exploration.
3. Effective Use of Computational Resources: Computational resources are used more effectively by improving the clones during the iterative process. Early-stage improvements help guide the search process towards better regions of the solution space, while late-stage refinements ensure that the best solutions are adequately polished.
4. Balanced Search Strategy: Integrating NI at position 2 provides a balanced approach between exploration and exploitation. This mid-iteration improvement helps adapt the search strategy dynamically, allowing the algorithm to explore new solutions initially and then focus on refining them, leading to better overall performance.

In conclusion, the superior performance of AIS+NI$_{pos2}$ can be attributed to its strategy of improving the entire population of clones rather than focusing solely on the best antibody. This broader application of NI promotes diversity and quality in the solutions, leading to more effective use of computational resources and a balanced search strategy that enhances the overall performance of the AIS.

5.2. Experiment B

Experiment B aimed to benchmark the performance of AIS with and without improvement techniques.

Tab. 3. AIS Algorithms and parameter settings

AIS algorithm	**Detail**	**The setting parameters of AIS**
AIS	AIS without improving technique and setting parameters according to those used in other research (Chandrasekaran et al., 2006).	P = 10, Imax = 500, %B = 30, inverse and pairwise interchange mutations
AIS-t	AIS with fine-tuned parameters.	P = 10, Imax = 500, %B = 10, inversion and shift operation mutations
AIS-h	AIS with hybridizing with other heuristics (Nipos2).	P = 10, Imax = 500, %B = 30, inverse and pairwise interchange mutations, hybridized with NIpos2
AIS-th	AIS with both fine-tuning and hybridizing techniques. (same as AIS+NIpos2)	P = 10, Imax = 500, %B = 10, inversion and shift operation mutations, hybridizing with NIpos2

The study focused on comparing four types of AIS configurations, each with specific parameter settings for antibody size (P), number of iterations (Imax), and percentage of antibody elimination (%B). Table 3 provides a detailed description of these AIS algorithms and their respective parameter settings.

The different settings for %B and the AIS with tuning techniques (indexed with t) were used to study the impact of mutation operations (inversion and shift operation mutations). As established in Experiment A, the hybridizing technique (indexed with h) involved modifying AIS using NI_{pos2}.

Computational runs using each AIS type were repeated 30 times with different random seed numbers for each symmetric and asymmetric TSP instant dataset. Tables 4 and 5 present the experimental results regarding both the quality of the solutions obtained and computational time usage.

Tab. 4. Performance comparison for STSP

Problem sizes	**AIS algorithms**	**Quality of solutions (tours) obtained**				**Time (s)**
		SD	**Shortest**	**Longest**	**Average**	
eil51	AIS	22.35	551	592.01	592.07	6.06
	AIS-t	8.83	442	480	460.10	9.77
	AIS-h	21.27	475	567	529.70	18.43
	AIS-th	8.54	**425**	464	**427.57**	19.13
berlin52	AIS	429.20	9760	11517	10552.60	4.47
	AIS-t	232.50	7848	8921	8291.90	5.57
	AIS-h	324.39	8761	10067	9349.70	18.50
	AIS-th	294.62	**7553**	8748	**8122.40**	18.13
pr76	AIS	15321.39	214455	292037	241995.10	8.17
	AIS-t	4193.55	127438	143004	136821.20	8.23
	AIS-h	12359.29	171977	215772	194261.57	63.30
	AIS-th	4173.78	**121069**	139274	**130034.27**	28.47
kroA100	AIS	3008.92	46241	58808	51667.97	8.83
	AIS-t	1234.97	32549	38073	34454.67	15.47
	AIS-h	2255.87	34825	43361	39685.03	48.37
	AIS-th	1207.85	**28867**	33704	**30900.77**	40.80
eil101	AIS	50.37	1125	1346	1245.00	8.93
	AIS-t	21.80	848	940	907.30	15.60
	AIS-h	40.59	960	1110	1018.03	54.50
	AIS-th	24.88	**784**	873	**834.17**	41.00
bier127	AIS	7146.29	235023	264670	248642.57	11.60
	AIS-t	6269.90	180596	207933	195079.10	14.60
	AIS-h	9451.14	189079	225388	206012.53	59.50
	AIS-th	5135.55	**165274**	186768	**179777.73**	100.53
rat195	AIS	413.39	9823	11632	10519.90	19.57
	AIS-t	123.84	6052	6505	6286.33	24.27
	AIS-h	507.18	6041	8305	7489.80	104.93
	AIS-th	196.19	**4767**	5448	**5075.37**	320.37
a280	AIS	574.76	15445	17779	16442.23	29.30
	AIS-t	238.55	10740	11742	11222.50	36.73
	AIS-h	497.22	9984	12032	11050.17	524.60
	AIS-th	268.47	**7668**	8972	**8163.80**	343.67

