جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

كلية العلوم والتقنيات فاس
FACULTÉ DES SCIENCES ET TECHNIQUES DE FÈS

N° d'ordre : 38/2024

Centre des Etudes Doctorales : Sciences et Techniques et Sciences Médicales
Formation Doctorale : Sciences de L'Ingénieur, Sciences Physique, Mathématiques et Informatique
Discipline : Mathématiques et Informatique
Spécialité* : Mathématiques Appliquées et Informatique
Laboratoire : Ingénierie, Systèmes et Applications

Thèse pour l'obtention de DOCTORAT

Hyperparameters tuning for recurrent neural networks, application to the real problems

Préparée par : Mme Safae Rbihou

Soutenue le vendredi 12 juillet 2024 à 10h devant le jury :

Nom et Prénom	Grade	Etablissement	Qualité
Mostafa MRABTI	PES	Ecole Nationale des Sciences Appliquées de Fès	Président
EL Mokhtar EN-NAIMI	PES	Faculté des Sciences et Techniques de Tanger	Rapporteur
ABDELLAH EL FALLAHI	PH	Ecole Nationale des Sciences Appliquées de Tétouan	Rapporteur
Nabil EL AKKAD	PH	Ecole Nationale des Sciences Appliquées de Fès	Rapporteur
Lahcen OUGHDIR	PES	Ecole Nationale des Sciences Appliquées de Fès	Examinateur
Ahmed ABERQI	PH	Ecole Nationale des Sciences Appliquées de Fès	Examinateur
Nour-Eddine JOUDAR	PH	Faculté des Sciences et Techniques de Fès	Examinateur
Khalid HADDOUCH	PH	Ecole Nationale des Sciences Appliquées de Fès	Directeur de Thèse
Imad KISSAMI	PH	College of Computing UM6P de Ben Guerir	Invité

Acknowledgement

This note of thanks is special to me. For me it is not just a formal note, it is more than that. It would allow me to express my feelings to all those who have been at my side during this important project of my life, and I have never found the opportunity to do so until now.

I was lucky to have Pr. **Haddouch Khalid** as my advisor. He is a kind-hearted and highly professional person. From the very start until the end, his contribution and guidance were of great value. Whenever I needed help, he was there for assistance. His precise and pertinent comments always helped me get out of difficult situations. Without his supervision, it would not have been possible for me to write this dissertation.

With equal fervour, I want to extend a heart-felt thank to my Committee members, Pr **Mostafa MRABTI**, Pr **EL Mokhtar EN-NAIMI**, Pr. **ABDELLAH EL FALLAHI** and Pr **Nabil EL AKKAD** for their reviews and helpful comments throughout the process as well as Pr. **Lahcen OUGHDIR** for presiding the doctoral committee. Further, a special thanks to the professors: **Ahmed ABERQI**, **Nour-Eddine JOUDAR** and **Imad KISSAMI**, they have been a source of encouragement, and their remarks were of great help.

Finally, I'd like to mention my parents and my husband, who mean so much to me. My parents have sacrificed a lot for my success, and I owe them a huge debt of gratitude. One day, I'd like to be able to give something back to my parents. My sisters and brother have always been there for me in all kinds of circumstances. I'm grateful for their support. I would like to thank my husband, **Abdelmajid**, for his patience and support. He has had to take care of many things during my long working hours.

Pr. Khalid HADDOUCH

Résumé

Les réseaux neuronaux récurrents (RNN) sont devenus un choix populaire pour résoudre les problèmes d'optimisation combinatoires. Leur efficacité réside dans leur capacité à traiter habilement les séquences de données et à saisir les dépendances à long terme, ce qui les rend bien adaptés à un large éventail de tâches. Un modèle particulier de réseau neuronal récurrent qui a fait l'objet d'une attention considérable est le réseau neuronal récurrent de Hopfield. (HNN) Ce modèle, introduit en 1985, est caractérisé par des neurones interconnectés qui forment une mémoire associative conçue pour récupérer les modèles stockés. Les réseaux neuronaux récurrents se sont utilisés par la suite pour résoudre les problèmes combinatoires. Leur efficacité réside dans leur capacité à traiter habilement les données séquentielles et à saisir les dépendances à long terme, ce qui les rend applicables à un large éventail de tâches. Malgré ces avantages, le modèle a confronté des défis importants, notamment l'infaisabilité des solutions obtenues et le choix optimal de ses paramètres. Dans cette thèse, notre recherche se concentre d'abord sur les défis associés au réglage des paramètres et hyperparamètres appropriés du HNN. Notre objectif est d'obtenir des solutions optimales aux problèmes d'optimisation combinatoire pour améliorer la performance globale du HNN. Pour atteindre cet objectif, nous introduisons une nouvelle technique de réglage des paramètres de HNN, en tenant compte de sa stabilité. Ensuite, nous avons introduit une approche automatisée pour relever le défi de la sélection automatique des hyperparamètres sur la base du problème d'optimisation. Cette approche est basée sur un nouveau modèle mathématique qui minimise la fonction d'énergie. En décomposant l'ensemble des solutions réalisables, cette méthode garantit la faisabilité de la solution optimale. La deuxième partie de cette thèse est consacrée au secteur du tourisme, où nous préconisons l'application des HNNs pour améliorer d'une manière significative l'expérience des touristes. Plus précisément, le HNN s'avère être un outil précieux pour suggérer des itinéraires personnalisés aux visiteurs, en tenant compte à la fois de leurs préférences et des distances impliquées dans l'itinéraire. En outre, nous présentons un nouveau modèle de programmation mathématique conçu pour organiser efficacement les visites de plusieurs groupes de touristes en fonction de leurs préférences individuelles. Ce modèle intègre différents objectifs, notamment la distance, la durée, les contraintes budgétaires et les facteurs de risque, offrant ainsi une approche globale de la gestion et de l'optimisation de l'expérience touristique globale. Pour chaque problème étudié dans cette thèse, un certain nombre de résultats numériques sont présentés et comparés aux plusieurs approches issues de la littérature.

Mots clés: Réseau Hopfield continu, Paramétrage, Réglage des hyperparamètres, Problèmes d'optimisation combinatoire, Tourisme intelligent, Tourisme à Fès-Maroc.

Abstract

Recurrent neural networks (RNNs) have become a popular choice for solving combinatorial problems. Their effectiveness lies in their ability to skillfully handle data sequences and capture long-term dependencies, making them well suited to a wide range of tasks. One particular recurrent neural network model that has received considerable attention is the Hopfield recurrent neural network. This model, introduced in 1985, is characterized by neurons that are interconnected and form an associative memory designed to retrieve stored patterns. Recurrent neural networks have emerged as the dominant choice for solving combinatorial problems. Their effectiveness lies in their ability to skillfully handle sequential data and capture long-term dependencies, making them applicable to a wide range of tasks. Despite these advantages, the model faces significant challenges, including the infeasibility of the solutions obtained and the optimal choice of its parameters. In this thesis, our investigation initially focuses on addressing the challenges associated with tuning appropriate parameters and hyperparameters of the Hopfield neural network. Our goal is to obtain optimal solutions to combinatorial optimization problems and, ultimately, to improve the overall performance of the continuous Hopfield neural network. To achieve this goal, we introduce a novel technique for tuning the parameters of the continuous Hopfield neural network, taking into account its stability. Next, we introduced an automated approach to address the challenge of automatically selecting hyperparameters based on the optimization problem. This approach is based on a new mathematical model that minimizes the energy function. By decomposing the set of feasible solutions, this method guarantees the feasibility of the optimal solution. The second part of this thesis is dedicated to the tourism sector, where we advocate the application of Hopfield neural networks to significantly improve the tourist experience. Specifically, the Hopfield neural network proves to be a valuable tool for suggesting personalized itineraries to visitors, taking into account both their preferences and the distances involved in the itinerary. In addition, we present a novel mathematical programming model designed to effectively organize the visits of multiple groups of tourists based on their individual preferences. This model incorporates various criteria, including distance, duration, budget constraints, and risk factors, providing a comprehensive approach to managing and optimizing the overall tourist experience. For each problem studied in this thesis, a number of numerical results are presented and compared with several approaches from the literature.

Key-words: Continuous Hopfield network, Parameter settings, Hyperparameters tuning, Combinatorial optimization problems, Smart tourism, Tourism in Fez-Morocco.

Contents

Acknowledgement **2**

Résumé **3**

Abstract **4**

Acronyms **12**

1 Fundamental theorems of combinatorial optimization **18**
1.1 Introduction . . . 18
1.2 Basic concepts of combinatorial optimization . . . 19
1.3 Combinatorial Optimization Problem Classification . . . 19
1.4 Resolution methods . . . 21
1.4.1 Exact Methods . . . 22
1.4.2 Approached Methods . . . 23
1.4.3 Stochastic Optimization . . . 23
1.4.4 Quantum Algorithms . . . 23
1.4.5 Algorithms based on machine learning . . . 23
1.4.6 Hybrid algorithms . . . 24
1.5 Famous combinatorial optimisation problems . . . 24
1.5.1 Weighted constraint satisfaction problem . . . 24
1.5.2 Task assignment problem . . . 25
1.5.3 Traveling salesman problem . . . 26
1.5.4 Max-Stable Problem . . . 26
1.5.5 Graph Coloring Problem . . . 27
1.5.6 Job Shop Scheduling Problem . . . 28
1.6 Materials tools and data . . . 29
1.7 Conclusion . . . 32

2 Exploring Hopfield Recurrent Neural Networks for Combinatorial Optimization: Methods and Dynamics **33**
2.1 Introduction . . . 33
2.2 General aspects for Hopfield recurrent neural networks . . . 34
2.2.1 Classifications of Hopfield neural networks . . . 34
2.2.2 Stability analysis of CHN . . . 37
2.2.3 CHN equilibrium points algorithm . . . 38
2.3 CHN for constrained optimization problems . . . 39
2.3.1 Parameter setting . . . 40
2.3.2 Initial state . . . 41

2.4 Comparative study between a Hopfield neural network, approach metaheuristic and exact method . . . 41
2.4.1 Comparative study . . . 41
2.4.2 Comparison and discussion . . . 43
2.5 Using Crank-Nicolson scheme for continuous Hopfield network equilibrium . . . 45
2.5.1 The proposed algorithm . . . 45
2.5.2 Using Crank Nicolson scheme for solving CHN applied to task assignment problems . . . 45
2.6 Conclusion . . . 48

3 Tuning and adaptation of HNN to combinatorial optimization problems 50
3.1 Introduction . . . 50
3.2 Hopfield energy function associated with constrained optimization problems . . . 51
3.2.1 HEF for TSP . . . 51
3.2.2 HEF for TAP . . . 51
3.2.3 HEF for WCSP . . . 52
3.2.4 HEF for MSP . . . 52
3.2.5 HEF for GCP . . . 53
3.2.6 HEF for JSSP . . . 53
3.2.7 HEF for MSP . . . 54
3.2.8 HEF for KNP . . . 54
3.2.9 HEF for PSP . . . 55
3.2.10 HEF for SPP . . . 55
3.3 New method for CHN parameters tuning . . . 56
3.3.1 Parameter tuning in CHN applied to COPs . . . 56
3.3.2 Experimental results and discussion . . . 59
3.4 Conclusion . . . 66

4 Hyperparameters optimization of Hopfield neural networks 67
4.1 Introduction . . . 67
4.2 Overview of existing research . . . 68
4.3 Optimizing hyperparameters in Hopfield neural networks using evolutionary search 69
4.3.1 Generic mathematical model . . . 70
4.3.2 Application to the COPs . . . 71
4.3.3 Proposed algorithm . . . 73
4.3.4 Experimentation and discussion . . . 79
4.4 Tuning HNN with metaheuristic hyperparameter selection . . . 87
4.4.1 Experimentation and discussion . . . 87
4.5 Conclusion . . . 91

5 Optimizing Tourist Circuits: A Dual Approach Using Hopfield Neural Networks and Innovative Routing Models 92
5.1 Introduction . . . 92
5.2 Optimization of tourist routes using a Hopfield neural network . . . 93
5.2.1 Continuous Hopfield network to optimize tourist routes . . . 93
5.2.2 Proposed approach . . . 94
5.2.3 Experimentation . . . 95
5.3 A new optimization model of the visitor routing problem applied to smart tourism 97
5.3.1 Proposed mathematical model . . . 98
5.3.2 Marginalization of proposed VRP . . . 102

5.3.3 Simulated annealing algorithm 103
5.3.4 Optimizing tourist visits in Fez using the proposed VRP 104
5.3.5 Results and discussion 106
5.4 Conclusion 108

General conclusion and Future works **110**

List of Figures

1.1 Combinatorial optimization problem building blocks 20
1.2 Approaches to solving combinatorial optimization problems 22
1.3 The location of Some POIs in Fez 29
1.4 Details of the tourist POIs 31

2.1 The discrete Hopfield network architecture 35
2.2 The continuous Hopfield network architecture 36
2.3 CHN equilibrium points algorithm [16] 39
2.4 Minimum and Mean that CHN, ACA and Cplex gives an optimal value for the different instances 43
2.5 Sum time and iteration that CHN, ACA and Cplex gives an optimal value for the different instances 44
2.6 The variation of metrics for two methods Euler and CN 48

3.1 Variation of the hyperparameter α as a function of NBS, SBV of FS and SIN . 63
3.2 Variation of the hyperparameter ε as a function of NBS, SBV of FS and SIN . 65
3.3 Variation of the starting point as a function of NBS, SBV of FS and SIN 65

4.1 GA procedures for selecting optimal hyperparameters 74
4.2 PSO procedures for selecting optimal hyper-parameters 77
4.3 Accuracy of handled problems using different methods of hyperparameter selection 81
4.4 Percentage of optimal solutions for problems studied using different methods of hyperparameter selection 82
4.5 Total time of handled problems using different methods of hyperparameter selection 82
4.6 Number of iterations for problems studied using different methods of hyperparameter selection 83
4.7 Convergence of feasible solutions using a random selection and the proposed method for some instances of TAP, TSP and MSP 85
4.8 Convergence of execution time using a random selection and the proposed method for some instances of TAP, TSP and MSP 86
4.9 Optimal solution of GCP using the three approaches 90
4.10 Sum of Iterations of GCP using the three approaches 90
4.11 Total Time of GCP using the three approaches 90

5.1 Steps of the proposed approach 94
5.2 Accuracy of treated scenarios using different approaches 96
5.3 Execution time in (ms) of scenarios treated using different approaches 96
5.4 Representative graph of 4 POIs 105
5.5 Comparison of each criteria between our model and the other models 108
5.6 Comparison by average of all criteria between our model and the other models 109

List of Tables

1.1 WCSP dataset . . . 30
1.2 TAP dataset . . . 30
1.3 MSP dataset . . . 30
1.4 GCP dataset . . . 30
1.5 TSP dataset . . . 31
1.6 Travel distance between 20 POIs in Km . . . 31
1.7 Travel time between 20 POIs in hours . . . 32

2.1 Computational experiments for some TSPLIB examples by the continuous Hopfield network . . . 42
2.2 Computational experiments for some TSPLIB examples by the Ant colony . . . 42
2.3 Computational experiments for some TSPLIB examples by the exact resolution 43
2.4 CHN, ACA and CPLEX rankings for the four properties . . . 44
2.5 The result of each TAP by the CHN-Euler . . . 46
2.6 The result of each TAP by the CHN-Crank-Nicolson . . . 47

3.1 Performance evaluation of our parameter settings on 20 WCSP instances . . . 60
3.2 Performance evaluation of our parameter settings on 20 TAP instance . . . 61
3.3 Performance evaluation of our parameter settings on 11 TSP instance . . . 61
3.4 Performance of our parameter settings approach and those proposed in [10] for WCSP . . . 61
3.5 Performance of our parameter settings approach and those proposed in [12] for TAP . . . 61
3.6 Performance of our parameter settings approach and those proposed in [29] for TSP . . . 62
3.7 Obtained results using different combinations of hyperparameters for WCSP . . . 63
3.8 Obtained results using different combinations of hyperparameters for TAP . . . 64
3.9 Obtained results using different combinations of hyperparameters for TSP . . . 64

4.1 An overview of common HNN applications, their parameters and appropriate optimization techniques . . . 69
4.2 GA parameters . . . 79
4.3 PSO parameters . . . 79
4.4 Summary of TSP simulation experiments . . . 80
4.5 Summary of TAP simulation experiments . . . 80
4.6 Summary of MSP simulation experiments . . . 81
4.7 The percentage of relative quality comparison between the methods used to select HNN hyperparameters . . . 86
4.8 Obtained result of GCP using HP_GA . . . 88
4.9 Obtained result of GCP using HP_AC . . . 89

4.10 Obtained result of GCP using HP_PSO 89

5.1 Test POI 95
5.2 Performance metrics for optimal routing in the medina of Fez 96
5.3 The paths obtained by each model 105
5.4 Solution for each model 106
5.5 Scenarios used in our study 106
5.6 Obtained results using our model and the other models 107

List of Algorithms

1 The algorithm to obtain the equilibrium point of CHN by the Crank-Nicolson method 46
2 Pseudocode for Hyperparameter Optimization using GA 74
3 POPULATION, CHN_FITNESS, SELECTION, CROSS and MUTATE function 76
4 Pseudocode for Hyperparameter Optimization using PSO 78
5 SA algorithm to generate a VRP 104

Acronyms

RNN : **R**ecurrent **N**eural **N**etwork
ANN : **A**rtificial **N**eural **N**etwork
HHN : **H**opfield **N**eural **N**etwork
DHN : **D**iscrete **H**opfield **N**eural Network
CHN : **C**ontinuous **H**opfield **N**eural Network
GA :**G**enetic **A**lgorithm
ACO :**A**nt **C**olony **O** ptimization
PSO :**P**article **S**warm **O**ptimization
CN :**C**rank **N**icolson
COP :**C**ombinatorial **O**ptimization **P**roblem
HEF :**H**opfield **E**nergy **F**unction
TAP :**T**ask **A**ssignment **P**roblem
TSP :**T**raveling **S**alesman **P**roblem
MSP :**M**ax **S**table **P**roblem
GCP :**G**raph **C**oloring **P**roblem
WCSP :**W**eighted **C**onstraint **S**atisfaction **P**roblem
JSSP :**J**ob **S**hop **S**cheduling **P**roblem
MSP :**M**eeting **S**cheduling **P**roblem
KNP :**K**ohonen **N**etwork **P**roblem
SPP :**S**hortest **P**ath **P**roblem
POI :**P**oints **O**f **I**nterest
ST :**S**mart **T**ourism
SA :**S**imulated **A**nnealing
VRP :**V**isitor **R**outing **P**roblem
tanh :**H**yperbolic **T**angent **F**unction

Author's Publications

The peer-reviewed journal articles, conference proceedings, and papers submitted for publication written by the authors during their PhD studies are listed below.

International Journal

1. S. Rbihou, N. Joudar, and K. Haddouch. Parameter tuning of continuous Hopfield network applied to combinatorial optimization. Annals of Mathematics and Artificial Intelligence (2023): 1-19.

2. Rbihou, S., Haddouch, K. El moutaouakil, K. Optimizing hyperparameters in Hopfield neural networks using evolutionary sea.rch. OPSEARCH (2024).

3. S. Rbihou, N. Joudar, and K. Haddouch. A new optimization model of the visitor routing problem applied to smart tourism. Int. J. of Operational Research (2023).

4. S. Rbihou, N. Joudar, and K. Haddouch. Optimizing Parameter Settings for Hopfield Neural Networks Using Reinforcement Learning (Submitted).

International Conferences

1. S. Rbihou, and K. Haddouch. Comparative study between a neural network, approach metaheuristic and exact method for solving Traveling salesman Problem. 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS). IEEE, 2021.

2. S. Rbihou et al. Using Crank-Nicolson Scheme for Continuous Hopfield Network Equilibrium. International Conference on Artificial Intelligence and Industrial Applications. Cham: Springer Nature Switzerland, 2023.

3. S. Rbihou, H.Senba, and K. Haddouch. An Approach to the Optimization of Tourist Circuits Using a Hopfield Neural Network: Application in the Old City of Fez. International Conference on Big Data and Internet of Things.

4. S. Rbihou, and K. Haddouch. Tuning Hopfield Neural Networks with Metaheuristic Hyperparameter Selection. The 7th International Conference on Networking, Intelligent Systems and Security

General Introduction

Recurrent neural networks (RNNs) are a class of neural models designed to excel at processing sequential or temporal data. Unlike conventional neural networks, RNNs incorporate recurrent connections, enabling them to maintain a form of internal memory. This unique feature allows RNNs to consider previous contextual information when processing new data, enabling more nuanced and context-aware analysis [138]. RNNs have attracted considerable attention and demonstrated their effectiveness in a wide range of tasks, including machine translation, text generation, sentiment analysis, speech recognition, and time series prediction. Their ability to capture sequential dependencies and temporal patterns makes them particularly well-suited to tasks where understanding context over time is critical. In machine translation, RNNs can effectively translate sentences by considering the full context of the input sequence before generating the output sequence. Similarly, in text generation, RNNs can produce coherent and contextually relevant text by using the context of previously generated words or characters [139, 140].

Hopfield neural networks (HNNs) are some of the most important recurrent neural networks models to date. Unlike traditional RNNs, HNNs are primarily concerned with information storage and retrieval. HNNs are widely used in artificial intelligence, exploiting principles from the biological concepts of the human brain. They excel at optimizing processes, recognizing patterns, and storing data efficiently [131, 19, 62]. The HNN operates on a basic model in which neurons form recurrent connections, similar to a network in which each neuron is intertwined with others like steps in a ladder. Within this architecture, neurons exist in binary states: they are either active (represented as 1) or inactive (represented as 0). The network's output emerges from the collective states of its neurons through iterative processes. The essence of this network lies in its dynamic interplay: neurons communicate and influence each other through these recurrent connections, mirroring the intricate dance of neural activity observed in biological systems. As iterations progress, the network evolves, reflecting the changing dynamics of its constituent neurons. The HNN iterative nature allows it to store and recall patterns, which is a key aspect of its computational capability. The Hopfield network is a testament to the power of recurrent connectivity, reflecting the nuanced interactions found within the complex fabric of the brain. Through its iterative operations, it offers insights into pattern recognition, memory retrieval, and associative learning, enriching our understanding of both artificial and biological neural networks [21, 5].

The HNN is a dynamic system of interconnected neurons, where each neuron has the capacity to exist in either an active or inactive state. The behaviour of the network is governed by a synchronous or asynchronous update rule, which determines how neurons adjust their states in response to incoming signals from other neurons [131]. The Hopfield neural network has associative memory properties, i.e., it can retrieve memory states from partial activation patterns. This ability is based on the minimization of a Lyapunov function that guarantees convergence to stable states. The dynamical system represented by the HNN can be interpreted as an energy landscape, where each state corresponds to a local minimum of energy. The at-

tractors in this landscape represent the stable states of the system to which it converges over time. This interpretation provides a powerful framework for understanding the behavior and computational capabilities of the Hopfield network in solving pattern recognition and classification problems [21, 16]. Based on output functions, HNNs can be divided into two common forms: discrete-time and continuous-time models.. In its discrete form, the HNN operates with binary states, usually denoted 1 and 0, representing the activation levels of the neurons. The connections between neurons are also binary weights. This type of network is used to solve problems of pattern recognition and associative memory storage. The states of the neurons evolve discretely over time, determined by synchronous or asynchronous update rules. In its continuous form, the HNN uses continuous states and connection weights. Neuron activations are real values, often modeled by sigmoid or Gaussian functions. Connections between neurons are real weights that can vary over a continuous range. Neuron activations are continuously updated over time, with differential equations describing the evolution of the system. This type of network is often used in analog computing and dynamic modeling applications [21, 16, 29].

Hopfield neural networks have been studied in various optimization contexts [103, 5, 21, 123, 116, 124, 125, 99, 31]. Their first application to the classical Traveling Salesman Problem (TSP) marked the beginning of their use [131]. Since then, HNNs have been extended to several other areas of optimization, including linear programming (LP), nonlinear programming (NLP), mixed integer linear programming (MILP), and multi-objective optimization problems [21]. The use of HNNs to solve various optimization problems is based on the management of a dynamic system characterized by an energy function. This energy function results from the combination of the objective function of the problem to be optimized and the constraints associated with it [29]. Three common approaches are used to solve these problems. Firstly, penalty functions are used to integrate the constraints into the objective function, allowing them to be taken into account during the optimization process. Secondly, Lagrange functions are used to formulate the optimization problem with Lagrange multipliers, making it easier to find optimal solutions. Finally, primal and dual techniques are used to explore the optimization problem from two complementary perspectives, allowing for in-depth analysis of potential solutions [21].

The ability of HNN to model nonlinear dynamical systems makes them suitable for solving complex problems. HNN provide an effective approach to solving combinatorial optimization problems by exploiting their ability to converge to stable states that represent optimal or suboptimal solutions [21, 103, 5, 123, 116, 124, 125, 99]. In this way, their recurrent architecture allows the interactions between the different variables in the problem to be taken into account, which is essential for many optimization problems where decisions made for one variable affect the others. In addition, Hopfield neural networks offer a parallel approach to problem solving, allowing a large space of potential solutions to be explored quickly. These factors make HNN widely used in combinatorial optimization problems [21, 5, 6, 9, 29].

Although HNN is widely used for optimization, it has some significant drawbacks. Firstly, the minimum solution obtained by HNN does not guarantee global optimization, meaning that the model may converge to local minima rather than the best possible outcome. In addition, the selection of the network's hyperparameters is not straightforward, making model optimization more complex and prone to error. Another major challenge with HNNs is their energy function. This function can contain many unnecessary local solutions, making it difficult to avoid these unwanted local minima [21, 29]. These limitations have motivated our research aimed at improving the performance of HNNs. In this thesis, we have attempted to address these drawbacks and propose potential solutions. We have explored various approaches to improve convergence to global optima, simplify hyperparameter tuning, and make the energy function more conducive to identifying relevant solutions. Our work makes a significant contribution to understanding the limitations of HNNs and opens the way for future studies in this area. The

contributions are organized in the following layout:

Chapter 1- Basic theories of combinatorial optimization: In this chapter, we delve into the field of combinatorial optimization problems, which form the backbone of many computational and real-world challenges. First, we lay the groundwork by elucidating the fundamental concepts of combinatorial optimization. These concepts are the foundation upon which we build our understanding, covering essential elements such as problem formulations, algorithmic approaches, and optimization criteria. In addition, we begin a detailed exploration of selected fundamental combinatorial optimization problems. By dissecting these problems, we aim to unravel their intricacies, examine their underlying structures, and elucidate the methods used to solve them. Through this comprehensive analysis, we aim to provide the reader with the knowledge and insight necessary to navigate the diverse landscape of combinatorial optimization with confidence and skill.

Chapter 2- Exploring Hopfield Recurrent Neural Networks for Combinatorial Optimization: Methods and Dynamics: This chapter outlines the basic concepts of Hopfield neural networks for solving optimization problems, providing an overview of their dynamics and methods for obtaining optimal solutions. Through empirical comparisons and methodological advances, Hopfield neural networks are emerging as powerful tools for solving complex optimization problems. In this study, we explore the treatment of the differential equations that define the behavior of Hopfield neurons using the Crank-Nicolson method, which provides a numerically robust approach to modeling and analyzing the complex dynamics of neural networks.

Chapter 3- Tuning and adaptation of HNN to combinatorial optimization problems: In this chapter, we present and study a parameter tuning methodology for continuous Hopfield neural networks. Our approach focuses on stability analysis and aims to provide an efficient solution to combinatorial optimization problems. In addition, we study the influence of Hopfield's hyperparameters on its convergence. The experimental results presented in this chapter highlight the critical importance of the proposed approach to CHN parameter tuning and the selection of optimal hyperparameter combinations. By closely examining the performance obtained through a series of rigorous tests, we have found that our CHN parameter tuning method offers significant improvements over traditional approaches.

Chapter 4- Hyperparameters optimization of Hopfield neural networks: The use of Hopfield neural networks faces a major problem: the automatic choice of hyperparameters depending on the optimization problem. This chapter presents an automatic method to overcome this difficulty, based on an original mathematical model that minimizes the energy function. These methods guarantee an optimal choice of hyperparameters from the Hopfield network. To demonstrate the effectiveness of our model, we use particle swarms and genetic algorithms for the proposed model. The numerical results presented in this chapter highlight the remarkable performance of our proposed approach compared to other hyperparameter selection methods for CHNs. This observation demonstrates the effectiveness and robustness of our method in the specific context of hyperparameter optimisation for CHNs.

Chapter 5- Optimizing Tourist Circuits: In this chapter, we presents an improved mathematical model for visit planning adapted to groups of visitors. Unlike previous distance-based approaches, our model integrates four key objective functions: distance from the route, duration of the visit, budget, and risk factors. Each visitor provides personal information, points of interest, and constraints that contribute to the development of a complete mathematical program. The experiments demonstrate that the proposed model offers numerous advantages for the development of the tourism sector.

Conclusions and Future work : This section provides the concluding remarks with more emphasis on achievements and limitations of the proposed schemes. The scopes for further research are outlined at the end. The contributions made in each chapter are discussed in sequel, which include proposed models, their simulation results, and the comparative analysis.

Chapter 1

Fundamental theorems of combinatorial optimization

1.1 Introduction

Combinatorial optimization, a branch of mathematical optimization, focuses on finding the best solution to a given problem from a finite set of possibilities. Its application extends to areas as diverse as communications, transportation, manufacturing and has led to extensive research in industrial engineering, computer science, and operations research. The aim of optimization is to find the maximum or minimum of an objective function while respecting certain constraints. To solve these combinatorial optimization problems (COPs), two categories of methods predominate: exact methods and approximate methods.

The goal of exact methods in combinatorial optimization is to guarantee that the optimal solution is found by exhaustively exploring the entire search space, either explicitly or implicitly. This approach has the undeniable advantage of providing an optimal solution, thus ensuring the quality of the answer. However, one of the main challenges associated with exact methods is the possibility of combinatorial explosion, a phenomenon where the number of possible combinations grows exponentially with the size of the problem. This rapid growth makes the computational time required to exhaustively explore the search space prohibitively large. Consequently, although these methods are capable of guaranteeing an optimal solution, their practicality is often limited, especially when faced with large-scale problems.

In many real-world applications, finding the optimal solution is a complex task, mainly because of the inherent dynamics of the system under study, the constraints imposed, and the large number of variables involved. Faced with these considerable challenges, most specialists opt for the use of approximate search techniques. These methods are proving to be an effective alternative, as they make it possible to obtain solutions that, while not guaranteeing absolute optimality, are close to it, all within a reasonable computational time. Commonly used approximation techniques include heuristic algorithms, metaheuristic algorithms, and stochastic optimization methods. These approaches are characterized by their ability to explore the solution space intelligently, avoiding the pitfalls associated with the combinatorial complexity of the problems.

The rest of this chapter is organized as follows: Section 1.2 presents the fundamental concepts of combinatorial optimization. In that section, a brief history is given of the types of combinatorial optimisation problems. Section 1.5 describes some basic combinatorial optimisation problems. Finally, a brief conclusion is given in section 1.7.

1.2 Basic concepts of combinatorial optimization

The combinatorial optimization problem (COP) is a class of optimization problems (OP) solved by optimization methods that aim to find an optimal solution from a large set of possible solutions to satisfy some objective criteria [1].

Definition 1. *A combinatorial optimization problem can be formulated as follows:*

$$P = (S, f, \Omega) \tag{1.1}$$

Where:

- P represents the combinatorial optimization problem;
- $S \subset \{0,1\}^n$ is the search space of the problem domain;
- $f : S \to R$ represents the objective function;
- Ω is the set of problem constraints.

The search space S is defined by a set of decision variables denoted $x = \{x_1, x_2, ..., x_n\}, x \in S \subset \{0,1\}$, representing the data searched for a solution. This space can be restricted by a set of constraints Ω, which serve as limits defining the feasibility of solutions. These constraints establish criteria for determining the feasibility of a solution in the context of the problem being studied. At the same time, the objective function f intervenes in the evaluation of the solutions by assigning a value to each of them. This function guides the search by directing the optimization process towards the best quality solutions within the space delimited by the variables and constraint [1, 2].

Definition 2. *The mathematical modeling of COPs can be formulated as follows:*

$$(P) \begin{cases} Min \ \ (Max) \quad f(x_1, x_2, ..., x_n) \\ s.c \\ \quad x \in \Omega \\ \quad x \in S \subset \{0,1\}^n \end{cases} \tag{1.2}$$

In the domain of possible solutions, we search for a particular value x^* that satisfies the constraints of the problem and minimizes (maximizes) the value of the associated objective function.

Definition 3. *$x^* \in S$ is a global minimum (global maximum) of the function f on the domain S if: $f(x^*) \leq f(x)$ $(f(x^*) \geq f(x))$, $\forall x \in S$*

Definition 4. *$x^* \in S$ is a local minimum (local maximum) of the function f if: $f(x^*) \leq f(x)$ $(f(x^*) \geq f(x))$, $\forall x \in S \cap B_\epsilon(x^*)$. Where,*

This means that there is a neighbourhood of x^* in which no other solution improves the objective function.

1.3 Combinatorial Optimization Problem Classification

The classification system for combinatorial optimization problems is crucial for organizing and understanding the many challenges in this complex field. The diversity of combinatorial optimization problems can be better understood by categorizing them according to different criteria.

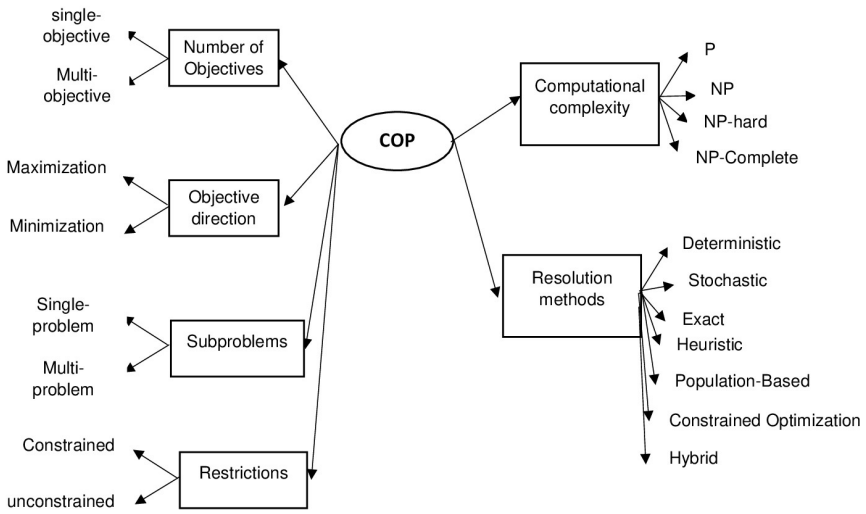


Figure 1.1: Combinatorial optimization problem building blocks

This approach allows researchers and practitioners to gain an in-depth understanding of the specific nature of each problem, facilitating the development of specific approaches and the judicious selection of relevant algorithms [4].

Figure 1.1 illustrates each classification dimension in detail. These dimensions are explored in more detail in the following subsections.

1. **Number of Objectives**

   Combinatorial optimization problems are classified according to the number of objectives to be optimized, with a distinction being made between single-objective and multi-objective problems.

   In single-objective problems, the objective is to optimize a single function. This simplification allows for clearer modeling and often easier solutions, as there is only one metric to improve. The inherent simplicity of these problems makes them easier to handle both conceptually and computationally.

   Multiobjective problems, on the other hand, are more complex. In this category, several competing objective functions need to be optimized, often with trade-offs between them. Solving such problems involves juggling the various metrics, as improving one objective may lead to a deterioration in another. This leads to a complex trade-off between the different dimensions to be optimized.

2. **Direction of a combinatorial optimization problem**

   The direction of a combinatorial optimization problem is another class of combinatorial optimization. The direction determines whether the objective function is maximized or minimized.

   Definition 5. *When the objective of a combinatorial optimization problem is defined in terms of maximization, the main objective is to obtain the best possible value for the objective function f.* $f(x^*) \geq f(x), \forall x \in S$

   Definition 6. *When the objective is defined in terms of minimization, the main objective is to reduce the value of the objective function as much as possible.* $f(x^*) \leq f(x), \forall x \in S$

3. **Subproblems**

   Optimization problems can be divided into two main categories based on their complexity: single optimization problems and multiple optimization problems, depending on the number of sub-problems.

   Single optimization problems consist of a single problem. These problems usually have one objective function.

   On the other hand, multiple optimization problems have several competing objectives that need to be optimized simultaneously. These problems require additional effort to select the optimal solutions (Pareto optimal) and also additional effort to balance the results of the sub-problems to deal with the overall perspective of the higher level problem.

4. **Restrictions**

   Optimization problems can be divided into two main categories depending on the presence or absence of constraints in the problems considered: constrained optimization problems and unconstrained optimization problems.

   Constrained optimization problems involve finding the best possible solution while respecting a set of specified conditions or constraints. These constraints can take various forms, such as equations or inequalities, and define the limits or requirements that the optimal solution must satisfy. The presence of constraints adds a complex dimension to the problem, as not only must the objective function be optimized, but it must be done while respecting the imposed conditions.

   In contrast, unconstrained optimization problems focus on finding the best possible solution without imposing additional conditions. The goal is to optimize an objective function without worrying about any particular constraints. These problems are often simpler to formulate and solve because they do not require additional constraints to be considered.

5. **Algorithmic complexity**

   The algorithmic complexity of combinatorial optimization problems is a crucial aspect in characterizing the difficulty of these computational challenges. These problems can be categorized according to the complexity of their solutions, providing valuable insights into the choice of appropriate algorithmic approaches. P-problems are those for which there is an efficient algorithm capable of providing a solution in polynomial time. This means that although the degree of difficulty may vary, the algorithmic complexity remains manageable to some extent. NP problems are those for which no solution is known in polynomial time, but if a solution is given, it can be verified in polynomial time. Most combinatorial optimization problems belong to this class. NP-complete problems are a subclass of NP problems with the interesting property that an algorithm that solves one of them in polynomial time can solve all NP problems in polynomial time. NP-hard problems are at least as hard as NP-complete problems, although not necessarily in NP. They present significant algorithmic challenges, and their optimal solution is often unattainable for large instances.

1.4 Resolution methods

In solving combinatorial optimization problems, various methods are used to find effective solutions to the complexity inherent in these challenges. Among the most commonly used approaches are exact algorithms. These guarantee the discovery of the optimal solution but are often limited in terms of scalability. In contrast, approximate algorithms offer a pragmatic

alternative by providing near-optimal solutions. These methods, such as local search or linear programming, are particularly well suited to complex problems where exhaustive search would be impractical. Another important category is metaheuristics, which includes approaches such as genetic algorithms, simulated annealing, and particle swarms. These methods are particularly effective at intelligently exploring the solution space, often overcoming the limitations of exact algorithms. Finally, decomposition and relaxation techniques are also widely used to solve combinatorial problems by breaking them down into more manageable sub-problems, allowing for iterative solving. The appropriate use of these different methods often depends on the specific nature of the problem and the time constraints involved.

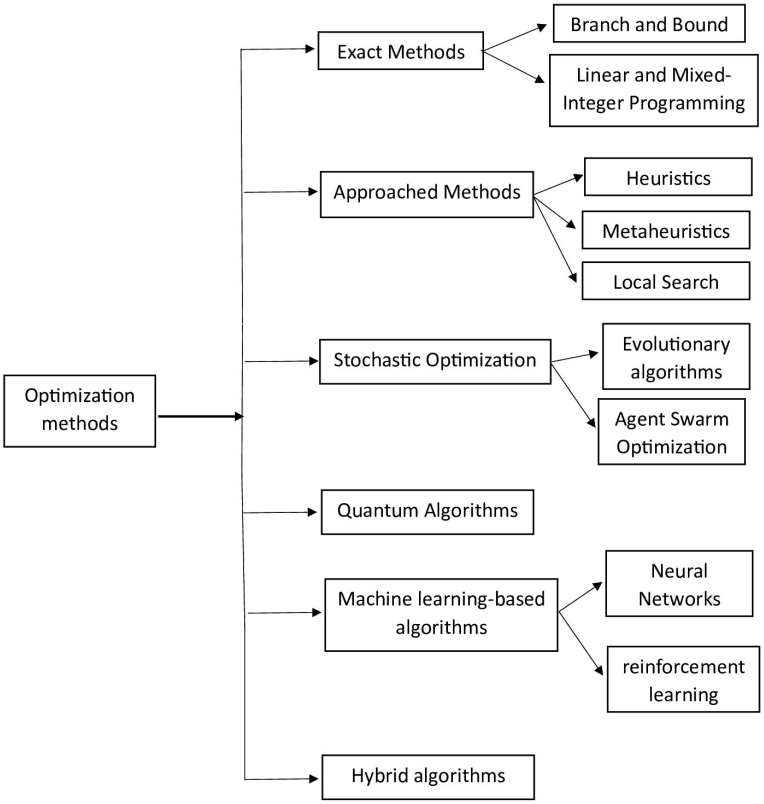


Figure 1.2: Approaches to solving combinatorial optimization problems

Figure 1.2 illustrates a brief description of the different approaches to solving combinatorial optimization problems. There are many optimization methods, generally they can be classified [3].

