A New Hybrid Algorithm Based on Ant Colony Optimization and Recurrent Neural Networks with Attention Mechanism for Solving the Traveling Salesman Problem

Anderson Nguetoum Likeufack[1] **and Mathurin Soh**[1]

[1]University of Dschang, Cameroon

*E-mail : nguetoumanderson@gmail.com,mathurinsoh@gmail.com

DOI : 10.46298/arima.13340
Submitted on April 3, 2024 - Published on January 17, 2025
Volume : **42** - Year : **2025**
Special Issue : **CRI 2023 - 2024**
Editors : *Paulin Melatagia, René Ndoundam, Kamel Barkaoui, Blaise Omer Yenke*

Abstract

In this paper, we propose a hybrid approach for solving the symmetric traveling salesman problem. The proposed approach combines the ant colony algorithm (ACO) with Recurrent Neural Networks based on the attention mechanism. The idea is to use the predictive capacity of Recurrent Neural Networks to guide the behaviour of ants in choosing the next cities to visit and to use the prediction results of the latter to update the pheromone matrix, thereby improving the quality of the solutions obtained. In concrete terms, attention is focused on the most promising cities by taking into account both distance and pheromone information thanks to the attention mechanism, which makes it possible to assign weights to each city according to its degree of relevance. These weights are then used to predict the next towns to visit for each city. Experimental results on instances TSP from the TSPLIB library demonstrate that this hybrid approach is better compared to the classic ACO.

Keywords

Traveling Salesman Problem; Recurrent Neural Networks; Hybridization; Attention Mechanism ; Ant Colony Algorithm.

I INTRODUCTION

In the field of robotics, motion and path planning play a crucial role in accomplishing complex tasks. Robots often need to perform efficient and optimized movements to reach different points of interest, collect information, or interact with their environment. Solving the Traveling Salesman Problem (TSP) in robotics can improve the operational efficiency of robots by reducing travel times and unnecessary movements. However, the TSP is considered an NP-hard problem[8, 12], meaning that there is no algorithm that can solve it in a reasonable amount of time.

Faced with this complexity, researchers have proposed several methods to obtain an approximate solution to the TSP. One of these methods is the Ant Colony Optimization (ACO) metaheuristic proposed by Marco Dorigo[5], inspired by experiments on the behavior of real ants[2]. Although effective in solving small instances, it has limitations in exploring the solution space, making it inefficient for large-scale TSP instances. One approach to overcome this limitation would be to combine ACO with neural networks with attention mechanisms to intelligently guide the ants in promising areas. In this context, we propose in this paper a new hybrid approach called ACO-RNN (Ant Colony Optimization-Recurrent Neural Network) to enhance ACO exploration by introducing a novel update formula for the pheromone matrix.

In the rest of this document, we provide a formal description of the TSP and its applications in the second section. We briefly review the state of the art in TSP resolution in the third section. Next, we explore the hybridization techniques used in our approach in the fourth section. In the fifth section, we provide a detailed description of the ACO-RNN approach we developed. Finally, in the last section, we present the results of our experiments.

II FORMAL DESCRIPTION OF THE TSP

The Traveling Salesman Problem involves finding an optimal tour for a traveling salesman who wants to visit exactly n cities and return to the starting city[8, 12]. Mathematically, the TSP can be formulated as follows:

Let $G = (V, E)$ be a complete undirected graph, where V represents the set of cities to be visited, and E represents the set of edges connecting the cities. Each edge $(i, j) \in E$ is associated with a distance or cost $d(i, j)$ representing the distance between cities i and j. The objective of the TSP is to find a Hamiltonian cycle of minimum length in the graph G, i.e., a path that visits each city exactly once and returns to the starting city. The total length of the cycle is given by the objective function.

$$\begin{aligned}
\text{Minimize} \quad & \sum_{(i,j)\in E} d(i,j)x(i,j) && (1)\\
\text{Subject to} \quad & \sum_{j\in V} x(i,j) = 1, \quad \forall i \in V && (2)\\
& \sum_{i\in V} x(i,j) = 1, \quad \forall j \in V && (3)\\
& \sum_{i\in S} \sum_{j\in V\setminus S} x(i,j) \leq |S| - 1, \quad \forall S \subset V, |S| \geq 2 && (4)\\
& x(i,j) \in 0, 1, \quad \forall (i,j) \in E && (5)
\end{aligned}$$

The TSP has attracted significant interest from the scientific community due to its applications beyond robotics. It finds applications in various other domains, such as logistics for optimizing delivery routes, bioinformatics for analyzing DNA sequences and finding common motifs, electronic circuit design for optimizing wire paths in integrated circuits, etc.

III STATE OF THE ART

The TSP has been the subject of extensive research for many years, leading to the proposal of multiple methods and algorithms to solve it. In this section, we will focus specifically on hybrid techniques and those using deep learning to solve the TSP.