Tables 4 and 5 show that the performance of conventional AIS was outperformed by AIS with improving techniques (AIS-t, AIS-h, and AIS-th) across all datasets. The average distance of the tours obtained using AIS-t was relatively better than those generated using AIS-h for all ATSP and 7 out of 8 STSP instant datasets. These results confirm that optimal parameter settings, especially for metaheuristic algorithms, significantly affect the quality of the solution.

Furthermore, it was found that the modified AIS (combining both tuning and hybridizing techniques as AIS-th) outperformed AIS, AIS-t, and AIS-h in terms of the quality of the solutions obtained for both STSP and ATSP instant datasets. However, the average execution time taken by AIS-th was the longest, followed by AIS-h, AIS-t, and AIS.

Tab. 5. Performance comparison for ATSP

Problem sizes	AIS algorithms	Quality of solutions (tours) obtained				Time (s)
		SD	Shortest	Longest	Average	
br17	AIS	2.57	39	53	39.57	11.10
	AIS-t	0	39	39	**39.00**	11.27
	AIS-h	0	39	39	**39.00**	11.50
	AIS-th	0	39	39	**39.00**	11.57
ftv35	AIS	159.22	1986	2594	2221.83	13.40
	AIS-t	90.58	1625	1984	1823.07	15.20
	AIS-h	107.92	1831	2294	2032.20	30.03
	AIS-th	106.14	**1613**	1998	**1804.03**	31.80
ft53	AIS	1090.36	11538	15337	13344.20	15.07
	AIS-t	610.83	8950	11415	10078.10	15.77
	AIS-h	976.00	10715	14484	12551.80	28.37
	AIS-th	663.01	**8468**	11446	**9828.27**	44.20
ftv70	AIS	353.81	4283	5758	4996.57	16.83
	AIS-t	185.15	3133	4039	3662.37	23.03
	AIS-h	219.44	3667	4507	4181.07	43.70
	AIS-th	220.94	**2969**	3904	**3404.33**	87.00
kro124p	AIS	4590.96	71465	89759	79640.50	20.10
	AIS-t	2396.78	55892	65199	61567.80	31.00
	AIS-h	4058.04	57070	74149	64503.40	64.63
	AIS-th	1853.27	**53777**	61364	**57116.50**	46.47
ftv170	AIS	994.43	13583	17660	15885.20	28.50
	AIS-t	341.89	10715	12163	11547.50	51.70
	AIS-h	479.93	10932	13344	12366.60	135.90
	AIS-th	439.14	**8937**	10597	**9765.97**	202.07
rbg323	AIS	87.18	5058	5736	5276.80	80.07
	AIS-t	67.31	3411	3673	3551.70	107.87
	AIS-h	200.14	3832	4692	4191.97	232.17
	AIS-th	89.56	**2790**	3111	**2946.43**	378.03
rbg443	AIS	77.90	7094	7462	7303.30	149.23
	AIS-t	64.39	5250	5514	5387.33	152.80
	AIS-h	204.74	5835	6604	6124.77	376.20
	AIS-th	142.972	**4503**	5133	**4757.33**	592.13

The results clearly emphasize the importance of fine-tuning parameters and hybridizing with NI. Fine-tuning (as seen in AIS-t) optimizes the algorithm's performance by adjusting parameters to suit the specific problem better, resulting in improved solution quality and consistency. Hybridization (as seen in AIS-h) further enhances the algorithm by incorporating additional heuristics, such as NI, to refine the search process and avoid local optima.