1.4.1 Exact Methods

Exact methods in combinatorial optimization are algorithmic approaches that aim to find the optimal solution to a given problem without compromising accuracy. These methods guarantee an exhaustive search of all possible solutions in the problem search space. The two main categories of exact methods are:

- **Branch and bound:** recursively decompose the problem into smaller subproblems, eliminating unpromising branches.
- **Linear and mixed-integer programming:** formulates the problem as a linear or mixed-integer program that is solved using algorithms such as the simplex or cutting plane method.

1.4.2 Approached Methods

Approximate methods, in the context of combinatorial optimization, are problem-solving approaches that aim to provide solutions of acceptable quality in a reasonable amount of time. Unlike exact methods, which guarantee the optimal solution, approximate methods are often used to deal with complex problems where the exhaustive search for the optimal solution is impractical due to the exponential size of the search space. These methods can be divided into three types:

- **Heuristic algorithms**: are techniques for constructing initial solutions that are often fast but not guaranteed to be optimal.
- **Metaheuristic algorithms:** generic frameworks for more robust exploration of the search space, such as genetic algorithms, simulated annealing, and particle swarms.
- **Local exploration:** iterative optimization by exploring the bounds of an initial solution.

1.4.3 Stochastic Optimization

Stochastic optimization is an approach to solving optimization problems that takes into account the uncertainty or variability inherent in certain aspects of the problem. Unlike deterministic methods, which assume fixed data and constant parameters, stochastic optimization introduces random elements into the optimization process. These approaches are divided into:

- **Evolutionary algorithms:** Simulate natural evolution to generate a population of potential solutions.
- **Agent Swarm Optimization:** Models the behavior of natural swarms to guide the search.

1.4.4 Quantum Algorithms

Quantum algorithms for solving combinatorial optimization problems are algorithms that exploit the principles of quantum mechanics to efficiently explore the solution space of combinatorial optimization problems. These algorithms exploit the unique properties of quantum bits, called qubits, which can exist in superpositions of states and be quantum-correlated.

1.4.5 Algorithms based on machine learning

Machine learning-based algorithms for solving combinatorial optimization problems are approaches that use machine learning techniques to find efficient solutions to difficult and often discrete optimization problems. These problems involve finding the best combination from a finite set of possible solutions, making them complex to solve. There are two possible approaches to these algorithms:

- **Neural Networks:** Models the optimization problem as a neural network to learn and predict solutions.
- **Reinforcement Learning Algorithms:** Learn to make sequential decisions to solve combinatorial problems.

1.4.6 Hybrid algorithms

Hybrid algorithms for solving combinatorial optimization problems are approaches that combine several different algorithmic techniques or methods to exploit their respective advantages. The basic idea behind hybrid algorithms is to improve solution performance by exploiting the complementary strengths of different approaches.

1.5 Famous combinatorial optimisation problems

In this section, we present an in-depth exploration of some current the most relevant and exciting combinatorial optimization problems. These stimulating mathematical challenges can be found in a wide variety of fields, from logistics and planning to engineering and computer science.

1.5.1 Weighted constraint satisfaction problem

A constraint satisfaction problem (CSP) refers to the problem of finding values for a set of variables subject to constraints. To solve this problem, you must assign values to variables that satisfy the most of the constraints set. In the case where a privilege tuple is relative to others, a CSP becomes a WCSP [112]. In such problem, each value in domains of variables and each tuple in the constraint has an associated weight [89, 101]. If a solution contains some values and tuples, then the weight associated with this affectation is incurred. More precisely, a weighted constraint satisfaction problem isdefined by a quadruplet sets $(Y, D, C, S(k))$ where:

- $Y = \{y_1, ..., y_n\}$ is the set of n variables,
- $D = \{D(y_1), ..., D(y_n)\}$ where each $D(y_i)$ is the set of d_i possible values for y_i,
- $C = \{C_1, ..., C_m\}$ is the set of m constraints which restricts the values that the variables can simultaneously take,
- $S(k) = (\{0, 1, ..., k\}, \oplus, >)$ is the valuation structure, where $k \in N^+$ denotes the maximum cost.
  - $\{0, 1, ..., k\}$ is the set of costs, which are natural numbers bounded by k.
  - $\oplus$ is the sum of costs $\forall a, b \in \{0, 1, ..., k\}, a \oplus b = min\{k, a + b\}$,
  - $>$ is the standard order among naturals.

For each constraint C_{ij} and each tuple t_p represented by two values (v_r, v_s) from the domains associated with the variables involved in C_{ij}, a cost $c_{rs} \in \{0, 1, ..., k\}$ is assigned to t_p. When a constraint C_{ij} assigns the cost k to a tuple t_k, it means that C_i forbids t_p. Otherwise, t_p is permitted by C_i with the corresponding cost. The cost of an instantiation of variables is the sum (using operator $\oplus$) over all constraints involving variables instantiated.

This problem was reformulated as [10]:

$$\begin{cases} Min \quad f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{r=1}^{d_i} \sum_{s=1}^{d_j} q_{irjs} x_{ir} x_{js} + \sum_{i=1}^{n} \sum_{r=1}^{d_i} q_{ir} x_{ir} \\ s.c \\ \qquad \sum_{r=1}^{d_i} x_{ir} = 1 \\ \quad x_{ir} \in \{0, 1\} \qquad i = 1, .., n \end{cases} \tag{1.3}$$

where:
For each variable y_i of the WCSP problem, we introduce d_i binary variables x_{ir} such that:

$$x_{ir} = \begin{cases} 1 & if\ y_i = v_r \\ 0 & otherwise \end{cases} \tag{1.4}$$

$$x_{ir}x_{js} = \begin{cases} 1 & if\ (v_r, v_s) \in R_{ij} \\ 0 & otherwise \end{cases} \tag{1.5}$$

Each unary constraint C_i is defined by its relation R_i, specifying the privileged values using the notion of cost. Recall that, for each constraint C_i and each value v_r, a cost c_{ir} is assigned to v_r. Then, for each value v_r we generate a constant:

$$q_{ir} = \begin{cases} c_{ir} & if\ v_r \in R_i \\ k & otherwise \end{cases} \tag{1.6}$$

In the same, each binary constraint C_{ij} between variables y_i and y_j is defined by its relation R_{ij} specifying the compatible values between y_i and y_j with certain cost. Recall that, for each constraint C_{ij} and each tuple t_l represented by two values (v_r, v_s) from the domains associated with the variables involved in C_{ij}, a cost c_{rs} is assigned to t_l. Then, for each couple (v_r, v_s) we generate a constant:

$$q_{irjs} = \begin{cases} c_{rs} & if\ (v_r, v_s) \in R_{ij} \\ k & otherwise \end{cases} \tag{1.7}$$

We have illustrated the reformulation of WCSP into a compact mathematical optimization problem, allowing the use of optimization algorithms to find optimal solutions. This reformulation facilitates the use of various optimization techniques, including mathematical programming and metaheuristic algorithms, to solve WCSP instances efficiently.

1.5.2 Task assignment problem

Task assignment problem is given by N tasks and M processors. A set of tasks must be assigned to a set of processors, thus the problem is defined by the minimization of the sum execution costs and the communication costs between the tasks [93, 90, 122, 91, 92]. This problem is stated as a two sets and two parameters where:

- $T = \{T_1, ..., T_N\}$ a set of N tasks.
- $P = \{P_1, ..., P_M\}$ a set of M processors.
- The execution cost e_{ik} of task i if is assigned to processor k.
- The communication cost c_{ikjl} between two different tasks i and j if they are respectively assigned to processors k and l.

Consider the following mathematical model for TAP [12]:

$$\begin{cases} Min \quad f(x) = \sum_{i=1}^{N-1} \sum_{k=1}^{M} \sum_{j=i+1}^{N} \sum_{l=1}^{M} c_{ikjl} x_{ik} x_{jl} + \sum_{i=1}^{N} \sum_{k=1}^{M} e_{ik} x_{ik} \\ s.c \\ \qquad \sum_{k=1}^{M} x_{ik} = 1 \qquad i = 1, .., N \\ \quad x_{ik} \in \{0, 1\} \qquad i = 1, .., N, \quad k = 1, .., M \end{cases} \tag{1.8}$$

For each task $i \in \{1, ..., N\}$, we introduce M binary variables x_{ik}, $k \in \{1, ..., M\}$, such that:

$$x_{ik} = \begin{cases} 1 & if\ the\ task\ i\ is\ assigned to\ processor\ k \\ 0 & Otherwise \end{cases} \tag{1.9}$$

The linear constraints mean that each task should be assigned to exactly one processor.

1.5.3 Traveling salesman problem

The formulation of the Traveling Salesman Problem (TSP) can be described as follows: Given a set of connected cities, with d_{ij} denoting the distances between each pair of cities, its objective is to determine the shortest path that connects all cities while ensuring that each city is visited exactly once [88, 31, 102, 124, 125]. The mathematical model for TSP is given by [29]:

$$\begin{cases} Min \quad f(x) = \frac{1}{2} \sum_k \sum_j \sum_i x_{ki} d_{kj} (x_{j,i-1} + x_{j,i+1}) \\ s.c \\ \qquad e_{j,.}(x) = 1 \quad j = 1, .., N \qquad (e_{j,.}(x) = \sum_{j=1}^{N} x_{ij}) \\ \qquad e_{.,i}(x) = 1 \quad i = 1, .., N \qquad (e_{.,i}(x) = \sum_{i=1}^{N} x_{ij}) \\ \quad x_{ij} \in \{0, 1\} \quad i = 1, .., N, \ \ j = 1, .., N \end{cases} \tag{1.10}$$

For each city $i \in \{1, ..., N\}$ with N the number of cities, we introduce binary variables $x_{ij} \forall i, j \in \{1, ..., N\}$, such that :

$$x_{ij} = \begin{cases} 1 & if\ the\ city\ i\ is\ visited\ in\ the\ order\ j \\ 0 & Otherwise \end{cases} \tag{1.11}$$

The first two constraints reflect the fact that each city must be visited exactly once.

1.5.4 Max-Stable Problem

The Max-Stable Problem (MSP) is a statistical concept related to extreme value theory. It deals with modeling the distributions of extreme values in a set of random variables. Extreme value theory is particularly useful in fields such as finance, environmental science, and engineering, where understanding the distribution of extreme events (e.g., maximum daily rainfall, maximum financial losses) is crucial [53, 85]. The MSP can be expressed in the following algebraic form:

$$\begin{cases} Min \quad f(x) = -\sum_{i=1}^{n} x_i \\ s.c \\ \quad \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} x_i x_j = 0 \\ \quad x_i \in \{0, 1\} \quad i = 1, .., n \end{cases} \tag{1.12}$$

For each node $v_i \in S$ (a set of nodes), we introduce the binary variables x_i such that:

$$x_i = \begin{cases} 1 & if\ v_i \in S \\ 0 & Otherwise \end{cases} \tag{1.13}$$

Two adjacent nodes v_i and v_j cannot be in the set S: $(v_i, v_j) \in E \Rightarrow x_i x_j = 0$.

$$b_{ij} = \begin{cases} 1 & if\ (v_i, v_j) \in E \\ 0 & \textbf{Otherwise} \end{cases} \tag{1.14}$$

1.5.5 Graph Coloring Problem

The Graph Coloring Problem (GCP) is a classic optimization problem in graph theory. The objective is to assign colors to the vertices of a graph in such a way that no two adjacent vertices share the same color. The primary goal is to minimize the number of colors used while satisfying this constraint. This problem finds applications in various real-world scenarios, such as scheduling, register allocation in compilers, and frequency assignment in wireless communication [126].

So that the GCP can be mapped onto a CHN, it must be expressed as a linear assignment problem with quadratic constraints. This problem is indicated by the parameter sets:

- N number of nodes,
- C assigns a number (color) to each node and verifies the following property: $C(i) \neq C(j) \forall \{i, j\} \in E$

The GCP can be expressed in the following algebraic form:

$$\begin{cases} Min \quad f(x) = -\sum_{k=1}^{N} (\frac{N}{2} + \frac{2k}{N}) x_{0k} \\ s.c \\ \quad \sum_{k=1}^{N} (1 - x_{0k}) \sum_{i=1}^{N} x_{ik} = 0 \\ \quad \sum_{k=1}^{N} \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ik} b_{ij} x_{jk} = 0 \\ \quad \sum_{k=1}^{N} x_{ik} = 1 \quad \forall i = 1, ..., N \\ \quad x_{ik} \in \{0, 1\} \quad i = 0, .., N; k = 1, .., N \end{cases} \tag{1.15}$$

Where: The decision variable is the binary variable, which indicates if node i is color k:

$$v_{ik} = \begin{cases} 1 & if\ C_i = k \\ 0 & \textbf{Otherwise} \end{cases} \tag{1.16}$$

The more colors that are used, the worse is the objective function. One way to accomplish this idea is through an auxiliary binary variable that takes the value 1 when a particular color has been used:

$$v_{0k} = \begin{cases} 1 & if\ \textbf{the color k has been used} \\ 0 & \textbf{Otherwise} \end{cases} \tag{1.17}$$

An N-node graph can be characterized by the symmetric adjacency $N \times N$-matrix B:

$$b_{ij} = \begin{cases} 1 & if\ \{i, j\} \in E \\ 0 & \textbf{Otherwise} \end{cases} \tag{1.18}$$

The first constraint ensures that the auxiliary variable v_{0k} takes the value 1 when the color k has been used. The second quadratic constraint, which means that two adjacent nodes cannot have the same color. The third quadratic constraint means that all nodes must have a unique color.

1.5.6 Job Shop Scheduling Problem

The Job Shop Scheduling Problem (JSSP) is one of the most common combinatorial optimization problems, widely recognized for its classification as an NP-hard problem [65]. In this problem, a collection of tasks and a set of resources are given, where each task consists of a sequence of activities to be performed in a specified order. The primary objective of JSSP is to derive a solution that minimizes the total processing time, defined as the time required to complete all tasks. JSSP has practical applications in manufacturing processes, where it has a significant impact on production times and associated costs [65].

The JSSP involves the temporal synchronization of theproduction of J jobs on M machines. Each job J_i is composed of a sequence of p_i operations; each operation op_{ik} has a fixed processing time (duration) pr_{ik} and a variable start time st_{ik} and it requires the exclusive use of a machine for its duration. Each job J_i has a release date rd_i and a due date dd_i between which it should be accomplished. A solution of this problem can be formulated in different ways, as optimization or decision problems. From an Artificial Intelligent (AI) perspective, the JSSP is a constraint satisfaction problem (CSP), i.e. we can formulate a job shop problem as a constraint satisfaction problem named CSP-JSSP. Formally speaking, a CSP-JSSP is defined by:

- $V = \{st_{1,1}, ..., st_{J,p_J}\}$ is a set of N start times, with $N = \sum_{i=1}^{J} p_i$
- $D = \{D(st_{ik})\}$ is a set of domains of each start time.
- $C = \{CP, CC\}$ is set of $m = n_{CP} + n_{CC}$ binary constraints where:
  - $CP = \{CP(st_{1,1}, st_{1,2}), ..., CP(st_{J,p_{J-1}}, st_{J,p_J})\}$ is a set of $n_{CP} = \sum_{i=1}^{J}(p_i - 1)$ constraints of operation precedence. Each constraint $CP(st_{i,k}, st_{i,k+1})$ is defined by linear inequalitie: $u + pr_{ik} \leq v \; u \in D(st_{i,k}), v \in D(st_{i,k+1}) \qquad i = 1, .., J$ and $k = 1, .., p_i - 1$
  - $CC = \{CC(st_{ix}, st_{Jy})\}$ is a set of $n_{CC} = J * M * (J-1)/2$ constraints resource capacity. The form of each constraint $CC(st_{ix}, st_{Jy})$ between two jobs J_i and J_j is: $min(u + pr_{ix}; v + pr_{jy}) \geq max(u; v)$ Where $u \in D(st_{i,x}), v \in D(st_{j,y})$

A solution for the constraint satisfaction problem CSP-JSSP is an assignment of values to all start times st_{ik} so that all constraints are satisfied. This problem CSP-JSSP is modelized as the following knapsack problem (QP):

$$\begin{cases} Min \quad f(x) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{r=1}^{d_i} \sum_{s=1}^{d_j} q_{irjs} x_{ir} x_{js} \\ s.c \\ \qquad \sum_{k=1}^{d_i} x_{ik} = 1 \qquad i = 1, .., N \\ \quad x_{ik} \in \{0,1\}^n \end{cases} \tag{1.19}$$

Where,

$$x_{ik} = \begin{cases} 1 & if \; y_i = v_k \in D(y_i) \\ 0 & Otherwise \end{cases} \tag{1.20}$$

$$q_{irjs} = \begin{cases} 1 & if \; (v_r, v_s) \notin R_{ij}, i \neq j \\ 0 & if \; (v_r, v_s) \in R_{ij}, i \neq j \; or \; i = j \end{cases} \tag{1.21}$$

We have illustrated the reformulation of CSP-JSSP into a compact mathematical optimisation problem, which allows the use of optimisation algorithms to find optimal solutions. This

reformulation facilitates the use of various optimisation techniques, including mathematical programming and metaheuristic algorithms, to solve CSP-JSSP instances efficiently.

1.6 Materials tools and data

In this section, we present the data used as benchmarks to evaluate the performance of the different approaches proposed in this thesis. We have chosen several commonly studied combinatorial optimization problems, including the weighted constraint satisfaction problems, the traveling salesman problem, the task assignment problem, the max-stable problem and the graph coloring problem, for which we have collected representative instances. In order to evaluate and demonstrate the practical relevance of the methodologies outlined in this thesis, we used the datasets provided by the WCSP in [94]. Table (1.1) present the WCSP benchmarks. We used the datasets provided by the TAP in [96]. Table (1.2) present the TAP benchmarks. We have demonstrated the practicality of our approaches in a series of experiments solving the maximum stable set problem. The evaluation instances are provided by the DIMACS Challenge [15]. This is a selected set of instances from the second DIMACS Implementation Challenge (1992-1993). In the following table 1.3, the optimal solution given by the instance generator, as well as the number of nodes and edges. Table (1.4) present the GCP benchmarks used in this thesis [95]. TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types. Table (1.5) present the TSP benchmarks used in this thesis [13].

On the other hand, we studied the planning of tourist tours for visitors to the city of Fez. We considered several metrics, such as the distance between certain points of interest (POIs) and the time it takes to travel between these POIs. To enrich our study, we used the Google Places API and Google Maps as key data sources. Both services provided us with detailed information about places and POIs, including their geographic location, category, distance between them, estimated travel time, and other relevant details. To illustrate the location of popular POIs in Fez, we present Figure 1.3, while Figure 4.11 highlights the different categories as well as the identifiers of the 20 POIs selected for our study. For a better understanding of the data used, tables 1.6 and 1.7 show the distance and time matrices, respectively, between these 20 POIs, obtained using the Google Places API (`https://developers.google.com/places`).

The experiments described in this thesis were carried out using Python on a personal computer equipped with an Intel Core i3-4005U @ 1.70 GHz processor and 4 GB RAM.

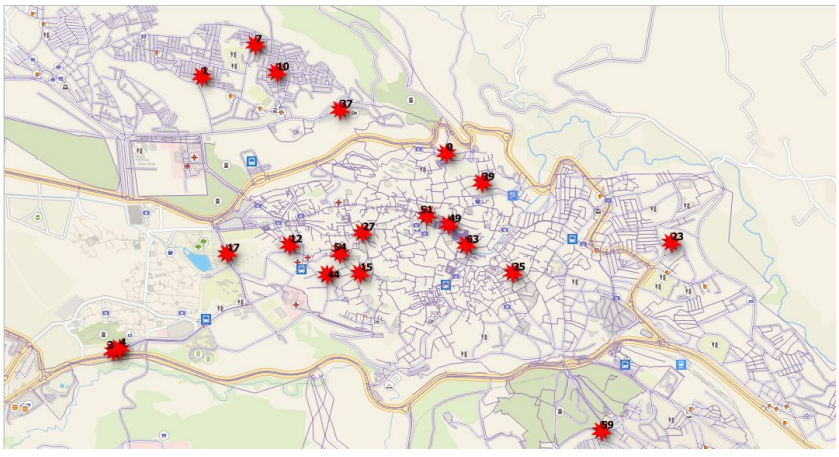


Figure 1.3: The location of Some POIs in Fez

Table 1.1: WCSP dataset

Instance name	Variable numbers	Constraint numbers	Optimal solution
4wqueens	4	10	0
spot5-8	8	15	2
8queens	7	28	0
langford-2-4	8	32	0
8wqueens	8	36	2
queen5-5-3	25	295	29
geom40-2	40	78	22
geom30a-3	30	81	11
queen5-5-4	25	320	12
zebre-ext	023	62	
geom30a-4	30	111	4
geom40-3	40	78	7
myciel5g-3	47	236	16
geom40-4	40	186	3
geom30a-6	30	81	0
myciel5g-4	47	236	4
geom40-5	40	197	1
myciel5g-5	1		
16wqueens	16	200	2
myciel5g-6	47	236	0

Table 1.2: TAP dataset

Instance name	Number of tasks	Number of processors	Optimal solution
tassnu_10_3_1	10	3	-719
tassnu_10_3_2	10	3	-790
tassnu_10_3_3	10	3	-624
tassnu_10_3_4	10	3	-734
tassnu_10_3_5	10	3	-871
tassnu_10_3_6	10	3	-677
tassnu_10_3_7	10	3	-613
tassnu_10_3_8	10	3	-495
tassnu_10_3_9	10	3	-750
tassnu_10_3_10	10	3	-486
tassnu_15_5_1	15	5	-1985
tassnu_15_5_2	15	5	-1568
tassnu_15_5_3	15	5	-1892
tassnu_10_5_4	15	5	-1806
tassnu_15_5_5	15	5	-1881
tassnu_15_5_6	15	5	-1950
tassnu_15_5_7	15	5	-1893
tassnu_15_5_8	15	5	-1733
tassnu_15_5_9	15	5	-1798
tassnu_15_5_10	15	5	-1763

Table 1.3: MSP dataset

Instance name	Optimal solution	Nodes	Edges
brock200_1	18	200	1480
brock200_2	12	200	9876
brock200_4	17	200	13089
brock400_4	33	400	59765
brock800_2	24	800	208166
C125.9	34	125	6963
C250.9	44	250	27984
C1000.9	68	1 000	450079
c_fat200_1	12	200	32000
gen200_p0.9_44	44	200	17910
gen400_p0.9_55	55	400	71820
hamming8_4	16	256	20864
hamming10_4	40	1024	434176
keller4	11	171	9435
keller5	27	776	225990
MANN_a27	126	378	70551
p_hat300_1	8	300	10933
p_hat300_3	36	300	33390
p_hat700_1	11	700	60999
p_hat700_2	44	700	121728

Table 1.4: GCP dataset

Instance name	Vertices	Edges	Optimal solution
myciel3	11	20	4
MYCIEL4	23	71	5
MYCIEL5	47	236	6
MYCIEL6	95	755	7
queen5_5	25	160	5
queen6_6	36	290	7
queen7_7	49	476	7
queen8_8	64	728	9
huck	74	301	11
jean	80	254	10
queen9_9	81	2112	10
1-FullIns_3	30	100	-
2-FullIns_3	52	201	-
1-Insertions_4	67	232	4
4-Insertions_3	79	156	3
2-Insertions_3	37	72	4
3-Insertions_3	56	110	4
mugg88_1	88	146	4
mugg88_25	88	146	4
queen10_10	100	2940	11

Table 1.5: TSP dataset

Instance name	Number of cities	Optimal solution
Buma14	14	3323
Ulysses16	16	6859
Gr17	17	2085
Gr21	21	2707
Ulysses22	22	7013
Gr24	24	1272
Fri26	26	937
Bays29	29	2020
Dantzig42	42	699
Att48	48	10628
Eil51	51	426
Berlin52	52	7542
St70	70	675
Eil76	76	538

Table 1.6: Travel distance between 20 POIs in Km

Id	Id_POI																			
POI	0	1	3	4	7	10	12	15	17	23	27	35	37	39	44	49	51	54	59	63
0	0.0	3.9	3.05	2.86	2.56	2.67	1.81	1.49	1.68	2.11	1.2	1.55	0.76	0.93	1.1	1.0	0.74	1.6	0.98	1.02
1	3.9	0.0	0.94	1.24	1.43	2.44	2.34	2.84	2.26	2.76	2.71	3.76	3.47	3.5	3.75	4.28	3.16	3.85	3.11	3.49
3	3.05	0.94	0.0	0.3	0.49	1.5	1.78	2.28	1.37	1.87	2.15	3.29	2.91	2.94	3.19	3.39	2.6	3.39	2.55	2.94
4	2.86	1.24	0.3	0.0	0.3	1.31	1.6	2.09	1.18	1.68	1.96	3.11	2.73	2.76	3.01	3.2	2.42	3.2	2.36	2.75
7	2.56	1.43	0.49	0.3	0.0	1.01	1.31	1.81	0.88	1.38	1.68	2.82	2.44	2.47	2.72	2.9	2.13	2.91	2.08	2.46
10	2.67	2.44	1.5	1.31	1.01	0.0	1.77	2.3	0.99	1.49	1.99	3.16	2.71	2.74	2.99	3.01	2.4	3.25	2.39	2.77
12	1.81	2.34	1.78	1.6	1.31	1.77	0.0	0.79	0.79	1.29	0.61	1.78	1.38	1.41	1.66	2.22	1.07	1.87	1.01	1.42
15	1.49	2.84	2.28	2.09	1.81	2.3	0.79	0.0	1.31	1.77	0.31	1.11	1.01	1.03	1.18	2.26	0.76	1.2	0.62	0.86
17	1.68	2.26	1.37	1.18	0.88	0.99	0.79	1.31	0.0	0.51	1.0	2.17	1.72	1.75	2.0	2.02	1.41	2.26	1.4	1.78
23	2.11	2.76	1.87	1.68	1.38	1.49	1.29	1.77	0.51	0.0	1.46	2.63	2.18	2.22	2.47	2.44	1.88	2.73	1.87	2.24
27	1.2	2.71	2.15	1.96	1.68	1.99	0.61	0.31	1.0	1.46	0.0	1.17	0.77	0.79	1.05	1.96	0.46	1.26	0.4	0.81
35	1.55	3.76	3.29	3.11	2.82	3.16	1.78	1.11	2.17	2.63	1.17	0.0	0.85	0.9	0.62	2.55	0.84	0.13	0.9	0.55
37	0.76	3.47	2.91	2.73	2.44	2.71	1.38	1.01	1.72	2.18	0.77	0.85	0.0	0.22	0.4	1.76	0.31	0.89	0.52	0.33
39	0.93	3.5	2.94	2.76	2.47	2.74	1.41	1.03	1.75	2.22	0.79	0.9	0.22	0.0	0.5	1.94	0.34	0.96	0.55	0.35
44	1.1	3.75	3.19	3.01	2.72	2.99	1.66	1.18	2.0	2.47	1.05	0.62	0.4	0.5	0.0	2.1	0.59	0.66	0.8	0.32
49	1.0	4.28	3.39	3.2	2.9	3.01	2.22	2.26	2.02	2.44	1.96	2.55	1.76	1.94	2.1	0.0	1.75	2.61	1.98	2.02
51	0.74	3.16	2.6	2.42	2.13	2.4	1.07	0.76	1.41	1.88	0.46	0.84	0.31	0.34	0.59	1.75	0.0	0.93	0.25	0.37
54	1.6	3.85	3.39	3.2	2.91	3.25	1.87	1.2	2.26	2.73	1.26	0.13	0.89	0.96	0.66	2.61	0.93	0.0	0.99	0.61
59	0.98	3.11	2.55	2.36	2.08	2.39	1.01	0.62	1.4	1.87	0.4	0.9	0.52	0.55	0.8	1.98	0.25	0.99	0.0	0.54
63	1.02	3.49	2.94	2.75	2.46	2.77	1.42	0.86	1.78	2.24	0.81	0.55	0.33	0.35	0.32	2.02	0.37	0.61	0.54	0.0

Category	Id_POI	Name	Category	Id_POI	Name
Cemeteries	0	Bab elguissa	Bab	7	Semmarine
	3	Juif		23	Sagma
Schools	39	Misbahiya	Tourist place	63	Seffarine
	54	Sbaiyine		1	Borj Fes
Museum	59	Belghazi		4	Ibn Danan
	12	Batha		10	Palais royal
	44	Tanneries Chouara		37	Zaouiat Sidi Ahemd Tijani
Mosque	35	El Oued		17	Jnane Sbil
Souk	51	Ain Allou		49	Borj Nord
Riad	15	Driss Moqri	Palace	27	Mnbhi

Figure 1.4: Details of the tourist POIs

Table 1.7: Travel time between 20 POIs in hours

Id	Id_POI																			
POI	0	1	3	4	7	10	12	15	17	23	27	35	37	39	44	49	51	54	59	63
0	0.00	0.81	0.62	0.58	0.52	0.53	0.36	0.31	0.34	0.42	0.24	0.31	0.15	0.19	0.22	0.19	0.15	0.33	0.20	0.21
1	0.79	0.00	0.19	0.25	0.28	0.48	0.48	0.58	0.44	0.46	0.55	0.75	0.70	0.70	0.76	0.71	0.64	0.77	0.63	0.71
3	0.62	0.19	0.00	0.06	0.10	0.30	0.36	0.45	0.28	0.28	0.43	0.65	0.58	0.58	0.63	0.54	0.52	0.67	0.51	0.58
4	0.58	0.24	0.06	0.00	0.06	0.26	0.32	0.42	0.24	0.24	0.39	0.61	0.54	0.55	0.60	0.50	0.48	0.63	0.47	0.55
7	0.52	0.29	0.10	0.06	0.00	0.20	0.26	0.36	0.18	0.18	0.34	0.56	0.49	0.49	0.54	0.44	0.43	0.58	0.41	0.49
10	0.53	0.49	0.30	0.26	0.20	0.00	0.34	0.44	0.19	0.19	0.38	0.61	0.53	0.53	0.58	0.45	0.47	0.63	0.46	0.54
12	0.36	0.48	0.36	0.32	0.26	0.34	0.00	0.16	0.16	0.25	0.12	0.35	0.27	0.27	0.33	0.33	0.21	0.37	0.20	0.28
15	0.31	0.57	0.45	0.42	0.36	0.44	0.16	0.00	0.26	0.34	0.06	0.22	0.21	0.21	0.24	0.42	0.15	0.24	0.12	0.17
17	0.34	0.47	0.28	0.24	0.18	0.19	0.16	0.26	0.00	0.09	0.19	0.42	0.34	0.34	0.39	0.26	0.28	0.44	0.27	0.35
23	0.42	0.47	0.28	0.24	0.18	0.19	0.25	0.34	0.09	0.00	0.28	0.50	0.42	0.42	0.47	0.28	0.36	0.52	0.36	0.43
27	0.24	0.55	0.43	0.39	0.34	0.38	0.12	0.06	0.19	0.28	0.00	0.23	0.15	0.15	0.21	0.35	0.09	0.24	0.08	0.16
35	0.31	0.75	0.65	0.61	0.58	0.58	0.58	0.22	0.22	0.50	0.23	0.00	0.17	0.18	0.13	0.51	0.16	0.03	0.17	0.11
37	0.15	0.70	0.58	0.54	0.49	0.53	0.27	0.21	0.34	0.42	0.15	0.17	0.00	0.04	0.08	0.35	0.06	0.18	0.11	0.07
39	0.19	0.70	0.58	0.55	0.49	0.53	0.27	0.21	0.34	0.42	0.15	0.18	0.04	0.00	0.10	0.38	0.06	0.19	0.11	0.07
44	0.22	0.75	0.63	0.60	0.54	0.58	0.33	0.24	0.39	0.47	0.21	0.13	0.08	0.10	0.00	0.42	0.12	0.13	0.16	0.07
49	0.19	0.73	0.54	0.50	0.49	0.49	0.38	0.26	0.26	0.28	0.35	0.51	0.35	0.38	0.42	0.00	0.35	0.52	0.40	0.40
51	0.15	0.64	0.52	0.48	0.43	0.47	0.21	0.15	0.28	0.36	0.09	0.16	0.06	0.06	0.12	0.35	0.00	0.18	0.18	0.09
54	0.33	0.76	0.67	0.63	0.58	0.63	0.37	0.24	0.44	0.52	0.24	0.03	0.18	0.19	0.13	0.52	0.18	0.00	0.19	0.12
59	0.20	0.63	0.51	0.47	0.41	0.46	0.20	0.12	0.27	0.36	0.08	0.17	0.11	0.11	0.16	0.40	0.05	0.19	0.00	0.11
63	0.05	0.70	0.58	0.58	0.49	0.54	0.28	0.17	0.35	0.43	0.16	0.11	0.07	0.07	0.07	0.40	0.07	0.12	0.11	0.00

1.7 Conclusion

In this chapter, we discuss the fundamental concepts of combinatorial optimization, a crucial area where the desired solutions often arise from a finite set of possibilities. We begin with a historical overview of the types of combinatorial optimization problems, providing an evolving view of this dynamic field.

The historical overview highlights the emergence and evolution of the main challenges and methods in combinatorial optimization and shows the progress from the first formulations of these problems to contemporary advances. In the second part of the chapter, we dive into a detailed examination of some fundamental combinatorial optimization problems. This exploration illustrates the variety and complexity of the situations encountered, as well as the specific algorithmic approaches used to solve them. Among the problems studied are classics such as the traveling salesman problem, the knapsack problem, the maximum flow problem, and many others. Each problem is presented with its own characteristics, its own constraints, and the most relevant algorithmic techniques for its effective solution.

Chapter 2

Exploring Hopfield Recurrent Neural Networks for Combinatorial Optimization: Methods and Dynamics

2.1 Introduction

Artificial Neural Networks (ANNs) are a powerful category of computer models inspired by the intricate workings of the human brain. These sophisticated structures, commonly referred to as neural networks, exhibit a remarkable ability to learn from data and extrapolate acquired knowledge to solve a wide range of complex problems. Modeled on the synaptic connections within the human brain, where interconnected neurons work together to process information in a distributed and parallel manner, ANNs showcase a design that harnesses the principles of neural computation. The versatility of ANNs spans multiple domains, from computer vision and natural language processing to speech recognition and beyond. Their ability to recognize intricate patterns within data sets gives them formidable capabilities, making them invaluable tools for a wide range of tasks. Neural networks can be divided into two categories. Feed-forward neural networks (FFNN) are the first and simplest category. In these networks, information flows only from the input layer directly through a set of hidden layers, generating the observed information stored in the output layer without cycles. Feedforward networks can be constructed with different types of units, such as binary or sigmoidal McCulloch-Pitts neurons. The best-known feedforward ANN models are multilayer perceptrons and the radial basis function (RBF). The other type of ANN, called a recurrent neural network (RNN), is a dynamic model that contains at least one feedforward connection so that activations can loop back and forth.

The rest of this chapter is organized as follows: Section 2.2, a brief describes the Hopfield model as one of the most known recurrent neural network. In Section 2.3, we take a brief look at solving constrained optimization problems with Hopfield neural network. Then, in Section 2.4, we carried out a comparative study between a Hopfield neural network, a metaheuristic approach, and an exact method. The use of the Crank-Nicolson scheme for the equilibrium of continuous Hopfield networks is discussed in Section 2.5. Finally, a brief conclusion is given in Section 2.6.

2.2 General aspects for Hopfield recurrent neural networks

Recurrent neural networks (RNNs) are an important class of models in the field of deep learning, characterized by the presence of recurrent connections that allow the storage of information about previous states. RNNs are designed to process sequential data by introducing recurrent connections between units, allowing an internal memory of previous states to be maintained. This ability to handle sequentiality makes them particularly well suited to tasks such as sequence prediction, machine translation, and text generation. Hopfield Neural Networks (HNNs) belong to non-training model. In particular, HNNs are considered as feedback (or recurrent) without learning from a pattern set. The HNNs consist of a set of neurons and a corresponding set of unit delays, forming a multiple loop feedback system. Each feedback loops is equal to a neuron. Basically, the output of each neuron is feedback, via a unit delay element, to each of the other neurons in the system. HNNs are not affected by additional inputs, and each neuron is fully connected with the remaining neurons by weights and always updates the connection weights. The associative memory or content addressable memory is a memory organization that accesses memory by its contents and it always searches for the closest matching from all stored prototypes. This is analogous to the storage of information in an associative memory as biological neurons. In other words, the process of association and information retrieval is simulated by the dynamical behavior of a highly interconnected system of non-linear circuit elements with collective computational abilities.

2.2.1 Classifications of Hopfield neural networks

The Hopfield neural network networks is a recurrent neural network model introduced by John Hopfield in 1982 [19, 131]. It is composed of a layer of fully connected neurons and has been widely applied in numerous domains, including optimization problems [103, 5, 21, 123, 116, 124, 125, 99, 31], operations research [104, 119, 22], image restoration [100, 105, 113, 114], economics [106, 108], electronic circuit design [109, 115, 117], computer science, and other applications.

HNN synaptic connection pattern, whereby there is a Lyapunov function for the activity dynamics. In the original HNN model, the state of the system evolves from any initial state to a final state, where it is a (local) minimum of the Lyapunov function. Based on the output functions, HNNs can be classified into two popular forms: discrete and continuous-time models.

a. **Discrete Hopfield networks**

The Hopfield neural network can be represented by a fully interconnected network with undirected weights (Figure 2.1). The actions of the neurons in discrete/discrete time HNN is calculated through the following formula:

$$Y_i(t+1) = sign(\sum_{j=1}^{n} T_{ij} x_j(t) - I_i), \quad i = 1, ..., n; j = 1, ..., n. \tag{2.1}$$

Thus, the vector Y is given as:

$$Y = sign(Tx - I) \tag{2.2}$$

Where,

- T is the weight matrix: $T = \begin{bmatrix} T_{11} & T_{12} & T_{13} & \dots & T_{1n} \\ T_{21} & T_{22} & T_{23} & \dots & T_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T_{d1} & w_{d2} & T_{d3} & \dots & T_{dn} \end{bmatrix}$

  where T_{ij} is the connection between the ith and j^{th} neurons.

- I is the bias vector: $I = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix}$

  where I_i is the threshold of unit i.

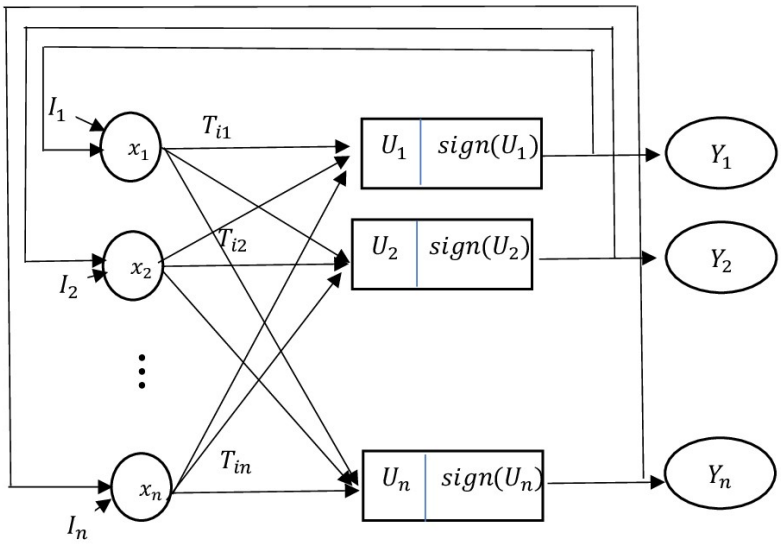


Figure 2.1: The discrete Hopfield network architecture

In the discrete HNN, the units use a bipolar output function where the states of the 1 or 0, i.e., the output of the units remain the same if the current state is equal to some threshold value:

$$x_i = \begin{cases} 0 & if \ \sum_j T_{ij} x_j(t) < I_i \\ 1 & Otherwise \end{cases} \tag{2.3}$$

In general, the property of a recurrent networkcan be described as an energy function, which is used to improvethe stability of recurrent network. Hopfield presented an energy function to demonstrate its stability as:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i T_{ij} x_j - \sum_{i=1}^{n} x_i I_i \tag{2.4}$$

Theorem 1. *The energy function moves from any state to a stable state, which is the local minimum of the energy function, given a symmetric and zero-diagonal weight matrix and asynchronous dynamics.*

Proof. Due to the asynchronous nature of the dynamics, only one unit is evaluated at a time. When unit k is selected, there are two possibilities:

- The state of unit k remains unchanged.
- The state of unit k changes from 0 to 1 or from 1 to 0.

