



CTU

**CZECH TECHNICAL
UNIVERSITY
IN PRAGUE**

**Faculty of Electrical Engineering
Department of Computer Science**

Bachelor's Thesis

Artificial Neural Networks in Solution of the Orienteering Problems

Jindřiška Deckerová

May 2018

Supervisor: doc. Ing. Jan Faigl, Ph.D.



BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Deckerová Jindřiška** Personal ID number: **420796**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Computer Science and Engineering**
Study program: **Open Informatics**
Branch of study: **Software Systems**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Artificial Neural Networks in Solution of the Orienteering Problems

Bachelor's thesis title in Czech:

Řešení úloh plánování cest s maximalizací užitku neuronovými sítěmi

Guidelines:

1. Familiarize yourself with the Orienteering Problem (OP) [1] and its solution based on Hopfield neural network [2].
2. Familiarize yourself with the generalization of the OP to the OP with Neighborhoods (OPN) and its solution based on self-organizing maps [3].
3. Implement [2] and evaluate its performance in standard OP benchmarks [4].
4. Proposed and implement improvement of [3] using local heuristics, e.g., [5].
5. Evaluate and compare the OP(N) solvers in selected benchmarks [4].

Bibliography / sources:

- [1] Pieter Vansteenwegen, Wouter Souffriau, Dirk Van Oudheusden, The orienteering problem: A survey, In European Journal of Operational Research, (2011) 209(1):1-10.
[2] Qiwen Wang, Xiaoyun Sun, Bruce L. Golden, Jiyou Jia: Using artificial neural networks to solve the orienteering problem. Ann. Oper. Res. (1995) 61:111-120.
[3] Jan Faigl: On self-organizing maps for orienteering problems. IJCNN 2017: 2611-2620.
[4] The Orienteering Problem: Test Instances, <http://www.mech.kuleuven.be/en/cib/op>, (cited on Jan 10 2018).
[5] I-Ming Chao, Bruce L. Golden, Edward A. Wasil: A fast and effective heuristic for the orienteering problem, European Journal of Operational Research, (1996), 88(3):475-489.

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Jan Faigl, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **31.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

doc. Ing. Jan Faigl, Ph.D.
Supervisor's signature

Head of department's signature

prof. Ing. Pavel Ripka, CSc.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

" If I don't have some cake soon, I might die."

-STANLEY HUDSON, THE OFFICE (SEASON 4, EP. 07)



Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze 25. května 2018



Acknowledgement

I would like to thank doc. Ing. Jan Faigl, Ph.D. for supervising my thesis and guiding me like Gandalf guided Frodo through the Middle Earth. I would also like to thank Ing. Petr Váňa for making fun of me and kind of helping. And I would like to thank my Life Master Mr. Bernard Black; he is truly an inspiration. And big thank you to my family who supported me and bought me a lot of coffee.

Abstract

In this thesis, neural networks (NN) approaches for the Orienteering Problem (OP) and their extensions to the Orienteering Problem with Neighborhoods (OPN) are studied. We focus on the Hopfield Neural Network (HNN) and the Self-Organizing Map (SOM) and we propose two new modifications built on the HNN and the SOM to address the OPN and improve the performance of the NN approaches in solving orienteering problems. The first modification extends the existing discrete HNN solver for the OP to address the OPN, for which a new data representation is proposed together with the modification of the energy function to reflect the new data representation. The second modification is built on the SOM solver, where we propose the new selection of the winner neurons and a combination of the unsupervised learning with the combinatorial metaheuristic called the Variable Neighborhood Search (VNS).

Keywords: neural networks; orienteering problem; orienteering problem with neighborhoods; hopfield neural network; self-organizing map


Abstrakt

V této práci jsou studovány přístupy neuronových sítí (NN) pro problematiku orientace (OP) a jejich rozšíření na problematiku orientace s okolím (OPN). Zaměřujeme se na umělé neuronové sítě, konkrétně na Hopfieldovu neuronovou síť (HNN) a samoorganizační mapy (SOM) a navrhujeme dvě nové modifikace postavené na HNN a SOM tak, aby se řešily OPN a zlepšily výkonnost NN v rámci řešení problematiky orientace. První modifikace rozšiřuje existující diskrétní HNN pro OP s cílem řešit OPN. Modifikace je založena na nové reprezentaci dat a modifikované energetické funkci, která respektuje novou reprezentaci dat. Druhá modifikace je založena na řešení SOM, kde navrhujeme nový výběr neuronů a kombinaci neuronového učení bez učitele s metaheuristickým přístupem Variable Neighborhood Search (VNS).

Klíčová slova: neuronové sítě; problematika orientace; problematika orientace s okolím; hopfieldova neuronová síť; samorganizační mapy

Contents



1	Introduction	1
2	Problem Statement	3
2.1	Orienteering Problem	3
2.2	Orienteering Problem with Neighborhoods	4
3	Related Work	5
3.1	Neural Network Based Approaches	7
4	Hopfield Neural Network	9
4.1	Introduction	9
4.2	Neural Network Representation	9
4.3	Energy Function	11
4.4	Update Algorithm	12
4.5	Proposed Implementation of the HNN for the OP	13
4.5.1	Modified Update Algorithm	13
5	HNN Extension to the OPN	21
5.1	Hopfield Neural Network for the OPN	21
5.2	Implementation of the HNN Solver for the OPN	24
6	Self-Organizing Map	25
6.1	Improvements of the SOM for the OPN	27
6.1.1	Proposed Modification of the Winner Neuron Selection	27
6.1.2	SOM-based Initialization of the VNS	28
7	Results	31
7.1	Results for the Orienteering Problem	32
7.2	Results for the Orienteering Problem with Neighborhoods	35
8	Conclusion	39



Bibliography	41
A Gallery of the Solutions of the OP and the OPN	43
B Content of the enclosed CD	49

List of Figures

4.1	Representation of the HNN for the OP	10
4.2	The structure of the HNN based solver	14
4.3	Representation of the HNN during updates	15
4.4	UML class diagram of the HNN based solver	18
4.5	Activity diagram of the HNN based solver	19
5.1	Sampled variant of the OPN	21
5.2	Representation of the HNN for the OPN	22
6.1	Data flow of the SOM-based initialization of the VNS	29
7.1	Average sum of rewards of the OP by the HNN and the SOM-VNS	33
7.2	Average sum of rewards the SOM-based approaches for the OP .	34
7.3	Average sum of rewards the NN approaches for the OPN	37
7.4	Average sun of rewards of the OPN by the SOMvX	37
A.1	Solutions of the selected problems of the OP by the HNN-OP . .	43
A.2	Solutions of the selected problems of the OP by the SOMv1 . . .	43
A.3	Solutions of the selected problems of the OP by the SOMv2 . . .	44
A.4	Solutions of the selected problems of the OP by the SOMv3 . . .	44
A.5	Solutions of he selected problems of the OP by the SOMv4 . . .	44
A.6	Solutions of the selected problems of the OP by the SOMv5 . . .	45
A.7	Solutions of selected the problems of the OP by the SOM-VNS .	45
A.8	Solutions of selected the problems of the OPN by the HNN-OPN	45
A.9	Solutions of selected the problems of the OPN by the SOMv1 . .	46
A.10	Solutions of selected the problems of the OPN by the SOMv2 . .	46
A.11	Solutions of selected the problems of the OPN by the SOMv3 . .	46
A.12	Solutions of selected the problems of the OPN by the SOMv4 . .	47
A.13	Solutions of selected the problems of the OPN by the SOMv5 . .	47
B.1	Content of the enclosed CD	49



List of Tables

6.1	Combination of proposed winner selections for the SOM	28
7.1	Aggregated results for the OP	33
7.2	Results of the Set 64 for the OP	34
7.3	t-test for the selected OP solvers and problems	35
7.4	Results for the OPN	36



List of Algorithms

1	The OP solver based on the Hopfield Neural Network	16
2	The winner determination of the SOM	26
3	The randomized VNS solver	29

Abbreviations

ARPE	Average Relative Percentage Error
GCOP	Generalized Clustered Orienteering Problem
GRASP	Greedy Randomized Adaptive Search Procedure
HNN	Hopfield Neural Network
ILP	Integer Linear Programming
MILP	Mixed Integer Linear Programming
MTZ	Miller-Tucker-Zemlin formulation of the OP
NN	Neural Network
OP	Orienteering Problem
OPN	Orienteering Problem with Neighborhoods
PC-TSP	Price-Collecting Travelling Salesman Problem
PR	Path Relinking
RPE	Relative Percentage Error
SOM	Self-Organizing Map
SOMv1	Self-Organizing Map based modification 1
SOMv2	Self-Organizing Map based modification 2
SOMv3	Self-Organizing Map based modification 3
SOMv4	Self-Organizing Map based modification 4
SOMv5	Self-Organizing Map based modification 5
SOM-VNS	Self-Organizing Map based VNS modification
TSP	Travelling Salesman Problem
TTDP	Tourist Trip Design Problems
VNS	Variable Neighborhood Search

Symbols Used

$ (i, j) $	Euclidean distance between locations $i, j \in \mathbb{R}^2$
δ	Sensing radius
p_{s_i}	Waypoint location at which s_i is visited (covered)
R	Sum of rewards
\mathcal{R}	Final route
$\overline{\mathcal{R}}$	Locations not included in the final route
S	Set of locations
s_i	i -th location
Σ	Sequence of visits to the locations $s \in S_k$
σ_i	i -th visit
T_{\max}	Travel budget
ς_i	Score of the location s_i
E	Energy function of the Hopfield Neural Network (HNN)
a, b, c, d, e, f	Parameters of the energy function E
\mathbf{A}	Adjacency matrix, $\mathbf{A} \in \mathbb{R}^{n,n}$
Δt	Time step of the HNN activation function
χ	3D state matrix, $\chi \in \mathbb{R}^{n,n,n+1}$
$\chi_{i,j,k}$	State (i, j, k) of the 3D state matrix χ
E	Energy function
$p_k^{s_i}$	k -th waypoint of s_i
Φ	State matrix, $\Phi \in \mathbb{R}^{n,n+1}$
$\Phi_{i,j}$	State (i, j) of the state matrix Φ
s_{insert}	Location to be inserted in the HNN solver
s_{remove}	Location to be removed in the HNN solver
T	Threshold for state selection of the HNN
ϑ	Threshold for network stabilization
α, G, μ	Learning parameters of Self-Organizing Map (SOM)
d	d neighborhood of the s counting as the no. neurons in the ring
$L_{T_{\text{win}}}$	Length of the final tour
\mathcal{N}	Set of neurons formed as the ring
ν	Neuron representign the location
ν^*, ν_n^*	Winner neurons
$\nu_l, \nu_{l_1}, \nu_{l_2}, \nu_f, \nu_r$	Winner neurons of the SOM learning epoch
Π	Permutation of the sensor locations
p'	Alternate location
p_s, p'_s	Closest and the weighted closest point
T	Tour

Introduction

This thesis focuses on the artificial neural networks in the solution of the Orienteering Problem (OP) and its variants arising from robotic data collection planning. The origin of the OP is set in sport, more specifically in the orienteering races that origin in the late 19th century in Sweden. In the orienteering race, the goal for the game participants is to find control stations, each with the associated reward, in the shortest time possible by using a given map of station locations. The OP is similar to the orienteering race. The objective is to visit as many control stations as possible in the shortest time. The OP was introduced by Golden, Levy, and Vohra in [1]. In [2], it is pointed out that the OP can be considered as a combination of the Knapsack Problem to determinate the most valuable locations, and the Travelling Salesman Problem for finding the shortest tour of the selected locations.

The OP can be used for data gathering, logistic tasks, or planning applications. Some of the OP applications need a specific modification and survey of the OP variants can be found in [3]. One of the first applications of the OP is mentioned in [4], where the traveling salesman knows the number of sales he can make in each city, and he needs to maximize the number of made sales while the distance traveled between the cities in one day needs to fit the given travel budget. The authors of [5] describe a complex routing problem where a fleet of trucks must deliver fuel to many customers. The fuel level of each customer must be maintained above a critical level. The objective is to select a subset of customers, which fuel is close to the critical level, and deliver fuel to these customers.

A recent application of the OP is in the tourist guides applications. A tourist visits a city and wants to visit as many sights and attractions as possible but does not have time to visit all of them. In [6], the Mobile Tourist Guide lets tourists select attractions and then it makes a feasible plan to visit the most valuable attractions in the limited time. This problem is a part of the Tourist Trip Design Problems.

In [7], the authors present the Generalized Clustered Orienteering Problems, which is motivated by the Pokémon GO game. The application seeks for time-effective and profitable tours. The places to visit are divided into clusters. The objective is to find an unrooted tour that maximizes the total selected prize while at least one node of each cluster is visited and the traveled distance satisfies the given travel budget.

In the data collection missions, an autonomous vehicle with limited travel budget collects data from sensor locations. Each location has associated reward according to the importance

of the respective data. However, the vehicle cannot visit all sensor locations, because of the limited travel budget. Therefore, only a subset of locations can be visited. In some cases, the data does not need to be obtained from the sensor location itself, the gathering device can collect data remotely from the sensor location within a communication range, which can save the travel cost, and thus collect data from more sensors. This variant of the OP is referred to as the Orienteering Problem with Neighborhoods (OPN).

Several approaches have been proposed in the literature to solve the OP including algorithms for finding optimal solutions and effective heuristics. However, the existing heuristics may not be the best option to solve the OPN. One of the reasons is that the OPN is a continuous problem and heuristics for the OP represent solutions of discrete problems. Although the neighborhood can be sampled, still to find the most suitable number of neighbors is quite challenging and heuristics may perform poorly. Therefore, this thesis is built on the neural network based approaches, such as the Hopfield Neural Network (HNN), [8], and the Self-Organizing Maps (SOM), [9].

The thesis is organized as follows. The OP is formally introduced as the optimization problem in Chapter 2. The existing approaches and heuristics are overviewed in Chapter 3. An artificial neural network based on the Hopfield-Tank model is introduced in Chapter 4. In Chapter 5 the Hopfield Neural Network is extended to the OPN. The proposed modifications of the SOM are presented in Chapter 6. Results on the evaluation of the solvers for the OP and the OPN are reported in Chapter 7. Conclusion is in Chapter 8.

Problem Statement

The Orienteering Problem (OP) and specifically the Orienteering Problem with Neighborhoods (OPN) studied in this thesis are motivated by the data collection missions to collect the most rewarding data by the specified vehicle data collector. Given a set of sensor locations $s_i \in S$, each location has associated reward $\varsigma \geq 0$ depending on its importance. The vehicle is requested to start at the specified location and terminate at the given end location. The problem is to determine a route that allows the vehicle to retrieve data from the most valuable locations while the total route length does not exceed the given travel budget of the vehicle.

The OP can be formulated as Mixed-Integer Linear Programming (MILP), i.e., some variables used in the problem formulation are constrained by integers while others not. In [2], Vansteenwegen formulates the OP as an Integer Linear Programming (ILP) problem with decision variables and sub tour eliminations constraints created according to the Miller-Tucker-Zemlin formulation of the Traveling Salesman Problem (TSP).

The Orienteering Problem with Neighborhoods extends the OP to the cases, where the vehicle can collect data within a specified range from the particular sensor location. Therefore, it is not necessary to visit the location itself.

The range for data collection is denoted as a sensing radius δ . The neighborhood of the sensor location can be presented as a disk with the radius δ centered at the location s . The problem is then to determine (in addition to the sequence of the visits of the most valuable sensor locations) the waypoint location p_i (a point on the disk) for each selected s_i to be visited. Both the OP and the OPN are formally described in the rest of the section.

2.1 Orienteering Problem

Having a set of locations $S = \{s_1, \dots, s_n\}$ at which data can be obtained, each location $s_i \in \mathbb{R}^2$ with the assigned reward $\varsigma_i \geq 0$, start and end locations denoted as s_1 and s_n with the assigned rewards $\varsigma_1 = \varsigma_n = 0$, the objective is to maximize the sum of rewards of the visited locations, while the sum of the total route length does not exceed the given travel budget T_{\max} . Due to the limited travel budget, not all locations can be visited, but only a subset of the most valuable locations. We follow the formulation presented in [10], where the subset of k locations is denoted as $S_k \subseteq S$ and the subset S_k can be defined by a sequence of the visits to the sensor locations expressed as a permutation $\Sigma = (\sigma_1, \dots, \sigma_k)$, where $1 \leq \sigma_i \leq n$, $\sigma_i \neq \sigma_j$, for $i \neq j$,

and the prescribed start and end of the sequence, $\sigma_1 = 1, \sigma_k = n$. The requested route \mathcal{R} can be represented as a sequence of the sensor locations $s_i \in S_k$, $\mathcal{R} = (s_{\sigma_1}, \dots, s_{\sigma_k})$. The OP can be then defined as follows.

$$\underset{k, S_k, \Sigma}{\text{maximize}} \quad R = \sum_{i=1}^k \varsigma_{\sigma_i} \quad (2.1)$$

$$\text{subject to} \quad \sum_{i=1}^{k-1} |(s_i, s_{i+1})| \leq T_{\max} \quad (2.2)$$

$$s_{\sigma_1} = s_1, s_{\sigma_k} = s_n, \quad (2.3)$$

where $|(s_i, s_{i+1})|$ denotes the Euclidean distance between the locations s_i and s_{i+1} .

