Doreen Körte

Cooperation of Swarms in Unknown Environments

FAKULTÄT FÜR
INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Cooperation of Swarms in Unknown Environments

Master Thesis

Doreen Körte

Thursday 24th September, 2020

Supervisor: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Advisor: Palina Bartashevich, M.Sc.

Doreen Körte: *Cooperation of Swarms in Unknown Environments*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2020

Abstract

This work analyzes the effect of neighborhoods on the performance of PSO swarms in unknown dynamic environments. In addition to commonly known topologies, new neighborhoods have been developed on the base of Game Theory. The goal is to improve the swarms fitness based on the selection of neighborhoods. Therefore, the effect of well chosen neighborhoods with fitness based generation in contrast to general neighborhoods is analyzed. Moreover, the effect of a limited communication radius which is more likely in real world scenarios is reviewed. For the analysis, two new PSO approaches have been developed being able to deal with unknown environmental influences. These influences are wind flows represented as vector fields. The new algorithms are called Zigzag PSO (Z-PSO) and Power PSO (P-PSO) pursuing different strategies. Z-PSO reduces the energy consumption by using the wind disturbance at its current position as a forecast of the wind in the next iteration. This forecast is used for adapting the particles velocity according to the wind orientation. In contrast, P-PSO does not predict any wind influence but always uses a minimal velocity vector in order to overcome possible wind influences.
This work shows that the difference between the topologies is not significant in most environments. Though, purposefully created topologies based on the particles fitness can outperform common topologies in strong vector fields. P-PSO is a promising approach for dynamic environments if the minimal velocity vector is adapted to the vector field. Besides, Z-PSO gains excellent results in all vector fields and is able to reduce the energy consumption. Analyzing the communication radius, Z-PSO reaches even better results using a smaller radius. Therefore, it is a good choice for real robotic applications which may not provide a fully connected swarm and have a limited energy supply.

Contents

List of Figures

List of Tables

List of Acronyms

2IPD	2-Player Iterated Prisoner's Dilemma
APSO	Asynchronous Particle Swarm Optimization
GBest	Global Best Topology
LBest	Local Best Topology
NIPD	N-Player Iterated Prisoner's Dilemma
P-PSO	Power Particle Swarm Optimization
PD	Prisoner's Dilemma
PSO	Particle Swarm Optimization
SR	Swarm Robotics
VF	Vector Field
VFM-PSO	Vector Field Map Particle Swarm Optimization
Z-PSO	Zigzag Particle Swarm Optimization

1. Introduction and Motivation

1.1. Motivation

During the last decade, swarm intelligence has become the focus of a high number of researchers. Swarm intelligence imitates the behaviour of natural social swarms like schools of fish or flocks of birds. A swarm consists of a number of individuals who can only perform simple tasks. However, the swarm as a whole is able to perform complex tasks in cooperation [7].
In the research field of Swarm Robotics (SR), strategies of swarm intelligence are applied to multirobot systems. SR offers various advantages over single-robot systems. This includes robustness in the first place. A swarm can self-organize and easily reorganize the task allocation because it consists of a number of simple and homogeneous agents. Furthermore, a swarm provides a high failure tolerance because the breakdown of one individual robot does not highly affect the task of the whole swarm. Additionally, it can be replaced effortlessly by a new robot. Another important factor is the decentralization which prevents one single failure point [28]. Moreover, a swarm of robots is time saving because it can work in parallel on multiple tasks and it is scalable to any number of robots [7].
Swarm intelligence techniques have as well been successfully applied to optimization problems. In 1995, Kennedy and Eberhart introduced a new optimization technique called Particle Swarm Optimization (PSO), which is inspired by flocks of birds. The PSO algorithm has also been used in SR to solve optimization problems. For example Jatmiko et al. used PSO for an odor source localization by mobile robots [14]. This approach could also be performed by robots in real-world scenarios. One scenario could be the detection of a carbon monoxide source. Another scenario could be the assistance of human fire fighters [25] or other real-world applications for mobile robot swarms [5].
Some real-world scenarios are solved with the help of aerial robots. However,

those robots bear a big disadvantage lying in their power supply. Compared to stationed robots, mobile robots are reliant on a battery and limited power resources. Therefore, energy consumption plays a major role. Furthermore, aerial robots have to deal with unknown external influences as, for example, wind flows. The robots need to be able to keep on performing their optimization task although they are disturbed by the environment. Thereby, the energy consumption may not be neglected, otherwise the swarm members could run out of energy.
To solve problems, the exchange of information between the single swarm members is of great importance. In recent literature, various topologies for PSO have been presented. Topologies are highly influencing the performance of a swarm by changing its diversity. In standard PSO, the swarm uses a fully connected network which allows all particles to communicate with each other [31]. However, results have shown that this topology does not always generate the best results [18]. Furthermore, this topology is based on the assumption that the swarm members are always close enough to each other guaranteeing exchange of data. Though, in real world scenarios the communication may be insufficient, for example, if the individuals are too far apart from each other.
This work shall give an insight about the impact of neighborhood selection on PSO approaches with a limited communication radius in dynamic environments. Therefore, various neighborhoods and pso approaches have been tested under different environmental situations and for varying optimization problems. The results can provide background information for further practical applications including real aerial robots.

1.2. Research Goals and Specific Objectives

The goal of this thesis is to investigate the influence of varying information exchange settings between members of a swarm performing PSO in unknown dynamic environments. Therefore, different topologies and communication radiuses are analyzed. In addition to some basic and well-known topologies, the set of tested topologies as well includes new topologies based on game theoretic approaches. For the communication radius, there are two extremes proved. One extreme is a communication radius as large as the search space, which implies a communication transfer between all swarm members. The other extreme is a small radius only allowing an inter-individual information exchange in small distances.

The unknown environment interpreted as wind flow is represented as a vector field. For further insights on the communication impact of swarms, two different swarms applying various PSO methods are tested and compared.

Goal
To analyze the impact of topologies and a limited communication radius on search mechanisms in unknown dynamic environments in robotic applications.

For accomplishing this goal, several objectives have been defined. The first objective is the implementation of a model for the general particles movement in vector fields. Thereby, the proximity to real robotic applications should be considered. Therefore, collision avoidance and continuous movement of the particles have been added to the model.

Objective 1
To implement a model for the general particles movement in dynamic environments.

The second objective observes further preparations for the later analysis. A main problem is the behaviour of PSO in unknown environments. The particles are highly disturbed by the wind flow and often not able to reach the optimum. However, an investigation of the neighborhood and communication radius is most interesting for swarms which are usually successful to see a difference. For this reason, two new PSO approaches have been designed, being able to cope with the vector field and reach the optimum. Besides, new topologies inspired by Game Theory have been created. Game Theory has been chosen because it studies the cooperative behaviours in social and biological systems. The idea is to analyze the difference between basic well-known topologies and new topologies which are modeled with the purpose of more intelligent and promising particle cooperation regarding the particles fitness.

Objective 2
To develop different variants of PSO for unknown dynamic environments and create new topologies based on game theoretic approaches.

In terms of applicability to real robotic systems, the energy consumption may not be neglected. Aerial mobile robots often only possess a battery as a power

supply. Consequently, the reduction of energy usage is an important goal for those robots. In this work, a simplified energy consumption model has been developed to provide first assumptions about the energy usage for different swarms and topologies.

> **Objective 3**
> To generate a simplified energy consumption model.

The basis of the analysis is the simulation of the PSO approaches in different settings. Test scenarios for the simulation have been developed. They include the application of varying neighborhoods and a limited communication radius. Moreover, different value functions and vector fields have been tested. The set of environments contains on the one hand vector fields calculated by functions and on the other hand real-world vector fields. These can be integrated by an interface for the input of longitude and latitude values requesting real-world wind data from NASA[1].

> **Objective 4**
> To implement test scenarios for simulating the performance of the new PSO variants with different topologies and a limited communication radius in varying environments.

The last objective deals with the analysis of the results generated by several simulations. This includes the evaluation of the two new PSO variants compared with each other as well as the comparison of each PSO variant with itself using different neighborhoods and a limited communication radius. On the one hand, the evaluation regards the fitness and convergence rate of the approaches. On the other hand, the energy consumption and orientation in relation to the vector field are reviewed.

> **Objective 5**
> To evaluate the PSO variants according to the used topology and compare the results.

[1] `https://www.nasa.gov/`

1.3. Structure Thesis

This section offers an overview over the structure of the thesis. First, Chapter 2 provides knowledge about the main concepts and basic topics of this thesis. Second, Chapter 3 summarizes related work to this thesis. Afterwards, the methods and concepts developed in this work are presented and explained in Chapter 4. Following, Chapter 5 gives an insight about the implementation and decisions in this regard. Afterwards, the results of the experiments executed in the scope of this work are analyzed in Chapter 6. At the end, Chapter 7 delivers a conclusion for the research goal and objectives of this work. Additionally, further elaboration possibilities for future work are mentioned.

2. Background

This chapter presents the knowledge about the underlying concepts of this thesis. In Section 2.1, the particle swarm algorithm is introduced and explained in detail. Furthermore, an asynchronous adaption of this approach is proposed and another variant including the application in vector fields. Afterwards, in Section 2.2 well-known topologies are presented generating the base for this work. Finally, in Section 2.3 game theoretic approaches are presented, which also inspired the methodical work in Chapter 4.

2.1. Particle Swarm Optimization

In 1995, Kennedy and Eberhart first introduced Particle Swarm Optimization (PSO). PSO is an approach for the optimization of nonlinear functions on the basis of biological structures. It imitates the behaviour of fish or bird swarms. Thereby, its implementation is straightforward due to its simple algorithm and the computation effort is low. A swarm consists of N individuals who are in contact with each other and share information. Together, they are able to develop a collective behaviour and create a self-organized swarm. This facilitates the search for optima in a search space. The ability to find the optimum is defined as the swarms fitness. In nature, this search would be, for example, led by the consideration of the food quality. In the work of Kennedy and Eberhart the search spaces were nonlinear functions. In the PSO algorithm, each individual i is given a position $\vec{x}_i(t+1)$ at time step $t+1$. The movement of each individual is calculated by the velocity vector $\vec{v}_i(t+1)$, which is described below. This velocity vector is added to the individuals position to calculate the new position in the following iteration $t+1$.

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + C_1\vec{\sigma_1}\left(\vec{P}_{best} - \vec{x}_i(t)\right) + C_2\vec{\sigma_2}\left(\vec{x}_g(t) - \vec{x}_i(t)\right) \quad (2.1)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (2.2)$$

The velocity vector $\vec{v}_i(t+1)$ consists of three components. The first one is the old velocity vector $\vec{v}_i(t)$ from the previous iteration. This vector is weighted by factor w. The second component is the cognitive or individual component, defined as $\vec{P}_{best}$. It represents the position of the best found solution of the individual. Thus, it is the personal optimum of this individual. In contrast, the last component is the social component $\vec{x}_g(t)$. This one describes the position of the best solution of the whole swarm. Therefore, it is also called global best or gbest component. The impact of each component in the search behaviour can be controlled by the acceleration coefficients C_1 and C_2. The vectors $\vec{\sigma}_1$ and $\vec{\sigma}_2$ define two random vectors in the range of $[0, 1]^n$, where n is the number of dimensions. The influence of the previous velocity vector is weighted by the inertia factor w, where $w > 0$ [17].

2.1.1. Asynchronous PSO

Asynchronous PSO (APSO) is an adaption of the standard PSO approach in relation to its update process. In general PSO, the swarm global best value or fitness is updated after the evaluation of all individuals. Consequently, the update process is done synchronously. However, this strategy is not useful for swarm robotics because the individuals can constantly move and gather information without awaiting the whole swarm to update its fitness. Therefore, APSO has been developed. In contrast, the update process is performed asynchronously. That means each individual updates its position and fitness value immediately after the calculation. As well, the global fitness of the whole swarm is updated instantly. Then, the individual's search information is partial and imperfect, resulting in diversity inside the swarm [1].

2.1.2. Vector Field Map PSO

The Vector Field Map PSO (VFM-PSO) approach is a variant of the basic PSO algorithm described before. It was designed for aerial microrobots working in unknown dynamic environments. These environments may, for example, be uncharted wind flows which are influencing the robots flight and search performance. The main ideas behind this approach are two cooperating swarms performing different tasks. One swarm is called optimizer swarm and tries to investigate the search space and find the optimum. In contrast, the other swarm is named explorer population. Its task is to gather information about the environment and provide it to the optimizer swarm. The explorer swarm creates a map illustrating, for example, wind flows in direction and intensity. The optimizer swarm uses this map by adapting its movement to overcome the disturbance of the environment. The environmental influence is studied in a two-dimensional search space defined as a Cartesian Grid. It is implemented as a planar Vector Field (VF). By interpolation, a vector $\overrightarrow{VF}(x_1, x_2)$ for each possible point (x_1, x_2) in the search space can be calculated using Equation 2.3.

$$\begin{aligned} \overrightarrow{VF}(x_1, x_2) \quad = \quad & (1-u_2)(1-u_1)\overrightarrow{VF}_{0,0} + (1-u_2)u_1\overrightarrow{VF}_{1,0} + \\ & u_2(1-u_1)\overrightarrow{VF}_{0,1} + u_1u_2\overrightarrow{VF}_{1,1} \end{aligned} \tag{2.3}$$

The explorer swarm collects information about the vector field at every position it passes. The resulting information map is used by the optimizer swarm to resist the vector field influence. Its movement is described by the following equation, where $\vec{x}_i(t)$ denotes an individuals current position. $\vec{v}_i(t+1)$ expresses the individuals velocity and $\sum_{k=0}^{K} \overrightarrow{VF}(p^k)$ describes the interpolated environment vector [6].

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) + \sum_{k=0}^{K} \overrightarrow{VF}(p^k) \tag{2.4}$$

2.2. Topologies

In PSO, a topology determines which particles are connected with each other to exchange information. For standard PSO, this network structure defines that all particles can communicate with all other members of the swarm. This is a so-called fully connected topology. Since the possibility of information exchange highly influences the performance of PSO, the neighborhood selection affects the search results as stated by Toscano-Pulido et al. [30]. In the past years, a wide range of neighborhoods for PSO has been introduced. This section offers an overview over the most common and widespread topologies including their preferences.

2.2.1. GBest

The most popular topology for PSO is the Global Best (GBest) topology. It is well known since it is used by standard PSO. The structure is fully connected, which means that each particle is linked to all other members of the swarm. Consequently, during the updating process of PSO, a particle can determine the global best value by communicating with the whole swarm. That is why it is called Global Best because the particle always gets the information of the whole swarm. The information is passed immediately inside the swarm so that it quickly finds an optimum. However, the fast convergence speed has the disadvantage that the chance of stagnation in local optima is high. If all particles share the same global best, they are all moving towards the same direction. As a result, the search diversity decreases and a local optimum as the final result is more probable [30].

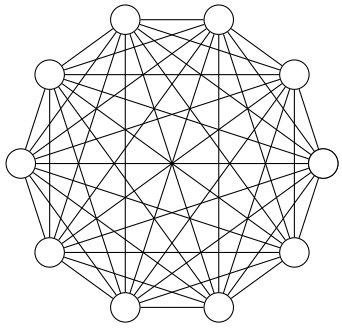


Figure 2.1.: GBest Topology

2.2.2. Ring

The Ring topology or also known as Local Best (LBest) is another common topology in PSO literature. In contrast to GBest, this network highly decreases the number of connections within the swarm. The structure arranges the nodes which are representing the particles in the shape of a circle. Afterwards, each node is connected to its left and right neighbors. Finally, the edges look like a ring giving the topology its name. The topology is also called Local Best because, compared to GBest, the particles can update their global best value only locally by their direct neighbors instead of the entire swarm. Following from this, the information transfer is a lot slower and the convergence may take a longer time. The particles need more iterations to pass the information and therefore follow the global best of the swarm. Due to the communication delay, the search diversity increases and the chance for finding the global optimum [30].

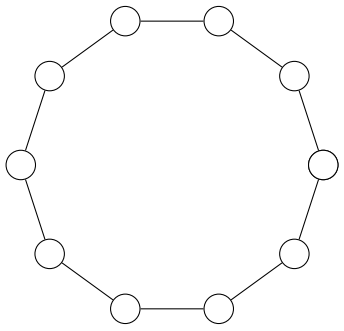


Figure 2.2.: LBest Topology

2.2.3. Von Neumann

The Von Neumann topology was first introduced in cellular automata by John Von Neumann. In 2002, Kennedy and Mendes first introduced the topology into PSO. The network structure is designed as a rectangular matrix in which each node is connected to its neighbors on the left side, on the right side, below and above itself. Transferring this structure to the nodes arranged in the circle shape, each node is no longer connected to its direct neighbors but to those after them [34]. Kennedy and Mendes tested different topologies and their effects on the performance of PSO. In their summary, they recommended the Von Neumann network because it performed more consistently than the other topologies [18].

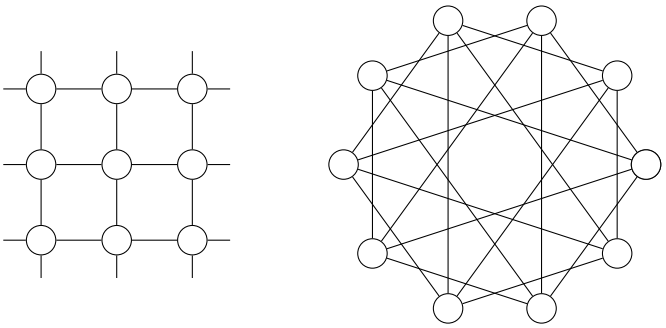


Figure 2.3.: Von Neumann Topology

2.2.4. Star

The Star topology or also known as Wheel is highly isolating the particles of each other. The structure is designed as a Star where the name comes from. All nodes are connected to one single node. This one node owns as many edges as other particles exist, but all other particles have only one connection. This means that all information needs to be passed to this one particle to be continued sending to the other particles [16]. There exist different variations of this network structure. The central particle which controls the information flow can either be the same over the whole process or it can be selected randomly every iteration as described in [30]. The reduced information passing through the central node acts as a buffering and therefore reduces the chance of stagnation at local minima.

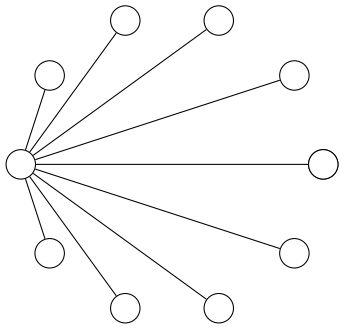


Figure 2.4.: Star Topology

2.3. Game Theory

Game Theory is about decision-making and understanding the occurrences of two decision-makers collaborate. The focus is driven towards the revealing of economic, political, and biological phenomena. Some exemplary applications are the following:

- job applicants competing for a job offer
- political candidates competing for votes
- jury members choosing a contest winner
- animals striving for loot
- competitors making bids in an auction

Game Theory includes various models which try to abstract and simplify real-world scenarios. Whereby, they take into account all important facts and leave out for the decision process unnecessary details. The goal of understanding Game Theory and human interactions may open new possibilities to change the behaviors and improve our own well-being. A wide range of models cover the theory of rational choice. The theory describes that a decision-makers choice among all possible actions will be the best action corresponding to his preferences. This indicates that the decision is depending on the set of possible actions and not on his likes and dislikes. For each pair of actions, the decision-maker knows which one he prefers or he wants both equally likely. To determine the preferences, a payoff function is used. This function maps a numerical ratio to each action. In the decision process, actions with higher numbers are preferable.
One game theoretic model is the strategic game. It models the interaction of multiple decision-makers defined as players. For each player, there are possible actions including preferences over these actions. An important difference is that an action has influence on all players and not only the one who is executing this action [24].

2.3.1. Prisoner's Dilemma

The Prisoner's Dilemma (PD) is a well-known strategic game derived from a situation in which two persons are suspected of a crime. They are both interrogated in different cells, so none of them knows what the other one will tell. The police does not have enough evidence to censure one of them for the major crime, but they could convict both of them of a minor crime. In this case, both will be arrested for one year. Another possibility is that one of them whistles blows the other one. In this case, the whistle-blower will not go to jail and the other one will stay three years in prison. In the case that both try to cheat on each other, they will both spend two years in prison [24]. The game is illustrated by a game matrix in Figure 2.1.

Table 2.1.: Prisoner's Dilemma

		Suspect B	
		Quiet	Cheat
Suspect A	Quiet	$1, 1$	$3, 0$
	Cheat	$0, 3$	$2, 2$

The interesting point is that they will get the highest payoff if they both cooperate. However, they do not know whether the other is cooperating or not. If one of them cooperates but the other one defects, the cooperating partner will lose and the other one will get a payoff [9].
The 2-player Iterated Prisoner's Dilemma (2IPD) is an extension of the PD with the characteristic that each step is replayed multiple times and the players hold a memory about the past decisions. One problem of the 2IPD is the fact that it is not applicable to most of the real-world problems like economics or social questions. Therefore, the N-Player Iterated Prisoner's Dilemma (NIPD) has been developed, which is able to cope better with real-world scenarios. NIPD works the same as the 2IPD but uses n players instead of two [33].

2.4. Bound Handling

Bound handling takes an import role in the simulation of swarm behaviour. Often the search space is limited and an appropriate bound handling technique is needed. In [13], different mechanisms are presented and compared with each other in the context of their performance. One of them is the Periodic Approach which is also applied in this work. The search space is extended by an infinite number of copies. For evaluation, the individual is mapped to the original search space with the help of a modulo operation. The individuals characteristics remain unchanged in this method. The resulting behaviour for a two-dimensional search space is illustrated in Figure 2.5.

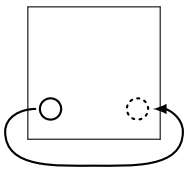


Figure 2.5.: Periodic Bound Handling

3. State of the Art

This chapter provides an overview of the literature dealing with topology in PSO. The chapter is structured as followed. First, static topologies and their different network structures are presented. Second, a review of dynamic topologies is given. As against static topologies, these imply a variation over time because they are, for example, based on probabilistic parameters. Furthermore, neighborhoods are introduced depending on the particles fitness. Subsequent, PSO approaches based on game theoretic approaches are introduced. Finally, approaches designed for asynchronous PSO are illustrated.

3.1. Static Topologies for PSO

The static topologies for PSO are predefined and fixed topologies. This means that they are not changing. Three different types of networks exist. The first type is the regular network like ring and star. These topologies are generated according to fixed rules and without any probabilistic parameter. The second type is a completely random network which is based only on a probabilistic framework and has no fixed properties. The third type is a network in between the two mentioned before. This means that it has regular features but is also influenced by probabilistic parameters. This section gives an overview of the research on static topologies [11].

Disregarding the basic topologies described in Section 2.2, researchers have revealed a lot of other regular network structures and integrated them into the process of particle swarm optimization.

In 2014, Li and Guo developed the Regular Network PSO (RN-PSO). In RN-PSO, a particle communicates with a number R of predefined other particles for updating the local best position in PSO. If R is equal to one,

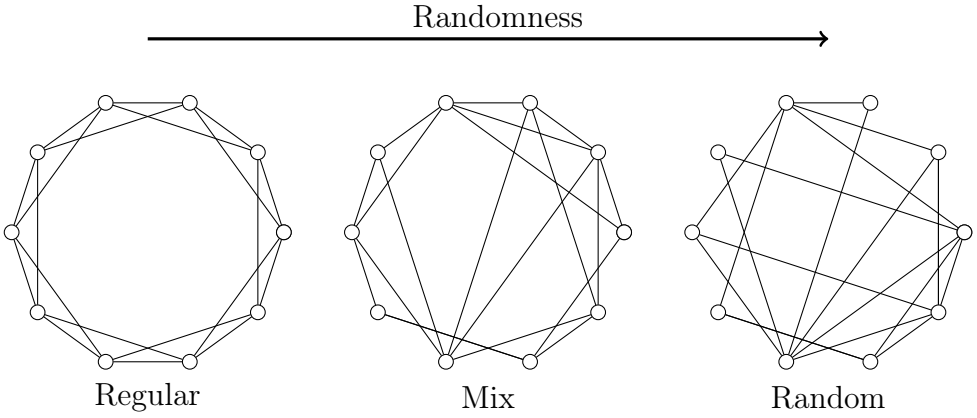


Figure 3.1.: Types of Networks

the structure is equivalent to the LBest topology, whereas by increasing R, it converges to the GBest topology [19].

Mendes et al. introduced other network structures. One of them is the four clusters topology. Thereby, particles are grouped into clusters. Inside a cluster, all particles are connected with each other. Additionally, the clusters are connected with each other over a gateway between single particles of each cluster. Thereby, the information exchange within a cluster is fast, but the transfer between clusters shows a delay. Another network structure presented by Mendes et al. is the pyramid. In this topology, the particles are arranged as a three-dimensional pyramid [22].

Moreover, Matsushita and Nishio developed the Network-Structured Particle Swarm Optimizer (NS-PSO), which uses a Small-World Model for the neighborhood determination. In this model, the particles are first connected according to the Ring lattice. Second, the connection of each particle is randomly rewired to another [20].

3.2. Dynamic Topologies for PSO

Apart from static topologies, a lot of authors have investigated dynamic topologies. Static topologies are fixed and not changing over time, whereas dynamic topologies are generated by varying parameters whereby the

neighborhoods are permanently redesigned. A dynamic topology may, for example, be based on the particles distances. A particle is connected to another particle if their distance is smaller than a predefined threshold. Consequently, the connections are changing over time because the particles are moving and the distance to other particles is unstable. Apart from this approach, there exist a lot more so that this section shall provide an overview over probabilistic fitness and selection mechanism based approaches.

3.2.1. Probability

The further concepts are based on probabilistic models like the Independent-Minded Paricle Swarm Optimization by Matsushita et al.. The particle global best position for the optimum is always influenced by its neighbors and their best found solutions. Since a fully connected swarm has a higher chance of stagnation at a local optimum, Matsushita et al. created independent particles. In each iteration, a random factor determines whether a particle is connected to the swarm or not. These independent particles are totally isolated from the swarm and cannot exchange information. As a result, they do not stop searching for better values even if the swarm might stagnate at a local minimum [21].

Ni et al. declared three different topologies in their work. One of them is the Fixed Cycle Based Dynamic Population which creates randomly a new topology periodically after a number of iterations. The second one is the Optimum Updating Status Based Dynamic Population Topology which tries to prevent stagnation at local minima by changing the topology. If the swarm's optimum has not changed after some generations, the degree of creating a new random topology increases. The last network is a combination of both previously mentioned. Resulting, a new random population topology is generated after the number of iterations or randomly as the factor is growing by stagnation [23].

Additionally, Akat and Gazi also presented a probabilistic neighborhood. In this neighborhood, each particle is the neighbor of another particle by a chance of 50 percent. Thereby, they stated two variants of his approach. In the first variant, the connection can be reciprocal so that regarding a

connection, the particles are neighbors of each other. In the second variant, only one particle is the neighbor of the other one but not vice versa [2].

An approach presented by Saxena and Vora is based on Small World Theory. It represents a combination of the Small Network and the Von Neumann Structure. Each particle can communicate with its four neighbors, but additionally it can interact with random particles. In contrast to the Von Neumann Structure, the particle can only exchange information with these selected particles and not particle across [29].

In addition, another structure based on the Small World Theory is the Adaptive Small World Particle Swarm Optimization developed by Gong and Zhang. In contrast to the Small World Particle Swarm Optimization algorithm, the neighborhood size and disorder probability depend on the convergence state of the swarm. First, a particle is connected to K successor particles. Second, the particle is instead connected to other random particles with a probability p [11].

Besides, new PSO algorithms have been created based on the Barabasi-Albert Model. One of them is the Scale-Free Network Particle Swarm Optimization, which is initialized with a small number of fully connected nodes and adds new nodes in each iteration. However, the new node is not necessarily connected to all other nodes, but a probabilistic parameter decides on the connection.

Junior et al. also proposed a structure based on the Barabasi-Albert Model. The basic idea is that the swarm changes its topology if it stagnates and no longer improves its solution [15].

3.2.2. Fitness-based

Since the particles shall improve their fitness to find the optimum, some literature deal with the topology generation based on the particles fitness. Akat and Gazi developed a PSO algorithm with a structure based on the nearest neighbors in function space. Whether two particles are connected depends on the difference of their function values. If it is smaller than a predefined threshold, the particles can interact with each other [2].

A hybrid approach has been presented by Hamdan using a combination of Star, Ring, and Von Neumann topology. For each of these three topologies, the resulting fitness value is calculated and afterwards the one with the smallest value is selected [12].

Besides, Wang and Xiang used a dynamic ring for the structure determination. The idea is that particles are ordered in a ring structure but their position depends on their personal fitness. According to their position in the ring, particles can only communicate with better or worse particles [32].

Junior et al. used a roulette wheel rank for the particle selection. It regulates that particles with a higher fitness are more likely to be chosen as neighbors to improve the optimization results [15].

3.3. Game Theory

A wide range of approaches apply methods of the research field of Game Theory to Particle Swarm Optimization. The strategic decision making of game theoretic approaches can facilitate the search process in PSO.

One approach is the Predicted-Velocity Particle Swarm Optimization using a game theoretic approach created by Cui et al.. In the sense of Game Theory, particles are treated as players in an artificial evolutionary game playing mixed strategies. The players strategies are represented by position or velocity vectors. Each strategy is selected by a specific probability which is set to 0.5 at the beginning. After each move, the probability for the selected strategy increases or decreases depending on the improvement of the particles fitness [8].

Another game theoretic approach has been presented by Di Chio et al. in 2008. The work is based on the well-known Prisoner's Dilemma (PD) game in which two players must choose between cooperation and defection. Di Chio et al. tried to apply this game scenario of cooperation and defection to Particle Swarm Optimization. For that matter, they have created two meanings of the two strategies. In the first version, cooperation means that the social component of a particle has a higher influence in the calculation. Whereby,

defection results in a higher impact on the individual component. The second version distinguishes cooperation and defection in such a way that cooperation means behaving as a normal PSO particle. Whereas, defection results in four different behaviours. One of them lets the particle only move towards its own best position. The second variant creates a movement in the opposite direction of the best position of the whole swarm. The third variant is similar to the second one but regards the whole force, so the individual best is also taken into account. At least, the fourth variant drives the particle towards a random position. Besides, different mechanisms have been presented for changing the strategy. One of them is called Flip-Strategy. If the particles fitness has not improved since the last iteration, it changes its strategy to the opposite one. Otherwise, it keeps its strategy. The other variant is that a particle always cooperates if the fitness improves or it always defects if the fitness decreases [9].

In addition to the Prisoner's Dilemma, many researchers have investigated the n-player Iterated Prisoner's Dilemma (NIPD). Since the cooperation in large populations is challenging, different communication strategies have been studied. Almanasra has worked on the problem of premature convergence and as a result a bad cooperative behaviour among the players. Therefore, he has developed an evolutionary model which is based on PSO. The PSO particle communication network is based on sub-swarms using a Von Neumann topology for communication inside the sub-swarm and towards the best particles of neighboring swarms. Additionally, each particle has a knowledge base in which the player's moves and strategies of the sub-swarm and the whole swarm are stored. Each group of players is represented by a sub-swarm of particles. Whereby, each sub-swarm tries to find the best particle inside its neighborhood. For diversity and information exchange between the NIPD participants, the sub-swarms are connected with each other. PSO is used to increase the cooperation ratio and create ambitious strategies [3].