Figures 3 and 4 present the percentage improvements of AIS variants for both STSP and ATSP problems. In Figure 3, the AIS-th variant demonstrates the highest improvement for the majority of STSP problems, with significant gains observed for pr76 (46.27%) and a280 (50.36%). Similarly, Figure 4 shows the AIS-th variant continuing to outperform other variants in ASTSP problems, particularly in rbg443 (34.84%) and ft53 (26.35%). These figures highlight the consistent advantage of hybridization and parameter tuning across different problem sizes.

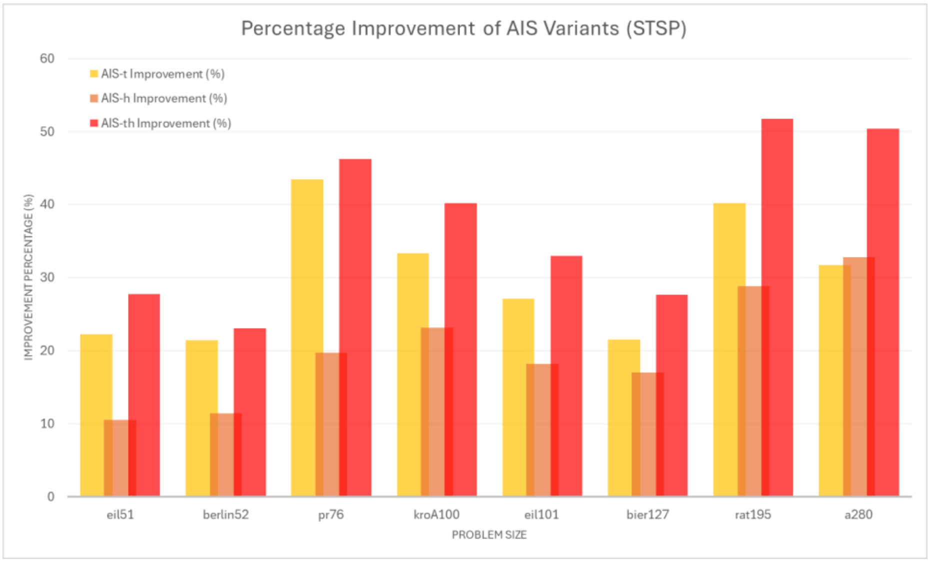

Fig. 3. Percentage improvement of AIS variants over baseline (STSP)