2.2 Orienteering Problem with Neighborhoods

The OPN is an generalization of the OP where we allow to collect the reward within the δ sensing range from the particular selected location. Let $S = \{s_1, \dots, s_n\}$ be a set of sensor locations, δ be a sensing radius within data from the location $s_i \in \mathbb{R}^2$ can be obtained, each location except the start and end has assigned reward $\varsigma_i > 0$, $\varsigma_1 = \varsigma_n = 0$, the start location and the end location be fixed locations denoted as s_1 and s_n , respectively, and considered without the sensing radius, $\delta_1 = \delta_n = 0$, because the vehicle is requested to precisely visit these locations. Similarly to the OP, the objective of the OPN is to maximize the sum of rewards of the visited locations, while the total route length does not exceed the travel budget T_{\max} . Thus not only the subset of the k visited locations $S_k \subseteq S$ and a sequence of the visits Σ need to be determined but request a set of the waypoint locations $p_i \in \mathbb{R}^2$ within the particular δ -neighborhood of each selected sensor locations. In this case, the route \mathcal{R} can be represented as a sequence of the waypoint locations $\mathcal{R} = (p_{\sigma_1}, \dots, p_{\sigma_k})$, where p_{σ_i} is the waypoint location from where data of the location s_{σ_i} can be gathered, and $1 \leq \sigma_i \leq n$, $\sigma_i \neq \sigma_j$, for $i \neq j$, and the start and end are prescribed, $\sigma_1 = 1, \sigma_k = n$. The OPN is then defined as follows

$$\underset{k, S_k, \Sigma, P}{\text{maximize}} \quad R = \sum_{i=1}^k \varsigma_{\sigma_i} \quad (2.4)$$

$$\text{subject to} \quad \sum_{i=1}^{k-1} |(p_{\sigma_i}, p_{\sigma_{i+1}})| \leq T_{\max} \quad (2.5)$$

$$|(p_{\sigma_i}, s_i)| \leq \delta_i \quad (2.6)$$

$$p_{\sigma_1} = s_1, p_{\sigma_k} = s_n, \quad (2.7)$$

where $|(p_{\sigma_i}, p_{\sigma_{i+1}})|$ denotes the Euclidean distance between the waypoint locations p_{σ_i} and $p_{\sigma_{i+1}}$.

Related Work

The Orienteering Problem (OP) and the Orienteering Problem with Neighborhoods (OPN) have been studied since 1980s and many approaches have been proposed. This thesis is built on existing approaches based on neural networks (NN) [8] and [10], and the Variable Neighborhood Search (VNS) [11], therefore an overview of these approaches is presented in this chapter together with the description of the existing heuristics.

The first approach to solve the OP was by Tsiligirides in [4], where he proposed two heuristics: S-Algorithm and D-Algorithm. The S-Algorithm is a stochastic heuristic based on the Monte Carlo method. The algorithm generates many random routes and selects the best one. The routes are built with probabilities based on the reward associated with the added location and the Euclidean distance to the added location. The second heuristic is deterministic and uses a method from Wren and Holiday's routing scheduling problem for one depot [12] to build up routes. To improve the found routes, Tsiligirides proposed R-I-Algorithm that includes the exchange of two locations, the insertion of a location and the 2-Opt. The 2-Opt is a heuristic used for the local route improvement proposed by Croes in [13] to solve the Traveling Salesman Problem (TSP). Given an initial route that crosses over itself, the heuristic reorders route to eliminate the crossovers.

The Center of gravity heuristic to solve the OP was proposed by Golden *et al.* in [1]. The heuristic consists of three steps: an initial route, which satisfies the OP constraints (starts at 1, ends at n , and the total length does not exceed the travel budget), is constructed by the insertion heuristic. In the next step, the route is improved by the 2-Opt and the cheapest insertion (the inserted location causes the lowest increase of the total route length). In the last step, a new route is created by adding decreasingly sorted ranked locations. The locations are ranked by the ratio of the reward to the distance from the location to the center of gravity $g = (\frac{\sum_{i=1}^n S_i x_i}{\sum_{i=1}^n S_i}, \frac{\sum_{i=1}^n S_i y_i}{\sum_{i=1}^n S_i})$, where (x_i, y_i) are the coordinates of the i -th location, and S_i is the reward of the location i . The locations are then sorted and via cheapest insertion added to the empty route. The two previous steps are then repeated.

In [14], Ramesh *et al.* introduced the Four-phase heuristic. The heuristic consists of the route improvement phases: the insertion, the improvement, and the reduction. In the first phase, locations are added to the route with the relaxed travel budget constraint. The next phase uses the 2-Opt and 3-Opt (the local search heuristic for the TSP) to improve the route. In the third phase, a location with the minimal ratio of the reward to distance is removed

from the route. The three phases are iteratively repeated. Finally, the fourth phase inserts as many locations as possible while the budget constraint is satisfied.

The heuristic by Chao *et al.* proposed in [15] consists of five steps: 1) initialization, 2) two-exchange, 3) one-point movement, 4) clean up, and 5) reinitialization. The method is referred to as the Five-step heuristic. The initialization of the procedure is following. An ellipse is constructed over all locations, two foci of the ellipse are in the start and the end locations, the length of the major axis is equal to the travel budget. Then routes are constructed by using the cheapest insertion. The route with the highest sum of rewards denoted as $route_{op}$, is then improved by the two-exchange, where a location from the $route_{op}$ is exchanged with a location from the $route_{nop}$, where $route_{nop}$ is the rest of routes generated in the initial step. The two-exchange step uses the cheapest insertion. The improvement step continues with the one-point movement, i.e., a location is moved between routes greedily. The clean up step shortens the $route_{op}$ by applying the 2-Opt. In the reinitialization step, the k locations with the smallest ratio of the reward to the insertion cost are moved from the $route_{nop}$ to the $route_{nop}$.

The combination of the Graph Randomized Adaptive Search Procedure (GRASP) and the Path Relinking (PR) was proposed by Campos *et al.* in [16]. The author proposed four GRASP methods for the OP. The GRASP solution is then linked to another solution via the PR. The PR method gradually transforms an initial solution to the requested solution. Each GRASP method starts with the initialization of the route and the candidate solution CL . Then a restricted candidate list RCL is constructed and evaluated by a greedy function. From the RCL , a location is selected and then inserted at the best position of the route. In each method, the construction of the RCL differs. The RCL in the first method is created from the CL and all locations with rewards higher than the specified threshold for the location from the CL . In the second method, the RCL is similar to the one from the first method, except the size of the RCL is limited by the cardinality threshold of the CL , and no locations are repeated within the set. The third method creates the RCL depending on a quotient between the reward and the smallest time step. All location with the quotient higher than the threshold quotient are used to construct the RCL . The last method differs from the third method like the seconds from the first. The size of the RCL is limited, and no repetitions are allowed.

The Variable Neighborhood Search (VNS) for the OP was proposed by Sevkli and Sevilgen in [11] and searches the solution space with changing its neighborhood. The proposal is based on the VNS metaheuristic introduced by Hansen *et al.* in [17]. The VNS heuristic consists of four structures: *point insert*, *points exchange*, *sub route insert*, and *sub routes exchange*. The *point insert* structure randomly selects a location from the solution (the current route) and inserts it at a random position of the solution. In the *points exchange* structure, two different locations of the solution are randomly selected and exchanged. The *sub route insert* is similar to the first structure, but a sub route is randomly selected and then inserted at a different position of the solution. In the *sub routes exchange* structure, two different sub routes are randomly selected from the solution and exchanged. The structures are used in two phases called *shake* and *local search*. In the *shake* phase, the current solution is exchange with a different random solution. The *local search* phase takes the current solution and searches its neighborhood to obtain a better solution if it is possible.

3.1 Neural Network Based Approaches

In this thesis, approaches based on the neural networks (NN) are mainly considered as a suitable technique to solve the OPN. The first artificial neural network for solving the OP was based on the Hopfield-Tank model and proposed by Wang *et al.* in [8]. The authors build the network for the OP on the neural network model from [18] combining it with traditional heuristics. The input data formed as a set of locations are encoded into the two-dimensional state matrix, where the state at the cell (i, j) represents that the i -th location is visited at the j -th position of the route. The neural part consists of updating the state matrix by an activation function until a local minimum is found. The activation function is a sigmoid function that minimizes the complex energy function. The energy function reflects the constraints of the OP, such as the route starts at the start location and ends at the end location, the total route length does not exceed the given travel budget, the locations in the route are not repeated and so on. After the network is stabilized, the route is decoded from the state matrix and improved by the 2-Opt. The cheapest insertion and the deletion of the least valuable location (the location with the largest ratio of its distance to its reward) are the heuristics employed in the NN-based solution of the OP.

The Self-Organizing Map (SOM) for solving the OP was proposed by Best *et al.* in [9]. The network is based on the SOM for solving the Price-Collecting Traveling Salesman Problem with Neighborhoods (PC-TSPN) proposed in [19]. The SOM is a two-layer competitive neural network based on Kohonen's unsupervised learning [20]. The first layer consists of sensor locations and the second layer consists of neurons. The second layer forms a ring of neurons that represent the solution (the final route). The SOM algorithm consists of updating the neuron ring for each location $s_i \in S$. In each epoch, a location is selected, and the neuron is adapted according to the neighboring function towards the current s_i . The SOM for the OPN in [21] differs from the SOM for the PC-TSPN [19] in the limited travel budget and to favor the locations with the highest rewards. In [21], the authors propose the duplication of locations by a factor of rewards divided by the greatest common divisor of the set of rewards to adapt network more often to the locations with the highest rewards during a single epoch. As addressed in [10], this proposition is computationally demanding, therefore the author of [10] proposed a new method of conditional adaptation in [10].

In each epoch, the winner neuron is determined for each location. In addition to the winner neuron, two neurons from the ring are selected, a neuron with the shortest distance to its location, and a neuron with the lowest reward associated with its location. Then, the winner neuron is adapted towards the current location; however, if the route represented by the SOM would exceed the travel budget, the two selected additional neurons are removed from the network. If the route represented by the SOM is still exceeding the travel budget, the adaptation and removal is reverted and the SOM continues with the next sensor location.

Since both NN-based methods are further extended and improved, their detail description is presented in the following chapters together with the proposed modifications.

Hopfield Neural Network

4.1 Introduction

The first attempt to solve the Orienteering Problem (OP) by the Hopfield Neural Network (HNN) was by Wang *et al.* in [8]. The HNN approach consists of the data representation, the design of the complex energy function and the application of traditional heuristics [13]. The Hopfield-Tank model in the designed network to solve the OP consists of the state matrix, where each cell denotes an activation level. The Hopfield-Tank model is a continuous model, where each activation level of the matrix is updated to a value from the interval $[0, 1]$ one at the time via the sigmoid *activation function*

$$\Phi_{i,j} = \frac{1}{1 + \exp^{-\alpha}}, \text{ with} \quad (4.1)$$

$$\alpha = \ln(\Phi_{i,j}) - \ln(1 - \Phi_{i,j}) - \frac{\partial E}{\partial \Phi_{i,j}} \Delta t, \quad (4.2)$$

where E is the *energy function* and Δt is the time step.

4.2 Neural Network Representation

The graph formulation of the OP is used for the representation of the HNN. Having a graph $G(V, E)$, where V is a set of n vertices, each vertex is assigned with a reward $\varsigma_i \geq 0$, $i \in [1, n]$, where $\varsigma_1 = \varsigma_n = 0$. There is a symmetrical edge from E with no orientation between every two vertices. The edge between vertices i and j is associated with the cost of the traveled distance between i and j . The undirected complete graph can be interpreted as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n,n}$, where n is the number of vertices.

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if there is an edge between } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}. \quad (4.3)$$

The matrix \mathbf{A} is then used for the conceptualization of the neural network as follows.

Let $\Phi \in \mathbb{R}^{n,n+1}$ be a state matrix that represents the neural network, see Fig. 4.1. The activation level of the location s_i that is visited at the position j is denoted as $\Phi_{i,j}$ and it

is referred as a *state*. During the updating of the matrix Φ , the state value is in the interval $[0, 1]$.

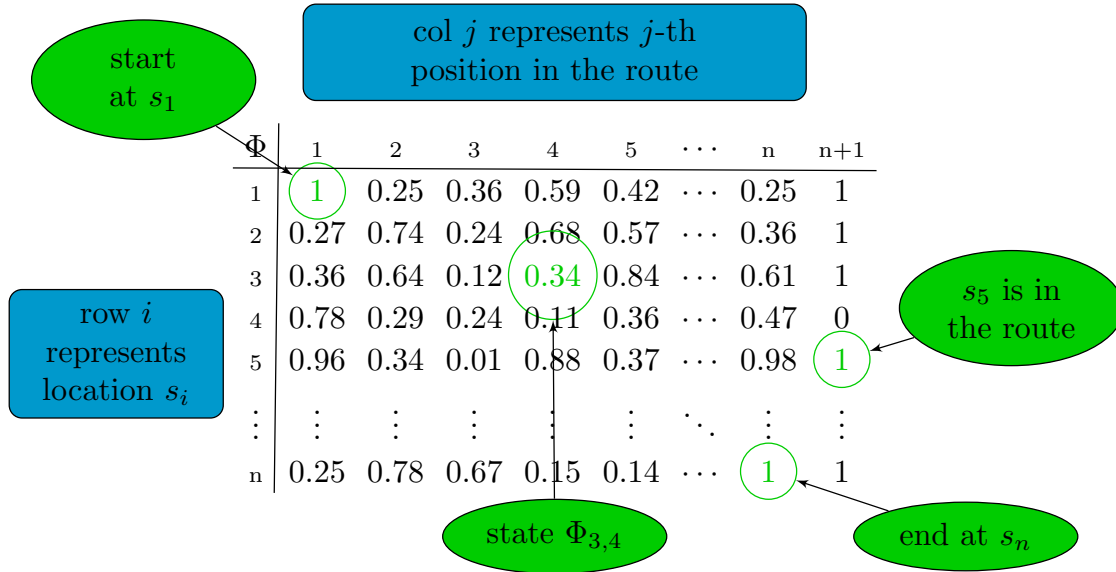


Figure 4.1: The state matrix Φ representing the Hopfield Neural Network. The rows represents locations, columns represents position at the route. The last column is set to 1 if the respective location is in the route. The start location s_1 and the end location s_n are represented by the states $\Phi_{1,1}$ and $\Phi_{n,n}$, respectively.

The start and end states are prescribed and therefore nodes $\Phi_{1,1}$ and $\Phi_{n,n}$ represent the start and the end, respectively, and they are set to 1. The $(n+1)$ -th column of the matrix Φ is used for the calculation of the route reward [8]; $\Phi_{i,n+1} = 1$, when location s_i is in the route, otherwise $\Phi_{i,n+1} = 0$.

4.3 Energy Function

The energy function is the essential part of the Hopfield-Tank based neural network. The neural network for the OP [8] minimizes the following quadratic energy function:

$$E = \frac{a}{2} \cdot \sum_{i=1}^n \sum_{j=1}^n \sum_{\substack{h=1 \\ h \neq i}}^n \Phi_{i,j} \cdot \Phi_{h,j} \quad (4.4)$$

$$+ \frac{b}{2} \cdot \left[\sum_{i=1}^n \sum_{j=1}^n \Phi_{i,j} - n \right]^2 \quad (4.5)$$

$$+ \frac{c}{2} \cdot \Gamma \left(\sum_{i=1}^n \sum_{j=1}^{n-1} \sum_{h=1}^n |(s_i, s_h)| \cdot \Phi_{i,j} \cdot \Phi_{h,j+1} - T_{max} \right) \quad (4.6)$$

$$+ d \cdot (2 - \Phi_{1,1} - \Phi_{n,n}) \quad (4.7)$$

$$+ e \cdot \sum_{i=1}^n \left[\Phi_{i,n+1} \cdot \left(1 - \sum_{j=1}^n \Phi_{i,j} \right) \right] \quad (4.8)$$

$$- f \cdot \sum_{i=1}^n (\zeta_i \cdot \Phi_{i,n+1}), \quad (4.9)$$

where a, b, c, d, e, f are the parameters of the energy function, Φ is the state matrix, T_{max} is the given travel budget, $|(s_i, s_j)| = \|s_i - s_j\|$ is the Euclidean distance from the location s_i to the location s_j , ζ_i is the reward of the location s_i , and $\Gamma(x)$ is a function

$$\Gamma(x) = \begin{cases} x^2 & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases} \quad (4.10)$$

The quadratic energy function consists of several terms. Each term has been proposed to meet the constraints of the OP and can be characterized as follows.

Eq. 4.4 penalizes columns of the matrix Φ that have more than one state activated, since only one location can be visited at the time.

Eq. 4.5 ensures that the number of activated states in the matrix Φ is equal to n . If the number of activated states is less than n , the activated states are consecutively repeated in the final sequence.

Eq. 4.6 ensures that the total route length does not exceed the travel budget T_{max} .

Eq. 4.7 ensures that the route starts at the start state and ends at the end state.

Eq. 4.8 sets the $(n + 1)$ -th column of the matrix Φ to 1, if the location s_i is in the route, otherwise it is set to 0.

Eq. 4.9 is proposed to maximize the total sum of rewards.

The neural network includes the energy function E and the weights that are then used for updating the matrix Φ . The weights depend on the current state of the matrix Φ and are calculated as the second derivate of the energy function by the respective state of the matrix Φ .