In the first case, the energy function remains unchanged $E(t+1) = E(t)$.

In the second case, The revised excitation value (x'_k) of the k unit is used to modify the energy function. The energy change of the system is represented as:

$$E(x) - E(x') = -(\sum_{j=1}^{n} x_j T_{kj} x_k - \sum_{k=1}^{n} I_k x_k) - (-(\sum_{j=1}^{n} x'_j T_{kj} x'_k - \sum_{k=1}^{n} I_k x'_k))$$

then:

$$E(x) - E(x') = -(x_k - x'_k) * (\sum_{j=1}^{n} T_{kj} x_j - I_k) \tag{2.5}$$

The second term of this equation $(\sum_{j=1}^{n} w_{kj} x_j - I_k)$ is the excitation of the k unit. The unit changed its state, so the excitation has a different sign than x_k and $x_{k'}$ according to (2.2). Therefore, the Eq (2.5) is positive, which implies that $E(x) > E(x^{'})$. So the energy will decrease for every change, and since there is only a finite number of possible states, the network should reach a state where the energy cannot decrease more, it will be a stable state. Of course the stable state is not unique, but it is certain that the system will converge. The final state will depend on the input and the initial state of the system. □

b. **Continuous Hopfield networks**

In 1982, Hopfield and Tank proposed another type of recurrent neural network, the continuous Hopfield network (CHN). The CHN is a generalization of the discrete case and has been widely used to solve combinatorial optimization problems. The CHN is characterized by a single layer of n neurons that are fully recurrently connected, and each neuron has an activation function Figure 2.2. The dynamics of this network are described by a differential equation that is defined in terms of the connection weight between two neurons, denoted by T, and the bias i^b. The equation is as follows [29]:

$$\frac{du}{dt} = -\frac{u}{\tau} + Wx + I \tag{2.6}$$

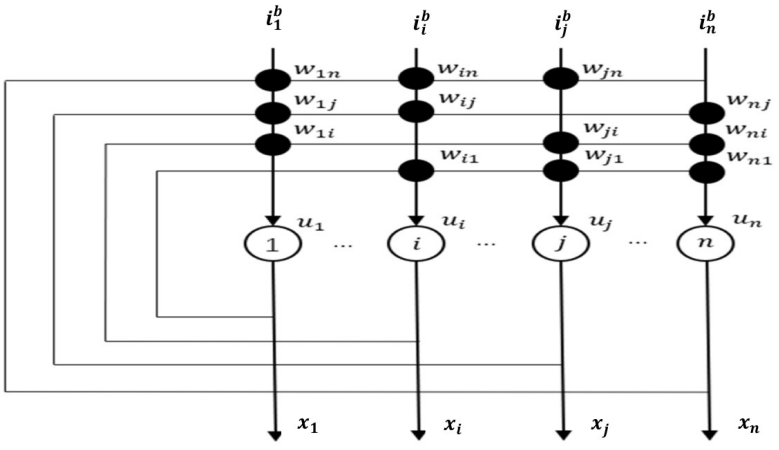


Figure 2.2: The continuous Hopfield network architecture

Where $W \in R^{(n\times n),(n\times n)}$ is the connection weight between the neurons, $I \in R^{(n\times n)}$ is the bias, u is the state of the neuron, and x is the output. τ is a time constant.

In a Hopfield neural network, the relationship between the internal state of a neuron u_i and its output level x_i is given by an activation function $x_i = g(u_i)$ (usually a hyperbolic tangent), which is bounded on the bottom by 0 and on the top by 1 defined by:

$$g(u_i) = \frac{1}{2}(1 + tanh(\frac{u_i}{u_0})) \quad u^0 > 0, \ \ \forall i = 1, .., n \tag{2.7}$$

Definition 7 (Equilibrium point)**.** *A point u^e is called an equilibrium point of the system (4.1) if for an input vector u^0, u^e satisfies $u(t) = u^e \ \forall t \geq t_e$ for some $t_e \geq 0$ [26].*

The Lyapunov function E_{Lyap} is used to study the stability of the HNN. This function is used to show that the network reaches a stable state since it must decrease with time ($\frac{dE_{Lyap}}{dt} \leq 0$) for the system to converge to a minimum. This Lyapunov function E_{Lyap} on $[0, 1]^n$ is expressed as follows [29]:

$$E_{Lyap}(x) = -\frac{1}{2}x^t W x - I^t x + \frac{1}{\tau}\sum_{i=1}^{n}\int_0^{x_i} g^{-1}(x)dx \tag{2.8}$$

when $\tau \to \infty$, the Lyaponuv function is associated with the cost function to be minimized in the combinatorial problem.

Thus, CHN will solve problems with the following form:

$$E(x) = -\frac{1}{2}x^t W x - I^t x \tag{2.9}$$

The equilibrium point of this network exists if the Lyapunov function exists. In 1984 Hopfield showed that if the matrix W is symmetric, then the Lyapunov function E_{Lyap} exists.

2.2.2 Stability analysis of CHN

In this section, we derive the conditions under which all neurons' activities are either firing at state 1 or off at state 0, i.e.,the network output vector converges to the vertices satisfying the constraints. There are three main approaches to determine the convergence states of a Hopfield network: eigenvalue analysis of the connection matrix, comparison of the energy values at vertices of the unit hypercube, and Lyapunov stability analysis.

A point $x \in H = \{x \in [0, 1]^n\}$, will be an equilibrium point for the CHN if and only if, the three following relations are verified:

$$E_i(x) = \frac{\partial E(x)}{\partial x_i} \geq 0 \quad \forall i = 1, ..., n \quad such \quad that \quad x_i = 0 \tag{2.10}$$

$$E_i(x) = \frac{\partial E(x)}{\partial x_i} \leq 0 \quad \forall i = 1, ..., n \quad such \quad that \quad x_i = 1 \tag{2.11}$$

$$E_i(x) = \frac{\partial E(x)}{\partial x_i} = 0 \quad \forall i = 1, ..., n \quad such \quad that \quad x_i \in]0, 1[\tag{2.12}$$

Lemma 2.1. *A vertex of the network state hyper-cube is asymptotically stable if:*

$$\frac{\partial E}{\partial x_{ik}}(x) \begin{cases} > 0 & (x_{ik} = 0) \\ < 0 & (x_{ik} = 1) \end{cases} \tag{2.13}$$

and unstable if [66, 29]:

$$\frac{\partial E}{\partial x_{ik}}(x) \begin{cases} < 0 & (x_{ik} = 0) \\ > 0 & (x_{ik} = 1) \end{cases} \tag{2.14}$$

Theorem 2. *In the Hopfield network, all the infeasible solutions are unstable [66].*

Theorem 3. *In the Hopfield network, a feasible solution is asymptotically stable or and unstable [66].*

2.2.3 CHN equilibrium points algorithm

Continuous Hopfield neural networks are used to solve optimization problems in the sense that the equilibrium points of a system of differential equations associated with CHN constitute the solution to these problems. The dynamics of this network are formulated as an analytic differential equation. This later gives the neural network scheme of numerical discretization, in which the state of each neuron changes with iteration. Much of the work on this type of network has been remarkably important in its application in various fields. On the other hand, the discretization of a continuous system does not always benefit from mathematical tools that can be applied. However, the CHN system of differential equations is very difficult to solve analytically because the model has a differential equation for each neuron and is therefore solved using numerical approaches. Euler's method is a numerical technique that relies on a linear approximation to predict future values based on the current state. While its simplicity is an advantage, it can have stability issues, particularly when applied to certain types of problems.

The simulation of CHN poses significant challenges because the model involves a differential equation for each neuron, resulting in a large and complex system of equations. In this complicated network, Euler's method stands out as a practical and efficient tool. Its simplicity and speed make it a necessary choice for tackling the systems of differential equations that characterize CHN. In situations where computational efficiency and straightforward implementation are crucial, Euler's method becomes an indispensable asset. Despite its known limitations in dealing with stability issues, its ability to quickly solve systems of differential equations such as those encountered in CHN simulations makes it a valuable approach in the field of neural network modeling. Therefore, Euler's method plays a crucial role in making CHN simulations feasible, providing a balance between computational efficiency and the intricate nature of the differential equations governing complex neural systems.

Talavan in 2005 [16] used Euler's method for the solution of the equation (4.1), which is considered a classical method with one step, allowing, from the initial condition, to calculate the values of the solution at each step. This method has been used in most of the CHN work. The Euler explicit scheme is given by the recurrence relation: Given a time step $\Delta t > 0$ the two versions of the Euler method [127]:

$$u_i(t + \Delta t) = u_i(t) + \Delta t \frac{du_i}{dt} \qquad \forall i \in \{1, \ldots, n\} \tag{2.15}$$

This formula outlines the iterative process of the Euler method, where $u_i(t+\Delta t)$ represents the solution at the next time step, $u_i(t)$ is the solution at the current time, and $\frac{du_i}{d}$ represents the derivative of the solution with respect to time. This recurrence relation is applied for each i in the range $\{1, \ldots, n\}$, where n is the number of neurons in the CHN.

This method has a local truncating error of magnitude $O(\Delta^2)$ in each step, which is relatively large, thereby requiring the use of a very small step size to the detriment of a lower speed and a more rounded error in the floating point computations.

The scheme of the algorithm which obtains an equilibrium point for the CHN is depicted in (Fig.2.3).

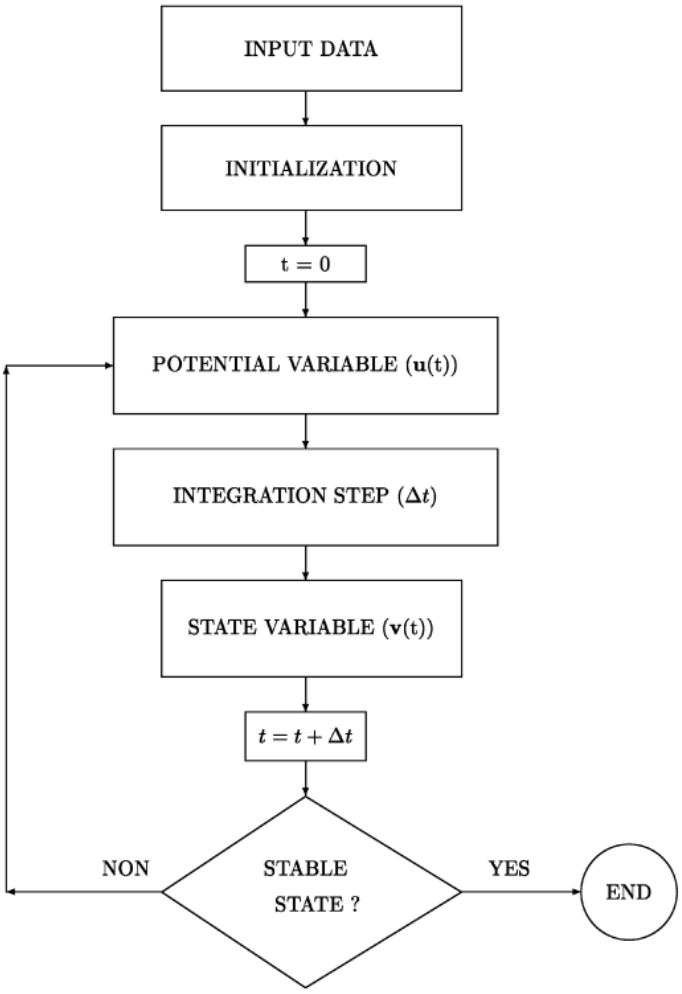


Figure 2.3: CHN equilibrium points algorithm [16]

2.3 CHN for constrained optimization problems

CHN is a highly effective tool for solving combinatorial optimization problems (COPs). The mathematical model of COP is described by the system of equations given by [20, 128, 12, 129, 130]:

$$(P)\begin{cases} Min \quad f(x) \\ s.c \\ \qquad g(x) = 0 \\ \qquad x \in \{0,1\} \end{cases} \tag{2.16}$$

The following sets are necessary for the resolution of the problem (P) [29]:

- The Hamming hypercube $H \equiv \{x \in [\ 0,1]\ ^n\}$: The hyperbolic hamming cube (H) represents all possible configurations of binary vectors of length n.
- The Hamming hypercube corners $H_C \equiv \{x \in H/\ x \in \{0,1\}^n\}$: The corners of the hyperbolic Hamming cube (H_C) represent specific configurations of the Hamming cube.

- The set of feasible solutions $H_F \equiv \{x \in H_C / \ g(x) = 0\}$: The set of feasible solutions (H_F) represents the valid configurations that satisfy all the constraints imposed by the problem (P).

Solving optimization problems using the continuous Hopfield neural network generally involves the use of an energy function. This function is a mathematical representation of the problem to be solved [43]. The local minima of this energy function correspond to the optimal solutions of the problem to be solved [62, 64, 29]. This function is defined as:

$$E(x) = \alpha f(x) + \sum_i \phi_i (G \circ g)(x) \qquad \forall x \in H \tag{2.17}$$

The first term is proportional to the objective function ($f(x)$) and the second term is proportional to the constraints ($g(x)$). In addition, $(\alpha, (\phi_i)_i)$ are the penalty parameters must be chosen for a particular problem.

Although CHN is the most popular model of NNs for optimization, it suffers from some drawbacks:

(i) the minimal solution does not guarantee global optimization;

(ii) the penalty parameters are not easily implemented;

(iii) the energy function may pose many unnecessary local solutions which are difficult to avoid.

Until now, many researchers have tried to improve the drawbacks of CHNs, and we briefly show offer attention so that future studies could follow

2.3.1 Parameter setting

Selecting appropriate parameters is a challenge when using CHNs. If these parameters are not accurately defined during the computation, the energy function may produce solutions that are ultimately impractical. The process of selecting parameters for Hopfield neural networks is critical to increasing their efficiency in solving optimization problems. In the field of CHNs, model-free algorithms are emerging as the dominant methods for parameter selection. These algorithms include the following techniques:

- Baby-sitting: This approach relies on empirical testing of numerous parameter values, drawing on empirical knowledge and insights gained through iterative experimentation [20].
- Grid search: This method systematically evaluates all possible combinations of parameters from a predefined configuration grid [20].
- Random search: This method involves randomly sampling a predetermined number of parameter configurations from the search space [20]. Unlike grid search, which exhaustively explores all combinations, random search provides a more efficient approach by focusing on a subset of parameter settings.

These methods involve an iterative adjustment of the parameters of the HNN to fit within a predefined range of values. An advantage of these approaches is their compatibility with parallel computing [46]. However, a significant limitation arises when researchers and practitioners create parameter configurations that may not be universally suitable for all scenarios. To elaborate, these methods involve a repetitive process in which HNN parameters are systematically modified within a given range of values.

2.3.2 Initial state

In the field of combinatorial optimization problems, the CHN is a promising tool for finding optimal solutions. However, the use of CHN comes with its own set of challenges, one of which revolves around the critical decision of selecting an appropriate initial state. The initial state has a direct impact on the network's convergence behavior and ultimately on the quality of the solution obtained. If the initial state is chosen arbitrarily, there is a risk of convergence to infeasible solutions, rendering the optimization process ineffective. To mitigate this risk, researchers advocate choosing the initial state within a feasible region of the problem space. By restricting the initial state to regions where valid solutions exist, the network is guided to explore solution spaces that are relevant and meaningful. In addition, the complexity involved in choosing the initial state leads some researchers to argue for its proximity to the optimal solution. Placing the initial state close to the optimal solution can speed up convergence by steering the optimization process towards promising regions of the solution space. This approach is particularly valuable in scenarios where computational resources are limited, as it accelerates the search for high-quality solutions. However, striking a balance between feasibility and proximity to optimality is a nuanced endeavor. Researchers must navigate the trade-offs between exploring diverse solution spaces and efficiently converging on optimal solutions. Thus, the choice of initial state remains a crucial consideration in realizing the full potential of CHNN for tackling combinatorial optimization problems.

2.4 Comparative study between a Hopfield neural network, approach metaheuristic and exact method

In this section, we have carried out a comparative study of three different approaches to solving combinatorial optimization problems: the continuous Hopfield network, ant colony optimization, and the exact CPLEX method. Each of these methods has unique characteristics and potential advantages in specific contexts. The continuous Hopfield network (CHN) uses a model of interconnected neurons to search for a stable state that represents the optimal solution to the problem. It is capable of handling complex search spaces thanks to its inherent parallelism, which allows rapid exploration of potential solutions and convergence to possible suboptimal states. Ant colony optimization (ACO) is inspired by the behavior of ants to solve combinatorial optimization problems. This approach is suitable for dynamic environments, has the ability to find global solutions through cooperation, but is sensitive to parameters such as the rate of pheromone evaporation and can converge to suboptimal solutions in some cases [137]. The exact method, implemented in particular by CPLEX, is based on rigorous mathematical algorithms to guarantee the optimal solution of the combinatorial optimization problem. It guarantees to find the optimal solution and is efficient for moderate-sized problems. However, it can be prohibitively expensive in terms of computational resources for large problems and may not scale as dynamically as heuristic methods.

2.4.1 Comparative study

In this section, we will present a series of experiments on the three methods (CHN, ACO, and exact resolution) on Traveling Salesman problems, comparing the results obtained from a group of aspects that allow us to draw a series of conclusions about the effectiveness of each of the above methods.

The results of the simulations are presented in tables (2.1, 2.2, 2.3), which are given in terms of the minimum, average, and mode of the rounds obtained, as well as the execution

time and the number of iterations. The measure of the quality of the solutions is given by the performance ratio:

$$ratio = \frac{tour\,length}{optimum\,tour\,length} \tag{2.18}$$

Table 2.1: Computational experiments for some TSPLIB examples by the continuous Hopfield network

TSP identification	Ratio	Mean	Mode	Execution Time (s)	Iteration Sum
Att48	1.39	1.43	1.43	22.574	273
Bays29	1.66	2.24	2.32	9.431	303
Berlin52	1.19	1.92	1.81	14.469	151
Burma14	1.11	1.15	1.18	0.45	61
Dantzig42	1.12	1.39	1.53	6.993	96
Eil51	1.42	2.63	2.42	36.1	350
Eil76	2.19	2.35	2.42	120	600
Fri26	1.62	2.13	2.21	0.803	32
Gr17	1.29	1.35	1.62	0.966	87
Gr21	1.13	1.22	1.17	1.391	79
Gr24	1.55	2.11	2	0.819	34
St70	2.09	2.25	2.3	10.1269	537
Ulysses16	1.22	1.75	1.66	0.145	15
Ulysses22	1.46	1.88	1.77	0.839	26

Table 2.2: Computational experiments for some TSPLIB examples by the Ant colony

TSP identification	Ratio	Mean	Mode	Execution Time (s)	Iteration Sum
Att48	1.52	1.48	1.41	169	201
Bays29	1.90	2.41	2.14	73	201
Berlin52	1.22	1.82	1.45	223	201
Burma14	1.35	1.40	1.45	22.7	101
Dantzig42	1.17	1.42	1.54	201	201
Eil51	1.55	2.84	2.80	207	201
Eil76	2.44	2.56	2.62	346	201
Fri26	1.92	2.02	2.02	85	201
Gr17	1.72	1.90	1.88	27.7	101
Gr21	1.54	1.66	1.61	65	201
Gr24	2.21	2.52	2.52	91	201
St70	2.24	2.55	2.64	376	201
Ulysses16	1,69	1,86	1.91	30.4	101
Ulysses22	2.10	2.41	2.30	85	101

Table 2.3: Computational experiments for some TSPLIB examples by the exact resolution

TSP identification	Ratio	Mean	Mode	Execution Time (s)	Iteration Sum
Att48	1	1.23	1.23	1565	10628
Bays29	1	2.82	3.07	9	9696
Berlin52	1	1	1	2220	1000
Burma14	1	1.05	1.05	2.3	1531
Dantzig42	1	1	1	1380	2200
Eil51	1	1	1	2040	3200
Eil76	1	1	1	3840	3750
Fri26	1	1.02	1.02	2	2541
Gr17	1	1.02	1.02	3.8	320
Gr21	1	1	1	3	97
Gr24	1	1	1	1.66	215
St70	1	1	1	3600	3200
Ulysses16	1	1.33	1.01	1.66	19479
Ulysses22	1	1.08	1.02	26.03	374175

2.4.2 Comparison and discussion

According to Tables (2.1, 2.2, 2.3) which represents the results as a function of good solutions, the minimum, average time and the iterations for different instances. The two figures at the bottom represent a comparison of the three solving methods.

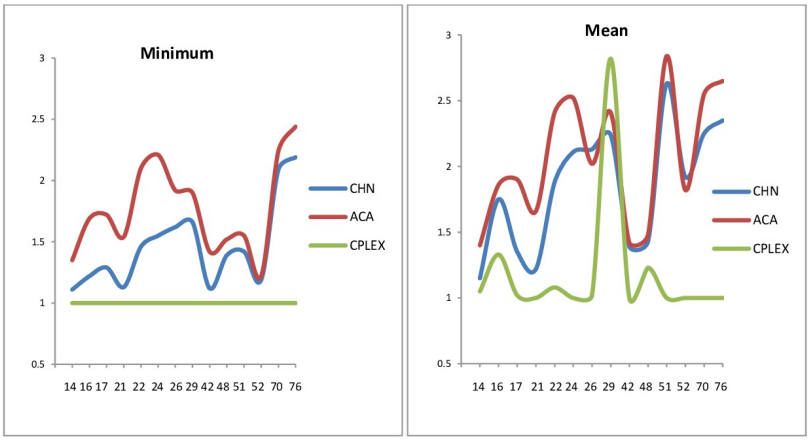


Figure 2.4: Minimum and Mean that CHN, ACA and Cplex gives an optimal value for the different instances

From the first figure concerning the Minimum of each instance for the three methods of resolution, we can see that, the solutions of the exact resolution are always optimal for all the instances what is normal, because the algorithm seeks among all the possible solutions the optimal solution, then the solutions of the continuous Hopfield network are the most approximated to the optimal solution compared to the solutions of the colony of ants which are of bad quality compared to CHN and CPLEX. This performance by exact solving and Hopfield continuous network is validated in the second graph which shows the average of feasible solutions. Finally, Cplex and CHN average the two approaches that gave optimal solutions.

The average of each instance can be justified by CPLEX, the CPLEX software guarantees

the "The solution pool" which allows to generate and store several solutions to a model to generate several solutions in addition to the optimal solution.

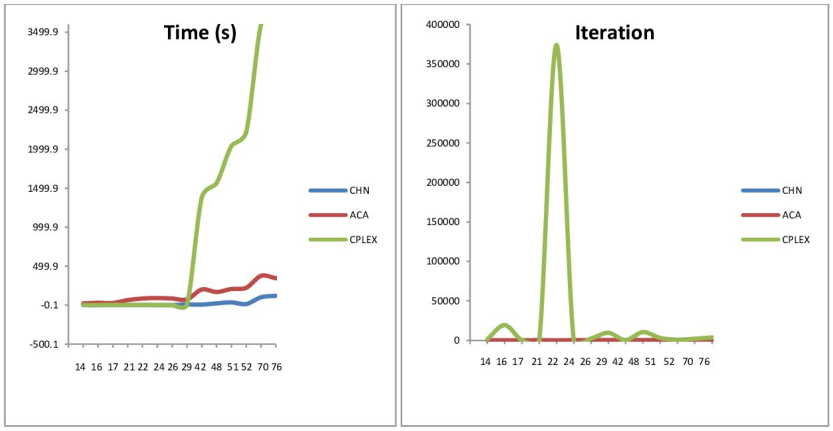


Figure 2.5: Sum time and iteration that CHN, ACA and Cplex gives an optimal value for the different instances

On the other hand, the second figure that represents the execution time and the sum of iterations for the different instances with the given resolution methods, we can notice that the execution time of CHN is smaller compared to CPLEX and ACA, then the sum of the execution time increases with the increasing number of cities, which is normal because the instances of the algorithms increase and this increase is also justified by the number of iterations. From the second figure, we can notice that the sum of the iterations by the Hopfield network and the ant colony is lower compared to the exact method.

From the results of tables (2.1, 2.2, 2.3) and the two figures, we can group these results to conclude the classification of each method used in relation to each property in the perspective of knowing the advantage of each method in different properties of the obtained solution.

Table 2.4: CHN, ACA and CPLEX rankings for the four properties

Methods of resolution	Minimum	Mean	Execution Time	Iteration Sum
CHN	2	2	1	1
ACA	3	3	2	2
CPLEX	1	1	3	3

From the result of the table above, we can see that the performance of each method if we take for example CHN takes first class in the execution time and also the number of iterations. On the other hand, CPLEX takes the first class at the level of the optimal solution.

This section presents a comparative study of two main categories of combinatorial optimization problem solving methods, exact methods and approximate methods called meta-heuristics, the two widely used optimization solving techniques, namely CHN and ACA. The experimental results show that CPLEX can obtain an exact optimal solution every time, but the computation time is long, on the other hand the approximate methods can give a result close to the optimal solution with a shorter execution time. The dynamics of the CHN is described by a system of differential equations that are difficult to solve analytically. Therefore, researchers use the Euler-Cauchy method to calculate the equilibrium point of the CHN. Unfortunately, this method suffers from several problems, including the quality of the decision for a large step, the sensitivity to the parameters of the gradient function, and the initial conditions. In what

follows, we use the numerical method known as the Crank-Nicolson method, which offers better numerical accuracy and is less sensitive to variations in numerical parameters such as the time step.

2.5 Using Crank-Nicolson scheme for continuous Hopfield network equilibrium

The dynamics of the CHN is described by the first-order differential equation that describes the state of each CHN neuron [16, 131]. Many works use Euler's method to approximate the Hopfield network differential equation because it is the simplest numerical method for solving differential equations [16]. The development of the results and efficiency of CHN has prompted some to consider using a different method among the existing methods for solving first-order differential equations. In the work [8], they used a method based on a multi-step scheme called Adams-Bashforth, which showed good results and better performance than the classical Euler method [8]. In order to increase the performance of the CHN for wedge in this work, we use another method for solving the differential equation of the CHN and stay within the framework of one-step methods based on the progressive and regressive Euler method, which is the method of Crank and Nicolson. The Crank and Nicolson scheme is more accurate than the two Euler schemes (progressive and regressive). The advantage of the Crank-Nicholson scheme is that it will reduce the error on the time derivative to $O(\Delta t^2)$, instead of $O(\Delta t)$ [132, 133].

2.5.1 The proposed algorithm

We will stay with one-step schemes the Crank-Nicolson (CN) method for simulating scattering systems described by Crank and Nicolson (1947) [134]. It is similar to the trapezoidal method in the domain of ordinary differential equations [135]. In general, the characteristic of finite difference methods is the convergence of the exact solution. In this work, we use the Crank-Nicolson method for solving the CHN differential equation (4.1), whose formula is given by: Given a time step of $\Delta t > 0$, the CN method scheme is as follows [132, 136]:

$$u_i(t + \Delta t) = u_i(t) + \frac{\Delta t}{2}(\frac{du_i(t)}{\mathrm{d}t} + \frac{du_i(t + \Delta t)}{\mathrm{d}t}) \tag{2.19}$$

In this work, we rely on the algorithm to minimize the CHN energy function (4.5) by using the CHN method to solve their differential equation (4.1). This algorithm is based on the principle of an algorithm to obtain the equilibrium point [16], but there are a set of modifications that we have made which are reflected in:

- We have used the CN scheme to solve the CHN differential equation (4.1).
- To get better solutions and reduce execution time, we changed the stopping criterion from the one used in [16].
- We have chosen the continuous type for selecting the energy function parameters (2.17).

2.5.2 Using Crank Nicolson scheme for solving CHN applied to task assignment problems

We conducted an analytical and comparative study in order to compare the performances of the two schemes (Euler and CN). To perform this experiment, we used the combinatorial optimization problem Task assignment problem (TAP) [12]. The objective of this study is to

Algorithm 1 The algorithm to obtain the equilibrium point of CHN by the Crank-Nicolson method

Input Data
Initialization of the problem data
$t \leftarrow 0$
Initialize v_0 (Starting point), $u(t=0) = \frac{u_0}{2} ln(\frac{v(t=0)}{1-v(t=0)})$, ε
while $max_i(|v_i(t+\Delta t) - v_i(t)|) \geq \varepsilon^3$ **do**
    Generate the hyper-parameters of function energy (2.17)
    Calculate the time step Δt
    Compute $u_i(t+\Delta t) = u_i(t) + \frac{\Delta t}{2}(f(u_i(t)) + f(u_i(t+\Delta t)))$
    Compute $v_i(t+\Delta t) = \frac{1}{2}(1 + tanh(\frac{u_i(t+\Delta t)}{u_0}))$
    $t \leftarrow t + \Delta t$
end while
Output Equilibrium point

make a comparison between the algorithm of obtaining the equilibrium point of CHN by the method of Euler [16] and the algorithm proposed in this work.

The two tables (2.5, 2.6) represent the results of the two methods used, Euler and Crank-Nicolson. We have taken the 20 instances that are given in the TAP challenge [14]. The results of this experiment are given as a function of: optimal value (OV) in the TAP challenge [14], best feasible solution obtained (BFS), average of the feasible solutions obtained (Mean), sum of the iterations of the feasible solutions (SI) and execution time in seconds (ET).

Table 2.5: The result of each TAP by the CHN-Euler

Instances	OV	BFS	Mean	Mode	SI	ET
tassnu_10_3_1	-719	-719	-319.55	-253	27524	19.83
tassnu_10_3_2	-790	-790	304.34	129	27277	37.97
tassnu_10_3_3	-624	-614	-256.73	-362	28993	32.21
tassnu_10_3_4	-734	-709	-349.06	-524	30755	44.01
tassnu_10_3_5	-871	-825	-357.68	-168	31215	26.87
tassnu_10_3_6	-677	-677	-196.02	-236	30343	36.07
tassnu_10_3_7	-613	-613	-214.35	145	29111	41.56
tassnu_10_3_8	-495	-479	-54.01	323	31611	26.15
tassnu_10_3_9	-750	-730	-310.64	-34	27809	22.45
tassnu_10_3_10	-486	-452	-81.63	-129	28640	20.31
tassnu_15_5_1	-1985	-1905	-844.38	-311	54827	100.9
tassnu_15_5_2	-1568	-1484	-556.56	43	53883	141.48
tassnu_15_5_3	-1892	-1565	-874.74	-259	49768	169.93
tassnu_10_5_4	-1806	-1539	-711.81	-156	55381	112.15
tassnu_15_5_5	-1881	-1796	-828.1	132	53673	103.36
tassnu_15_5_6	-1950	-1822	-881.98	-141	53139	102.36
tassnu_15_5_7	-1893	-1817	-720.73	-213	54866	143.94
tassnu_15_5_8	-1733	-1694	-616.27	-174	51382	101.99
tassnu_15_5_9	-1798	-1512	-791.53	-531	56952	131.43
tassnu_15_5_10	-1763	-1481	-774.23	-1118	54644	134.98
Mean	-1251.4	-1161.15	-502.22	-192.3	41589.65	77.49

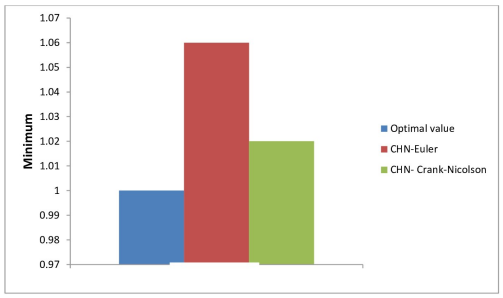
Table 2.6: The result of each TAP by the CHN-Crank-Nicolson

Instances	OV	BFS	Mean	Mode	SI	ET
tassnu_10_3_1	-719	-719	-82.11	-620	20490	26.3
tassnu_10_3_2	-790	-790	72.45	753	15974	226.54
tassnu_10_3_3	-624	-624	-154.52	-354	23284	431.25
tassnu_10_3_4	-734	-671	-285.45	-567	25194	451.66
tassnu_10_3_5	-871	-871	-353.33	-704	22721	109.71
tassnu_10_3_6	-677	-677	-241.15	-677	21451	112.41
tassnu_10_3_7	-613	-613	-200.12	-613	29241	104.74
tassnu_10_3_8	-495	-493	82.32	-20	22000	126.41
tassnu_10_3_9	-750	-730	-116.26	-669	21697	125.13
tassnu_10_3_10	-486	-486	119.54	461	23845	133.43
tassnu_15_5_1	-1985	-1985	-1300.83	-1250	48466	151
tassnu_15_5_2	-1568	-1515	-237.73	381	17671	667.65
tassnu_15_5_3	-1892	-1781	-522.29	-149	18103	626.77
tassnu_15_5_4	-1806	-1659	-290.38	214	18057	689.97
tassnu_15_5_5	-1881	-1796	-88.51	-3	16595	567.19
tassnu_15_5_6	-1950	-1922	-647.46	-466	15518	564.38
tassnu_15_5_7	-1893	-1818	-42.01	-340	15923	566.6
tassnu_15_5_8	-1733	-1711	-154.128	376	21276	731.78
tassnu_15_5_9	-1798	-1560	-308.11	416	24555	835.37
tassnu_15_5_10	-1763	-1762	-474.33	213	16508	461.92
Mean	-1251.4	-1209.15	-263.02	-180.9	21928.45	385.51

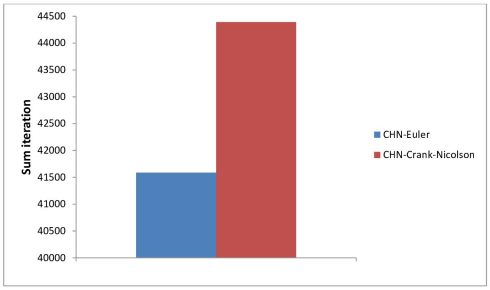
From the results presented in the previous subsection, we can group the different results in the following three Figures (2.6a, 2.6b, 2.6c). The measure we have adopted in the Figure is given by the performance ratio Eq.(2.18).

From the results obtained in the experimental phase represented in the two tables (5.5,5.6) represent respectively the application of algorithm to obtain the equilibrium point of CHN by the two methods of solving the differential equation Euler and Crank-Nicolson. The three Figures (2.6a,2.6b,2.6c) generalize the results of the two tables above, it seems that the minimum obtained for each instance by our algorithm is much better than the classical CHN whose percentage of variance is estimated to 40% and then the percentage of obtaining the obtimal point for the CHN-Crank-Nicolson method is 40% and for CHN-Euler represents a percentage of 20%. On the other hand for the sum of the iterations, our method requires more iterations which is normal whose percentage of variance is estimated at 28% for the simple reason that the Crank-Nicolson method combines both the explicit and implicit Euler method which also leads to a higher execution time.

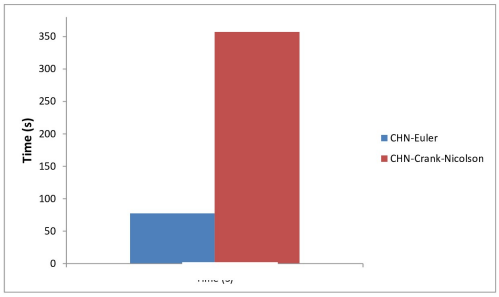
There are many criteria to measure the performance of a differential equation solution method. In this study, we rely on the comparison of the two methods (Euler and Crank-Nicolson) on three important elements: first, the quality of the CHN output solution; second, the number of iterations required for each execution; and finally, the execution time required. The Crank-Nicolson method is a one-step method used in this work to calculate the equilibrium point of the CHN in order to confirm the effectiveness of this method by a comparative and analytical study with the classical CHN. Therefore, we apply this method to an additional assignment problem. Through the results presented in this work, it has been found that the use of the Crank-Nicolson method is better and more efficient compared to the classical CHN method which depends on the method. This is what we are looking for in our research, which is



(a) Average ratio of CHN-Euler, CHN-CN



(b) Sum of iteration of CHN by the two methods Euler and CN



(c) Average execution time of each method

Figure 2.6: The variation of metrics for two methods Euler and CN

the development of Hopfield network to improve the results of different uses of Hopfield network in different areas.

2.6 Conclusion

Artificial neural networks play an important role in scientific and engineering applications. In this chapter, the second section describes the basic general aspects of Hopfield recurrent neural networks. In the first part, a brief presentation of the classes of Hopfield neural networks is given. In the second part, we discuss the analysis of the stability of CHNs. In the third part, we discuss the algorithm adapted to obtain the equilibrium points of the CHN. In the third section, we presented the dynamics involved in solving combinatorial optimization problems using the continuous Hopfield network. In addition, we discussed the challenge associated with parameter tuning within the CHN. Hopfield recurrent neural networks, a major focus of this thesis, are discussed in this section. First, a comparative analysis of three approaches to solving COPs was conducted: metaheuristic methods (specifically, ant colony optimization), exact methods (using CPLEX), and the Hopfield network. The experimental results show that CPLEX consistently achieves an exact optimal solution, albeit with a long computation time. On the other hand, Hopfield networks show the ability to provide a near-optimal solution in a shorter time. The dynamics of CHN are described by a set of challenging differential equations, making analytical solutions elusive. Consequently, researchers resort to the Euler-Cauchy method to compute the equilibrium point of CHN. Unfortunately, this approach encounters various problems, including suboptimal decision quality for large steps and sensitivity to both the slope function parameters and initial conditions. In the fifth section, we used the Crank-Nicolson method as a numerical alternative. This method introduces an intermediate step that improves the accuracy of the Euler-Cauchy method. The experimental results underline that the (CHN+Crank-Nicolson)

method outperforms the (CHN+Euler-Cauchy) method, demonstrating superior performance.

Tuning the hyperparameters of the Hopfield network is an essential component in the fields of artificial intelligence and machine learning. Hopfield networks, inspired by the functioning of the human brain, are widely used to solve a variety of problems, including combinatorial optimization. However, despite their potential, these networks are sensitive to the configuration of their hyperparameters. The optimal choice of these hyperparameters can greatly influence the performance and ability of the network to converge on accurate and stable solutions. Thus, in the following chapters, we set out to explore in depth the mechanisms for tuning the hyperparameters of the Hopfield network. Our goal is to develop innovative methods and techniques to automatically and efficiently determine the optimal values of the hyperparameters in order to improve the performance and robustness of Hopfield networks in different application contexts. On the other hand, we aim to analyze the effect of the choice of these hyperparameters on the performance of the network.

Chapter 3

Tuning and adaptation of HNN to combinatorial optimization problems

3.1 Introduction

The Continuous Hopfield Network (CHN) is a recurrent neural network model introduced by John Hopfield in 1982 [19, 131]. It is composed of a layer of fully connected neurons and has been widely applied in numerous domains, including optimization problems [103, 5, 21, 123, 116, 124, 125, 99, 31], operations research [104, 119, 22], image restoration [100, 105, 113, 114], economics [106, 108], electronic circuit design [109, 115, 117], computer science, and other applications. The CHN is based on a differential equation starting from an initial state and evolving over time by minimizing the associated energy function until it reaches an equilibrium point [5, 66, 110, 97]. The equilibrium point of the CHN is determined by the energy function, that represents the objective function being optimized. Previous research in this field has marked a significant breakthrough [131] thanks to Hopfield and Tank's contributions in combinatorial optimization problems (COPs), particularly through the introduction of the CHN. It is therefore an attractive choice for solving COPs due to its numerous advantages. There are many privileges associated with using this network. To begin with, it converges towards an optimal solution for a given COP (using appropriate parameter settings and choosing suitable CHN hyperparameters) [21, 16]. Another advantage is its ability to deal with complex problems [21]. Additionally, the CHN works in parallel, allowing it to simultaneously deal with multiple aspects of the problem. On top of that, this powerful network has a convenient solving time, especially for large problems [31]. Despite all these advantages, there are still some difficulties associated with CHN. The primary challenges that render CHN unsuitable for conflict resolution are evident in the parameterization of the network and the initial selection of hyperparameters.