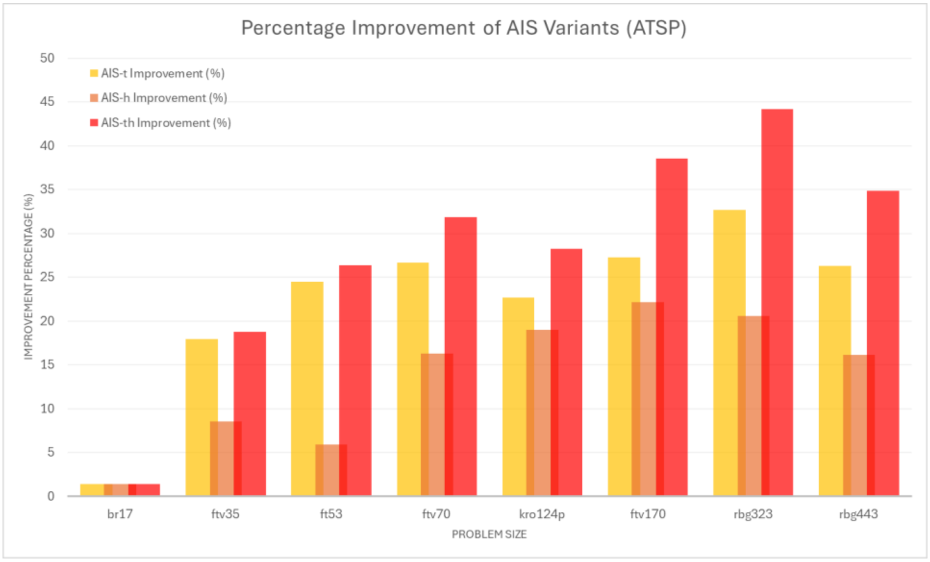

Fig. 4. Percentage improvement of AIS variants over baseline (ATSP)

To ensure statistical significance, a paired t-test was applied to compare the performance of each algorithm variant. The comparison is based on three key metrics: Shortest Tours (the best possible solution found by the algorithm), Average Tours (the average performance across multiple runs), and Best-to-Worst Difference (the difference between the longest and shortest tours found by each algorithm, indicating performance consistency).

A paired t-test was used to statistically compare the performance of each AIS variant with the others across these metrics. The results, summarized in Table 6, report only the p-values for each paired comparison.

Tab. 6. Paired t-test P-values for Shortest Tours, Average Tours and Best-to-Worst Difference

Comparison	**Shortest Tours (p-value)**	**Average Tours (p-value)**	**Best-to-Worse Difference (p-value)**
AIS vs AIS-t	0.028	0.12	0.06
AIS vs AIS-t	0.037	0.08	0.10
AIS vs AIS-t	0.023	0.10	0.04

The Shortest Tours comparison measures the best solution found by each algorithm. The p-values for the comparisons between AIS and its variants (AIS-t, AIS-h, AIS-th) are all below the 0.05 threshold (0.028, 0.037, and 0.023, respectively). This indicates that the difference between AIS and each variant is statistically significant, suggesting that the tuned and hybrid version of AIS consistently outperforms the core AIS in finding the shortest tours.

The Average Tours metric provides a broader view of the algorithms' performance across multiple runs. None of the p-values in this category (0.12, 0.08, and 0.10) fall below the significance threshold of 0.05, indicating that the differences in average performance between AIS and its variants are not statistically significant. This suggests that while the variants may occasionally find better solutions, their average performance is comparable to that of the core AIS algorithm.

The Best-to-Worst Difference reflects the variability or consistency of the algorithms. A lower difference indicates more consistent performance. The comparison between AIS and AIS-th shows a significant p-value of 0.04, indicating that AIS-th is more consistent in performance than AIS. However, no significant differences were found between AIS and the other variants (AIS-t and AIS-h), with p-values of 0.06 and 0.10, respectively.

The results demonstrate that the AIS variants, particularly AIS-th, are capable of finding shorter tours and exhibit more consistent performance compared to the core AIS algorithm. However, in terms of average performance, the differences are not statistically significant, suggesting that the overall improvements provided by the variants may be more subtle and situational. These findings highlight the importance of selecting the appropriate variant of AIS based on specific performance goals, such as optimizing the best solution or ensuring consistent performance.

This analysis indicates that the hybrid and tuned AIS variants offer tangible improvements over the core AIS, particularly when focusing on the best solution or performance consistency.

6. CONCLUSIONS

This study demonstrates the effectiveness of enhancing the Artificial Immune System (AIS) through hybridization with the Neighborhood Improvement (NI) heuristic and fine-tuning parameters for solving the Traveling Salesman Problem (TSP). The research was conducted in two main experiments: Experiment A, which investigated the optimal positions for integrating NI within AIS, and Experiment B, which benchmarked the performance of various AIS configurations.