Hybridization involves combining features from two different methods to leverage the advantages of both. Hybrid metaheuristic algorithms were introduced in the works of Glover [1], J. J. Grefenstette [3], and Mühlenbein et al.[4]. According to the classification of hybridization methods proposed in [7], the hierarchical classification is characterized by the level and mode of hybridization [9]. The level of hybridization can be either low-level or high-level. In low-level hybridization, a metaheuristic replaces a specific operator of another method that encompasses it. On the other hand, in high-level hybridization, each metaheuristic retains its own characteristics throughout the hybridization. Each level of hybridization gives rise to two cooperation modes: the relay mode and the co-evolutionary mode. In the relay mode, methods are executed sequentially, where the result of the first method becomes the input to the next method. When different methods work in parallel to explore the search space, it is called the co-evolutionary mode[9]. In this context, Ant-Q was developed by Gunter Stützle and Holger H. Hoos in 1996 [6], as an improvement of the Ant System. Ant-Q combines ACO with q-learning technique by integrating a learning mechanism based on a Q-value table. This allows it to combine pheromone-based exploration with reward-based exploitation, thus improving the search performance and adaptability of the algorithm. Pu Y-F et al [18] propose a new conceptual formulation of the Fractional Order Ant Colony Algorithm (FACA), which is based on fractional long-term memory. This approach aims to improve the optimization capability of traditional ant colony algorithms by modifying the transition behavior of the ants. Similarly, a cooperative parallel hybrid approach proposed in the article [13] by Gulcu was developed by combining the ACO algorithm to generate solutions in parallel with the 3-Opt local search heuristic to improve these solutions. Similarly, PSO-ACO-3Opt [10] uses the PSO algorithm to adjust ACO parameters and integrates the 3-Opt technique to avoid premature convergence of ACO. Xiaoling et al propose a new hybrid approach based on the Adaptive State Slime Mold (SM) organism model and the Fractional Order Ant Colony System (SSMFAS) [16] to tackle the TSP.

The approach developed in this paper uses a similar process to Ant-Q[6], with the difference being that it is based on pheromone exploration combined with predictions from a neural network using the attention mechanism. With the advent of artificial intelligence, some researchers have been using deep learning techniques to solve the TSP. Notably, the work of Bresson et al. proposes a new approach called TSP Transformer[14]. This model is based on a classic Transformer encoder with multi-head attention and residual connections, but it uses batch normalization instead of layer normalization. In this approach, decoding is performed in an auto-regressive manner, and a self-attention block is introduced in the decoder part. Minseop et al.[15] propose a new CNN-Transformer model based on partial self-attention, where attention is computed only on the recently visited nodes in the decoder. This is because the linear embedding in the standard Transformer model does not take into account local spatial information and has limitations in learning local compositionality.

In this paper, a similar idea is employed by combining the Ant Colony Optimization (ACO) process with the results of predictions from a neural network using the attention mechanism. This hybrid approach, named ACO-RNN, aims to leverage the predictive power of neural networks to intelligently guide the ants in exploring promising areas while optimizing the pheromone-based exploration process. By combining the strengths of both ACO and neural networks with attention mechanisms, the proposed approach seeks to improve the exploration efficiency and overall performance for solving the TSP.

IV PRINCIPLE OF USED HYBRIDIZATION TECHNIQUES

4.1 Principle of ACO

The classical Ant Colony Optimization (ACO) proceeds as follows [5]:

1. Initialization: Place each ant on a city randomly.
2. Construction of solutions: Each ant moves from one city to another following specific rules. The probability of choosing a particular city depends on the amount of pheromone deposited on that city and the distance between cities.
3. Pheromone update: After all ants have constructed their solutions, the amount of pheromone deposited on each edge is updated. Ants deposit more pheromone on the edges of higher-quality solutions. The update of the pheromone matrix is done using the formula:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{6}$$

   where the symbols have the following meanings:
   -τ_{ij} represents the amount of pheromone on the edge connecting vertex i to vertex j,
   -ρ is the pheromone evaporation rate,
   -m is the number of ants in the colony,
   -$\Delta\tau_{ij}^{k}$ is the amount of pheromone deposited by ant k on the edge ij,

4.2 Autoencoder Model with Attention Mechanism

An autoencoder is a type of algorithm designed to learn an "informative" data representation that can be utilized for various applications by effectively reconstructing a set of input observations. [17]

In this architecture, the encoder is constructed using an LSTM layer that returns sequences. Then, the attention mechanism is applied to the encoder's output to highlight important parts of the sequence. Next, the decoder uses another LSTM layer to generate the reconstructed output. The overall model is created by connecting the input to the outputs of the encoder and decoder.