The activation function uses the weights for the network update, therefore with knowledge of the energy function, new values of the matrix Φ can be calculated from the partial derivative $\frac{\partial E}{\partial \Phi_{i,j}}$

$$\frac{\partial E}{\partial \Phi_{i,j}} = a \cdot \sum_{\substack{h=1 \\ h \neq i}}^n \Phi_{h,j} \quad (4.11)$$

$$+ b \cdot \left[\sum_{h=1}^n \sum_{k=1}^n \Phi_{h,k} - n \right] \quad (4.12)$$

$$+ c \cdot \gamma \left(\sum_{p=1}^n \sum_{q=1}^{n-1} \sum_{h=1}^n \left[|(s_p, s_h)| \cdot \Phi_{p,q} \cdot \Phi_{h,q+1} - T_{\max} \right] \right) \cdot \varrho(i, j) \quad (4.13)$$

$$- d \cdot \lambda(i, j) \quad (4.14)$$

$$+ e \cdot \epsilon(i, j) \quad (4.15)$$

$$- f \cdot \zeta(i, j), \quad (4.16)$$

where $\gamma(x)$ is

$$\gamma(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}, \quad (4.17)$$

and the following partial terms.

$$\varrho(i, j) = \begin{cases} \sum_{h=1}^n \left[|(s_i, s_h)| \cdot (\Phi_{h,j-1} + \Phi_{h,j+1}) \right] & \text{if } 1 < j < n \\ \sum_{h=1}^n \left[|(s_i, s_h)| \cdot \Phi_{h,j-1} \right] & \text{if } j = n \\ \sum_{h=1}^n \left[|(s_i, s_h)| \cdot \Phi_{h,j+1} \right] & \text{if } j = 1 \end{cases}, \quad (4.18)$$

where $|(s_i, s_j)| = \|s_i - s_j\|$ is the Euclidean distance between the locations s_i and s_j , and $\Phi_{i,j}$ is the state of the matrix Φ . The function λ is

$$\lambda(i, j) = \begin{cases} 1 & \text{if } (i = 1 \text{ and } j = 1) \text{ or } (i = n \text{ and } j = n) \\ 0 & \text{otherwise} \end{cases}. \quad (4.19)$$

The term $\epsilon(i, j)$ is

$$\epsilon(i, j) = \begin{cases} 1 - \sum_{h=1}^n \Phi_{i,h} & \text{if } j = n + 1 \\ -\Phi_{i,n+1} & \text{if } j \leq n \end{cases}, \quad (4.20)$$

where $\Phi_{i,j}$ is the state of the matrix Φ and the state $\Phi_{i,n+1}$ is the $(n + 1)$ -th column of the location s_i that is set to 1, if the location is in the route. The term $\zeta(i, j)$ is a function

$$\zeta(i, j) = \begin{cases} \varsigma_i & \text{if } j = n + 1 \\ 0 & \text{if } j \leq n \end{cases}, \quad (4.21)$$

where ς_i is the reward of the location s_i .

4.4 Update Algorithm

In this section, the update algorithm proposed in [8] is described. First, the parameters are set to initial values, and the state matrix is initialized. Next, by updating the network, local

minima are found. Last, a route that represents the state matrix is constructed and further improved.

The parameters are set to values $a = b = 1$, $c = 20$, $d = 10$, $g = 20$, $f = 15$, and the time step Δt is set to 1. After the parameters are initialized, a process of finding local minima is repeated until the network is stabilized. The process of the network stabilization goes as follows. A state with the largest value of $\frac{\partial E}{\partial \Phi_{i,j}}$ is selected from a random row or column. The state $\Phi_{i,j}$ is then updated by the activation function (Eq. 4.1). If the absolute value of $\frac{-\partial E}{\partial \Phi_{i,j}} \Delta t$ is smaller than a threshold $\Delta \vartheta = 2$ three times in succession, a local minimum is found. Then a threshold is applied to the state matrix. From each column of the matrix Φ , a state with its value larger than a threshold ($T = 0.5$) is selected and set to 1, others states are set to 0. If there are more states larger than the threshold per the column, the state with the largest value is selected.

$$\Phi_{i,j} = \begin{cases} 1 & \text{when the location } s_i \text{ is visited at the } j\text{-th position} \\ 0 & \text{otherwise} \end{cases}. \quad (4.22)$$

After the application of the threshold, only n states of the matrix are active, i.e., their value is equal to 1. From these states, a route is constructed according to Eq. 4.22. The 2-Opt is applied to the constructed route. If the route is not feasible, i.e., the budget constraint is violated after the improvement, the parameter f is decreased. Otherwise, the parameter is increased. The route is further examined and if the total route length exceeds the travel budget, a location with the largest ratio of the distance to the reward is removed from the route. If the total route length does not exceed the travel budget, a location is added via the cheapest insertion. The examination is over when the budget constraint is satisfied. If the route is feasible and so far has the best total sum of rewards, the route is marked as the best solution. The matrix Φ is then adjusted to reflect the current route and is slightly perturb.

If the number of repetitions exceeds the prescribed limit, the algorithm continues with the process of finding a local minima. Otherwise, the algorithm continues from the initialization of the parameters. If the number of iterations exceeds the given limit, the algorithm stops.

4.5 Proposed Implementation of the HNN for the OP

The algorithm used in the evaluation reported in Chapter 7 of this thesis differs from the original algorithm [8] described in Section 4.4. The modifications of the algorithm are proposed to improve the performance of the network, and because the original algorithm [8] was unclear in some parts. The modified algorithm and the implementation of the based solver are described in this section.

4.5.1 Modified Update Algorithm

The modified algorithm consists of three parts, see Fig. 4.2. The first is the initialization of the parameters and the state matrix Φ . Next follows the update of the state matrix until the network is stabilized. Last, the route that reflects the state matrix is constructed and improved. The parameters are also initialized to the values $a = b = 1$, $c = 20$, $d = 10$, $g = 20$ and $f = 15$, the time step Δt is set to 0.001 and the threshold of the local minima is $\Delta \vartheta = 2$. If the time step is smaller, the local minimum is found faster. The state matrix Φ is initialized

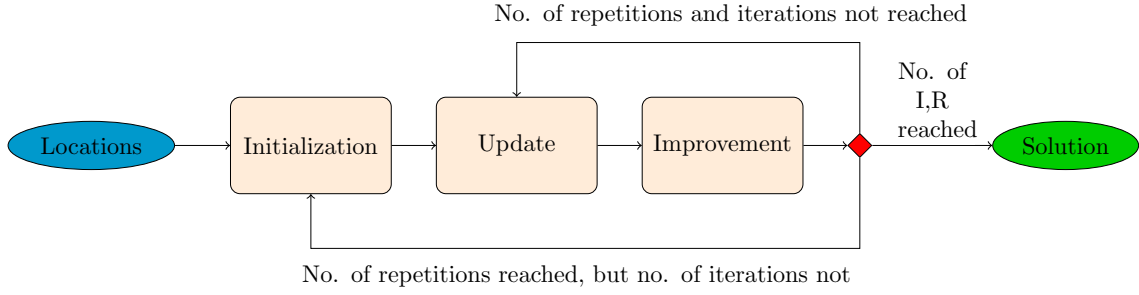


Figure 4.2: The structure of the solver based on the HNN consists of three main parts: Initialization, Update, and Improvement. I denotes iterations, R denotes repetitions.

to random values from the interval $[0, 0.5]$. Then the matrix is updated and reasons for doing particular modifications are listed.

First, all counters for calculating the local minima are reset, and the state matrix is reinitialized. Then a random row from the matrix Φ is selected. The whole row is selected to improve the performance of the network, the local minimum is found in the less number of updates, and it does not affect the quality of the solution. All states of the row are updated by the activation function (Eq. 4.1). Three updates from the start of the update, the middle of the update, and the end of the update are presented in Fig. 4.3 to show how the network is stabilizing while satisfying the OP constraints. All states are compared with the threshold $\Delta\vartheta$ to check if the network is stabilized. If the value of a state is less than the threshold three times in succession, a local minimum is found. Otherwise, another random row is selected, and the process of updating is repeated. When the network is stabilized, a filter is applied to the state matrix, i.e., from each column of the matrix Φ , a state with the largest value is selected and set to 1, and the other states are set to 0.

A route that reflects the matrix is constructed such that the state $\Phi_{i,j} = 1$ represents the location s_i at the j -th position of the route. The route obtained from the matrix might contain duplicate locations. As mentioned in Section 4.3, if the number of activated states after the filter application is smaller than n , the activated states are consecutively repeated, i.e., the duplicates of the location s_i appear in the route just behind the original location s_i , and therefore the duplicates can be removed from the route. The current route may not be feasible, therefore the 2-Opt heuristic is applied. If the route is feasible after the heuristic improvement, then the parameter f is increased. Otherwise, f is decreased to satisfy the length constraint in the next repetition. The route is further examined and improved. The examination and improvement are as follows. If the total route length exceeds the travel budget, then a location s_{remove} is selected from the route and then removed. The location is selected according to

$$s_{\text{remove}} = \arg \max_{s_i \in \mathcal{R}} \frac{|(s_i, s_{i-1})|}{\varsigma_{s_i}}, \quad (4.23)$$

where s_i is an location of the route, $|(s_i, s_{i-1})| = \|s_i - s_{i-1}\|$ is the Euclidean distance of two locations, and ς_{s_i} is the reward of the location s_i . If the route length does not exceed the travel budget, a location s_{insert} is inserted to the route. The location is selected from

Φ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
0	1.0000	0.4402	0.4851	0.5642	0.4175	0.3770	0.4375	0.1333	0.2048	0.2706	0.1196	0.0132	0.3270	0.1161	0.2070	0.2803	0.4389	0.0682	0.0765	0.3551	0.2783	0.3053	0.4484	0.4752	0.2652	0.6077	0.1983	0.0846	0.2171	0.1434	0.3994	0.1907	0.6536	0.0000

(a) The first update

Φ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

(b) The middle update

Φ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

(c) The last update

Figure 4.3: The state matrix Φ during updates of the network while solving problem from Tsiligrides Set 3 with $T_{\max} = 45$ with $I = 0$ and $R = 1$.

Algorithm 1: The OP solver based on the Hopfield Neural Network

Input: S - the set of locations, I - no. of iterations, R - no. of repetitions

Output: $\mathcal{R}_{\text{best}}$ - the sequence of locations, S_k - the subset of locations, Σ - the sequence of visits, k - no. of visited locations

```

1 begin Initialization
2    $graph \leftarrow loadGraph(S)$ ;
3    $neuralNetwork \leftarrow initializeNetwork()$ ;
4    $parameters \leftarrow initializeParameters()$ ;
5    $i \leftarrow 0$ ;
6    $r \leftarrow 0$ ;
7 end
8 begin Iteration - graph, neuralNetwork, parameters, i, r
9   for  $i < I$  do
10     $parameters \leftarrow initialize$ ;
11     $neuralNetwork \leftarrow initialize$ ;
12    for  $r < R$  do
13       $neuralNetwork \leftarrow resetLocalMinima()$ ;
14      while not local minima is found do
15         $row \leftarrow getRandomRow()$ ;
16         $neuralNetwork \leftarrow updateActivationLevel(row)$ ;
17      end
18       $neuralNetwork \leftarrow applyFilter()$ ;
19       $\mathcal{R} \leftarrow constructRoute(\Phi)$ ;
20       $\mathcal{R} \leftarrow improveRoute()$ ;
21      if  $\mathcal{R}$  is feasible then
22         $parameters \leftarrow decreaseF()$ 
23      else
24         $parameters \leftarrow increaseF()$  ;
25      end
26       $\mathcal{R} \leftarrow examineRoute()$ ;
27       $neuralNetwork \leftarrow adjustNetwork(\mathcal{R})$ ;
28      if  $\mathcal{R}$  is better than  $\mathcal{R}_{\text{best}}$  then
29         $\mathcal{R}_{\text{best}} \leftarrow \mathcal{R}$ ;
30      end
31       $r \leftarrow r + 1$ ;
32    end
33     $i \leftarrow i + 1$ ;
34  end
35 end

```

the set of the unvisited locations denoted as $\overline{\mathcal{R}}$:

$$s_{\text{insert}} = \arg \max_{\substack{s_i \in \overline{\mathcal{R}}, \\ s_a \in \mathcal{R}, s_b \in \mathcal{R}}} \frac{\varsigma_{s_i}}{|(s_a, s_i)| + |(s_i, s_b)| - |(s_a, s_b)|}. \quad (4.24)$$

The formula is proposed according to the heuristic in [14]. The selection method of s_{insert} has been proved to be the best between the explored methods. The process of inserting and removing locations is repeated until the budget constraint is satisfied or the number of prespecified attempts is reached. After this examination, the route is feasible. If the route is the best solution obtained so far, it is marked as the best. Then the state matrix is adjusted to reflect the solution, i.e., the state is set to either 0.99 if the respective location is in the final route, or to 0.01, if it is not. The values are not rounded because with round values 1 and 0 the network did not function properly, it got stuck in a global minima and could not be stabilized.

If the number of repetitions R has not been reached, the algorithm continues with the re-setting the counters, the reinitialization of the matrix, and finding a new local minima. If the number of repetition has been reached, but the number of the prespecified iterations I has not been exceeded, the algorithm continues with the reinitialization of the parameters. Otherwise, the algorithm stops. The whole procedure is summarized in Algorithm 1.

Implementation

The HNN-based solver is implemented in C++ and the implementation uses libraries provided by the thesis supervisor. The implementation is based on an object-oriented programming. The objects represent parts of the solver: the graph that represents distribution of locations, the network representation, the energy function, the parameters and the route representation.

The object-oriented UML class diagram is shown in Fig. 4.4. The main object is the class `NNOP` that encapsulates methods `iterate()` and `initialize()`. The instances are initialized in method `initialize()`. The method `iterate()` is the main method of the solver. The detailed description of the implemented methods is on the enlisted CD, and the activity diagram of the solver is depicted in Fig. 4.5.

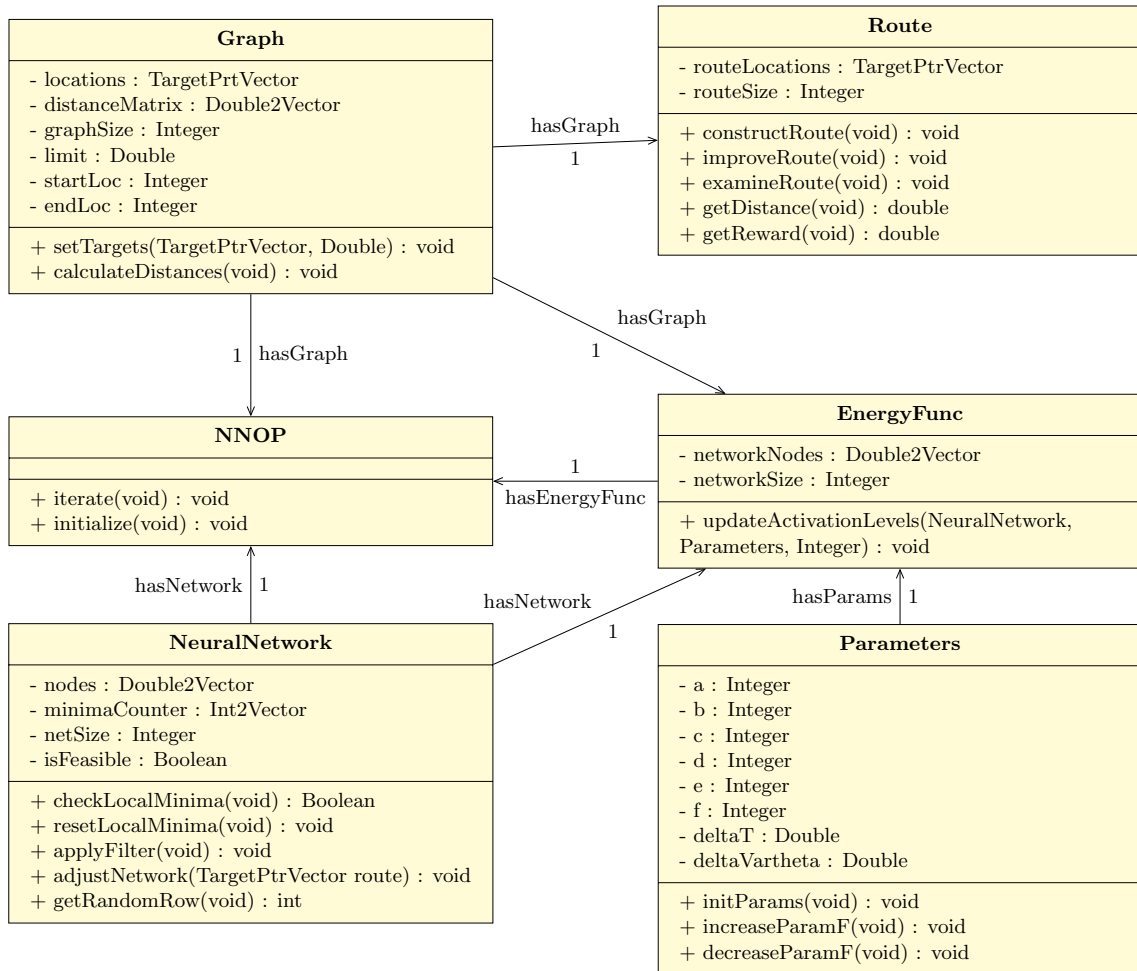


Figure 4.4: The UML class diagram of the solver based on the Hopfield Neural Network [8], where `TargetPtrVector` is a vector of `STarget*` that represents sensor locations, `Double2Vector` labels vector of vectors of Doubles, and `Int2Vector` labels vector of vectors of Integers.