Experiment A identified that integrating NI at position 2 (AIS+NI_{pos2}) consistently yielded better average solutions than integrating NI at positions 1 and 3 for almost all problem sizes. This strategic integration improves the overall quality and diversity of the antibody population by refining solutions in the middle of the iterative process, leading to a more balanced search strategy. Experiment B demonstrated the importance of fine-tuning parameters and hybridizing with NI. The results showed that AIS configurations with both fine-tuning and hybridization (AIS-th) outperformed other configurations regarding solution quality for both symmetric and asymmetric TSP datasets. Despite requiring more computational time, the superior quality of solutions obtained by AIS-th justifies the increased computational effort.

The study contributes to the field of combinatorial optimization by providing a detailed analysis of how hybridization and parameter tuning can enhance AIS performance. The proposed approach of integrating NI within AIS at optimal positions and fine-tuning parameters can be generalized and applied to other combinatorial optimization problems beyond TSP. The findings underscore the importance of strategic hybridization and parameter tuning in developing robust optimization algorithms. Combining these techniques allows for more effective exploration and exploitation of the solution space, leading to higher-quality solutions and more efficient use of computational resources.

Future research could explore adaptive strategies for hybridization and real-time parameter adjustment to enhance AIS performance further. Additionally, extending the proposed approach to other combinatorial optimization problems, such as scheduling and resource allocation, could provide valuable insights into the generalizability of the techniques. Integrating other local search heuristics and advanced metaheuristic frameworks with AIS could also yield significant improvements.

In conclusion, this study demonstrates the potential of hybridizing AIS with NI and fine-tuning parameters to solve complex optimization problems effectively. The proposed approach enhances the performance of AIS and provides a foundation for future research in this area.

Author Contributions

P.T.: investigation, methodology, data analysis, writing an original draft, data curation, review and editing; C.K.: investigation, methodology, data analysis, ethical oversight, publishing and dissemination; W.T.: conceptualization, research design, funding acquisition, project administration, supervision, ethical oversight, data curation, review and editing, publishing and dissemination, collaboration, responsibility for outcomes. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

Adenso-Díaz, B., & Laguna, M. (2006). Fine-Tuning of algorithms using fractional experimental designs and local search. *Operations Research*, *54*(1), 99-114. https://doi.org/10.1287/opre.1050.0243

Akhand, M. A. H., Akter, S., & Rashid, M. A. (2014). Velocity Tentative Particle Swarm Optimization to solve TSP. *2013 International Conference on Electrical Information and Communication Technology (EICT)* (pp. 1-6). IEEE. https://doi.org/10.1109/EICT.2014.6777868

Akram, M., & Habib, A. (2023). Hybridizing simulated annealing and genetic algorithms with Pythagorean fuzzy uncertainty for Traveling Salesman Problem optimization. *Journal of Applied Mathematics and Computing*, *69*, 4451-4497. https://doi.org/10.1007/s12190-023-01935-y

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, *35*(3), 268-308. https://doi.org/10.1145/937503.937505

Boryczka, U., & Szwarc, K. (2019). The Harmony Search algorithm with additional improvement of harmony memory for Asymmetric Traveling Salesman Problem. *Expert Systems with Applications*, *122*, 43-53. https://doi.org/10.1016/j.eswa.2018.12.044

Braun, H. (1991). On solving travelling salesman problems by genetic algorithms. In H.-P. Schwefel & R. Männer (Eds.), *Parallel Problem Solving from Nature* (Vol. 496, pp. 129-133). Springer-Verlag. https://doi.org/10.1007/BFb0029743

Burke, E. K., Cowling, P. I., & Keuthen, R. (2001). Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem. In E. J. W. Boers (Ed.), *Applications of Evolutionary Computing* (Vol. 2037, pp. 203-212). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45365-2_21

Burnet, F. M. (1959). *The clonal selection theory of acquired immunity; the Abraham Flexner lectures of Vanderbilt University*. Cambridge University Press.