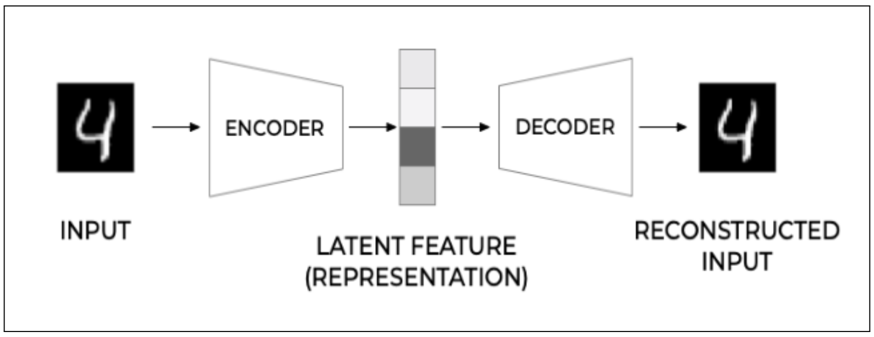

Figure 1: General structure of an autoencoder [17]

In our case:

- **Encoder:** The input to the model is a sequence representing the initial encoded solutions. An LSTM is used to capture the sequential information, and an attention layer is applied to focus on the important parts. The output of the attention layer represents the features extracted by the encoder.

- **Decoder:** The input to the decoder is a sequence of features extracted by the encoder. An LSTM generates an output sequence based on these features, and a Dense layer with softmax predicts the probability of each city to be visited.
- **Autoencoder:** The autoencoder links the input to the outputs of the decoder and encoder. It encodes the initial solutions, then decodes them to generate an output sequence. The aim is to reconstruct the input accurately in order to learn useful representations.

V HYBRIDIZATION OF ANT COLONY OPTIMIZATION AND NEURAL NETWORKS WITH ATTENTION MECHANISM

5.1 Basic Idea

The fundamental idea of our approach, which is a low-level co-evolutionary hybridization, is to guide the behavior of ants using predictions from a Recurrent Neural Networks with an attention mechanism. Traditionally, ACO uses pheromones to guide ants towards neighboring cities and explore the solution space. However, this approach may be limited in terms of efficiency and quality of solutions obtained. The objective of this approach is to use a Recurrent Neural Networks to predict the next cities to visit based on the best solution obtained by ACO so far. This approach is divided into two main steps.

5.2 Resolution Steps

5.2.1 Model Training

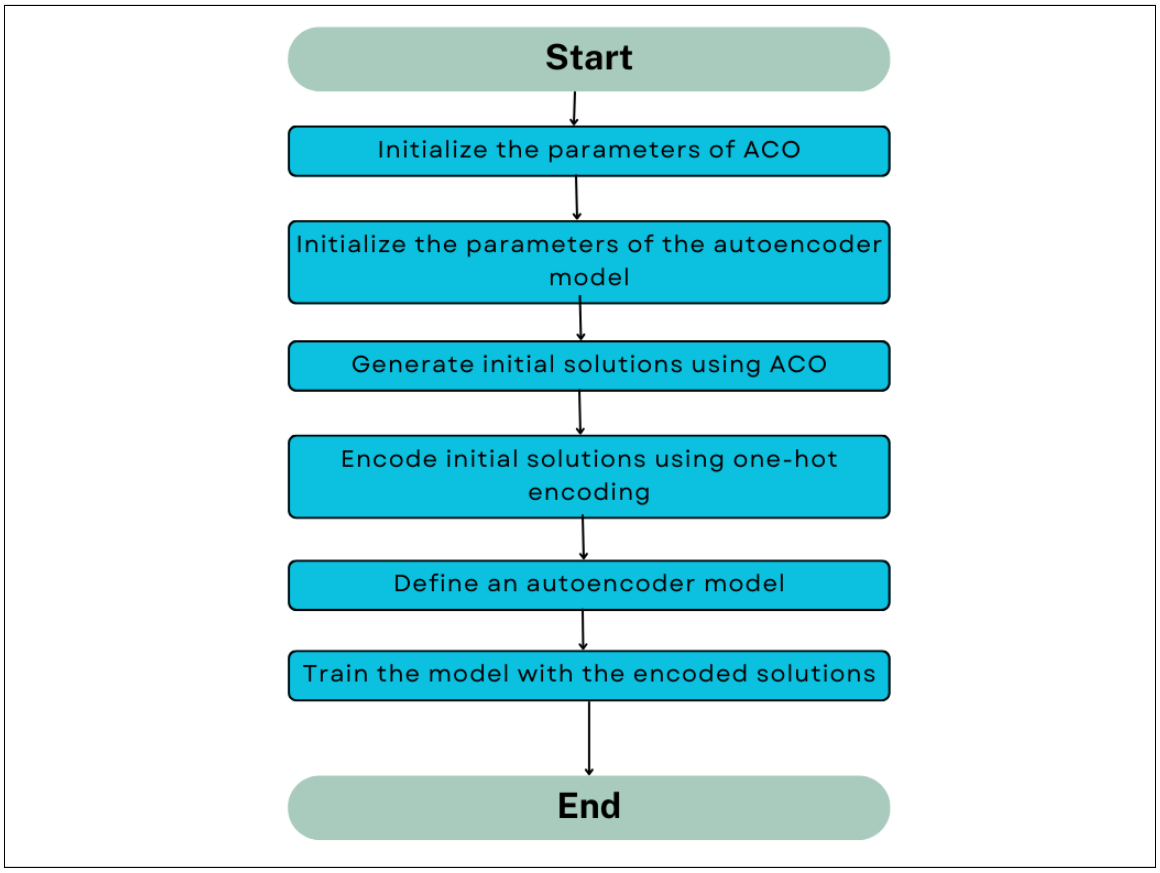
Start
Initialize the parameters of ACO
Initialize the parameters of the autoencoder model
Generate initial solutions using ACO
Encode initial solutions using one-hot encoding
Define an autoencoder model
Train the model with the encoded solutions
End