This chapter aims to ensure the convergence of valid solutions for combinatorial optimization problems. Our approach proposes a new CHN technique to treat network stability by enhancing its parameter settings and choosing appropriate hyperparameters. We have successfully applied our approach to solve the weighted constraint satisfaction problem (WCSP), task assignment problem (TAP), and traveling salesman problem (TSP). The validation of our approach enables us to conduct a comprehensive experimental and analytical study with two main objectives. Firstly, we aim to demonstrate the superiority of our parameters compared to those described in existing references for the handled problems. Secondly, we aim to conduct an empirical

study to select the optimal hyperparameters and understand their impact on the convergence of CHN. The results obtained from our studies demonstrate improvements in the quality of solutions for the handled problems.

The rest of this chapter is organized as follows: Section 3.2 describes the Hopfield energy function associated with constrained optimisation problems. In Section 3.3, we propose a new technique for CHN parameter tuning by treating the stability of the CHN. Finally, the Section 3.4 presents the conclusion.

3.2 Hopfield energy function associated with constrained optimization problems

As mentioned earlier, CHNs have gained considerable popularity in the field for solving combinatorial optimization problems (COPs). In this subsection, our main objective is to study and map certain combinatorial optimization problems using CHN.

3.2.1 HEF for TSP

The Traveling Salesman Problem (TSP) is a classic optimization problem in computer science and operations research. It aims to optimize the traversal route among multiple cities, adhering to the stipulation that each city must be visited exactly once and that the path should be of the shortest distance [63]. With the expansion in the problem's dimensions, denoted by the number of cities, the intricacy of the conundrum escalates [5, 31, 6, 66].

Using a Hopfield neural network to solve the TSP involves an interesting approach to optimization. The energy function Eq.(2.17) associated with TSP which must be minimized is given by [29, 5, 66, 6]:

$$E_{TSP}(x,\alpha,\phi,\phi^{\mathrm{C}},\phi^{F},\gamma,\beta) = \frac{\alpha}{2}x^{t}Px + \frac{\phi^{F}}{2}\sum_{i}e_{i,.}(x)^{2} + \frac{\phi^{C}}{2}\sum_{j}e_{.,j}(x)^{2} + \frac{\phi}{2}\sum_{i}\sum_{j}e_{i,.}(x)e_{.,j}(x) + \beta\sum_{i}e_{i,.}(x) + \gamma x^{t}(1-x) \quad (3.1)$$

Where:

- α controls the total length of a tour for the TSP based on the distance matrix P,
- ϕ^{F}, ϕ^{C} and ϕ represents the quadratic formula of the feasible trajectory based on the two TSP constraints: $e_{i,.}(x) \equiv \sum_{j} x_{ij} = 1$ and $e_{.,j}(x) \equiv \sum_{i} x_{ij} = 1$,
- β penalizes the linear terms of the TSP constraints,
- γ ensures the binarity of the variables.

The parameters α, ϕ^{F}, ϕ^{C}, β and γ must be chosen so that the Hopfield network equilibrium point associated with the TSP is realized.

3.2.2 HEF for TAP

The Task Assignment Problem (TAP) is another venerable optimization challenge that involves the allocation of a set of tasks to a collective of agents [60]. The objective of TAP is to reduce the collective sum of execution and communication costs associated with the tasks assigned to the agents [12, 23]. This problem is applicable in various fields, such as project management, work scheduling, and resource allocation.

The energy function Eq.(2.17) of TAP is given by [12]:

$$E_{TAP}(x,\alpha,\phi,\beta,\gamma) = \alpha(\frac{1}{2}x^tCx + e^tx) + \frac{\phi}{2}x^tx + \beta x + \gamma x^t(1-x) \tag{3.2}$$

Where:

- α controls the sum of the execution and communication costs between the tasks according to the communication cost matrix C and the execution cost vector e,
- ϕ penalizes any product of constraints of TAP,
- β penalizes the linear terms of the TAP constraints,
- γ ensures the binarity of the variables.

The parameters α, ϕ, β and γ must be chosen so that the Hopfield network equilibrium point associated with the TAP is realized.

3.2.3 HEF for WCSP

WCSP is an NP-hard problem to determine a value assignment to variables with the minimum cost [10]. The energy function Eq.(2.17) for WCSP [10]:

$$E_{WCSP}(x,\alpha,\Phi,\beta,\gamma) = \alpha(\frac{1}{2}x^tQx + q^tx) + \frac{\Phi}{2}x^tx + \beta x + \gamma x^t(1-x) \tag{3.3}$$

Where:

- α controls the minimum cost of the attribution to each variable of a value in its domain, as a function of the matrix Q and the vector q that present the costs of assigning values to the variables,
- ϕ penalizes any product of constraints of WCSP,
- β penalizes the linear terms of the WCSP constraints,
- γ ensures the binarity of the variables.

The parameters α, ϕ, β and γ must be chosen so that the Hopfield network equilibrium point associated with the WCSP is realized.

3.2.4 HEF for MSP

The Max-Stable Problem (MSP) is a statistical concept related to extreme value theory. It deals with modeling the distributions of extreme values in a set of random variables. Extreme value theory is particularly useful in fields such as finance, environmental science, and engineering, where understanding the distribution of extreme events (e.g., maximum daily rainfall, maximum financial losses) is crucial.

The use of HNN to solve MSP consists in minimizing the energy function Eq.(2.17) [8, 53]:

$$E_{MSP}(x,\alpha,\phi,\gamma) = -\alpha x + \frac{\phi}{2}x^tbx + \gamma x^t(1-x) \tag{3.4}$$

Where:

- α controls a stable set in the graph $G = (V, E)$ of maximal cardinality,
- ϕ penalizes any product of constraints of MSP,

- γ ensures the binarity of the variables.

The parameters α, ϕ and γ must be chosen so that the Hopfield network equilibrium point associated with the MSP is realized.

3.2.5 HEF for GCP

The Graph Coloring Problem (GCP) is an NP-hard problem that consists of finding the smallest number of different colors that can be assigned to nodes such that two adjacent vertices do not have the same color [9]. The energy function Eq.(2.17) for GCP given by [9]:

$$E_{GCP}(x,\alpha,\rho^A,\rho^B,\phi,\beta,\gamma_0,\gamma_1) = \alpha\sum_k(\frac{N}{2}+\frac{2k}{N})x_{0k} + \rho^A h_A(x) + \frac{\rho^B}{2}h_B(x) + \\ \frac{\phi}{2}\sum_i e_i(x)^2 + \beta\sum_i e_i(x) + \gamma_0\sum_k x_{0k}(1-x_{0k}) + \gamma_1\sum_{i,k} x_{ik}(1-x_{ik}) \quad (3.5)$$

Where:

- α controls the lowest number of N different colors that can be assigned to the nodes of a graph,
- ρ_A and ρ_B penalize the two quadratic constraints $h_A \equiv \sum_k(1-x_{0k})\sum_i x_{ik} = 0$ and $h_B \equiv \sum_{k,i,j} x_{ik}b_{ij}x_{jk} = 0$, respectively,
- ϕ penalizes any product of constraints of GCP,
- β penalizes the linear terms $e_i(x) \equiv \sum_k x_{ik} = 1$ of GCP constraints,
- γ_0 and γ_1 penalize the non-extreme values of x_{0k} and x_{ik}, respectively.

The parameters α, ρ_A, ρ_B, ϕ, β, γ_0 and γ_1 must be chosen so that the Hopfield network equilibrium point associated with the GSP is realized.

3.2.6 HEF for JSSP

The Job Shop Scheduling Problem (JSSP) is one of the most common combinatorial optimization problems, widely recognized for its classification as an NP-hard problem [65]. In this problem, a collection of tasks and a set of resources are given, where each task consists of a sequence of activities to be performed in a specified order. The primary objective of JSSP is to derive a solution that minimizes the total processing time, defined as the time required to complete all tasks. JSSP has practical applications in manufacturing processes, where it has a significant impact on production times and associated costs.

Solving the JSSP by the Hopfield network requires the following generalized energy function Eq.(2.17) in algebraic form [65]:

$$E_{JSSP}(x,\alpha,\phi,\beta,\gamma) = \frac{\alpha}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{d_i}\sum_{s=1}^{d_j} q_{irjs}x_{ir}x_{js} + \frac{\phi}{2}\sum_{i=1}^{N}\sum_{r=1}^{d_i}\sum_{s=1}^{d_j} x_{ir}x_{is} + \\ \beta\sum_{i=1}^{N}\sum_{r=1}^{d_i} x_{ir} + \gamma\sum_{i=1}^{N}\sum_{r=1}^{d_i} x_{ir}(1-x_{ir}) \quad (3.6)$$

Where:

- α is the penalty parameter associated with this term represents the contribution of the interaction cost between tasks (x_{ir}) for each pair of tasks i and j executed at times r and s, respectively.
- ϕ this penalty parameter is associated with the second term, which represents the contribution of the cost of interaction between the machines for each task performed at times r and s, respectively.
- β this penalty parameter is associated with the third term, which represents the cost of using the machines for each task i at times r.
- γ this last penalty parameter is associated with the term modeling the cost of violating the sequence constraints for each task i at times r.

The parameters α, β, ϕ and γ must be chosen so that the Hopfield network equilibrium point associated with the JSSP is realized.

3.2.7 HEF for MSP

The Meeting Scheduling Problem (MSP) has received a lot of attention from the operations research community and is a problem formulated as a constraint satisfaction problem. It consists of efficiently organizing meetings between a group of people, taking into account various constraints and objectives [43, 57]. This problem has been solved by the HNN, so that its energy function is defined by [57]:

$$E_{MSP}(x, \alpha, \phi, \beta, \gamma) = \frac{\alpha}{2} x^t Q x + \frac{\phi}{2} x^t x + \beta x + \gamma x^t (1 - x) \tag{3.7}$$

Where:

- α controls the objective function that minimises overlapping meetings, which could reduce the overall quality of scheduled meetings,
- ϕ is a regularization parameter, so that the term associated with this parameter encourages a solution in which a minimum number of meetings is scheduled,
- β penalizes the linear terms of the MSP constraints,
- γ ensures the binarity of the variables.

The parameters α, β, ϕ and γ must be chosen so that the Hopfield network equilibrium point associated with the MSP is realized.

3.2.8 HEF for KNP

Kohonen Network Problem (KNP) architectures allow for the projection of the observations in a low-dimensional space and, at the same time, preserve the notion of neighborhood defined on the set of observations [33]. The appropriate energy function Eq.(2.17) for the KNP [33]:

$$E_{KNP}(x, \alpha, \rho^C, \phi, \beta, \gamma_0, \gamma_1) = \alpha f(x) + \rho^C h_C(x) + \frac{\phi}{2} \sum_i e_i(x)^2 + \beta \sum_i e_i(x) + \gamma_0 \sum_k x_{0k}(1 - x_{0k}) + \gamma_1 \sum_{i,k} x_{ik}(1 - x_{ik}) \tag{3.8}$$

Where:

- α controls the lowest number of different colors that can be assigned to the nodes of a graph,
- ρ_C penalizes the quadratic constraint $h_C \equiv \sum_j (1 - x_{0j}) \sum_i x_{ij} = 0$,
- ϕ penalizes any product of constraints to the Kohonen architecture maps,
- β penalizes the linear terms constraints $e_i(x) \equiv \sum_j x_{ij} = 1$,
- γ_0 and γ_1 penalize the non-extreme values of x_{0k} and x_{ik}, respectively.

The parameters α, ρ_C, ϕ, β, γ_0 and γ_1 must be chosen so that the Hopfield network equilibrium point associated with the KNP is realized.

3.2.9 HEF for PSP

The Portfolio Selection Problem (PSP) is the problem of allocating capital over a number of available assets to minimize risk and maximize return [35]. Comparing the PSP with the energy function equation Eq.(2.17) of the Hopfield model, it can be translated into [34, 35]

$$E_{PSP}(x, \alpha, \phi_1, \phi_2, \beta_1, \beta_2) = \frac{\alpha}{2}[\lambda x^t \sigma x - (1-\lambda)\mu x] + \frac{\phi_1}{2} \sum_{ij} x_i \delta_i \delta_j x_j + \frac{\phi_2}{2} \sum_{ij} x_i x_j + \beta_1 \sum_i x_i + \beta_2 \sum_i x_i \delta_i \quad (3.9)$$

Where:

- α controls the objective function of the PSP based on $\lambda \in [0, 1]$, μ is the return and σ is the risk,
- ϕ_1 and β_1 penalizes any product of constraints to the PSP,
- ϕ_2 and β_2 penalizes the linear terms constraints.

The parameters α, ϕ_1, ϕ_2, β_1, β_2 and γ_1 must be chosen so that the Hopfield network equilibrium point associated with the PSP is realized.

3.2.10 HEF for SPP

The Shortest Path Problem (SPP) is based on the routing problem of finding a path from s (the starting point) to d (the ending point) with a minimum length [107]. Comparing the SPP with the energy function (2.17), it can be translated into [107]:

$$E_{SPP}(x, \alpha_1, \alpha_2, \phi, \mu) = \frac{\alpha_1}{2} Cx + \frac{\alpha_2}{2} \rho x + \gamma x(1-x) + \frac{\phi}{2} \sum_i (\sum_j x_{ij} - \sum_j x_{ji})^2 + \mu(1 - x_{ds}) \quad (3.10)$$

Where:

- α_1 controls the set of arcs minimum total weight C and α_2 prevents non-existent links from being included in the chosen path,
- ϕ controls constraint of SPP,
- γ pushes the state of the neural network to converge to one of the nodes of the hypercube, defined by $x_{ij} \in \{0, 1\}$,

- μ equals 0 when the output of the neuron at location (d, s) sets to 1.

The parameters α_1, ϕ, γ and μ must be chosen so that the Hopfield network equilibrium point associated with the SPP is realized.

3.3 New method for CHN parameters tuning

In order to solve COPs efficiently using CHNs, several important considerations should be taken into account. These essential points can be summarized as follows:

- Avoiding infeasible solutions: In COPs, it is essential to avoid infeasible solutions that violate the given constraints.
- Achieving optimal solutions: In many cases, the focus is on finding solutions that are close to optimal solutions rather than the absolute global optimum.
- Fixing the values of the penalty hyperparameters: The penalty hyperparameters play a crucial role in ensuring the convergence of the CHN. Then, it is important to select the appropriate values for these hyperparameters to achieve the good results.

The objective of this section is to address the first and second points mentioned earlier. To achieve these objectives, we propose fine-tuning the CHN parameters.

3.3.1 Parameter tuning in CHN applied to COPs

In this section, we leverage Lemma (2.1) and Theorems (2, 3) to search the parameter settings for weighted constraint satisfaction problems, task assignment problem and traveling salesman problem.

1. **Parameter tuning in CHN applied to WCSP**

   To use Theorems 2 and 3 for the WCSP, we initiate the process by calculating the partial derivatives of the energy function Eq.(3.3) [10]:

$$\frac{\partial E(x)}{\partial x_{ir}} = \alpha(\sum_{j=1}^{n}\sum_{x=1}^{d_i} q_{irjx}x_{jx} + q_{ir}) + \Phi\sum_{x=1}^{d_i} x_{ix} + \beta + \gamma\,(1 - 2x_{ir}) \tag{3.11}$$

   Theorem 4. *In WCSP, all infeasible solutions obtained by CHN are unstable if:*

$$\alpha\,(Q_{\min} + q_{\min}) > \Phi + \beta - \gamma$$

$$\alpha\,(Q_{\max} + q_{\max}) < \Phi - \beta - \gamma$$

   Where, $Q_{\min} = \operatorname{Min}\{q_{irjs}\}, q_{\min} = \operatorname{Min}\{q_{ir}\}$ *and* $Q_{\max} = \operatorname{Max}\{q_{irjs}\}, q_{\max} = \operatorname{Max}\{q_{ir}\}$

   Proof. For any infeasible solution $x = (x_{ir})$ in the Hamming hypercube, one of the following constraints applies:

   (a) $\exists i \in \{1, \ldots, n\}$, $\sum_{r=1}^{d_i} x_{ir} \geq 1$

   (b) $\exists i \in \{1, \ldots, n\}$, $\sum_{r=1}^{d_i} x_{ir} \leq -1$

   Regarding the first point a), and $x_{ir} = 1$, we have $\frac{\partial E(x)}{\partial x_{ir}} \geq \alpha \sum_{j=1}^{n}\sum_{x=1}^{d_1} q_{irjx} + \alpha q_{ir} + \Phi + \beta - \gamma$. If, $\alpha\,(Q_{\min} + q_{\min}) > \Phi + \beta - \gamma$, we have $\frac{\partial E(x)}{\partial x_{ir}} \geq 0$, so from (2.14), x is unstable.

- For the second point b), and $x_{ir} = 0$, we have $\frac{\partial E(x)}{\partial x_{ir}} \leq -\alpha \sum_{j=1}^{n} \sum_{x=1}^{d_1} q_{irjx} + \alpha q_{ir} - \Phi + \beta + \gamma$
If, $\alpha (Q_{\max} + q_{\max}) < \Phi - \beta - \gamma$, we have $\frac{\partial E(x)}{\partial x_{ir}} \leq 0$, so from (2.14), x is unstable. □

Theorem 5. *In WCSP, a CHN feasible solution $x = (x_{ik})$ is*

- *stable if:* $\gamma > \alpha (Q_{\max} + q_{\max}) + \Phi + \beta$
- *unstable if:* $\gamma < \alpha (Q_{\max} + q_{\max}) + \Phi + \beta$

Proof. For any feasible solution $x = (x_{ir})$, we have:

$$\frac{\partial E}{\partial x_{ir}}(x) = \begin{cases} \alpha \sum_{j=1}^{n} \sum_{x=1}^{d_1} q_{irjx} + \alpha q_{ir} + \Phi + \beta + \gamma \, (x_{ir} = 0) \\ \alpha \sum_{j=1}^{n} \sum_{x=1}^{d_1} q_{irjx} + \alpha q_{ir} + \Phi + \beta - \gamma \, (x_{ir} = 1) \end{cases}$$

- For $x_{ir} = 0$, we have $\frac{\partial E(x)}{\partial x_{ir}} > 0$.
- For $x_{ir} = 1$, if $\gamma > \alpha (Q_{\max} + q_{\max}) + \Phi + \beta$ we have $\frac{\partial E(x)}{\partial x_{ir}} < 0$.
So from (2.13), x is stable.
On the other hand, if $\gamma < \alpha (Q_{\max} + q_{\max}) + \Phi + \beta$, we have x is unstable. □

Based on the Theorems (4) and (5), we propose the following parameter settings for WCSP:

$$\begin{cases} \alpha(Q_{\max} + q_{\max}) + \varepsilon = \phi - \beta - \gamma \\ \alpha(Q_{\min} + q_{\min}) = \phi + \beta - \gamma + \varepsilon \\ \gamma = \alpha(Q_{\max} + q_{\max}) + \phi + \beta \end{cases} \tag{3.12}$$

2. **Parameter tuning in CHN applied to TAP**

In this paragraph, we adopt a similar approach for TAP. The parameter setting procedure relies on the calculation of partial derivatives of the generalized energy function Eq.(3.2) represented by:

$$\frac{\partial E(x)}{\partial x_{ik}} = \alpha(\sum_{j=1}^{N} \sum_{l=1}^{M} c_{ikjl} x_{jl} + e_{ik}) + \Phi \sum_{1=1}^{M} x_{il} + \beta + \gamma \, (1 - 2x_{ik}) \tag{3.13}$$

Theorem 6. *In TAP, all infeasible solutions obtained by CHN are unstable if:*

$$\alpha (C_{\min} + e_{\min}) > \Phi + \beta - \gamma$$

$$\alpha (C_{\max} + e_{\max}) < \Phi - \beta - \gamma$$

Where $C_{\min} = \text{Min} \left\{ c_{ikjl} / (i,j) \in \{1, \ldots, N\}^2 \text{ and } (k,l) \in \{1, \ldots, M\}^2 \right\}$ $e_{\max} = \text{Max} \{ e_{ik} / i \in \{1, \ldots, N\}$ *and* $k \in \{1, \ldots, M\} \}$

Proof. For any infeasible solution $x = (x_{ik})$, a solution in the Hamming hypercube, one of the following constraints applies:

(a) $\exists i \in 1, \ldots, N, \sum_{k=1}^{M} x_{ik} \geq 1$
(b) $\exists i \in 1, \ldots, N, \sum_{k=1}^{M} x_{ik} \leq -1$

- For the constraint a) and $x_{ik} = 1$, we have $\frac{\partial E(x)}{\partial x_{ik}} \geq \alpha \sum_{j=1}^{N} \sum_{l=1}^{M} c_{ikjl} + \alpha e_{ik} + \Phi + \beta - \gamma$.
If $\alpha (C_{\min} + e_{\min}) > \Phi + \beta - \gamma$, we have $\frac{\partial E(x)}{\partial x_{ik}} \geq 0$, so from (2.14), x is unstable.
- For the constraint b) and $x_{ik} = 0$, we have $\frac{\partial E(x)}{\partial x_{ik}} \leq -\alpha \sum_{j=1}^{N} \sum_{l=1}^{M} c_{ikjl} + \alpha e_{ik} - \Phi + \beta + \gamma$.
If $\alpha (C_{\max} + e_{\max}) < \Phi - \beta - \gamma$, we have $\frac{\partial E(x)}{\partial x_{ik}} \leq 0$, so from (2.14), x is unstable. □

Theorem 7. *In TAP, a CHN feasible solution $x = (x_{ik})$ is*

- *stable if:* $\gamma > \alpha\,(C_{\max} + e_{\max}) + \Phi + \beta$
- *unstable if:* $\gamma < \alpha\,(C_{\max} + e_{\max}) + \Phi + \beta$

Proof. For any feasible solution $x = (x_{ik})$, we have :

$$\frac{\partial E}{\partial x_{ik}}(x) = \begin{cases} \alpha \sum_{j=1}^{N} \sum_{l=1}^{M} c_{ikjl} + \alpha e_{ik} + \Phi + \beta + \gamma\,(x_{ik} = 0) \\ \alpha \sum_{j=1}^{N} \sum_{l=1}^{M} c_{ikjl} + \alpha e_{ik} + \Phi + \beta - \gamma\,(x_{ik} = 1) \end{cases}$$

- For $x_{ik} = 0$, we have $\frac{\partial E(x)}{\partial x_{ik}} > 0$.
- For $x_{ik} = 1$, if $\gamma > \alpha\,(C_{\max} + e_{\max}) + \Phi + \beta$ we have $\frac{\partial E(x)}{\partial x_{ik}} < 0$. So from (2.13), x is stable.
On the other hand, if $\gamma < \alpha\,(C_{\max} + e_{\max}) + \Phi + \beta$, we have x is unstable. □

In a similar way for the TAP, we can establish a system of parameters by using the Theorems (6) and (7).

$$\begin{cases} \alpha(C_{\max} + e_{\max}) = \phi + \beta - \gamma + \varepsilon \\ \alpha(C_{\min} + e_{\min}) + \varepsilon = \phi - \beta - \gamma \\ \gamma = \alpha(C_{\max} + e_{\max}) + \phi + \beta \end{cases} \tag{3.14}$$

3. Parameter tuning in CHN applied to TSP

In the case of the TSP, the derivative of its energy function Eq.(3.1) [29]:
$\frac{\partial E(x)}{\partial x_{ij}} = \alpha \sum_k d_{ik} \left(x_{k(j-1)} + x_{k(j+1)}\right) + \Phi^F\,(e_{j,\cdot}(x) - 1) + \Phi^C\,(e_{\cdot,i}(x) - 1) + \gamma\,(1 - 2x_{ij})$

Theorem 8. *In TSP, all infeasible solutions obtained by CHN are unstable if:*

$$\Phi^F > \gamma + \alpha(N-1)d_u$$

$$\Phi^C > max\{\gamma, \gamma + \alpha(N-1)d_u\}$$

Where $d_u = \max_{ij} d_{ij}$

Proof. For any infeasible solution $x = (x_{ij})$ a solution in the Hamming hypercube, one of the following constraints applies:

(a) $\exists j \in 1, \ldots, N, e_{j,\cdot}(x) - 1 \geq 1$
(b) $\exists i \in 1, \ldots, N, e_{\cdot,i}(x) - 1 \geq 1$
(c) $\exists j \in 1, \ldots, N, e_{j,\cdot}(x) - 1 \leq -1$
(d) $\exists i \in 1, \ldots, N, e_{\cdot,i}(x) - 1 \leq -1$

- For the constraint a) and $x_{ij} = 1$, we have $\frac{\partial E(x)}{\partial x_{ij}} \geq \Phi^F - \gamma$. If $\Phi^F \geq \gamma$, we have $\frac{\partial E(x)}{\partial x_{ij}} \geq 0$, so from (2.14), x is unstable.
- For the constraint b) and $x_{ij} = 1$, we have $\frac{\partial E(x)}{\partial x_{ij}} \geq \Phi^c - \gamma$. If $\Phi^C \geq \gamma$, we have $\frac{\partial E(x)}{\partial x_{ij}} \geq 0$, so from (2.14), x is unstable.
- For the constraint c) and $x_{ij} = 0$, we have $\frac{\partial E(x)}{\partial x_{ij}} \leq -\Phi^F + \gamma + \alpha(N-1)d_u$. If $\Phi^F > \gamma + \alpha(N-1)d_u$, we have $\frac{\partial E(x)}{\partial x_{ij}} < 0$, so from (2.14), x is unstable.
- For the constraint d) and $x_{ij} = 0$, we have $\frac{\partial E(x)}{\partial x_{ij}} \leq -\Phi^C + \gamma + \alpha(N-1)d_u$. If $\Phi^C > \gamma + \alpha(N-1)d_u$, we have $\frac{\partial E(x)}{\partial x_{ij}} < 0$, so from (2.14), x is unstable. □

We can conclude that the instability conditions depend on two parameters Φ^F and Φ^C. For the stability and instability of feasible solutions, we can also derive the following theorem.

Theorem 9. *In TSP, a CHN feasible solution $x = (x_{ik})$ is*

- *stable if:* $\gamma > 2\alpha \max_{ij,jk} (d_{ij} + d_{jk})$
- *unstable if:* $\gamma < 2\alpha \max_{ij,jk} (d_{ij} + d_{jk})$

Proof. For any feasible solution $x = (x_{ij})$, we have :

$$\frac{\partial E}{\partial x_{ij}}(x) = \begin{cases} \gamma + \alpha (d_{ij} + d_{jk}) (x_{ij} = 0) \\ -\gamma + \alpha (d_{ij} + d_{jk}) (x_{ij} = 1) \end{cases}$$

- For $x_{ij} = 0$, we have $\frac{\partial E(x)}{\partial x_{ij}} > 0$.
- For $x_{ij} = 1$, if $\gamma > \alpha \max_{ij,jk} (d_{ij} + d_{jk})$, we have $\frac{\partial E(x)}{\partial x_{ij}} < 0$. So from (2.13), x is stable. On the other hand, if $\gamma < \alpha \max_{ij,jk} (d_{ij} + d_{jk})$, we have x is unstable. □

Based on the aforementioned Theorems (8) and (9), we propose the following parameter settings for TSP:

$$\begin{cases} \phi^F = \gamma + \alpha(N-1)d_u + \varepsilon \\ \phi^C = max\{\gamma, \gamma + \alpha(N-1)d_u\} + \varepsilon \\ \gamma = 4\alpha d_u \end{cases} \tag{3.15}$$

Remark 1. *The instability conditions of infeasible solutions can be used as exclusion criteria. By identifying the conditions under which a solution is unstable, we can avoid unnecessary iterations or costly calculations for solutions that cannot be realized. This saves time and resources by focusing only on feasible solutions that have a chance of converging into stable states. Additionally, it allows for adjustments of problem parameters to make solutions feasible.*

The parameter settings described in the theorems (4 to 9) are examined using the stability analysis method. These settings include carefully designed constraints aimed at facilitating convergence towards feasible solutions and preventing infeasible solutions. To solve the systems (3.12,3.14,3.15) of parameter settings, the process requires setting the values of certain hyperparameters, such as $(\alpha, \varepsilon, \phi, ...)$, which vary from one problem to another. In this respect, we have adopted the algorithm introduced in [16], which requires the selection of an initial state, also known as the point. Thus, the hyperparameters and the initial state of the CHN play a key role in fine-tuning the network and determining its behavior and convergence properties.

3.3.2 Experimental results and discussion

In this section, we validate and compare the parameter settings proposed in theorems (4 to 9) with those proposed in other references. Furthermore, we conduct an analytical and comparative study to examine the influence of hyperparameter selection (α, ε) and the choice of starting point on the convergence of the network and the quantity of solutions. For these purposes, we have organized our study into two main plans:

- In the first plan, we use a specific combination of the hyperparameters (α, ε) and a starting point to assess and compare the performance of our parameter settings proposed in theorems (4 to 9) with those proposed in other references.

Table 3.1: Performance evaluation of our parameter settings on 20 WCSP instances

Instance name	OS [94] [1]	NFS [2]	BS [3]	ABS[4]	MBS[5]	SIN [6]
4wqueens	0	8887	0	0,27	0	222429
spot5-8	2	9998	10	10,03	10	475378
8queens	0	2892	0	2,36	1	2541126
langford-2-4	0	9376	0	1,55	1	2658537
8wqueens	2	1466	4	68,65	11	2709990
queen5-5-3	29	8	62	138,91	216	3660552
geom40-2	22	0	79	79,00	79	3741221
geom30a-3	11	1	87	87,00	87	2651412
queen5-5-4	12	7	73	147,30	112	3697844
zebre-ext	0	8257	1	10,52	7	1278973
geom30a-4	4	99	4	13,07	5	1947780
geom40-3	7	56	7	9,34	10	1629192
myciel5g-3	16	337	307	526,98	575	572800
geom40-4	3	1683	3	6,13	4	1771321
geom30a-6	0	4439	0	1,89	1	3183044
myciel5g-4	4	6	438	753,18	660	529999
geom40-5	1	7273	1	1,48	1	3893209
myciel5g-5	1	10	1	13,31	3	933881
16wqueens	2	2475	3	105,91	37	2473519
myciel5g-6	0	43	0	99,31	0	1331227

[a]Optimal Solution
[b]Number of Feasible Solutions
[c]Best Solution founded
[d]Average of the Best Solutions founded
[e]Mode of the the Best Solutions founded
[f]Sum of the Iterations Number

- In the second plan, we study the impact of the hyperparameters (α, ε) selection and the starting point of our parameter settings on the performance of the network. This second plan allows us to obtain a more comprehensive understanding of the hyperparameters selection space.

In Tables (3.1 to 3.3), we evaluate WCSP, TAP, and TSP to validate the parameters presented in Theorems (4 to 9). We fixed the values of the hyperparameters for each problem by: ($\alpha = \frac{1}{2}$, $\varepsilon = 10^{-4}$) for WCSP, (Incremental value of α, $\varepsilon = 10^{-4}$) for TAP, and ($\alpha = \frac{1}{2}$, $\varepsilon = 10^{-4}$) for TSP. For the starting point, we followed the approach suggested in [11] for all three problems. This experiment used specific examples drawn from the existing literature [94] for WCSP [96] for TAP, and [13] for TSP.

In Tables (3.4 to 3.6), we conduct a performance comparison between our parameter settings and other parameter settings were adopted from various references related to the handled problems. This comparative study is based on three evaluation metric:

$$PBV = \frac{Number\,of\,BS}{Number\,of\,instances} \tag{3.16}$$

$$FFR = \frac{\sum(BS - OS)}{\sum BS} \tag{3.17}$$

AIN: the average of sum of the iterations number.
Two randomly generated combinations of the specific CHN hyperparameters are used.

The results presented in the tables serve to confirm the effectiveness of our parameter settings proposed in Section 3.3 compared with the other parameters for the three combinatorial

Table 3.2: Performance evaluation of our parameter settings on 20 TAP instance

Instance name	OS [96]	NFS	BS	ABS	MBS	SIN
tassnu_10_3_1	-719	8134	-719	-504,74	-659	764083
tassnu_10_3_2	-790	8425	-790	-490,00	-611	780301
tassnu_10_3_3	-624	7867	-624	-332,70	-362	714981
tassnu_10_3_4	-734	8186	-734	-454,56	-603	751908
tassnu_10_3_5	-871	7743	-801	-571,09	-775	751835
tassnu_10_3_6	-677	8908	-677	-336,84	-376	862569
tassnu_10_3_7	-613	8651	-613	-398,18	-481	821542
tassnu_10_3_8	-495	9963	-495	-171,72	-287	974442
tassnu_10_3_9	-750	8446	-750	-495,62	-669	727686
tassnu_10_3_10	-486	8616	-486	-174,16	-161	805300
tassnu_15_5_1	-1985	9181	-1985	-943,64	-1323	866612
tassnu_15_5_2	-1568	9579	-1389	-728,83	-911	971735
tassnu_15_5_3	-1892	9427	-1565	-1000,79	-1194	909723
tassnu_15_5_4	-1806	9513	-1539	-767,23	-819	928863
tassnu_15_5_5	-1881	9416	-1881	-1177,47	-1382	922772
tassnu 15_5_6	-1950	9515	-1822	-1055,78	-1225	943855
tassnu_15_5_7	-1893	9432	-1817	-1040,90	-1186	958541
tassnu_15_5_8	-1733	9463	-1698	-766,04	-883	921775
tassnu_15_5_9	-1798	9387	-1612	-761,70	-927	949422
tassnu_15_5_10	-1763	9508	-1681	-850,44	-891	936690

Table 3.3: Performance evaluation of our parameter settings on 11 TSP instance

Instance name	OS [13]	NFS	BS	ABS	MBS	SIN
Buma14	3323	362	3323	5682.33	6469	23221
Ulysses16	6859	424	6859	9739,78	10979	51317
Gr17	2085	517	2085	3878,1	4092	11087
Gr21	2707	398	4398	5928,33	4886	18178
Ulysses22	7013	412	9614	12483,14	12983	12798
Gr24	1272	387	2129	2264,16	2262	17199
Fri26	937	514	1407	1958,33	1930	29052
Bays29	2020	318	2974	4282,4	3988	23588
Dantzig42	699	432	1364	2355,63	2083	26396
Att48	10628	302	31855	34222.16	31855	92565
Eil76	538	320	538	1963,7	1969	98487

Table 3.4: Performance of our parameter settings approach and those proposed in [10] for WCSP

Hyperparatemrs values	Parameter settings	PBV	FFR	AIN
$\alpha = \frac{1}{2}$, $\varepsilon = 10^{-4}$,	Our parameter settings	50%	87%	41427
starting point proposed in [11]	Parameters proposed in [10]	50%	89%	41893
$\alpha = \frac{1}{N}$, $\varepsilon = 10^{-3}$,	Our parameter settings	51%	89%	41840
starting point proposed in [10]	Parameters proposed in [10]	50%	88%	42004

Table 3.5: Performance of our parameter settings approach and those proposed in [12] for TAP

Hyperparatemrs values	Parameter settings	PBV	FFR	AIN
$\alpha \in [0, 1]$, $\varepsilon = 10^{-4}$,	Our parameter settings	20%	14%	16501
starting point proposed in [11]	Parameters proposed in [12]	18%	11%	16742
$\alpha = \frac{1}{2}$, $\varepsilon = 10^{-3}$,	Our parameter settings	17%	6%	1860
starting point proposed in [12]	Parameters proposed in [12]	15%	5%	1874.8

Table 3.6: Performance of our parameter settings approach and those proposed in [29] for TSP

Hyperparatemrs values	Parameter settings	PBV	FFR	AIN
$\alpha = \frac{1}{2}$, $\varepsilon = 10^{-4}$,	Our parameter settings	1%	38%	400
starting point proposed in [11]	Parameters proposed in [29]	0%	36%	387
$\alpha = \frac{1}{d_u{}^1}$, $\varepsilon = 10^{-8}$,	Our parameter settings	2 %	39%	367
starting point proposed in [29]	Parameters proposed in [29]	0%	39%	352

[a]The maximum value of the distance matrix d

optimization problems used to make the comparison.

On the other hand, it is important to note that, while a specific combination of hyperparameters can yield favorable results for a particular problem, it does not guarantee similar success for a different problem. For instance, when considering the combination ($\alpha = \frac{1}{2}$, $\varepsilon = 10^{-4}$, and starting point proposed in [11]), we achieved satisfactory results for WCSP in terms of PBV and FFR (Table 3.4). However, using the same combination of hyperparameters, we observed less efficient results for TSP (Table 3.6). Moreover, different combinations of hyperparameters can produce varying outcomes for the same problem. For example, in TAP presented in Table (3.2) instances such as tassnu_10_3_1, tassnu_10_3_2 ,tassnu_10_3_3, etc., the best solution founded (BS) is equal to the optimal solution (OS). However, in instances like tassnu_15_5_3, tassnu_15_5_4, etc., the BS differs significantly from the OS. This emphasizes the significance of studying the impact of hyperparameter selection on the quality of solutions obtained for each specific problem instance.

The results of the second plan, as shown in Tables (3.7 to 3.9), involve exploring different combinations of hyperparameters (α, ε) and the starting point of our parameter settings proposed in Theorems (4 to 9). To provide a clear understanding and to visually represent the results presented in Tables (3.7 to 3.9), Figures (3.1 to 3.3) depict the outcomes obtained for α, ϵ, and the starting point using NBS, SIN, and SBV for FS. The main objective of this plan is to investigate the impact of the choice of these hyperparameters on the quality of the solutions to the handled problems.

Based on the results presented in Table 3.7 and the corresponding Figures (3.1 to 3.3) for WCSP, the optimal selection of α, ε and the starting point to achieve the best performance in terms of NBS, NFS, and SBV for FS are as follows: $\alpha \in \{\frac{1}{N}, [0,1]\}$, $\varepsilon \in \{10^{-4}, 10^{-5}\}$, and the starting point in $[0,1]$. For the SIN, it is recommended to set $\alpha \in [0,1]$, $\varepsilon = 0.1$, and use the starting point proposed in [11].

Regarding TAP, the results presented in Table 3.8 and the Figures (3.1 to 3.3) indicate that choosing $\alpha \in [0,1]$ and using the logarithmic function leads to a higher NBS and a higher sum of feasible solutions compared to other options. Additionally, ε has a significant impact on the quality of solutions. Specifically, when $\varepsilon \in [0.1, 10^{-3}]$, the solution quality decreases. On the other hand, choosing a starting point in $[0,1]$ yields the highest SBV and NFS metrics, while the starting point proposed in [29] produces the highest NBS metric. For the SIN, it is recommended to set $\alpha = Logfunction, \varepsilon = 0.1$, and use the starting point = 0.5.

The results obtained for TSP, presented in Table 3.9 and the Figures (3.1 to 3.3), indicate that selecting $\alpha \in \{\frac{1}{2}, [0,1]\}$ yields the best performance, considering metrics such as NBS, NFS, SBV for FS, and SIN. Furthermore, we observe that smaller values of ε correspond to lower solution quality for all the mentioned metrics. Additionally, choosing a starting point in $[0,1]$, as proposed in [11, 29], is the most appropriate choice to achieve optimal results.

Based on the results of the handled problems, this empirical study gives us a clear and comprehensive view of the impact of each hyperparameter selection (α, ε) and the starting

Table 3.7: Obtained results using different combinations of hyperparameters for WCSP

Hyper-parameters	Values	Mean			
		NBS [1]	NFS [2]	SBV for FS[3]	SIN [4]
α	1/2	490,64	169,11	731,93	83338,78
	1/N	548,56	166,39	1566,22	82133,25
	Continuously decreased in [0,1]	506,47	191,36	857,21	79947,17
	Randomly generated in [0,1]	495,61	171,64	615,78	82615,22
	Log function	490,86	168,25	610,23	86305,53
ε	0,1	0,73	19,77	136,34	1686,60
	0,01	142,90	102,40	529,08	33129,57
	10^{-3}	695,00	242,37	2260,33	97149,43
	10^{-4}	731,00	229,80	864,55	112818,47
	10^{-5}	734,23	222,40	851,14	121185,90
	10^{-6}	734,70	223,37	522,04	131237,97
starting point	0	678,17	1,47	2,07	107042,63
	0.5	707,50	4,43	4,48	26191,70
	$[0, 1]$	57,43	605,77	4615,71	106694,73
	proposed in [11]	687,32	4,68	6,38	18486,68
	proposed in [29]	670,57	3,70	7,17	108366,20

[a]Number of best solutions found in 1000 algorithm runs.
[b]Number of feasible solutions.
[c]Sum of the values of the best solutions found for feasible solutions.
[d]Sum of the iteration numbers.

point on the quality of the solutions in terms of the four evaluation categories (NBS, NFS, SBV for FS and SIN).

