Artificial Rabbits Optimization for Traveling Salesman Problem

Yustinus Wahyu Surya Adityatamaa), Eliza Okta Viyantib), Dyah Ari Cahyaningrumc), and Prayoga Yudha Pamungkasd)

Department of Industrial Engineering, Bina Nusantara University, Jakarta, Indonesia

d) Corresponding author: [prayoga.pamungkas@binus.ac.id](mailto:prayoga.pamungkas@binus.ac.id)

a) yustinus.adityatama@binus.ac.id

b) eliza.oktaviyanti@binus.ac.id

c) dyah.cahyaningrum@binus.ac.id

**Abstract.**  Artificial Rabbits Optimization (ARO) has been efficiently applied to solve Traveling Salesman Problem (TSP). Three major phases of rabbit foraging behavior are mimicked in ARO: detour foraging, random hiding, and energy shrink. During the stage of detour foraging, since rabbits look for the optimum path by detour, solution diversity is enhanced. The random hiding phase introduces stochastic variation, which enables the algorithm to jump out of a local optimum by randomly changing the search trajectory. The energy shrink stage of the algorithm incrementally reduces the search space to facilitate refined exploitation of the promising routes. Comparative studies were conducted with established algorithms like Genetic Algorithm, Simulated Annealing, and Particle Swarm Optimization; it emerged that ARO produced the shortest total distance of 33.68 units among the four benchmarked algorithms. These results demonstrate the power and robustness of Artificial Rabbits Optimization for TSPs and allow the inference of its good prospects to turn out to be a robust metaheuristic framework for the resolution of a wide scope of combinatorial problems.

**Keywords:** Artificial Rabbits Optimization, Traveling Salesman Problem, Metaheuristics, Combinatorial Problem

# INTRODUCTION

The Traveling Salesman Problem (TSP) represents a pure optimization problem and has been attracting researchers for years because of the combinatorial complexity and very wide range of applications. Formulation of the problem is to find the shortest possible route through a number of cities that allows a traveling salesman to visit each city once and return to the initial city. Though easy to define, TSP is an NP-hard problem, meaning it cannot be solved in polynomial time [1]. This difficulty has thus motivated the design of a huge number of exact methods and heuristic approaches for the solution of TSP to optimality or close to optimality in an efficient way. Interest in optimizing the TSP remains high for three main reasons: (1) it is computationally challenging despite its simple formulation, (2) it has wide applicability in various engineering domains, and (3) it serves as a standard benchmark for comparing optimization methods [2]. From among these, metaheuristic methods have attracted a lot of attention lately, as they tend to return good-quality solutions within reasonable computational times.

Metaheuristics are high-level algorithm that employ strategies to escape local optima and explore the solution space more extensively. Metaheuristic are design to solve complex optimization problems, combinatorial real-world optimation problem where other optimization methods have failed to be either effective or efficient [3]. They have been successfully applied to a variety of optimization problems, including TSP. Examples of well-known metaheuristics include Genetic Algorithms (GA), Simulated Annealing (SA), and Particle Swarm Optimization (PSO). These algorithms often mimic natural processes and behaviors, leveraging mechanisms such as selection, mutation, pheromone trails, and social interactions to guide the search for optimal solutions. A metaheuristic will be successful on a given optimization problem if it can provide a balance between the exploration and exploitation [4]. The flexibility and robustness of metaheuristics make them particularly suitable for tackling complex and large-scale instances of TSP.

One of the recent novel metaheuristics inspired by the social and foraging behaviors of rabbits is called Artificial Rabbit Optimization (ARO). ARO imitates ways in which rabbits search for food, communicate with other members, or run away from danger. ARO has two main strategy in solving optimization problem, food foraging and random hiding [5]. This algorithm introduces new exploration and exploitation mechanisms that balance the search between local refining and global optima searching. ARO already implemented into feature selection of medical diagnosis [6], IoT Botnet Exploitation Mitigation [7], and bridge network optimization problem [8]. Inherent in the nature of ARO are dynamic search strategies and adaptiveness, leaving room for further improvement gains in solving complex optimization problems such as TSP. Preliminary studies have already shown very promising results for ARO in finding competitive solutions compared to traditional metaheuristics.

The application of ARO to TSP involves representing potential solutions as sequences of cities and applying the algorithm's operators to modify these sequences iteratively. The fitness value will be evaluated based on the total traveled distance, with the aim of reducing it. ARO makes use of mechanisms like path crossover, mutation, and local search to improve solution quality. Besides, ARO has adaptive parameters that change the behavior of the algorithm depending on the state of the search, hence allowing dynamic balance between exploration and exploitation. The adaptiveness will, therefore, assume paramount importance when it has to sail through the complex solution landscape of the TSP.