Figure 2: Algorithm flowchart for model training phase

As part of training the model, we use ACO to generate initial solutions. We randomly place ants on cities that will perform cuter a full turn. Once these solutions appear, we encode the best solution into a data format suitable for training the model using One-Hot(per position) encoding. One-hot encoding [11], This allows us to represent the solution in a structured and coherent way, thus making it easier to learn the model.

To illustrate how we use one-hot encoding, assume a 5-city TSP problem with solution [2, 0, 3, 1, 4]. To encode this solution using one-hot encoding, each city will be represented by a binary vector of size 5, where all values are zero, except the one corresponding to the city index, which will be one. Here is the one-hot encoding for the given TSP solution:

- **City 2 :** [0, 0, 1, 0, 0]
- **City 0 :** [1, 0, 0, 0, 0]
- **City 3 :** [0, 0, 0, 1, 0]
- **City 1 :** [0, 1, 0, 0, 0]
- **City 4 :** [0, 0, 0, 0, 1]

After representing each city in the TSP solution as a distinct binary vector. We form a tensor by concatenating these one-hot vectors into an ordered sequence, where each vector represents the state of a city at a certain time. In our example, this tensor will be:

[[0, 0, 1, 0, 0], [1, 0, 0, 0, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 0], [0, 0, 0, 0, 1]] .

Then we train a recurrent neural network with attention mechanism on this sequence of one-hot vectors to allow the model to learn the sequential relationships between cities and capture the dependencies and relationships between cities. Instead of treating all cities in the sequence uniformly, the attention mechanism allows the model to give more weight or importance to cities that are deemed more relevant for predicting the next city to visit.

5.2.2 Finding the Solution

Finding the final solution involves the iterative execution of ACO. At each iteration, the update of the pheromone matrix is carried out based on the predictions provided by the model. This approach combines the use of ACO to explore the search space and the model's ability to provide predictions based on the information learned during training Cf. Figure 3. As a result, the update of the pheromone matrix takes into account both the knowledge of ACO and the information provided by the model, allowing for a more effective exploitation of both approaches for finding the optimal solution. The new formula for updating the pheromone matrix that we propose is as follows:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \text{predictions}[i, j] \tag{7}$$

where the symbols have the following meanings:

-τ_{ij} represents the amount of pheromone on the edge connecting vertex i to vertex j,

-ρ is the pheromone evaporation rate,

-$predictions[i, j]$ corresponds to the probability predicted by the model between city i and j .