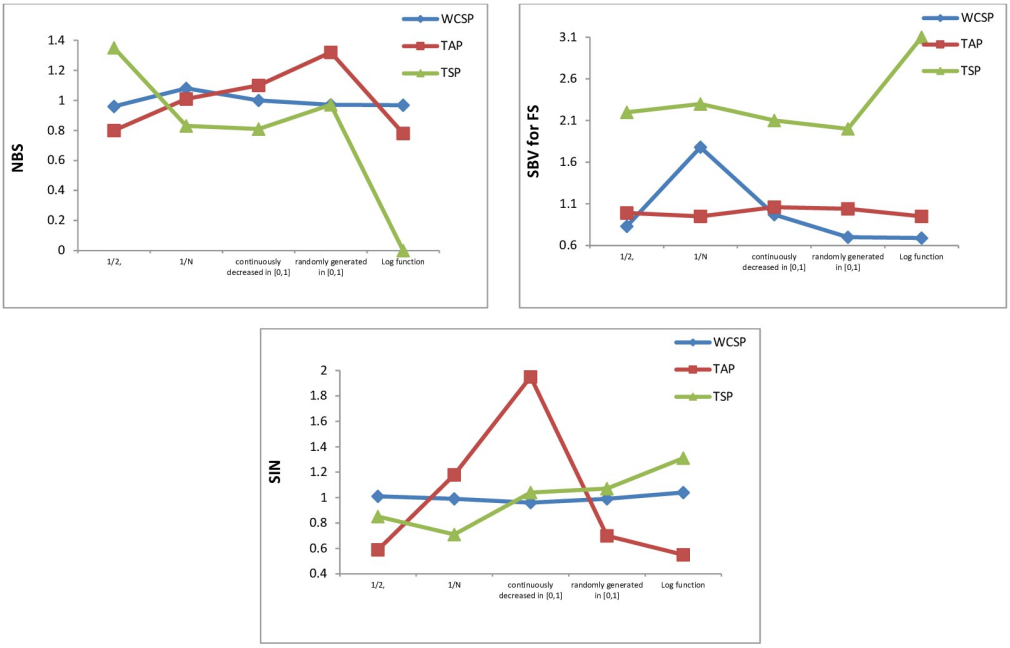


Figure 3.1: Variation of the hyperparameter α as a function of NBS, SBV of FS and SIN

Table 3.8: Obtained results using different combinations of hyperparameters for TAP

Hyper-parameters	Values	Mean NBS	NFS	SBV for FS	SIN
α	1/2	9,63	434	-275109,56	41791,83
	1/N[1]	12,06	542,33	-264024,59	83306,2
	Continuously decreased in [0,1]	13,21	535.54	-294724,11	137187,48
	Randomly generated in [0,1]	15,83	570,66	-290869,38	49230,76
	Log function	9,43	864,66	-264445,04	39066,84
ε	0,1	8,75	200	$-39982,46$	2097,15
	0,01	12,55	482	$-222801,5$	83734,36
	10^{-3}	12,5	632	$-330388,05$	44452,55
	10^{-4}	14,8	660	$-344262,07$	59132,25
	10^{-5}	16,85	680	$-356232,01$	64992,9
	10^{-6}	11,9	650	$-340680,69$	60282
starting point	0	0	86,66	$-71141,66$	16916,66
	0.5	0	125	-65425	7870,83
	$[0,1]$	26.5	1000	-115088,06	46541,83
	proposed in [11]	0	708,33	-556183,33	72379,25
	proposed in [29]	37.95	833,33	-561789,18	128194,38

[a]The number of neurons for each optimization problem.

Table 3.9: Obtained results using different combinations of hyperparameters for TSP

Hyper-parameters	Values	Mean NBS	NFS	SBV for FS	SIN
α	1/2	0.5	275.58	6544.9	57350.17
	$1/N$	0.316	238.44	6723.1	47819.69
	Continuously decreased in [0,1]	0.3	288.31	6141.25	70287.31
	Randomly generated in [0,1]	0.36	287.28	6029	72169.69
	Log function	0	360	9116.46	87723.78
ε	0,1	0	56	2593.44	13707.37
	0,01	0	387.04	8225.07	65549
	10^{-3}	0.5	332.08	5438.53	75103.5
	10^{-4}	0.54	350.67	5316.59	66180.29
	10^{-5}	0.37	351.75	5709.65	83592.33
	10^{-6}	0.33	387.92	5682.10	93295.95
starting point	0	0	0	0	185354.16
	0.5	0	541.67	541.67	26191.70
	$[0,1]$	1.75	580.62	6944.35	143896.87
	proposed in [11]	0	23.08	5567.40	12919.21
	proposed in [29]	0	719.08	5340.82	53424.87

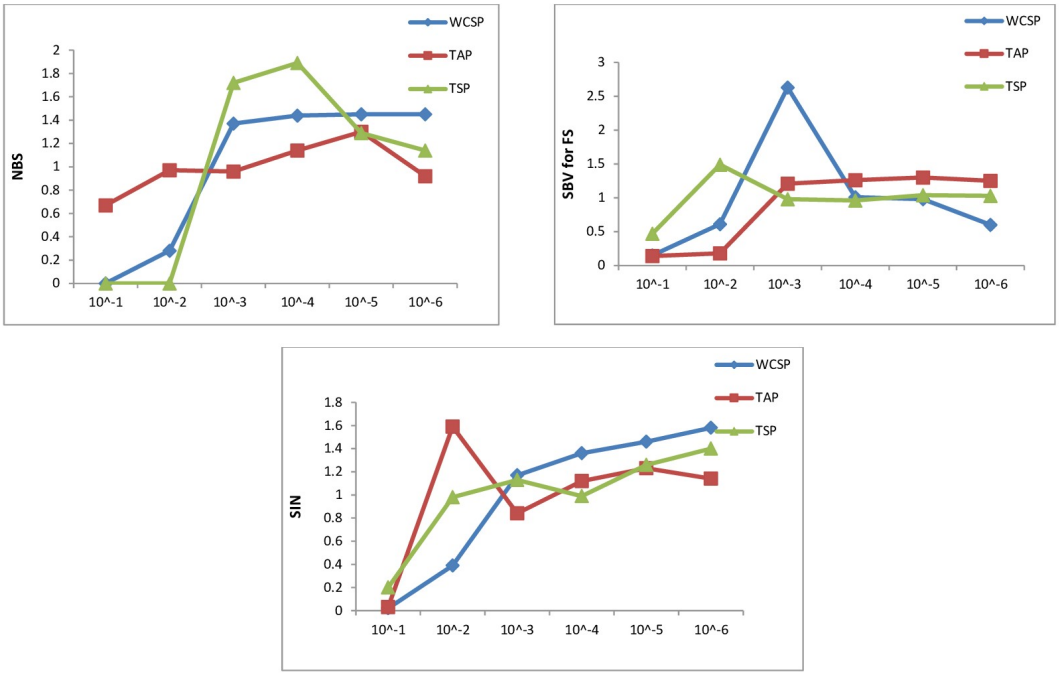


Figure 3.2: Variation of the hyperparameter ε as a function of NBS, SBV of FS and SIN

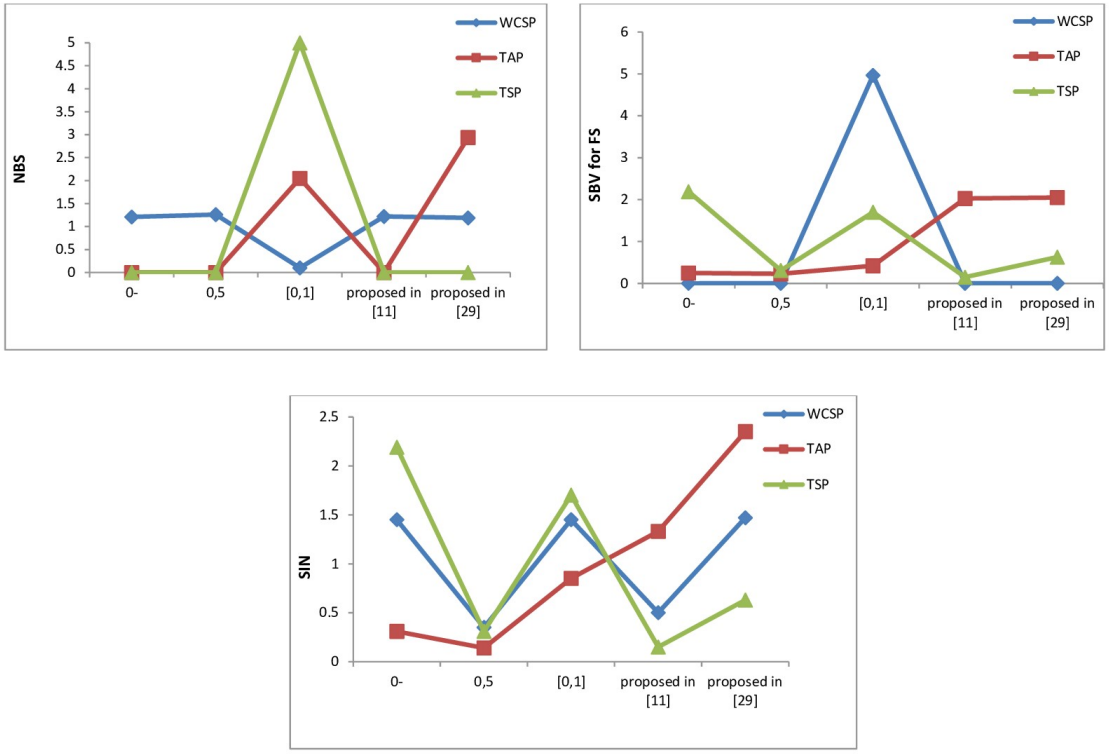


Figure 3.3: Variation of the starting point as a function of NBS, SBV of FS and SIN

In this Section, we propose and study an optimal tuning approach for continuous Hopfield networks applied to combinatorial optimization. We employ a stability analysis, notably the

equilibrium point criteria, to establish a parameter system that ensures the convergence of CHN while avoiding infeasible solutions. Additionally, we conducted a comparative study to analyze the impact of CHN hyperparameters and the starting point. To validate our proposed method, we apply it to three well-known combinatorial optimization problems: weighted constraint satisfaction problems, task assignment problems, and the traveling salesman problem. Several experiments have been conducted to demonstrate the performance of the suggested technique. The obtained numerical and graphical results highlighted the efficiency of our parameter settings and highlighted the selection of optimal hyperparameter combinations, which yield the best possible solutions and enhance the overall performance of CHN in solving COPs.

3.4 Conclusion

In this chapter, the Hopfield continuous neural network has been used to solve combinatorial optimization problems. As described in this chapter, the energy function of the optimization problem requires careful parameter selection. The network parameters are derived from the Lyapunov function, while the energy parameters are estimated through the stability analysis of the continuous Hopfield network.

In the third section, we introduce and investigate an optimal tuning approach designed for continuous Hopfield neural networks used in combinatorial optimization based on stability analysis. Furthermore, we study the influence of the CHN hyperparameters and the choice of the initial starting point. The experimental results confirm the effectiveness of our technique, as demonstrated by various numerical and visual tests.

Chapter 4

Hyperparameters optimization of Hopfield neural networks

4.1 Introduction

Combinatorial Optimization Problems (COPs) are used in many areas of the real world, including logistics, planning, finance, and many others. These problems revolve around identifying optimal solutions from a set of solutions. Classical methods are increasingly proving to be inadequate in providing optimal solutions within the required time frame for problems of large size. In response to this challenge, research has made significant progress by harnessing the potential of neural networks, which provide a robust and adaptive approach to optimization problems. This research aims to further advance the use of neural networks, with a particular focus on recurrent neural networks (RNNs) for solving COP. The main objective is to adapt RNNs for solving optimization problems to provide efficient solutions.

The Hopfield Neural Network (HNN) stands out as a special type of recurrent neural network that has developed rapidly and attracted considerable attention from researchers in various fields [59, 131, 19], particularly in the area of optimization [21, 43]. This network has been shown to be highly efficient compared to alternative approaches, particularly in providing high performance solutions to the combinatorial optimisation problems [31, 58, 61]. To effectively harness the power of this network to address COPs, it is imperative to construct a well-suited energy function. This energy function should include both the objective function, which defines the optimization objective, and the constraints specific to the problem. The correct formulation of such an energy function is crucial for achieving optimal solutions in the context of COPs. In addition, the construction of this energy function requires the initialization of penalty parameters. Some of these parameters are determined by stability analysis techniques [5]. However, there are certain parameters, known as hyperparameters, that require manual configuration, and no automated methods have actually been developed to determine their optimal values [52, 21, 20].

In related research, the determination of hyperparameters is usually achieved through a series of experimental trials based on trial and error, grid search or random selection [20]. These methods are inadequate for large problems because they require more time to find optimal solutions and have other drawbacks, notably sub-optimal solutions. Furthermore, the best hyperparameter settings for one problem may not be appropriate for another [52]. The challenge of selecting the best configuration hyperparameters of HNN to solve COPs has a major impact on network performance. To address this challenge, we propose a linear programming model as a solution. The proposed mathematical model aims to minimize the energy function, with

constraints representing penalty parameters, so that each parameter is constrained within its specific domain. To solve the mathematical model, we use two metaheuristic optimization algorithms [27, 28, 30]: Genetic Algorithms (GA) [37, 38, 39] and Particle Swarm Optimization (PSO) [40, 41, 42], both well-established methods known for their effectiveness in optimizing complex problem domains [24, 32, 25, 44, 47, 48, 49, 50].

The main propose of this chapter is to improve the performance of HNN for solving COPs. To this end, we introduce an original approach based on a linear programming model to optimize the hyperparameters of the HNN. The objective function is a linear majoration of the energy function (independent from the states and the outputs of the HNN) and the constraints represent a sufficient conditions to obtain a feasible solutions of the COP. We have applied our method to solve a number of COPs, including the Task Assignment Problem (TAP), the Traveling Salesman Problem (TSP), and the Max Stable Problem (MSP). In this context, we built appropriate energy functions from which we formulate our linear programming model. Then, we used GAs and PSO to estimate the optimal hyperparameters. The results presented in this paper demonstrate the importance of hyperparameter selection in the context of Hopfield networks and provide compelling evidence for the effectiveness of the proposed approach in solving COPs.

The organization of this chapter is as follows: the Section 4.2 present related work on selecting HNN hyper-parameters. In Section 4.3, we present the mathematical model to optimize HNN hyper-parameter. In the first part of this section, we present the mathematical model related to the optimization of CHN hyperparameters. Then, in the next part, we present several combinatorial optimization problems related to the mathematical model presented. In the third part, we give an overview of the algorithms tailored to the presented model. The results of the experimental study and the subsequent discussions are presented in the fourth part. In the Section 4.4, we present the tuning Hopfield Neural Networks with Metaheuristic Hyperparameter Selection. Finally, Section 4.5 summarizes the conclusions derived from the presented results.

4.2 Overview of existing research

Hyperparameter selection of Hopfield neural networks is a crucial step in optimizing their performance in solving optimisation problems. Model-free algorithms are the dominant methods for hyperparameter selection in HNNs. These algorithms include the following techniques: Baby-sitting , grid search, random search [46]. These methods involve an iterative adjustment of the hyperparameters of the HNN to fit within a predefined range of values. An advantage of these approaches is their compatibility with parallel computing [46]. However, a significant limitation arises when researchers and practitioners create hyperparameter configurations that may not be universally suitable for all scenarios. To elaborate, these methods involve a repetitive process in which HNN hyperparameters are systematically modified within a given range of values. This section of the literature examines a variety of COPs where model-free hyperparameter selection algorithms have been implemented for HNNs, as shown in Table (4.1).

Table 4.1: An overview of common HNN applications, their parameters and appropriate optimization techniques

Problem type	References	Main parameters	Parameters methods	Optional Hyper-parameters	Hyper-parameters methods
Traveling Salesman	[29] [31]	$\phi^{\mathrm{C}}, \phi^{F}$ γ, β	Hyperplane	α, ϕ ε	Babysitting
Traveling Salesman	[66]	A, B D	Stability analysis	C	Grid search
Task Assignment	[12] [23]	ϕ, β γ	Hyperplane	α ε	Random search
Weighted Constraint Satisfaction	[10]	ϕ, β γ	Hyperplane	α ε	Random search
Graph Coloring	[9]	ρ^{A}, ρ^{B} ϕ, β, γ_1	Stability analysis	α, γ_0 ε	Random search
Max-Stable	[8]	γ	Hyperplane	α ϕ, ε	Grid search
Meeting scheduling	[57]	ϕ, β γ	Hyperplane	α ε	Babysitting
Shortest path	[54] [22]	α_2 μ, ϕ	Hyperplane	α_1 γ, ε	Grid search
Maximal Constraint Satisfaction	[55]	ϕ, γ β	Stability analysis	α ε	Random search
Portfolio selection	[61]	ϕ_2 β_1, β_2	Stability analysis	α, ϕ_1 ε	-
Kohonen network architecture	[33]	ϕ ρ^{C}, β	Hyperplane	α, γ_0 γ_1, ε	Random search
Job Shop Scheduling	[65]	ϕ β, γ	Hyperplane	α ε	Random search
Vehicle routing	[56]	-	-	A, B C, D	Random search

4.3 Optimizing hyperparameters in Hopfield neural networks using evolutionary search

Classical methods for selecting hyperparameters of HNNs, which rely on model-free algorithms, often consume a significant amount of time evaluating poorly performing regions within the search space and do not guarantee convergence to optimal solutions. In this section, we address this challenge by introducing a novel linear optimization model designed to optimize the choice of hyperparameter tuning in Hopfield neural networks. For pedagogical reasons, we explain

the proposed model on the NP-hard problems meeting scheduling problem, Kohonen Network Problem, portfolio selection problem, traveling salesman problem, task assignment problem and max-stable problem.

4.3.1 Generic mathematical model

Considering the following generalized optimization problem:

$$(P)\begin{cases} Min \quad f(x) \\ s.c \\ \quad g_i(x) = 0 \quad \forall i = 1, ..., k \\ \quad x \in D = \{0,1\}^n \end{cases} \tag{4.1}$$

To solve (P) using HNN, we introduce the coming energy function:

$$E(x, \phi_1, ..., \phi_{k+2}) = \phi_1 f(x) + G(\phi_2, ..., \phi_{k+1}, g_1(x), ..., g_k(x)) + \phi_{k+2}\delta_D(x)$$

where ϕ_i are the penalty parameters, $\delta_D(x) = 0$ if $x \in D$ and G realize a quadratic or linear penalty of the constraints [131, 66, 5, 16, 29, 21].
Based on the first and second derivative of the energy function E and on the stability conditions, the domains of the penalty parameters $D_1, ..., D_{k+2}$ are explicited.
The optimal hyperparameters of the energy function E are the solution of the coming problem:

$$(P^{'})\begin{cases} Min \;\; Min \;\; E(x, \phi_1, ..., \phi_{k+2}) \\ s.c \\ \quad \phi_1 \in D_1, ..., \phi_{k+2} \in D_{k+2} \\ \quad x \in D \end{cases} \tag{4.2}$$

The function $E(x, \phi_1, ..., \phi_{k+2})$ represents an objective function that depends on the variables x and the set of parameters $(\phi_1, ..., \phi_{k+2})$. The purpose of this model is to identify values for both x and the parameters that will minimize the function E. In essence, it serves as a mathematical tool for the optimization process, searching for the most favorable values of x and the associated parameters that will lead to the minimization of the function, thereby achieving the optimization objective.

Consider the Nernst function, denoted as F, associated with the energy function E such that $E(x, \phi_1, \ldots, \phi_{k+2}) \leq F(\phi_1, \ldots, \phi_{k+2})$. This function is designed to optimize the energy function E in terms of only the parameters $(\phi_1, ..., \phi_{k+2})$; instead of solving the NP-hard problem, we will solve the following problem:

$$(P^{''})\begin{cases} Min \quad F(\phi_1, ..., \phi_{k+2}) \\ s.c \\ \quad \phi_1 \in D_1, ..., \phi_{k+2} \in D_{k+2} \end{cases} \tag{4.3}$$

In the next subsections we will apply this procedure to explicit the model $(P^{''})$ for solving some combinatorial optimization problems.

4.3.2 Application to the COPs

In the following subsection, we explain the ($P^{''}$) model proposed in the previous subsection for certain combinatorial optimization problems.

1. **Weighted Constraint Satisfaction Problem**

   The mathematical model ($P^{''}$) associated with the WCSP, where the penalty parameters are determined by the hyperplane method, is defined by:

$$\begin{cases} Min \quad F_{WCSP}(\alpha, \phi, \varepsilon) \\ s.c \\ \quad -\phi + 2\gamma \geq 0 \\ \quad \alpha(Q_{\min} + q_{\min}) + 2\phi + \beta \geq \gamma \\ \quad \alpha(Q_{\max} + q_{\max}) + \beta \leq -\gamma \\ \quad \phi \geq 0,\ \alpha > 0 \end{cases} \tag{4.4}$$

   Where, $Q_{max} = max\{q_{irjs}\}$, $q_{max} = max\{q_{ir}\}$, $Q_{min} = min\{q_{irjs}\}$ and $q_{min} = min\{q_{ir}\}$

2. **Traveling Salesman Problem**

   The mathematical model ($P^{''}$) associated with the TSP, where the penalty parameters are determined by the hyperplane method, is defined by:

$$\begin{cases} Min \quad F_{TSP}(\alpha, \phi, \phi^{\mathrm{C}}, \phi^{F}, \gamma, \beta, \varepsilon) \\ s.c \\ \quad 3\alpha d_U \leq \phi^C + \varepsilon \\ \quad 3\alpha d_U \leq \phi^F + \alpha d_L + \varepsilon \\ \quad 2\gamma \geq \phi^F + \phi^C + \phi + \varepsilon \\ \quad -2\beta \geq 3\phi^C + \phi^F + (2N-1)\phi + \varepsilon \\ \quad \phi \geq 0, \alpha > 0, \varepsilon > 0 \end{cases} \tag{4.5}$$

   Where, d_L and d_U are the lower and upper limits of the Euclidean distance between any two cities. N is the total number of neurons.

3. **Task Assignment Problem**

   The proposed model ($P^{''}$) of TAP is:

$$\begin{cases} Min \quad F_{TAP}(\alpha, \phi, \beta, \gamma, \varepsilon) \\ s.c \\ \quad -\phi + 2\gamma \geq 0 \\ \quad \alpha d_{\min} + 2\phi + \beta - \gamma \geq \varepsilon \\ \quad \alpha d_{\max} + \beta + \gamma \leq -\varepsilon \\ \quad \phi \geq 0, \alpha > 0, \varepsilon > 0 \end{cases} \tag{4.6}$$

   Where, $d_{min} = M(N-1)C_{min} + e_{min}$, $d_{max} = M(N-1)C_{max} + e_{max}$. N is the number of tasks, M is the number of processors. C_{max} and C_{min} are the maximum and minimum of the communication cost matrix C. e_{max} and e_{min} are the maximum and minimum of the execution cost vector e.

4. **Max-Stable Problem**

The proposed model ($P^{''}$) associated with the MSP as follows:

$$\begin{cases} Min \quad F_{MSP}(\alpha, \phi, \gamma, \varepsilon) \\ s.c \\ \quad -\alpha + \phi - \gamma \geq \varepsilon \\ \quad \phi > 0, \alpha > 0, \varepsilon > 0 \end{cases} \tag{4.7}$$

The constraints on the penalty parameters are obtained by analyzing the first and second derivatives of the energy function Eq.(3.4) and the stability conditions.

5. **Graph Coloring Problem**

The proposed model ($P^{''}$) associated with the GCP as follows:

$$\begin{cases} Min \quad F_{GCP}(\alpha, \gamma_0, \varepsilon) \\ s.c \\ \quad \alpha(\frac{N}{2} + 2) - \rho^A + \gamma_0 \leq -\varepsilon \\ \quad \rho^B + \phi + \beta - \gamma_1 \geq \varepsilon \\ \quad 2\phi + \beta - \gamma_1 \geq \varepsilon \\ \quad \rho^A + \beta + \gamma_1 \leq -\varepsilon \\ \quad \rho^A \geq 0, \rho^B \geq 0, \phi \geq 0 \end{cases} \tag{4.8}$$

The constraints on the penalty parameters are obtained by analyzing the first and second derivatives of the energy function Eq.(3.5) and the stability conditions.

6. **Meeting scheduling problem**

The proposed model ($P^{''}$) to optimize the choice of hyperparameters of MSP is:

$$\begin{cases} Min \quad F_{MSP}(\alpha, \phi, \beta, \gamma, \varepsilon) \\ s.c \\ \quad -\phi + 2\gamma \geq 0 \\ \quad 2\phi + \beta - \gamma \geq \varepsilon \\ \quad 2d + \beta + \gamma \leq \varepsilon \\ \quad \phi \geq 0, \alpha > 0, \varepsilon > 0 \end{cases} \tag{4.9}$$

The constraints on the penalty parameters are obtained by analyzing the first and second derivatives of the energy function Eq.(3.7) and the stability conditions.

7. **Kohonen Network Problem**

The proposed model ($P^{''}$) associated with KNP to optimize its hyperparameters:

$$\begin{cases} Min \quad F_{KNP}(\alpha, \rho^C, \phi, \beta, \gamma_0, \gamma_1, \varepsilon) \\ s.c \\ \quad -\phi + 2\gamma_1 \geq 0 \\ \quad \rho^C + \gamma_1 + \beta \geq \varepsilon \\ \quad \alpha(M_2^{t-1} + q) - \rho^C \leq -\varepsilon \\ \quad \alpha M_1^{t-1} + \beta + \gamma_1 \leq -\varepsilon \\ \quad \alpha > 0, \gamma_1 \geq 0, \gamma_0 \geq 0, \varepsilon \geq 0 \end{cases} \tag{4.10}$$

The constraints on the penalty parameters are obtained by analyzing the first and second derivatives of the energy function Eq.(3.8) and the stability conditions.

8. **Portfolio Selection Problem (PSP)**

   The proposed mathematical model ($P^{''}$) associated with PSP is as follows:

$$\begin{cases} Min \quad F_{PSP}(\alpha, \phi_1, \phi_2, \beta_2, \beta_2, \varepsilon) \\ s.c \\ \quad \alpha(1-\lambda)\sigma - \alpha\lambda\mu + \beta_1 + \beta_2\delta_i + \phi_1\delta_i|I| \leq \varepsilon \\ \quad \alpha(1-\lambda)\sigma_{min} - \alpha\lambda\mu_{min} + \beta_1 + \beta_2 + \phi_1|I| \leq \varepsilon \\ \quad \phi_1 > 0, \alpha > 0, \varepsilon > 0 \end{cases} \tag{4.11}$$

   Where, I is the set of the asset.

In the next section, we present two algorithms for solving the proposed mathematical model ($P^{''}$). These algorithms are based on genetic and particle swarm optimization techniques.

4.3.3 Proposed algorithm

The use of heuristic algorithms to achieve optimal hyperparameter selection has received considerable attention in the research community. This chapter focuses on the optimization of hyperparameters for the Hopfield neural network, as represented by the mathematical model ($P^{''}$), by exploiting the capabilities of genetic algorithms and particle swarm optimization. Our primary goal is to improve the overall performance of the network, in particular its ability to effectively tackle complex combinatorial optimization problems.

1. **Genetic algorithms to solve the proposed model**

   Genetic algorithms (GA), belonging to the family of evolutionary algorithms based on natural selection and genetics [36, 25, 18, 17, 51], are used to solve the proposed model ($P^{''}$) for determining the optimal HNN hyperparameters. The following Fig.(4.1) and Algorithm (2) present the detailed steps of using GA to solve the mathematical model ($P^{''}$).

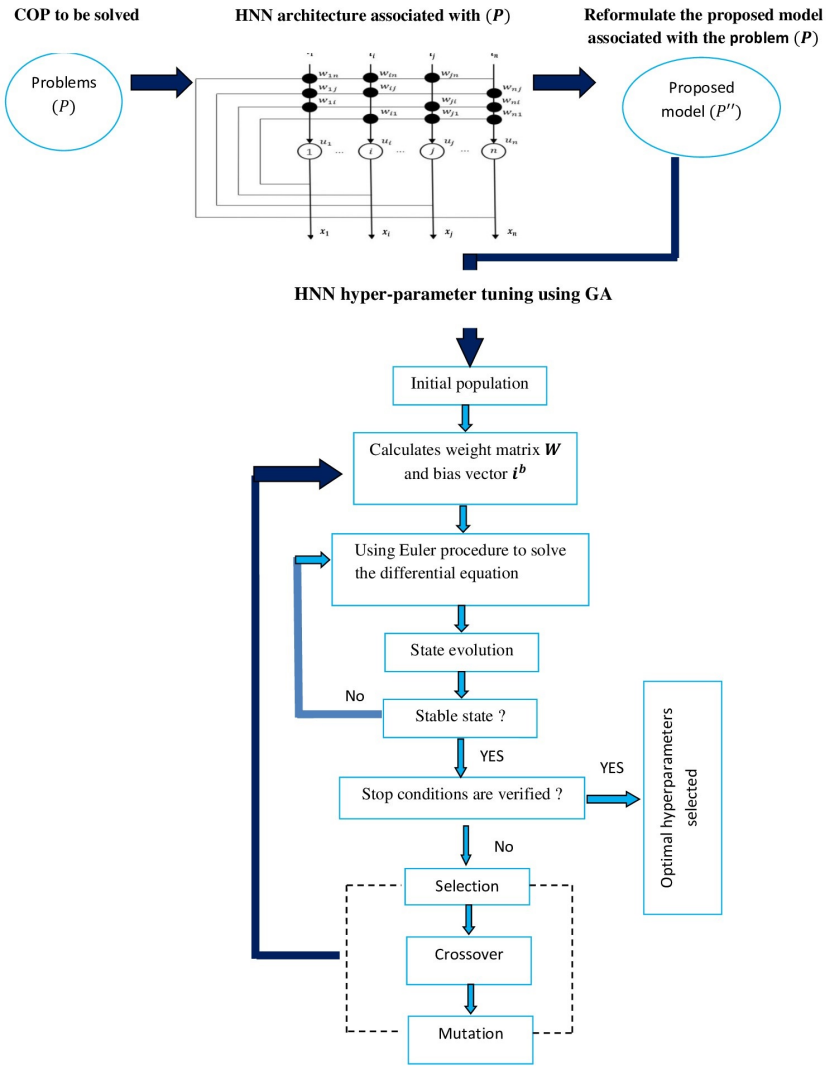


Figure 4.1: GA procedures for selecting optimal hyperparameters

Algorithm 2 Pseudocode for Hyperparameter Optimization using GA

1: Input $population_size$, $number_generation$, $individual_size$, $D_1, \ldots, D_{individual_size}$
2: $Genes \leftarrow [random.choice(D_1), \ldots, random.choice(D_p)]$
3: $Population \leftarrow POPULATION(population_size, Genes)$
4: $t \leftarrow 0$
5: **while** $t < number_generation$ **do**
6: $\quad Fitness \leftarrow HNN_FITNESS(population_size, Population)$
7: $\quad$ **if** $min(Fitness) \leftarrow optimal_solution$ **then**
8: $\quad\quad break$
9: $\quad$ **end if**
10: $\quad Newpopulation \leftarrow SELECTION(population_size, Population)$
11: $\quad Newpopulation \leftarrow CROSS(population_size, individual_size, Newpopulation)$
12: $\quad Newpopulation \leftarrow MUTATION(population_size, individual_size, Genes, Newpopulation)$
13: $\quad t++$
14: **end while**

The Algorithm (2) outlines our method for optimizing HNN hyper-parameters using GAs with the functions defined above (POPULATION, HNN_FITNESS, SELECTION, CROSS, and MUTATION). The GA parameter and the HNN hyper-parameters associated with each problem used are specified (line 1). The function POPULATION is

called to initialize the population (lines 2-3). Next, the fitness function computed using the function CHN_FITNESS (line 6). If the optimal solution is reached, the algorithm stops at this iteration (lines 7-9), otherwise, it continues. The selection process is created using the function SELECTION (line 10). Crossover is performed using the function CROSS (line 11), and mutation using the function MUTATION (line 12). These steps are repeated for the specified *number_generation* iterations (lines 5-14).

Algorithm (3) defines the functions used in the proposed model to select optimal hyper-parameters. The algorithm begins by initializing a randomly selected population of individuals with size *population_size* using the POPULATION function. The HNN_FITNESS function is then used to calculate the fitness function for the population using the equilibrium point calculation algorithm [16]. The algorithm then enters a loop that runs for a specified number of generations. During each iteration, the selection function selects individuals from the population to create a new population. The type of selection used in this algorithm is a "Roulette wheel". The CROSS function is then used to perform crossover on the new population. The two-point crossover is used in this algorithm. Finally, the Mutate function is used to perform point mutation on the new population. The mutation point with a flip bit is used in this algorithm. The algorithm ends by returning the best individual found during the optimization process.

During the execution of the proposed algorithm, an important concept to consider is the algorithmic complexity, which represents the number of elementary operations performed by the algorithm.

Lemma 4.1. *The Algorithm 2 function runs in polynomial time* $O(number_generation \times population_size \times individual_size \times max_iter \times n)$.

The following corollaries outline the complexity of our proposed algorithm (alg.2) in relation to the addressed optimization problems (TSP, TAP, and MSP).

Corollary 4.1. *The proposed Algorithm (2) for TSP with* N *cities and* 4 *hyper-parameters has a complexity of* $O(NG \times PS \times 4 \times max_iter \times N^2)$.

Corollary 4.2. *The proposed Algorithm (2) for TAP with* N *tasks,* M *processors and* 3 *hyper-parameters has a complexity of* $O(NG \times PS \times 3 \times max_iter \times N \times M)$.

Corollary 4.3. *The proposed Algorithm (2) for MSP with* N *Nodes and* 4 *hyper-parameters has a complexity of* $O(NG \times PS \times 4 \times max_iter \times N^2)$.

2. **Particle Swarm Optimization to solve the proposed model**

Particle Swarm Optimization (PSO) is a probabilistic, swarm-based intelligence algorithm known for its conceptual simplicity and straightforward implementation [44, 47, 48, 49, 50, 45]. During the particle swarm optimization process, the velocity ($velocities_i$) of each particle ($particle_i$) is continuously fine-tuned through iterations, leading the particle to undergo stochastic oscillations around both best position ($p_{best_positions}$) and neighborhood best ($g_{best_positions}$) [44, 46, 45]. The following Fig.4.2 and Algorithm 4 present the detailed steps of use PSO to select optimal hyperparameters of HNN.

Algorithm 3 POPULATION, CHN_FITNESS, SELECTION, CROSS and MUTATE function

function POPULATION(POPULATION_SIZE,GENES)
Population$_i$ ← (random.choice (Genes$_j$) for j in range(*individual_size*) for i in range(*population_size*))
return (*Population*)
end function
function HNN_FITNESS(POPULATION_SIZE, POPULATION)
for i **in** *population_size* **do**
Compute (W, i^b)
$t \leftarrow 0$
while $t <$ *max_iter* **do**
Compute $x_i \leftarrow CHN(W, i^b)$ [16]
if $x_i \in H_F$ **then**
Compute $f(x_i)$
$Fitness_i \leftarrow \frac{f(x_i)}{\sum_i f(x_i)}$
end if
if $x_i \notin H_F$ **then**
$Fitness_i \leftarrow 0$
end if
end while
end for
return (*Fitness*)
end function
function SELECTION(POPULATION_SIZE,POPULATION)
$Fitness \leftarrow HNN_FITNESS(population_size, Population)$
$New_population \leftarrow [Population_i$ for i in range(*population_size*) if $Fitness_i > 0]$
return (*New_population*)
end function
function CROSS(POPULATION_SIZE,INDIVIDUAL_SIZE,NEWPOP)
population_size ← *len*(*newpop*)
for i **in** *population_size* **do**
$p \leftarrow random(individual_size)$, $r_1, r_2 \leftarrow random(1, population_size)$
$New_population_{r_1,p} \leftarrow New_population_{r_2,p}$, $New_population_{r_2,p} \leftarrow New_population_{r_1,p}$
end for
return (*New_population*)
end function
function MUTATE(POPULATION_SIZE,IS,GENES,NEWPOP)
$New_population_{i,t}$ ← (random.choice(Genes$_t$) t ← random(individual_size) for i in range(*population_size*))
return (*New_population*)
end function

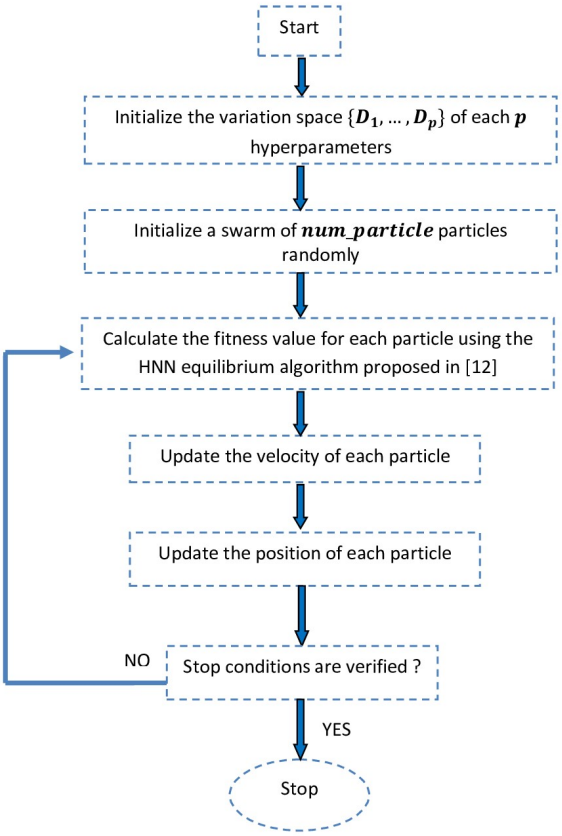


Figure 4.2: PSO procedures for selecting optimal hyper-parameters

Notes:
In the Algorithm (2):
- H_F present the set of feasible solutions.
In the Algorithm (4), the following points are significant:
- $None$ means that the variable has no value assigned to it at the moment.
- $float('inf')$ means it sets the variable to positive infinity.

The Algorithm (4) outlines our method for optimizing HNN hyper-parameters using PSO. The set parameter of PSO used are specified (line 1). (Line 2) denotes the variation space of the hyperparameters under optimization. (Lines 3-10) represents the initialization of particles' velocity and position, $p_{best_positions}$, p_{best_values}, $g_{best_positions}$ and g_{best_values}. For each particle (combination of hyperparameters), we compute the fitness function using the algorithm introduced in [16] to determine the equilibrium point. Should the newly generated solution surpass the current personal best result, the particle's best-known position is updated accordingly (Lines 13-23). The update particle's velocity and position are in (Lines 24-28). The algorithm concludes its pursuit of optimal hyperparameter combinations for such a problem upon reaching the maximum number of iterations ($max_iterations$).

Lemma 4.2. *The Algorithm (4) function runs in polynomial time* $O(max_iterations \times [num_particle \times max_iter \times n \times p + 4\ num_particle])$. *(m is HNN size)*

The following corollaries outline the complexity of our proposed Algorithm (4) in relation to the handled problems (TSP, TAP, and MSP).

Corollary 4.4. *The proposed Algorithm 4 for TSP with N cities and 4 hyper-parameters has a complexity of* $O(max_iterations \times [num_particle \times max_iter \times 6N^2 \times 4 + 4\ num_particle])$.