# METHODS

The Artificial Rabbit Optimization algorithm (ARO) is known for its very impressive performance in solving complex optimization problems, far beyond those using many conventional and state-of-the-art approaches. Artificial Rabbit Optimization (ARO) is a novel metaheuristic inspired by the social and foraging behaviors of rabbits [9]. ARO imitates the rabbit's intelligent foraging behavior via adaptive search strategies, thereby enhancing its capability to avoid local optima and effectively converge onto global solutions. Specifically, dynamic parameter adjustment has been included in the algorithm by balancing the position of the search agents—rabbits—between exploration and exploitation. As such, ARO efficiently explores highly complex and high-dimensional search spaces, which is key to solving intricate optimization problems.

The other excellence element in ARO is a memory-based mechanism to store and make use of information about the best solutions found during the process of search. This feature enables ARO to refine its search strategy at each iteration by learning from the past experience in order to make more informed decisions. Inherent in this algorithm is a diversity preservation technique that assists in keeping a broad perspective on the search, reducing the possibility of premature convergence. Technical characteristics of ARO drive it to be very effective in issues connected with multi-modal functions, for which many other algorithms cannot demonstrate such a balance of exploration and exploitation.

Artificial Rabbit Optimization algorithm (ARO) initializes a population of possible solutions and assesses their fitness with respect to travel distance. These solutions are then refined iteratively through adaptive strategies balancing exploration and exploitation. ARO has two main stage, detour foraging and random hiding, at has transition from those 2 stage through energy shrink strategy.

1. Detour Foraging (exploration)

Detour foraging is one of the behaviors of rabbits in searching for food when rabbits do not find food on the path they are taking. When rabbits do not find food on the path they are taking, they will take a different path to look for possible new food sources. In the context of optimization algorithms, this detour path represents a wider exploration of the solution space with the aim of finding an optimal solution that might be missed if only following one search path. This Detour Path is important because it can help the algorithm avoid getting stuck in a local minimum, which is a condition where the algorithm finds a solution that is good enough but not the best solution globally. The formula used to calculate the detour path in ARO varies depending on the specific implementation of the algorithm. However, in general, the formula involves a combination of the rabbit's current position, the rabbit's best position (global best), and a random component.

|  |  |
| --- | --- |
| 𝑖 (𝑡 + 1) = j (𝑡) + 𝑅 ⋅ (𝑖 (𝑡) − j (𝑡)) + *round*(0.5 ⋅ (0.05 + )) ⋅ , 𝑖, j = 1, …, 𝑛 and j ≠ 𝑖 | (1) |
| R = L ⋅ c | (2) |
|  | (3) |
| *c*(*k*) = *k* =1, …, *d* and *l* = 1,…, | (4) |
| g = randperm(d) | (5) |
| n1 ~ N(0,1) | (6) |

1. Random Hiding (exploitation)

The random hiding in the Artificial Rabbits Optimization algorithm allows for the exploration of different areas in the solution space. It is a strategy to update the rabbits positions so that it ensures exploration and avoids local optima. In the ARO, at every iteration, a rabbit will always reproduce d burrows around itself along each dimension of the search space and it will always randomly choose one from all burrows for hiding to reduce the possibility of being preyed. The following equation is given in this regard. The 𝑗th burrow of the 𝑖th rabbit is generated by:

|  |  |
| --- | --- |
| 𝑖,𝑗(𝑡) = (𝑡) + 𝐻 ⋅ 𝑔 ⋅ (𝑡), 𝑖 = 1, …, 𝑛 and 𝑗 = 1, …, 𝑑 | (7) |
| 𝐻 = ⋅ 𝑟4 | (8) |
| 𝑛2 ∼ 𝑁(0, 1) | (9) |
| *gr*(*k*) = k =1, …, d | (10) |

From Eq. (7), d burrows are generated in the neighborhood of a rabbit along every dimension. H is the hiding parameter, and it is decreased from 1 to 1/T linearly with a random perturbation over the course of iterations. This parameter initially generates these burrows in a bigger neighborhood of a rabbit. As the iterations increase, this neighborhood gets decreased. As explained, rabbits must undergo searching and catching by predators. To be saved, rabbits must find a den for sheltering in a safe place. So they can't afford to choose randomly one of their dens for sheltering to avoid catching. The following equations are proposed to model this random strategy mathematically:

|  |  |
| --- | --- |
| 𝑖 (𝑡 + 1) = 𝑖 (𝑡) + 𝑅 ⋅ ( ⋅ 𝑖,𝑟(𝑡) − 𝑖 (𝑡)), 𝑖 = 1, …, 𝑛 | (11) |
| gr(*k*) = k =1, …, d | (12) |
| 𝑖,𝑟(𝑡) = 𝑖 (𝑡) + 𝐻 ⋅ ⋅ 𝑖 (𝑡) | (13) |