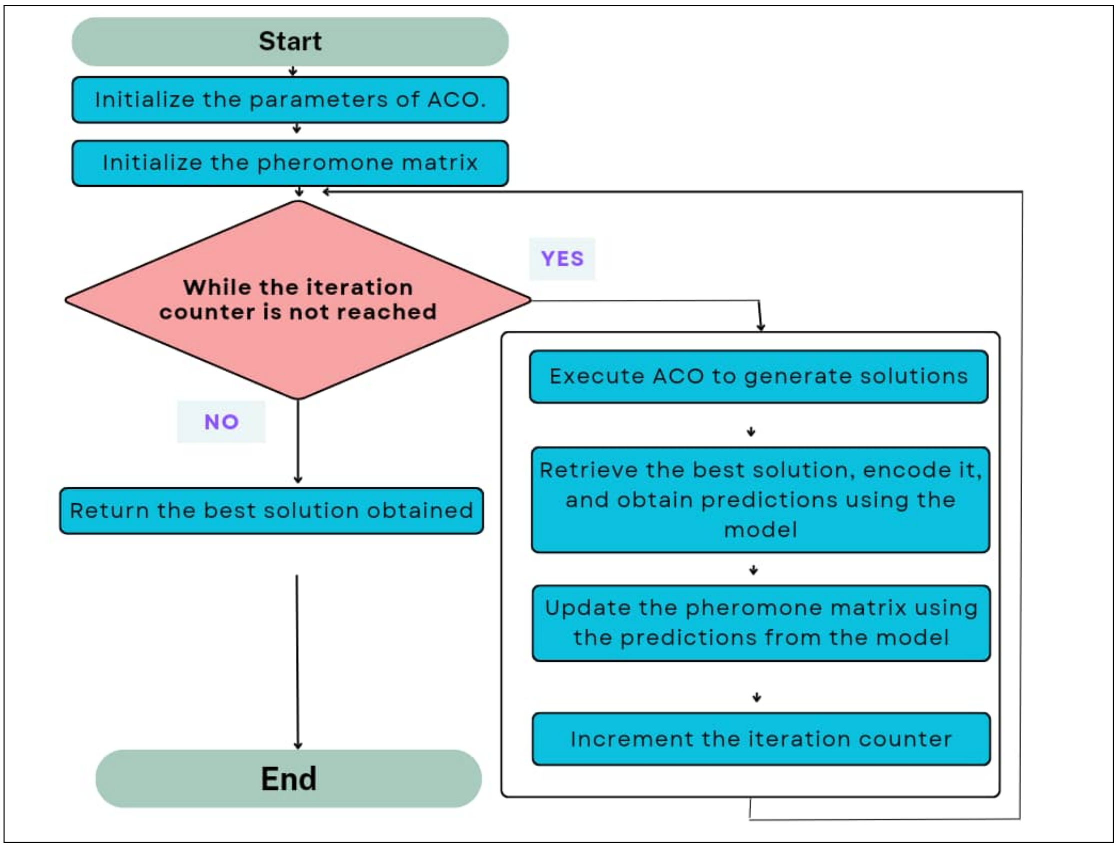
Start
Initialize the parameters of ACO.
Initialize the pheromone matrix
While the iteration counter is not reached
YES
NO
Execute ACO to generate solutions
Retrieve the best solution, encode it, and obtain predictions using the model
Update the pheromone matrix using the predictions from the model
Increment the iteration counter
Return the best solution obtained
End

Figure 3: Algorithm flowchart for ACO-RNN solution search phase

The predictions matrix predictions is an $n \times n$ matrix, where n represents the number of cities. Each element predictions$[i, j]$ indicates the probability that the path between city i and city j is included in the final solution. For example, consider the prediction results for a TSP problem with cities A, B, C, D, and E:

$$\text{predictions} = \begin{bmatrix} 0.0 & 0.8 & 0.5 & 0.1 & 0.0 \\ 0.8 & 0.0 & 0.7 & 0.2 & 0.1 \\ 0.5 & 0.7 & 0.0 & 0.6 & 0.4 \\ 0.1 & 0.2 & 0.6 & 0.0 & 0.9 \\ 0.0 & 0.1 & 0.4 & 0.9 & 0.0 \end{bmatrix} \tag{8}$$

Analyzing the first row (City A), we observe that city A has a high probability of 0.8 of being connected to city B. This suggests that once city A is visited, it is very likely that the next stop will be city B. The other connections to cities C, D, and E show lower probabilities, indicating that they are less likely to be chosen as the next steps in the route.

Using this update formula, we combine information from neural network predictions with traditional pheromones. This allows the ants to take into account both the global information provided by the neural network and the local information provided by the pheromones when making decisions about which cities to visit next.

This integration of the model's results into the pheromone update process allows us to leverage the model's prediction capabilities to guide the search towards more promising solutions and potentially improve the performance of ACO in solving the TSP. The Cf. Figure 3 illustrates the process of the final search for the solution in our approach.

This hybrid approach offers several advantages. Firstly, it exploits the modelling and generalisation capabilities of the neural network, which can lead to more accurate and informative predictions about which cities to visit next. Secondly, it maintains the distributed and parallel nature of ant colony optimisation by using traditional pheromones to guide local choices. Finally, the update formula allows the neural network predictions to be seamlessly integrated into the overall ant search process.

VI EXPERIMENTS

6.1 Implementation Environment

The experiments were conducted using the Python programming language on a specific machine with a configuration of 6 GB of RAM and an Intel® Core i5−3210M CPU @ 2.50GHz 4. The machine was running on the 64-bit Ubuntu 20.04 operating system.

For our experimentation, we utilized instances of the Traveling Salesman Problem sourced from the online TSPLIB library. This library provides a comprehensive collection of known solutions for the Traveling Salesman Problem, including various best-known solutions.