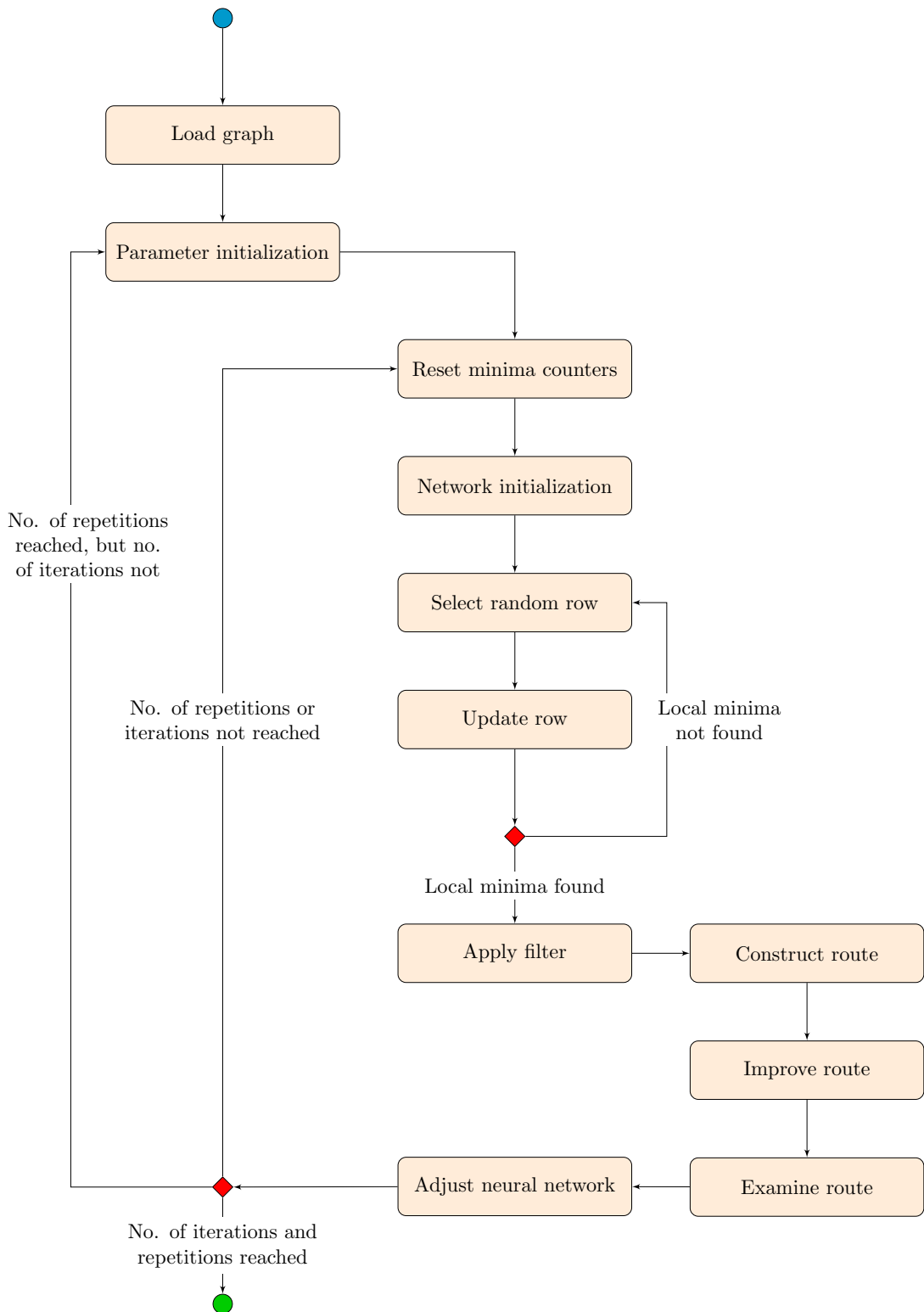


Figure 4.5: The activity diagram the solver based on the Hopfield Neural Network [8].

Hopfield Neural Network Extension to the Orienteering Problem with Neighborhoods

The Hopfield Neural Network (HNN) proposed for the Orienteering Problem (OP) in [8] was designed to fit the discrete problem of the OP. In this chapter, the HNN solver proposed in Chapter 4 is modified to address the Orienteering Problem with Neighborhoods (OPN).

5.1 Hopfield Neural Network for the OPN

The HNN consists of the data representation, the energy function, and the traditional heuristic [13]. The data representation and the energy function need to be modified as follows to reflect the OPN. Let m be the number of fixed waypoints. Waypoints are situated on the disk with the radius δ centered at the location s_i . A set of possible waypoints of the location s_i ,

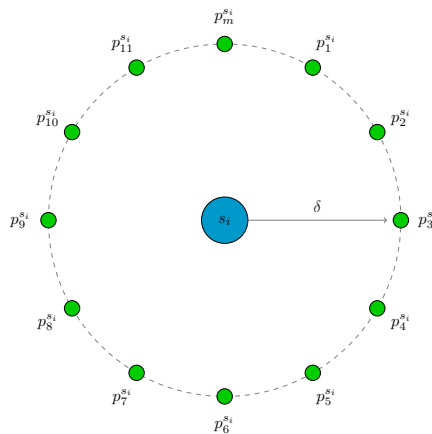


Figure 5.1: The sampled variant of the OPN. The location s_i is represented by its waypoints $p_k^{s_i}$, where $k \in \{1, \dots, m\}$, the distance between the waypoint and the location is equal to the sensing radius δ .

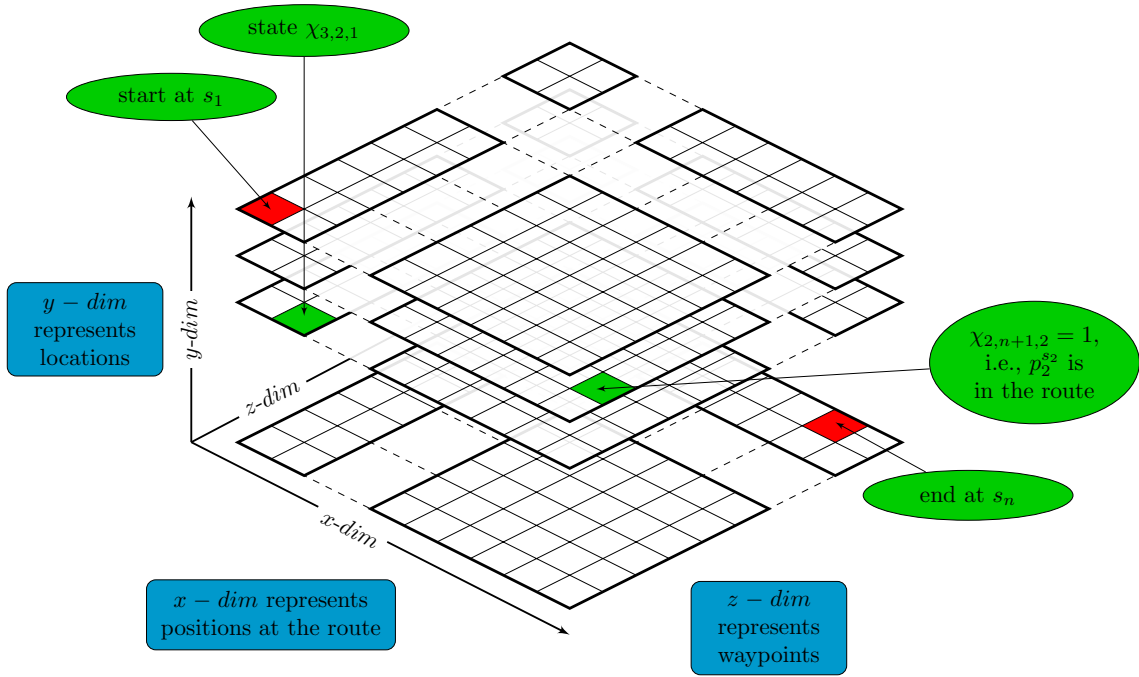


Figure 5.2: The representation of the HNN for the OPN. The three dimensional matrix represents the sensor locations: $y-dim$, and their respective waypoints: $z-dim$, and the position of the waypoint of the route is represented by $x-dim$.

$i \in [2, n-1]$ is denoted as $P_{s_i} = \{p_1^{s_i}, \dots, p_m^{s_i}\}$, $P_{s_1} = \emptyset$ and $P_{s_n} = \emptyset$. This sampled variant of the OPN is shown in Fig. 5.1. The sampled OPN can be formulated as a graph $G(V, E)$, where V is set of n clusters - the possible waypoints of the respective locations. Each cluster contains m vertices, except the start and end cluster (with only single waypoint), and is associated with the reward $\varsigma_i \geq 0$, where $i \in [1, n]$ and $\varsigma_1 = \varsigma_n = 0$. The edge (without orientation) between every two vertices i and j from two different clusters exists and it is associated with the cost of the traveled distance between the vertices. The graph can be interpreted as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n,n}$, where n is the number of clusters.

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if there is an edge between clusters } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}. \quad (5.1)$$

The matrix \mathbf{A} needs to be expanded to reflect individual vertices within each cluster. Then the matrix is used for the conceptualization of the network as follows.

Let $\chi \in \mathbb{R}^{n,n+1,m}$ be the 3D state matrix that represents the HNN for the OPN, see the visualization of the matrix presented in Fig. 5.2. The activation level $\chi_{i,j,k}$ is referred as the *state* and denotes that data for the location s_i were collected from the waypoint $p_k^{s_i}$ and the waypoint is visited at the j -th position of the route. The value of the state is:

$$\chi_{i,j,k} = \begin{cases} 1 & \text{if waypoint } p_k^{s_i} \text{ is visited at } j\text{-th position of the route} \\ 0 & \text{otherwise} \end{cases}. \quad (5.2)$$

The state $\chi_{1,1,1}$ represents the start location and the state $\chi_{n,n,m}$ represents the end location. These states are prespecified and set to 1. States $\chi_{i,n+1,k}$ are set to 1, if the waypoint $p_k^{s_i}$ is

in the route, otherwise they are set to 0. Each state of the matrix χ is updated according to the activation function:

$$\chi_{i,j,k} = \frac{1}{1 + \exp^{-\alpha}}, \text{ with} \quad (5.3)$$

$$\alpha = \ln(\chi_{i,j,k}) - \ln(1 - \chi_{i,j,k}) - \frac{\partial E}{\partial \chi_{i,j,k}} \Delta t, \quad (5.4)$$

where E is the energy function similar to the one in Section 4.3, and Δt is the time step. The modified energy function is as follows.

A sum over the waypoints is added to each term of E for the OP (Eq. 4.4) except the term Eq. 4.6. Two sums over the waypoints are added to the term Eq. 4.6 to ensure that the total route length does not exceed the travel budget. The derivative of the energy function used in the activation function Eq. 5.4 is following

$$\frac{\partial E}{\partial \chi_{i,j,k}} = a \cdot \sum_{\substack{h=1 \\ h \neq i}}^n \sum_{\substack{l=1 \\ l \neq k}}^m \chi_{h,j,l} \quad (5.5)$$

$$+ b \cdot \left[\sum_{h=1}^n \sum_{r=1}^n \sum_{l=1}^m \chi_{h,r,l} - n \right] \quad (5.6)$$

$$+ c \cdot \gamma \left(\sum_{r=1}^n \sum_{q=1}^{n-1} \sum_{h=1}^n \sum_{o=1}^m \sum_{l=1}^m \left[|(p_o^{s_r}, p_l^{s_h})| \cdot \chi_{r,q,o} \cdot \chi_{h,q+1,l} - T_{\max} \right] \right) \cdot \varrho(i, j, k) \quad (5.7)$$

$$- d \cdot \lambda(i, j, k) \quad (5.8)$$

$$+ e \cdot \epsilon(i, j, k) \quad (5.9)$$

$$- f \cdot \zeta(i, j, k), \quad (5.10)$$

where $\gamma(x)$ is

$$\gamma(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}. \quad (5.11)$$

The function $\varrho(i, j, k)$ is

$$\varrho(i, j, k) = \begin{cases} \sum_{h=1}^n \sum_{l=1}^m \left[|(p_k^{s_i}, p_l^{s_h})| \cdot (\chi_{h,j-1,l} + \chi_{h,j+1,l}) \right] & \text{if } 1 < j < n \\ \sum_{h=1}^n \sum_{l=1}^m \left[|(p_k^{s_i}, p_l^{s_h})| \cdot \chi_{h,j-1,l} \right] & \text{if } j = n \\ \sum_{h=1}^n \sum_{l=1}^m \left[|(p_k^{s_i}, p_l^{s_h})| \cdot \chi_{h,j+1,l} \right] & \text{if } j = 1 \end{cases}, \quad (5.12)$$

where $|(p_k^{s_i}, p_l^{s_h})| = \|p_k^{s_i} - p_l^{s_h}\|$ is the Euclidean distance between the waypoints $p_k^{s_i}$ and $p_l^{s_h}$. The partial functions are

$$\lambda(i, j, k) = \begin{cases} 1 & \text{if } (i = 1 \text{ and } j = 1 \text{ and } k = 1) \text{ or } (i = n \text{ and } j = n \text{ and } k = m) \\ 0 & \text{otherwise} \end{cases}. \quad (5.13)$$

The state $\chi_{i,j,k}$ represents the waypoint $p_k^{s_i}$ visited at the j -th position of the route

$$\epsilon(i, j, k) = \begin{cases} 1 - \sum_{h=1}^n \chi_{i,h,k} & \text{if } j = n + 1 \\ -\chi_{i,n,k} & \text{if } j \leq n \end{cases}. \quad (5.14)$$

$$\zeta(i, j, k) = \begin{cases} \varsigma_i & \text{if } j = n + 1 \\ 0 & \text{if } j \leq n \end{cases}, \quad (5.15)$$

where ς_i is the reward of the waypoints $p_k^{s_i}$, $k \in [1, m]$.

5.2 Implementation of the HNN Solver for the OPN

The HNN based solver for the OPN has the same structure as the solver for the OP presented in Fig. 4.2, however, some parts are slightly changed to reflect the sampled OPN variant. In the method `setTargets(TargetPtrVector, Double)` of the class `Graph` depicted in Fig. 4.4, the sampled variant of the OPN is created. For each location, m possible waypoints are generated at the location s_i according to trigonometric formulas:

$$x = s_i^x + \delta \cdot \cos(i\Delta\theta/360), \quad (5.16)$$

$$y = s_i^y + \delta \cdot \sin(i\Delta\theta/360), \quad (5.17)$$

for $i \in \{1, \dots, m\}$ where x, y are coordinates of the possible waypoint, δ is the sensing radius, $\Delta\theta$ is the angle at which the waypoint is located on the disc, and s_i^x, s_i^y are coordinates of the location s_i . The neural network represented by the 3D state matrix is implemented by using `Double3Vector`, i.e., a vector of vectors of vectors of Doubles. The waypoints are also calculated for the start and the end location, but are completely ignored, to simplify the implementation.

The algorithm for solving the OPN by the HNN proposed in this chapter consists of three parts depicted in Fig. 4.2. The initialization, the update, and the improvement parts are built on the proposed algorithm for the OP [8] and modified to reflect the OPN as follows.

In the initialization part, only the parameter Δt is changed. The time step is initialized with the value $\Delta t = 0.0001$.

In the second part (the update), the randomly selected row i and its respective states $\chi_{i,j,k}$, $j \in [1, n]$, $k \in [1, m]$ are updated by the activation function. The counters for finding local minima are updated according to the location and not the waypoint, therefore the network is stabilized faster.

The most important change is in the last part of the solver, where the route is constructed from the state matrix and further improved. The obstacle is to construct the route from the three-dimensional matrix. Given a state $\chi_{i,j,k}$ with m possible waypoints, the waypoint $p_k^{s_i}$ is encoded as the location p at the j -th position of the route, and it is labeled by the value $p = i \cdot m + k$. After improving and examining the route, the matrix is adjusted to reflect the route. The route location p at the j -th position is represented by the state $\chi_{i,j,k}$, where $i = p/m$, j is the position at the route, and $k = p \bmod m$.

Self-Organizing Map

The application of the Self-Organizing Map (SOM) for solving the Orienteering Problem (OP) and the Orienteering Problem with Neighborhoods (OPN) is reported in [10] and it is briefly described in Section 3.1; however, the proposed modifications to improve the quality of solutions by the SOM-based approach are directly modifying the approach [10], and therefore the SOM-based solution of the orienteering problems is described in detail in this chapter to make the thesis more self-contained. The SOM is using the unsupervised learning, where particular locations are presented to the two-layer network where the locations represent the input layer, while the output layer consists of the array of connected neurons. The neuron weights share the space with the input locations and thus the array forms a ring of neurons which evolves in the same space as the input locations, therefore the array of neurons directly represents the requested route in the input space. The main difficulty of addressing the solution of the OP and the OPN by the SOM is to satisfy the travel budget T_{\max} . Therefore, if the route represented by the network would exceed T_{\max} after the adaptation towards the particular location, the network is not adapted, i.e., the so-called conditional adaptation of the SOM. Besides, the author of [10] proposes to weight the power of the adaptation according to the rewards associated with locations. The conditional adapt and the learning according to [10] can be described as follows.