Algorithm 4 Pseudocode for Hyperparameter Optimization using PSO

Input PSO parameters: $num_particle$, c_1, c_2, w,$max_iterations$
Set the number of hyperparameters p and specify the variation space for each of them $D_1, \ldots, D_p$
for i **in** $num_particle$ **do**
    $particle_i = [random.choice(D_1), \ldots, random.choice(D_p)]$
    $velocities_i \leftarrow$ vector generated at 0.
end for
$p_{best_positions} = particle$
$p_{best_values} \leftarrow$ vector generated at 0.
$g_{best_positions} = None$
$g_{best_values} = float('inf')$
$iteration \leftarrow 1$
while $iteration \leq max_iterations$ **do**
    for i **in** $num_particle$ **do**
        Compute (W, i^b)
        $t \leftarrow 0$
        while $t < max_iter$ **do**
            Compute $particle_i \leftarrow HNN(W, i^b)$ [16]
        end while
        $fitness = f(particle_i)$
        if $fitness < p_{best_values_i}$ **then**
            $p_{best_values} = fitness$
            $p_{best_positions_i} = particle_i$
        end if
        if $fitness < g_{best_values}$ **then**
            $g_{best_values} = fitness$
            $g_{best_positions} = particle_i$
        end if
    end for
    for i **in** $num_particle$ **do**
        $r_1, r_2 \leftarrow$ two independent vectors randomly generated from $[0, 1]$
        $velocities_i = w * velocities_i + c_1 * r_1 * (p_{best_positions_i} - particle_i) + c_2 * r_2 * (g_{best_positions} - particle_i)$
        $particle_i = particle_i + velocities_i$
    end for
    $iteration \leftarrow iteration + 1$
end while

Corollary 4.5. *The proposed Algorithm (4) for TAP with N tasks, M processors and* 3 *hyper-parameters has a complexity of* $O(max_iterations \times [num_particle \times max_iter \times 6N \times M \times 3 + 4\ num_particle])$

Corollary 4.6. *The proposed Algorithm (4) for MSP with N Nodes and* 4 *hyper-parameters has a complexity of* $O(max_iterations \times [num_particle \times max_iter \times 6N^2 \times 4 + 4\ num_particle])$.

In the next section, to implement the proposed algorithms, we propose to conduct an experiment on the combinatorial optimization problems.

4.3.4 Experimentation and discussion

In this section, we implement the proposed Algorithms (2 and 4) to address the proposed model ($P^{''}$). We have chosen to study three different problems: the Traveling Salesman Problem (TSP), the Task Assignment Problem (TAP) and the Max-Stable Problem (MSP). Our primary objective is to evaluate and compare the effectiveness of our proposed approach. In order to thoroughly evaluate the advantages and possible improvements of our method, we will perform a comparative analysis with the traditional 'random selection' method for hyperparameter selection.

Table 4.2: GA parameters

Parameter	Signification	value
Population Size	Number of individuals	[500, 1000]
Number of executions	Number of times that algorithm is run	[500, 1000]
Number of generations	Number of iterations	[500, 1000]
Initial population	-	Random draw
Selection type	-	Roulette wheel
Selection point	-	Random choice
Crossover type	-	Two point
Mutation type	-	Flip bit

Table 4.3: PSO parameters

Parameter	Signification	value
num_particle	Number of particles	[500, 1000]
max_iter	Maximum number of CHN iterations	[500, 1000]
c_1	cognitive parameter	0.7
c_2	social parameter	0.7
w	inertia weight	0.7
max_iterations	Maximum number of PSO iterations	[200,500]

The results presented in Tables (4.4 to 4.6) show the results generated by our proposed model, which was approached using two different algorithms: Algorithm (5), denoted as (PA_GA), and Algorithm 4, denoted as (PA_PSO). In addition, we used a classical model-free method where we use 'random selection' in the case of selected instances related to TSP, TAP and MSP. The results of this study are categorized into the following aspects: performance ratio (PR), number of iterations for all feasible solutions (NI), and Total time in ms (TT) .

In the existing literature, different studies have tackled the three problems outlined in this paper using the HNN, relying mainly on classical model-free methods for its hyperparameters.

Table 4.4: Summary of TSP simulation experiments

	Random selection			*PA_GA*			*PA_PSO*		
Instances	PR	NI	TT	PR	NI	TT	PR	NI	TT
FIVE	1.21	12	0.12	**1**	140	0.19	**1**	22	0.13
Buma14	1.45	55	1.95	**1**	137	71.01	**1**	55	15.5
P01	1.13	20	5	**1**	308	27	**1**	145	25
Ulysses16	2.07	15	2.20	**1**	318	27	**1**	220	24
Gr17	1.74	20	3.15	**1**	410	68	1.07	85	66
Gr21	1.63	95	11.5	**1**	132	56	1.07	115	49
Ulysses22	1.81	103	18.14	**1**	180	41	1.22	155	18
Gr24	2.28	125	24.315	1.19	187	41	1.6	165	44.8
Fri26	1.86	34	35.88	1.19	601	47	1.83	315	42
Bays29	2.20	130	46.5	1.36	290	63	1.68	195	53
Dantzig42	2.41	346	231.9	1.43	438	147	2	320	134
Att48	2.96	477	335	1.89	429	180	2	315	175
Mean	**1.89**	**119.33**	**59.64**	**1.17**	**297.5**	**64**	**1.37**	**175.58**	**53.8**

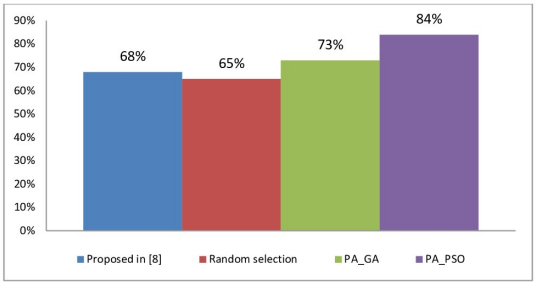
Table 4.5: Summary of TAP simulation experiments

	Random selection			*PA_GA*			*PA_PSO*		
Instances	PR[1]	NI	TT	PR	NI	TT	PR	NI	TT
tassnu_10_3_1	**1**	764	3.56	**1**	966	21	**1**	876	17
tassnu_10_3_2	**1**	780	3.35	1	884	22.12	**1**	165	18
tassnu_10_3_3	0.99	714	3.17	**1**	2486	77	**1**	1337	35
tassnu_10_3_4	0.84	751	3.34	**1**	1760	80	**1**	1005	52
tassnu_10_3_5	0.92	751	3.3	**1**	3437	35	0.92	3320	40
tassnu_10_3_6	**1**	862	3.7	**1**	3134	60	**1**	1652	37
tassnu_10_3_7	**1**	821	3.6	**1**	3275	52	**1**	1750	38
tassnu_10_3_8	0.97	974	4.2	**1**	1750	40.35	**1**	1362	16
tassnu_10_3_9	0.98	727	3.2	**1**	1473	35	**1**	1000	12
tassnu_10_3_10	0.93	805	3.5	**1**	7240	30	**1**	1300	15
tassnu_15_5_1	0.9	866	17.6	**1**	4846	48	0.96	3850	42
tassnu_15_5_2	0.89	971	19.38	0.98	2750	145	0.97	2129	141
tassnu_15_5_3	0.83	909	18.36	**1**	4208	52	0.94	2500	45
tassnu_10_5_4	0.85	928	18.53	**1**	5500	70	0.97	4940	60
tassnu_15_5_5	0.95	922	18.35	0.98	1334	87	0.95	1290	44
tassnu_15_5_6	0.93	943	19	**1**	1401	80	0.99	1300	46
tassnu_15_5_7	0.96	958	18.66	0.98	8382	55	0.98	1700	28
tassnu_15_5_8	0.97	921	17.98	**1**	9368	70	0.99	3700	46
tassnu_15_5_9	0.84	949	18.67	0.97	1538	108	0.96	1500	95
tassnu_15_5_10	0.84	936	18.5	**1**	2443	54	0.97	900	51
Mean	**0.93**	**862.6**	**10.99**	**0.99**	**3408.75**	**61.07**	**0.98**	**1878.8**	**43.9**

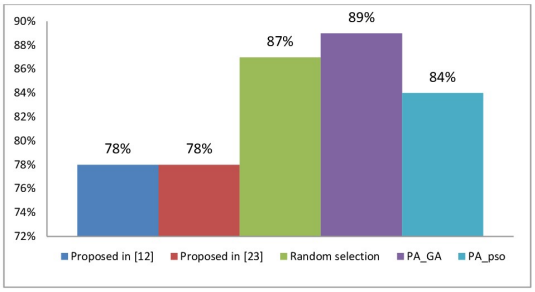
[a] $PR \leq 1$: explain this in reference to the negative optimal value of each instance problem.

Table 4.6: Summary of MSP simulation experiments

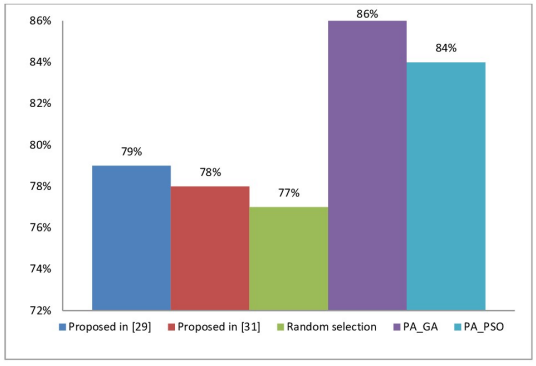
	Random selection			*PA_GA*			*PA_PSO*		
Instances	PR	NI	TT	PR	NI	TT	PR	NI	TT
brock200_1	0.28	55	10	**1**	184	20	**1**	175	10
brock200_2	0.92	57	51.4	0.92	252	60	0.83	90	50
brock200_4	0.53	70	31	0.82	290	45	0.82	262	41
brock400_4	0.21	110	37.1	0.85	163	86	0.79	145	45
brock800_2	0.5	154	54	**1**	175	200	0.96	170	113
C125.9	0.76	38	9.6	0.91	298	18	0.65	120	14
C250.9	0.85	23	15.8	0.84	770	94	0.95	150	73
C1000.9	0.5	75	17	0.15	102	87	0.91	100	68
c_fat200_1	0.58	82	12	0.92	853	120	**1**	650	94
gen200_p0.9_44	0.61	28	11.2	**1**	180	34	0.7	148	15
gen400_p0.9_55	0.69	80	51.5	0.91	552	100	0.93	450	61
hamming8_4	**1**	103	54	**1**	146	95	**1**	125	50
hamming10_4	**1**	200	74	**1**	274	110	**1**	237	55
keller4	0.63	130	20.8	0.91	259	96	0.91	230	87
keller5	0.37	154	35	0.85	373	53	0.96	210	42
MANN_a27	0.57	12	30	0.14	102	97	0.86	37	84
p_hat300_1	**1**	73	31	**1**	474	65	**1**	241	57
p_hat300_3	0.86	35	20	0.78	531	64	0.89	450	41
p_hat700_1	**1**	47	74	**1**	426	200	**1**	355	185
p_hat700_2	0.8	72	85	**1**	495	134	**1**	114	103
Mean	**0.68**	**79.9**	**36.22**	**0.85**	**344.95**	**88.9**	**0.91**	**222.95**	**63.05**



(a) Accuracy of MSP

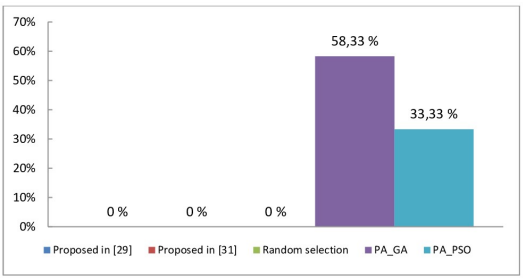


(b) Accuracy of TAP

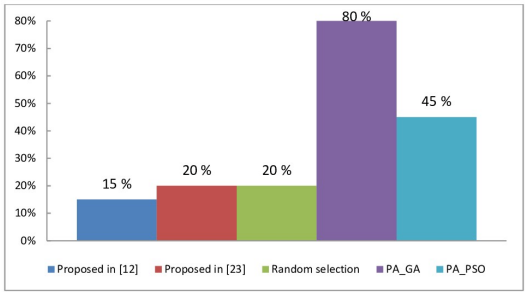


(c) Accuracy of TSP

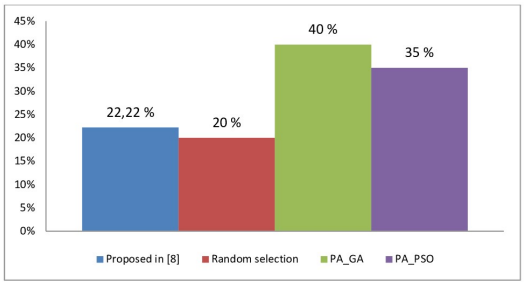
Figure 4.3: Accuracy of handled problems using different methods of hyperparameter selection



(a) Percentage of optimal solutions (%) of TSP

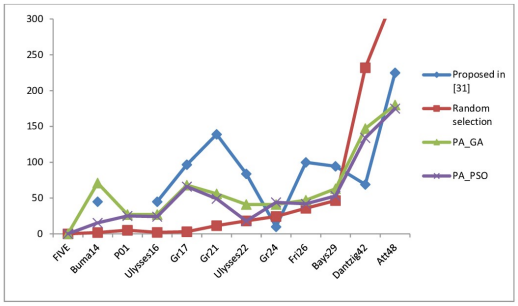


(b) Percentage of optimal solutions (%) of TAP

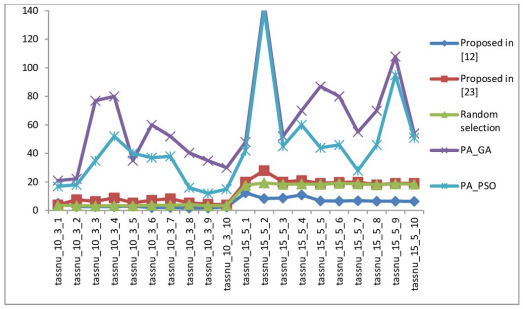


(c) Percentage of optimal solutions (%) of MSP

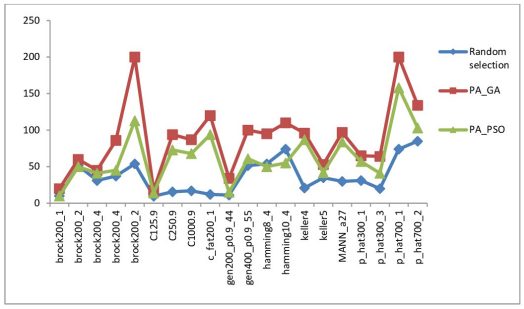
Figure 4.4: Percentage of optimal solutions for problems studied using different methods of hyperparameter selection



(a) Total time for TSP instances

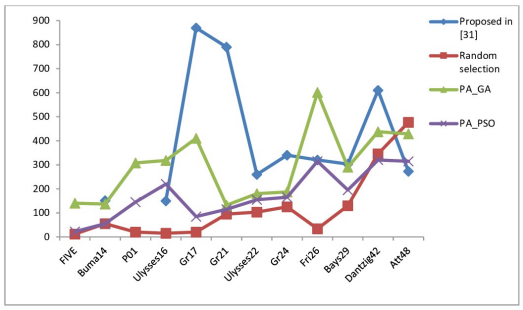


(b) Total time for TAP instances

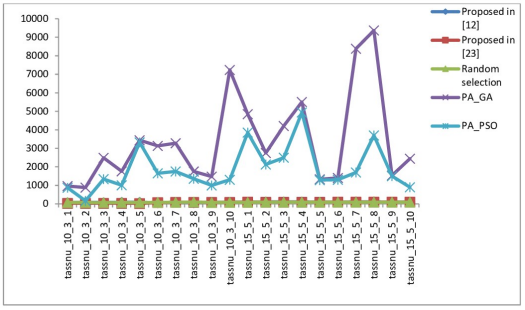


(c) Total time for MSP instances

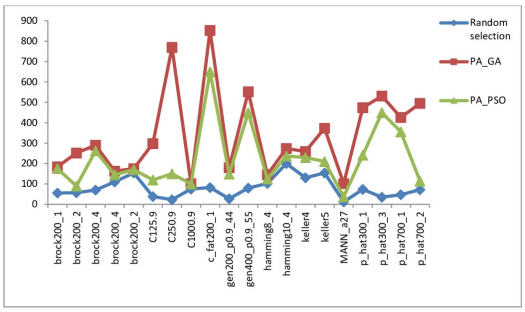
Figure 4.5: Total time of handled problems using different methods of hyperparameter selection



(a) Number of iterations for TSP instances



(b) Number of iterations for TAP instances



(c) Number of iterations for MSP instances

Figure 4.6: Number of iterations for problems studied using different methods of hyperparameter selection

Our contribution to this paper includes additional research focused on the same problems, with the aim of providing a comparative analysis between the traditional methods for selecting hyperparameters of the HNN and our proposed approach. We included visual representations of the results presented in the Fig.(4.3) to Fig.(4.6) shows the accuracy graphs of TSP, TAP, and MSP using our proposed approach and the classical model-free methods. The optimal solution percentages obtained by the HNN hyperparameter selection methods are shown in Fig.(4.4). The graphs representing the total time and iterations are shown in Fig.(4.5) and Fig.(4.6), respectively.

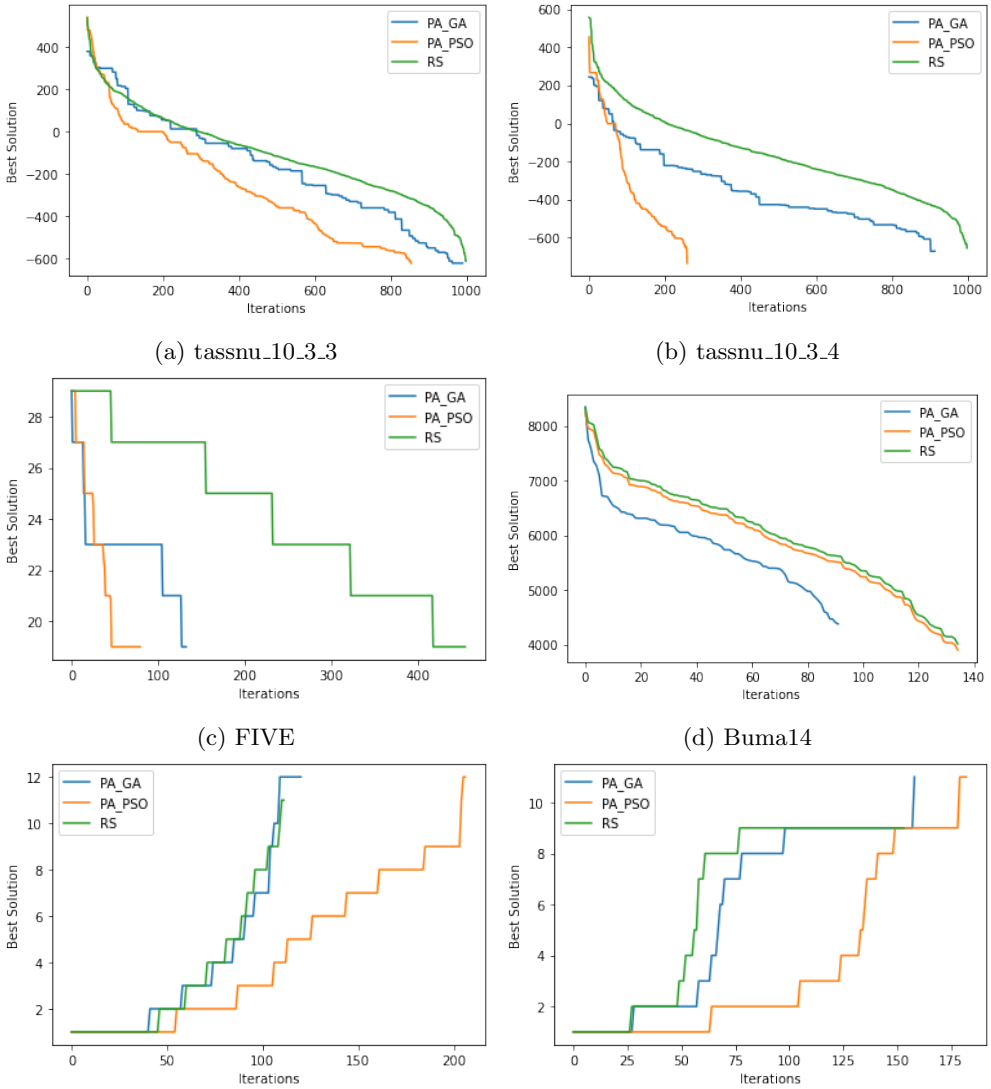
The results concerning the two criteria of accuracy and optimality are visually presented in Fig.(4.3) and Fig.(4.4), respectively. A careful analysis of these criteria for the three COPs examined shows that the results do not vary significantly when model-free selection methods are used. This observation applies not only to the model-free selection methods used in our experiment but also to those cited in the references used in our study. In simpler terms, the accuracy and optimality results are consistently close to each other and generally show a lower level of performance compared to the results obtained using our proposed model (PA_GA and PA_PSO).

Moreover, regarding the criterion of the number of iterations shown in Fig. (4.6), a comparison between the TSP results shown in Fig. (4.6a) shows that the number of iterations of [31] and those presented in Table (4.4) (derived from selection methods without models) exceed those of our proposed models (PA_GA and PA_PSO) for cases such as Att48, GR17, GR21, Ulysses22, Gr24, and Danzig42. This increase in iterations can be attributed to the fact that classical selection methods inherently require more iterations, and we also repeat the execution to improve the performance of the results. For instances Five, P01, Ulysses16 and Fri26 we observed that our proposed model required more iterations compared to the selection without model. This increase in the number of iterations can be attributed to the fact that these particular instances have a larger number of feasible solutions. Consequently, the search

space becomes more constrained, leading to longer convergence times at the iteration level. For the TAP and the MSP as depicted in Fig.(4.6b) and Fig.(4.6c) respectively, it is evident that our proposed model based approaches (PA_GA and PA_PSO) exhibit an increase in terms of the number of iterations when compared to the model-free methods. This increase is due to the significantly larger number of feasible solutions explored by our model based algorithms.

Furthermore, for the execution time criterion shown in Fig.(4.5), we can see a similar pattern of variation to that observed for the iteration criterion for all three COPs. This consistency in the pattern of variation is readily apparent: an increase in the number of iterations corresponds to an increase in execution time, and conversely, a decrease in the number of iterations results in a decrease in execution time.

The Figure (4.7) shows the convergence of some instances for each problem addressed, namely: tassnu_10_3_3, tassnu_10_3_4, tassnu_15_5_5 for the TAP , and FIVE, Buma14 for the TSP, as well as brock200_2 and keller4 for the MSP. These curves illustrate the dynamic evolution of the solutions over the iterations, highlighting the importance of the proposed approach (solved by the two metaheuristics GA and PSO) compared to the classical method "random selection" (RS). On the other hand, we have based our analysis on the execution time of the solutions, as shown in Figure (4.8). In-depth analysis of these results will allow meaningful conclusions to be drawn about the performance of the proposed approach, which is solved using either the GA or PSO metaheuristics.



(a) tassnu_10_3_3

(b) tassnu_10_3_4

(c) FIVE

(d) Buma14

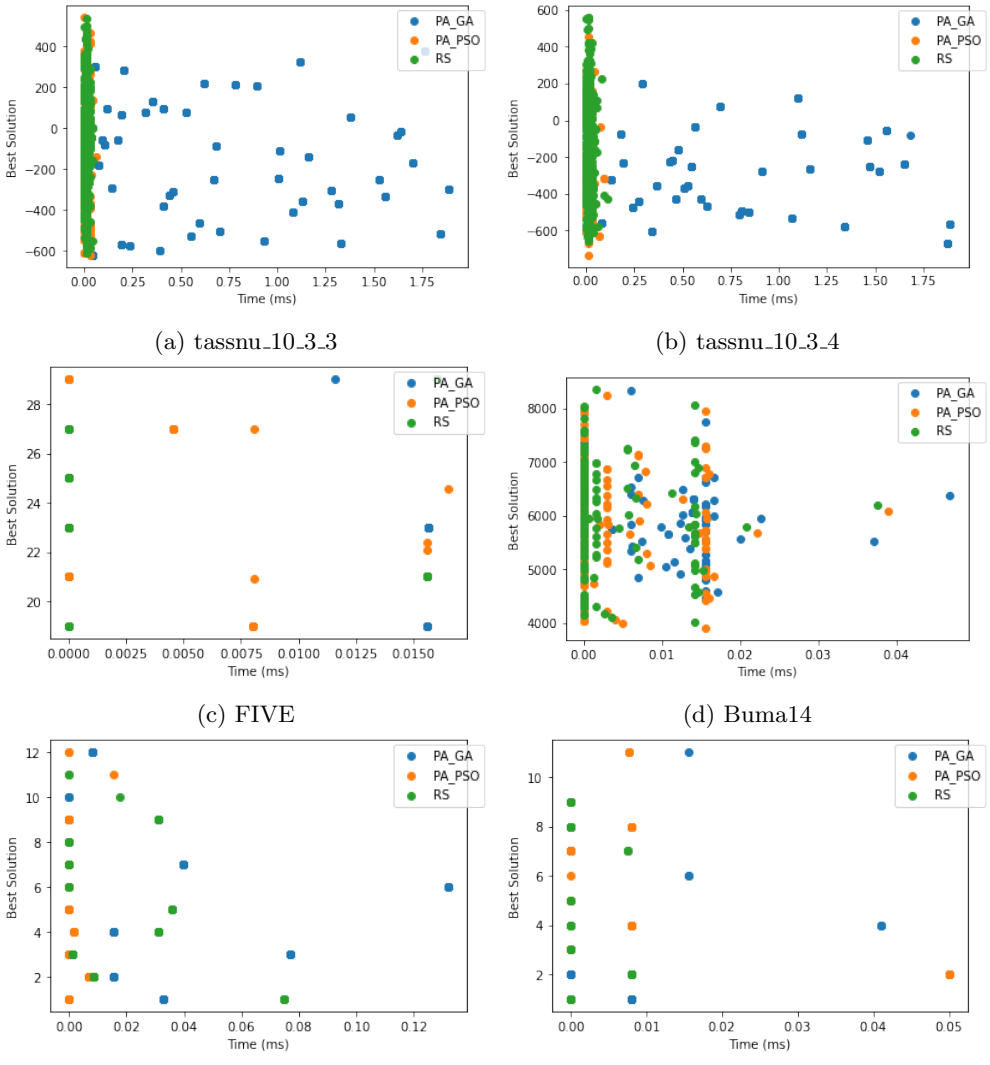
(e) brock200_2

(f) keller4

Figure 4.7: Convergence of feasible solutions using a random selection and the proposed method for some instances of TAP, TSP and MSP

Table 4.7: The percentage of relative quality comparison between the methods used to select HNN hyperparameters

	TSP		TAP		MSP	
	PA_GA	*PA_PSO*	*PA_GA*	*PA_PSO*	*PA_GA*	*PA_PSO*
Random selection	-61.54%	-37.96 %	-6.45%	-5.38%	-25%	-33.82%



(a) tassnu_10_3_3

(b) tassnu_10_3_4

(c) FIVE

(d) Buma14

(e) brock200_2

(f) keller4

Figure 4.8: Convergence of execution time using a random selection and the proposed method for some instances of TAP, TSP and MSP

The Table (4.7) below quantifies the relative quality or improvement, expressed as a percentage, between the approaches used in the experiment, including random selection, *PA_GA* and *PA_PSO*. We observe that the two proposed approaches, (*PA_GA*) and (*PA_PSO*), exhibit higher precision compared to the classical method 'Random selection' for all three problems considered. Specifically, for TSP, *PA_GA* and *PA_PSO* outperform random selection by 61.54% and 37.96% respectively. For TAP, *PA_GA* and *PA_PSO* outperform random selection by 6.45% and 5.38% respectively. Finally, for MSP, *PA_GA* and *PA_PSO* outperform random selection by 25% and 33.82% respectively.

The results of this work also suggest that the optimization of the HNN hyperparameters using the proposed model is significant. The different meta-heuristics used to solve our proposed model obtained good results compared to classical search methods. On the other hand, we can use alternative metaheuristic methods to solve our proposed model, which may give better results compared to those obtained by using the two algorithms we have used. The approach proposed in this work is not limited to the specific combinatorial optimization problems considered in this paper, but it is valid and allows for good results in other applications.

4.4 Tuning HNN with metaheuristic hyperparameter selection

A continuous Hopfield neural network is used in a variety of tasks such as optimization problems. These networks are inspired by biological neural networks and work by minimizing an energy function. The energy function determines the state of the network and is minimized to find optimal solutions to a target problem. Neurons adjust their states to minimize the energy of the whole network, converging on stable states called attractors. The energy function associated with combinatorial optimization problems combines the energy function and its constraints in the form of penalty parameters.

Penalty parameters play a crucial role in fine-tuning the behavior and convergence properties of CHN. These parameters are an integral part of the energy function that governs the dynamics of the network during the learning phases, which are generally determined by the stability method. On the other hand, the hyperparameters associated with the Hopfield network are external variables that play an important role in the convergence of the network, since their values have a profound impact on the overall behavior and performance of the network. These hyperparameters are usually set before the training process and can be adjusted by trial and error or by specialized research methods such as evolutionary research. It is important to understand that hyperparameters influence the network's ability to converge to stable states and to reliably recover stored patterns. Therefore, careful tuning of these hyperparameters is crucial to optimizing the performance and robustness of the Hopfield network in various applications

The performance of Hopfield neural networks is heavily influenced by the careful selection of their hyperparameters. In this study, we delve into various strategies for hyperparameter selection, emphasizing metaheuristic techniques. Notably, we explore the efficacy of Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) within this context. Our primary goal is to assess and compare the performance of these methodologies in pinpointing optimal hyperparameters for Hopfield neural network configurations.

4.4.1 Experimentation and discussion

In this section, metaheuristic methods are used to optimize CHN hyperparameters, including genetic algorithms (GAs), ant colony optimization (ACO), and particle swarm optimization (PSO). We conduct a comparative analysis of these methods in CHN hyperparameter optimization. To achieve this objective, we conducted a comprehensive series of experiments using a diverse range of Graph Coloring Problem (GCP) instances as benchmarks. Through these experiments, we aim to illuminate the relative strengths and weaknesses of metaheuristic methods, namely GA, ACO, and PSO, in the optimization of CHN hyperparameters. Ultimately, our findings are poised to offer valuable insights into enhancing the efficiency and practical

Table 4.8: Obtained result of GCP using HP_GA

Instances	Optimal Solution	Best Solution	Mean	Mode	Sum of Iterations	Total Time
myciel3	4	4	7,59	11	6678	78
MYCIEL4	5	5	13,34	8	3620	71
MYCIEL5	6	6	29,12	17	49704	667
MYCIEL6	7	7	49,83	20	54700	1000
queen5_5	5	5	14,63	9	14888	127
queen6_6	7	7	22,86	34	37898	63,9
queen7_7	7	7	27,22	38	124173	229,4
queen8_8	9	10	37,27	36	14423	146,2
huck	11	11	42,47	43	14004	112,8
jean	10	11	41,36	30	23900	313,1
queen9_9	10	11	42,47	43	14004	112,8
1-FullIns_3	-	7	18,53	23	5810	11,7
2-FullIns_3	-	8	31,19	8	17105	65,6
1-Insertions_4	4	4	33,56	58	57557	17
4-Insertions_3	3	3	41,77	9	2090	179,7
2-Insertions_3	4	5	23,47	28	44145	42,1
3-Insertions_3	4	4	29,47	28	14830	71,5
mugg88 1	4	5	23,47	28	44145	421
mugg88 25	4	4	51,95	82	33870	235,1
queen10_10	11	11	61,36	82	36108	363,6

applicability of Hopfield networks across various real-world scenarios.

Our aim is to thoroughly investigate the effectiveness of each metaheuristic method through rigorous analysis and experimentation. By highlighting their respective strengths and limitations, we aim to provide practical guidance to practitioners and researchers facing similar CHN optimization challenges. This comparative study serves as a road-map for selecting the most appropriate optimization approach tailored to specific optimization problems. This investigation will be carried out through two different studies:

- HP_GA study: This study involves the selection of CHN hyperparameters using GA.
- HP_AC study: The second study, called "HP_AC", uses AC to select CHN hyperparameters.
- HP_PSO Study: Finally, the third study, called "HP_PSO", uses PSO for the selection of CHN hyperparameters.

These studies will allow us to explore the strengths and weaknesses of each approach, providing valuable insights into their applicability and performance in optimizing CHN hyperparameters.

The tables (4.8,4.9,4.10) and the Figures (4.9,4.10,4.11) present the results obtained from the selected GCP instances for the three hyperparameter selection approaches: HP_GA, HP_AC, and HP_PSO, respectively. The results for each approach depend on the following metrics: Optimal Solution: This represents the optimal solution for each instance of GCP. Best Solution: This is the best solution obtained by each approach. Mean: This is the average feasible solution obtained by each approach. Mode: This could refer to the mode or the most common value among the generated solutions. Sum of Iterations: This is the total number of iterations required to find feasible solutions. Total Time: This is the cumulative execution time for each approach.

The selection of CHN hyperparameters by GAs tends to converge to high-quality solutions, but the precision can vary depending on the parameters and the size of the problem. HP_PSO

Table 4.9: Obtained result of GCP using HP_AC

Instances	Optimal Solution	Best Solution	Mean	Mode	Sum of Iterations	Total Time
myciel3	4	4	5	5	1471	62,23
MYCIEL4	5	12	15,33	16	680	54,35
MYCIEL5	6	7	19,34	28	7180	109,46
MYCIEL6	7	9	36,8	23	551	218
queen5_5	5	5	12,45	9	12167	24,46
queen6_6	7	7	17	16	2689	37,57
queen7_7	7	7	19,61	17	4249	146,72
queen8_8	9	9	25,65	18	2401	37,98
huck	11	11	28,6	35	2401	38,29
jean	10	14	34,7	24	2401	92,32
queen9_9	10	11	37,05	34	2401	101,07
1-FullIns_3	-	8	12,4	15	401	2,34
2-FullIns_3	-	9	22,25	20	674	16,94
1-Insertions_4	4	4	29,15	34	487	45,81
4-Insertions_3	3	5	25,6	32	3078	86,65
2-Insertions_3	4	4	13,85	20	230	4,89
3-Insertions_3	4	5	22,35	8	987	22,47
mugg88 1	4	5	35,35	5	2420	135,77
mugg88 25	4	4	35,65	56	2347	131,94
queen10_10	11	18	46,5	63	1253	226,82

Table 4.10: Obtained result of GCP using HP_PSO

Instances	Optimal Solution	Best Solution	Mean	Mode	Sum of Iterations	Total Time
myciel3	4	5	6	7	1615	21,18
MYCIEL4	5	5	11,16	12	1927	317,5
MYCIEL5	6	6	17,84	11	2939	96,04
MYCIEL6	7	7	37,81	18	1123	220,26
queen5_5	5	5	11,33	10	3451	125,11
queen6_6	7	7	16,63	19	3713	35,64
queen7_7	7	10	21,78	29	4523	154,37
queen8_8	9	9	21,95	18	2401	46,29
huck	11	11	28,95	24	2401	86,87
jean	10	13	38,1	47	2401	116,96
queen9_9	10	10	34,25	26	2401	122,57
1-FullIns_3	-	8	12,85	10	400	2,6
2-FullIns_3	-	12	22,65	24	697	19,48
1-Insertions_4	4	7	30,7	36	478	53,62
4-Insertions_3	3	6	32,45	30	3241	104,62
2-Insertions_3	4	4	15,05	22	300	5,6
3-Insertions_3	4	5	24,1	23	1074	25,56
mugg88 1	4	4	26,75	22	2521	174,92
mugg88 25	4	5	31,65	33	2578	180,37
queen10_10	11	12	41	54	1354	292,66

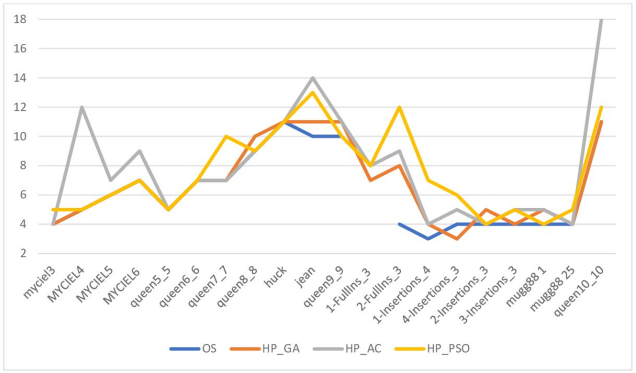


Figure 4.9: Optimal solution of GCP using the three approaches

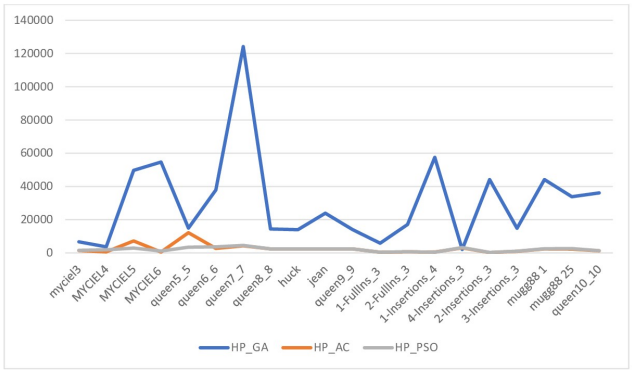


Figure 4.10: Sum of Iterations of GCP using the three approaches

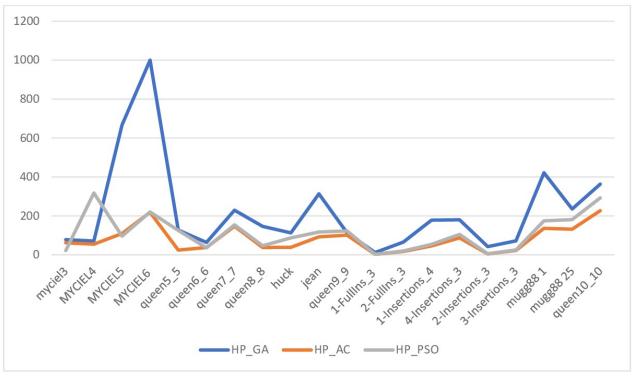


Figure 4.11: Total Time of GCP using the three approaches

can converge quickly to good-quality solutions but can sometimes get stuck in local optima. HP_AC is often effective in finding good-quality solutions but can be sensitive to the parameters and structure of the problem. On the other hand, HP_GA can be slower due to its iterative nature and the need to evaluate many candidate solutions. HP_PSO often converges quickly on solutions, which can reduce the computation time required. HP_AC may be slower due to the nature of its distributed search process.

From the results presented in the tables and figures, we can see that the HP_GA method seems to have a slightly better average performance in most GCP instances, but it is also more prone to significant fluctuations. The HP_PSO method offers good average performance with relatively low computation times in certain instances. HP_AC shows competitive performance with relatively high stability in some instances, although computation times can be longer in some scenarios.

4.5 Conclusion

A Hopfield Neural Network (HNN) is a promising solution to optimization problems, but the difficulty lies in the choice of hyperparameters. Traditional methods often rely on random selection to set these hyperparameters. This chapter presents an innovative approach by formulating a linear optimization problem with constraints. Genetic algorithms and particle swarm optimization are then used to solve this problem, with the aim of selecting hyperparameters that improve the performance of the HNN.

Unlike previous approaches, our method is rooted in a systematic optimization framework. By using genetic algorithms and particle swarm optimization in tandem, we address the challenge of hyperparameter selection in a more strategic manner. Experimental results, obtained through rigorous testing on real-world combinatorial optimization problems, validate the effectiveness of our proposed approach. The synergy of two heuristic hyperparameter selection algorithms significantly improves the overall hyperparameter optimization for the Hopfield neural network.

Chapter 5

Optimizing Tourist Circuits: A Dual Approach Using Hopfield Neural Networks and Innovative Routing Models

5.1 Introduction

Tourism plays a key role in the exploration and appreciation of different cultures and historical landmarks. In this chapter, we delve into the intricacies of optimizing tourist itineraries through a dual approach, using cutting-edge technologies and innovative routing models. The chapter consists of two distinct sections, each addressing unique aspects of tourist route optimization.