Listed below are the parameters used in our experiment, which encompass different values and configurations selected for our study:

Parameter	**Value**
learning_rate (learning rate)	0.01
num_timesteps (number of timesteps in the autoencoder)	number of cities
input_dim (input dimension of the autoencoder)	number of cities
hidden_size (size of the hidden layer of the autoencoder)	64
num_epochs (total number of training iterations)	1000
batch_size (size of mini-batches used during training)	64
number of ants	100
α **(Pheromone influence parameter in solution construction)**	1
β **(Visibility influence parameter in solution construction)**	5
τ **(Initial pheromone deposit on each graph edge)**	0.1
number of iterations	200

Table 1: ACO-RNN parameters

6.2 Results and Analysis

In this section, we present and discuss the results obtained in our study, focusing on the performance comparison between ACO-RNN and classical ACO.

To evaluate the effectiveness of our approach, we analyze the errors associated with ACO-RNN, classical ACO, and the known solutions for both small Cf. Figure 4 and medium-sized Cf. Figure 5 problem instances. We employ a formula to calculate the error rates, enabling a comprehensive comparison of these different approaches:

$$\text{Error Rate} = \left(\frac{\text{Distance of the known solution} - \text{Distance of the approach}}{\text{Distance of the known solution}}\right) \times 100 \quad (9)$$

In this formula, the distance of the approach refers to the total length of the path found by our method or the ACO, while the distance of the known solution represents the total length of the path for the best-known solution.

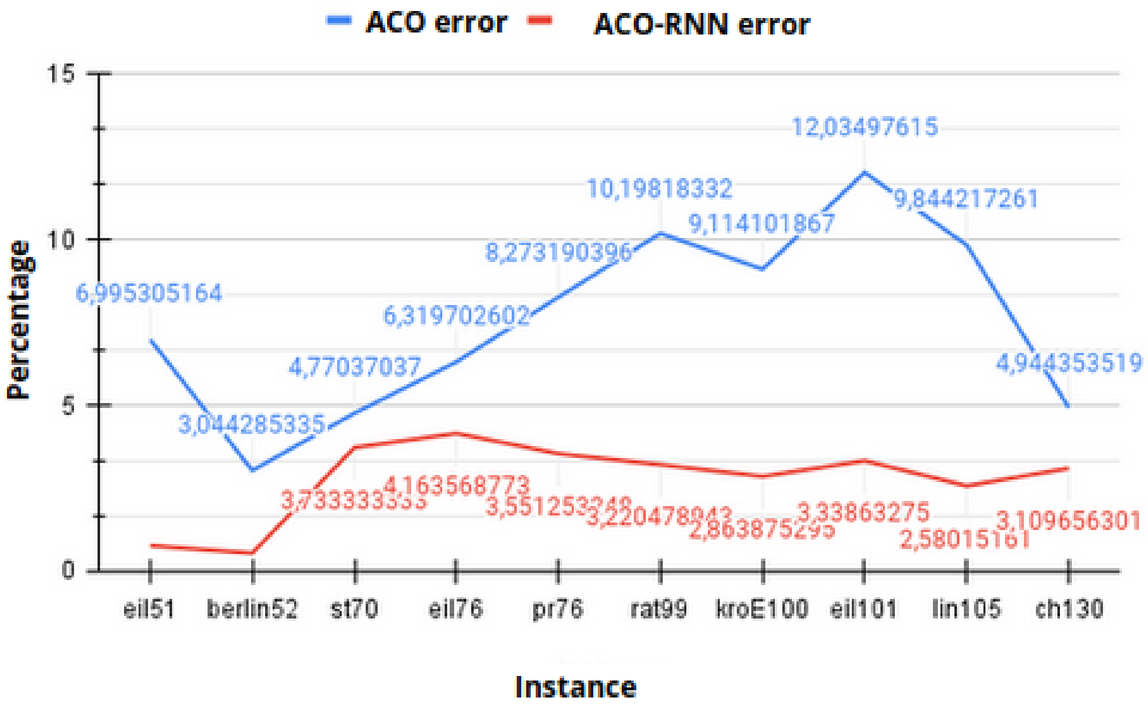

Figure 4: Comparative error curves of ACO and ACO-RNN in terms of cost on small instances

Based on the presented results in Cf. Figure 4 and Cf. Figure 5 (highlighted in red), it is evident that ACO-RNN exhibits a significant superiority over classical ACO, leading to considerably reduced errors. This notable improvement can be attributed to the enhanced exploration capabilities of ACO-RNN, which leverages the quality of predictions provided by the neural network model.

To further evaluate the performance of our approach, we have organized a comprehensive comparison between ACO-RNN and the known solutions for small and medium-sized instances. These comparisons are presented in Cf. Table 2 and Cf. Table 3, respectively. These tables provide a detailed analysis of the distances calculated by our approach and the distances associated with the known solutions.

It is important to note that at the conclusion of our comparative study, we utilized the parameters presented in the Table 1. The results of our analysis are heavily dependent on these parameters, meaning that any modifications made to them could lead to different outcomes. Therefore, it is essential to select appropriate parameters.