Each epoch, all locations $S = \{s_1, \dots, s_n\}$ are given to the network (in random order to avoid the local optima) and the adaptation of the network to the particular $s \in S$ consists of the determination of the closest point p_s , which is the point of the ring that has the shortest distance to s . Since the adaptation is performed only if the network would represent a route not exceeding T_{\max} , the network is saved as a sequence of M neurons $\mathcal{N} = \{\nu_1, \dots, \nu_M\}$, and therefore the network can be easily reverted to its previous state, if the conditional adapt is not performed. The closest point p_s is used for the weights of the new winner neuron ν^* added to the network. Then, the winner neuron with its neighboring neurons is adapted towards the location s according to the neighboring function $f(G, d)$ similarly as in the regular SOM [20].

$$f(G, d) = \begin{cases} e^{-\frac{d^2}{G^2}} & \text{for } d < 0.2M \\ 0 & \text{otherwise} \end{cases}, \quad (6.1)$$

where d is the number of neurons in the ring, M is the current number of the neurons in

the ring, and G is the learning gain that is updated according to the gain decreasing rate α after each epoch.

The initial ring of neurons consists of neurons that represent the start and the end locations, $\mathcal{N} = \{\nu_1, \nu_{\text{end}}\}$. The learning parameters are initialized to the following values, the learning gain $G = 10$, the gain decreasing rate $\alpha = 0.1$, the default learning rate $\mu = 0.6$, and the maximal reward $R_{\text{max}} = \arg \max_{s \in S} \zeta(s)$ is determined. The current best found solution $T = (s_1, s_n)$ is set with its sum of rewards to $R = 0$ and the learning epoch counter is set to $i = 1$.

Each learning epoch, the network is given a permutation of all locations, except for the start and the end locations, $\Pi \leftarrow \text{permute}(S \setminus \{s_1, s_n\})$, since the visitation of s_1 and s_n is prescribed in the OPN. For each location s from Π , the following steps are performed. First, the current configuration of the network is saved as $\mathcal{N}' \leftarrow \mathcal{N}$. Then two winner neurons are determined, a summary of the winner selection is depicted in Algorithm 2.

Algorithm 2: The winner neuron selection and determination of the closest point p_s from [10]. The highlighted lines have been modified in this chapter.

Input: s - the location, \mathcal{N} - the set of neurons, T_{max} - the travel limit, i - epoch
Output: \mathcal{N} - the set of neurons, p_s - the closest point

```

1 begin winnerWeights
2   Get all neurons marked as winner in the current epoch  $i$ :
    $\mathcal{N}_{\text{win}} \leftarrow \text{winners}(\mathcal{N} \setminus \{\nu_1, \nu_{\text{end}}\}, i)$ .
3   Let each winner  $\nu \in \mathcal{N}$  be associated with target location  $s_\nu = s(\nu)$  and reward
    $\zeta(s_\nu)$ .
4   Determine the winner  $\nu_f$  which has the longest distance to its associated location
    $s_{\nu_f}: \nu_f = \arg \max_{\nu \in \mathcal{N}_{\text{win}}} |(\nu, s_\nu)|$ .
5   Determine the winner  $\nu_l$  which associated location  $s_{\nu_l}$  has the lowest reward:
    $\nu_l = \arg \min_{\nu \in \mathcal{N}_{\text{win}}} \zeta(s_\nu)$ .
6   Determine the closest point  $p_s$  of the ring  $\mathcal{N}$  to the location  $s$ .
7   if the expected route length after adapting  $p_s$  would be longer than  $T_{\text{max}}$  then
8     if  $\zeta(s_{\nu_f}) < \zeta(s)$  AND  $|(\nu_f, s(\nu_f))| > |(p_s, s)|$  then
9       Remove  $\nu_f$  from the ring  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\nu_f\}$ .
10    end
11    if  $\zeta(s_{\nu_l}) < \zeta(s)$  AND  $|(\nu_l, s(\nu_l))| > |(p_s, s)|$  then
12      Remove  $\nu_l$  from the ring  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\nu_l\}$ .
13    end
14  end
15  return  $(\mathcal{N}, p_s)$  ;
16 end

```

The additional winner neurons are removed from the network in the method `winnerWeights` of Algorithm 2 to support the adaptation of the network to s while not exceeding the travel budget. The first neuron ν_f from the ring has the longest distance to its associated location. The second neuron ν_l represents the location with the lowest reward. Along with the additional neurons, the closest point p_s is determined as the point of the ring that has the shortest distance to the location s . The winner ν_f is removed from the ring $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\nu_f\}$, if the reward of the location associated with the neuron is lower than the reward of the current location s

and the distance between the neuron and its associated location is larger than the distance between p_s and s . The winner ν_l is removed from the ring $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\nu_l\}$, if the reward of the location associated with the neuron is lower than the reward of the current location s and the distance between the neuron and its associated location is larger than the distance between p_s and s . If p_s is situated in the neuron ring, determine the previous winner ν_p and the next winner ν_n . Then the conditional adapt follows.

If the length of the new tour $L_{T_{\text{win}}}$ represented by the neuron ring including the closest point p_s satisfies the travel budget, then a new neuron ν^* with weights identical to p_s is created and added to the neuron ring \mathcal{N} at the position corresponding to p_s . The winner neuron ν^* and its neighboring neurons are adapted towards the location s according to the neighboring function (Eq. 6.1). The location s is associated with the neuron ν^* and the neuron is marked as the winner for the current epoch. If the length $L_{T_{\text{win}}}$ exceeds the travel budget, the network is reverted to the previous state $\mathcal{N} \leftarrow \mathcal{N}'$.

At the end of each learning epoch, the update is made to remove all non-winner neurons from the ring, and the learning parameters are updated. If the total sum of the rewards of the new solution is higher than the sum of the rewards of the current best solution, then the new solution is marked as the best solution.

For the OPN, the SOM solver is slightly modified to reflect the data collected from the location s within the sensing radius δ . In [10], an alternative location p' is determined between the point p_s and the location s . Then data from the location s can be read at the alternative location p' , $|(p', s)| \leq \delta$, therefore p' is used instead of the location s in the adaptation and the learning epoch. The final route is then recreated from the sequence of winner neurons associated with alternate locations.

6.1 Improvements of the SOM for the OPN

We propose two types of SOM modifications. The first type of modifications is based on the selection of the winner neuron, and the second type of modifications is based on the combinatorial heuristic.

6.1.1 Proposed Modification of the Winner Neuron Selection

Even though the approach [10] addressed the OPN, the quality of the solution suffers from stinking in local optima. It is partially caused by the heuristic nature of the SOM, but also because the only way how to escape the local optima is in the randomized construction of the route in each learning epoch, where more rewarding location is visited only if T_{max} is not exceeded after removing ν_l and ν_f . Therefore, we focus on improving the solution quality by considering different approaches how to identify locations that should not be visited by the route in benefit of adding more rewarding locations, and thus having a higher sum of the collected rewards.

In the method `winnerWeights` from [10] depicted in Algorithm 2 two neurons ν_f and ν_l are selected and if the conditions on the lines 8 and 11 are met, the neurons are removed from the ring \mathcal{N} . Along with the neurons, the closest point p_s of the ring to the location s is determined. If the condition of the adapt is met, the winner neuron ν^* representing p_s is created.

A new selection of the winner neuron and new removal methods denoting the determination of the winner neurons that are removed from the ring are proposed to improve the quality of

Table 6.1: Combinations of proposed methods for the selection of the winner and selected neurons, and the method from [10].

	SOM [10]	SOMv1	SOMv2	SOMv3	SOMv4	SOMv5
winner neuron	ν^*	ν_n^*	ν_n^*	ν_n^*	ν^*	ν^*
	ν_f	ν_{l_1}		ν_{l_1}	ν_{l_1}	
winners	ν_l	ν_{l_2}	ν_r	ν_{l_2}	ν_{l_2}	ν_r
		ν_r				

the found solutions. The removal methods of neurons ν_l and ν_f are determined on the lines 4 and 5 of Algorithm 2. These methods are replaced by the new proposed methods that determine the neurons ν_r , ν_{l_1} , and ν_{l_2} to be removed. The neuron ν_r is the neuron of the ring \mathcal{N} which has the largest ratio of the distance (between the neuron and its associated location) to the reward associated with the location:

$$\nu_r = \arg \max_{\nu \in \mathcal{N}_{\text{win}}} \frac{|(\nu, s_\nu)|}{\varsigma(s_\nu)}. \quad (6.2)$$

The removal method for the neurons ν_{l_1} and ν_{l_2} is inspired by the determination of the neuron ν_l of [10]. The neurons are determined together and represent two neurons which are associated with the lowest rewards.

On the line 6 of Algorithm 2, the closest point of the ring to the location s is determined. After the conditional adapt, p_s is represented as the winner neuron ν^* . The new point p'_s for the location s is determined as the weighted closest point, i.e., the point with the minimal ratio of the reward associated with s to the distance between s and the ring. After the conditional adapt, the point p'_s is represented by the newly proposed winner neuron ν_n^* :

$$\nu_n^* = \arg \min_{\nu \in \mathcal{N}} \frac{\varsigma(s_\nu)}{|(\nu_{\text{prev}}, s)| + |(\nu, s)| - |(\nu, \nu_{\text{next}})|}. \quad (6.3)$$

The proposed removal methods of the selected neurons and the closest point is combined with the original determination of the closest point from [10] to improve the found solutions. The combinations are summarized in Table 6.1.

6.1.2 SOM-based Initialization of the VNS

The idea of this proposed modification is that the final solution from the SOM solver [10] is used as the initial solution for the VNS solver [22]. The motivation for the combination of the SOM-based approach with the combinatorial heuristic is to improve the found solutions from the neural network. Moreover, we also aim to find out, if the VNS solver performs better with a random initial solution or with the quality solution obtained by the SOM solver.

The VNS solver from [22] consists of four core methods: `foundInitialSolution`, `shake`, `localSearch`, `selectLocations`, see Algorithm 3. The methods `shake` and `localSearch` have been described and the VNS has been overviewed in Chapter 3. The `foundInitialSolution` method generates the first feasible solution to use as the initial solution for the solver. The method is changed for the method `getSomSolution` to pass a solution generated by the SOM-based solver, and use it as the initial solution for the solver. The method `select-`

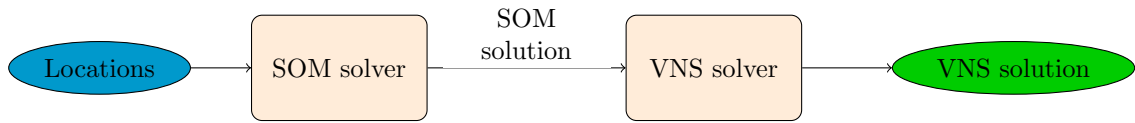


Figure 6.1: Data flow of the SOM-based initialization of the VNS.

Algorithm 3: The randomized VNS solver from [22], where $\mathcal{L}(\Sigma_k'')$ denotes the length of the solution. The highlighted line is modified for the SOM solutions.

Input: S - the set of locations, T_{\max} - the travel limit
Output: Σ_k - the representation of the solution

```

1  $P, \Sigma_k \leftarrow \text{foundInitialSolution}()$  ;
2 while termination condition is met do
3    $P' \leftarrow \text{shake}(P)$ ;
4    $P'' \leftarrow \text{localSearch}(P')$  ;
5    $\Sigma_k'' \leftarrow \text{selectLocations}(P'', T_{\max})$  ;
6   if  $\mathcal{L}(\Sigma_k'') \leq T_{\max}$  then
7      $P \leftarrow P'$  ;
8      $\Sigma_k \leftarrow \Sigma_k''$  ;
9   end
10 end
  
```

Locations of Algorithm 3 iteratively determines the number of the selected locations to be visited, while the length of the tour does not exceed the travel budget T_{\max} .

The data flow of the proposed modification is depicted in Fig. 6.1. First, a solution is generated by the SOM-based solver. The solution is presented to the VNS solver as the initial solution. The VNS solver then runs for the limited time t_m and calculates the best solution.

The solution of the SOM-based solver needs to be discretized to be used by the VNS [22]. Therefore a solution is represented by the label of the location s and coordinates of the waypoint p_s that represents the location s . the initial solution for the VNS solver is thus created by placing k waypoints on the disk with the sensing radius δ centered at the location s . The first placed waypoint is the waypoint from the SOM solution. Then $k - 1$ waypoints are equally distributed on the disk. The VNS solver then runs for the limited time t_m .

Results

The objective of this thesis is to compare and evaluate the neural network (NN) based solvers for solving the Orienteering Problem (OP) and the Orienteering Problem with Neighborhoods (OPN). The first solver is the Hopfield Neural Network (HNN) based solver proposed by Wang *et al.* in [8] consisting of the complex energy function and robust data representation. The second NN-based solvers are based on the Self-Organizing Map (SOM) proposed in [10] and here improved by two approaches. In the first modification, a determination of the winner neuron and the removal methods are combined in five SOM variants. The second proposed modification of [10] uses the SOM-based initialization in the combinatorial Variable Neighborhood Search (VNS) heuristic.

The performance of all these solvers is compared with existing approaches for the OP and the OPN. The proposed solvers and available existing methods (further marked by *) are following.

HNN-OP is our implementation of the HNN based solver for the OP that was proposed by Wang *et. al* in [8].

HNN-OPN is the extension of the HNN-OP solver for the sampled OPN.

NoNN is the solver of the OP based on the heuristics used in HNN-based approach [8].

SOMv1 is the SOM solver based on [10] for the OP and the OPN. It combines new winner neuron selection ν_n^* and new removal methods ν_{l_1} , ν_{l_2} , and ν_r .

SOMv2 is the SOM solver based on [10] for the OP and the OPN. It combines new winner neuron selection ν_n^* and new removal method ν_r .

SOMv3 is the SOM solver based on [10] for the OP and the OPN. It combines new winner neuron selection ν_n^* and new removal methods ν_{l_1} and ν_{l_2} .

SOMv4 is the SOM solver based on [10] for the OP and the OPN. It combines the original winner neuron selection ν^* and new removal methods ν_{l_1} and ν_{l_2} .

SOMv5 is the SOM solver based on [10] for the OP and the OPN. It combines the original winner neuron selection ν^* and new removal method ν_r .

SOM-VNS is the modified randomized VNS solver from [22] for the OP and the sampled OPN. The initial solution of the solver is the solution from the SOM solver [10].

VNS* is the OP solver of [22] using the randomized generation of the initial solution.

MILP* is the Mixed Integrated Linear Programming solver for the OP.

GRASP* is the OP solver proposed in [16].

SOM1* is the SOM based solver for the OP and the OPN proposed by Faigl *et. al* in [21].

SOM2* is the SOM based solver for the OP and the OPN proposed in [10].

The solvers have been run on five datasets. Three datasets Tsiligirides Set 1, Tsiligirides Set 2, and Tsiligirides Set 3 are provided by Tsiligirides from [4] and denoted as Set 1, Set 2, Set 3, respectively. The Tsiligirides Sets consist of overall 49 problems with budgets in the range from 5 to 110. The next two datasets, Chao diamond-shaped Set and Chao squared-shaped Set available from [23] were proposed by Chao *et. al* in [15] and are denoted as Set 64, Set 66, respectively. The Set 64 and Set 66 include 40 problems with budgets in the range from 5 to 130. Overall, the solvers have been tested on 89 problems for the OP. And for the OPN, all problems have been run with the varying sensing radius $\delta = \{0.5, 1.0, 1.5, 2.0\}$, and the sampled OPN solvers have been run with the different numbers of the possible waypoints $m = \{4, 6, 8, 10, 12\}$. Overall, the solvers for the OPN have been tested on 1 780 instances.

All the solvers have been implemented in C++, compiled using gcc version 5.4.0. The problems have been run on Intel Core i5-5200U CPU with 2.2 GHz. The solvers are randomized and have been run for 10-20 trials for each problem, the best reward R for given problem is obtained among the performed trials. Alongside of the best rewards R , the average computational time t and R_{avg} are reported. For the evaluation of the results, a relative percentage error (RPE), Eq. 7.1, and an average percentage error (ARPE), Eq. 7.2, have been used, where R_{ref} denotes the reference solution of the problem, which is the optimal solution for the OP and in the case OPN, it is the best solution found among all the existing solvers and the provided solutions.

$$\text{RPE} = 100 \cdot \frac{R_{\text{ref}} - R}{R_{\text{ref}}} [\%] \quad (7.1)$$

$$\text{ARPE} = 100 \cdot \frac{R_{\text{ref}} - R_{\text{avg}}}{R_{\text{ref}}} [\%] \quad (7.2)$$

The ARPE indicates the robustness of the algorithm. The $\overline{\text{RPE}}$ and $\overline{\text{ARPE}}$ denote the average values of RPE and ARPE for solved problems in the respective dataset, i.e. they are aggregated quality indicators among all budgets per particular problem set. The statistical t-test has been performed for selected datasets with the null hypothesis that the mean value of the solution of the problem obtained by the specific solver being the optimal (reference) solution of the problem. The results for individual evaluations are reported in following sections.

7.1 Results for the Orienteering Problem

The results obtained by the OP solvers HNN-OP, SOMv1, SOMv2, SOMv3, SOMv4, SOMv5, and SOM-VNS are evaluated in this section. The solvers have been initialized and run with

following settings.

HNN-OP have been run with the number of *iterations* set to 2, the number of *repetitions* set to 20 for 10 *trials*. The parameters of the solver have been set to $a = b = 1$, $c = 20$, $d = 10$, $g = 20$, $f = 15$, $\Delta t = 0.001$, and $\Delta \vartheta = 2$.