Where 𝑖,𝑟 b is a randomly chosen burrow for hiding from its d burrows, and 𝑟4 and 𝑟5 are two random numbers within the range of (0,1). According to Eq. (11), the 𝑖th searching individual will try to update its position towards the randomly chosen burrow from its d burrows. Once one of the two actions, detour foraging and random hiding, is completed, the position update of the 𝑖th rabbit is updated as:

|  |  |
| --- | --- |
| 𝑖 (𝑡 + 1) = k =1, …, d | (14) |

The equation expresses that if the candidate position of the 𝑖th rabbit has better fitness than that of the current one, then it will abandon the current position and stay at the candidate position generated by either Eq. (1) or Eq. (11).

1. Energy shrink (switch from exploration to exploitation)

The Artificial Rabbit Optimization (ARO) algorithm incorporates an energy reduction strategy inspired by the behavior of rabbits. As rabbits approach an optimal solution or food source, they gradually decrease their search efforts or energy. Initially, rabbits frequently engage in detour foraging, but as iterations progress, they tend to hide randomly. This behavior gradually reduces the rabbit’s energy over time. This mechanism balances exploration and exploitation during the optimization process, where extensive exploration helps avoid local optima, while the decreasing energy focuses the search on the best solution.Therefore, an energy factor, A(t), is designed to model the switch from exploration to exploitation, defined as follows:

|  |  |
| --- | --- |
|  | (15) |

Where r is the random number in (0,1). From Eq.15, the energy factor A(t) declines towards zero with oscillation amplitude over iterations. High energy factors show that a rabbit has enough energy for detour foraging, whereas low energy factors suggest the rabbit is less active and needs random hiding. To study the effect of the energy factor on the searching behavior of the searching algorithm, the probability of *A* > 1 is computed. Let θ = then *A*(*t*) = 2 (θ , the probability that *A* > 0 is obtained by:

|  |  |
| --- | --- |
|  | (16) |

Therefore, the probability of detour foraging will be approximately 0.5 during the iterative process. In other words, the ARO algorithm deals with almost an equal proportion of detour foraging and random hiding in the iterative process, which has a great contribution to balancing exploration and exploitation. Pseudocode of Artificial Rabbits Optimization is shown in **Algorithm.**

**Algorithm** Artificial Rabbits Optimization

1: **Initialize** set of rabbits randomly (*Xi)*

2: **Evaluate** fitness of each rabbit (*Fiti)* and determine the best solution as *Xbest*

3: **While** the stop criterion is not satisfied **do**

4: **for** (*i* = 1 : *n*)

5: Calculate energy factor *A* using Eq.15

6: **if** (*A*>1) **then**

7: Choose a rabbit randomly from other other individuals

8: Determine *R* using Eq.2-6

9: Calculate detour foraging through Eq.1

10: Compute the fitness *Fiti*

11: Update the position of the current individual using Eq.14

12: **else**

13: Producing random number of *d* burrows and randomly pick one as hiding using Eq.13

14: Derive random hiding using Eq.11

15: Compute the fitness *Fiti*

16: Adjust the individual position through Eq.14

17: **end if**

18: Renew the current best solution as *Xbest*

19: **end for**

20: **End While**

21: **Return** *Xbest*

# RESULTS AND DISCUSSION

In this work, the dataset used is the Burma14 TSPLib instances [10]. This is a famous library of test problems and in wide use within the operations research community for benchmarking and comparing the different algorithms that have been developed to solve the TSP. The performance assessments with the Burma14 dataset inclusion in this study underline its significance and usefulness as one of the tools towards exploring optimization techniques. This dataset which consist of 14 nodes is small and simple in an academic sense, it is very useful for testing preliminary algorithms. The distances between them are Euclidean. Each city is represented with coordinates on a 2D plane, and distances are computed by the Euclidean formula, which makes this dataset good for testing algorithms on realistic, spatially defined problems. It is an instance of symmetric TSP, where it is assumed that the distance from city A to city B is the same as the distance from city B to city A. This removes directionality and therefore helps to simplify the problem. The Burma14 dataset is very often used in research because of its manageable size and simplicity; it provides a controlled environment to assess and compare different optimization algorithms for their performance and efficiency.