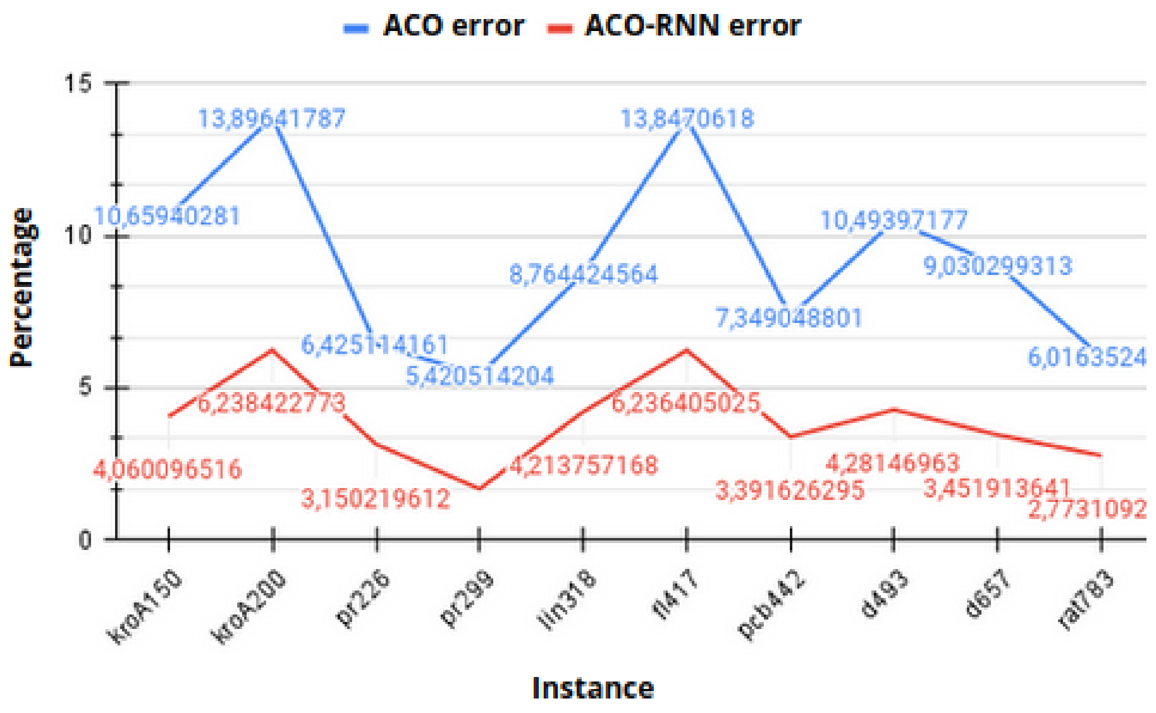

Figure 5: Comparative error curves of ACO and ACO-RNN in terms of cost on medium instances

Instance	BKS	ACO-RNN
eil51	426	429.3
berlin52	7542	7583.8
st70	675	700.2
eil76	538	560.4
pr76	108159	112000.1
rat99	1211	1334.5
rd100	7910	8036.0
kroA100	21747	21554.2
eil101	629	650.8
lin105	14379	14750

Table 2: Small Instance Comparisons

Instance	BKS	ACO-RNN
ch130	6110	6300.1
pr144	6110	57189.8
ch150	6528	6987.4
kroA200	29368	33449.1
rd400	15281	15900.5
fl417	**11861**	13503.4
pr439	107217	112286.9
pcb442	50778	53850.0
rat575	6773	7053.7
p654	34643	35982.0
u724	41910	44002.5
rat783	8806	9335.8

Table 3: Medium Instance Comparisons

VII CONCLUSION AND REFERENCES

7.1 Discussion

By examining these comparison tables (Cf. Table 2 and Cf. Table 3,), we can gain insights into the effectiveness of ACO-RNN in approximating the optimal solutions for the given TSP instances. The calculated distances serve as a quantitative measure of the quality of the solutions generated by our approach. This analysis enables us to assess the performance of ACO-RNN across a range of problem instances and provides valuable information on the overall efficiency of our proposed solution.

Through these extensive comparisons, we can ascertain the significant advancements offered by ACO-RNN in accurately approximating optimal solutions for small and medium-sized TSP instances. These findings reinforce the potential of our approach and lay the foundation for further exploration and refinement in solving larger and more complex instances of the TSP.

7.2 Conclusion

In this study, we proposed a low-level co-evolutionary hybridization approach, combining the ant colony algorithm and attention mechanism-based neural networks (ACO-RNN), to tackle the Traveling Salesman Problem (TSP). The results demonstrated the effectiveness of ACO-RNN in significantly outperforming classical ACO methods. However, it is important to acknowledge that ACO-RNN has its limitations, one of which is its inherent stochastic nature that can lead to different results with each execution.