3.4. Asynchronous PSO

The particles in Synchronous Particle Swarm Optimization have perfect information about their neighbors before they update. This means that all particles are evaluated and afterwards updated. In contrast, in Asynchronous Particle Swarm Optimization, the particles have imperfect information about

their neighbors. This is due to the fact that the particles are evaluated and updated asynchronously and therefore directly change their position after evaluation [27]. Rada-Vilela et al. stated that synchronous PSO fits best for small neighborhoods due to its slow convergence. Asynchronous PSO reaches a faster convergence. However, the bottle neck of this approach is the higher change of premature convergence. Random Asynchronous Particle Swarm Optimization shows the fastest convergence and therefore is most applicable to large neighborhoods. Random Asynchronous PSO is a variant in which the particles randomly update and evaluate. As a result, some particles do not update or change at all in some iterations [26].

Fernandes et al. proposed an asynchronous steady-state PSO inspired by co-evolution. The idea is that in each iteration only the worst particles of a neighborhood are updated and the other particles remain steady until they fulfill the update criterion [10].

Furthermore, a Switch Particle Swarm Optimization algorithm has been developed by Aziz et al.. In this approach, the fitness of the best particle is observed. If the fitness keeps unchanged for a number of iterations, the update strategy is changed. If the previous update strategy was synchronous, it becomes asynchronous and vice versa. This approach showed better results than Synchronous and Asynchronous PSO [4].

4. Methodology

This chapter describes the methodology for the experiments presented in Chapter 6. This work shall give an insight on the influence of neighborhoods on PSO swarms in unknown dynamic environments. Therefore, the implementation of the environmental impact has been used from the research about VFM-PSO in Section 2.1.2. Transferring the simulation to real word scenarios, the environment illustrated by a Vector Field (VF) shall represent the wind influence affecting the swarms performance. The work of Bartashevich et al. has shown that a normal PSO swarm is not able to deal with the wind influence and gain a good fitness [6]. Fitness describes the ability of a swarm to find the optimum. Due to this fact, two new PSO approaches for swarms in dynamic environments have been developed in Section 4.3, which are able to cope with the vector fields disturbance. Since the effect of neighborhoods shall be regarded, Section 4.4 presents further newly developed neighborhoods which will extend the already known basic topologies from Section 2.2. Moreover, an energy model is introduced in Section 4.2 for illustrating the energy consumption of the swarms. In the first Section, general assumptions and constraints are explained which found the base for the simulation. This includes especially the individuals collision avoidance (Section 4.1.1) and laser usage (Section 4.1.2).

4.1. Assumptions and Constraints

This section specifies and explains the assumptions and constraints stated of the following models and simulations. A well-arranged list can be found below. First, the Search Space underlies some preliminaries. This includes the assumption that it is two-dimensional. Additionally, the wind velocity is limited to a defined threshold. Besides, the wind is constantly affecting the individuals. This means an individual is always pushed in one or the other direction due to the wind. Therefore the wind is always affecting the

individual, whether it is moving or not. Another important part is the bound handling. As already explained in Section 2.4, the periodic bound handling technique is applied.

- **Search Space**
  - is two-dimensional
  - wind velocity is limited
  - wind is constantly affecting individuals
  - periodic bound handling

Second, the individuals characteristics need to be defined. In the initialization phase, the individuals are distributed randomly across the whole search space. For the PSO calculations, they are using the asynchronous PSO approach described in Section 2.1.1. Since the experiments shall derive information about the energy consumption of the individuals, it is based on the individuals movement and the wind. The concrete calculation is revealed in Section 4.2. Furthermore, the individuals communication radius is limited to a predefined value to analyze its impact on the search results and the neighborhood formation. To make the conditions as similar as possible to real-world scenarios, the individuals shall not collide as it is common for real robots. The whole collision avoidance calculation is explained in Section 4.1.1. As the collision avoidance may downgrade the search results as it is hindering the search process, the individuals are equipped with a laser (4.1.2). The laser enables the individuals to search inside a radius instead on being fixed to their local position. However, the laser usage underlies further preliminaries.

- **Individuals Characteristics**
  - initializes at random start position
  - uses asynchronous PSO
  - energy computation is based on wind and individuals movement
  - communicates inside a limited radius
  - uses collision avoidance

- can use lasers for scanning their local area
- have a defined spatial expansion

Third, the individuals movement underlies a few conditions. One of them is the constraint that, as well as the wind velocity, the individuals velocity is limited. Additionally, the movement is continuous like for real-world individuals. As the energy plays a major role in real-world robotic applications, the individual wastes energy for each movement. Thereby, a movement in the opposite direction of the wind wastes more energy than going in the same direction as the wind flows. The individuals power is provided by a battery. Inferring, the individual can no longer move on its own if it has expended the full energy and the battery is empty. However, one may not neglect the fact that the wind is still moving the individual even if its battery is empty. If the individual does not get information from other individuals, it may only use its own data and therefore may be mislead.

- **Individuals Movement**
  - moves by limited velocity
  - makes continuous movement
  - uses energy for movement
  - uses battery as energy supply
  - only move by wind flow if battery is empty

4.1.1. Collision Avoidance

In real-world scenarios where robots and drones are being used for exploring an area, collision avoidance plays a major role in performing the task. Therefore, collision avoidance has also been considered in this work. In this connection, two main phases have been considered. The first phase is collision detection illustrated by Figure 4.1. In this phase, the swarm is scanned and possible collisions between individuals are uncovered. The individuals i and j are colliding if the $distance_{ij}(t)$ between each others positions $\overrightarrow{pos}_i(t)$ and $\overrightarrow{pos}_j(t)$ is smaller than a defined minimum min. This minimum is twice the size s of an individual plus a defined repulsion distance rep.

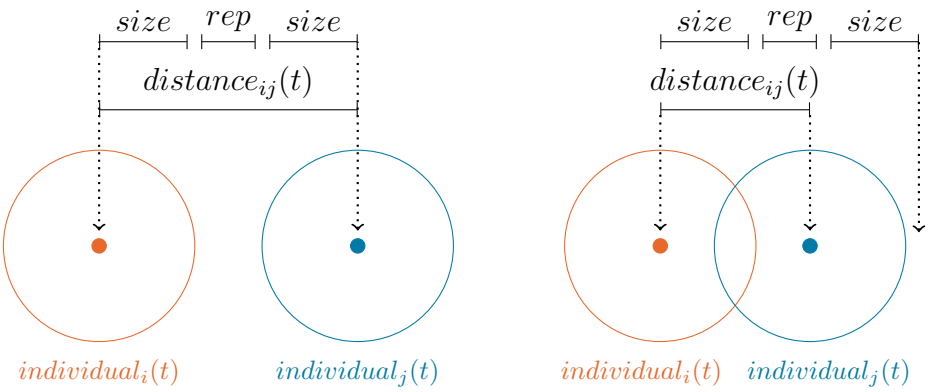


Figure 4.1.: Collision Detection

$$collision_{ij}(t) = \begin{cases} 1, distance_{ij}(t) < min \\ 0, else \end{cases} \tag{4.1}$$

$$distance_{ij}(t) = \left| \left(\overrightarrow{pos}_i(t) - \overrightarrow{pos}_j(t) \right) \right| \tag{4.2}$$

$$min = 2 \cdot s + rep \tag{4.3}$$

Subsequently, the collision resolution phase follows. If a collision has been detected, the individuals need to be rearranged. The goal is to resolve all detected conflicts. The new repositioning of individual i which is colliding with individual j is shown in Figure 4.2.

The new position of individual i, denoted as $\overrightarrow{pos}_i(t+1)$, is calculated by adding a force to the old position $\overrightarrow{pos}_i(t)$. This force creates a repulsion of the collision. It is generated by a direction vector and a strength factor. The direction vector determines the path of the repulsion. While the strength factor determines the length of the repulsion. In order to avoid another collision between individual i and individual j, individual i is pulled into the opposite direction of individual

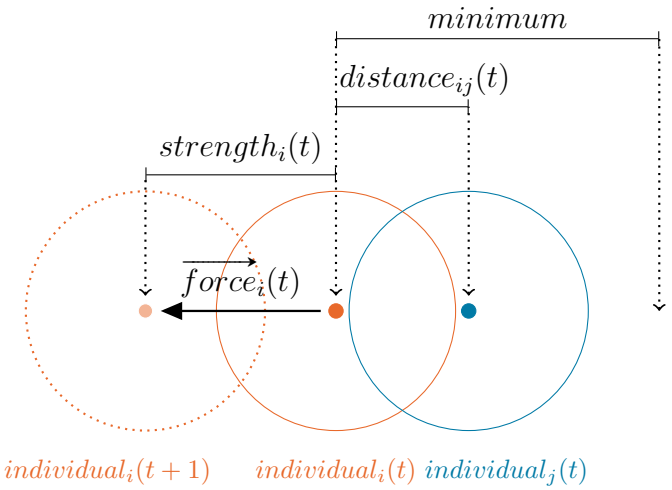


Figure 4.2.: Collision Resolution

j, which is displayed in Equation 4.6. The strength factor is based on the distance between the individuals, which means a weaker pulling for a larger distance and a stronger pulling if the distance is small. Since the reformation can result in new collisions, the two phases are repeated until all collisions are detached.

$$\overrightarrow{pos_i}(t+1) = \overrightarrow{pos_i}(t) - \overrightarrow{force_i}(t) \tag{4.4}$$

$$\overrightarrow{force_i}(t) = strength_i(t) \cdot \overrightarrow{direction_i}(t) \tag{4.5}$$

$$\overrightarrow{direction_i}(t) = \left\| \left(\overrightarrow{pos_i}(t) - \overrightarrow{pos_j}(t)\right) \right\| \tag{4.6}$$

$$strength_i(t) = \Big((2 \cdot size + rep) - distance_{ij}(t) \Big) \tag{4.7}$$

$$distance_{ij}(t) = \left| \left(\overrightarrow{pos_i}(t) - \overrightarrow{pos_j}(t)\right) \right| \tag{4.8}$$

4.1.2. Laser

Applying collision avoidance to PSO can result in a decreasing success rate for the optimization problem. That happens because the PSO swarm is designed in such way that individuals start accumulating at the global best position. The velocity vectors become smaller with time and the individuals start searching in very small areas. In contrast, collision avoidance prevents this accumulation and searches inside very small spaces because the individuals are hindering each other. In most cases, one single individual reaches the best found solution so far and searches for better solutions while the other individuals are moving around it as they cannot reach the global best position due to the collision as shown in Figure 4.3. As a result, this single individual remains the only one improving the search results. However, the PSO algorithm is based on the search of a collective of individuals and therefore, the results become worse compared to a PSO swarm without collision detection.

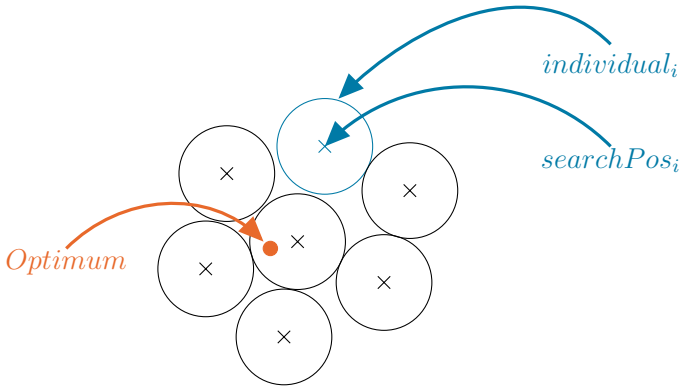


Figure 4.3.: Particle Swarm Without Laser

For the reason that collision avoidance is an important condition in real-world scenarios, this problem has been solved by the development of a laser presented in Figure 4.4. The laser works as an artificial antenna for each individual, increasing its search radius. Until then, an individual could only analyze the search space at the center of its body. However, with the laser, an individual can investigate the whole area around its center inside a predefined radius. As a result, the crowding at the best found solution is no longer a problem because

the individuals can reach the position of the best solution by its laser even if they cannot reach it by the center of their body. Consequently, this highly improves the fitness. The laser shall only be used in the case of crowding and searching in local areas which the individual cannot reach because of collisions. Thus, the laser is only activated after a determined threshold of unimproved iterations. To avoid the individuals from being stuck at local minima, the laser is also deactivated after a number of iterations in which the results do not change.

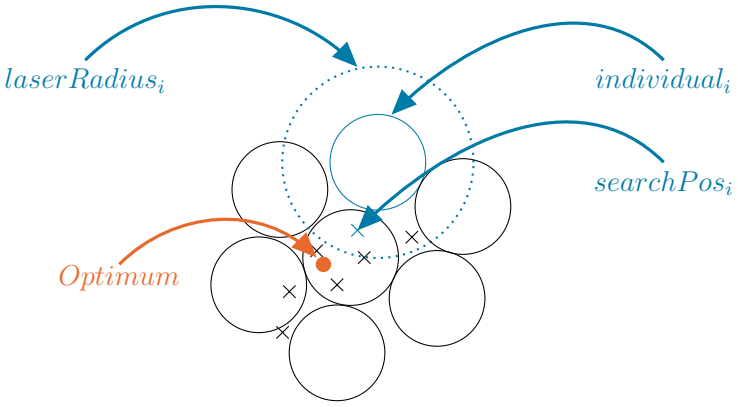


Figure 4.4.: Particle Swarm Using Laser

4.2. Energy Model

As already explained in the assumptions, each individual uses a battery for energy provision. This indicates that the usable energy is limited to the maximum of the batteries payload. In each iteration, an individual i wastes energy denoted as $eTotal_i(t)$. This energy is subtracted from the battery $battery_i(t)$ and determines the new battery load $battery_i(t+1)$. The total energy $eTotal_i(t)$ is calculated by three factors $eMove_i(t)$, $eWind_i(t)$ and $eAngle_i(t)$ as shown in Figure 4.5.

$$battery_i(t+1) = battery_i(t) - eTotal_i(t) \tag{4.9}$$

$$eTotal_i(t) = eMove_i(t) \cdot \big(1 + (eAngle_i(t) \cdot (1 + Wind_i(t))\big) \tag{4.10}$$

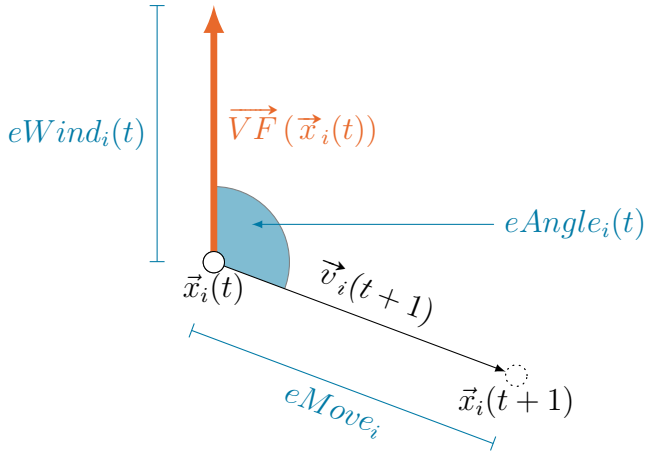


Figure 4.5.: Energy Model

One factor is the individual's velocity $eMove_i(t)$ or individuals movement. This includes the length of an individuals moving vector $\vec{v}_i(t+1)$ and generates a higher energy usage, the greater the vector turns out. The second factor $eWind_i(t)$ considers the wind vector $\overrightarrow{VF}\big(\vec{x}_i(t)\big)$ at the individuals position. The third factor $eAngle_i(t)$ describes the angle between the wind vector $\overrightarrow{VF}\big(\vec{x}_i(t)\big)$ and the individuals velocity vector $\vec{x}_i(t)$. The resulting value is inside a range of $[-180, 180]$, whereby the sign indicates if the angle is generated on the left hand of the wind vector or right-hand. Each factor is normalized by its maximum value. Consequently, $eTotal_i(t)$ is in a range of $[0, 3]$.

$$eMove_i(t) = \frac{|\vec{v}_i(t+1)|}{v_{max}} \tag{4.11}$$

$$eWind_i(t) = \frac{|\overrightarrow{VF}\big(\vec{x}_i(t)\big)|}{VF_{max}} \tag{4.12}$$

$$eAngle_i(t) = \frac{10^{angleRatio_i(t)} \cdot \big(angle_i(t) - 150\big) + 150}{max(10^{angleRatio_i(t)} \cdot \big(angle_i(t) - 150\big) + 150)} \tag{4.13}$$

$$angleRatio_i(t) = \frac{angle_i(t)}{360} \tag{4.14}$$

$$angle_i(t) = \frac{180}{\pi} \cdot \arctan 2\big(dott_i(t), cross_i(t)\big) \tag{4.15}$$

$$dott_i(t) = \vec{v}_i(t+1) \times \overrightarrow{VF}\big(\vec{x}_i(t)\big) \tag{4.16}$$

$$cross_i(t) = \vec{v}_i(t+1) \bullet \overrightarrow{VF}\big(\vec{x}_i(t)\big) \tag{4.17}$$

It is supposed that an individual needs the most energy if it is moving in the opposite direction of the wind vector and the absolute value of the angle is large. In contrast, the energy is the smallest if the individual is moving in the same direction as the wind, so the absolute value of the angle is small. The impact of the angle on the energy is illustrated in Figure 4.6.

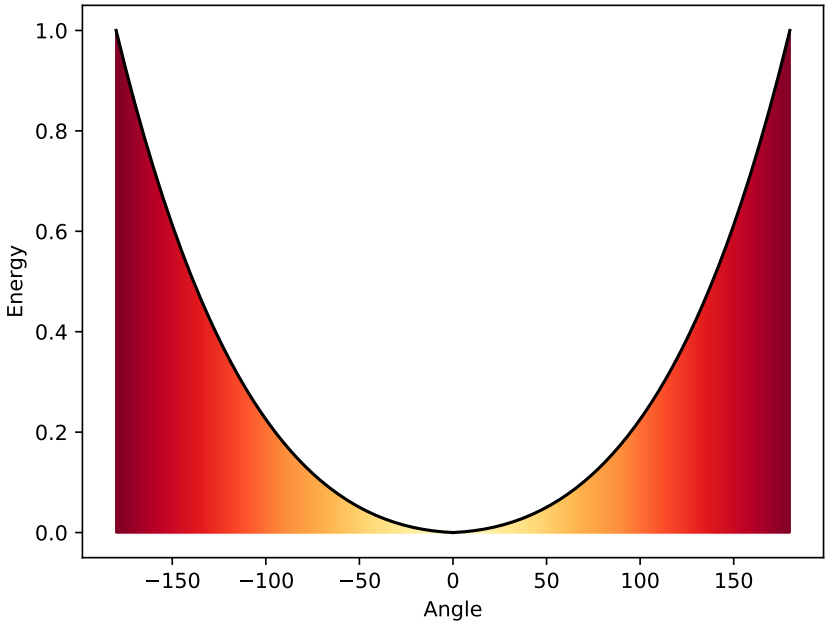


Figure 4.6.: Energy Plot

Another presentation is demonstrated by Figure 4.7. This figure refers to Figure 4.5 showing the size of the energy value by the color selection. Similar to the last figure, yellow indicates a small and red a high energy value. The black arrow represents the wind vector $\overrightarrow{VF}\big(\vec{x}_i(t)\big)$.

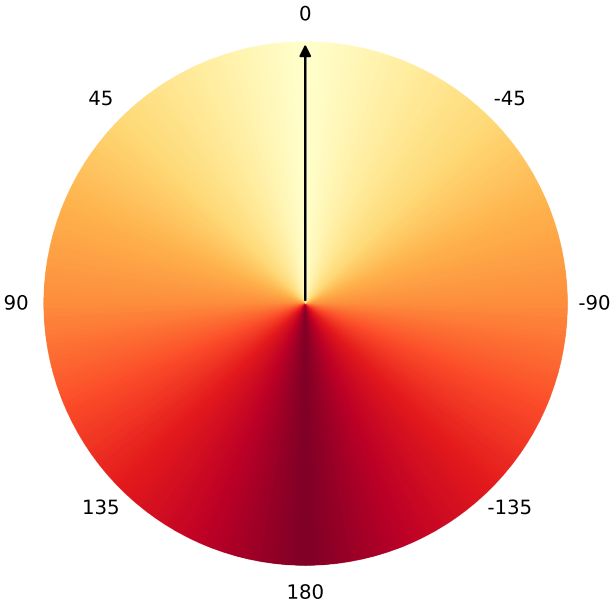


Figure 4.7.: Energy Circle

4.3. Swarm Models

In this work, two different swarm models based on PSO (2.1) will be presented. The first one is Power PSO (P-PSO) explained in Section 4.3.1 and the second one is Zigzag PSO (Z-PSO) introduced in Section 4.3.2. The goal of each of them is to overcome the influence of unknown vector fields and improve the search results. For this purpose, each model is applying a different strategy. While P-PSO is trying to reach the optimum as fast as possible and accepting a higher energy consumption for tackling the wind, Z-PSO takes a different approach by decreasing the angle between the wind velocity and its own velocity. The general velocity calculation $\vec{v}_i(t+1)$ for each particle in a swarm is shown below.

$$\vec{v}_i(t+1) = \overrightarrow{AV}\big(\vec{x}_i(t)\big) + \overrightarrow{VF}\big(\vec{x}_i(t)\big) \tag{4.18}$$

$$\overrightarrow{PSO}\big(\vec{x}_i(t)\big) = w\vec{v}_i(t) + C_1\sigma_1\big(\vec{P}_{best} - \vec{x}_i(t)\big) + c_2\sigma_2\big(\vec{x}_g(t) - \vec{x}_i(t)\big) \tag{4.19}$$

The particles new velocity $\vec{v}_i(t+1)$ is generated by the sum of the swarm specific velocity vector $\overrightarrow{AV}_i(t)$ and the wind vector $\overrightarrow{VF}\big(\vec{x}_i(t)\big)$. If a particle runs out of energy, the velocity vector $\overrightarrow{AV}_i(t)$ becomes zero and the velocity is only determined by the wind vector. As a result, a particle is following the wind stream. Since both models are established from PSO, they are calculating a velocity vector $\overrightarrow{PSO}\big(\vec{x}_i(t)\big)$ precisely as standard PSO, which determines their movement vector according to the optimization problem. However, this vector functions only as an input for each model and will be adapted to conform to their strategy. Consequently, each algorithm generates another velocity vector.

4.3.1. P-PSO

The Power PSO (P-PSO) is a modification of the standard PSO algorithm purpose-built for the application in unknown vector fields. The main problem of the standard PSO is that individuals are drifted away by the wind, especially if the individuals start accumulating at the optimum. That happens because the velocity vector becomes smaller, the closer the individuals get to their global best position. Accordingly, their movement vector is small, but the wind vector is as large as before. On these grounds, the wind is pushing the individuals away from the optimum. For overcoming this drawback, the P-PSO algorithm uses another strategy. If the PSO vector $\overrightarrow{PSO}\big(\vec{x}_i(t)\big)$ is smaller than a predefined threshold, it is increased until it reaches a given length denoted as $\overrightarrow{minPSO}\big(\vec{x}_i(t)\big)$ and displayed in Figure 4.8. In areas without wind influence this behaviour would be misleading, but in vector fields it is very useful for the individuals in fighting the disturbances caused by the wind.

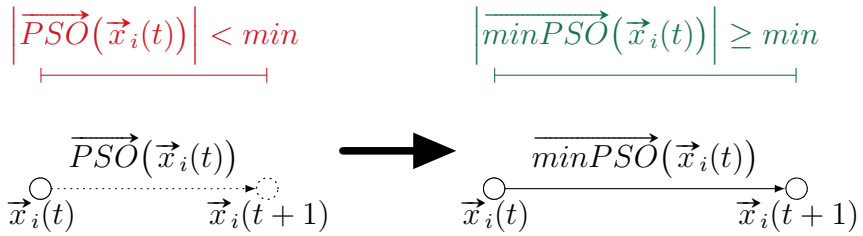


Figure 4.8.: P-PSO

In Figure 4.9, the complete P-PSO algorithm is illustrated, including three different cases. For each case, the evaluation of the resulting velocity vector $\overrightarrow{AV}(\vec{x}_i(t))$ is described below.
First, the PSO vector is calculated and its length is verified. The first case is applied if the length is equal to zero. In this case, an individual would not be moving at all. However, the P-PSO strategy provides a constant movement and does not tolerate stagnancy. Therefore, the velocity vector needs to have a minimum length. This is ensured by using the same velocity vector as in the last iteration indicated by $\overrightarrow{AV}(\vec{x}_i(t-1))$.
In the second case, the PSO vector is larger than zero but still smaller than a predefined minimal value *min*. Thus, the existing PSO vector is multiplied by the factor ten until it reaches the minimum required length. As a result, the swarm is constantly moving even if it is close to the optimal position. This strategy would not have worked out for case one because a vector with length zero cannot be increased by multiplication.
For the last case, the PSO vector already reaches the needed length and no modification is needed. That means that the velocity vector is exactly equal to the PSO vector.

$$Case1 : \overrightarrow{AV}(\vec{x}_i(t)) = \overrightarrow{AV}(\vec{x}_i(t-1)) \tag{4.20}$$

$$Case2 : \overrightarrow{AV}(\vec{x}_i(t)) = \overrightarrow{minPSO}(\vec{x}_i(t)) \tag{4.21}$$

$$Case3 : \overrightarrow{AV}(\vec{x}_i(t)) = \overrightarrow{PSO}(\vec{x}_i(t)) \tag{4.22}$$

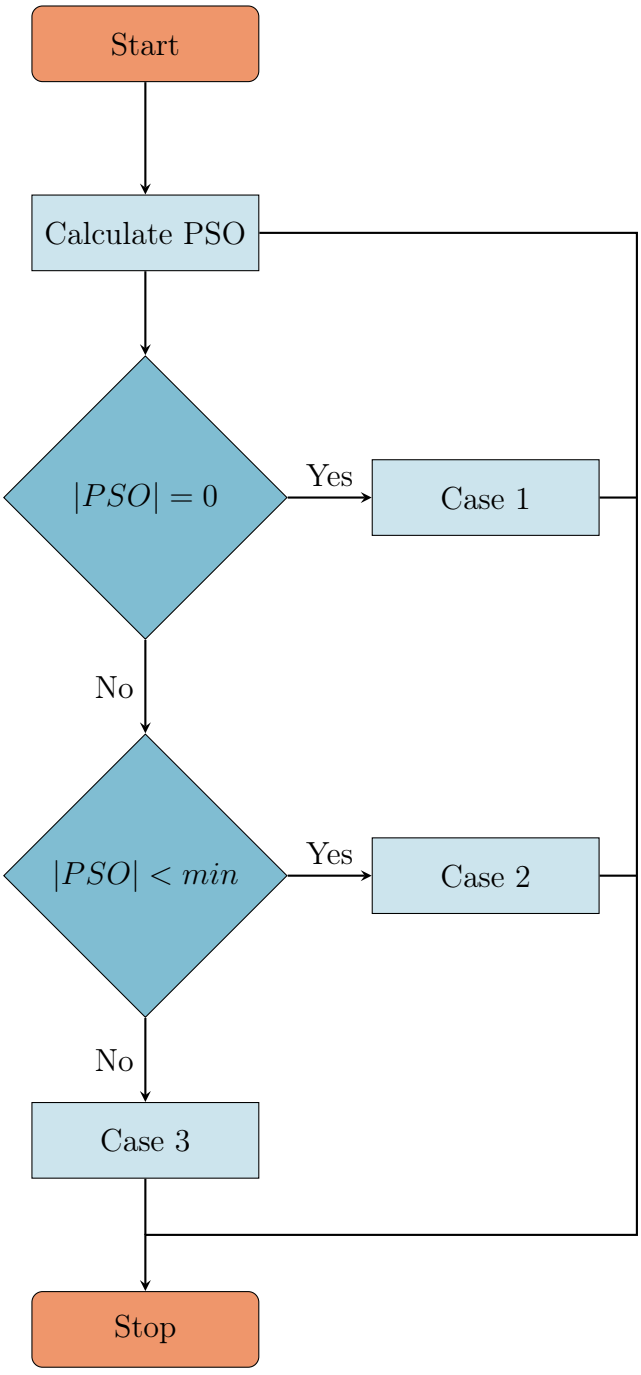


Figure 4.9.: P-PSO Algorithm

4.3.2. Z-PSO

The Zigzag PSO (Z-PSO) algorithm is an adaption of the standard PSO algorithm with the intention to reduce energy consumption while not neglecting the fitness. The Z-PSO shall prohibit moving exactly in the opposite direction of the wind vector and by this wasting a lot of energy. The idea is based on sailboats which are doing zigzag movements if they are sailing in the opposite direction of the wind. Similar to sailboats, the Z-PSO algorithm shall adjust the individuals velocity to the wind vector in a suitable way. Since the wind is unknown, the individuals need to try to predict the wind force indicated by $\overrightarrow{VFP}\big(\vec{x}_i(t)\big)$. The forecast is generated by the error of their last movement. This means in particular, an individual i knows its last position $\overrightarrow{pos}_i(t-1)$, the last move vector $\overrightarrow{AV}\,(\vec{x}_i(t-1))$ and its new position $\overrightarrow{pos}_i(t)$. Due to wind interruption, in most cases the individual will not have reached the lately targeted position $\overrightarrow{tpos}_i(t-1)$ which it should have reached by its move vector $\overrightarrow{AV}\,(\vec{x}_i(t-1))$. As a result, the individual can calculate the difference between the planned position $\overrightarrow{tpos}_i(t)$ and its real position $\overrightarrow{pos}_i(t)$. This is exactly the wind vector $\overrightarrow{VF}\big(\vec{x}_i(t-1)\big)$ distracting at the last position. One disadvantage of this approach is that the wind prediction for the new position is based on the wind at the last position, but the wind might be different at the new position. However, this is insignificant because the wind is very similar at local positions and is not highly varying in between the range of the individuals possible movement.
As shown in Figure 4.10, the PSO vector is rotated if the angle α is not inside the range of $[-135, 135]$ to prevent the individual from moving in the opposite direction of the wind vector. Thus, the individuals velocity will result in zigzag movement. For the case that a particle does not have any neighbors, the particle does not move at all but follows the flow. Because of the assumption that a single particle without interaction with other particles is not able to find the optimum. Therefore it reduces its energy to a minimum until it reaches other particles again.

$$\overrightarrow{VFP}\big(\vec{x}_i(t)\big) = \overrightarrow{pos}_i(t) - \overrightarrow{tpos}_i(t-1) \tag{4.23}$$

$$\overrightarrow{tpos}_i(t-1) = \overrightarrow{pos}_i(t-1) + \overrightarrow{AV}\,(\vec{x}_i(t-1)) \tag{4.24}$$

$$\alpha_i(t) = \frac{180}{\pi} \cdot \arctan 2\left(dott_i(t), cross_i(t)\right) \tag{4.25}$$

$$dott_i(t) = \overrightarrow{VFP}\left(\vec{x}_i(t)\right) \times \overrightarrow{PSO}\left(\vec{x}_i(t)\right) \tag{4.26}$$

$$cross_i(t) = \overrightarrow{VFP}\left(\vec{x}_i(t)\right) \bullet \overrightarrow{PSO}\left(\vec{x}_i(t)\right) \tag{4.27}$$