**TABLE 1** shows the current performance for some of the well-known metaheuristics including Genetic Algorithms (GA) [11], Simulated Annealing, and Particle Swarm Optimization [12]. These algorithms are very often deployed due to their ability to solve a lot of optimization problems. In spite of the effectiveness of metaheuristics in optimization problems, solution gaps toward the optimum solution exist. It therefore gives a very good opportunity for developing new algorithms that decrease the solution gap to near the optimum value.

**TABLE 1**. Solution Gap between Optimum and Metaheuristics

| Methods | Total Distance | Gap (%) |
| --- | --- | --- |
| Optimum | 30.87 | 0% |
| GA | 33.96 | 9.97% |
| PSO | 35.37 | 14.5% |
| SA | 34.98 | 13.27% |

The optimum solution of the Burma14 instance dataset is 30.87, while for the other well-known metaheuristics algorithms like GA, PSO, and SA, it goes to 33.96, 35.37, and 34.98 respectively. Referring to the optimum solution, these algorithms have gap solutions with the smallest gap of 9.97 per cent. The performance gap underlines the challenges in approximating the optimum solution of the TSP using metaheuristics. It reflects the fundamental trade-offs between solution quality and processing efficiency when solving complex optimization problems with metaheuristic algorithms. In this respect, gaps like this are closed through continuous research and development aimed at improving the algorithmic approaches to enable solutions closer to the optimum value while remaining computationally feasible.

The ARO for TSP was implemented in Python and run on a computer system equipped with an AMD Ryzen 5 5600H CPU and 16GB of memory. This provided a viable environment to test the algorithm's effectiveness and efficiency. Before conducting these steps, the algorithm was set up by specifying the ARO parameters regarding the number of search agents and iteration number. The efficiency of TSP is able to treat the parameters for better settings while solving the problem. **TABLE 2** shows specific parameters used in this research work. **TABLE 2** presents parameters used for the study.

**TABLE 2**. Parameter Setting ARO Algorithm

| Parameter | Value |
| --- | --- |
| Population Size | 100 |
| Iteration | 1,000 |

Parameters of the ARO were fine-tuned in such a way as to make it sure that the balance of this algorithm's exploration and exploitation is appropriate to solve TSP. Then, with such parameters, ARO could return a very good solution to TSP after running. The best route identified started from node 13 and went sequentially through nodes 11 – 5 – 4 – 3 – 2 – 6 - 12 – 10 – 8 – 9 – 0 – 7 – 1, and eventually returned to node 13. The total distance for this route proved to be 33.68, thus affirming the capability of the programme in giving back the best solutions to very complex combinatorial optimization problems.

Convergence rate on **FIGURE 1** illustrates the progression of the ARO in solving the TSP. The objective value is plotted over 1,000 iterations. The chart starts with a high objective value of about 44, which significantly decreasing all through the first 50 iterations. The objective value still reducing its number until almost 200 iteration. At the 200th iteration, the solution value stabilized and remained steady through to the 1000th iteration, where the final solution value of 33.68 was obtained. This steep drop in early iteration states means that ARO is very efficient in finding out quick solutions and exploring them.

A graph with blue lines

Description automatically generated

**FIGURE 1**. Global Objective of ARO for TSP

The provided chart illustrates the progression of exploration and exploitation percentages over 1000 iterations in solving the Traveling Salesman Problem using a bio-inspired metaheuristic. Initially, the exploration phase dominates, comprising nearly 100% of the search effort, as seen in **FIGURE 2**. This high level of exploration is critical for thoroughly scanning the solution space and identifying diverse potential solutions. Around the 200th iteration, a notable shift occurs where exploitation starts to increase rapidly, surpassing exploration. This transition signifies a focus on refining and optimizing the identified solutions, reducing randomness, and honing in on more promising areas of the solution space. As the iterations progress, the exploitation percentage stabilizes at around 80%, while exploration diminishes to approximately 20%. This balance reflects the algorithm's adaptive mechanism, ensuring that sufficient exploration prevents premature convergence while primarily concentrating on exploiting the best solutions found. The dynamic adaptation of these percentages underscores the efficiency and robustness of the bio-inspired metaheuristic in achieving optimal or near-optimal solutions for the Traveling Salesman Problem.

A graph of an orange and blue line

Description automatically generated

**FIGURE 2**. Exploration vs exploitation percentages each iteration

In this paper, the efficiency of the Artificial Rabbits Optimization Algorithm in solving a Traveling Salesman Problem is compared with the existing metaheuristics using the burma14 instance. The Genetic algorithm, Particle Swarm Optimization, and Simulated Annealing are chosen to benchmark the performance of ARO. The result of the comparison is given in **TABLE 3**, which includes the total distances obtained by each algorithm and their gaps from the optimal solution.