In the first, we harness the power of continuous Hopfield networks to create personalized itinerary maps. These neural networks, capable of fast and accurate data analysis, provide an ideal framework for creating itineraries that take into account factors such as the number of tourist areas and distances. By applying the continuous Hopfield network to the historic city of Fez, we aim to estimate optimal distances and generate personalized itineraries to enhance the traveler's overall experience.

The second section addresses the challenge of organizing tours for groups of visitors. Recognizing the shortcomings of previous approaches that prioritized time, distance, or budget objectives independently, we introduce a mathematical programming model with four key objectives: distance, time, budget, and risk factor. Focusing on the city of Fez, we present an intelligent scheduling approach that plans multiple visits to avoid overcrowding at tourist attractions. Our model takes into account individual visitor preferences, incorporates additional criteria related to risk and budget factors, and addresses the intricate structure of Fez's ancient alleyways.

The organization of this chapter is as follows: Section 5.2 presents an approach to the Optimization of Tourist Circuits Using a Hopfield Neural Network: application in the old city of Fez. In Section 5.3, we present the new mathematical model of visitor route planning. Finally, we conclude in Section 5.4.

5.2 Optimization of tourist routes using a Hopfield neural network

The neural network technology that we use in this paper has several advantages over traditional travel planning methods. Neural networks can analyze large amounts of data quickly and accurately, and they can learn from patterns in the data to make better predictions over time. This makes them ideal for creating a personalized travel planning map that takes into account a range of factors, such as the number of tourist areas to be visited, the distance between them, the duration of the visit, and the interests of the individual traveler. To create our travel planning map, we use the continuous Hopfield network, which is a type of neural network that is particularly well-suited for solving optimization problems [31]. The Hopfield neural network uses a mathematical technique called energy minimization to find the most efficient path between different points on the map, taking into account factors such as distance, time, and other constraints. By applying this technique to the tourist areas of Fez, we can estimate the optimal distance required to visit all the tourist areas, and generate a personalized travel plan that maximizes the enjoyment of the traveler.

In this paper, we propose the use of continuous Hopfield networks (CHN) to create a comprehensive tourist itinerary based on the tourist's schedule. In other words, our method aims to find the optimal path that allows tourists to visit all the POIs while respecting their time constraints. We formulate this problem as a variation of the classic traveling salesman problem (TSP), which involves finding the shortest path through a set of given points. By using CHN, we can efficiently solve this problem and generate an optimal tourist map for the city of Fez. In order to evaluate the effectiveness of our approach, we conduct several experiments, which demonstrate the usefulness and practicality of our method in real-world scenarios.

5.2.1 Continuous Hopfield network to optimize tourist routes

Every country in the world has its unique cultural identity that distinguishes it from others. Countries aspire to promote their culture by encouraging tourism in their different cities. However, to develop and promote tourism successfully, it is essential to organize tourist stays and meet their demands while respecting various criteria. This work focuses on optimizing the tour of the points of interest (POIs) chosen by tourists.

Path scheduling is a crucial aspect of tourism planning that involves identifying an optimal route through a set of POIs to ensure that tourists can cover as many POIs as possible during their visit. To achieve this, we adopt the TSP as a mathematical framework for modeling path scheduling. The TSP is a well-known problem in computer science that seeks to find the shortest path through a set of locations visited exactly once. Our objective is to determine the most efficient route that connects all POIs and minimizes the total distance traveled. We represent the mathematical model for finding the optimal routes using a directed connected graph $G = (X, U, W)$:

- X represents the set of POIs to be visited by tourists.
- U is a family of elements in $X \times X$ representing the arcs, where an arc connects two POIs.
- W: $U \to R$ is a function that represents the distance between two POIs connected by an arc.

This approach allows us to find the shortest path that visits all POIs and returns to the starting point, which is the ultimate goal of path scheduling.

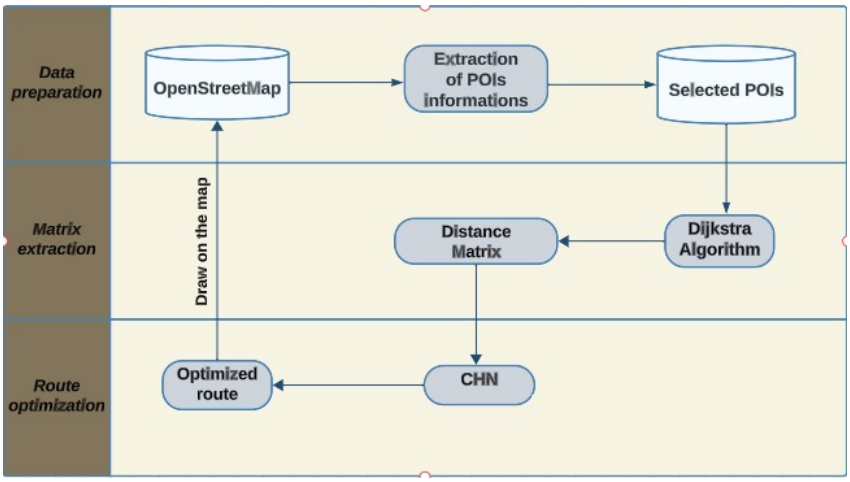


Figure 5.1: Steps of the proposed approach

In 1985, Hopfield and Tank proposed a type of recurrent neural network called the Continuous Hopfield Network (CHN) [131, 19, 21], which has found applications in various fields. This network is characterized by a single layer of n recursively connected neurons [29]. We apply the CHN for optimizing tourist maps. The architecture of the CHN for optimization problems is intricately defined by an energy function that unites the objective function and the constraints inherent in the problem. In our specific context, aimed at optimizing tourist tours, this structure coincides with the classical puzzle known as the Traveling Salesman Problem (TSP).

5.2.2 Proposed approach

In this subsection, we aim to solve the problem of finding the shortest path that connects different points of interest selected by tourists. To achieve this, we have adopted the Continuous Hopfield Network, which is a type of recurrent neural network known for its ability to optimize paths in various fields. The problem of finding the shortest path for the POIs is complex and depends on several factors, such as the distances between the points, the number of points, and the constraints that need to be respected. Therefore, we have identified a set of stages that need to be followed to solve this problem, as illustrated in Figure 5.1. These stages involve prepossessing the data to obtain a set of coordinates representing the POIs, defining the energy function and constraints, and training the CHN to obtain the optimal path. By adopting the Continuous Hopfield Network, we expect to achieve better results in terms of the efficiency and accuracy of the optimized path compared to other optimization techniques. The proposed approach can be useful in various applications, such as tourism planning, logistics, and transportation.

The proposed approach consists of 6 steps, as shown in Figure 5.1:

1. **Select the POIs in OpenStreetMaps**: Tourists select the points of interest they wish to visit.

2. **Extract the distance between POIs using Dijkstra algorithm**: We use Dijkstra algorithm to extract the distance matrix between the selected POIs.

3. **Reformulate the mathematical model**: After obtaining the distance matrix, we reformulate the mathematical model that serves to minimize the distance between these POIs under the constraint of visiting each POI once and only once.

4. **Use the CHN**: We use the CHNk to solve the mathematical model, based on the Talavan algorithm [16], which allows us to minimize the energy function Eq.(3.1) of CHN.

5. **Determine the optimal tours**: The CHN algorithm gives us feasible solutions, from which we extract the best solution that optimizes the energy function of CHN.

6. **Draw the optimal tours and maps**: The final step involves designing the optimal circuits in the maps.

In the next subsection, we will be applying our approach to the city of Fez, which is renowned for its rich cultural heritage and is one of Morocco's most popular tourist destinations.

5.2.3 Experimentation

In this subsection, we used CHN to optimize tourist routes in the city of Fez, which is well-known for its diversity of tourist zones and historical monuments. Here, we have utilized the algorithm proposed by [16] to obtain the equilibrium point of the CHN. As mentioned earlier, our goal of finding an optimal tourist map for the tourist zones in Fez can be viewed as a TSP. Fez is a city that has many tourist places; we present in our experimental study only 40 selected zones from the city of Fez, Morocco, which are presented in table 5.4.

Table 5.1: Test POI

POI_ID	POI_Name	POI_ID	POI_Name
109	Bab Mahrouq	159	Rue Fes Jdid
530	Synagogue ibn danan	1373	Bab Boujloud
1392	Bibliotheque de La Qaraouiyine	1422	Place Moulay Hassan
1472	Jamaa Al Hamra	1575	Bab el Guissa
1992	Palais Abdeslam Moqri	1999	Dar Pacha Tazi
2133	Souk Ain Allou	2288	ouk El Attarine
2794	Bab Ftouh	2775	Place Seffarine
3065	Talaa Sghira	3194	Riad Driss Moqri
3273	Musee Batha	3278	Mosquée el Oued
3387	Synagogue Al Fassiyine	539	Cimtiere juif
3635	Palais Mnbhi	3832	Bab Riafa
3834	Jardin Jnane Sbil	4028	Université Al Quaraouiyine
2197	Souk El Henna	4397	Zaouiat Sidi Ahemd Tijani
4701	Méderssa Attarine	4732	Méderssa Misbahiya
4873	Talaa Kbira	4787	Ateliers de broderie de Tissage
4935	Mausole Moulay Driss	1011	Bab Sagma
2558	Dar ba Mohammed Chergui	1383	Complex(madrassa) Bou'Inania
2294	Borj Nord	3036	Borj Sud
2742	Musée Ennejarin	1634	Mosquée Rssif
2559	Hammam Bousouifa	2803	Mausole Moulay Driss

We present different categories of the problem of finding the optimal route between points of interest in the Medina of Fez. Firstly, we present the number of feasible solutions (NFS) and the optimal routes obtained in the selected number of executions (NE) for each instance, as shown in Table (5.2). Secondly, we present the minimum results obtained from different instances (BSF)) in this study, along with the required execution time (NI) and the number of iterations associated with the optimal solutions (ST) for each instance of the selected points of interest.

Figures (5.2, 5.3) present a comparison between our proposed approach to tour optimization using the continuous Hopfield network and other methods such as genetic algorithms (GA) and

Table 5.2: Performance metrics for optimal routing in the medina of Fez

Scenarios	NE	NFS	BSF	NI	ST	Route
5_1	1000	806	13.29	14	12.69	[2294-1373- 2742-2558-2803]
5_2	1000	815	5.46	22	5.53	[1999-159-4397-4732-2133]
7_1	1000	806	11.36	47	42.72	[3273-4873-2294-1383-2803-2558-530]
7_2	1000	826	5.07	36	45.77	[2294-4873-3635-3387-539-3834-3194]
9_1	1000	841	11.58	128	100.19	[1575-1383-2803-2794-3635-530-3065-2294-4028]
9_2	1000	888	7.64	107	106	[2133 -157-4397-3832-3194-1999-2794-1392-2775]
11_1	1000	904	11.78	73	273.65	[2775-4028-2133-4397-1383-2794-1992-1472-530-3273-1373]
11_2	1000	892	12.46	354	278	[539-3387-1472 -1422 -3832-3273-4701-2288 -1011-2803-3036]
13_1	1000	880	16.81	86	431.95	[3065-2197-1383-2794-2803-4732-2133-3635-530-2742-1422-3832-3273]
13_2	1000	906	15.05	180	814.32	[2742-1634-4397-4787-2294-2794-2775-4701-1392-4873-1373-3273-3194]

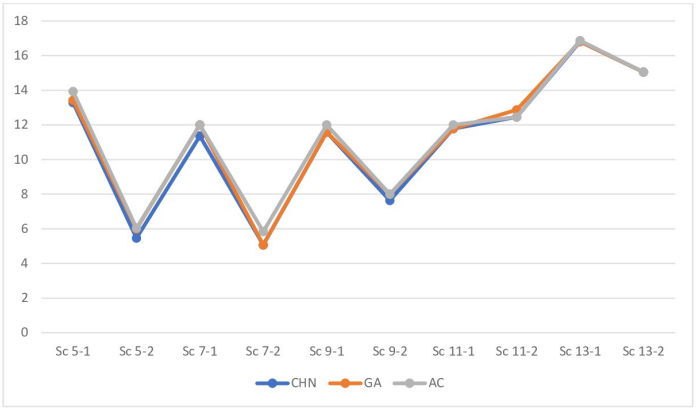


Figure 5.2: Accuracy of treated scenarios using different approaches

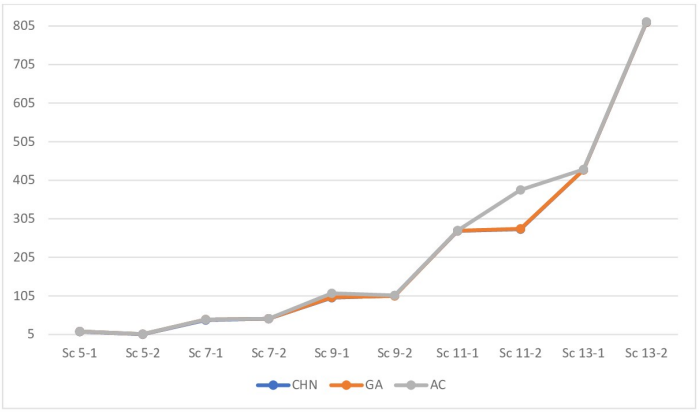


Figure 5.3: Execution time in (ms) of scenarios treated using different approaches

ant colonies (AC). The figures illustrate the relative performance of each method in terms of the quality of tours generated, the convergence time, and the number of iterations required to converge. This comparative analysis allows us to assess the effectiveness and applicability of our approach compared to existing techniques in the field of tour optimization.

In Figure (5.2), we present the quality of the solutions obtained (shortest path) for the tourist routes using our approach based on the CHN, as well as other approaches based on metaheuristic methods. We observe that the approach proposed in this paper presents an optimal quality compared to the other approaches. The tourist routes generated by our approach are characterized by efficient paths and minimized distances traveled, which guarantees an optimal tourist experience for travelers. Compared to metaheuristic methods such as GA or AC, our approach based on the CHN has demonstrated its ability to find high quality solutions with fast convergence and increased efficiency. This observation underlines the effectiveness and relevance of the proposed approach in the context of tourist route optimization, offering a promising alternative for the design and planning of tourist itineraries. The high quality of the solutions obtained reinforces the credibility and added value of our approach to solving complex optimization problems in the field of tourism.

Figure (5.3) illustrates the execution time results obtained by the approaches to solving the tourist circuit planning problem. We observe that our itineraries, using the CHN approach proposed in this paper, have a faster average convergence time compared to genetic algorithms and ant colonies. The increased efficiency of the CHN compared to other methods such as GAs and ACs can be attributed to several factors. First, the CHN benefits from its ability to converge quickly to stable states, which allows optimal solutions to be reached efficiently. Unlike genetic approaches, which require exhaustive exploration of the search space, and AC, which rely on stochastic search processes, the CHN uses an iterative process of updating neuronal states, which often converges to high-quality solutions more quickly. Thus, the simplicity of the CHN architecture and its ability to avoid local minima allow faster and more reliable convergence, even for problems of great complexity.

In this first section, we demonstrated the application of neural networks, particularly the CHN, to enhance and plan tourist circuits around selected points of interest. This approach has enabled us to promote and advance tourism in the city of Fez while improving the mobility of tourists visiting different tourist sites in an organized manner. The utilization of CHN to optimize the optimization problems has multiple benefits, including minimal execution time compared to other optimization methods. However, our current work only focuses on optimizing circuits based on distance. In next section, we plan to incorporate additional constraints, such as time and budget, to optimize tourist circuits in the city of Fez further.

5.3 A new optimization model of the visitor routing problem applied to smart tourism

Most of the approaches proposed in the past have tackled the time, distance, or budget objectives independently, which could result in prioritizing one objective over the others. Furthermore, all previous works in this field have studied the routing problem for a single group of visitors, which is insufficient in today's context, particularly for authorities responsible for managing visitor group assignments. In this section, we aim to develop a mathematical programming model for organizing tours for visitor groups while prioritizing their needs and constraints. In the first step, each visitor provides personal information, points of interest to visit, and constraints, which are then used to build a mathematical program with four objective functions: distance, time, budget, and risk factor. The first objective function involves optimizing the tour distance for each group of tourists, ensuring they visit the selected points of interest (POIs) in the most efficient way possible. The second objective function is related to the tour duration, taking into account any potential traffic jams or extended travel times. The third objective is the budget, allowing visitors to experience a complete visit with minimal costs.

Safety is another objective introduced in the model, as tours may be affected by risk factors, such as traffic jams or public works. The proposed model considers the inconsistency between the distance traveled by a group of visitors between two POIs and the time consumed between the same two points, which may be due to traffic jams.

To properly develop and test our proposed model, we have chosen to apply it to the country of Morocco. Morocco has a rich tourism history, with its unique combination of African and Arab cultures and customs attracting many visitors [67, 77]. Specifically, we will focus our study on Fez, which is the oldest of the imperial cities and one of the most significant tourist destinations in Morocco. The main objectives of this section are listed as follows:

- The first objective is to present a smart scheduling approach for organizing multiple visits to avoid overcrowding of tourists at the same POI. In this context, we propose a new mathematical model for planning the visits of multiple groups of visitors instead of a single group as typically used in the literature.
- The second objective is to incorporate additional criteria related to the risk and budget factors while also considering the preferred POIs of the visitors.

5.3.1 Proposed mathematical model

In this section, we propose a new mathematical programming model to plan visits for multiple groups of visitors who have selected a number of POIs to visit. This model aims to generate the most satisfactory visits based on four main criteria: distance, time of visits, risk, and budget.

The VRP for groups of visitors can be formulated as an optimization problem, which can be solved using various optimization techniques. The following sets are used in the mathematical modeling that implements the VRP module:

- N is the set of groups.
- M is the set of permutations.
- $V = \{i : i \in P_j, j \in N\}$, where P_j is the set of POIs that group j wishes to visit.
- $A = \{a_{ij}\}$ be the set of arcs that connect different POIs, where $a_{ij} \neq 0$ if there exists a direct path between POI i and j, without passing through other POIs.
- $T^{max} = \{T_i^{max} : i \in N\}$ is the set of maximum allowed time periods for each visitor group i.
- $T^{min} = \{T_i^{min} : i \in N\}$ is the set of minimum allowed time periods for each visitor group i.
- $t = \{t_{ih} : i \in N, h \in V\}$ represents the time allotted for group i to visit POI h.
- $D = \{d_{ij} : (i, j) \in A\}$ is the set of distances between two POIs; d_{ij} : the distance between POIs i and j.
- $T = \{t_{hl} : (h, l) \in A\}$ denotes the time spent by a group of visitors on the path (h, l), such that $t_{hl} \in [T_{hl}^{min}, T_{hl}^{max}]$. Here, T_{hl}^{max} is the maximum time and T_{hl}^{min} is the minimum time needed for a group of visitors to walk the path (h, l), taking into account factors such as traffic, path condition, stairs, construction work, etc.
- $O = \{o_{ij} \in [\,0, 1\,] : (i, j) \in A\}$ represents the risk factor of each path (i, j), with o_{ij} being a value between 0 and 1 that indicates the difficulty for a group of visitors to walk on that path.

- $C = \{C_{hl} : (h,l) \in A\}$ is the cost of the transport required to go from POI h to l.
- $C^{max} = \{C_i^{max} : i \in N\}$ represents the maximum budget proposed by each visitor group that must not be exceeded.

1. **The variables**

   (a) **Decision variables**

   $$X_{ij}^k = \begin{cases} 1 & if\ group\ i\ visits\ POI\ j\ in\ permutation\ k \\ 0 & otherwise \end{cases} \qquad \forall i \in N, j \in V, k \in M$$

   (b) **Artificial variables**

   - Path variable

     $$Y_{hl}^i = \begin{cases} 1 & if\ group\ i\ visit\ the\ path\ (h,l) \\ 0 & otherwise \end{cases} \qquad \forall i \in N, (h,l) \in A$$

   - Time allowed in each POI

     $t = \{t_{ih} \in IR_+^* : i \in A, h \in V\}$ is the time allotted to each visitor group i to visit each point of interest h.

2. **Objective function**

   The objective function (o.f) is defined as the satisfaction of a group of visitors desires through four metrics on which we base our work. These metrics include minimizing the total length of the tour of the preferred POIs (i.e., finding the optimal routes to visit the preferred POIs for each group of visitors), minimizing the total travel time that combines the time between paths and the time to visit each POI so that the total time to visit all POIs in each group is included as much as possible in the predetermined expected visit time for each group, minimizing the risk factor as much as possible because a higher risk factor for a route for many reasons (safety, obstacles...) increases the likelihood that the route will be blocked, and minimizing the total budget, i.e., planning each visit of the visitor group so that its total budget is as close as possible to the predetermined maximum budget for each group. By considering these four metrics in our (o.f), we aim to provide the most satisfactory experience for the group of visitors. The multi-objective function consists of the following terms:

   (a) Term for the total length of the route :

   $$\sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^k d_{hl} X_{il}^{k+1} \qquad \forall i \in N \tag{5.1}$$

   The non-linear term (o.f. 5.1) minimizes the total distance traveled by each group of visitors, ensuring that each group follows a route with minimal distance between the POIs.

   (b) Physical distance term :

   $$\sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^k t_{hl} X_{il}^{k+1} + \sum_{k \in M} \sum_{s \in V} t_{is} X_{is}^k \qquad \forall i \in N \tag{5.2}$$

   The non-linear term (o.f. 5.2) minimizes the total travel time for each group of visitors, which includes both the time required to travel between selected POIs and the time allocated to visit each POI for each group.

(c) Budget term :

$$\sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^{k} C_{hl} X_{il}^{k+1} \qquad \forall i \in N \tag{5.3}$$

This term (o.f. 5.3) controls the amount of budget consumed by each group of visitors when they use modes of transportation to travel between the POIs that they have selected to visit.

(d) Safety term :

$$\sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^{k} o_{hl} X_{il}^{k+1} \qquad \forall i \in N \tag{5.4}$$

The term (o.f. 5.4) is designed to penalize routes with a higher risk factor, allowing our model to plan the tour of each group by minimizing overall risk.

3. **Constraints**

The constraints associated with the proposed model are:

(a) Initial constraint :

$$\sum_{i \in N} X_{ij}^{k} = 1 \qquad \forall j \in V, k \in M \tag{5.5}$$

The constraint (c.5.5) guarantees that each POI of each permutation is visited by at most one group of visitors.

(b) Tour start constraints :

$$\sum_{j \in V} X_{ij}^{k} = 1 \qquad \forall i \in N, k \in M \tag{5.6}$$

The constraint (c.5.6) ensures that each permutation for each visitor group must contain exactly one POI.

(c) Constraints of the visit :

$$\sum_{k \in M} X_{ij}^{k} \geq 1 \qquad \forall i \in N, j \in V \tag{5.7}$$

The constraint (c.5.7) ensures that each group of visitors must visit each POI at least once.

(d) Constraint of POI :

$$\sum_{h \in V} Y_{hl}^{i} \geq 1 \qquad \forall i \in N, l \in V \tag{5.8}$$

The constraint (c.5.8) requires that each group of visitors visit each POI at least once.

(e) Tour duration constraint :

$$T_i^{min} \leq \sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^{k} t_{hl} X_{il}^{k+1} + \sum_{k \in M} \sum_{s \in V} t_{is} X_{is}^{k} \leq T_i^{max} \qquad \forall i \in N \tag{5.9}$$

The constraint (c.5.9) requires that the total visit time $T + t$ of each group $i \in N$ be within the predetermined minimum T_i^{min} and maximum T_i^{max} expected total visit time for each group.

(f) Time constraint of the path :

$$T_{hl}^{min} \leq Y_{hl}^{i} t_{hl} \leq T_{hl}^{max} \qquad \forall i \in N, (h,l) \in A \tag{5.10}$$

The constraint (c.5.10) requires each group $i \in N$ to travel a path $(h,l) \in A$ at a given time t_{hl} without exceeding the maximum time T_{hl}^{max} and also without being less than the minimum time T_{hl}^{min}.

(g) Time constraint :

$$\sum_{l \in V} X_{il}^{k} t_{il} \leq \sum_{j \neq i \in N} \sum_{h \in V} X_{jh}^{k+1} (\sum_{l \in V} X_{jl}^{k} t_{jl} + t_{lh}) \qquad \forall i \in N, k \in M-1 \qquad (5.11)$$

The constraint (c.5.11) ensures that each group $i \in N$ leaves a visited POI before the next group arrives at the same POI, thus avoiding any waiting time for the next group to start their visit.

(h) budget constraint :

$$\sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^{k} C_{hl} X_{il}^{k+1} \leq C_{i}^{max} \qquad \forall i \in N \qquad (5.12)$$

The constraint (c.5.12) ensures that the total cost of the tour for each group $i \in N$ does not exceed the maximum expected budget C_i^{max} predetermined for that group.

(i) Variable constraints :

$$X_{ij}^{k} \in \{0,1\} \qquad \forall i \in N, j \in V, k \in M \qquad (5.13)$$

$$Y_{hl}^{i} \in \{0,1\} \qquad \forall i \in N, (h,l) \in A \qquad (5.14)$$

$$t_{il} \in IR_{+}^{*} \qquad \forall i \in N, l \in V \qquad (5.15)$$

The constraint (c.5.13) enforces binary decision variables, which can only take the values of 0 or 1. The constraints (c.5.14) ensure that the artificial variables are also binary. Lastly, the constraints (c.5.15) require all decision and artificial variables to have positive real values.

The following mixed nonlinear programming model consolidates all the aspects discussed previously.

$$(P) \begin{cases} Min \sum_{i \in N} \sum_{k \in M-1} (\sum_{(h,l) \in A} X_{ih}^{k} d_{hl} X_{il}^{k+1} + \sum_{(h,l) \in A} X_{ih}^{k} t_{hl} X_{il}^{k+1} + \\ \qquad \sum_{s \in V} t_{is} X_{is}^{k} + \sum_{(h,l) \in A} X_{ih}^{k} C_{hl} X_{il}^{k+1} + \sum_{(h,l) \in A} X_{ih}^{k} o_{hl} X_{il}^{k+1}) \\ s.c \\ \sum_{i \in N} X_{ij}^{k} = 1 \qquad \forall j \in V, k \in M \\ \sum_{j \in V} X_{ij}^{k} = 1 \qquad \forall i \in N, k \in M \\ \sum_{k \in M} X_{ij}^{k} \geq 1 \qquad \forall i \in N, j \in V \\ \sum_{h \in V} Y_{hl}^{i} \geq 1 \qquad \forall i \in N, l \in V \\ T_{i}^{min} \leq \sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^{k} t_{hl} X_{il}^{k+1} + \sum_{k \in M} \sum_{s \in V} t_{is} X_{is}^{k} \leq T_{i}^{max} \quad \forall i \in N \\ T_{hl}^{min} \leq Y_{hl}^{i} t_{hl} \leq T_{hl}^{max} \qquad \forall i \in N, (h,l) \in A \\ \sum_{l \in V} X_{il}^{k} t_{il} \leq \sum_{j \neq i \in N} \sum_{h \in V} X_{jh}^{k+1} (\sum_{l \in V} X_{jl}^{k} t_{jl} + t_{lh}) \quad \forall i \in N, k \in M-1 \\ \sum_{k \in M-1} \sum_{(h,l) \in A} X_{ih}^{k} C_{hl} X_{il}^{k+1} \leq C_{i}^{max} \qquad \forall i \in N \\ X_{ij}^{k} \in \{0,1\} \qquad \forall i \in N, j \in V, k \in M \\ Y_{hl}^{i} \in \{0,1\} \qquad \forall i \in N, (h,l) \in A \\ t_{il} \in IR_{+}^{*} \qquad \forall i \in N, l \in V \end{cases} \qquad (5.16)$$

5.3.2 Marginalization of proposed VRP

To show the significance and efficacy of the proposed VRP (Eq. 5.16), we have divided it into four models, each associated with a specific criterion under study. The resulting mathematical programming models are as follows:

- **Model 1**: This model is focused on minimizing the distance traveled by visitor groups, in order to find an optimal tour.

$$(P_1)\begin{cases} Min \quad \sum_{i\in N}\sum_{k\in M-1}\sum_{(h,l)\in A} X_{ih}^k d_{hl} X_{il}^{k+1} \\ s.c \\ \qquad \sum_{i\in N} X_{ij}^k = 1 \qquad \forall j \in V, k \in M \\ \qquad \sum_{j\in V} X_{ij}^k = 1 \qquad \forall i \in N, k \in M \\ \qquad \sum_{k\in M} X_{ij}^k = 1 \qquad \forall i \in N, j \in V \\ \qquad X_{ij}^k \in \{0,1\} \qquad \forall i \in N, j \in V, k \in M \end{cases} \tag{5.17}$$

  The objective of this model is to minimize the total distance traveled by each group of visitors while visiting all selected POIs exactly once. The associated objective function aims to minimize the sum of distances between POIs for each group of visitors. The constraints ensure that each POI is visited by at most one group, and each group visits each POI exactly once.

- **Model 2**: This model is related to the time criterion and is used to minimize the time term, which includes the travel time between different POIs and the time spent at each selected POI to be visited by each group.

$$(P_2)\begin{cases} Min \quad \sum_{i\in N}\sum_{k\in M-1}\sum_{(h,l)\in A} X_{ih}^k t_{hl} X_{il}^{k+1} + \sum_{k\in M}\sum_{s\in V} t_{is} X_{is}^k \\ s.c \\ \sum_{i\in N} X_{ij}^k = 1 \qquad \forall j \in V, k \in M \\ \sum_{j\in V} X_{ij}^k = 1 \qquad \forall i \in N, k \in M \\ \sum_{k\in M} X_{ij}^k = 1 \qquad \forall i \in N, j \in V \\ T_i^{min} \leq \sum_{k\in M-1}\sum_{(h,l)\in A} X_{ih}^k t_{hl} X_{il}^{k+1} + \sum_{k\in M}\sum_{s\in V} t_{is} X_{is}^k \leq T_i^{max} \quad \forall i \in N \\ \sum_{l\in V} X_{il}^k t_{il} \leq \sum_{j\neq i\in N}\sum_{h\in V} X_{jh}^{k+1}(\sum_{l\in V} X_{jl}^k t_{jl} + t_{lh}) \qquad \forall i \in N, k \in M-1 \\ X_{ij}^k \in \{0,1\} \qquad \forall i \in N, j \in V, k \in M \\ t_{il} \in IR_+^* \qquad \forall i \in N, l \in V \end{cases} \tag{5.18}$$

  The objective function is to minimize the sum of the time required for traveling between different POIs and the time spent at each selected POI. The constraints ensure that the visits of each group are planned within the minimum and maximum time intervals provided for each group.

- **Model 3**: This model is associated with the criterion of safety; it allows for the minimization of the risk factor associated with each path taken by the visitor groups. The risk factor may depend on various factors such as security measures, the number of visitors in

the group, and the popularity of the POI.

$$(P_3)\begin{cases} Min \quad \sum_{i\in N}\sum_{k\in M-1}\sum_{(h,l)\in A} X_{ih}^{k} o_{hl} X_{il}^{k+1} \\ s.c \\ \quad \sum_{i\in N} X_{ij}^{k} = 1 \qquad \forall j \in V, k \in M \\ \quad \sum_{j\in V} X_{ij}^{k} = 1 \qquad \forall i \in N, k \in M \\ \quad \sum_{k\in M} X_{ij}^{k} \geq 1 \qquad \forall i \in N, j \in V \\ \quad \sum_{h\in V} Y_{hl}^{i} \geq 1 \qquad \forall i \in N, l \in V \\ \quad Y_{hl}^{i} \in \{0,1\} \qquad \forall i \in N, (h,l) \in A \\ \quad X_{ij}^{k} \in \{0,1\} \qquad \forall i \in N, j \in V, k \in M \end{cases} \tag{5.19}$$

The objective function associated with this model allows us to find the paths with the lowest risk factor, and the constraints consist of planning the visits of each group while taking into account the maximum allowable risk factor for each group. In order to minimize the risk, the model may penalize paths that are considered more risky, which could result in allowing a group to visit a POI that has already been visited.

- **Model 4**: This model is associated with the budget criterion, and its objective is to minimize the total cost of each group's visit.

$$(P_4)\begin{cases} Min \quad \sum_{i\in N}\sum_{k\in M-1}\sum_{(h,l)\in A} X_{ih}^{k} C_{hl} X_{il}^{k+1} \\ s.c \\ \quad \sum_{i\in N} X_{ij}^{k} \leq 1 \qquad \forall j \in V, k \in M \\ \quad \sum_{j\in V} X_{ij}^{k} \leq 1 \qquad \forall i \in N, k \in M \\ \quad \sum_{k\in M} X_{ij}^{k} \leq 1 \qquad \forall i \in N, j \in V \\ \quad \sum_{k\in M-1}\sum_{(h,l)\in A} X_{ih}^{k} C_{hl} X_{il}^{k+1} \leq C_i^{max} \qquad \forall i \in N \\ \quad X_{ij}^{k} \in \{0,1\} \qquad \forall i \in N, j \in V, k \in M \end{cases} \tag{5.20}$$

The objective function of this model aims to minimize the total cost of each group's visit, taking into account the budget constraint. The constraints ensure that the planned visits for each group of visitors do not exceed their respective maximum budget limits.

The four models are specific instances of the same general proposed VRP, where the objective function and constraints are adjusted to optimize different criteria.

5.3.3 Simulated annealing algorithm

The heuristic methods are effective in solving complex optimization problems, such as the proposed VRP. For evaluating the performance of our approach, we use a popular heuristic technique called simulated annealing (SA). This technique is known to produce good results for problems of large size [70] The SA algorithm starts with a random initial solution and then searches for a new solution at each iteration. If the new solution is better than the previous one, it is taken as the current solution; otherwise, the previous one remains the current solution [86, 87].

The SA algorithm used to solve the proposed VRP, which is detailed in Algorithm 5, consists of two main steps: data initialization and solution improvement. In the first step, the problem data (line 1) and SA parameters (line 2) are defined, followed by the random initialization of

Algorithm 5 SA algorithm to generate a VRP

1: Input Data
2: Initialize $(Temp, \alpha)$
3: Generate a random solution X
4: $X_{best} = X$
5: **while** Temp $> 10^{-8}$ **do**
6: $X_{curr} = X_{best}$
7: Generate a random $(i, j \in V)$
8: $X_{curr}[i], X_{curr}[j] \longleftarrow X_{curr}[j], X_{curr}[i]$
9: $\triangle = f(X_{curr}) - f(X_{best})$
10: **if** $\triangle < 0$ **then**
11: $X_{best} = X_{curr}$
12: **end if**
13: **if** $\triangle \geq 0$ **then**
14: **if** Generating a random number $\in [0, 1] < exp(-\triangle/temp)$ **then**
15: $X = X_{curr}$
16: **end if**
17: **end if**
18: Temp $* = \alpha$
19: **end while**

a solution that represents the tour of the visitor groups (line 3). In the second step, at each iteration, a new solution is generated, and it becomes the current solution if it is better than the previous one and the objective function value of the current solution $f(X_{curr})$ is equal to the objective function value of the best solution found so far $f(X_{best})$. If the new solution is worse, it may still be accepted with a certain probability, which depends on a temperature parameter that decreases over time (line 5-19).

5.3.4 Optimizing tourist visits in Fez using the proposed VRP

The proposed model can be effectively applied to a variety of situations, locations, and smart tourism issues. In this study, we have specifically applied the proposed model to the city of Fez.

1. **Tourism in the city of Fez**

   Tourism is a rapidly growing economic sector and a potential driver of rapid national development. It has become an area of great interest for scientists and researchers alike. With the advancement of scientific research, a focus has been placed on the growth of the commercial tourism sector. Morocco is regarded as one of the countries with a rich tourism history due to its unique combination of African and Arab cultures, as well as their customs, which attract tourists from all over the world [67, 77]. In this study, we will focus on a city in Morocco that is considered the oldest of the imperial cities, located in the northern part of the country and is one of the most important tourist destinations: the city of Fez. In 1981, the historical medina of Fez was added to UNESCO's list of world heritage sites [71, 79]. In recent decades, tourism in Fez has been growing continuously and has become an essential economic factor for the city. Fez is known for its rich heritage, including bazaars, castles, ramparts, districts, mosques, mausoleums, museums, fortifications, gates, palaces, and fountains.

   In the data section (1.6), a complete set of data was presented, including the essential features for modeling the problem. This data includes the names of the POIs, their geographical locations, their categories, and the distances and travel times between the different POIs.

2. **Illustrative example**

We use an illustrative example composed of 4 POIs $\{1, 17, 35, 59\}$ and 4 groups of visitors $\{G_1, G_2, G_3, G_4\}$ to compare the results obtained for the models that we have developed for different criteria. Figure 5.4 shows an abstract graph of our example data, where nodes represent the POIs to visit for each group of visitors. The line connecting two POIs (i, j) represents the path based on four criteria: distance (d_{ij}), travel time (t_{ij}), budget (c_{ij}) if a transport mode can be used, and risk factor (o_{ij}).

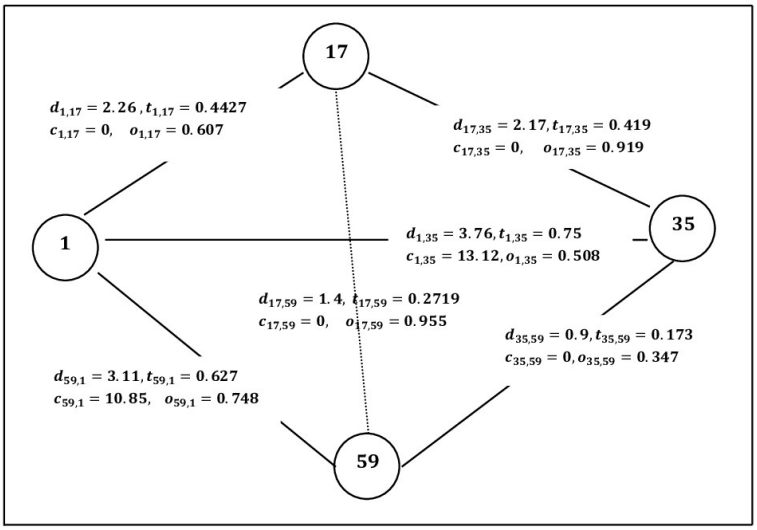


Figure 5.4: Representative graph of 4 POIs

Table 5.3: The paths obtained by each model

	Groups			
Models	G_1	G_2	G_3	G_4
Model 1	17-59-35-1	35-1-17-59	59-35-1-17	1-17-59-35
Model 2	35-17-1-59	17-35-59-1	59-1-17-35	1-59-35-17
Model 3	59-**35**-17-1-35	1-35-17-**1**-59	35-1-59-**35**-17	17-**1**-59-35-1
Model 4	1-17-35-59	35-59-1-17	17-1-59-35	59-35-17-1
Our Model	1-59-35-17	35-17-1-59	17-1-59-35	59-35-17-1

The plans depicted in Table 5.3 represent the solutions obtained by different models. The Table 5.4 presents the results obtained in this application example. Each model (1, 2, 3, 4) minimizes a specific criterion (distance, duration, risk, and budget) and we have also calculated the other criteria associated with the solution obtained by each model. For example, for model 1, we obtained an optimized solution for the total distance of the tour for the four groups and then calculated the total duration, total risk, and total budget associated with this solution.

Table 5.4: Solution for each model

Models	Total distance	Total duration	Total risk	Total budget
Model 1	33.28	-	7.258	52.489
Model 2	33.76	10.433	8.5838	43.415
Model 3	36.0	-	4	95.905
Model 4	46.309	-	11.189	43.415
Our model	33.76	9.424	7.8705	43.415

Regarding Table 5.3, the path $(17-59)$ has a higher risk factor ($o_{17,59} \approx 1$). For group G_1, it takes the path $(59, 35, 17)$ instead of the risky path $(59, 17)$, so this group visits POI 35 twice: the first time to avoid the risky path (time spent at this POI is 0), and the second time to visit it (time spent at this POI is different from 0). For G_2 and G_4, POI 1 is visited twice to avoid the risky path $(17, 59)$. For group G_3, POI 35 is visited twice: the first time to visit it (time spent at this POI is different from 0). In the second time, G_3 uses POI 35 to avoid the risky path (time spent at this POI is 0).

Based on the results presented in Table 5.4, the proposed model outperforms the other models in terms of the average of all criteria, with a value of 2.6605 for model 1, 1.437 for model 2, 16.95 for model 3, and 5.28 for model 4.