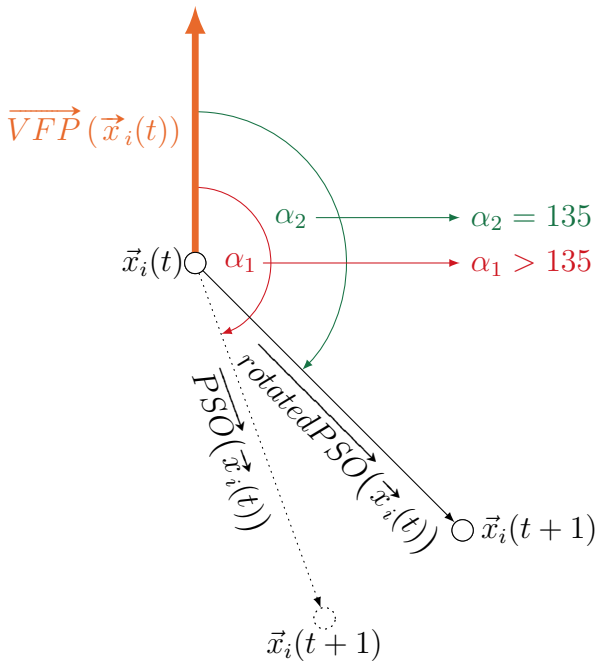


Figure 4.10.: Z-PSO

An overview of the algorithm is provided by Figure 4.11. The algorithm can result in three different cases and velocity vectors described below.
In the first case, the algorithm proofs if the individual is connected to any other particle in the swarm. If this is not the case, the velocity vector is described as a zero vector. As a result, the particle follows the flow and does not do any movement itself. Second, the algorithm checks if the particle would move in

the opposite direction of the predicted wind vector $\overrightarrow{VFP}\big(\vec{x}_i(t)\big)$ if it follows the standard PSO vector $\overrightarrow{PSO}\big(\vec{x}_i(t)\big)$. Therefore, the individual calculates the angle α between the two vectors.
A movement in the opposite direction of the wind vector is defined by an angle between $-135°$ and $-180°$ or, respectively, $135°$ and $180°$. Consequently, if the angle is inside a range of $[-135, 135]$ degree, the individual is not moving in the contrary direction of the wind vector. Thus, case two is applied and the standard PSO vector $\overrightarrow{PSO}\big(\vec{x}_i(t)\big)$ is set as the new velocity. Additionally, the predicted wind $\overrightarrow{VFP}\big(\vec{x}_i(t)\big)$ is subtracted. This increases the chance that the individual is moving towards the targeted position by the PSO vector and is less influenced by the wind.
Though, if the angle is not inside this range, the PSO vector needs to be rotated until the angle fits the scope. The newly generated vector $\overrightarrow{rotatedPSO}\big(\vec{x}_i(t)\big)$ is as well subtracted by the predicted flow distribution $\overrightarrow{VFP}\big(\vec{x}_i(t)\big)$.

$$Case1 : \overrightarrow{AV}\big(\vec{x}_i(t)\big) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{4.28}$$

$$Case2 : \overrightarrow{AV}\big(\vec{x}_i(t)\big) = \overrightarrow{PSO}\big(\vec{x}_i(t)\big) - \overrightarrow{VFP}\big(\vec{x}_i(t)\big) \tag{4.29}$$

$$Case3 : \overrightarrow{AV}\big(\vec{x}_i(t)\big) = \overrightarrow{rotatedPSO}\big(\vec{x}_i(t)\big) - \overrightarrow{VFP}\big(\vec{x}_i(t)\big) \tag{4.30}$$

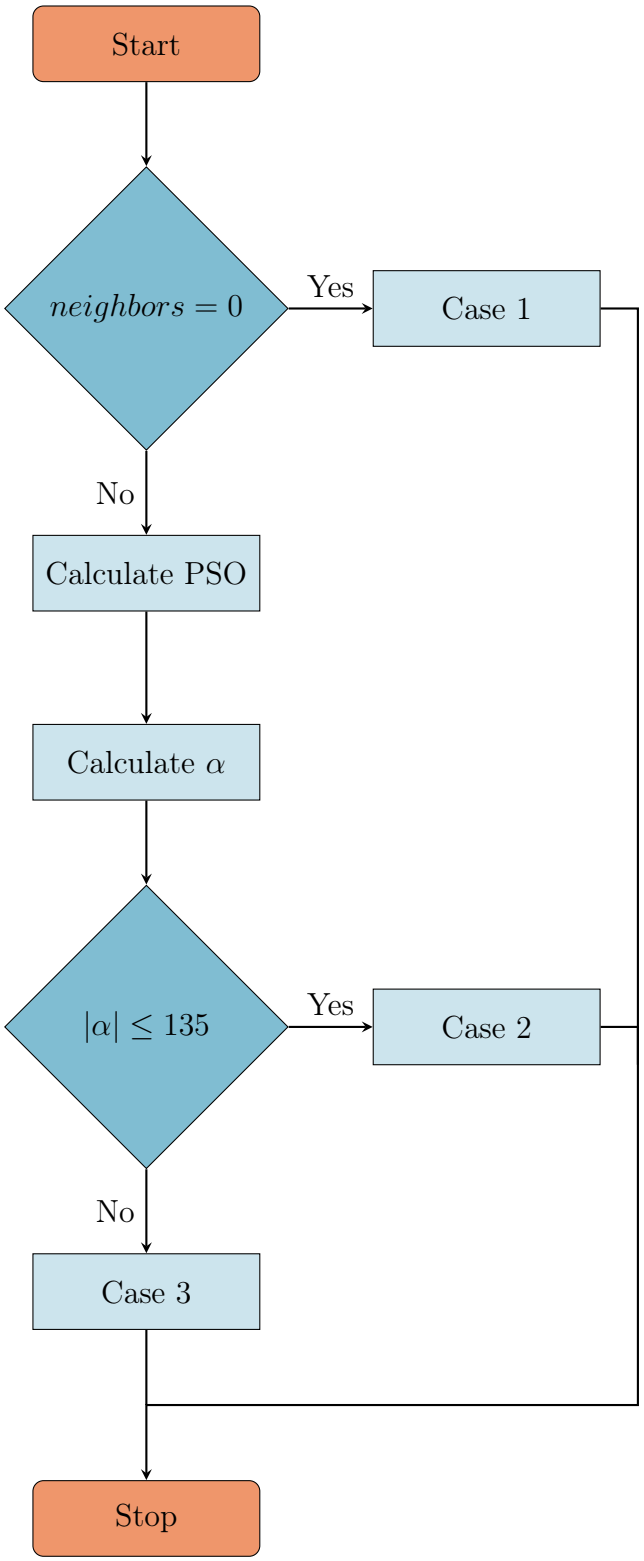


Figure 4.11.: Z-PSO Algorithm

4.4. Neighborhood Models

This section is about the newly developed neighborhood models. The concepts lying behind these approaches have been inspired by the insights of the research in Chapter 3. Some of them have been advanced on the basis of game theoretic approaches like the Payoff Model in Section 4.4.1. In contrast, the Probability Model in Section 4.4.2 is grown on stochastic parameters. Furthermore, hybrid approaches have been created by combining both models in Section 4.4.3. In Section 4.4.4 models are explained performing different, already known models by applying a switch strategy.

4.4.1. Payoff Model

The Payoff Model is based on the n-player Iterated Prisoner's Dilemma (NIPD) [33]. Each individual represents one player who needs to decide whether he cooperates or defects. Compared to the known NIPD, the individual does not cooperate with a single other player but with a group of players called coalition. The strategies of cooperation and defection are interpreted as the following. If an individual cooperates, it defines a coalition as its neighborhood and exchanges information. If it defects, there is no information exchange. In this model, an individual only cooperates with one coalition, which is the one providing the highest payoff. If two or more coalitions score the same value, one of them will be selected randomly. The payoff values for each coalition are stored in each individuals memory. As a result, each individual can hold different payoff values for the same coalition. At each iteration, the recently chosen coalition will be rated and the payoff table will be updated accordingly. If the individuals global best value has improved since the last iteration, the taken coalition will get a higher payoff, increasing the chance for being picked again. However, if the fitness did not improve, the payoff will be decreased. Independent of the shift, the payoff is always between zero and one. After the update, the possible new coalitions are generated. Therefore, a maximum of five individuals is selected from the individuals inside the communication radius. All possible combinations or coalitions are formed from this set of individuals. Consequently, there are in total 2^5 coalitions. Without this limitation, the computation time would be beyond the scope. In the case of more than five possible individuals inside the radius, it is decided by chance. Then, all possible subsets of these individuals will be created and the one with

the highest payoff will be determined. As already explained, this coalition will be the one with whom the individual will cooperate and exchange information.

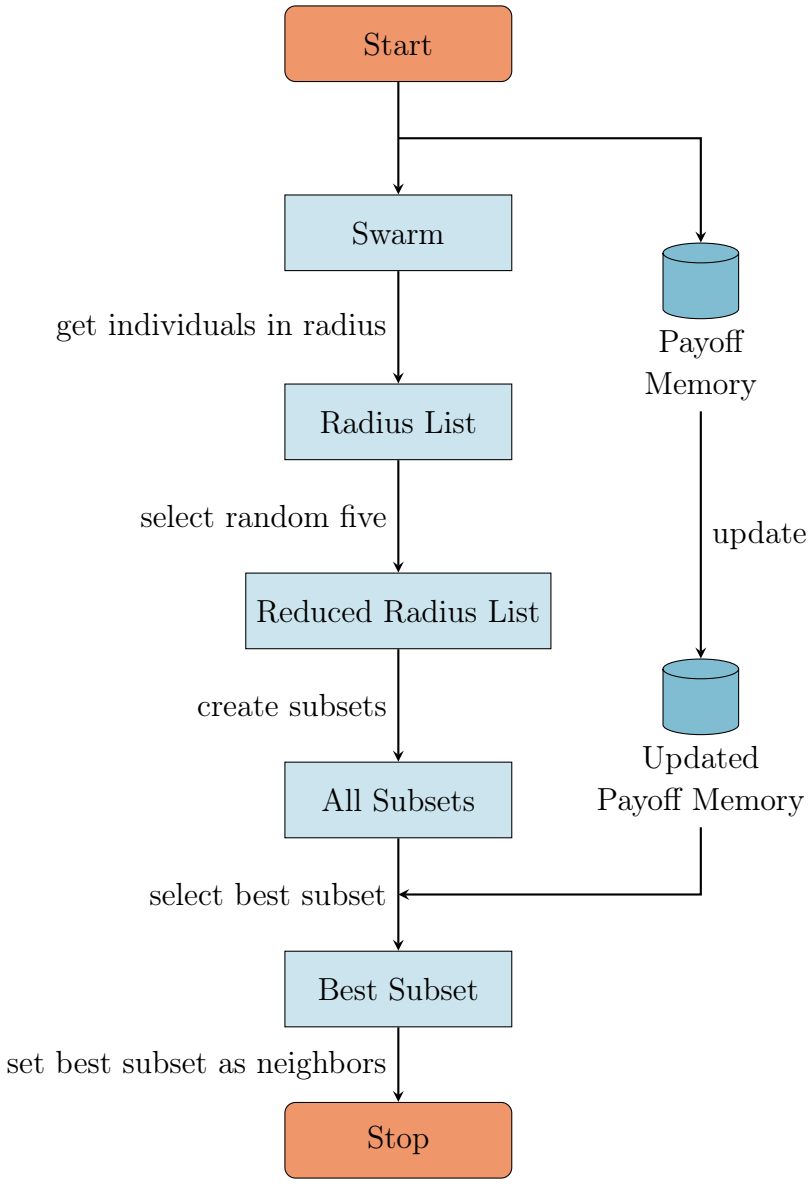


Figure 4.12.: Payoff Algorithm

4.4.2. Probability Model

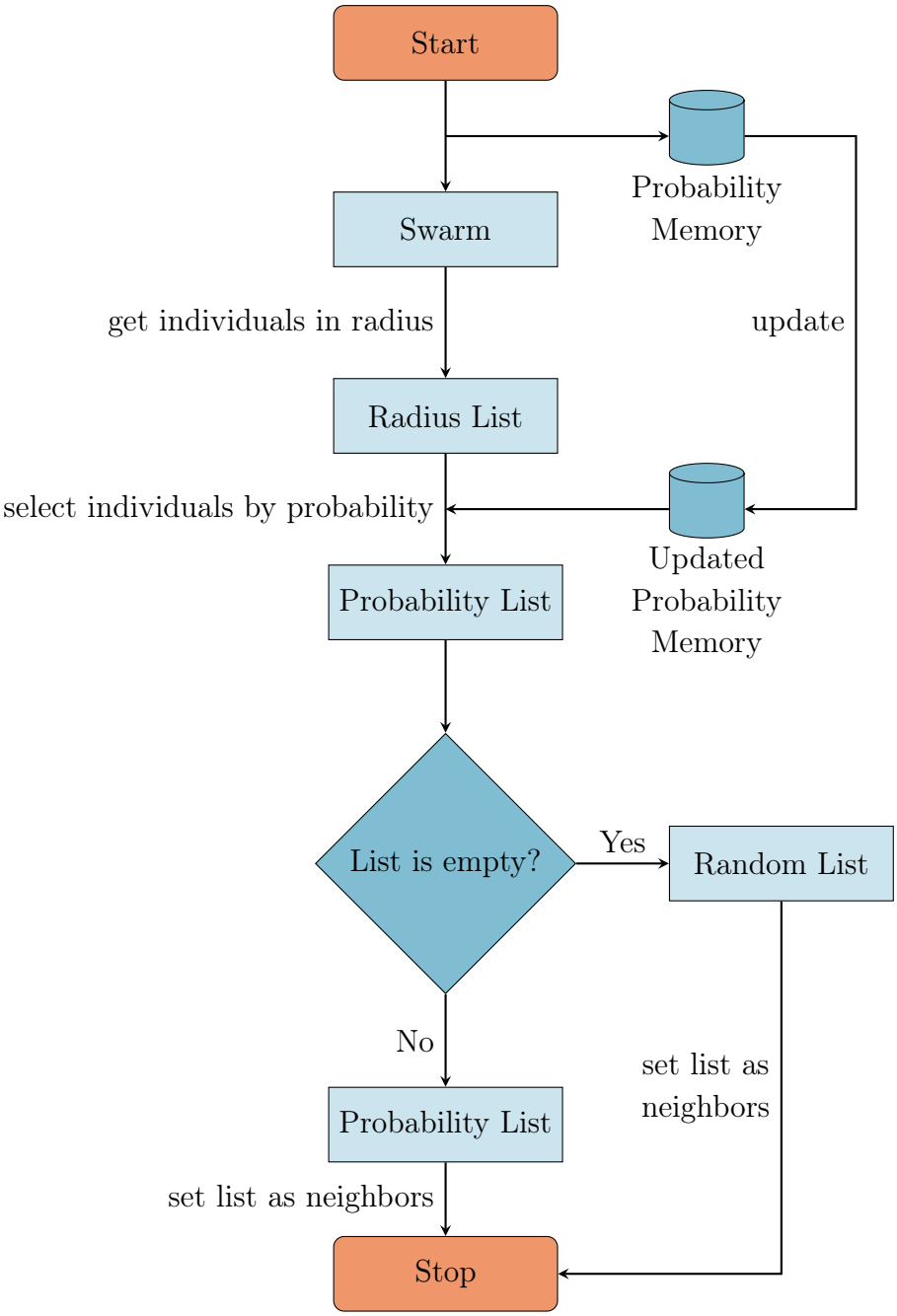


Figure 4.13.: Probability Algorithm

The fundamental base for the Probabilistic Model is build on the work of Cui et al. [8]. They proposed a new PSO variant combined with game theoretic approaches. The PSO individuals needed to select different strategies based on a probabilistic value equal to players strategy selection in Game Theory. After each move, the values were updated according to the individuals fitness [8]. Analog to this approach does the Probabilistic Model work. An individual gets a list of individuals inside its communication radius. Afterwards, each of them is selected according to a probability factor stored in the individuals memory. The chosen individuals build the new neighborhood in this iteration. If the probability values are low and no individual is selected, at least one random individual is defined as a neighbor. This exception is made because the PSO algorithm does not work well if an individual has no opportunity to exchange information. It would be the same scenario as if an individuals decision is only based on its individual component. After each iteration, the probability values of the neighbors are updated. If the individual's global best value has improved during the last iteration, the probabilities of the last neighbors are increased. In contrast, the probabilities decrease if the value did not improve. The resulting value is always inside a range of 0.05 and 1. This guarantees that there always is a chance that an individual is selected.

4.4.3. Payoff Probability Model

The Payoff Probability Model is a hybrid approach adapted from the Payoff Model (Section 4.4.1) and the Probability Model (Section 4.4.2). At the beginning, as well as in the Payoff Model, the coalition with the highest payoff is selected. Though, not all the individuals inside the coalition will be chosen with certainty. Instead, the Probability Model is applied. Each individual is added to the resulting neighborhood by the chance of its probability value. In the case that no individual of the coalition is selected, at least one random individual will be selected. After each iteration, the payoff memory and probability memories are updated according to the fitness trend of the individual which searched for a neighborhood.

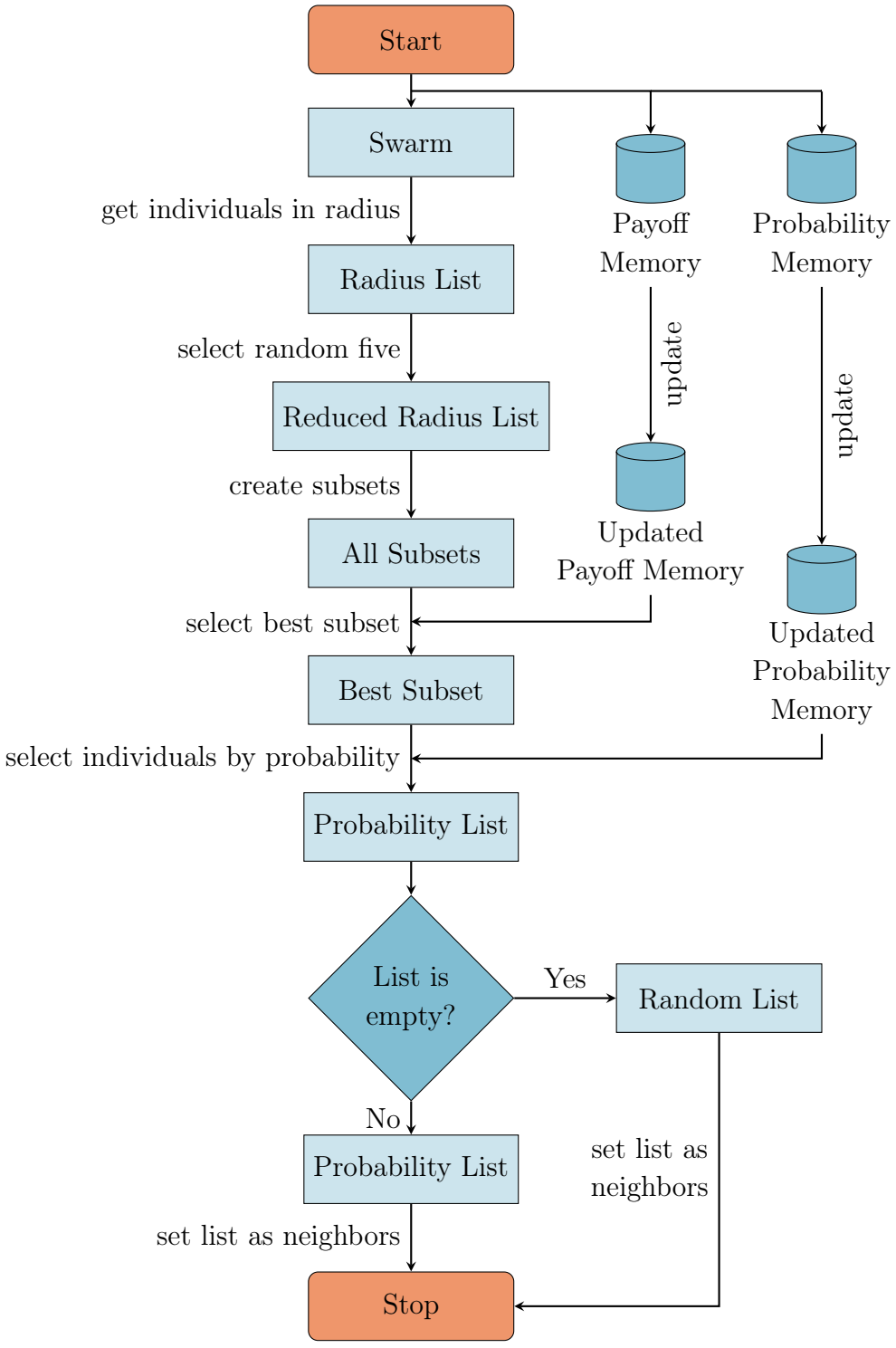


Figure 4.14.: Payoff Probability Algorithm

4.4.4. Switch Strategy Models

Another approach inspired by Game Theory has been proposed by Di Chio et al. [9]. In their study, they applied game theoretic approaches to individual Swarm Optimization. They interpreted the swarm individuals as players in a Prisoner's Dilemma Game (Section 2.3.1) who need to decide between cooperation and defection. For the understanding of cooperation and defection, they came up with different variants. Furthermore, they introduced different mechanisms for changing a player's strategy in a game based on an individual's fitness. Consequently, a player can change between cooperation and defection. In this work, three different models have been developed which use different strategies. They are listed below. Each model is based on one of the previously announced models. Since the only difference of the models is the underlying neighborhood model, they will be explained exemplary by the Payoff Switch Model.

- Payoff Switch Model
- Probability Switch Model
- Payoff Probability Switch Model

The model is similar to the Payoff Model (Section 4.4.1) but with the fundamental difference that an individual can change its strategy. In the context of the Prisoner's Dilemma (Section 2.3.1), the strategies cooperate and defect are construed as subsequent. Cooperation is considered as behaving according to the Payoff Model. Whereas, defection lets the individual use another neighborhood model. In this work, the other neighborhood model was selected from the basic and common topologies described in Section 2.2. The available selections were the neighborhoods GBest, Ring, Von Neumann and Star. Since GBest is a fully connected network and has been highly investigated, it is excluded. The Star topology is also not a good selection since the topology requires that the whole swarm acts in conform to this model since one individual of the swarm needs to be defined as the master which acts as the central point of transfer. As a result, only Ring and von Neuman are left. Several test simulations have shown that a swarm using Ring did slightly better than a swarm using Von Neumman topology considered in overall simulation

cases. Therefore, the Ring topology has been chosen as the alternative strategy. After each iteration, the strategy is changed by a probability value which is stored in the memory. If the fitness has improved, the value decreases and if the fitness stagnates, the probability rises. Consequently, stagnation makes it more likely that a strategy switch is performed.

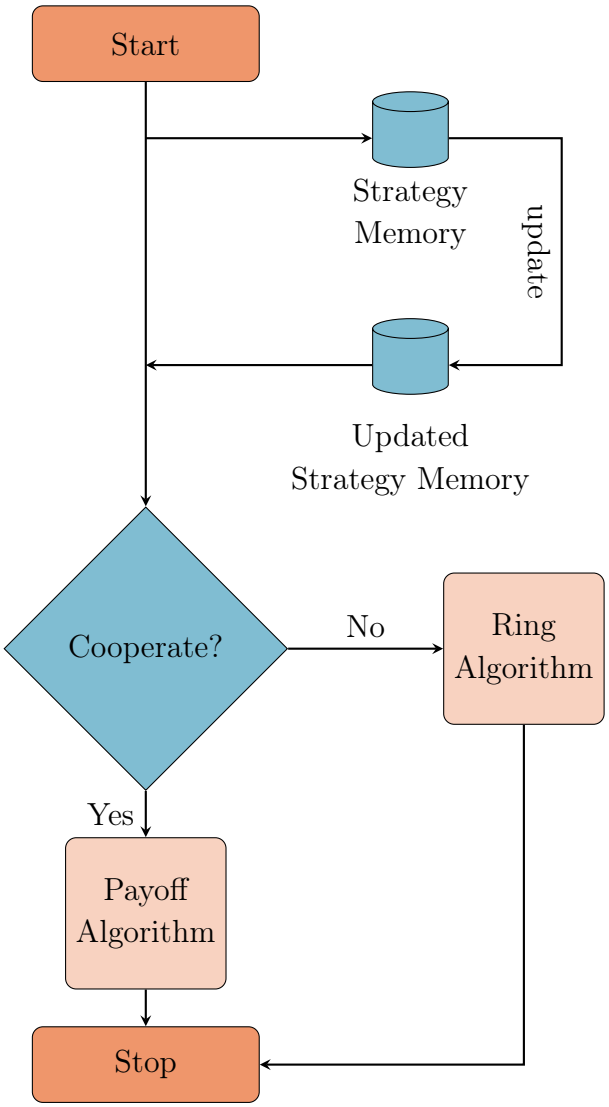


Figure 4.15.: Switch Algorithm

5. Implementation

This chapter offers information about the implementation of the experiments in Chapter 6. In Section 5.1, the general simulation setting and structure are introduced. The results of this thesis may provide important insides especially for real robotic systems. Therefore, it is engaging to consider as many real-world components in the simulation as possible. Section 5.2 presents those factors regarding the particle implementation and particle movement. Additional, Section 5.3 introduces further realistic adaptions in relation to the implementation of the environment.

5.1. Simulation

The simulation environment was developed using the open-source language Processing[1] which is based on the Java[2] programming language. The advantages of Processing are that it comes with built-in graphical visualization which is important especially while testing and developing new approaches. Furthermore, the Java foundation makes it possible to import all other available Java libraries and therefore to extend the functional scope.
The selected development environment was Atom[3], which is an open-source integrated development environment (IDE) providing a wide range of extendable add-ons and plugins also for Processing. This includes for example auto-completion and syntax highlighting, which are highly facilitating the implementation process. Additional, the version control system git[4] can be integrated as it was done in this work.
For the phase of the methodology development, a graphical user interface was created. It offers the possibility to easily observe the swarms behaviour.

[1] https://processing.org/
[2] https://www.java.com/de/
[3] https://atom.io/
[4] https://git-scm.com/

Using the GUI, one can effortlessly change the selected vector field, objective function, neighborhood or swarm. Figure 5.1 shows a screenshot of the interface.
Since Processing is designed for graphical output and visualization but not for statistical computation, the results of the experiments were not evaluate using Processing. Instead, the results were stored in separate text files for further processing and analysis.
For the evaluation, the programming language Python[5] was selected. One of its advantages is that it allows an easy and fast data processing with low effort. Moreover, there is an enormous collection of extendable libraries including plugins for analytical issues.

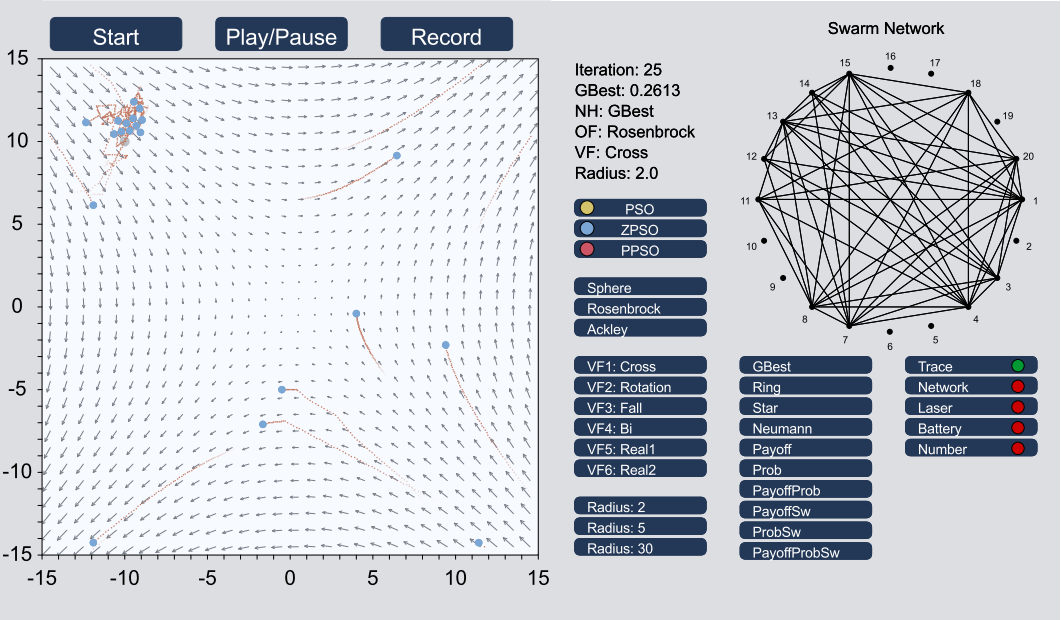


Figure 5.1.: Graphical User Interface

5.2. Particles

Several components have been considered in this work to make the simulation more realistic. This also includes the implementation of the particles movement. Therefore, an important factor is a continuous movement. During one iteration, a particle is able to change its position inside a defined radius.

[5] https://www.python.org/

However, this iteration needs to be discretized in respect to real movement. Otherwise, a particle would create jumps between two iterations. The iteration has been divided or discretized into sub iterations. Equally, the velocity vector has been divided allowing a proper movement. This includes that a particle is also influenced by the vector field on more positions during one iteration, which is more similar to real world robotic system. In this work, the iterations have been divided by a value of ten. A higher value possesses the advantage of more realistic particle behaviour. However, in return, the computational effort rises due to more calculations.
Apart from a discretized movement, collision avoidance has been considered. In real-world scenarios, robots will avoid collisions, particularly to prevent damage but also to not hinder each other. Consequently, collision detection and resolution are an important factor in realistic simulation development. In the field of PSO, collision avoidance will highly affect the search process and fitness value because the particles have less opportunities to find the optimum. A particle has a body which defines its collision radius. However, in the search process, a particle can only investigate the point exactly where it is positioned and not inside its body shape. Though, the other particles cannot search inside this area because of the collision detection. In this work, the collision avoidance has been implemented in such a way that particles calculate their new position and before a new iteration starts, a collision detection is applied over all particles. If two or more particles are colliding, their new velocities and positions are generated as explained in Section 4.1. This is an easy and efficient handling, whereby it leaves some problems unsolved. Real robots would not be colliding in the first place but preventing such movement and taking into consideration the positions of the other robots. Nevertheless, an exact and concrete collision avoidance would have been beyond the scope of this work. Therefore, this simple but well-working solution has been chosen.

5.3. Real Data Vector Fields

The set of vector fields for the simulations are generated by functions and are computable. However, this thesis shall provide a basis for real-world scenarios. Therefore, the list of vector fields has been extend by two real-world vector fields from ocean wind data to simulate a more realistic environment. NASA[6]

[6] `https://www.nasa.gov/`

offers online access to real-world and time-dependent data about wind flows all over the world. The data is accessible by OPenDAP[7], which is an open-source framework facilitating scientific data networking over the internet in an easily processible format. Python provides a specific OPenDAP library for accessing and downloading this data.
In this thesis, wind flow data from Indian Ocean[8] was used. The data collection extends from 1987-07-02 to 31.12.2011. The date for the selected data was randomly set to 12.01.2010. The chosen area for vector field *Real1* has been defined by the longitude values in the range of $[60, 80]$ and latitude values in the range of $[0, 20]$. Vector field *Real2* was defined by the longitude values $[70, 90]$ and latitude values $[-50, -30]$. The selection of the longitude and latitude values has been made on the divergence of the wind flows inside this area. One has tried using areas with varying wind directions and strengths.