SOMvX denotes all versions of the SOMv1, SOMv2, SOMv3, SOMv4, and SOMv5. The solvers have been run for 500 *learning epochs* and 20 *trials*.

SOM-VNS consists of the SOM initialization and the VNS solver. The SOM have been run with number of *learning epochs* set to 500. The VNS solver have been run with the maximal computational time of the VNS $t_m = 5s$, the maximal number of iterations is set to 30 000, and the maximal number of iterations when the solution is not improved to 10 000. The number of *trials* is 10.

Table 7.1: Aggregated results for the Orienteering Problem.

Set	SOM2 [10]		HNN-OP		SOMv1		SOMv2		SOMv3		SOMv4		SOMv5		SOM-VNS	
	RPE	ARPE	RPE	ARPE	RPE	ARPE	RPE	ARPE	RPE	ARPE	RPE	ARPE	RPE	ARPE	RPE	ARPE
Set 1	0.10	1.05	1.81	5.57	0.10	1.32	0.10	1.27	0.55	2.92	0.57	2.78	0.24	1.17	0.10	0.40
Set 2	0.92	1.10	1.53	3.20	0.92	1.61	0.92	1.52	0.92	1.91	1.22	2.05	0.92	1.55	0.92	1.12
Set 3	0.00	0.89	0.61	2.09	0.00	1.00	0.00	1.17	1.22	3.49	1.36	3.60	0.00	1.19	0.00	0.28
Set 64	1.62	4.37	1.46	3.62	1.33	3.54	1.23	3.43	5.68	9.91	6.07	9.63	1.30	3.38	0.57	1.72
Set 66	1.53	5.27	4.89	7.89	2.46	5.61	3.04	5.77	3.67	6.98	3.34	6.82	2.93	5.79	0.42	1.92

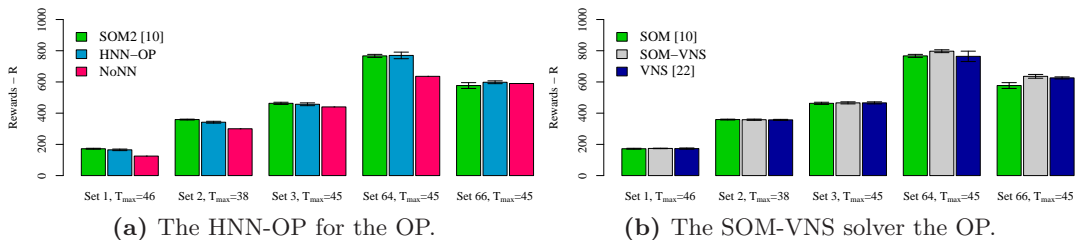


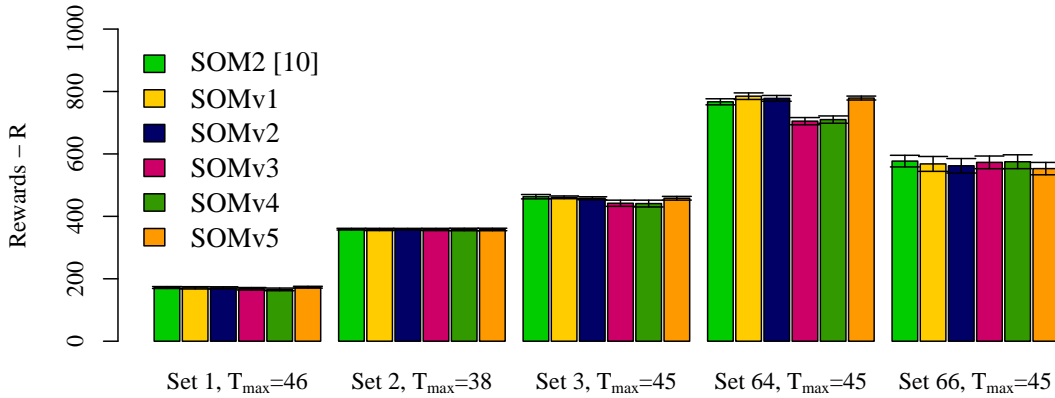
Figure 7.1: The average sum of rewards with the standard deviations as error bars for the HNN-OP [8], SOM2 [10], and the proposed SOM-VNS.

The HNN-OP implemented according to [8] is evaluated and the results depicted in Table 7.1 indicates that the HNN-based solver provides competitive solutions for the OP in the comparison of the SOM-based approach [10]. The HNN-OP has been compared with the NoNN solver based solely on the heuristic methods used in [8] to evaluate the influence of the neural network in the studied approach. The results reported in Table 7.2 and the overview presented in Fig. 7.1a indicate that the neural network based on the Hopfield-Tank model is the core part of the HNN-based solver to obtain the quality solutions because without it the NoNN provides poor solutions.

From the overview reported in Table 7.1 it can be noticed that the HNN-based solver provides overall better solutions for Set 64 than the half of the SOM-based solvers. The results in Table 7.1 also indicate that the modifications of the winner selection proposed to improve the quality of the solutions by the SOM-based approach [10] have all the similar

Table 7.2: Results for the problems of Chao diamond-shaped Set 64 for the Orienteering Problem.

T_{max}	MILP		GRASP [16]		SOM2 [10]		HNN-OP		NoNN	
	R	t [s]	R	t [s]	R	t [s]	R	t [s]	R	t [s]
15	96	1.86	96	0.01	96	0.01	96	2.36	96	0.25
20	294	1.15	294	0.05	294	0.03	294	3.11	156	0.93
25	390	6.90	390	0.06	384	0.03	390	4.53	252	2.40
30	474	1 147.50	468	0.06	462	0.04	468	6.53	348	4.24
35	576	3 111.59	576	0.08	570	0.05	576	9.11	492	7.77
40	714	33.94	714	0.09	696	0.05	696	15.72	564	12.00
45	816	67.98	816	0.09	804	0.06	798	21.01	636	16.81
50	900	81.63	900	0.11	870	0.06	882	27.55	702	27.74
55	984	49.44	984	0.17	960	0.07	972	36.34	810	38.38
60	1062	56.75	1062	0.12	1032	0.07	1032	50.52	912	48.59
65	1116	26.77	1116	0.11	1098	0.08	1110	65.23	1050	67.76
70	1188	119.94	1188	0.09	1164	0.08	1146	78.79	1098	85.27
75	1236	260.94	1236	0.09	1230	0.08	1206	104.12	1158	96.70
80	1284	338.63	1284	0.09	1272	0.08	1260	131.03	1212	115.67

**Figure 7.2:** Average sum of rewards with standard deviations as the errors bars for the selected problems for the Orienteering Problem.

performance in solution of Set 1, Set 2, and Set 3. The performance of the solvers shown in Fig. 7.2 for the most demanding problems of Set 64 and Set 66 varies. The solvers SOMv3 and SOMv4 performs poorly for these sets. On the other hand, the solvers SOMv1, SOMv2, and SOMv5 performs better than the SOM2 [10]. Overall, the modified SOM-based solvers SOMv1, SOMv2, SOMv3, SOMv4, and SOMv5 are competitive with the SOM2 [10] and the examples of the found solutions for the selected problem can be found in Appendix A.

The results reported in Table 7.1 imply that the best performing solver is the VNS solver with the SOM-based initialization. The SOM-based initialization of the VNS solver has been proposed to improve the found solutions for the OP and to determine whether the VNS solver provides better solutions with the random initialization or with the feasible quality SOM

Table 7.3: The evaluation of the t-test for the selected problems and the selected solvers.

Problem (T_{\max})	MILP	HNN-OP		SOMv1		SOMv2		SOM-VNS	
	R_{\max}	p-value	R	p-value	R	p-value	R	p-value	R
Set 1 (46)	175	2.025e-04	165	1.707e-04	171	3.758e-07	171	3.434e-01	176
Set 2 (38)	360	8.538e-06	342	8.281e-02	359	1.608e-01	359	1.679e-01	358
Set 3 (45)	470	1.896e-03	457	2.793e-08	461	2.350e-09	489	1.093e-01	466
Set 64 (45)	816	1.043e-04	770	7.011e-11	784	2.930e-13	778	1.498e-04	797
Set 66 (45)	650	3.195e-08	598	5.52e-12	568	8.800e-13	562	5.887e-03	636

solution as the initial solution. The results depicted in Fig. 7.1b imply the latter option of the initialization provides solution with the higher sums of the rewards.

The statistical one-sample t-test have been performed to evaluate the deviation of the selected problems from the optimal solution of MILP. Have the optimal solution of the selected problem with the budget T_{\max} denoted as R_{\max} , the set of rewards provided by the particular solver, and the statistical significance level $\alpha = 0.05$. The null hypothesis is following. The optimal value of the problem is the mean value of the set of rewards. The test can be interpreted according to obtained p-value, if the p-value is larger than α it can be considered that the mean value of the set is close to the optimal value and the null hypothesis is accepted, otherwise the obtained rewards are statistically significantly different, i.e., the mean value of the set is not close to the optimal value and the null hypothesis is not accepted. For example, the selected problem for Set 2 with $T_{\max} = 38$ the optimal value is $R = 360$, the p-value for the results provided by HNN-OP is 8.538e-06 and therefore the rewards are significantly different from the optimal value, and the null hypothesis is disregarded. On the other hand, the p-value of results provided by SOM-VNS is 0.1679 and therefore the null hypothesis is accepted and the rewards are close to the optima. The problems that accept the null hypothesis and therefore the solvers perform close to the optimal solution are presented in Table 7.3.

The overall results of the proposed solvers are evaluated and presented in Table 7.1. The proposed solvers are compared with the SOM based solver SOM2 [10]. The best results of the solvers are highlighted in bold. Overall, the best performing solver is the VNS solver with the SOM-based initialization and the examples of the found solutions provided by the solvers for the selected problems are presented Appendix A.

7.2 Results for the Orienteering Problem with Neighborhoods

The results obtained for the OPN by solvers HNN-OPN, SOMv1, SOMv2, SOMv3, SOMv4, SOMv5, and SOM-VNS are evaluated in this section. The solvers have been initialized and run with following settings.

HNN-OPN have been run with the number of *iterations* set to 2, the number of *repetitions* set to 10 for ten *trials*. The parameters of the solver have been set to $a = b = 1$, $c = 20$, $d = 10$, $g = 20$, $f = 15$, $\Delta t = 0.0001$, and $\Delta \vartheta = 2$. The sensing radius of the solver is $\delta = \{0.5, 1.0, 1.5, 2.0\}$ and the number of possible waypoints per each location is set to 10.

SOMvX denotes all versions of the SOMv1, SOMv2, SOMv3, SOMv4, and SOMv5. The solvers have been run for 500 *learning epochs* in 20 *trials*.

SOM-VNS consists of the SOM initialization and the VNS solver. The SOM have been run with the number of *learning epochs* set to 500. The VNS solver have been run with the maximal computational time of the VNS $t_m = 5s$, the maximal number of iterations set to 30 000, and the maximal number of iterations without solution improvement set to 10 000. The sensing radius of the solvers is $\delta = \{0.5, 1.0, 1.5, 2.0\}$ and the number of possible waypoints per each location is set to 10. The number of *trials* is set to 10.

Table 7.4: Results for the selected problems of the Orienteering Problem with Neighborhoods

Problem	HNN-OPN		SOMv1		SOMv2		SOMv3		SOMv4		SOMv5		SOM-VNS	
	R	t [s]	R	t [s]	R	t [s]	R	t [s]	R	t [s]	R	t [s]	R	t [s]
Set 1 ($\delta = 0.5, T_{\max} = 46$)	185	12.674	165	0.151	150	0.152	155	0.152	150	0.165	160	0.190	240	4.996
Set 1 ($\delta = 1, T_{\max} = 46$)	205	16.924	185	0.181	185	0.170	200	0.171	210	0.171	190	0.178	285	4.994
Set 1 ($\delta = 1.5, T_{\max} = 46$)	225	20.774	225	0.266	240	0.201	250	0.217	240	0.246	240	0.198	285	4.996
Set 1 ($\delta = 2, T_{\max} = 46$)	235	23.618	275	0.196	275	0.185	275	0.059	275	0.051	275	0.191	285	4.989
Set 2 ($\delta = 0.5, T_{\max} = 38$)	430	7.015	450	0.011	450	0.010	450	0.005	450	0.005	450	0.009	450	4.362
Set 2 ($\delta = 1, T_{\max} = 38$)	430	39.695	450	0.003	450	0.004	450	0.005	450	0.004	450	0.004	450	3.738
Set 2 ($\delta = 1.5, T_{\max} = 38$)	430	69.235	450	0.004	450	0.003	450	0.004	450	0.003	450	0.004	450	2.666
Set 2 ($\delta = 2, T_{\max} = 38$)	430	100.070	450	0.004	450	0.004	450	0.004	450	0.003	450	0.003	450	1.817
Set 3 ($\delta = 0.5, T_{\max} = 50$)	570	15.478	450	0.174	480	0.160	410	0.163	430	0.204	490	0.181	660	4.998
Set 3 ($\delta = 1, T_{\max} = 50$)	600	18.994	540	0.201	540	0.179	550	0.199	520	0.207	580	0.186	750	4.984
Set 3 ($\delta = 1.5, T_{\max} = 50$)	620	23.006	580	0.219	590	0.219	620	0.212	620	0.217	580	0.190	790	4.993
Set 3 ($\delta = 2, T_{\max} = 50$)	650	25.536	640	0.211	660	0.204	680	0.216	700	0.208	670	0.206	800	5.002
Set 64 ($\delta = 0.5, T_{\max} = 45$)	1020	200.904	1014	0.562	1068	0.583	900	0.569	900	0.562	1038	0.565	1326	5.010
Set 64 ($\delta = 1, T_{\max} = 45$)	1266	277.860	1308	0.639	1320	0.593	1344	0.360	1344	0.373	1308	0.601	1344	5.012
Set 64 ($\delta = 1.5, T_{\max} = 45$)	1260	346.660	1344	0.029	1344	0.031	1344	0.029	1344	0.039	1344	0.026	1344	5.013
Set 64 ($\delta = 2, T_{\max} = 45$)	1308	384.104	1344	0.025	1344	0.027	1344	0.025	1344	0.024	1344	0.025	1344	5.011
Set 66 ($\delta = 0.5, T_{\max} = 60$)	995	197.827	715	0.490	740	0.530	705	0.509	740	0.467	745	0.533	1420	5.011
Set 66 ($\delta = 1, T_{\max} = 60$)	1290	249.464	1415	0.581	1390	0.599	1415	0.615	1425	0.557	1400	0.547	1680	5.013
Set 66 ($\delta = 1.5, T_{\max} = 60$)	1385	287.172	1605	0.708	1605	0.797	1650	0.676	1650	0.681	1605	0.619	1680	5.017
Set 66 ($\delta = 2, T_{\max} = 60$)	1470	344.874	1680	0.016	1680	0.016	1680	0.019	1680	0.016	1680	0.016	1680	5.007

The performance overview of the proposed solvers for the OPN with $\delta = 1$ is reported in Fig 7.3.

The results depicted in Table 7.4 indicates that the HNN-based solver for the solution of the OPN is competitive with the SOM-based approaches, however the SOM-based approaches are capable of finding the optimal solution value of the problem with the $\delta > 0$ in tens of milliseconds, but for the problems with the lower budgets the HNN-OPN performs better than the SOM-based approaches SOMv3, SOMv4, SOMv5.

The SOM-based approaches SOMv1, SOMv2, SOMv3, SOMv4, SOMv5 modifying the winner selection are competitive with the existing approaches [21] and [10], see overview for the selected problems with $\delta > 0$ in Fig. 7.4. The flexible $\delta > 0$ gives the solver benefit to improve the quality of the solution and achieve the optimal values for the smaller budgets. The solvers provide almost optimal solutions for $\delta = 2$ and for $\delta = 1.5$, the obtained solutions are slightly worse than the SOM approach [21].

The results of the VNS solver with the SOM-based initialization for the selected problems with $\delta = \{0.5, 1.0, 1.5, 2.0\}$ are reported in Table 7.4. The SOM-VNS outperforms the other proposed solvers with the number of possible waypoint samples per each location set to 10. The computational time of the SOM-VNS is set to 5 seconds, the generation of the SOM initialization is in milliseconds, which is negligible and it is not included in the reported results.

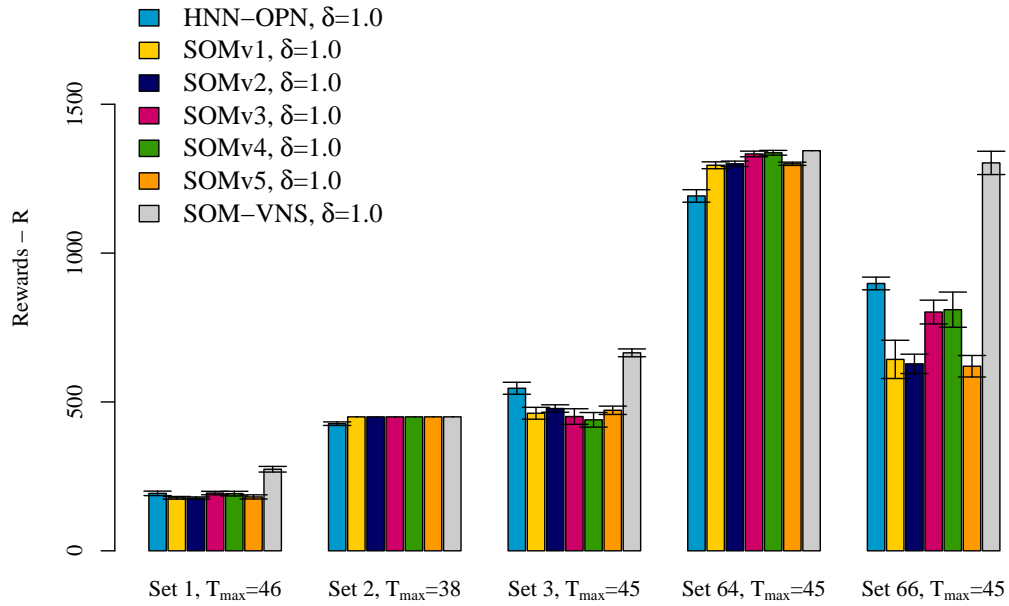


Figure 7.3: Average sum of rewards with standard deviations as the errors bars for the selected problems for the Orienteering Problem with Neighborhoods. The number of waypoints for the location of the samples solvers HNN-OPN and SOM-VNS is set to 10.