5.3.5 Results and discussion

The proposed model aims at guiding groups of visitors to visit a set of preferred POIs while proposing an optimal route that minimizes risk under constraints of distance, time, and budget. This mathematical model takes into account the personal preferences and needs of visitors. In this section, to apply the proposed model, we test it on the tourism sector in Fez, Morocco. We use Algorithm 5 to solve all the models in the experimental study.

We conducted 5 validation tests using randomly selected POIs, as presented in Table 5.5, to examine the proposed model's ability to optimize visitor group itineraries. In this study, we compare and analyze the performance of the proposed model to demonstrate its relevance. We evaluate the model's performance based on four criteria: TD represents the total minimum distance traveled by all groups of visitors, TT represents the total minimum travel time for all groups (including time spent between different paths and at each POI during their visit), TO represents the total minimum risk for all groups, and TC represents the minimum budget for all groups.

Table 5.5: Scenarios used in our study

scenarios	Number of POIs	Group number	Selected POIs	Forbidden paths	$[T_{min}, T_{max}]$	C_{max}
scenarios 1	4	4	$\{1, 17, 15, 27\}$	$\{17-27\}$	[2,4]	80
scenarios 2	5	5	$\{3, 10, 35, 44, 51\}$	-	[3,4.5]	90
scenarios 3	6	6	$\{7, 10, 17, 27, 37, 49\}$	$\{17-49\}$	[3, 5.5]	90
scenarios 4	7	7	$\{0, 4, 12, 15, 23, 35, 39\}$	$\{15-39\}$	[3, 6.5]	95
scenarios 5	8	8	$\{63, 3, 44, 51, 54, 4, 17, 59\}$	$\{63-3\}$; $\{63-44\}$; $\{44-54\}$	[4, 7.5]	90

To demonstrate the importance and effectiveness of our proposed model, we compared it with four other mathematical models that are based on individual criteria used in our proposed model. Specifically, we compared our proposed model that considers all the criteria with each criterion independently. The Table 5.6 present the results of each model associated with each scenario. In addition, we calculated the other criteria associated with the solution obtained by each model (1,2,3,4). For example, in Model 2, we obtained an optimized solution for the

TT of the tour for the groups of visitors, and then we calculated the TD, the TO, and the TC associated with this solution.

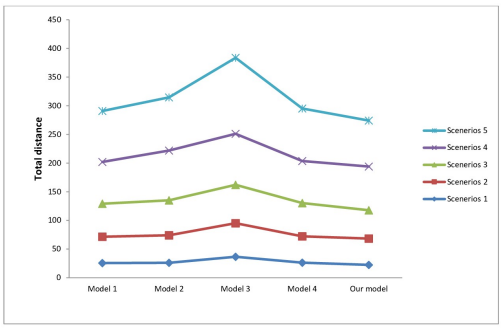
Table 5.6: Obtained results using our model and the other models

scenarios	Models	TD	TT	TO	TC
Scenarios 1	Model 1	25.64	-	8.20	39.65
	Model 2	26	10.34	7.899	38.74
	Model 3	36.63	-	5.83	99.08
	Model 4	26.36	-	7.49	37.83
	Our model	22.47	10.19	7.72	38.74
Scenarios 2	Model 1	45.87	-	12.86	84.84
	Model 2	48.06	16.75	11.69	117.93
	Model 3	58.56	-	8.53	176.45
	Model 4	45.87	-	13.48	84.84
	Our model	45.87	15.79	12.85	84.84
Scenarios 3	Model 1	57.64	-	12.91	39.16
	Model 2	61.08	23.91	11.18	58.74
	Model 3	66.93	-	9.24	80.41
	Model 4	57.93	-	13.35	37.83
	Our model	49.43	21.01	15.76	58.74
Scenarios 4	Model 1	72.91	-	19.14	40.87
	Model 2	86.63	32.91	23	100.55
	Model 3	88.96	-	17.03	50.08
	Model 4	73.35	-	20.26	30.01
	Our model	76.19	30.18	21.28	38.07
Scenarios 5	Model 1	88.87	-	29.80	121.90
	Model 2	92.76	44.57	31.34	152.83
	Model 3	132.49	-	24.24	253.86
	Model 4	91.71	-	32.09	126.19
	Our model	80.16	42.17	32.11	142.85

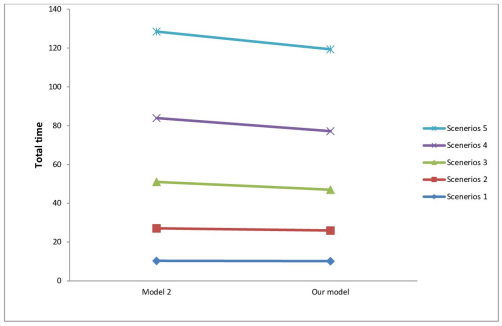
In Figure 5.5a, we can see that the proposed model performed better than model 1 in terms of finding the TD for scenarios (1,3,5), and both models found the same solution for scenario 2. However, for scenarios (4), model 1 found paths with a better TD than the proposed model. In Figure (5.5b), we can observe that our model was able to schedule tours and minimize the total visit time better than model 2 for all scenarios. Regarding the risk criterion presented in Figure (5.5c), the proposed model was successful in planning visits with a lower total risk than model 3 in scenarios (1,2), whereas in the other scenarios, model 3 was able to minimize the total risk better than the proposed model. Furthermore, with regard to the budget criterion shown in Figure (5.5d), the proposed model was able to find optimal visits that minimized the total budget better than model 4 in scenarios (1,4), and the same total budget was found in scenario 2.

By minimizing each criterion (distance, time, risk factor, and budget), the proposed models perform better in most scenarios. In Figure (5.6), we not only compare the model that minimizes such a criterion with the proposed model, but we also take the average of (TD, TT, TO, TC) for models 1, 2, 3, and 4 and compare them with the proposed model. In terms of the means of (TD, TC, TC), we can see that the proposed model found better planning in scenarios (1, 2) than model 1. In Figure (5.6b), the proposed model outperformed model 2 in finding the optimal tours in terms of mean (TD, TT, TO, TC) in all scenarios. Figure (5.6c) shows that in scenarios (1, 2, 3, 4, 5) the proposed model outperformed model 3 in terms of mean (TD, TO, TC). In Figure (5.6d), the proposed model found better optimal turns in scenarios (1, 2) in terms of mean (TD, TO, TC) than model 4.

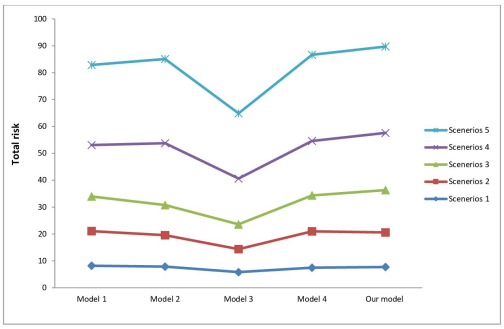
In summary, the proposed model was successful in simultaneously minimizing the four



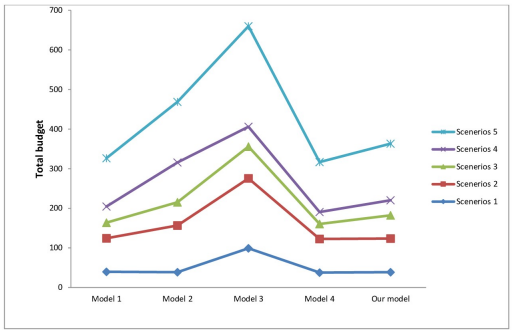
(a) Comparison of TD between our model and the other models



(b) Comparison of TT between our model and the other models



(c) Comparison of TO between our model and the other models



(d) Comparison of TC between our model and the other models

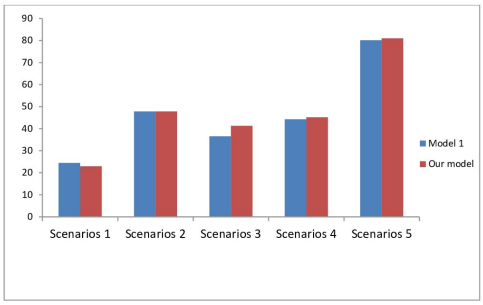
Figure 5.5: Comparison of each criteria between our model and the other models

criteria of distance, time, risk, and budget in planning visits and finding an optimal itinerary.

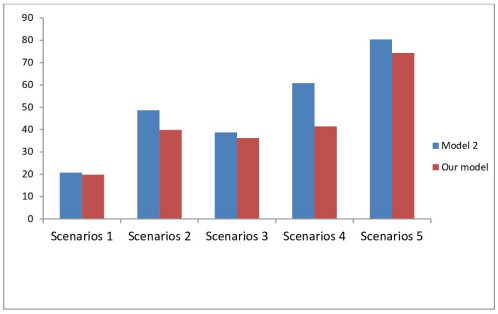
This section proposes an improved mathematical model for route planning for visitor groups. The main goal is to satisfy the preferences and needs of several visitors who are divided into groups. Our model generates tour plans for each visitor group based on four main criteria: the distance of the route, the duration of visits, the risk that visitors may face, and the visit budgets. The proposed model was applied to FEZ-city, which is one of the most visited cities in Africa. The obtained results were promising in terms of generating robust schedules for visits by multiple groups while taking into account visitors' preferences. Moreover, the tour planning done by our model, which considers all the criteria simultaneously, is better and more reasonable compared to models that only consider one criterion.

5.4 Conclusion

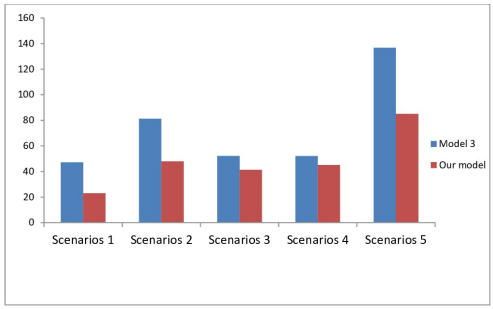
In this chapter, the first section uses the CHN to optimize tourist routes, with a particular focus on distance-based route optimization. The CHN is shown to be highly effective in generating improved tour routes around selected POI, ensuring minimal execution time compared to alternative optimization methods. The second part of the chapter presents an improved mathematical model for planning tours tailored to groups of visitors. In contrast to previous approaches that focused on distance, our model integrates four key objective functions: route distance, tour duration, budget, and risk factors. Each visitor provides personal information, points of interest, and constraints that contribute to the development of a comprehensive mathematical program. The proposed model addresses the limitations of previous studies by focusing on organizing tours for multiple groups of visitors simultaneously. It aims to optimize tour



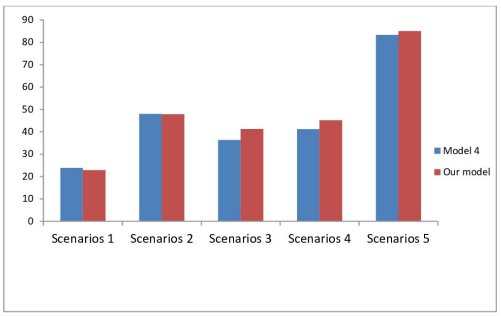
(a) Comparison between our model and Model 1



(b) Comparison between our model and Model 2



(c) Comparison between our model and Model 3



(d) Comparison between our model and Model 4

Figure 5.6: Comparison by average of all criteria between our model and the other models

distances by considering the most efficient way to visit selected points of interest while taking into account tour duration, budget constraints, and potential risk factors. The application of this model to Morocco, specifically the city of Fez, demonstrates its adaptability to diverse and complex tourist destinations. The objectives of this section include the presentation of an intelligent planning approach for organizing multiple visits to avoid overcrowding at tourist sites. In addition, the model incorporates criteria related to risk and budget factors while taking into account the preferred points of interest for visitors. The results demonstrate the effectiveness of the proposed visitor routing model in generating robust itineraries for multiple groups of visitors, outperforming models that focus on individual criteria. This comprehensive approach provides a more sensible and rounded solution to visit planning, taking into account the diverse needs and preferences of different visitor groups.

General conclusions and Future works

Recurrent neural networks (RNNs) are a class of neural models in which the connections between units form a directed graph along a sequence. This property gives them the ability to capture dynamic temporal behavior in data sequences. Unlike Forward Propagation Neural Networks, RNNs are able to use their internal state, also known as memory, to efficiently process input sequences. This makes them particularly suitable for modeling sequential data, such as time or text sequences. RNNs are also often used in optimization tasks, where they help find efficient solutions to complex problems. An emblematic example of RNNs are Hopfield neural networks (HNN), which are widely recognized for their ability to solve combinatorial optimization problems. Hopfield neural networks use a recurrent approach to converge to stable states, which makes them particularly well suited to solving optimization problems.

In the first part of this thesis, the use of the Continuous Hopfield Neural Network (CHN) to manage optimization models has been studied at a deep level. In fact, the CHN was used to solve the optimization problem. In this context, the CHN was successively adapted to optimize stability, and parameter estimates were established. The Continuous Hopfield Network has provided a powerful approach to optimization problems and has shown good performance in various domains. However, two major challenges remain for this network: the definition of appropriate parameters and hyperparameters. Therefore, we propose a novel stability-aware parameter tuning technique for the CHN to address the challenges associated with the CHN and to obtain optimal solutions to combinatorial optimization problems, thus improving the overall performance of the continuous Hopfield network. Also in this thesis, the problem of choosing hyperparameters associated with the CHN is discussed and supported by a new approach: we propose a linear programming model as a solution. The proposed mathematical model aims to minimize the energy function, with constraints representing the penalty parameters, so that each parameter is constrained in its specific domain. The model has been solved using metaheuristic algorithms. The performance of all the contributions was demonstrated through several experiments, divided into visual, graphical, and numerical experiments.

In the second part of this thesis, the tourism sector is an indispensable and flourishing industry that has contributed significantly to the global economy. It has become one of the most significant and dominant sectors worldwide, driving economic growth, creating jobs, and promoting cultural exchange. The advent of technology in the modern era has played a pivotal role in the evolution of traditional tourism practices worldwide. However, despite the technological advancements, tourists still face a critical challenge in planning and optimizing their visits to their preferred points of interest, akin to the street vendor problem. To address this problem, this thesis proposes using the Hopfield neural network, which can effectively improve tourist visits. Specifically, the Hopfield neural network can recommend visitors with personalized travel itineraries, taking into account their preferences. On the other hand, the visitor

routing problem (VRP) is currently a major issue facing the tourism sector. The increasing flow of visitors demands efficient scheduling of their movements to ensure an optimal visitor experience. In the contribution of this thesis, we propose a new mathematical programming model for managing tours for multiple groups of visitors based on their individual preferences. The proposed model takes into account various criteria, such as distance, duration, budget, and risk factors. The well-known heuristic simulated annealing algorithm is well-suited to solve the proposed model. The experiments demonstrate that the proposed model offers numerous advantages for the development of the tourism sector.

The outcomes of the research conducted as part of this thesis have unveiled numerous avenues for further exploration, paving the way for additional in-depth investigations. Primarily, our focus will be on enhancing the methodologies for hyperparameter selection within the Hopfield network framework by integrating novel approaches aimed at tailoring them to diverse optimization challenges. This endeavor seeks to refine the network's adaptability and efficacy across a spectrum of problem domains. Furthermore, our objective encompasses the validation of the models posited within the tourism sector as well as their applicability to a broader array of real-world scenarios. This entails rigorous testing and evaluation of the proposed models in various contexts beyond tourism, thereby gauging their versatility, robustness, and practical utility across different domains. By undertaking these endeavors, we aim to fortify the theoretical foundations of the Hopfield network while also advancing its practical applicability in addressing real-world optimization tasks.

Bibliography

[1] Cook, William J., et al. "Combinatorial optimisation." Wiley-Interscience Series in Discrete Mathematics and Optimization, USA 1 (1998): 998.

[2] Walshaw, Chris. "Multilevel refinement for combinatorial optimisation problems." Annals of Operations Research 131 (2004): 325-372.

[3] Chu, Paul CH. "A genetic algorithm approach for combinatorial optimisation problems." (1997).

[4] Peres, Fernando, and Mauro Castelli. "Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development." Applied Sciences 11.14 (2021): 6449.

[5] Talaván, P. M., Yáñez, J. (2002). Parameter setting of the Hopfield network applied to TSP. Neural Networks, 15(3), 363-373.

[6] García, L., Talaván, P. M., Yáñez, J. (2017). Improving the Hopfield model performance when applied to the traveling salesman problem. Soft Computing, 21(14), 3891-3905.

[7] Hopfield, J. J., Tank, D. W. (1985). "Neural" computation of decisions in optimization problems. Biological cybernetics, 52(3), 141-152.

[8] El Alaoui, M., El Moutaouakil, K., Ettaouil, M. (2016). A multi-step method to calculate the equilibrium point of the Continuous Hopfield Networks: Application to the max-stable problem. International Journal of Computer Science and Information Security (IJCSIS), 14(6), 216-221.

[9] Talaván, P. M., Yáñez, J. (2008). The graph coloring problem: A neuronal network approach. European Journal of Operational Research, 191(1), 100-111.

[10] Haddouch, K., Elmoutaoukil, K., Ettaouil, M. (2016). Solving the Weighted Constraint Satisfaction Problems Via the Neural Network Approach. Int. J. Interact. Multim. Artif. Intell., 4(1), 56-60.

[11] Haddouch, K., El Moutaouakil, K. (2018, April). New Starting Point of the Continuous Hopfield Network. In International Conference on Big Data, Cloud and Applications (pp. 379-389). Springer, Cham.

[12] Ettaouil, M., Loqman, C., Hami, Y., Haddouch, K. (2012). Task assignment problem solved by continuous hopfield network. International Journal of Computer Science Issues (IJCSI), 9(2), 206.

[13] Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. ORSA journal on computing, 3(4), 376-384.

[14] S. Elloumi. The task assignment problem, a library of instances, aviable at: `http://cedric.cnam.fr/oc/TAP/TAP.html`, 2004.

[15] The Second DIMACS Implementation Challenge, aviable at: `ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique/`.

[16] Talaván, P. M., Yáñez, J. (2005). A continuous Hopfield network equilibrium points algorithm. Computers operations research, 32(8), 2179-2196.

[17] Sheppard, C. (2017). Genetic algorithms with Python. S. l: Smashwords Edition.

[18] Han, S. S., May, G. S. (1996, November). Optimization of neural network structure and learning parameters using genetic algorithms. In Proceedings Eighth IEEE International Conference on Tools with Artificial Intelligence (pp. 200-206). IEEE.

[19] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8), 2554-2558.

[20] Yang, L., Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295-316.

[21] Wen, U. P., Lan, K. M., Shih, H. S. (2009). A review of Hopfield neural networks for solving mathematical programming problems. European Journal of Operational Research, 198(3), 675-687.

[22] Ahn, C. W., Ramakrishna, R. S., Kang, C. G., Choi, I. C. (2001). Shortest path routing algorithm using hopfield neural network. Electronics Letters, 37(19), 1176-1178.

[23] Rbihou, S., Joudar, N. E., En-Naimani, Z., Haddouch, K. (2023, February). Using Crank-Nicolson Scheme for Continuous Hopfield Network Equilibrium. In International Conference on Artificial Intelligence and Industrial Applications (pp. 201-210). Cham: Springer Nature Switzerland.

[24] Lu, P., Ye, L., Zhao, Y., Dai, B., Pei, M., Tang, Y. (2021). Review of meta-heuristic algorithms for wind power prediction: Methodologies, applications and challenges. Applied Energy, 301, 117446.

[25] Kramer, O. (2017). Genetic algorithms. In Genetic algorithm essentials (pp. 11-19). Springer, Cham.

[26] Joudar, N. E., En-Naimani, Z., Ettaouil, M. (2019). Using continuous Hopfield neural network for solving a new optimization architecture model of probabilistic self organizing map. Neurocomputing, 344, 82-91.

[27] K. El Moutaouakil and A. Touhafi, "A New Recurrent Neural Network Fuzzy Mean Square Clustering Method," 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), Marrakesh, Morocco, 2020, pp. 1-5.

[28] El Moutaouakil, K., Yahyaouy, A., Chellak, S. et al. An Optimized Gradient Dynamic-Neuro-Weighted-Fuzzy Clustering Method: Application in the Nutrition Field. Int. J. Fuzzy Syst. 24, 3731–3744 (2022).

[29] Talaván, P. M., Yáñez, J. (2006). The generalized quadratic knapsack problem. A neuronal network approach. Neural networks, 19(4), 416-428.

[30] El Moutaouakil, Karim, et al. "OPT-RNN-DBSVM: OPTimal Recurrent Neural Network and Density-Based Support Vector Machine." Mathematics 11.16 (2023): 3555.

[31] Rbihou, S., Haddouch, K. (2021, October). Comparative study between a neural network, approach metaheuristic and exact method for solving Traveling salesman Problem. In 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS) (pp. 1-5). IEEE.

[32] Wei, H., Tang, X. S., Liu, H. (2015). A genetic algorithm (GA)-based method for the combinatorial optimization in contour formation. Applied Intelligence, 43(1), 112-131.

[33] Ettaouil, M., Lazaar, M., Elmoutaouakil, K., Haddouch, K. (2013). A new algorithm for optimization of the kohonen network architectures using the continuous Hopfield networks. wseas transactions on computers, 12(4), 155-163.

[34] Senhaji, K., El Moutaouakil, K., Ettaouil, M. (2016, May). Portfolio selection problem: New multicriteria approach for the mean-semivariance model. In 2016 3rd International Conference on Logistics Operations Management (GOL) (pp. 1-6). IEEE.

[35] Mahajan, R. P. (2011). A quantum neural network approach for portfolio selection. International Journal of Computer Applications, 29(4), 47-54.

[36] K. El Moutaouakil, A. Ahourag, S. Chakir Z. Kabbaj, S. Chellack, M. Cheggour, H. Baizri, Hybrid firefly genetic algorithm and integral fuzzy quadratic programming to an optimal Moroccan diet, Mathematical Modeling and Computing, Vol. 10, No. 2, pp. 338–350 (2023)

[37] El Moutaouakil, Karim, et al. "Multi-Objective Optimization for Controlling the Dynamics of the Diabetic Population." Mathematics 11.13 (2023): 2957.

[38] El Moutaouakil, K.; Baizri, H.; Chellak, S. Optimal fuzzy deep daily nutrients requirements representation: Application to optimal Morocco diet problem. Math. Model. Comput. 2022, 9, 607–615.

[39] El Moutaouakil, Karim, Mouhamed Roudani, and Abdellatif El Ouissari. "Optimal Entropy Genetic Fuzzy-C-Means SMOTE (OEGFCM-SMOTE)." Knowledge-Based Systems 262 (2023): 110235.

[40] El Moutaouakil, Karim, et al. "Multi-objectives optimization and convolution fuzzy C-means: control of diabetic population dynamic." RAIRO-Operations Research 56.5 (2022): 3245-3256.

[41] El Moutaouakil, K., Saliha, C., Hicham, B., Mouna, C. (2023). Intelligent Local Search Optimization Methods to Optimal Morocco Regime. IntechOpen. doi: 10.5772/intechopen.105600;

[42] El Moutaouakil, K., et al. "Metaheuristics optimization algorithm to an optimal Moroccan diet." 2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC). IEEE, 2021.

[43] Smith, K. A. (1999). Neural networks for combinatorial optimization: a review of more than a decade of research. Informs journal on Computing, 11(1), 15-34.

[44] Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: a systematic review. Archives of computational methods in engineering, 29(5), 2531-2561.

[45] Haidar, A., Field, M., Sykes, J., Carolan, M., Holloway, L. (2021). PSPSO: A package for parameters selection using particle swarm optimization. SoftwareX, 15, 100706.

[46] Kang, L., Chen, R. S., Xiong, N., Chen, Y. C., Hu, Y. X., Chen, C. M. (2019). Selecting hyper-parameters of Gaussian process regression based on non-inertial particle swarm optimization in Internet of Things. IEEE Access, 7, 59504-59513.

[47] Sun, J., Lai, C. H., Wu, X. J. (2016). Particle swarm optimisation: classical and quantum perspectives. Crc Press.

[48] Rezaee Jordehi, A., Jasni, J. (2015). Particle swarm optimisation for discrete optimisation problems: a review. Artificial Intelligence Review, 43, 243-258.

[49] Venter, G., Sobieszczanski-Sobieski, J. (2003). Particle swarm optimization. AIAA journal, 41(8), 1583-1589.

[50] Poli, R., Kennedy, J., Blackwell, T. (2007). Particle swarm optimization: An overview. Swarm intelligence, 1, 33-57.

[51] Aszemi, N. M., Dominic, P. D. D. (2019). Hyperparameter optimization in convolutional neural network using genetic algorithms. International Journal of Advanced Computer Science and Applications, 10(6).

[52] Satake, T., Morikawa, K., Nakamura, N. (1994). Neural network approach for minimizing the makespan of the general job-shop. International Journal of Production Economics, 33(1-3), 67-74.

[53] El Alaoui, M., Ettaouil, M. (2021). An adaptive hybrid approach: combining neural networks and simulated annealing to calculate the equilibrium point in max-stable problem. IAENG International Journal of Computer Science, 48(4), 893-898.

[54] Lin, J. S., Liu, M., Huang, N. F. (2000). The shortest path computation in MOSPF protocol using an annealed Hopfield neural network with a new cooling schedule. Information Sciences, 129(1-4), 17-30.

[55] Ettaouil, M., Loqman, C., Haddouch, K., Hami, Y.(2013). Maximal constraint satisfaction problems solved by continuous hopfield networks. WSEAS Transactions on Computers, 12(2), 29-40.

[56] Comert, S., Yazgan, H., Turk, G. (2022). Hopfield neural network based on clustering algorithms for solving green vehicle routing problem. International Journal of Industrial Engineering Computations, 13(4), 573-586.

[57] Bouhouch, A., Chakir, L., El Qadi, A. (2016, October). Scheduling meeting solved by neural network and min-conflict heuristic. In 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt) (pp. 773-778). IEEE.

[58] Yogeesha, C. B., Pujeri, R. V. (2014). A Comparative Study of Geometric Hopfield Network and Ant Colony Algorithm to Solve Travelling Salesperson Problem. International Journal of Advanced Computer Research, 4(3), 843.

[59] Zhao, J. (2015). A Note on Hopfield neural network stability. IAENG International Journal of Computer Science, 42(4), 332-336.

[60] Jain, Ekta, Kalpana Dahiya, and Vanita Verma. "A priority based unbalanced time minimization assignment problem." OPSEARCH 57.1 (2020): 13-45.

[61] Fernández, A., Gómez, S. (2007). Portfolio selection using neural networks. Computers operations research, 34(4), 1177-1191.

[62] Joya, G., Atencia, M. A., Sandoval, F. (2002). Hopfield neural networks for optimization: study of the different dynamics. Neurocomputing, 43(1-4), 219-237.

[63] Singh, Prashant, Ankush R. Kamthane, and Ajinkya N. Tanksale. "Metaheuristics for the distance constrained generalized covering traveling salesman problem." Opsearch 58.3 (2021): 575-609.

[64] Wang, L. (1998). On the dynamics of discrete-time, continuous-state Hopfield neural networks. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 45(6), 747-749.

[65] Ettaouil, M., Loqman, C., Haddouch, K. (2010). Job shop scheduling problem solved by the continuous hopfield networks. Journal of Advanced Research in Computer Science, 2(1), 31-47.

[66] Tan, K. C., Tang, H., Ge, S. S. (2005). On parameter settings of Hopfield networks applied to traveling salesman problems. IEEE Transactions on Circuits and Systems I: Regular Papers, 52(5), 994-1002.

[67] Aarab, Z., Elghazi, A., Saidi, R., Rahmani, M. D. (2018). Toward a smart tourism recommender system: Applied to tangier city. In Innovations in Smart Cities and Applications: Proceedings of the 2nd Mediterranean Symposium on Smart City Applications 2 (pp. 643-651). Springer International Publishing.

[68] Rbihou, S., Joudar, N. E., Haddouch, K. (2023). Parameter tuning of continuous Hopfield network applied to combinatorial optimization. Annals of Mathematics and Artificial Intelligence, 1-19.

[69] Abdullah, A. (2015). The Fez Medina Heritage, Tourism, and Resilience. In International Conference Proceedings: Heritage Tourism Hospitality (pp. 1-10).

[70] Chou, S. Y., Lin, S. W. (2007). Museum visitor routing problem with the balancing of concurrent visitors. In Complex Systems Concurrent Engineering: Collaboration, Technology Innovation and Sustainability (pp. 345-353). Springer London.

[71] El Garouani, A., Alobeid, A. (2013, April). Digital surface model generation for 3D city modeling (Fez, Morocco). In Eighth National GIS Symposium in Saudi Arabia, "The Road For Building Saudi Arabia GIS" organized by the high Committee of the GIS at the Eastern Province.

[72] Heydar, M., Yu, J., Liu, Y., Petering, M. E. (2016). Strategic evacuation planning with pedestrian guidance and bus routing: a mixed integer programming model and heuristic solution. Journal of Advanced Transportation, 50(7), 1314-1335.

[73] Kontogianni, A., Alepis, E. (2020). Smart tourism: State of the art and literature review for the last six years. Array, 6, 100020.

[74] Mathias, M., Moussa, A., Zhou, F., Torres-Moreno, J. M., Poli, M. S., Josselin, D., Rigat, F. (2014, September). Optimisation using natural language processing: Personalized tour recommendation for museums. In 2014 Federated Conference on Computer Science and Information Systems (pp. 439-446). IEEE.

[75] Ntakolia, C., Iakovidis, D. K. (2021). A route planning framework for smart wearable assistive navigation systems. SN Applied Sciences, 3(1), 104.

[76] Sirirak, W., Pitakaso, R. (2018). Marketplace location decision making and tourism route planning. administrative sciences, 8(4), 72.

[77] Steenbruggen, J. G. M. (2014). Tourism geography: Emerging trends and initiatives to support tourism in Morocco. Vrije Universiteit Amsterdam, Faculty of Economics and Business Administration.

[78] Tanaka, K. I., Miyashiro, R., Miyamoto, Y. (2016). A Bi-objective Optimization Model for Designing Safe Walking Routes for School Children. Geographical Analysis, 48(4), 448-464.

[79] Teimouri, M. S., Zaree, A. The effect of valuable and identification places revitalized on tourism development; comparative study: Istanbul, Fez and Cairo.

[80] Vincent, F. Y., Lin, S. W., Chou, S. Y. (2010). The museum visitor routing problem. Applied Mathematics and Computation, 216(3), 719-729.

[81] Wang, Z., Zlatanova, S. (2019). Safe route determination for first responders in the presence of moving obstacles. IEEE transactions on intelligent transportation systems, 21(3), 1044-1053.

[82] Wörndl, W., Hefele, A., Herzog, D. (2017). Recommending a sequence of interesting places for tourist trips. Information Technology and Tourism, 17, 31-54.

[83] Wu, X., Guan, H., Han, Y., Ma, J. (2017). A tour route planning model for tourism experience utility maximization. Advances in Mechanical Engineering, 9(10), 1687814017732309.

[84] Yuan, C., Uehara, M. (2020). An optimal travel route recommendation system for tourists' first visit to Japan. In Advanced Information Networking and Applications: Proceedings of the 33rd International Conference on Advanced Information Networking and Applications (AINA-2019) 33 (pp. 872-882). Springer International Publishing.

[85] Karim, Awatif, Chakir Loqman, and Jaouad Boumhidi. "Determining the number of clusters using neural network and max stable set problem." Procedia Computer Science 127 (2018): 16-25.

[86] Kirkpatrick, S., Gelatt Jr, C. D., Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

[87] Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. Journal of statistical physics, 34, 975-986.

[88] Hoffman, K. L., Padberg, M., Rinaldi, G. (2013). Traveling salesman problem. Encyclopedia of operations research and management science, 1, 1573-1578.

[89] Lecoutre, C., Paris, N., Roussel, O., Tabary, S. (2013, November). Solving WCSP by extraction of minimal unsatisfiable cores. In 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (pp. 915-922). IEEE.

[90] Arora, R. K., Rana, S. P. (1980). Heuristic algorithms for process assignment in distributed computing systems. Information Processing Letters, 11(4-5), 199-203.

[91] de la Vega, W. F., Lamari, M. (2003). The task allocation problem with constant communication. Discrete Applied Mathematics, 131(1), 169-177.

[92] Roupin, F. (1997). On approximating the memory-constrained module allocation problem. Information Processing Letters, 61(4), 205-208.

[93] Ullman, J. D. (1975). NP-complete scheduling problems. Journal of Computer and System sciences, 10(3), 384-393.

[94] Lecoutre, instances aviable at: `http://www.cril.univ-artois.fr/~lecoutre/benchmarks.html` (2008).

[95] `http://mat.gsia.cmu.edu/COLOR/instances.html` (2008).

[96] Elloumi, S. (2004). The task assignment problem, a library of instances.

[97] Bousoño-Calzón, C., Salcedo-Sanz, S. (2004). A discrete-time quantized-state Hopfield neural network. Annals of Mathematics and Artificial Intelligence, 42(4), 345-367.

[98] Medsker, L. R., Jain, L. C. (2001). Recurrent neural networks. Design and Applications, 5, 64-67.

[99] Hopfield, J. J. (2007). Hopfield network. Scholarpedia, 2(5), 1977.

[100] Paik, J. K., Katsaggelos, A. K. (1992). Image restoration using a modified Hopfield network. IEEE Transactions on image processing, 1(1), 49-63.

[101] Wang, Y., You, J. H., Lin, F., Yuan, L. Y., Zhang, M. (2010). Weight constraint programs with evaluable functions. Annals of mathematics and artificial intelligence, 60(3), 341-380.

[102] Khachay, M., Neznakhina, K. (2020). Complexity and approximability of the Euclidean generalized traveling salesman problem in grid clusters. Annals of Mathematics and Artificial Intelligence, 88(1), 53-69.

[103] Wang, Q., Sun, X., Golden, B. L., Jia, J. (1995). Using artificial neural networks to solve the orienteering problem. Annals of Operations Research, 61(1), 111-120.

[104] Smith, K. A., Gupta, J. N. (2000). Neural networks in business: techniques and applications for the operations researcher. Computers Operations Research, 27(11-12), 1023-1044.

[105] Smith, K., Palaniswami, M., Krishnamoorthy, M. (1996). Traditional heuristic versus Hopfield neural network approaches to a car sequencing problem. European Journal of Operational Research, 93(2), 300-316.

[106] Dieu, V. N., Ongsakul, W. (2012). Hopfield lagrange network for economic load dispatch. In Innovation in power, control, and optimization: Emerging energy technologies (pp. 57-94). IGI Global.

[107] Ali, M. K. M., Kamoun, F. (1993). Neural networks for shortest path computation and routing in computer networks. IEEE transactions on neural networks, 4(6), 941-954.

[108] Walsh, M. P., O'malley, M. J. (1997). Augmented Hopfield network for unit commitment and economic dispatch. IEEE Transactions on Power Systems, 12(4), 1765-1774.

[109] Ettaouil, M., Elmoutaouakil, K., Ghanou, Y. (2009). The continuous hopfield networks (CHN) for the placement of the electronic circuits problem. WSEAS Transactions on Computer, 8(12).

[110] Szu, H. (1988). Fast TSP algorithm based on binary neuron output and analog neuron input using the zero-diagonal interconnect matrix and necessary and sufficient constraints of the permutation matrix. In Proc. Int. Conf. on Neural Networks, San Diego, CA, 1988 (pp. 259-266).

[111] Ackley, D. H., Hinton, G. E., Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. Cognitive science, 9(1), 147-169.

[112] Ettaouil, M., and C. Loqman. "A new optimization model for solving the constraint satisfaction problem." Journal of Advanced Research in Computer Science 1.1 (2009): 13-31.

[113] Alenezi, F., Santosh, K. C. (2021). Geometric regularized hopfield neural network for medical image enhancement. International Journal of Biomedical Imaging, 2021.

[114] Lai, Q., Wan, Z., Zhang, H., Chen, G. (2022). Design and analysis of multiscroll memristive hopfield neural network with adjustable memductance and application to image encryption. IEEE Transactions on Neural Networks and Learning Systems.

[115] Wan, Q., Yan, Z., Li, F., Chen, S., Liu, J. (2022). Complex dynamics in a Hopfield neural network under electromagnetic induction and electromagnetic radiation. Chaos: An Interdisciplinary Journal of Nonlinear Science, 32(7), 073107.

[116] Xiao, Y., Zhang, Y., Dai, X., Yan, D. (2022). Clustering Based on Continuous Hopfield Network. Mathematics, 10(6), 944.

[117] Sun, J., Xiao, X., Yang, Q., Liu, P., Wang, Y. (2021). Memristor-based Hopfield network circuit for recognition and sequencing application. AEU-International Journal of Electronics and Communications, 134, 153698.

[118] Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I., Atkinson, P. M. (2021). Explainable artificial intelligence: an analytical review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 11(5), e1424.

[119] Seidl, P., Renz, P., Dyubankova, N., Neves, P., Verhoeven, J., Wegner, J. K., Klambauer, G. (2022). Improving Few-and Zero-Shot Reaction Template Prediction Using Modern Hopfield Networks. Journal of chemical information and modeling, 62(9), 2111-2120.

[120] Joudar, N. E., Ettaouil, M. (2019). Mathematical mixed-integer programming for solving a new optimization model of selective image restoration: modelling and resolution by CHN and GA. Circuits, Systems, and Signal Processing, 38(5), 2072-2096.

[121] Hartmanis, J. (1982). Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). Siam Review, 24(1), 90.

[122] Billionnet, A., Costa, M. C., Sutter, A. (1992). An efficient algorithm for a task allocation problem. Journal of the ACM (JACM), 39(3), 502-518.

[123] Kasihmuddin, M. S. M., Mansor, M. A., Sathasivam, S. (2017). Hybrid Genetic Algorithm in the Hopfield Network for Logic Satisfiability Problem. Pertanika Journal of Science Technology, 25(1).

[124] Feng, G., Douligeris, C. (2000, July). Using Hopfield networks to solve traveling salesman problems based on stable state analysis technique. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium (Vol. 6, pp. 521-526). IEEE.

[125] Mérida-Casermeiro, E., Galán-Marín, G., Munoz-Perez, J. (2001). An efficient multivalued Hopfield network for the traveling salesman problem. Neural Processing Letters, 14, 203-216.

[126] Demange, Marc, et al. "On some applications of the selective graph coloring problem." European Journal of Operational Research 240.2 (2015): 307-314.

[127] QURESHI, S., SHAIKH, A. A., CHANDIO, M. S. (2019). A Newiterative Integrator for Cauchy Problems. Sindh University Research Journal-SURJ (Science Series), 45(3).

[128] Borchardt, M. (1988). An exact penalty approach for solving a class of minimization problems with boolean variables. Optimization, 19(6), 829-838.

[129] De Marchi, A. (2022). On a primal-dual Newton proximal method for convex quadratic programs. Computational Optimization and Applications, 81(2), 369-395.

[130] Hager, W. W., Hungerford, J. T. (2015). Continuous quadratic programming formulations of optimization problems on graphs. European Journal of Operational Research, 240(2), 328-337.

[131] Hopfield, J. J., Tank, D. W. (1985). "Neural" computation of decisions in optimization problems. Biological cybernetics, 52(3), 141-152.

[132] Crank–Nicolson method, `https://en.wikipedia.org/wiki/Crank-Nicolson\_method`.

[133] Euler method, `https://en.wikipedia.org/wiki/Euler\_method`.

[134] Crank, J., Nicolson, P. (1947, January). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In Mathematical proceedings of the Cambridge philosophical society (Vol. 43, No. 1, pp. 50-67). Cambridge University Press.

[135] Lapidus, L., Seinfeld, J. H. (1971). Numerical solution of ordinary differential equations. Academic press.

[136] Faragó, I. (2004, June). Splitting methods and their application to the abstract Cauchy problems. In International Conference on Numerical Analysis and Its Applications (pp. 35-45). Springer, Berlin, Heidelberg.

[137] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." IEEE computational intelligence magazine 1.4 (2006): 28-39.

[138] Medsker, Larry R., and L. C. Jain. "Recurrent neural networks." Design and Applications 5.64-67 (2001): 2.

[139] Salehinejad, Hojjat, et al. "Recent advances in recurrent neural networks." arXiv preprint arXiv:1801.01078 (2017).

[140] Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A critical review of recurrent neural networks for sequence learning." arXiv preprint arXiv:1506.00019 (2015).