**TABLE 3**. Result comparison over metaheuristics and optimum solution

| Methods | Total Distance | Gap (%) |
| --- | --- | --- |
| Optimum | 30.87 | 0 % |
| GA | 33.96 | 9.97 % |
| PSO | 35.37 | 14.5 % |
| SA | 34.98 | 13.27 % |
| ARO | 33.68 | 9.1% |

Addressing the Traveling Salesman Problem, ARO outperforms other metaheuristic algorithms. According to **TABLE 3**, ARO resulted in a total distance of 33.68, which is just 9.1 percent far from the optimum distance, 30.87. Of all the techniques reviewed, this consequence goes nearest to the optimum solution, outperforming GA with a distance of 33.96, PSO with 35.37, and SA with 34.98. These effects obviously demonstrate that ARO is able and productive for combinatorial optimization issues by achieving results near an optimal solution with a smaller discrepancy than other generally perceived metaheuristics. The solution generated by ARO offered a lucid explanation alongside requisite detail for intricate problems.

# CONCLUSIONS

A classic combinatorial optimization like the Travelling Salesman Problem was attempted to solve using Artificial Rabbits Optimization, which belongs to the category of bio-inspired metaheuristics. Basically, inspired by the foraging behavior of rabbit, consist of detour foraging and random hiding which has transition stage through energy shrink. ARO’s parameters that have been used are population size equated to 100 and the iterations were at 1,000 to find optimized efficient routes. This showed ARO to have performed the best route in the setup relating to high-performance computing, wherein the total distance was 33.68. In order to validate the results further, ARO's results are compared with other popular metaheuristic algorithms such as Genetic Algorithm, Simulated Annealing, and Particle Swarm Optimization. Among these, ARO emerged as the best among the selected approaches with a distance nearest to the optimum at 33.68 and only 9.1 percent far from the optimal solution. These results illustrate the strength and high efficiency of ARO for the solution of any problem of combinatorial optimization, making it one of the leading metaheuristic methods in attempts to solve such complex problems as TSP. Results of the conducted research work evidence that ARO can be referred to as quite a promising and potent tool for further applications in the sphere of industrial optimization, not mentioning related fields connected with sophisticated strategies of problem-solving.

# References

1. Dahiya, C. and S. Sangwan, *Literature review on travelling salesman problem.* International Journal of Research, 2018. **5**(16): p. 1152-1155.

2. Fogel, D.B., *An evolutionary approach to the traveling salesman problem.* Biological Cybernetics, 1988. **60**(2): p. 139-144.

3. Ólafsson, S., *Metaheuristics.* Handbooks in operations research and management science, 2006. **13**: p. 633-654.

4. Boussaïd, I., J. Lepagnot, and P. Siarry, *A survey on optimization metaheuristics.* Information sciences, 2013. **237**: p. 82-117.

5. Zoralioğlu, Y. and S. Arslan, *COMPARISON OF METAHEURISTIC ALGORITHMS WITH DIFFERENT PERFORMANCE CRITERIA.* Adıyaman Üniversitesi Mühendislik Bilimleri Dergisi, 2023. **10**(21): p. 266-275.

6. Awadallah, M.A., et al., *An enhanced binary artificial rabbits optimization for feature selection in medical diagnosis.* Neural Computing and Applications, 2023. **35**(27): p. 20013-20068.

7. Almseidin, M., et al., *DT-ARO: Decision tree-based artificial rabbits optimization to mitigate IoT Botnet exploitation.* Journal of Network and Systems Management, 2024. **32**(1): p. 14.

8. Ramu Naidu, Y. *A New Version of Artificial Rabbits Optimization for Solving Complex Bridge Network Optimization Problem*. Springer.

9. Wang, L., et al., *Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems.* Engineering Applications of Artificial Intelligence, 2022. **114**: p. 105082.

10. Reinhelt, G., *{TSPLIB}: a library of sample instances for the TSP (and related problems) from various sources and of various types.* URL: <http://comopt>. ifi. uniheidelberg. de/software/TSPLIB95, 2014.

11. Yang, W., et al. *Improved shuffled frog leaping algorithm for solving multi-aisle automated warehouse scheduling optimization*. Springer.

12. Li, L., et al. *A discrete artificial bee colony algorithm for TSP problem in Bio-Inspired Computing and Applications*. in *Heidelberg: Springer Berlin Heidelberg*. 2012. Berlin: Springer.