Campuzano, G., Obreque, C., & Aguayo, M. M. (2020). Accelerating the Miller–Tucker–Zemlin model for the asymmetric Traveling Salesman Problem. *Expert Systems with Applications*, *148*, 113229. https://doi.org/10.1016/j.eswa.2020.113229

Chandrasekaran, M., Asokan, P., Kumanan, S., Balamurugan, T., & Nickolas, S. (2006). Solving job shop scheduling problems using artificial immune system. *International Journal of Advanced Manufacturing Technology*, *31*, 580-593. https://doi.org/10.1007/s00170-005-0226-3

Chen, S.-M., & Chien, C.-Y. (2011). Solving the Traveling Salesman Problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, *38*(12), 14439-14450. https://doi.org/10.1016/j.eswa.2011.04.163

De Castro, L., & Timmis, J. (2002). *Artificial immune systems: A new computational intelligence approach*. Springer.

De Castro, L., & Von Zuben, F. (2001). The clonal selection algorithm with engineering applications. *Workshop Proceedings of GECCO* (pp. 36-37).

Deng, W., Chen, R., He, B., Liu, Y., Yin, L., & Guo, J. (2012). A novel two-stage hybrid swarm intelligence optimization algorithm and application. *Soft Computing*, *16*, 1707-1722. https://doi.org/10.1007/s00500-012-0855-z

Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, *10*(1), 27-36. https://doi.org/10.1016/0167-6377(91)90083-2

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *Biosystems*, *43*(2), 73-81. https://doi.org/10.1016/S0303-2647(97)01708-5

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

Eiben, A. E., & Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, *1*(1), 19-31. https://doi.org/10.1016/j.swevo.2011.02.001

Engin, O., & Döyen, A. (2004). Artificial immune systems and applications in industrial problems. *Journal of Science*, *17*(1), 71-84.

Freitas, A. A., & Timmis, J. (2003). Revisiting the foundations of artificial immune systems: A problem-oriented perspective. In J. Timmis, P. J. Bentley, & E. Hart (Eds.), *Artificial Immune Systems* (Vol. 2787, pp. 229-241). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-45192-1_22

Garrett, S. M. (2005). How do we evaluate artificial immune systems? *Evolutionary Computation*, *13*(2), 145-177. https://doi.org/10.1162/1063656054088512

Glover, F. W. (1989). Tabu Search - Part I. *ORSA Journal on Computing*, *1*(3), 190-206. https://doi.org/10.1287/ijoc.1.3.190

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc.

Gouveia, L., & Pires, J. M. (1999). The asymmetric travelling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints. *European Journal of Operational Research*, *112*(1), 134-146. https://doi.org/10.1016/S0377-2217(97)00358-5

Greensmith, J., Whitbrook, A., & Aickelin, U. (2010). Artificial immune systems. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (Vol. 146, pp. 421–448). Springer US. https://doi.org/10.1007/978-1-4419-1665-5_14

Hart, E., & Timmis, J. (2008). Application areas of AIS: The past, the present and the future. *Applied Soft Computing*, *8*(1), 191-201. https://doi.org/10.1016/j.asoc.2006.12.004

Huang, C., Li, Y., & Yao, X. (2019). A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation*, *24*(2), 201-216. https://doi.org/10.1109/TEVC.2019.2921598

Joy, G., Huyck, C., & Yang, X.-S. (2023). Review of parameter tuning methods for nature-inspired algorithms. In X.-S. Yang (Ed.), *Benchmarks and Hybrid Algorithms in Optimization and Applications* (pp. 33–47). Springer Nature Singapore. https://doi.org/10.1007/978-981-99-3970-1_3

Kang-Ping, W., Lan, H., Chun-Guang, Z., & Wei, P. (2003). Particle swarm optimization for Traveling Salesman Problem. *2003 International Conference on Machine Learning and Cybernetics* (pp. 1583-1585). IEEE. https://doi.org/10.1109/ICMLC.2003.1259748

Karaboga, D., & Gorkemli, B. (2011). A combinatorial Artificial Bee Colony algorithm for Traveling Salesman Problem. *2011 International Symposium on Innovations in Intelligent Systems and Applications* (pp. 50-53). IEEE. https://doi.org/10.1109/INISTA.2011.5946125

Eberhart, R. C., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence*. Morgan Kaufmann Publishers Inc.