- Vector Field *Real1*
  - Date: 2010-01-12
  - Longitude: $[60, 80]$
  - Latitude: $[0, 20]$
- Vector Field *Real2*
  - Date: 2010-01-12
  - Longitude: $[70, 90]$
  - Latitude: $[-50, -30]$

The data was downloaded and stored in a temporary text file before it was imported and converted by the simulation program. In contrast to the other vector fields, the database from real-world flow is not continuous. For this reason, the search space has been divided into grid cells and one vector has been assigned to one grid cell. The resulting vector fields are shown in Figure 5.2 and Figure 5.3.

[7] `https://www.opendap.org/`
[8] `https://thredds.jpl.nasa.gov/thredds/dodsC/ncml_aggregation/OceanWinds/ccmp/aggregate__CCMP_MEASURES_ATLAS_L4_OW_L3_0_WIND_VECTORS_FLK.ncml.html`

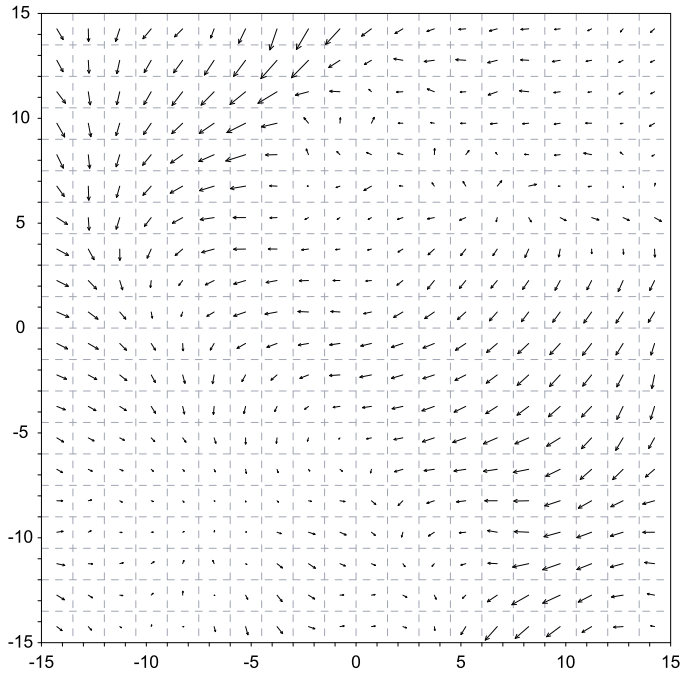


Figure 5.2.: Vector Field Real 1

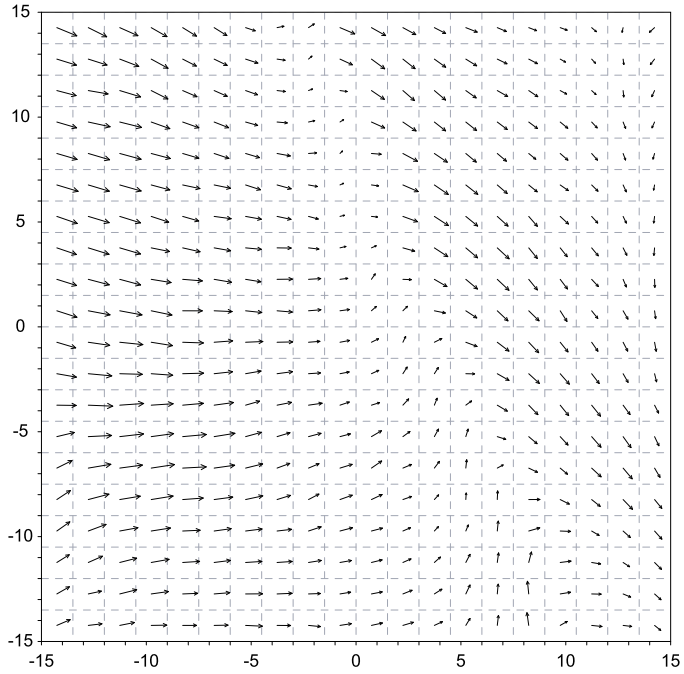


Figure 5.3.: Vector Field Real 2

6. Evaluation

The PSO approaches and neighborhoods presented in Chapter 4 are tested and compared with each other in different scenarios. This includes various dynamic environments and objective functions. The considered metrics evaluate the general approaches of fitness, energy consumption, accumulation, and success rate. Furthermore, the particles trajectory of the PSO approaches is analyzed. First, the experimental setup is proposed in Section 6.1. Second, the results are presented in Section 6.2. Finally, the main results and insights derived from the experiments are presented in Section 6.3.

6.1. Experiments

The search space for the experiments is defined as a two-dimensional grid with $(x_1, x_2) \in [-15, 15]$ in which N swarm members are placed randomly at the beginning. The optimum, which is defined as 0 ± 0.01 is placed at $[-10, 10]$. The parameter settings for the PSO algorithm have been derived from extensive tests in advance. Thus, the inertia weight is set to 0.6 and the acceleration coefficients are defined as 1.0. The end of the simulation is reached after 150 iterations. Each iteration consists of 10 sub iterations resulting in a more realistic particle movement.
Additionally, velocity constraints and further parameters are set. This includes the limitation of a particles velocity to v_{max} as well as the restriction of the wind velocity to w_{max}. The minimal velocity of the P-PSO particles is 1. For the collision handling, a particle size of 0.7 and a repulsion value of 0.15 are used. If an individual does not improve its fitness for 5 iterations, the laser is activated as described in Section 4.1.2, scanning the area in a radius of 2.5. Additionally, the communication radius is limited to 2.0.

Table 6.1.: Parameter Values

Description	Parameter	Value
Search Area		
Search Space	d	2
Grid Width	X	$[-15, 15]$
Grid Length	Y	$[-15, 15]$
Optimum Value	p	0 ± 0.01
Optimum Position	p_{pos}	$[-10, 10]$
PSO		
Population Size	N	20
Inertia Weight	w	0.6
Acceleration Coefficients	C_1, C_2	1.0
Iterations	I	150
Sub Iterations	I_{sub}	10
Simulation		
Particle Velocity Limit	v_{max}	2.5
Flow Velocity Limit	w_{max}	2
P-PSO Velocity Minimum	v_{min}	1
Particle Size	s	0.7
Repulsion	rep	0.15
Laser Counter	L_c	5
Laser Radius	L_r	2.5
Communication Radius	c_{max}	2
Neighborhoods		
Memory Value Initial	m_{init}	0.5
Memory Value Minimum	m_{min}	0.05
Memory Value Maximum	m_{max}	1
Memory Update Value	m_{update}	± 0.05

The tested neighborhoods are described in Section 4.4. Though, not all neighborhoods are used in the experiments. For the neighborhoods which are based on a switching strategy, tests in advance have shown that Probability Switch topology provided the best results. Therefore, the other two neighborhoods Payoff Switch and Payoff Probability Switch are neglected. For the updating process, the memory values are initialized equally for all neighborhoods by m_{init} and can be in a range of m_{min} and m_{max}. For the updating process, the memory values are increased or decreased by m_{update}.

The experiments analyze the performance of two PSO swarms introduced in Section 4.3. P-PSO and Z-PSO have been intentionally designed for aerial robots dealing with unknown environmental influences. The main focus is based on a limited communication radius and the impact of varying topologies. Therefore, three different objective functions are tested in 31 simulations, each consisting of 150 iterations. The analyzed optimization problems are presented in the following.
The first objective function is the Sphere Function, which is also the simplest optimization problem. It can be seen as a general examination of the approaches and neighborhoods.

$$f(x) = \sum_{i=1}^{|X|} (x_i)^2, \quad x \in X \tag{6.1}$$

The second optimization problem is Rosenbrock Function which is more challenging than Sphere because it provides plateaus which may be misleading and hindering the search process.

$$f(x) = \sum_{i=1}^{|X|-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \quad x \in X \tag{6.2}$$

Ackley function is the last optimization problem offering multiple local minima. The challenge is that the particles do not stagnate at these local minima but find the global optimum.

$$f(x) = -20 \exp\Big(-0.2\sqrt{\tfrac{1}{|X|}\sum_{i=1}^{d} x_i^2} - \exp\big(\tfrac{1}{d}\sum_{i=1}^{d} \cos(\tfrac{1}{2}\pi x_i)\big)\Big) + a + e, \quad x \in X \tag{6.3}$$

Each of the objective function is tested in four different vector fields which are listed below. The vector fields Cross and Fall are generated by functions. In contrast, vector fields Real1 and Real2 are calculated by real data as described in Section 5.3.

Table 6.2.: Vector Fields

Description	Title	Function
"Cross"	VF1	$\overrightarrow{VF}_1(x_1, x_2) = (x_2, x_1)$
"Fall"	VF2	$\overrightarrow{VF}_2(x_1, x_2) = (0, -2)$
"Real1"	VF3	Longitude: $[60, 80]$, Latitude: $[0, 20]$
"Real2"	VF4	Longitude: $[70, 90]$, Latitude: $[-50, -30]$

6.2. Results

This section presents the results of the analysis. Therefore, the convergence and fitness of the swarms have been compared in Section 6.2.1. Furthermore, the results of the neighborhoods have been compared inside the PSO approaches. The energy analysis in Section 6.2.2 presents the general energy usage for each swarm and neighborhood during the simulation. Section 6.2.3 accumulation and success rate are analysed. Afterwards, Section 6.2.4 explains the general particles movement for each approach for two vector fields.

6.2.1. Convergence Analysis

This sections analyzes the convergence rate of the two PSO swarms mentioned in Section 4.3 with respect to the topologies introduced in Section 4.4. The convergence rate represents the fitness of a swarm according to the selected neighborhood. In each iteration, it displays the value of the best solution found so far by the swarm. In the following, the well-known topologies GBest, Ring, Star and Von Neumann are compared to the newly developed topologies based on game theoretic approaches.

P-PSO

First, the convergence rate for P-PSO in Sphere Function is analyzed with the help of Figure 6.1 and the according tables in the appendix in Section B.1.1.

The figure shows the convergence rate at each iteration for all neighborhoods. The statistical difference between the different topologies is given by the tables supporting the analysis. For the statistical tests, an interval of 95% was chosen, which means that two neighborhoods are significantly different if the value is smaller than 0.05.
The statistical results show that the convergence rate is almost the same for each neighborhood and there is in most cases not a significant difference between the results. For most of the vector fields, P-PSO is able to find the optimum. However, the results show that for vector field Fall the swarm is struggling in improving its fitness and finding the optimum. In Section 6.2.4 it is shown that the particles struggle for compensating the wind in vector field Fall and do not have enough strength to move in their desired direction. Instead, they are blown away by the flow. This behaviour leads to the bad results in the fitness rate because the particles cannot perform their search process as usual. The results for the other vector fields show that P-PSO is able to find the optimum in vector fields with less influence. This leads to the conclusion that for P-PSO the neighborhood selection does not play a major role in the process of finding the optimum for simple optimization problems like Sphere Function. Though, environments with strong wind influence highly complicate and hinder the search process for P-PSO.

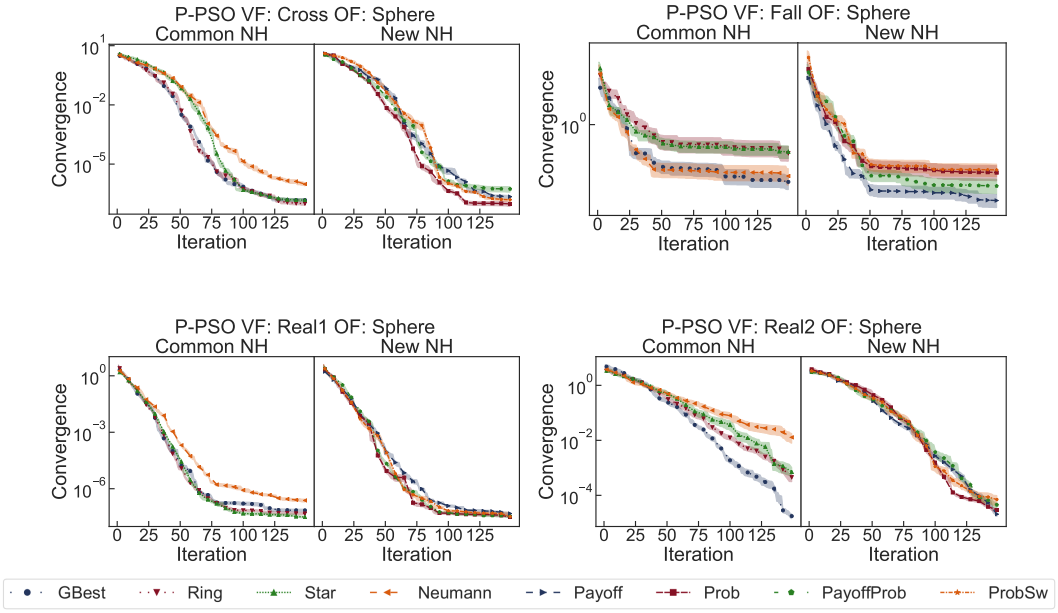


Figure 6.1.: Convergence P-PSO Sphere Function

Figure 6.2 shows the convergence rate for P-PSO in Rosenbrock Function. Similar to Sphere Function, P-PSO is not able to find the optimum. The swarms fitness is even worse than before. This leads to the assumption that optimization problems including plateaus like Rosenbrock Function are more challenging for P-PSO to solve. Though, the results for the different neighborhoods do not statistically vary much as presented in Section B.1.1. Consequently, for P-PSO, the neighborhood selection does not affect the swarms fitness.

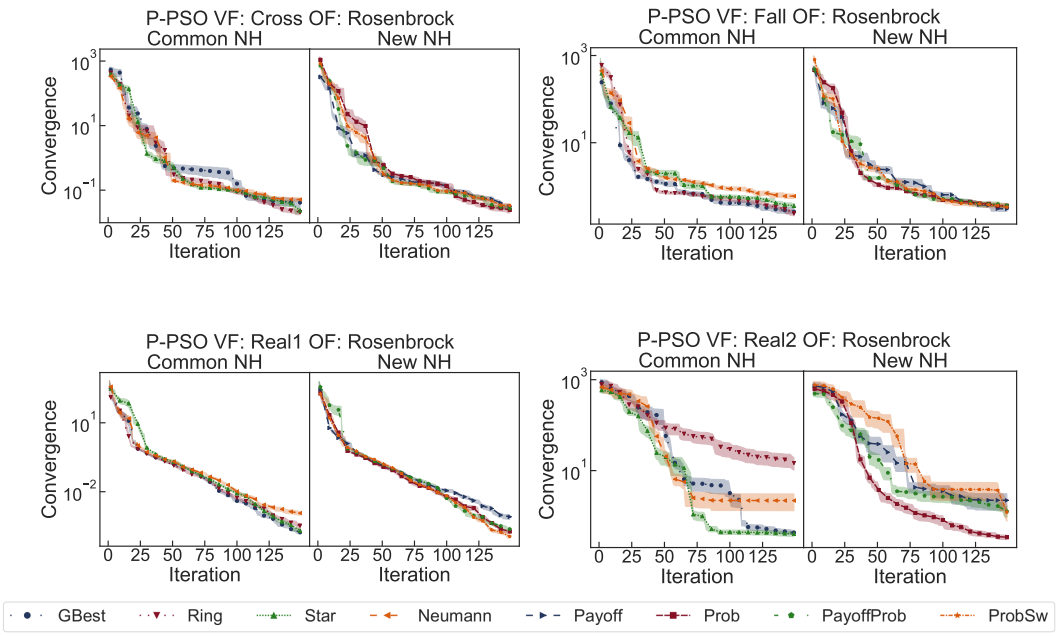


Figure 6.2.: Convergence P-PSO Rosenbrock Function

The results for Ackley Function are presented by Figure 6.3. Finding the optimum is challenging in vector field Fall and Real2 regardless of the chosen topology. The results are not significantly different in almost all cases as shown in Table B.18 and Table B.20. As already written before, Fall generates the worst results due to the strong wind influence which P-PSO is not able to cope with. Real2 shows similar results because the vector field and the wind disturbances are similar to vector field Fall, but the wind is less strong at some positions. Thus P-PSO can compensate the wind at some positions, but vector fields including straight wind flows seem to be challenging for P-PSO. In contrast, the results for Cross and Real1 are better for all topologies even

if no meaningful difference between them is visible.
Overall, the results show that P-PSO in some cases finds the optimal solution for problems with multiple local minima like Ackley Function. However, the results are highly depending on the environmental influence and the neighborhood does not affect the results. It seems P-PSO performs better in vector fields with varying wind directions and worth in more or less linear wind directions.

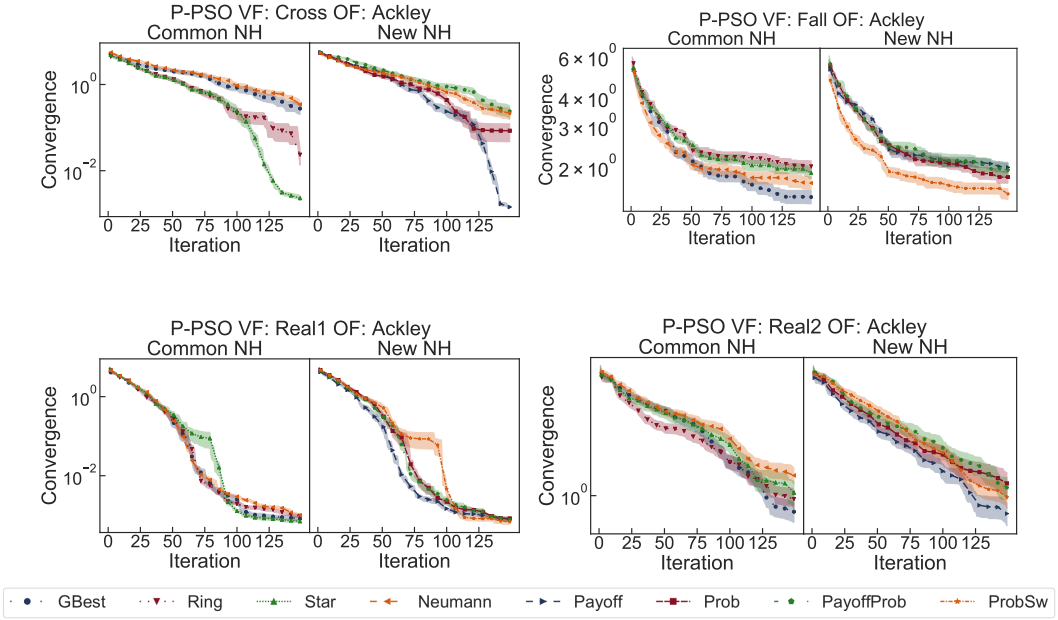


Figure 6.3.: Convergence P-PSO Ackley Function

Z-PSO

This section deals with the convergence analysis of Z-PSO for the three objective functions Sphere, Rosenbrock and Ackley. The statistical analysis values can be found in the appendix. Figure 6.4 shows excellent search results in vector field Cross, Real1 and Real2 for Sphere Function. Though, the convergence rate does not significantly vary between the different neighborhoods (B.1.2). Only the results of Von Neumann and Probability Switch topology are slightly worse than the others. This is due to the fact that cooperation inside these topologies is harder to accomplish. The performance of vector field Fall falls out of line compared to the others, but this vector field is also the most challenging one with respect to the environmental influence. Nevertheless, the different topologies generate almost the same results as shown in Table B.6, with one exception Probability Switch, which is a little worse than the others. The plots lead to the conclusion that Z-PSO can perform better than P-PSO and is able to find the optimum even in difficult environments like Fall. Indeed, the variation of the neighborhoods did not have an impact on the results in Sphere Function.

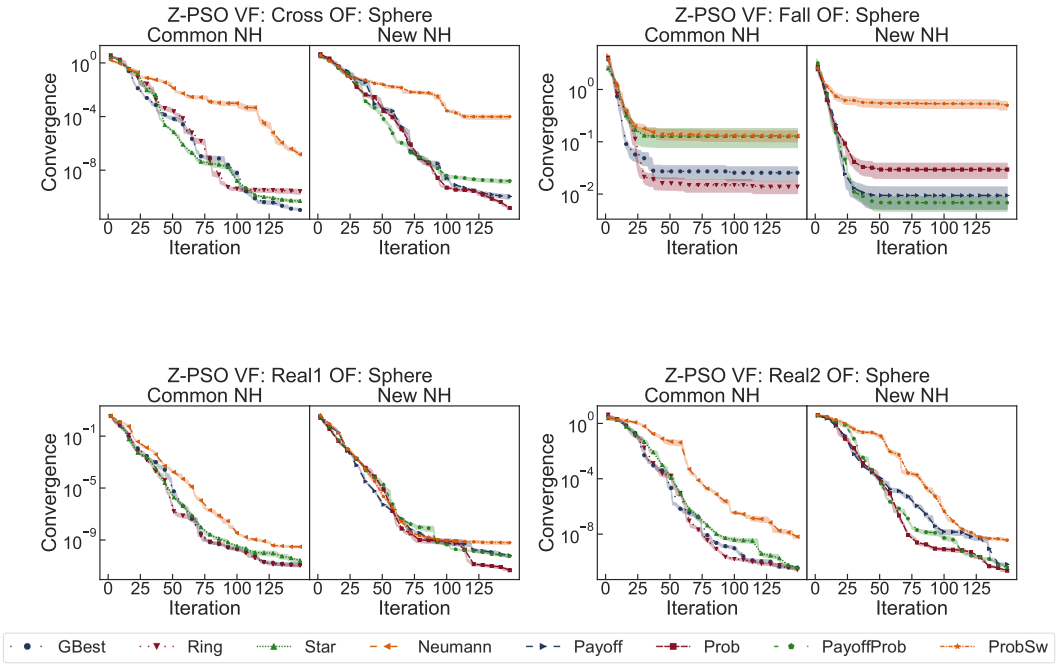


Figure 6.4.: Convergence Z-PSO Sphere Function

Similar to Sphere Function, the results for the different neighborhoods are in most cases not significantly varying (B.2.2). Only Von Neumann and Probability Switch generate different results compared to the others. In general, the fitness is worse than for Sphere Function, which is due to the more complex optimization problem. Though, Z-PSO still performs well in all vector fields and better than P-PSO. In vector field Fall still the worst convergence rate is achieved, but the vector field is also the most challenging one. In summary, Z-PSO is still able to solve also optimization problems including plateaus. Whereby, the results are getting worse, the stronger the vector field gets and the selected topology does not highly influence the search results.

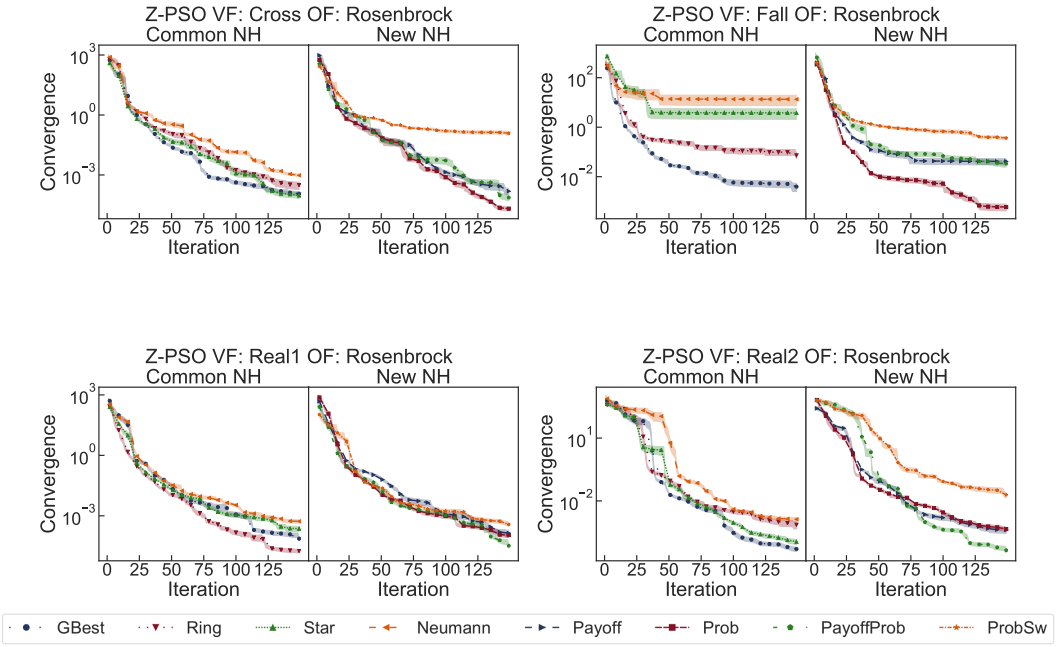


Figure 6.5.: Convergence Z-PSO Rosenbrock Function

The convergence rate for vector field Cross, Real1 and Real2 has become even better in Ackley than in Rosenbrock Function. This leads to the assumption that optimization problems including multiple local minima are easier to solve for Z-PSO than problems consisting of plateaus. Though, the statistical analysis shows that for Cross and Real1 most of the neighborhoods are more or less the same. Whereas, in Real2 especially Von Neumann and Probability Switch vary compared to the others. As already mentioned, these topologies do

not easily create cooperations between the swarm members and consequently, the results are worse than the others. In vector field Fall, the swarm performed worse than in the optimization problems before. However, Payoff Probability is significantly better than the other topologies and can perform very good. The conclusion is drawn that Payoff Probability is very suitable for strong environmental influences like Fall including multiple local minima. In general, Z-PSO performs better than P-PSO and can generate even for Ackley Function good results.

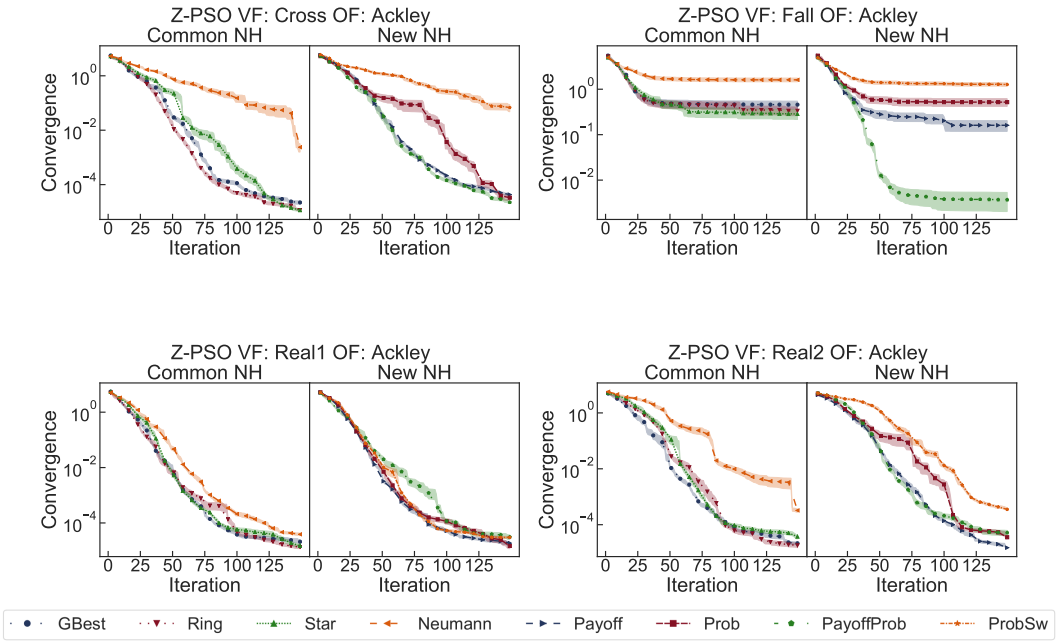
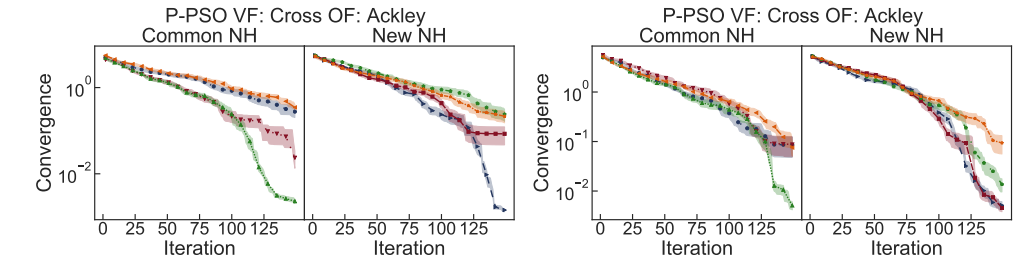


Figure 6.6.: Convergence Z-PSO Ackley Function

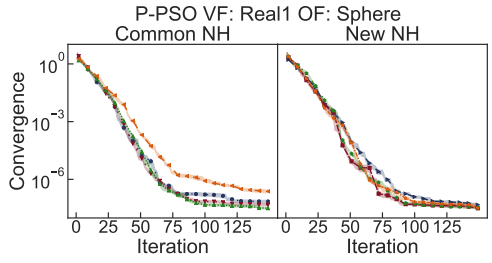
Communication Radius

For further information about the impact of the communication radius on PSO, the convergence rate was as well analysed for a global communication radius of 30. This means that all particles may exchange information with all other particles in the whole search space independent of their distance. This assumption is commonly used in the research field of PSO. However, it is not applicable for real-world robot scenarios. In the following, the convergence rates for radius 2 and radius 30 are compared with each other.
The convergence rates for P-PSO do not significantly vary in most vector fields. The results of the topologies are almost the same for each communication radius. Consequently, the communication radius does not highly influence the fitness of P-PSO. Only plots of vector field Cross show that the convergence rate is slightly better using a common communication radius. Though, in vector field Real1, at least for Sphere Function, P-PSO can gain better results using the limited communication radius.

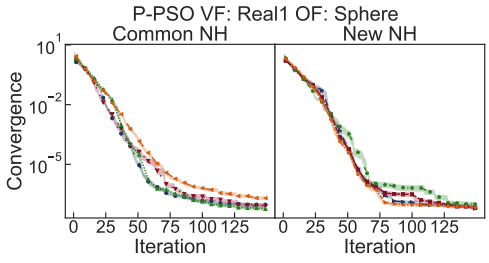


Cross Radius 2

Cross Radius 30



Fall Radius 2



Fall Radius 30

Figure 6.7.: Convergence P-PSO Radius 2 and Radius 30

The convergence rate for Z-PSO using a limited communication radius was in all vector fields at least as good as the rate for the unlimited communication radius or even better. Especially, in strong vector fields as Fall, the small communication radius could generate better results for the uncommon topologies. Consequently, Z-PSO is a promising approach for applications including a limited communication radius. Furthermore, it shows that topologies which are created based on the fitness values of the particles can improve the fitness.