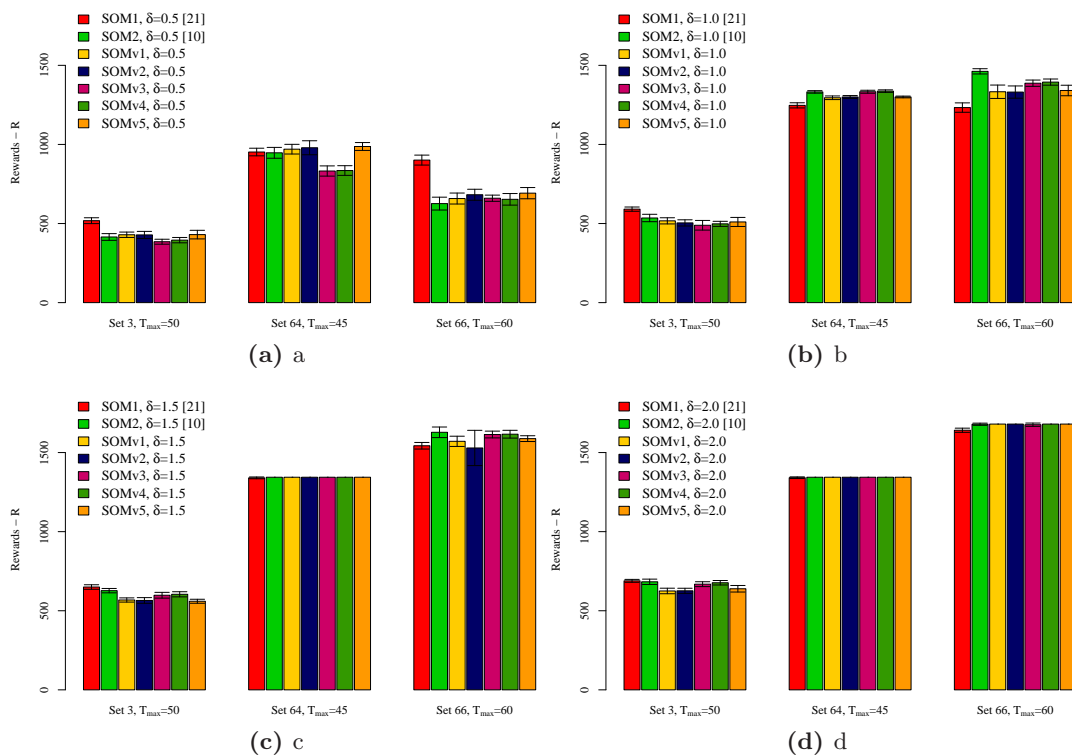


Figure 7.4: Average sum of rewards with standard deviations as the errors bars for the selected problems for the Orienteering Problem with Neighborhoods. The number of sampled waypoints per each sensing location in HNN-OPN and SOM-VNS is set to 10.

Chapter 7. Results

The overall results in Fig. 7.3 are indicating that the SOM-VNS solver is the best performing solver for the OPN regardless the sensing radius. The proposed HNN-based solver of the HNN for the OPN [8] is competitive but in comparison with the SOM solvers based on [10] the HNN-OPN is computationally more demanding. The examples of the found solutions provided by selected solvers for the selected problems are presented in Appendix A.

Conclusion

In this thesis, we focus on neural network (NN) approaches in the solution of the Orienteering Problem (OP) and their extension to the Orienteering Problem with Neighborhoods (OPN), specifically using the Hopfield Neural Network (HNN) proposed by Wang *et al.* in [8] and the Self-Organizing Map (SOM) originally proposed by Best *et al.* in [9]. The contribution of the thesis can be found in modifications of the approaches based on [8] and [10] that have been proposed to address the OPN and to improve the performance of the NN in the solving of the OP and the OPN.

The first proposed modification based on the HNN approach extends the OP solution to the OPN. First, the HNN-based solver for the OP has been implemented according to [8]. Based on the evaluation results, the HNN-based solver for the OP provides competitive results to the existing SOM-based approaches, and it is shown that the utilized heuristics in HNN cannot provide quality solutions without the Hopfield-Tank model.

The HNN-based solver has been further extended to fit the OPN. The challenge of applying the discrete Hopfield-Tank model to the continuous neighborhood has been overcome, and the HNN-based solver for the OPN provides competitive solutions for the OPN problems, however, it is computationally demanding.

The second modification is directly built on the SOM-based approach [10]. Even though the SOM-based solver [10] addresses the OPN, the solution can suffer from the sticking in the local optima, therefore new removal methods and the winner neuron determination has been proposed. The proposed selections are combined with the original methods of [10] to provide the most rewarding solutions. The reported results indicate that the OP addressed by the modified solver outperforms the SOM-based solver [10] for the problems of Set 64. The new proposed SOM-based solver provides competitive solutions for the OPN.

The last modification combines the SOM-based approach [10] and the Variable Neighborhood Search (VNS) [22] to improve the quality of the SOM-based solutions and determine whether the VNS solver performs better with the randomly generated initial solution or the SOM-based initial solution. The presented results indicate the latter option is better for the VNS performance. The VNS with the SOM-based initialization proves to be the best performing solver for the OP and the OPN; it outperforms both the HNN-based solver and SOM-based solver. The only drawback of the SOM-based VNS is the required computational time which is significantly higher than for the pure SOM-based approaches.

Overall, the proposed modifications of the NN approaches based on the HNN [8] and the SOM [10] are competitive. Amongst the proposed solvers, the worst performing is the HNN-based solver, however, the initial doubts about the quality of the solution and the time of the solution have been denied. The best performing solver is the VNS [22] with the SOM-based initialization [10].

For the future work, the determination of winner neurons and the selected neurons for removal of the SOM [10] can be further improved to determine neurons with even more significance to the conditional adaptation, and thus make a tighter connection between the principles of the unsupervised learning and the combinatorial heuristics. The HNN approach can be further studied to optimize the computational time for the OP and the OPN solutions.

Bibliography

- [1] B. L. Golden, L. Levy, and R. Vohra, “The orienteering problem,” *Naval research logistics*, vol. 34, no. 3, pp. 307–318, 1987.
- [2] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, “The orienteering problem: A survey,” *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.
- [3] A. Gunawan, H. C. Lau, and P. Vansteenwegen, “Orienteering problem: A survey of recent variants, solution approaches and applications,” *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.
- [4] T. Tsiligirides, “Heuristic methods applied to orienteering,” *Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, 1984.
- [5] B. Golden, A. Assad, and R. Dahl, “Analysis of a large scale vehicle routing problem with an inventory component,” *Large scale systems*, vol. 7, no. 2-3, pp. 181–190, 1984.
- [6] W. Souffriau, P. Vansteenwegen, G. V. Berghe, and D. Van Oudheusden, “A path re-linking approach for the team orienteering problem,” *Computers & operations research*, vol. 37, no. 11, pp. 1853–1859, 2010.
- [7] E. Álvarez Miranda, M. Luipersbeck, and M. Sinnl, “Gotta (efficiently) catch them all: Pokémon go meets orienteering problems,” *European Journal of Operational Research*, vol. 265, no. 2, pp. 779–794, 2017.
- [8] Q. Wang, X. Sun, B. L. Golden, and J. Jia, “Using artificial neural networks to solve the orienteering problem,” *Annals of Operations Research*, vol. 61, no. 1, pp. 111–120, 1995.
- [9] G. Best, J. Faigl, and R. Fitch, “Multi-robot path planning for budgeted active perception with self-organising maps,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3164–3171.
- [10] J. Faigl, “On self-organizing maps for orienteering problems,” in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2611–2620.

Bibliography

- [11] Z. Sevkli and F. E. Sevilgen, “Variable neighborhood search for the orienteering problem,” in *Computer and Information Sciences – ISCIS*, 2006, pp. 134–143.
- [12] A. Wren and A. Holliday, “Computer scheduling of vehicles from one or more depots to a number of delivery points,” *Operational Research Quarterly*, vol. 23, no. 3, pp. 333–344, 1972.
- [13] G. A. Croes, “A method for solving traveling-salesman problems,” *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.
- [14] R. Ramesh and K. M. Brown, “An efficient four-phase heuristic for the generalized orienteering problem,” *Computers & Operations Research*, vol. 18, no. 2, pp. 151–165, 1991.
- [15] I.-M. Chao, B. L. Golden, and E. A. Wasil, “A fast and effective heuristic for the orienteering problem,” *European journal of operational research*, vol. 88, no. 3, pp. 475–489, 1996.
- [16] V. Campos, R. Martí, J. Sánchez-Oro, and A. Duarte, “GRASP with path relinking for the orienteering problem,” *Journal of the Operational Research Society*, vol. 65, no. 12, pp. 1800–1813, 2014.
- [17] P. Hansen, N. Mladenović, and J. A. Moreno Pérez, “Variable neighbourhood search: methods and applications,” *Annals of Operations Research*, vol. 175, no. 1, pp. 367–407, 2010.
- [18] Q. Wang, X. Sun, and B. Golden, “Neural networks as optimizers: A success story,” *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 3, pp. 649–656, 1993.
- [19] J. Faigl and G. A. Hollinger, “Autonomous data collection using a self-organizing map,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1703–1715, 2018.
- [20] T. Kohonen, “The self-organizing map,” *Neurocomputing*, vol. 21, no. 1, pp. 1–6, 1998.
- [21] J. Faigl, R. Pěnička, and G. Best, “Self-organizing map-based solution for the orienteering problem with neighborhoods,” in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 001 315–001 321.
- [22] P. Váňa, J. Faigl, J. Sláma, and R. Pěnička, “Data collection planning with dubins airplane model and limited travel budget,” in *European Conference on Mobile Robots (ECMR)*, 2017, pp. 1–6.
- [23] “The orienteering problem: Test instances,” accessed: 2017-09-30. [Online]. Available: <https://www.mech.kuleuven.be/en/cib/op/#section-0>

Gallery of the Solutions of the OP and the OPN

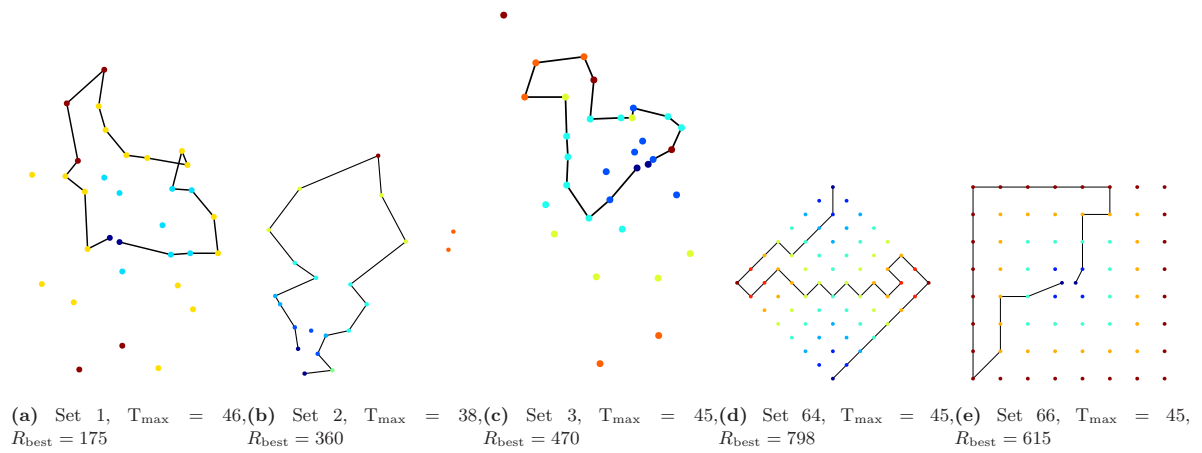


Figure A.1: The best solutions of the selected problems obtained by the HNN-OP.

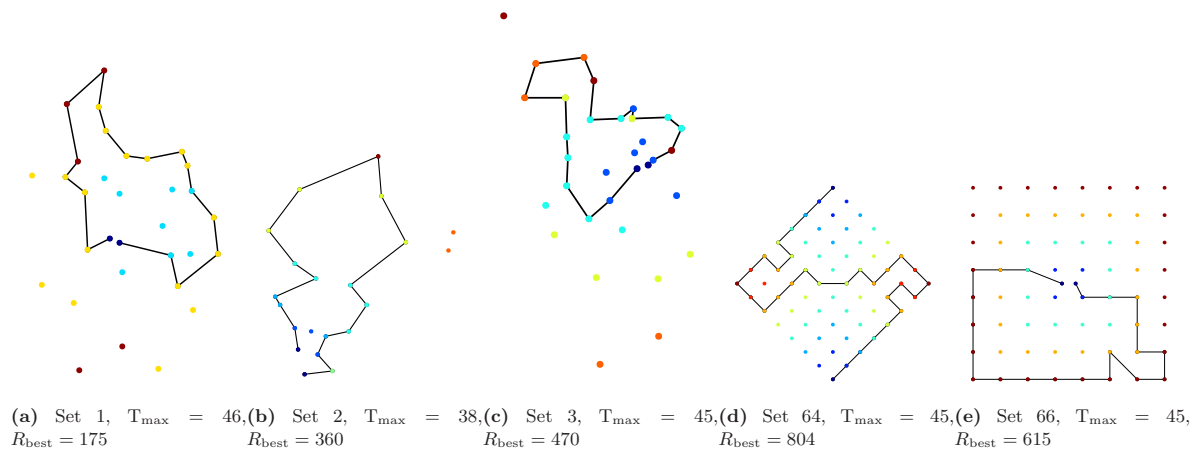


Figure A.2: The best solutions of the selected problems obtained by the SOMv1.

Appendix A. Gallery of the Solutions of the OP and the OPN

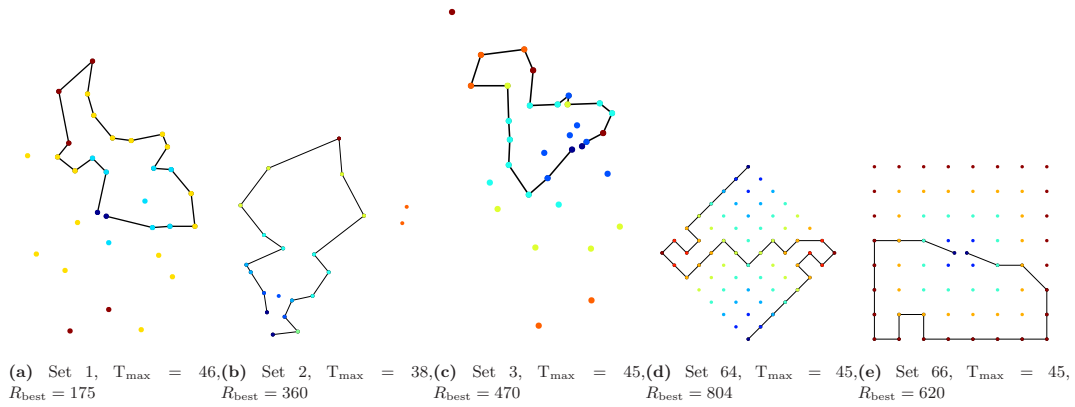


Figure A.3: The best solutions of the selected problems obtained by the SOMv2.

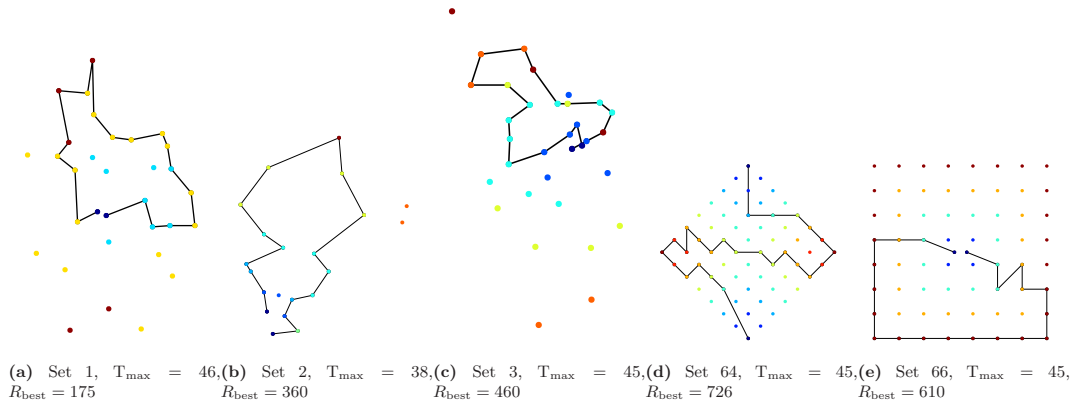


Figure A.4: The best solutions of the selected problems obtained by the SOMv3.