Khan, I., & Maiti, M. K. (2019). A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem. *Swarm and Evolutionary Computation*, *44*, 428-438. https://doi.org/10.1016/j.swevo.2018.05.006

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671-680. https://doi.org/10.1126/science.220.4598.671

Krishna, M., Panda, N., & Majhi, S. (2021). Solving Traveling Salesman Problem using Hybridization of Rider Optimization and Spotted Hyena Optimization Algorithm. *Expert Systems with Applications*, *183*, 115353. https://doi.org/10.1016/j.eswa.2021.115353

Laporte, G. (1992). The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*(2), 231-247. https://doi.org/10.1016/0377-2217(92)90138-Y

Laporte, G., & Nobert, Y. (1980). A Cutting Planes Algorithm for the m-Salesmen Problem. *The Journal of the Operational Research Society*, *31*(11), 1017-1023. https://doi.org/10.2307/2581282

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic algorithms for the Travelling Salesman Problem: A Review of representations and operators. *Artificial Intelligence Review*, *13*, 129-170. https://doi.org/10.1023/A:1006529012972

Lawler, E. L. (1985). *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons.

Li, M., Ma, J., Zhang, Y., Zhou, H., & Liu, J. (2015). Firefly algorithm solving multiple Traveling Salesman Problem. *Journal of Computational and Theoretical Nanoscience*, *12*(7), 1277-1281. https://doi.org/10.1166/jctn.2015.3886

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of Traveling Salesman Problems. *Journal of the ACM*, *7*(4), 326-329. https://doi.org/10.1145/321043.321046

Nagata, Y., & Soler, D. (2012). A new genetic algorithm for the asymmetric Traveling Salesman Problem. *Expert Systems with Applications*, *39*(10), 8947-8953. https://doi.org/10.1016/j.eswa.2012.02.029

Osaba, E., Ser, J. D., Sadollah, A., Bilbao, M. N., & Camacho, D. (2018). A discrete water cycle algorithm for solving the symmetric and asymmetric Traveling Salesman Problem. *Applied Soft Computing*, *71*, 277-290. https://doi.org/10.1016/j.asoc.2018.06.047

Osaba, E., Yang, X.-S., Diaz, F., Lopez-Garcia, P., & Carballedo, R. (2016). An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. *Engineering Applications of Artificial Intelligence, 48,* 59-71. https://doi.org/10.1016/j.engappai.2015.10.006

Panwar, K., & Deep, K. (2021). Discrete Grey Wolf Optimizer for symmetric Traveling Salesman Problem. *Applied Soft Computing, 105,* 107298. https://doi.org/10.1016/j.asoc.2021.107298

Reinelt, G. (1991). TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, *3*(4), 267-384. https://doi.org/10.1287/ijoc.3.4.376

Ruan, D. (1997). *Intelligent hybrid systems : fuzzy logic, neural networks, and genetic algorithms*. Springer.

Sengupta, S., Basak, S., & Peters, R. A. (2019). Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, *1*(1), 157-191. https://doi.org/10.3390/make1010010

Yang, X.-S. (2023). Nature-Inspired algorithms in optimization: Introduction, hybridization, and insights. In X.-S. Yang (Ed.), *Benchmarks and Hybrid Algorithms in Optimization and Applications* (pp. 1–17). Springer Nature Singapore. https://doi.org/10.1007/978-981-99-3970-1_1

Zhang, T., Zhou, Y., Zhou, G., Deng, W., & Luo, Q. (2023). Discrete Mayfly Algorithm for spherical asymmetric Traveling Salesman Problem. *Expert Systems with Applications*, *221*, 119765. https://doi.org/10.1016/j.eswa.2023.119765