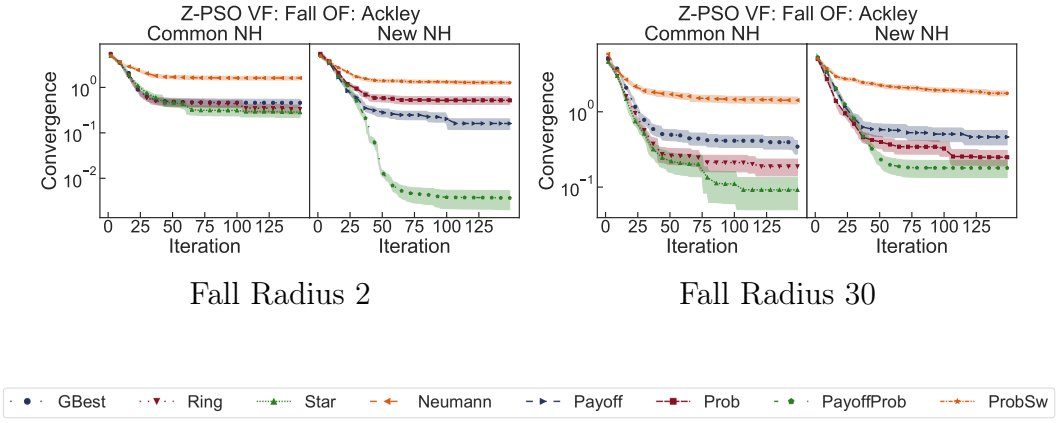


Figure 6.8.: Convergence Z-PSO Radius 2 and Radius 30

6.2.2. Energy Analysis

This section analyzes the energy consumption for the newly developed neighborhoods and compares PPSO and ZPSO with each other. The following plots show the average energy usage of a swarm member in each iteration. The corresponding energy calculation is described in Section 4.2. It takes into account the particles velocity and wind force as well as the orientation of the particle in relation to the wind direction. In this work, the assumption is made that a particle spends energy for each movement. Thereby, it is irrelevant if this move is in the same direction of the flow or not. However, the energy calculation generates higher values for movements in the opposite direction of the wind vectors and lower values for movements in the same angle.
Since the results for the three objective functions do not significantly vary, only the results of Sphere Function are regarded in this Section. Nevertheless, all results can be found in the appendix Section C.2.
The results show that the neighborhood selection for P-PSO does not affect the energy consumption since it is almost the same for each neighborhood in each vector field. However, the energy value varies for the vector fields. Vector field Fall generates the highest energy consumption since it is the most strongest and challenging vector field. The particles try to reach the optimum. However, they are not able to compensate the wind though they are spending a lot of energy. The Vector Field Cross and Real2 show little lower values because the wind is less strong in some places and so the particles do need less energy in those areas. Vector field Real1 seems to be the most pleasing one of all vector fields as the particles spend the least energy. Overall, the results show a big difference compared to Z-PSO. P-PSO does use a lot more energy independent of the neighborhood selection.
In contrast Z-PSO is able to reduce the energy usage. Whereby, the results for the neighborhoods do not vary much except for the Probability Switch topology. The values for this topology are significantly lower than those of the others because less particles are communicating with each other. As a consequence, more particles do not have any neighbors and are following the flow as it is designed by the algorithm. This causes a lower energy usage, however, the convergence and success rate suffer. In vector field Fall, the difference between Probability Switch and the other neighborhoods is most evident. In Probability Switch, nearly no particle has found a neighbor and therefore follows the flow. In general, Fall shows that less particles are connected with each other also for the other neighborhoods because the energy

usage almost stagnates. In the other vector fields, the energy usage increases over time because more particles get connected with each other and find the optimum which requires higher energy usage to stay in the area of the best found solution.

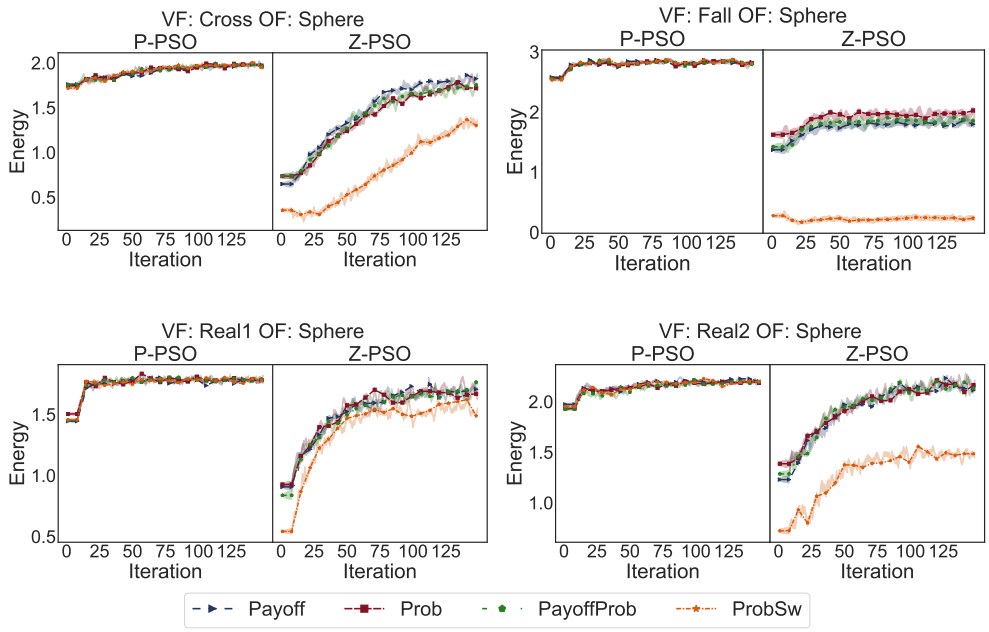


Figure 6.9.: Energy P-PSO and Z-PSO Sphere Function

6.2.3. Accumulation and Success Analysis

This section is about the analysis of the accumulation and success rate of the different neighborhoods for each swarm. Therefore, the accumulation rate shall reveal the percentage of particles accumulating at the optimum. It counts the number of particles of a swarm which are in a radius of two around the optimum. If the radius is set too small, too less particles are considered since the collision avoidance hinders the particles of overlapping. However, if the radius is too big, too many particles may be considered as false positives. The success rate indicates the percentage of simulations in which the optimum has been found. Since the results do not differ much between the functions, only the results for Rosenbrock Function are reviewed exemplary. Though, all results can be found in Section C.3.

P-PSO

The accumulation rate as well the success rate for P-PSO is a lot lower than for Z-PSO. The highest success rates could be reached in vector field Real1, which was already indicated as the most pleasing vector field in the previous section. In a vector field which has a smaller wind influence, P-PSO is able to find the optimum. However, environments including strong wind vectors make it hard for P-PSO to reach the optimum. Especially vector field Fall, which provides the strongest wind impact, generates the worst results. In Cross, the results are relatively good, though the vector field is strong as well. This might be due to the fact that the wind flow at the optimums position is less than in Real2. Overall, the success rate is higher if the accumulation rate is higher. This is due to the effect that more particles close to the optimum have a higher chance of finding it. However, the accumulation rate is not in direct proportion to the success rate. Comparing the neighborhoods with each other, there is not a significant difference. In general, the results are highly depending on the underlying vector field.

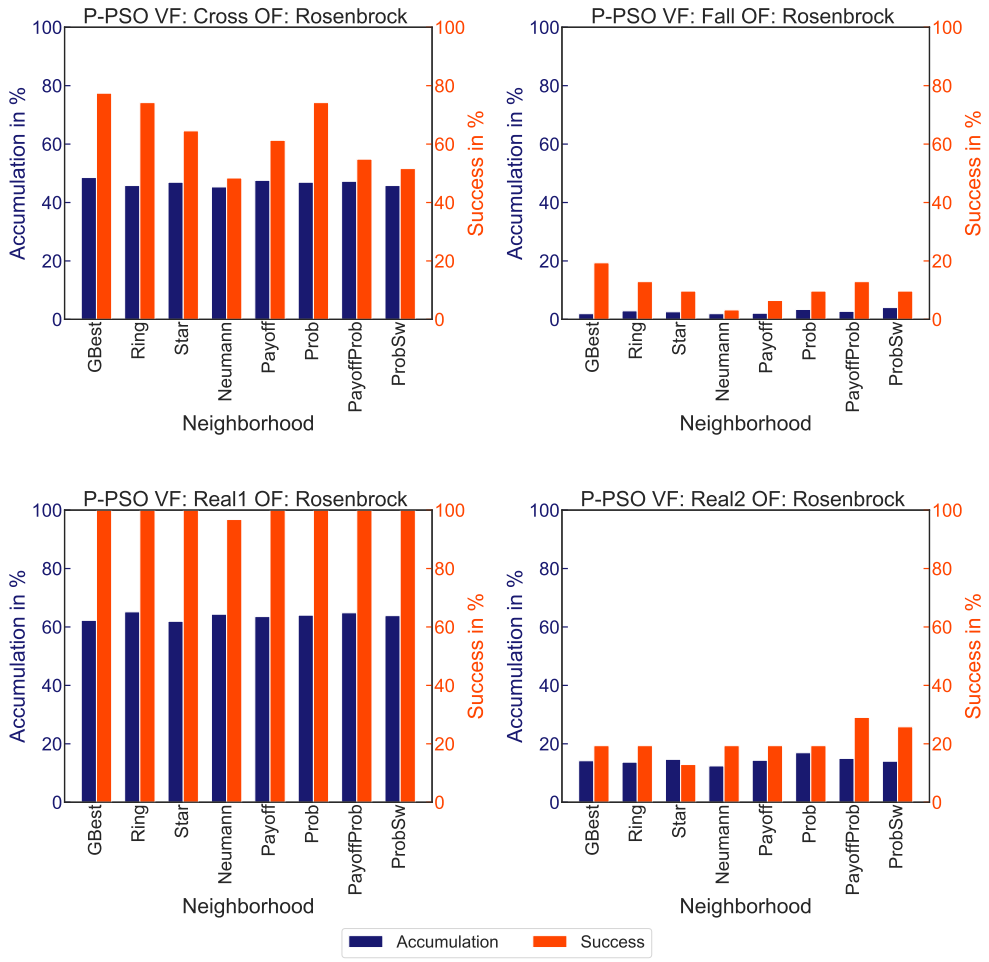


Figure 6.10.: Accumulation P-PSO Rosenbrock Function

Z-PSO

In contrast to P-PSO, Z-PSO reaches excellent success rates for almost all vector fields. Even for vector field Fall, the results are a lot better and for nearly all neighborhoods, the approach is successful in more than half of the simulations. In general, Ring, Neumann and Probability Switch topology show the worst results of all neighborhoods. A direct relationship between success and accumulation rate can be seen in each vector field. The differences may be bigger or smaller, but comparing the neighborhoods with each other the ratio stays the same. It becomes clear that for Z-PSO the accumulation is directly influencing the success rate. While for P-PSO it was only a general indicator for the quality of the results, for Z-PSO it directly maps the difference between the

neighborhoods. Consequently, the accumulation plays a bigger role for Z-PSO. In summary, the wind forecast enables Z-PSO on the one hand to accumulate at the optimum and on the other hand, to reach a high success rate.

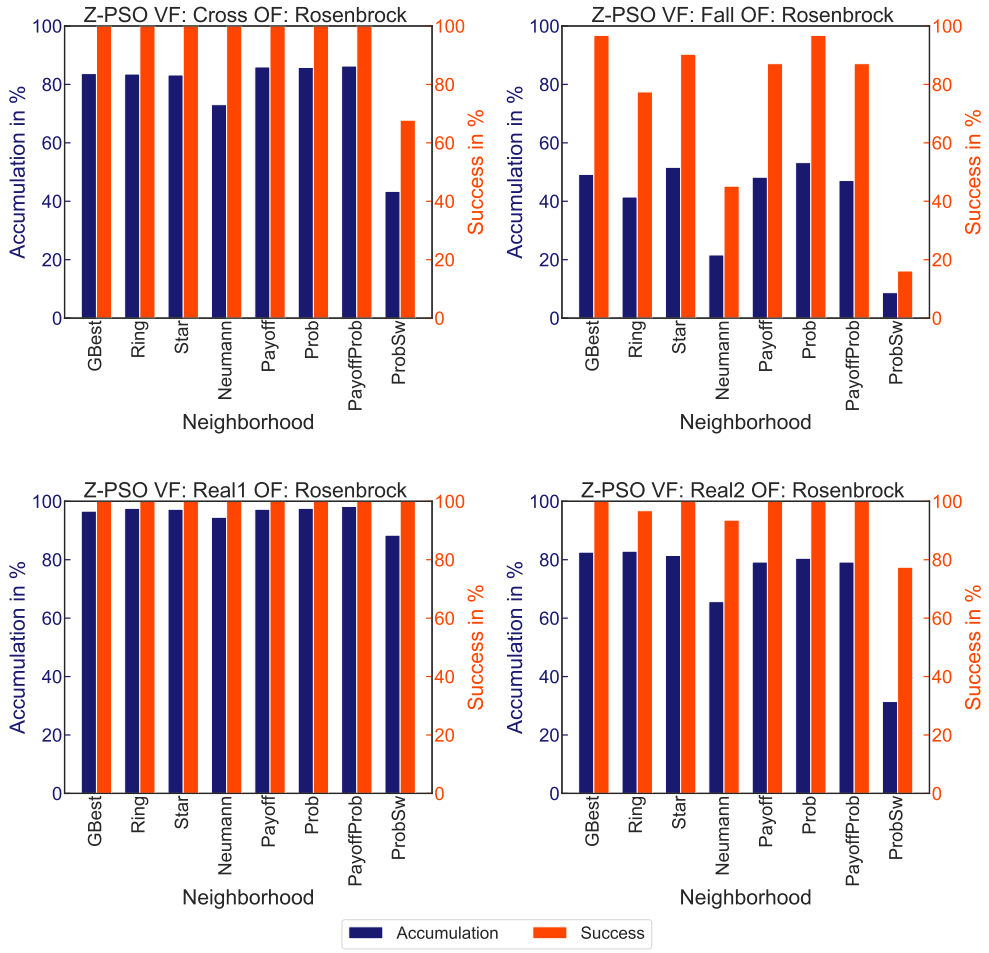


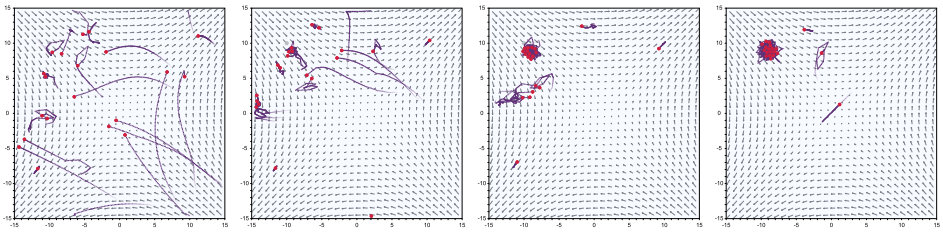
Figure 6.11.: Accumulation Z-PSO Rosenbrock Function

6.2.4. Agents Trajectory Analysis

The PSO approaches presented in Section 4.3 apply different strategies for compensating the wind influence and trying to reach the global optimum. The differences become visible by visualizing the agents trajectories. This section analyzes the trajectories and explains the general particles movement for each approach. Since the results have shown that overall the neighborhoods do not highly influence the search results, only GBest topology is used as an example. In the following, the most divergent trajectories for Sphere Function are presented. Nevertheless, the swarms behaviour for all objective functions can be found in the appendix in Chapter D. Sphere Function has been chosen because P-PSO reached the best success rate in this function. Consequently, the two approaches can be better compared if both of them can reach the most successful results. The plots displayed in this section show the agents trajectories at iterations 10, 25, 50 and 100.

Vector Field Cross

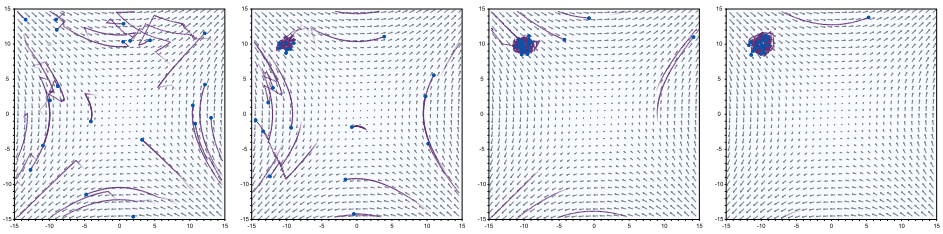
The plots in Figure 6.12 show that P-PSO does not consider the wind orientation and is moving in the direction that the pso vector pretends. This also means moving in the opposite direction of the wind vector and spending more energy. In iteration 25, it is visible that some particles have found neighbors and are cooperating. Though, there are still small groups of interacting particles. With time, the particles are able of accumulating near the global optimum in iteration 50. In iteration 100, nearly all particles have accumulated. However, the figure shows that the particles are crowding at the edge of the optimum and it is not in the center of the swarm. This is due to the fact that the particles are struggling compensating the wind. It seems that the wind at the optimums position is equivalent to the minimal move vector of the P-PSO swarm. Consequently, the particles just reach the optimum but they are at the maximum of their possibilities. Nevertheless, the figure shows that P-PSO is able to reach the optimum.



Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure 6.12.: Trajectory P-PSO OF: Sphere Function VF: Cross NH: GBest

In contrast, Z-PSO follows the approach of reducing the energy usage and following the flow if a particle does not interact with any other swarm member (Figure 6.13). In iteration 10, this behaviour becomes obvious since almost all particles are following the flow. In iteration 25, some particles already accumulate at the global optimum, which is a lot early than for P-PSO. Additionally, in iterations 10 and 25, the particles which are not following the vector field, are doing zigzag movements instead of moving straight in the opposite direction of the wind vector. In iterations 50 and 100, nearly all particles have reached the optimum. A main advantage of the vector field Cross is that the particles which are not compensating the wind influence, are blown towards the other particles with time. If this is not the case, for example, in vector field Fall, it may be the case that the particles keep separate and never find any other neighbors. As a result, the particles keep on following the flow for ever and never reach the optimum. In comparison to P-PSO, the optimum is in the center of the swarm in iteration 100. Consequently, the results are also better than for P-PSO.

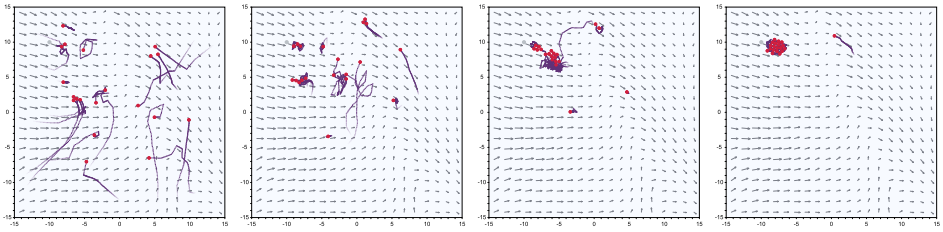


Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure 6.13.: Trajectory Z-PSO OF: Sphere Function VF: Cross NH: GBest

Vector Field Real2

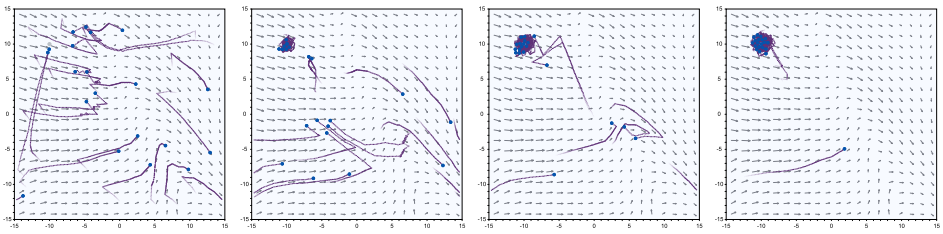
In Figure 6.14, the P-PSO particles are indeed moving in the opposite direction of the wind vector. Similar to vector field Cross, the particles are forming small groups in iteration 25. Those groups are accumulating close to the optimum in iterations 50 and 100. However, the plots as well show that the swarm is close to the optimum but it does not reach it. The wind strength at this position is hard for P-PSO to compensate. Consequently, it is evident that the results for more challenging objective functions are worse.



Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure 6.14.: Trajectory P-PSO OF: Sphere Function VF: Real2 NH: GBest

In Figure 6.15, the particles of Z-PSO are mostly following the flow. However, some particles are already moving in the direction of the optimum. Though, those particles are doing zigzag movements with respect to the wind orientation. Moreover, the particles in Z-PSO do not form small groups, but the particles are again faster accumulating at the optimum. In contrast to P-PSO, the particles do not have any problems reaching the optimum and make it the center of the swarm. Consequently, Z-PSO reaches excellent results in all objective functions.



Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure 6.15.: Trajectory Z-PSO OF: Sphere Function VF: Real2 NH: GBest

6.3. Summary

The convergence analysis has shown that for both PSO approaches, the topologies in most cases did not significantly vary. This reveals that the topology selection does not highly influence the convergence rate. In Ackley Function were the biggest differences between the neighborhoods, which brings up the hypotheses that the topology selection has a bigger influence on complex optimization problems. A possible reason for the equality of the topologies may be the fact that the search space was designed too small. In a bigger search space, the topology selection might have a greater meaning because the particles are more diffused in the search space and not get into touch so easily. The comparison of the communication radiuses has shown that the convergence rate of P-PSO is in most cases not influenced. However, Z-PSO generates a better rate using a smaller communication radius. Additionally, the results of the newly developed topologies improve in strong vector fields as Fall and beat the common neighborhoods. This shows that a targeted neighbor selection may meaningfully improve the convergence rate.
The analysis has also revealed that the model of Z-PSO is a good approach for unknown environments since it gained excellent results in all objective functions and vector fields. Even in the challenging vector field Fall, the swarm did relatively well in terms of fitness. In general, the convergence rate of Z-PSO was a lot better than the rate of P-PSO. It was difficult for P-PSO to compensate the wind in strong vector fields, especially for complex optimization problems like Rosenbrock and Ackley Function.
The energy analysis revealed that again the neighborhood selection did not influence the energy consumption. Only for Z-PSO the energy usage varied by the neighborhood because the Probability Switch topology caused less connections between the particles. As a consequence, more particles followed the flow and did not use any energy. In general, P-PSO used approximately the same energy in each iteration. Though, the energy usage did vary between the vector fields. It is clearly visible that those vector fields which caused a worse fitness also caused the highest energy supply. Therefore, it is nearby that for those vector fields, P-PSO did not overcome the wind influence. For each vector field, Z-PSO used less energy than P-PSO. If the particles accumulate at the optimum, the swarm has the highest energy consumption since the particles try to stay at their position and need to fight the wind disturbances. The success and accumulation rates for P-PSO were a lot lower than for Z-PSO.

Nevertheless, in vector field Real1, P-PSO was as good as Z-PSO. This makes the appearance that P-PSO is able to perform as well as Z-PSO if the vector field is not too strong. The other way around, the minimal velocity vector of P-PSO may have been too small in some vector fields. There is the possibility that the results would have been better, if the velocity vector would have been set stronger for compensating the vector field. The success rates for the different neighborhoods vary slightly, which shows that there is a difference between the neighborhood selection. Whereby, the difference is not big and could have been more clearly in a bigger search space. Z-PSO reached excellent results in almost all vector fields. Only vector field Fall has slightly worse success and accumulation rates. This is due to the fact that the vector field makes it hard for the particles to meet each other. If they follow the flow at the beginning of the simulation, because they do not have neighbors, there is only a small chance that they might come in touch with other swarm members. Especially neighborhoods like Von Neumann and Probability Switch which offer a lower probability for connecting with other particles show bad results. In contrast to P-PSO, Z-PSO shows a direct relation between the accumulation rate and the success rate. Consequently, the number of particles accumulating at the optimal solution directly influences the success rate. For P-PSO, such proportions have not been visible.
The agent trajectory for P-PSO showed that the swarm is first forming little sub-swarms before accumulating near the optimum. Furthermore, the optimum does not become the center of the swarm, which causes a worse fitness especially in complex optimization problems. Nevertheless, the swarm is able to compensate the wind and reach the optimum. The trajectories for Z-PSO showed that the particles did follow the flow if they do not have any neighbor. Additionally, they did zigzag movements as soon as they were moving in the contrary direction of the wind orientation. Furthermore, the particles did earlier accumulate at the optimum and the swarm center was at the position of the optimum. As a result, the swarm did improve its fitness compared to P-PSO.

7. Conclusion and Future Work

This chapter summarizes the results and insights of this work in Section 7.1. Furthermore, Section 7.2 provides possibilities and suggestions for future work.

7.1. Conclusion

The goal of this work was to analyze the impact of topologies and a limited communication radius on search mechanisms in unknown dynamic environments in robotic applications. First, a model for the particles movement has been implemented using Processing[1] including collision avoidance and continuous movement to generate realistic simulations. Moreover, an energy model has been created for measuring the energy consumption of a swarm during the search process. Furthermore, the dynamic environment has been represented by vector fields. Therefore, two vector fields have been calculated by functions and two vector fields have been created by real-world data provided by NASA[2].
For the experiments, two new PSO approaches have been developed, which are able to handle unknown environmental influences. P-PSO overcomes the wind disturbances by constantly using a minimal velocity vector. In contrast, Z-PSO tries to predict the wind vector at its current position for taking countermeasures. Moreover, Z-PSO tries to reduce the energy usage by preventing movements in the contrary direction of the wind orientation. Instead it tries not to move straight against the wind, but do zigzag movements similar to sail boats. The PSO approaches were analyzed applying a set of different topologies. This set included the common topologies GBest, Ring, Star, and Von Neumman and newly developed neighborhoods based on game theoretic approaches. The goal was to create neighborhoods which are formed

[1] `https://processing.org/`
[2] `https://www.nasa.gov/`

on the base of the particles fitness.
Since Z-PSO aims to reduce the energy, a simple energy model was designed for the analysis. It considers the particles velocity, wind strength, and angle between the particles velocity vector and wind vector. The PSO approaches and neighborhoods have been tested in four vector fields and three objective functions. They included Sphere Function, Rosenbrock Function and Ackley Function. Furthermore, the communication radius of the particles was limited to two. The results of the experiments were analysed in terms of convergence rate, energy consumption, accumulation rate, and success rate. Additionally, the general particle trajectory has been regarded.

The results of the convergence analysis have shown that the neighborhood selection in most cases did not significantly influence both swarms. Though, in vector fields with strong influences, like Fall, Payoff Probability topology could gain better results than all other neighborhoods. Furthermore, it turned out that Z-PSO can generate excellent results in almost all vector fields. Additionally, Z-PSO is a good choice for scenarios including a limited communication radius because the results improved compared to an unlimited radius. In contrast, P-PSO sometimes did struggle to compensate the vector field. Though, a larger minimal velocity vector for P-PSO might solve the problem. In terms of energy consumption, the neighborhood selection did not show any relevant difference. Z-PSO outperformed P-PSO in all vector fields and was able to reduce the average energy usage. The success rates for Z-PSO support the results from the convergence analysis and show that Z-PSO is an excellent approach for dealing with unknown environments. Besides, P-PSO could reach nearly the same success rates as Z-PSO in less challenging vector fields, which maintains the thesis that a higher minimal velocity vector might improve the results of challenging vector fields.
In summary, newly developed neighborhoods can gain better results especially in strong vector fields. This shows that a targeted neighborhood selection can gain better results. However, in most test scenarios, the neighborhoods did not significantly vary. The results of the topologies might become more divergent in a bigger search space. P-PSO is a good approach for dealing with unknown environments, but the minimal velocity vector needs to be adapted according to the wind strength. Z-PSO was able to gain excellent results and additionally reduce the energy consumption. Therefore, it is a very promising approach for dynamic environments. Both approaches offer the possibility for real robotic applications since the experiments did consider

a limited communication radius, collision avoidance and limited energy supply. Based on the results of this work, Z-PSO is preferable since the convergence rate improved by limiting the particles interaction.

7.2. Future Work

This work reveals a lot of potential for future research. The further research possibilities are partitioned into three subcategories. The first category regards the exploration of the developed PSO approaches. The second category presents investigations in the neighborhoods and the last category deals with the general experimental setup.

7.2.1. PSO Approaches

The PSO approaches developed in Section 4.3 offer the opportunity for more experiments. Especially the Z-PSO approach was tested in this thesis under ideal circumstances. The individuals location and thereby calculated wind influence was as exact as possible. However, robotics in real-world scenarios will probably not be able to determine their location as precisely as in this computer simulation. Therefore, it would be interesting to add a random error factor to the location determination to make it more realistic.
Another adaption of the Z-PSO swarm is the behaviour of the particle if they have found the so far best position. In this simulation, the particles are starting accumulating at the optimal position and only one particle is able to search on close distance to this position due to collision avoidance. A new approach could be that particles reaching the so-far best position or entering a predefined radius around this position are randomly pushed away from this optimal position in every second iteration. This would hopefully facilitate the search since the particles would be still moving around the optimum and the accumulation would be contained. Moreover, P-PSO could be adapted by using different minimal velocity values with respect to the vector field velocity. A hybrid approach which dynamically adapts the vector strength could be promising.

Future Work PSO Approaches

- error value for Z-PSO
- random vector at optimum for Z-PSO
- vary minimal velocity vector for P-PSO

7.2.2. Neighborhoods

The neighborhoods presented in this work offer a first insight of the influence of neighborhoods on PSO in vector fields. Though, there is still a lot of research left. This includes the testing of further neighborhood topologies. Additionally, the hybrid approaches from Section 4.4.4 which were based on switching their strategy could be analyzed in detail. Different neighborhoods than Ring could be tested and examined.
Furthermore, future work could test other than the selected parameters of the newly developed topologies. In this work, the parameters have not been selected by significant test but by best practice. Consequently, the performance could be still improved by analysing the parameter setting.

Future Work Neighborhoods

- hybrid neighborhood topologies
- vary Ring neighborhood
- vary neighborhood parameters

7.2.3. Experiments

The experiments done in this thesis offer an insight in the performance of swarms in unknown environments. There is still a high number of customization options for further researches. The list of potential modifications is divided into two parts. One is focused on adaption of the search space and the other one on adjustment of the swarm.
The search space can be further analyzed in the sense of more objective functions and vector fields both real and calculated. Additionally, different limitations of the search space can bring new results. Another interesting adaption would be the modification of the search space into a three-dimensional

search space, which is more realistic, for example, in respect to wind influences. The swarm settings can be adapted by a varying number of particles. It will be interesting to see the difference of smaller or larger swarms particularly in consideration with the neighborhoods. Moreover, the particle size could be varied because it is influencing the particles collision and therefore also the results. Another important factor for the results is the initialization position of the particles. Depending on this position, the influence of the vector field dealing on the particles is also different. Further experiments could be run defining, for example, a spawning area at the bottom of the search space or at the top. In this work, the periodic bound handling technique was used, however there exist a lot more techniques. It can generate new results by varying the technique. At least, the energy calculation can be improved by real physical equations. This would facilitate the transmission of the results to real-world scenarios and robots.