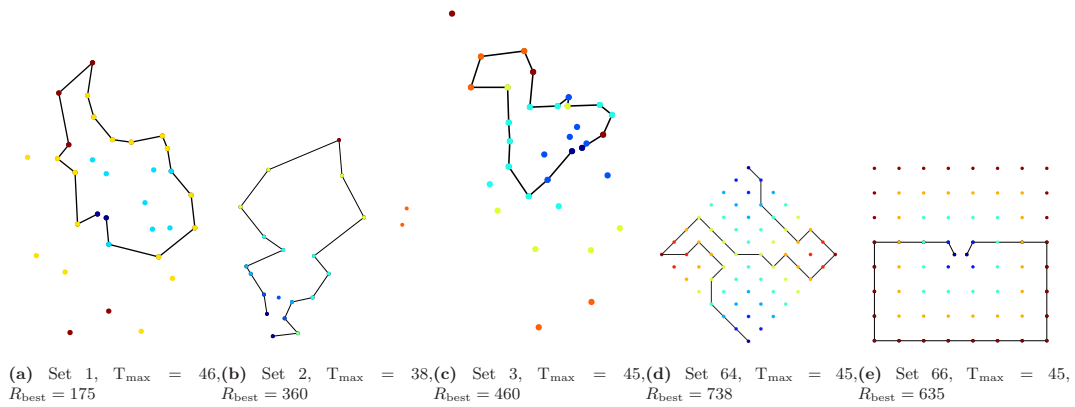


Figure A.5: The best solutions of the selected problems obtained by the SOMv4.

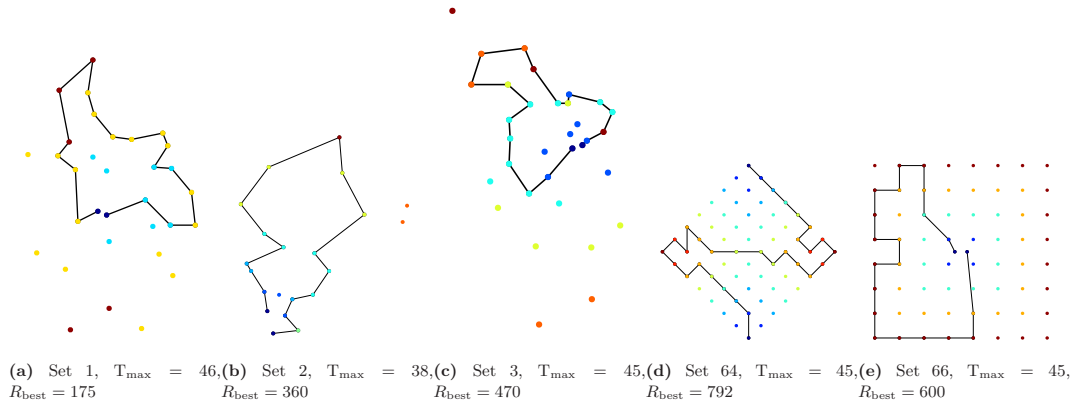


Figure A.6: The best solutions of the selected problems obtained by the SOMv5.

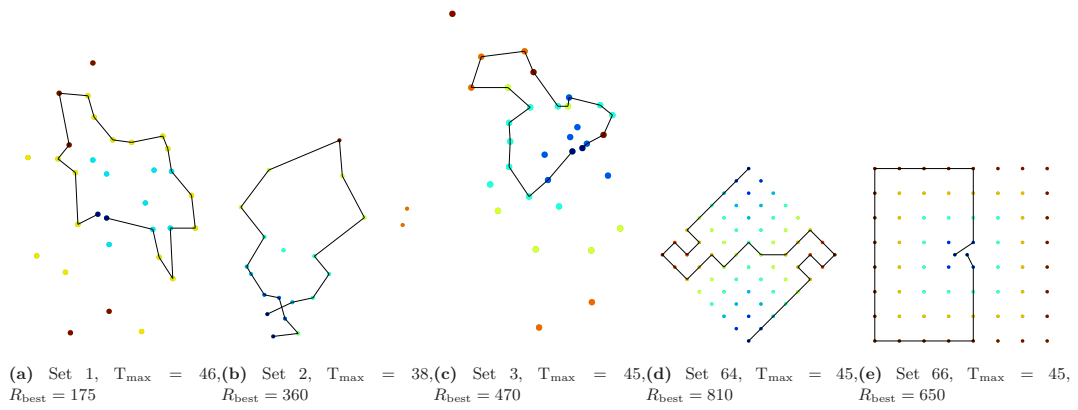


Figure A.7: The best solutions of the selected problems obtained by the SOM-VNS.

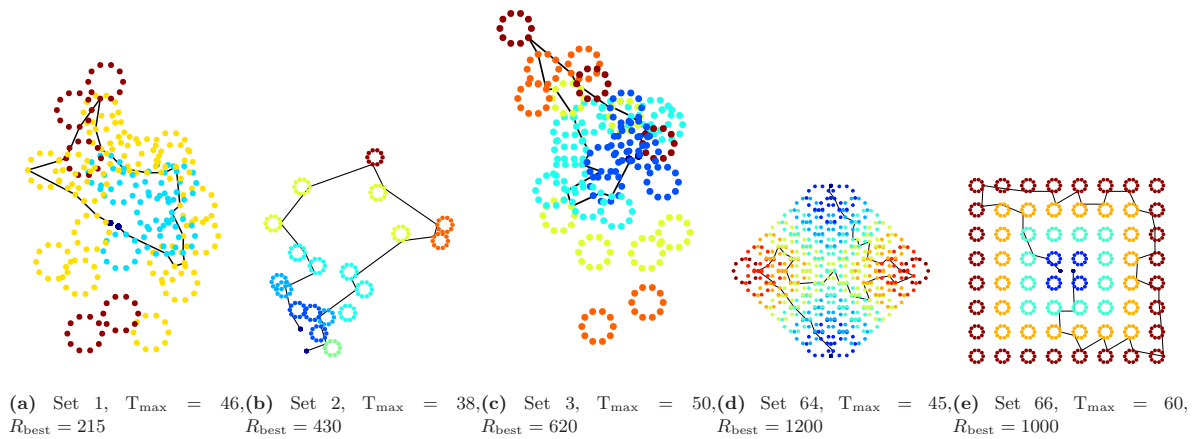


Figure A.8: The best solutions of the selected problems of the OPN obtained by the HNN-OPN.

Appendix A. Gallery of the Solutions of the OP and the OPN

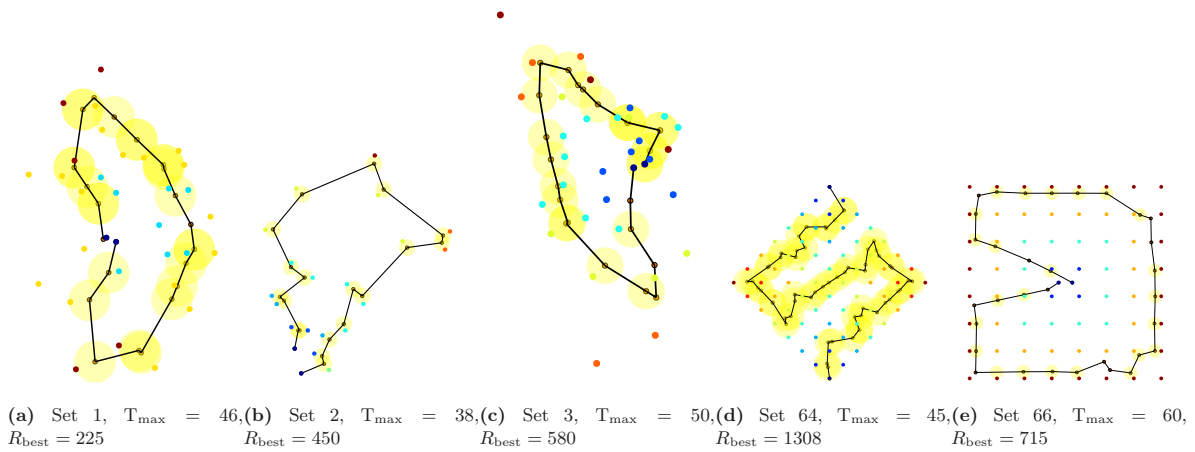


Figure A.9: The best solutions of the selected problems of the OPN obtained by the SOMv1.

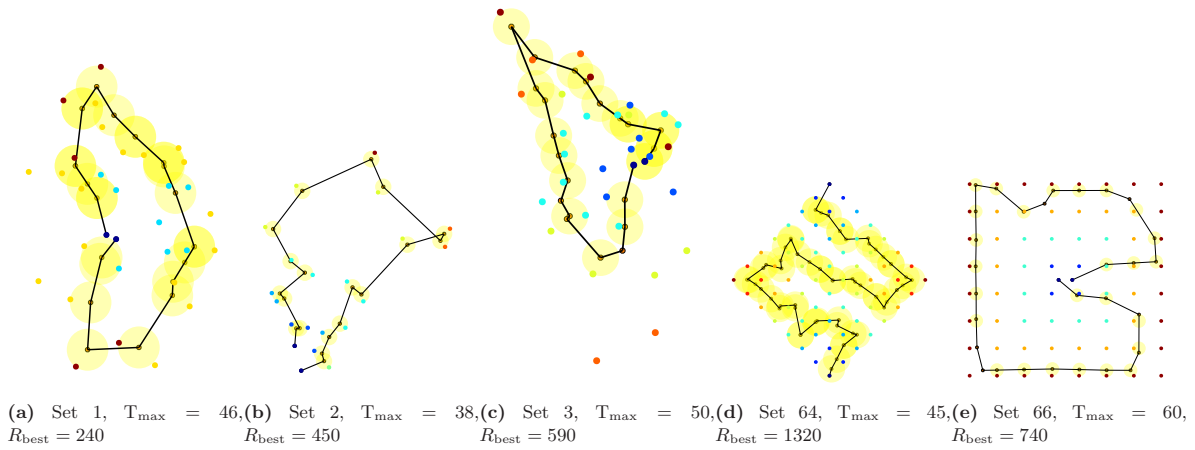


Figure A.10: The best solutions of the selected problems of the OPN obtained by the SOMv2.

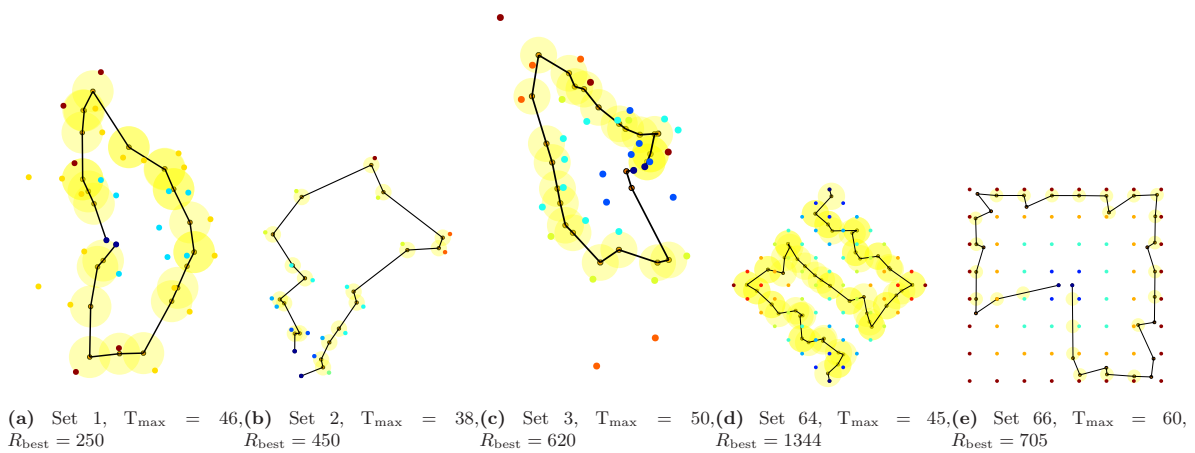


Figure A.11: The best solutions of the selected problems of the OPN obtained by the SOMv3.

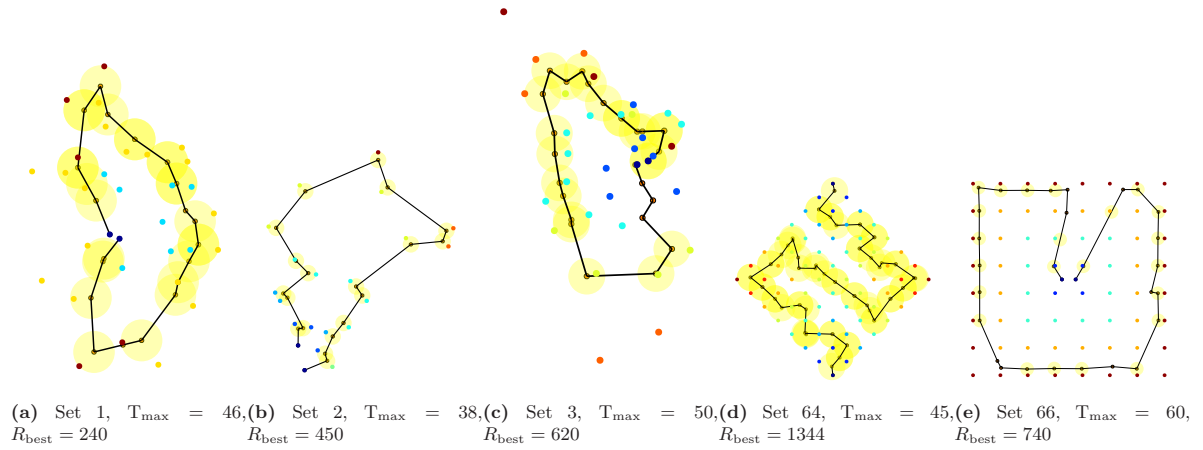


Figure A.12: The best solutions of the selected problems of the OPN obtained by the SOMv4.

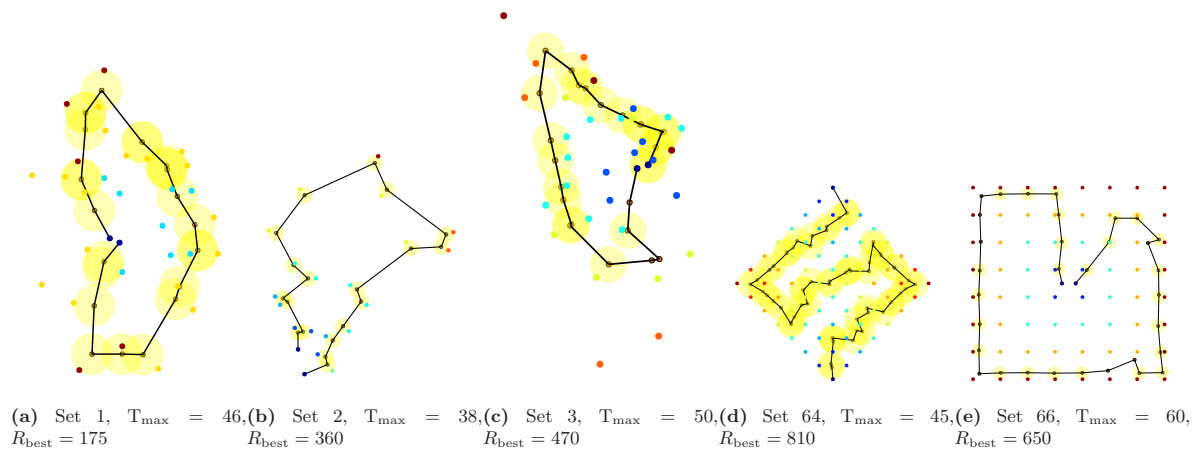


Figure A.13: The best solutions of the selected problems of the OPN obtained by the SOMv5.

Appendix A. Gallery of the Solutions of the OP and the OPN

Content of the enclosed CD

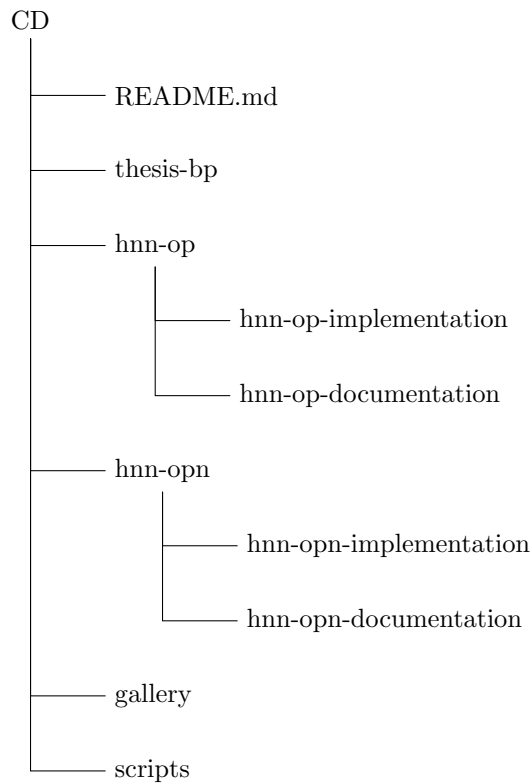


Figure B.1: The content of the enclosed CD.

Appendix B. Content of the enclosed CD