Another limitation of ACO-RNN is its high execution time, primarily due to the training process and the predictions made during the solution search. Additionally, the performance of ACO-RNN is influenced by various parameters, which requires careful parameter tuning for each TSP instance. To address this limitation, our future work will focus on developing an adaptive parameter system that can dynamically determine the optimal parameters for each specific TSP instance.

Furthermore, we plan to explore the utilization of more advanced neural network architectures, such as transformers, to further enhance the capabilities of ACO-RNN. Additionally, we intend to implement a clustering system within ACO-RNN to reduce the computational time required to find solutions.

REFERENCES

Publications

[1] F. Glover. "Heuristics for integer programming using surrogate constraints". In: *Decision Sciences* (1977), pages 156–166.

[2] J. Deneubourg and Goss. "Probabilistic behavior in ants". In: *Journal of Theoretical Biology* 105 (1983), pages 259–271.

[3] J. Grefenstette. "Incorporating problem specific knowledge into genetic algorithms". In: *Genetic Algorithms and Simulated Annealing*. Edited by L. Davis. London: Pitman, 1987.

[4] M. Gorges-Schleuter, O. Kramer, and H. Mühlenbein. "Evolution algorithms in combinatorial optimization". In: *Parallel Computing* (1988), pages 65–88.

[5] M. Dorigo. "Optimization, learning, and natural algorithms". PhD thesis. University of Brussels, 1992.

[6] L. Gambardella and M. Dorigo. *Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem*. Palo Alto, CA: Morgan Kaufmann, 1995, pages 252–260.

[7] L. Jourdan, C. Dhaenens, E. Talbi, and S. Gallina. "A data mining approach to discover genetic and environmental factors involved in multifactorial diseases". In: *Knowledge-Based Systems* 15.4 (2002), pages 235–242.

[8] S. Suwannarongsri and D. Puangdownreong. "Solving traveling salesman problems via artificial intelligent search techniques". In: *Recent Researches in Artificial Intelligence and Database Management* 2.2 (2012), pages 1–5.

[9] H. Hachimi. “Hybridations d’algorithmes métaheuristiques en optimisation globale et leurs applications”. Spécialité: Mathématiques appliquées et Informatique. Option: Optimisation, Analyse numérique, Statistique. PhD thesis. Rabat, Maroc; Rouen, France: Université Mohammed V - Agdal, Rabat, Institut National des Sciences Appliquées de Rouen, 2013.

[10] M. Mahi, O. Baykan, and H. Kodaz. “A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem”. In: *Applied Soft Computing* 30 (2015), pages 484–490.

[11] A. Chieng Hoon Choong and N. K. Lee. “Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences using Ordinal versus one-hot Encoding Method”. In: *bioRxiv* (2017).

[12] M. DEHAN. “Distribution of medicinal products derived from blood plasma in Belgium: a case study”. PhD thesis. Catholic University of Louvain, 2018, pages 50–73.

[13] S. Gulcu, M. Mahi, O. Baykan, and H. Kodaz. “A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem”. In: *Soft Computing* 22 (2018), pages 1669–1685.

[14] X. Bresson and T. Laurent. “The transformer network for the traveling salesman problem”. In: *arXiv preprint arXiv:2103.03012* (2021).

[15] M. Jung, J. Lee, and J. Kim. “A Lightweight CNN-Transformer Model for Learning Traveling Salesman Problems”. In: *arXiv preprint arXiv:2305.01883v1* (2021).

[16] X. Gong, Z. Rong, J. Wang, et al. “A hybrid algorithm based on state-adaptive slime mold model and fractional-order ant system for the travelling salesman problem”. In: *Complex Intelligent Systems* (2022).

[17] U. Michelucci. “An introduction to autoencoders”. In: *arXiv preprint arXiv:2201.03898* (2022).

[18] Y. Pu, P. Siarry, W. Zhu, J. Wang, and N. Zhang. “Fractional-order ant colony algorithm: a fractional long term memory based cooperative learning approach”. In: *Swarm and Evolutionary Computation* 69 (2022).

VIII BIOGRAPHY

- Mathurin Soh:
  Mathurin Soh has a PhD in computer science and is an expert in combinatorial optimisation problem solving and operations research. His research focuses on the development of advanced algorithms and methods for efficiently solving complex optimisation problems. His expertise includes the use of artificial intelligence techniques, metaheuristics and mathematical programming to find optimal or near-optimal solutions. Mathurin has extensive experience of supervising research projects and has published his work in leading scientific journals.
- Anderson Nguetoum Likeufack:
  Anderson Nguetoum Likeufack is a passionate researcher in the field of artificial intelligence and operations research. Currently a PhD student, his work focuses on applying the combination of AI and operations research techniques to solve combinatorial optimisation problems. He is particularly interested in hybrid approaches that exploit the advantages of both fields to obtain efficient, high-quality solutions.