Future Work Experiments

- Search Space
  - more real world vector fields
  - vary limitation of search space
  - other objective functions
  - three dimensional
- Swarm
  - vary particle size
  - vary number of particles
  - vary initialization area
  - other bound handling technique
  - realistic energy calculation

Appendices

A. Vector Fields

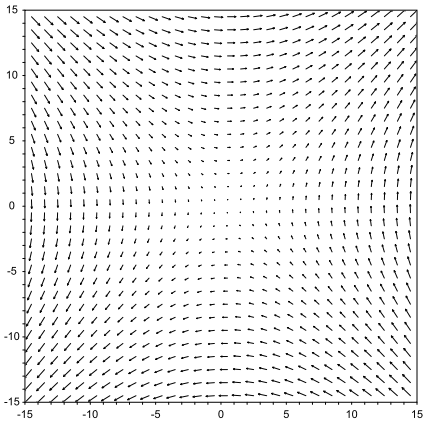


Figure A.1.: VF1 Cross

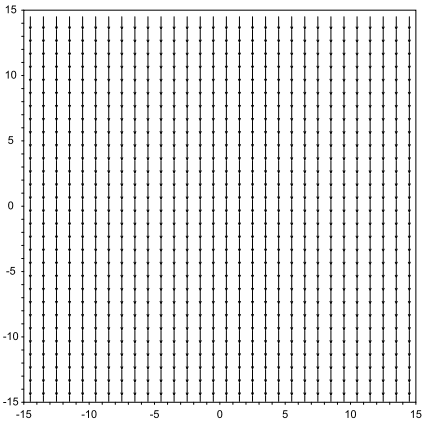


Figure A.2.: VF2 Fall

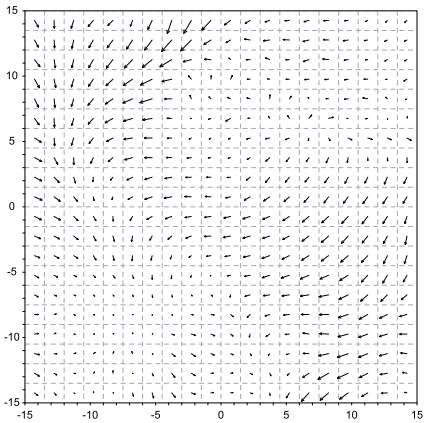


Figure A.3.: VF3 Real1

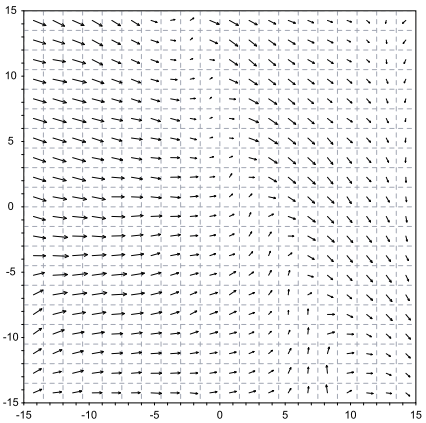


Figure A.4.: VF4 Real2

B. Statistical Data

B.1. Sphere Function

B.1.1. P-PSO

Table B.1.: Convergence P-PSO OF: Sphere Function VF: Cross

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.475658	0.978311	0.0482972	0.47986	0.501014	0.300605	0.972295
Ring	0.475658	nan	0.444516	0.0347577	0.163769	0.956021	0.239038	0.366809
Star	0.978311	0.444516	nan	0.0470139	0.441477	0.481567	0.296231	0.944571
Neumann	0.0482972	0.0347577	0.0470139	nan	0.0728549	0.0348372	0.45405	0.0479433
Payoff	0.47986	0.163769	0.441477	0.0728549	nan	0.189479	0.398166	0.458353
Prob	0.501014	0.956021	0.481567	0.0348372	0.189479	nan	0.237727	0.418274
PayoffProb	0.300605	0.239038	0.296231	0.45405	0.398166	0.237727	nan	0.301213
ProbSw	0.972295	0.366809	0.944571	0.0479433	0.458353	0.418274	0.301213	nan

Table B.2.: Convergence P-PSO OF: Sphere Function VF: Fall

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.250471	0.174843	0.807516	0.406948	0.686875	0.853509	0.563394
Ring	0.250471	nan	0.946876	0.361795	0.09534	0.420568	0.202082	0.459422
Star	0.174843	0.946876	nan	0.282341	0.0497536	0.337347	0.133091	0.36992
Neumann	0.807516	0.361795	0.282341	nan	0.345925	0.88771	0.684575	0.783501
Payoff	0.406948	0.09534	0.0497536	0.345925	nan	0.26097	0.514122	0.159372
Prob	0.686875	0.420568	0.337347	0.88771	0.26097	nan	0.568486	0.90027
PayoffProb	0.853509	0.202082	0.133091	0.684575	0.514122	0.568486	nan	0.444058
ProbSw	0.563394	0.459422	0.36992	0.783501	0.159372	0.90027	0.444058	nan

Table B.3.: Convergence P-PSO OF: Sphere Function VF: Real1

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.421785	0.120898	0.0741544	0.392499	0.151146	0.156603	0.126352
Ring	0.421785	nan	0.310656	0.0394217	0.976138	0.389264	0.415608	0.327629
Star	0.120898	0.310656	nan	0.0242361	0.270143	0.980415	0.847509	0.950188
Neumann	0.0741544	0.0394217	0.0242361	nan	0.0382045	0.0250086	0.0261667	0.0246436
Payoff	0.392499	0.976138	0.270143	0.0382045	nan	0.367929	0.386736	0.287132
Prob	0.151146	0.389264	0.980415	0.0250086	0.367929	nan	0.868676	0.946281
PayoffProb	0.156603	0.415608	0.847509	0.0261667	0.386736	0.868676	nan	0.888298
ProbSw	0.126352	0.327629	0.950188	0.0246436	0.287132	0.946281	0.888298	nan

Table B.4.: Convergence P-PSO OF: Sphere Function VF: Real2

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.168953	0.311527	0.246863	0.775774	0.32851	0.278328	0.402473
Ring	0.168953	nan	0.733796	0.263383	0.171961	0.180623	0.198077	0.234072
Star	0.311527	0.733796	nan	0.273985	0.313686	0.319783	0.331218	0.351352
Neumann	0.246863	0.263383	0.273985	nan	0.246976	0.247293	0.247868	0.248795
Payoff	0.775774	0.171961	0.313686	0.246976	nan	0.474776	0.336778	0.430844
Prob	0.32851	0.180623	0.319783	0.247293	0.474776	nan	0.543479	0.517025
PayoffProb	0.278328	0.198077	0.331218	0.247868	0.336778	0.543479	nan	0.706413
ProbSw	0.402473	0.234072	0.351352	0.248795	0.430844	0.517025	0.706413	nan

B.1.2. Z-PSO

Table B.5.: Convergence Z-PSO OF: Sphere Function VF: Cross

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.278217	0.150499	0.215307	0.270639	0.532744	0.153044	0.209375
Ring	0.278217	nan	0.362359	0.216075	0.528962	0.285926	0.243504	0.209376
Star	0.150499	0.362359	nan	0.215427	0.530972	0.196467	0.163937	0.209375
Neumann	0.215307	0.216075	0.215427	nan	0.2156	0.21532	0.219957	0.21009
Payoff	0.270639	0.528962	0.530972	0.2156	nan	0.291083	0.181923	0.209375
Prob	0.532744	0.285926	0.196467	0.21532	0.291083	nan	0.154126	0.209375
PayoffProb	0.153044	0.243504	0.163937	0.219957	0.181923	0.154126	nan	0.209382
ProbSw	0.209375	0.209376	0.209375	0.21009	0.209375	0.209375	0.209382	nan

Table B.6.: Convergence Z-PSO OF: Sphere Function VF: Fall

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.529535	0.367275	0.140618	0.402912	0.884662	0.284462	0.0265845
Ring	0.529535	nan	0.311398	0.0945797	0.722296	0.488613	0.455112	0.0228787
Star	0.367275	0.311398	nan	0.988981	0.293652	0.388581	0.282099	0.120471
Neumann	0.140618	0.0945797	0.988981	nan	0.0837034	0.162359	0.0753931	0.095965
Payoff	0.402912	0.722296	0.293652	0.0837034	nan	0.385757	0.806823	0.0217517
Prob	0.884662	0.488613	0.388581	0.162359	0.385757	nan	0.295491	0.028077
PayoffProb	0.284462	0.455112	0.282099	0.0753931	0.806823	0.295491	nan	0.0210026
ProbSw	0.0265845	0.0228787	0.120471	0.095965	0.0217517	0.028077	0.0210026	nan

Table B.7.: Convergence Z-PSO OF: Sphere Function VF: Real1

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.886169	0.553127	0.0257074	0.168086	0.370371	0.11548	0.123131
Ring	0.886169	nan	0.492148	0.0248172	0.150333	0.31675	0.0967602	0.122184
Star	0.553127	0.492148	nan	0.034782	0.360618	0.305975	0.34079	0.131715
Neumann	0.0257074	0.0248172	0.034782	nan	0.0739568	0.0217623	0.0646935	0.436681
Payoff	0.168086	0.150333	0.360618	0.0739568	nan	0.102598	0.916796	0.158537
Prob	0.370371	0.31675	0.305975	0.0217623	0.102598	nan	0.0552182	0.118299
PayoffProb	0.11548	0.0967602	0.34079	0.0646935	0.916796	0.0552182	nan	0.154459
ProbSw	0.123131	0.122184	0.131715	0.436681	0.158537	0.118299	0.154459	nan

Table B.8.: Convergence Z-PSO OF: Sphere Function VF: Real2

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.72003	0.956963	0.109506	0.424575	0.557416	0.916387	0.0554712
Ring	0.72003	nan	0.598936	0.10896	0.231501	0.81728	0.60065	0.0548114
Star	0.956963	0.598936	nan	0.109581	0.386445	0.349991	0.94493	0.0555562
Neumann	0.109506	0.10896	0.109581	nan	0.111085	0.108733	0.109674	0.526492
Payoff	0.424575	0.231501	0.386445	0.111085	nan	0.142596	0.4486	0.0573946
Prob	0.557416	0.81728	0.349991	0.108733	0.142596	nan	0.40618	0.0545354
PayoffProb	0.916387	0.60065	0.94493	0.109674	0.4486	0.40618	nan	0.0556704
ProbSw	0.0554712	0.0548114	0.0555562	0.526492	0.0573946	0.0545354	0.0556704	nan

B.2. Rosenbrock Function

B.2.1. P-PSO

Table B.9.: Convergence P-PSO OF: Rosenbrock Function VF: Cross

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.441512	0.517995	0.727991	0.72253	0.540518	0.59044	0.776108
Ring	0.441512	nan	0.853115	0.0729094	0.405139	0.788323	0.595116	0.361123
Star	0.517995	0.853115	nan	0.133597	0.584949	0.945557	0.796299	0.524985
Neumann	0.727991	0.0729094	0.133597	nan	0.268318	0.1429	0.149941	0.330057
Payoff	0.72253	0.405139	0.584949	0.268318	nan	0.626655	0.719779	0.903666
Prob	0.540518	0.788323	0.945557	0.1429	0.626655	nan	0.852056	0.561831
PayoffProb	0.59044	0.595116	0.796299	0.149941	0.719779	0.852056	nan	0.639497
ProbSw	0.776108	0.361123	0.524985	0.330057	0.903666	0.561831	0.639497	nan

Table B.10.: Convergence P-PSO OF: Rosenbrock Function VF: Fall

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.859998	0.439288	0.0668548	0.72875	0.363035	0.419649	0.588302
Ring	0.859998	nan	0.32054	0.0464579	0.584948	0.233918	0.306428	0.469886
Star	0.439288	0.32054	nan	0.189757	0.66276	0.987832	0.960853	0.875201
Neumann	0.0668548	0.0464579	0.189757	nan	0.10705	0.166363	0.205675	0.173597
Payoff	0.72875	0.584948	0.66276	0.10705	nan	0.602578	0.633396	0.814351
Prob	0.363035	0.233918	0.987832	0.166363	0.602578	nan	0.967053	0.849906
PayoffProb	0.419649	0.306428	0.960853	0.205675	0.633396	0.967053	nan	0.842179
ProbSw	0.588302	0.469886	0.875201	0.173597	0.814351	0.849906	0.842179	nan

Table B.11.: Convergence P-PSO OF: Rosenbrock Function VF: Real1

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.411027	0.752494	0.0526482	0.111205	0.855039	0.582581	0.4133
Ring	0.411027	nan	0.512047	0.123763	0.276746	0.454759	0.647506	0.2572
Star	0.752494	0.512047	nan	0.0608203	0.130903	0.875322	0.781782	0.306744
Neumann	0.0526482	0.123763	0.0608203	nan	0.571794	0.0561349	0.0730199	0.0402176
Payoff	0.111205	0.276746	0.130903	0.571794	nan	0.119565	0.160278	0.0816228
Prob	0.855039	0.454759	0.875322	0.0561349	0.119565	nan	0.672164	0.30857
PayoffProb	0.582581	0.647506	0.781782	0.0730199	0.160278	0.672164	nan	0.27133
ProbSw	0.4133	0.2572	0.306744	0.0402176	0.0816228	0.30857	0.27133	nan

Table B.12.: Convergence P-PSO OF: Rosenbrock Function VF: Real2

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.153836	0.857951	0.367252	0.358895	0.510964	0.246623	0.431596
Ring	0.153836	nan	0.153058	0.218546	0.219507	0.151285	0.179978	0.179452
Star	0.857951	0.153058	nan	0.359853	0.351592	0.650387	0.231689	0.415775
Neumann	0.367252	0.218546	0.359853	nan	0.992015	0.343004	0.655358	0.655852
Payoff	0.358895	0.219507	0.351592	0.992015	nan	0.334959	0.645221	0.646268
Prob	0.510964	0.151285	0.650387	0.343004	0.334959	nan	0.197733	0.37974
PayoffProb	0.246623	0.179978	0.231689	0.655358	0.645221	0.197733	nan	0.967886
ProbSw	0.431596	0.179452	0.415775	0.655852	0.646268	0.37974	0.967886	nan

B.2.2. Z-PSO

Table B.13.: Convergence Z-PSO OF: Rosenbrock Function VF: Cross

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.478932	0.798693	0.00687746	0.776486	0.103378	0.591168	0.0223244
Ring	0.478932	nan	0.434309	0.108992	0.595456	0.284943	0.390187	0.0225314
Star	0.798693	0.434309	nan	0.00584721	0.663585	0.257622	0.810692	0.0223018
Neumann	0.00687746	0.108992	0.00584721	nan	0.0144556	0.00243842	0.00460861	0.0232291
Payoff	0.776486	0.595456	0.663585	0.0144556	nan	0.280712	0.549693	0.0223646
Prob	0.103378	0.284943	0.257622	0.00243842	0.280712	nan	0.314328	0.0222257
PayoffProb	0.591168	0.390187	0.810692	0.00460861	0.549693	0.314328	nan	0.0222813
ProbSw	0.0223244	0.0225314	0.0223018	0.0232291	0.0223646	0.0222257	0.0222813	nan

Table B.14.: Convergence Z-PSO OF: Rosenbrock Function VF: Fall

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.0815923	0.32122	0.31632	0.150232	0.302115	0.15354	0.00206383
Ring	0.0815923	nan	0.330053	0.318788	0.504914	0.0671865	0.376152	0.0164346
Star	0.32122	0.330053	nan	0.4879	0.326032	0.320793	0.325022	0.369609
Neumann	0.31632	0.318788	0.4879	nan	0.317669	0.316199	0.317387	0.329472
Payoff	0.150232	0.504914	0.326032	0.317669	nan	0.114892	0.811238	0.00658215
Prob	0.302115	0.0671865	0.320793	0.316199	0.114892	nan	0.108719	0.00188062
PayoffProb	0.15354	0.376152	0.325022	0.317387	0.811238	0.108719	nan	0.0050625
ProbSw	0.00206383	0.0164346	0.369609	0.329472	0.00658215	0.00188062	0.0050625	nan

Table B.15.: Convergence Z-PSO OF: Rosenbrock Function VF: Real1

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.089569	0.299576	0.0494901	0.586861	0.566291	0.251245	0.138395
Ring	0.089569	nan	0.151377	0.0269585	0.29646	0.103841	0.429406	0.076616
Star	0.299576	0.151377	nan	0.262446	0.625126	0.448338	0.183558	0.556059
Neumann	0.0494901	0.0269585	0.262446	nan	0.125145	0.0754116	0.0317599	0.596946
Payoff	0.586861	0.29646	0.625126	0.125145	nan	0.826773	0.362564	0.305596
Prob	0.566291	0.103841	0.448338	0.0754116	0.826773	nan	0.183803	0.203809
PayoffProb	0.251245	0.429406	0.183558	0.0317599	0.362564	0.183803	nan	0.0900784
ProbSw	0.138395	0.076616	0.556059	0.596946	0.305596	0.203809	0.0900784	nan

Table B.16.: Convergence Z-PSO OF: Rosenbrock Function VF: Real2

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.260354	0.405731	0.0350889	0.210441	0.100856	0.87011	0.101484
Ring	0.260354	nan	0.30908	0.485805	0.626913	0.66829	0.256309	0.114294
Star	0.405731	0.30908	nan	0.0460244	0.312927	0.17925	0.363017	0.102575
Neumann	0.0350889	0.485805	0.0460244	nan	0.167152	0.177849	0.0342825	0.12592
Payoff	0.210441	0.626913	0.312927	0.167152	nan	0.902853	0.203132	0.107924
Prob	0.100856	0.66829	0.17925	0.177849	0.902853	nan	0.096079	0.108724
PayoffProb	0.87011	0.256309	0.363017	0.0342825	0.203132	0.096079	nan	0.101383
ProbSw	0.101484	0.114294	0.102575	0.12592	0.107924	0.108724	0.101383	nan

B.3. Ackley Function

B.3.1. P-PSO

Table B.17.: Convergence P-PSO OF: Ackley Function VF: Cross

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.131795	0.0998283	0.74315	0.0987815	0.301138	0.874756	0.778275
Ring	0.131795	nan	0.343125	0.0218684	0.323787	0.480674	0.195749	0.180604
Star	0.0998283	0.343125	nan	0.013602	0.203219	0.326025	0.152209	0.13285
Neumann	0.74315	0.0218684	0.013602	nan	0.0133781	0.104583	0.61736	0.503637
Payoff	0.0987815	0.323787	0.203219	0.0133781	nan	0.320982	0.150739	0.131306
Prob	0.301138	0.480674	0.326025	0.104583	0.320982	nan	0.401697	0.424702
PayoffProb	0.874756	0.195749	0.152209	0.61736	0.150739	0.401697	nan	0.91059
ProbSw	0.778275	0.180604	0.13285	0.503637	0.131306	0.424702	0.91059	nan

Table B.18.: Convergence P-PSO OF: Ackley Function VF: Fall

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.111938	0.194938	0.489551	0.103729	0.307384	0.219446	0.869465
Ring	0.111938	nan	0.704244	0.369814	0.953561	0.548029	0.854572	0.121412
Star	0.194938	0.704244	nan	0.573547	0.733118	0.806945	0.883166	0.21602
Neumann	0.489551	0.369814	0.573547	nan	0.374065	0.753895	0.536145	0.557877
Payoff	0.103729	0.953561	0.733118	0.374065	nan	0.564896	0.889218	0.110944
Prob	0.307384	0.548029	0.806945	0.753895	0.564896	nan	0.726138	0.347169
PayoffProb	0.219446	0.854572	0.883166	0.536145	0.889218	0.726138	nan	0.245188
ProbSw	0.869465	0.121412	0.21602	0.557877	0.110944	0.347169	0.245188	nan

Table B.19.: Convergence P-PSO OF: Ackley Function VF: Real1

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.623604	0.622793	0.515977	0.912588	0.96577	0.869443	0.661403
Ring	0.623604	nan	0.286878	0.922212	0.506667	0.610431	0.49885	0.369234
Star	0.622793	0.286878	nan	0.165744	0.647788	0.520401	0.743513	0.956421
Neumann	0.515977	0.922212	0.165744	nan	0.373233	0.479106	0.382458	0.277349
Payoff	0.912588	0.506667	0.647788	0.373233	nan	0.857426	0.942082	0.698092
Prob	0.96577	0.610431	0.520401	0.479106	0.857426	nan	0.815452	0.598279
PayoffProb	0.869443	0.49885	0.743513	0.382458	0.942082	0.815452	nan	0.76248
ProbSw	0.661403	0.369234	0.956421	0.277349	0.698092	0.598279	0.76248	nan

Table B.20.: Convergence P-PSO OF: Ackley Function VF: Real2

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.666939	0.599546	0.195005	0.954111	0.443795	0.477539	0.650076
Ring	0.666939	nan	0.852334	0.372029	0.641083	0.651655	0.71979	0.948035
Star	0.599546	0.852334	nan	0.576807	0.579515	0.807225	0.887245	0.903582
Neumann	0.195005	0.372029	0.576807	nan	0.194559	0.799063	0.677806	0.445895
Payoff	0.954111	0.641083	0.579515	0.194559	nan	0.430873	0.462988	0.626252
Prob	0.443795	0.651655	0.807225	0.799063	0.430873	nan	0.908283	0.705725
PayoffProb	0.477539	0.71979	0.887245	0.677806	0.462988	0.908283	nan	0.779068
ProbSw	0.650076	0.948035	0.903582	0.445895	0.626252	0.705725	0.779068	nan

B.3.2. Z-PSO

Table B.21.: Convergence Z-PSO OF: Ackley Function VF: Cross

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.149779	0.147809	0.185892	0.0604568	0.55194	0.9876	0.131063
Ring	0.149779	nan	0.938967	0.184001	0.000949138	0.223927	0.0698888	0.131006
Star	0.147809	0.938967	nan	0.183951	0.000998057	0.219675	0.0718617	0.131004
Neumann	0.185892	0.184001	0.183951	nan	0.189775	0.18795	0.185915	0.145008
Payoff	0.0604568	0.000949138	0.000998057	0.189775	nan	0.609861	0.0435381	0.131179
Prob	0.55194	0.223927	0.219675	0.18795	0.609861	nan	0.545955	0.131124
PayoffProb	0.9876	0.0698888	0.0718617	0.185915	0.0435381	0.545955	nan	0.131064
ProbSw	0.131063	0.131006	0.131004	0.145008	0.131179	0.131124	0.131064	nan

Table B.22.: Convergence Z-PSO OF: Ackley Function VF: Fall

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.628155	0.487485	0.00390334	0.175997	0.834756	0.0236902	0.0111653
Ring	0.628155	nan	0.820311	0.000941905	0.349074	0.498276	0.041552	0.00194326
Star	0.487485	0.820311	nan	0.000553886	0.488769	0.381842	0.0684102	0.000986523
Neumann	0.00390334	0.000941905	0.000553886	nan	8.2403e-05	0.00778148	8.5173e-06	0.408812
Payoff	0.175997	0.349074	0.488769	8.2403e-05	nan	0.137006	0.0981331	6.41852e-05
Prob	0.834756	0.498276	0.381842	0.00778148	0.137006	nan	0.021634	0.0244431
PayoffProb	0.0236902	0.041552	0.0684102	8.5173e-06	0.0981331	0.021634	nan	1.98095e-06
ProbSw	0.0111653	0.00194326	0.000986523	0.408812	6.41852e-05	0.0244431	1.98095e-06	nan

Table B.23.: Convergence Z-PSO OF: Ackley Function VF: Real1

	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.382882	0.502791	0.125504	0.814581	0.524807	0.505639	0.343053
Ring	0.382882	nan	0.782928	0.00774223	0.480367	0.874148	0.192011	0.0146996
Star	0.502791	0.782928	nan	0.0147507	0.637119	0.957883	0.240525	0.035298
Neumann	0.125504	0.00774223	0.0147507	nan	0.060577	0.0248995	0.612355	0.391967
Payoff	0.814581	0.480367	0.637119	0.060577	nan	0.652799	0.389176	0.177397
Prob	0.524807	0.874148	0.957883	0.0248995	0.652799	nan	0.254238	0.0688713
PayoffProb	0.505639	0.192011	0.240525	0.612355	0.389176	0.254238	nan	0.948318
ProbSw	0.343053	0.0146996	0.035298	0.391967	0.177397	0.0688713	0.948318	nan

Table B.24.: Convergence Z-PSO OF: Ackley Function VF: Real2

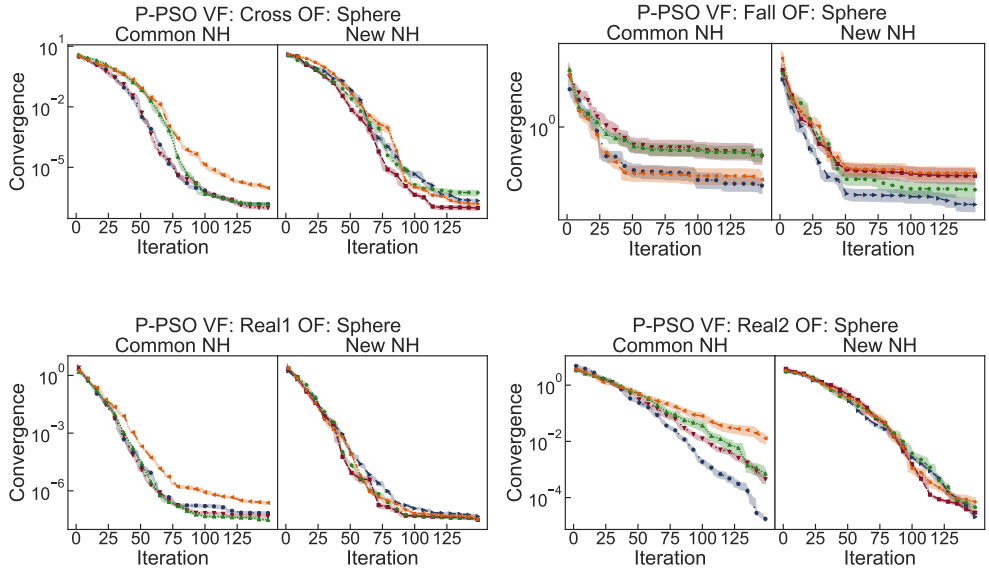
	GBest	Ring	Star	Neumann	Payoff	Prob	PayoffProb	ProbSw
GBest	nan	0.71175	0.290531	0.00382975	0.283629	0.221756	0.21333	0.00052941
Ring	0.71175	nan	0.221491	0.00351133	0.67379	0.16077	0.174742	0.000477387
Star	0.290531	0.221491	nan	0.0063359	0.114736	0.910202	0.599315	0.00101588
Neumann	0.00382975	0.00351133	0.0063359	nan	0.00313848	0.00580117	0.0104661	0.826336
Payoff	0.283629	0.67379	0.114736	0.00313848	nan	0.0477323	0.122514	0.000412096
Prob	0.221756	0.16077	0.910202	0.00580117	0.0477323	nan	0.52384	0.000896326
PayoffProb	0.21333	0.174742	0.599315	0.0104661	0.122514	0.52384	nan	0.00200027
ProbSw	0.00052941	0.000477387	0.00101588	0.826336	0.000412096	0.000896326	0.00200027	nan

C. Plots

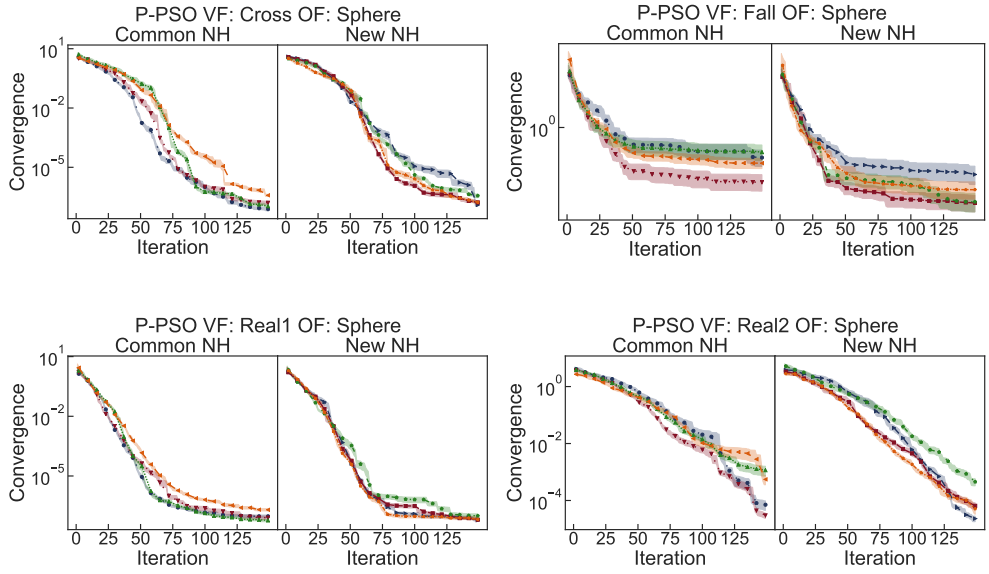
C.1. Convergence Plots

C.1.1. Sphere Function

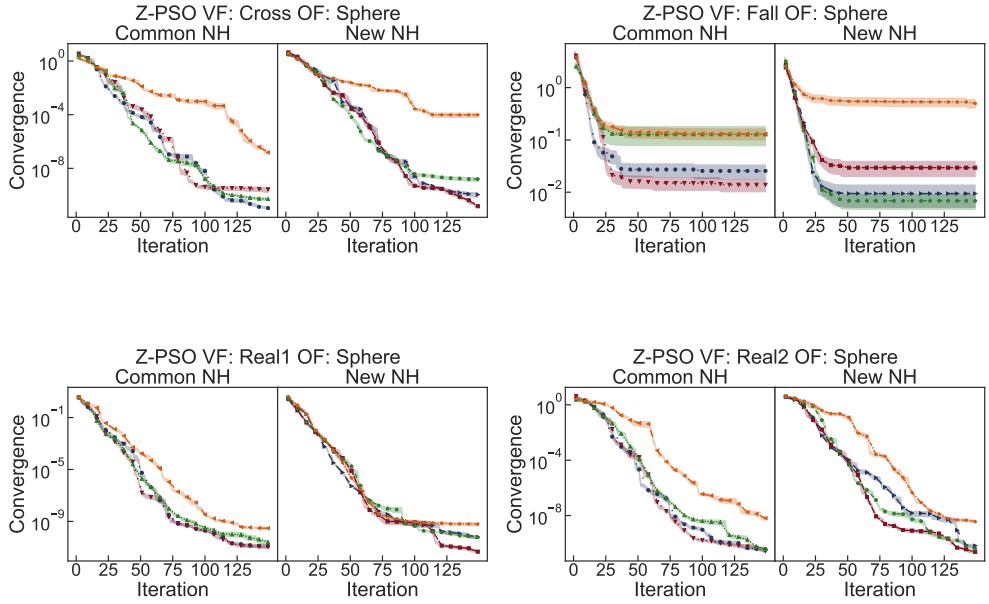
P-PSO Radius 2



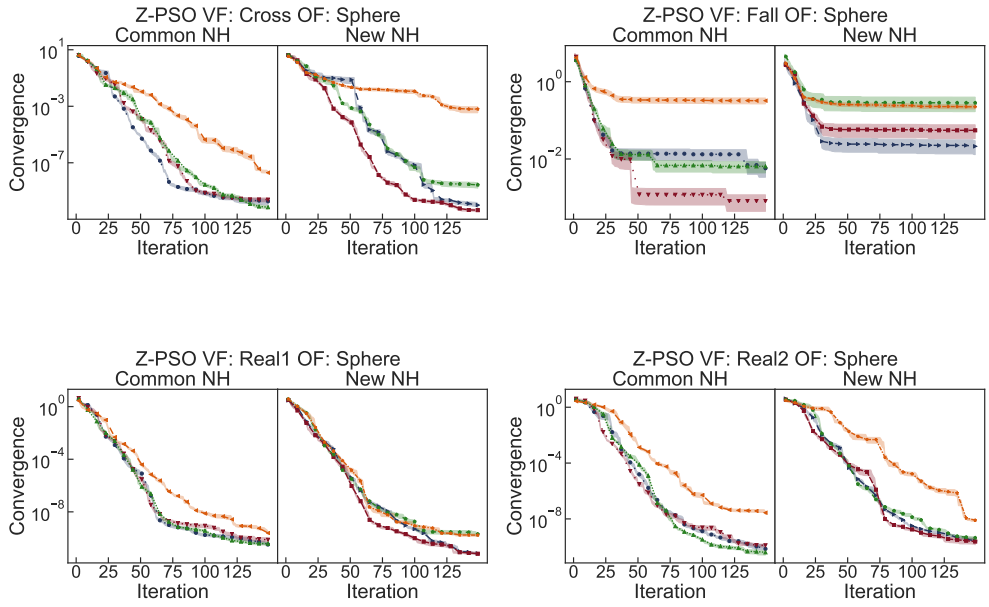
P-PSO Radius 30



Z-PSO Radius 2

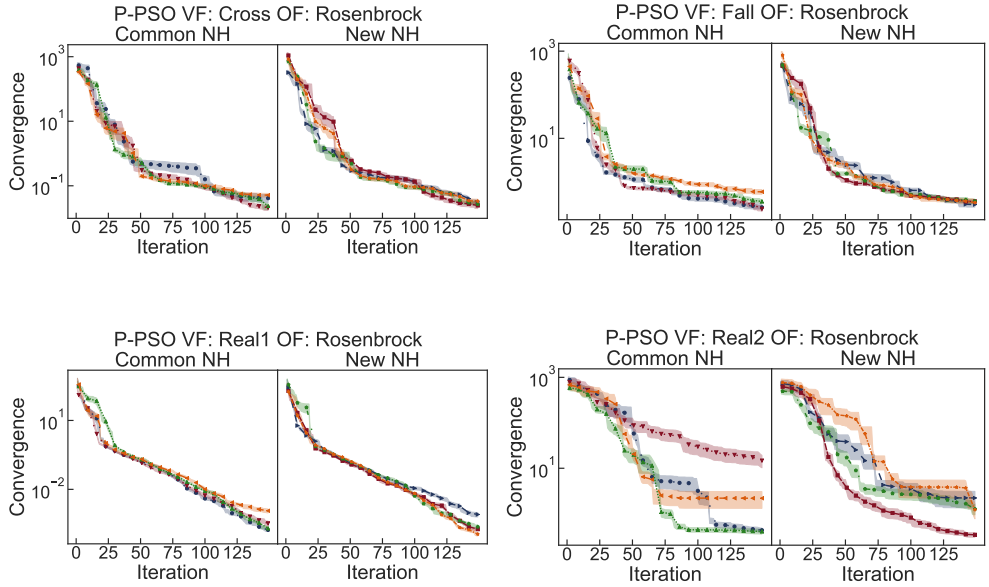


Z-PSO Radius 30

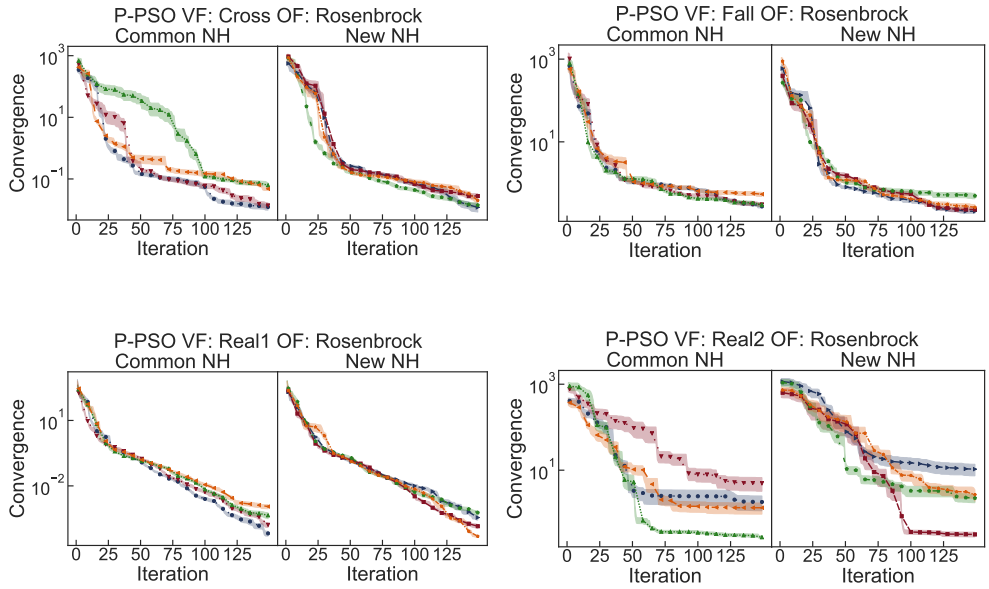


C.1.2. Rosenbrock Function

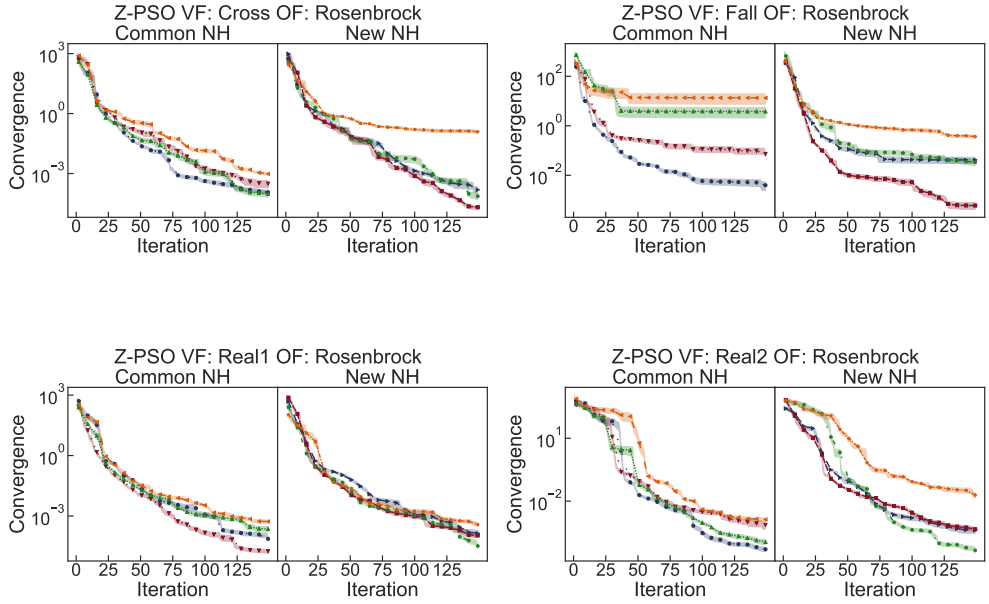
P-PSO Radius 2



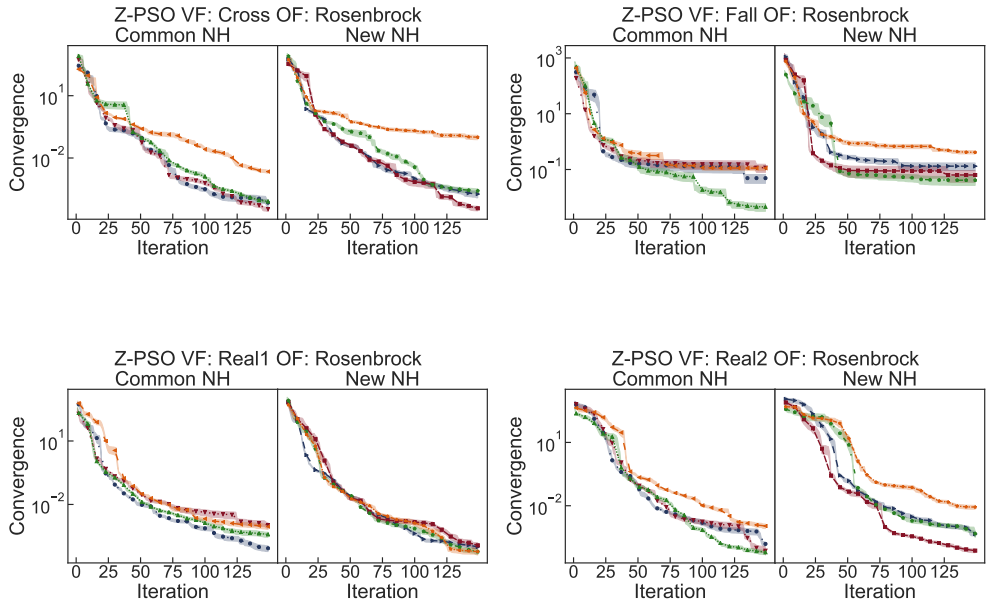
P-PSO Radius 30



Z-PSO Radius 2

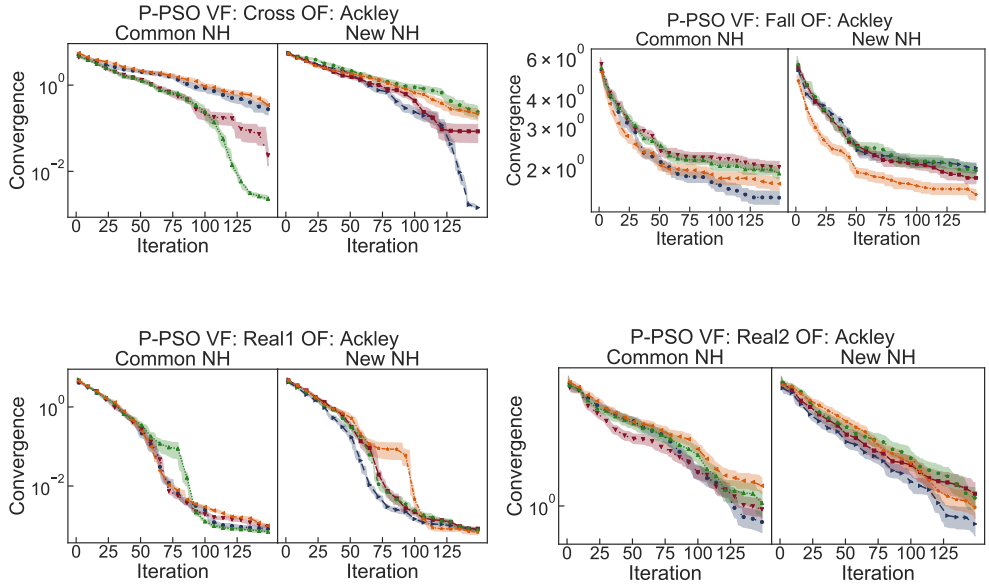


Z-PSO Radius 30

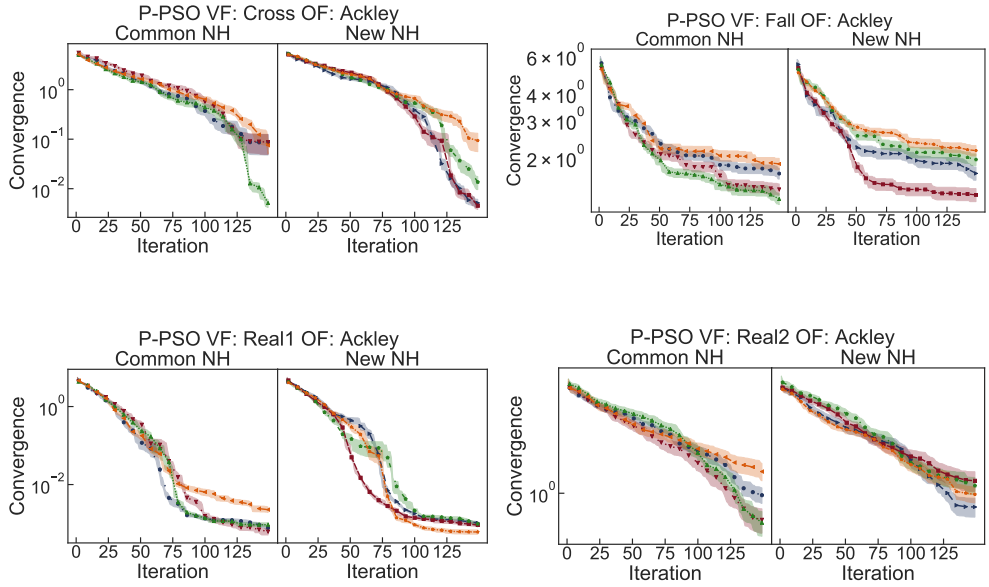


C.1.3. Ackley Function

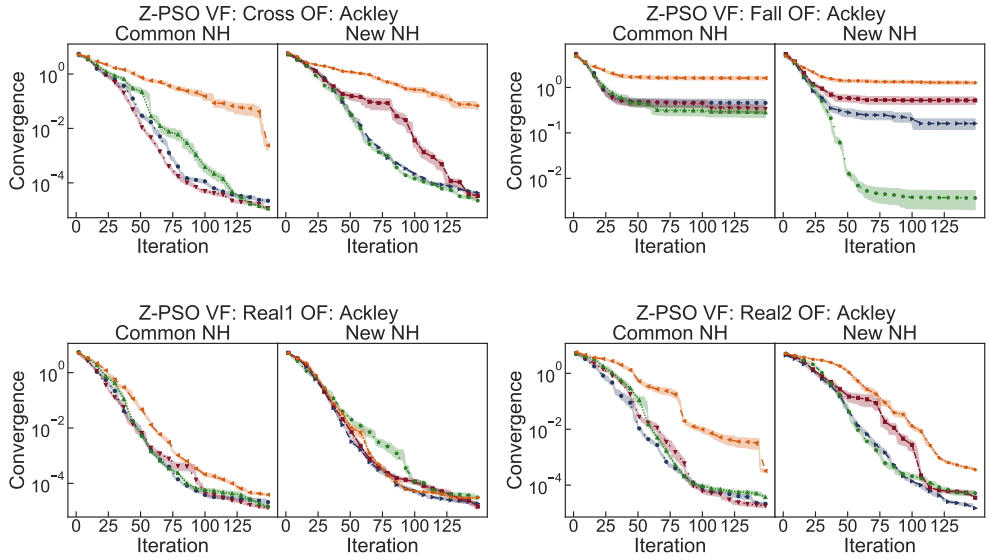
P-PSO Radius 2



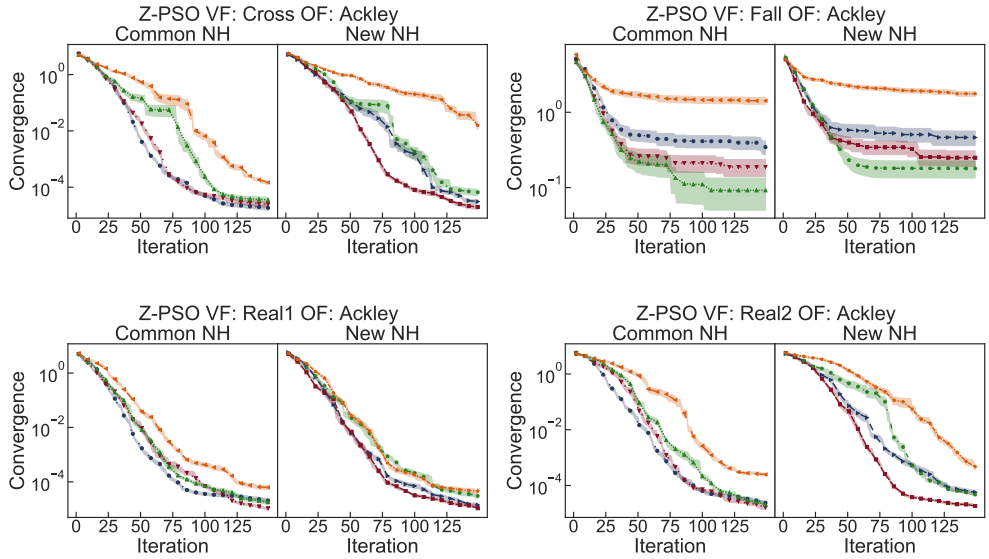
P-PSO Radius 30



Z-PSO Radius 2

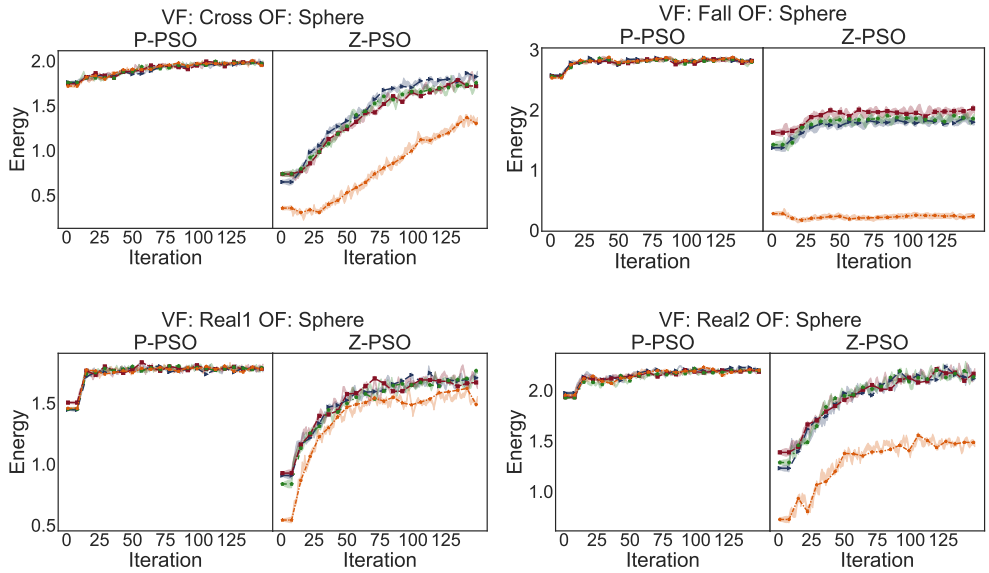


Z-PSO Radius 30

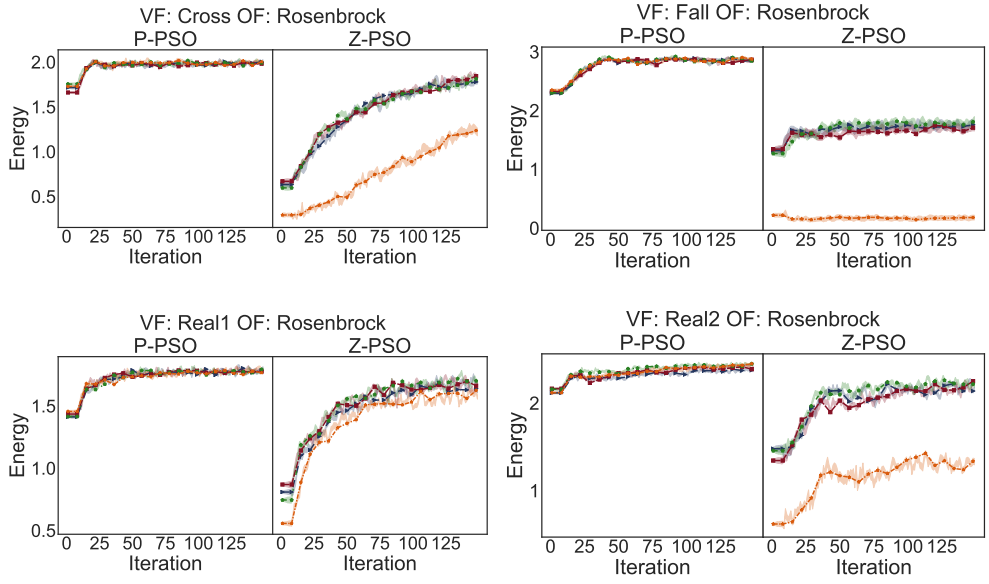


C.2. Energy Plots

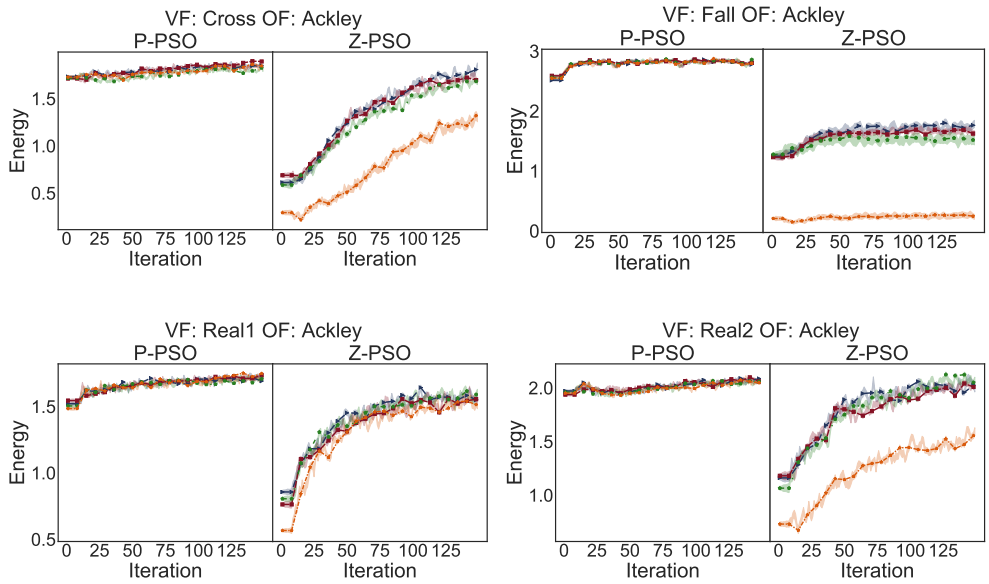
C.2.1. Sphere Function



C.2.2. Rosenbrock Function



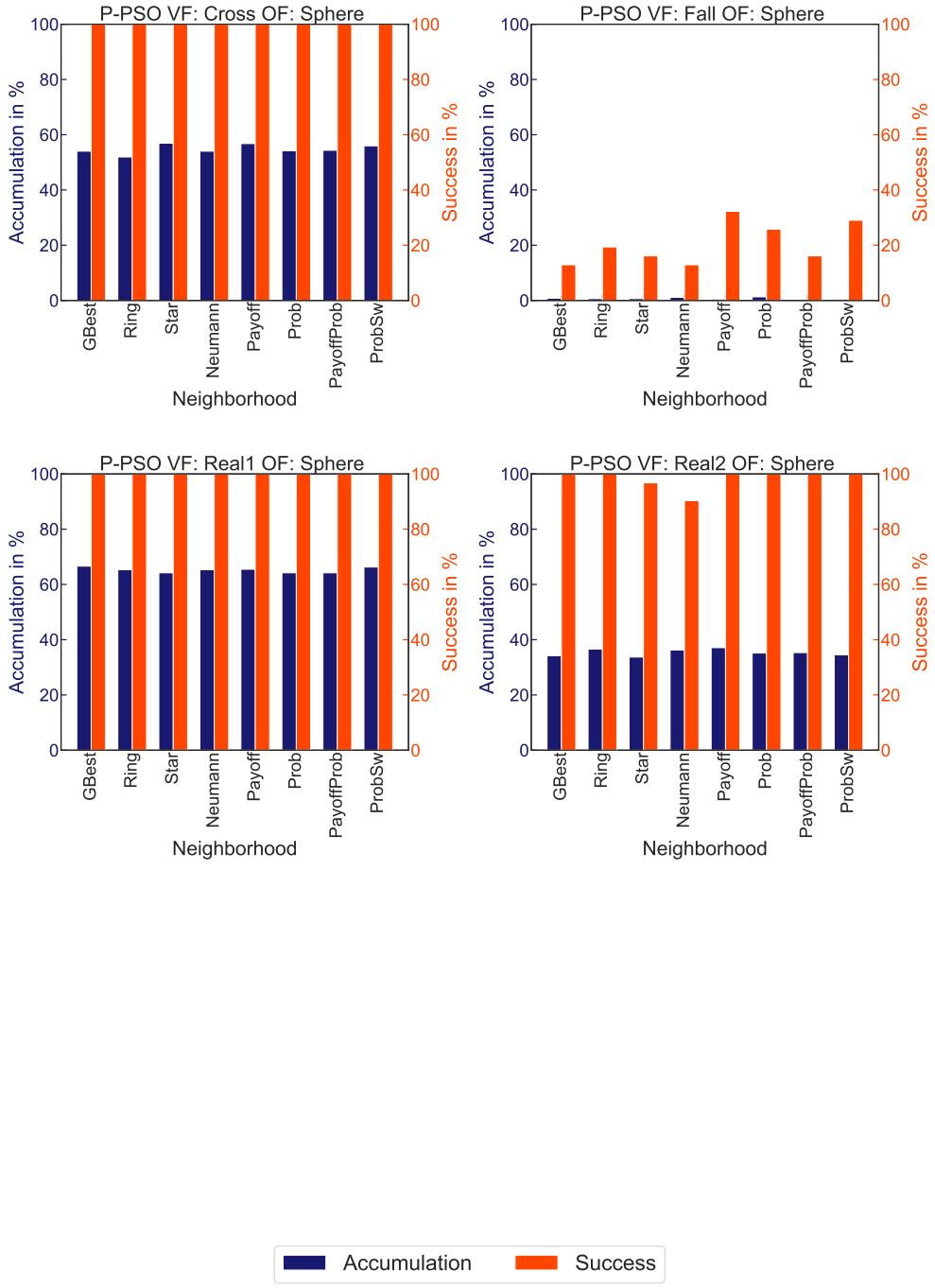
C.2.3. Ackley Function



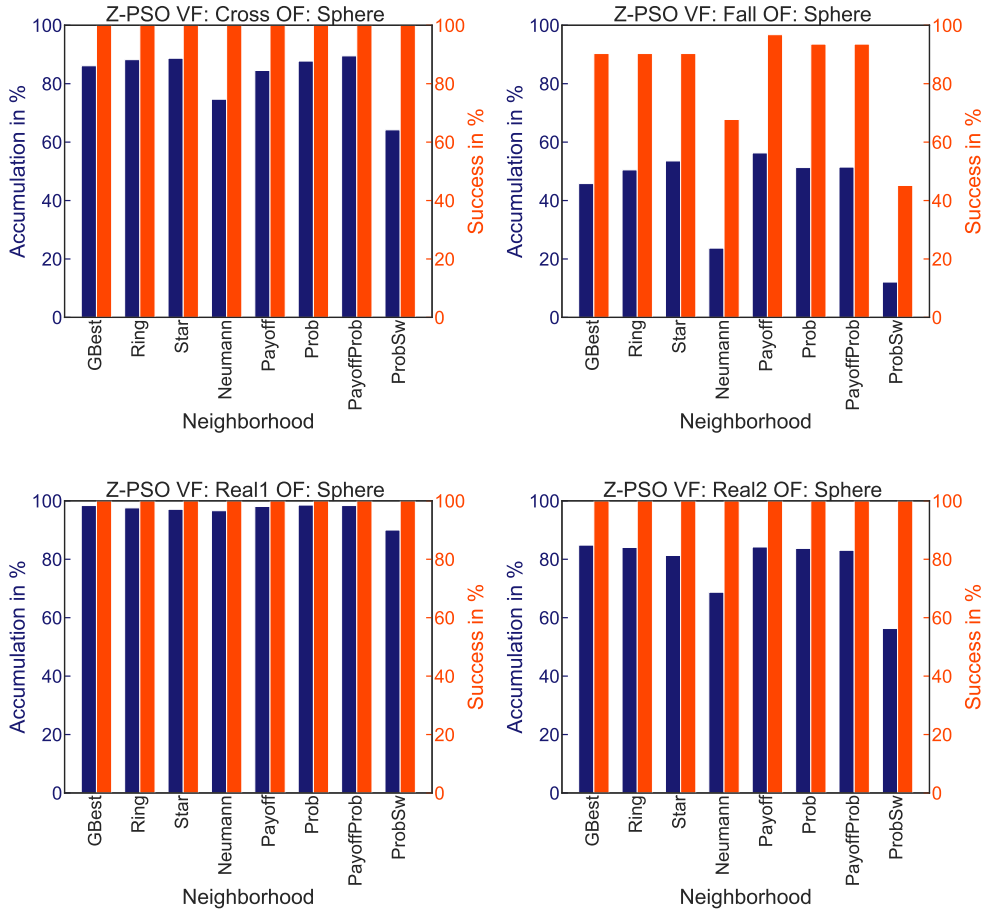
C.3. Accumulation and Success Plots

C.3.1. Sphere Function

P-PSO

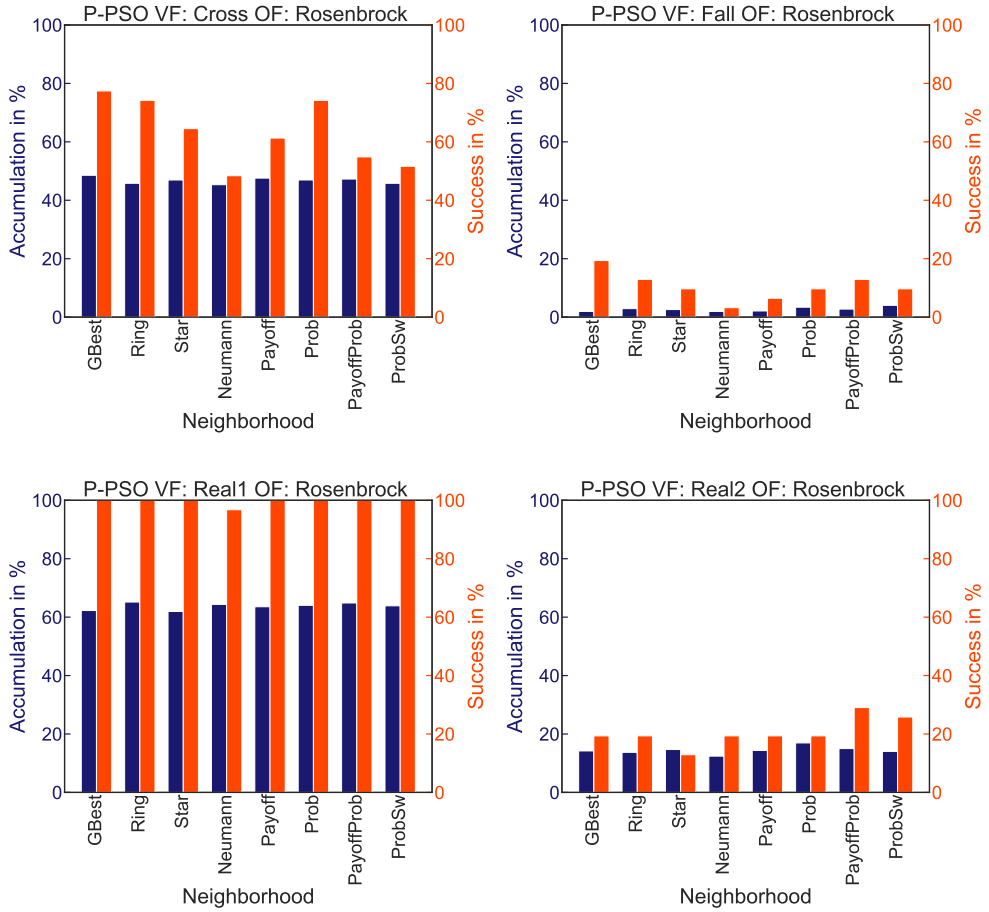


Z-PSO

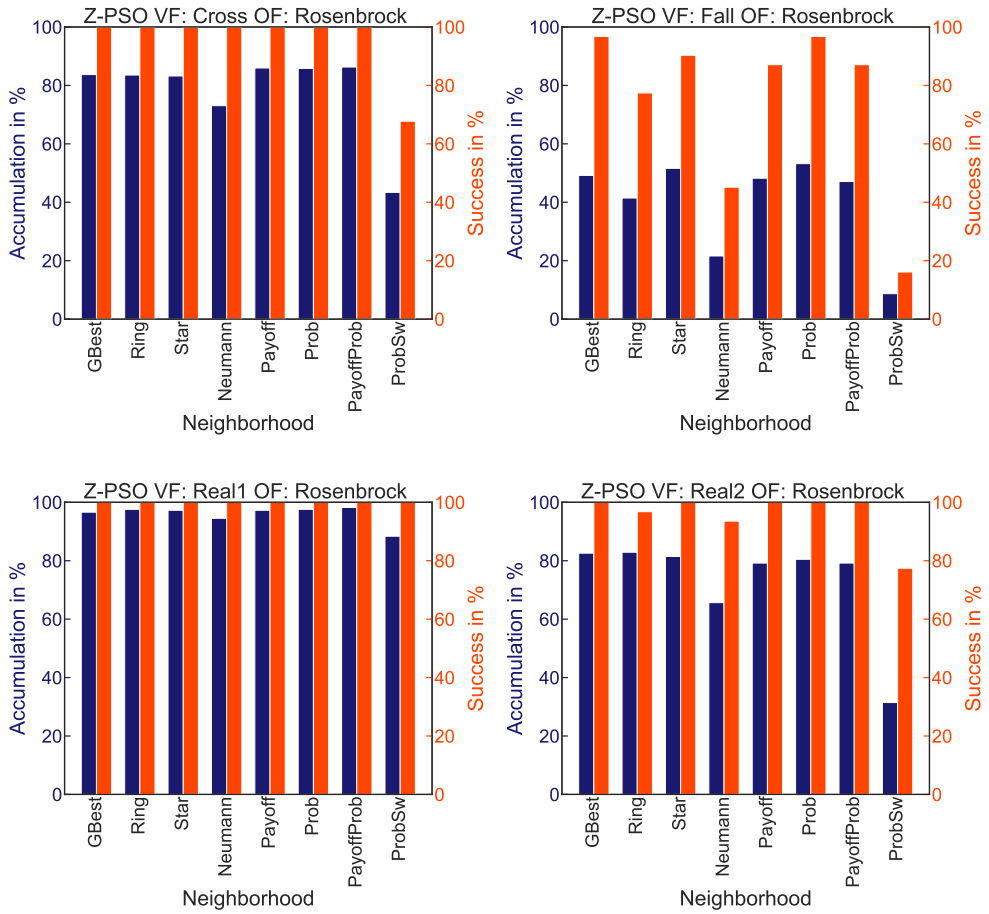

Z-PSO VF: Cross OF: Sphere
Z-PSO VF: Fall OF: Sphere
Z-PSO VF: Real1 OF: Sphere
Z-PSO VF: Real2 OF: Sphere
Accumulation in %
Success in %
Neighborhood
GBest
Ring
Star
Neumann
Payoff
Prob
PayoffProb
ProbSw
0
20
40
60
80
100
Accumulation
Success

C.3.2. Rosenbrock Function

P-PSO

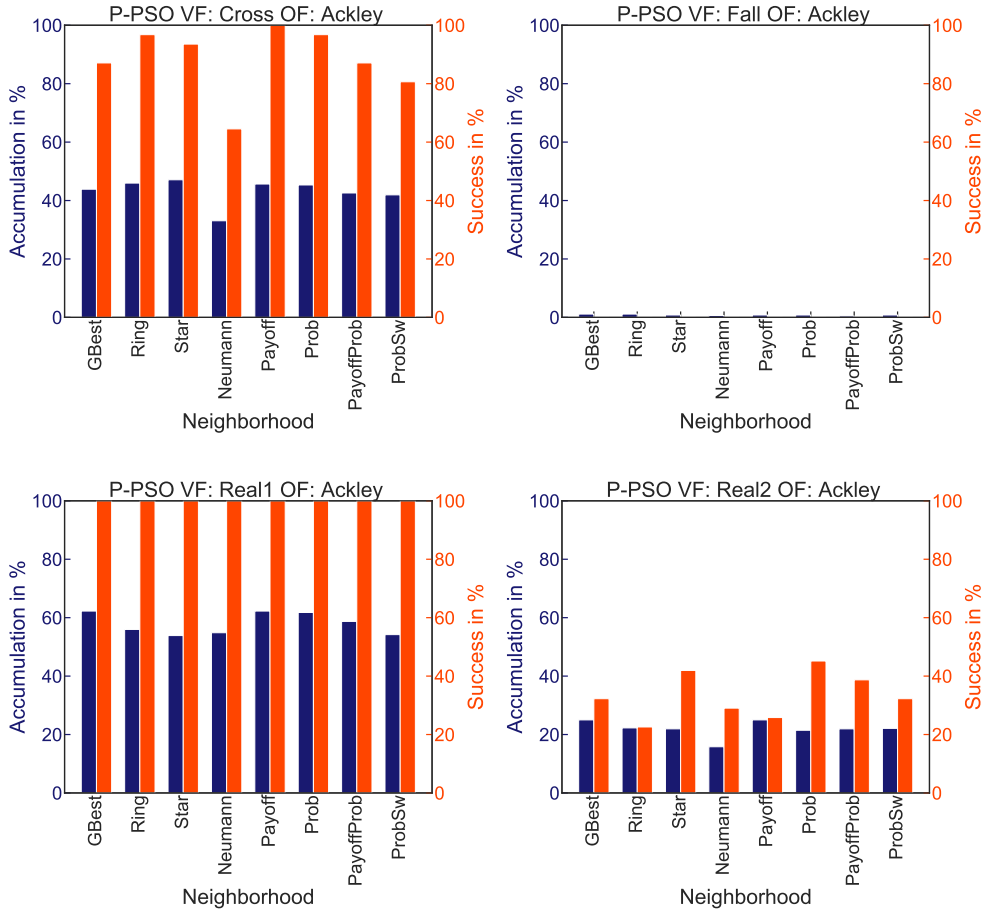


Z-PSO

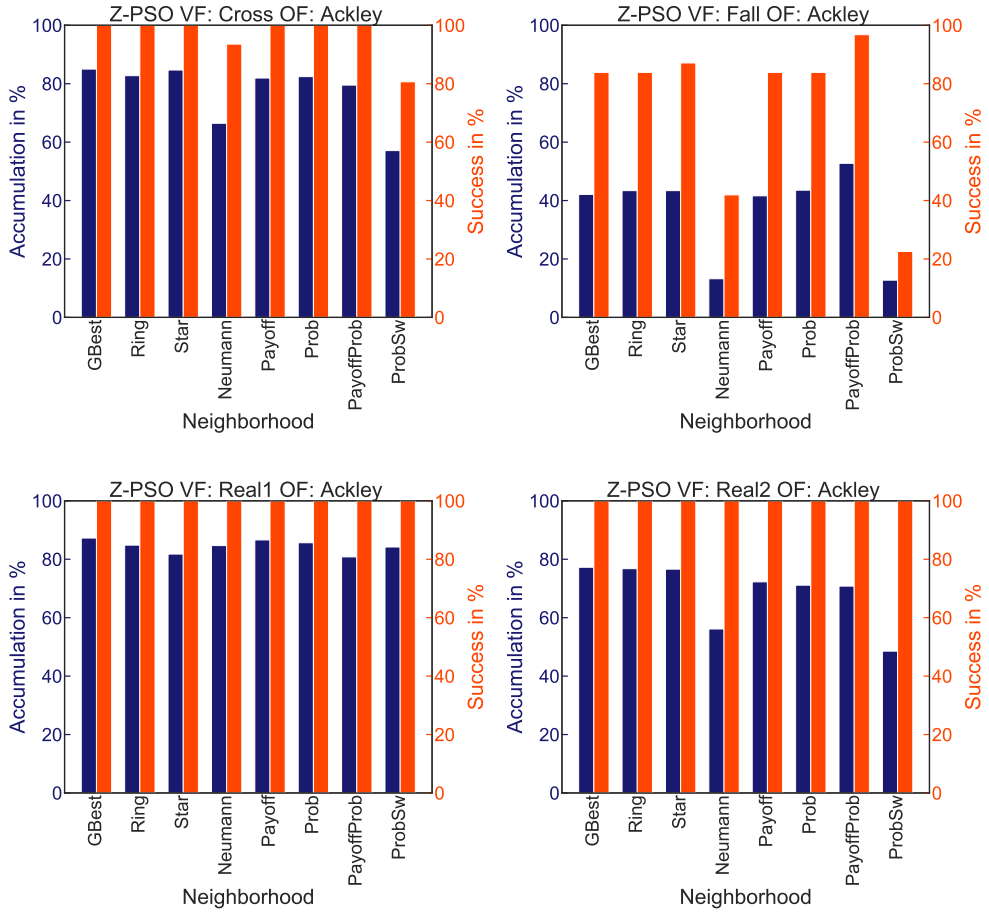

Z-PSO VF: Cross OF: Rosenbrock
Z-PSO VF: Fall OF: Rosenbrock
Z-PSO VF: Real1 OF: Rosenbrock
Z-PSO VF: Real2 OF: Rosenbrock
Accumulation in %
Success in %
Neighborhood
GBest
Ring
Star
Neumann
Payoff
Prob
PayoffProb
ProbSw
0
20
40
60
80
100
Accumulation
Success

C.3.3. Ackley Function

P-PSO



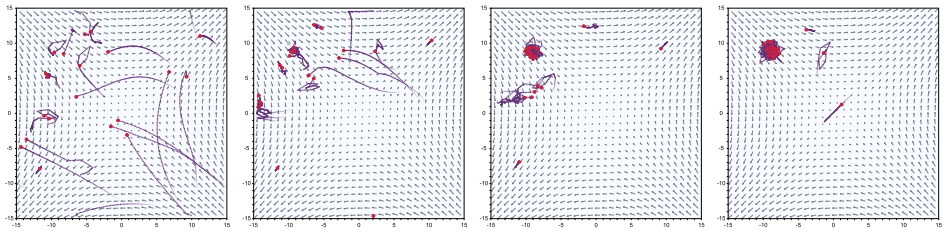
Z-PSO


Z-PSO VF: Cross OF: Ackley
Z-PSO VF: Fall OF: Ackley
Z-PSO VF: Real1 OF: Ackley
Z-PSO VF: Real2 OF: Ackley
Accumulation in %
Success in %
Neighborhood
GBest
Ring
Star
Neumann
Payoff
Prob
PayoffProb
ProbSw
0
20
40
60
80
100
Accumulation
Success

D. Agents Trajectory

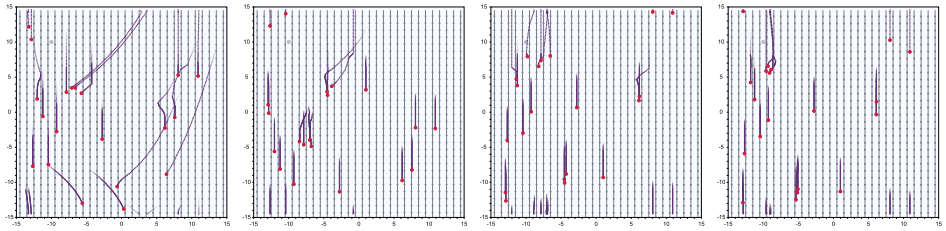
D.1. Sphere Function

D.1.1. P-PSO



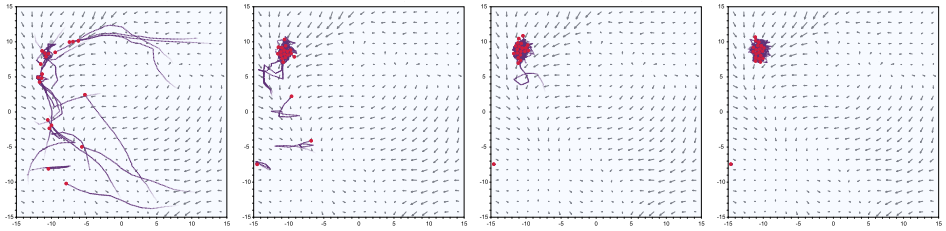
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.1.: Trajectory P-PSO OF: Sphere Function VF: Cross NH: GBest



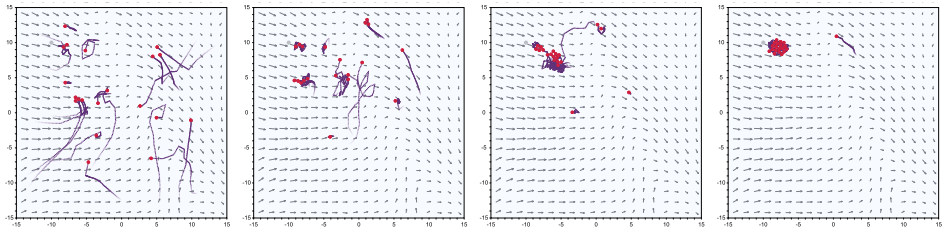
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.2.: Trajectory P-PSO OF: Sphere Function VF: Fall NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

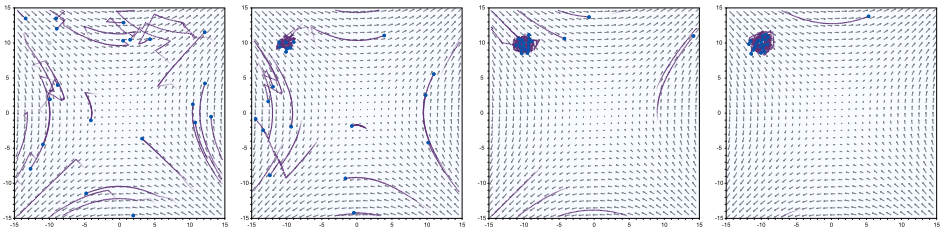
Figure D.3.: Trajectory P-PSO OF: Sphere Function VF: Real1 NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

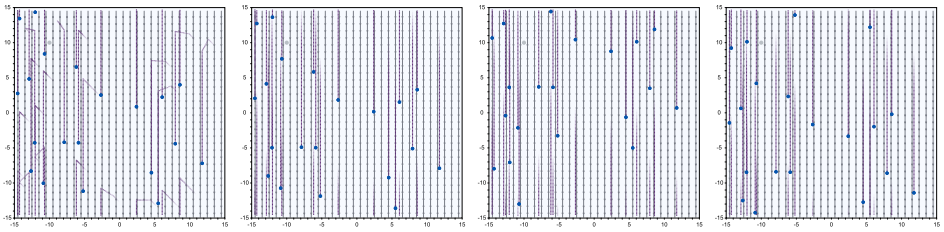
Figure D.4.: Trajectory P-PSO OF: Sphere Function VF: Real2 NH: GBest

D.1.2. Z-PSO



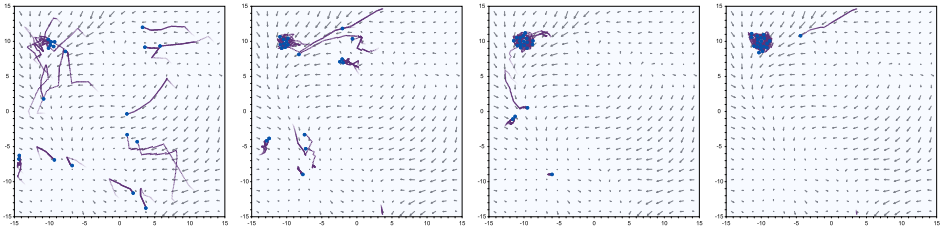
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.5.: Trajectory Z-PSO OF: Sphere Function VF: Cross NH: GBest



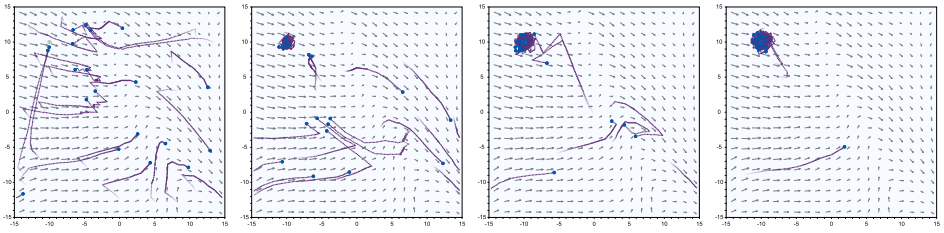
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.6.: Trajectory Z-PSO OF: Sphere Function VF: Fall NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.7.: Trajectory Z-PSO OF: Sphere Function VF: Real1 NH: GBest

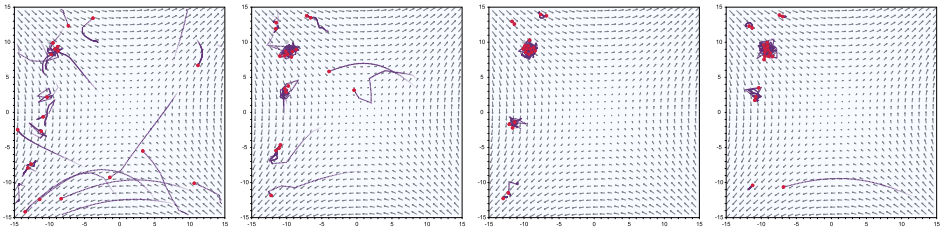


Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.8.: Trajectory Z-PSO OF: Sphere Function VF: Real2 NH: GBest

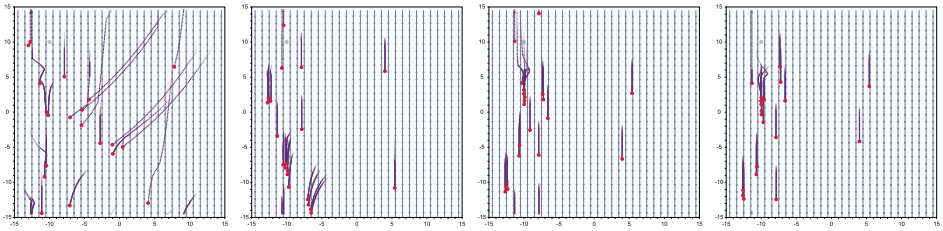
D.2. Rosenbrock Function

D.2.1. P-PSO



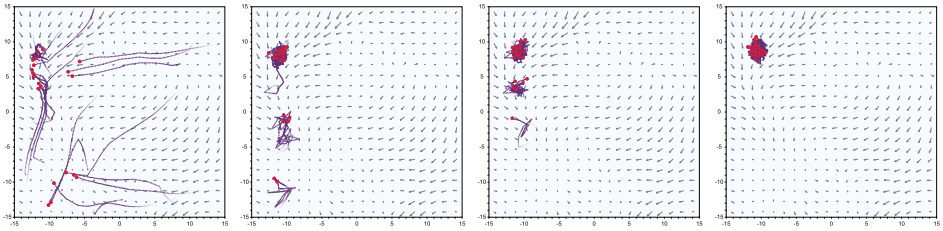
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.9.: Trajectory P-PSO OF: Rosenbrock Function VF: Cross NH: GBest



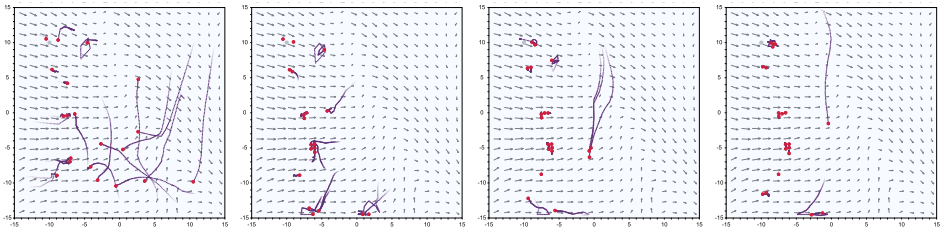
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.10.: Trajectory P-PSO OF: Rosenbrock Function VF: Fall NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

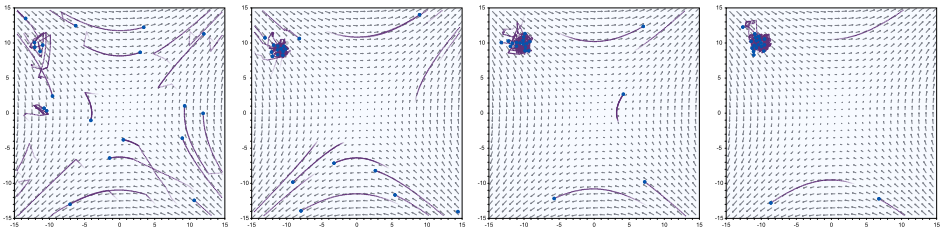
Figure D.11.: Trajectory P-PSO OF: Rosenbrock Function VF: Real1 NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

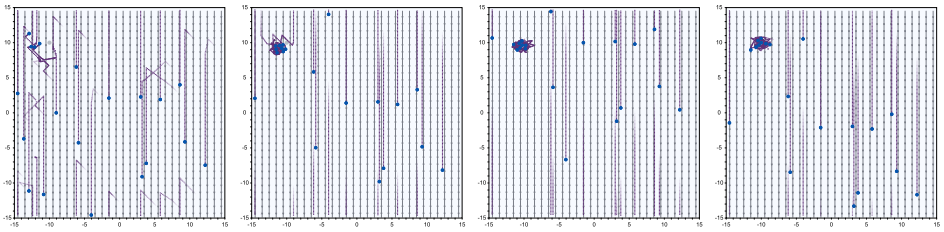
Figure D.12.: Trajectory P-PSO OF: Rosenbrock Function VF: Real2 NH: GBest

D.2.2. Z-PSO



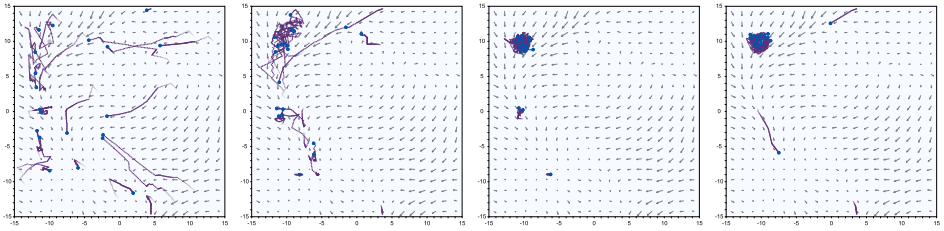
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.13.: Trajectory Z-PSO OF: Rosenbrock Function VF: Cross NH: GBest



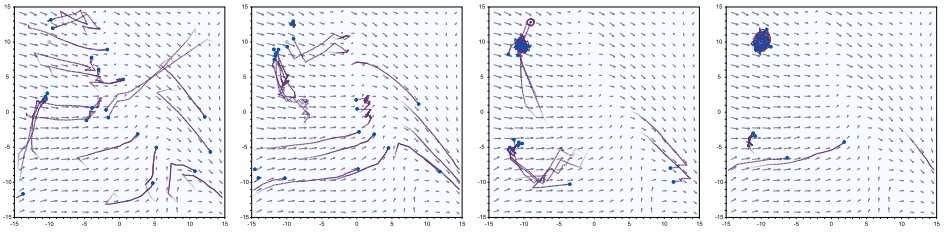
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.14.: Trajectory Z-PSO OF: Rosenbrock Function VF: Fall NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.15.: Trajectory Z-PSO OF: Rosenbrock Function VF: Real1 NH: GBest

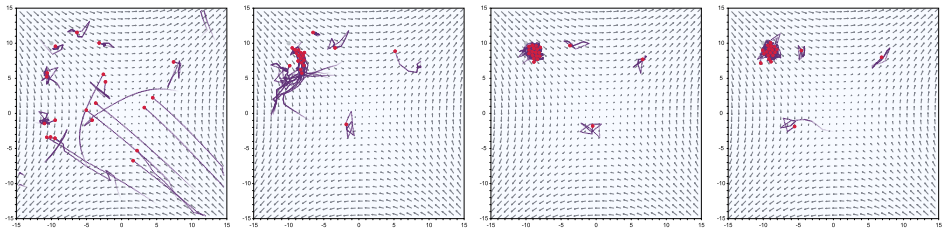


Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.16.: Trajectory Z-PSO OF: Rosenbrock Function VF: Real2 NH: GBest

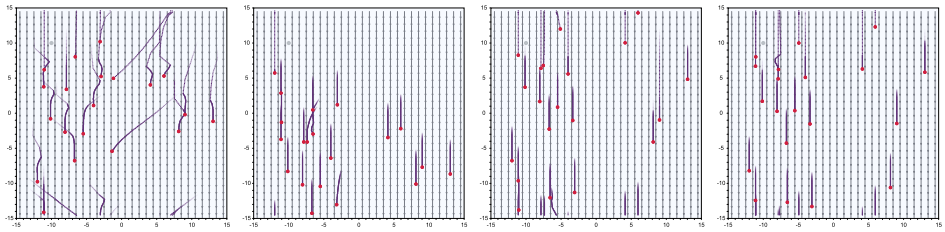
D.3. Ackley Function

D.3.1. P-PSO



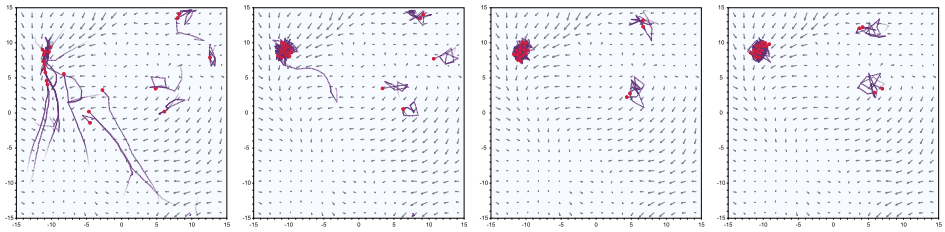
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.17.: Trajectory P-PSO OF: Ackley Function VF: Cross NH: GBest



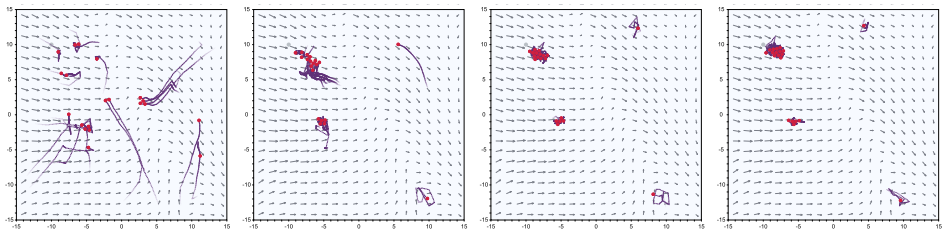
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.18.: Trajectory P-PSO OF: Ackley Function VF: Fall NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

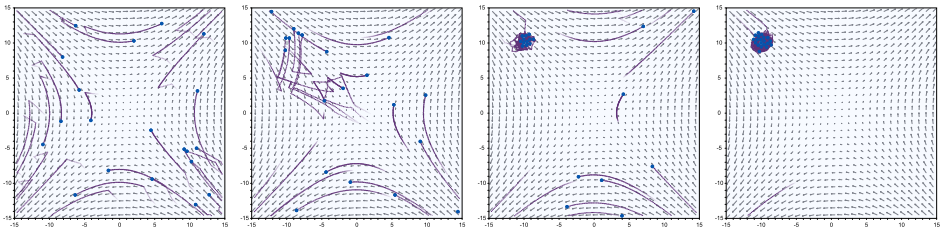
Figure D.19.: Trajectory P-PSO OF: Ackley Function VF: Real1 NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

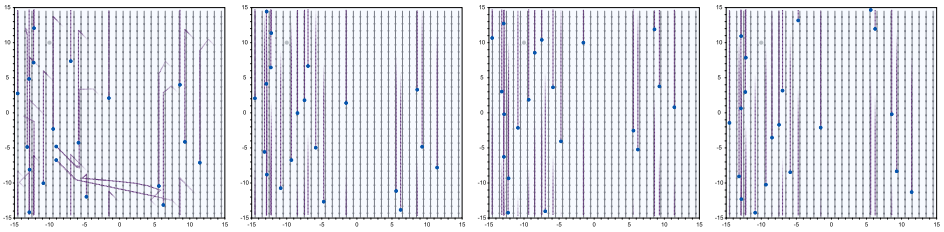
Figure D.20.: Trajectory P-PSO OF: Ackley Function VF: Real2 NH: GBest

D.3.2. Z-PSO



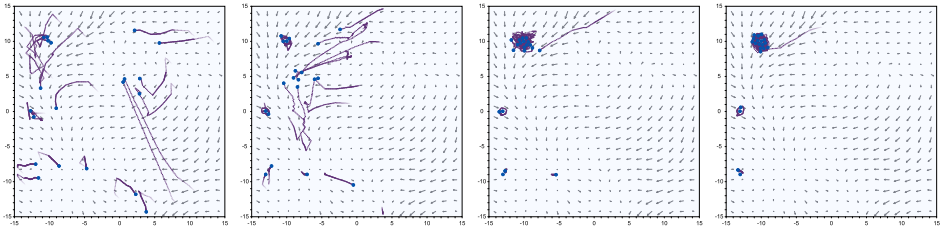
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.21.: Trajectory Z-PSO OF: Ackley Function VF: Cross NH: GBest



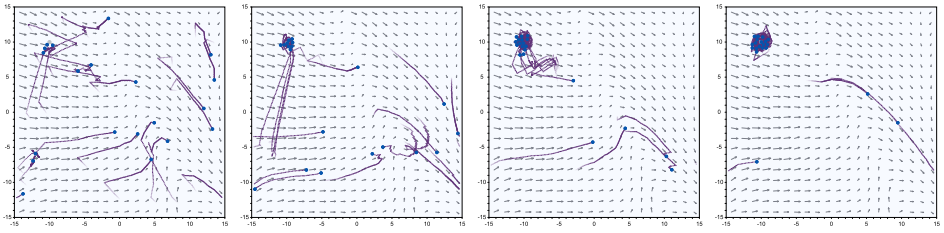
Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.22.: Trajectory Z-PSO OF: Ackley Function VF: Fall NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.23.: Trajectory Z-PSO OF: Ackley Function VF: Real1 NH: GBest



Iter: 10 Iter: 25 Iter: 50 Iter: 100

Figure D.24.: Trajectory Z-PSO OF: Ackley Function VF: Real2 NH: GBest

E. Implementation

E.1. Swarms

E.1.1. PSO

Algorithm 1: PSO

```
1 coefficientA = Vector( 1, 1 )
2 coefficientB = Vector( 1, 1 )
3 vel = particle.velocity
4 pos = particle.position
5 pBest = particle.pBestPosition
6 gBest = particle.gBestPosition
7 comp1 = coefficientA * random( 0, 1 ) * ( pBest - pos )
8 comp2 = coefficientB * random( 0, 1 ) * ( gBest - pos )
9 psoVector = weight * vel + comp1 + comp2
```

E.1.2. P-PSO

Algorithm 2: P-PSO

```
1 pso ← particle.getPSOVector()
2 if  0 < pso.length < 1 then
3 |   pso.mult( 10 )
4 if  pso.length == 0 then
5 |   pso ← particle.getOldPSOVector()
6 pso.limit( maximumVelocity )
7 particle.velocity = pso
```

E.1.3. Z-PSO

Algorithm 3: Z-PSO

```
1  if len(neighbors) == 0 then
2  |  particle.velocity = (0,0)
3  |  return
4  maxAngle = 135
5  pso ← particle.getPSOVector()
6  pso.limit( maximumVelocity )
7  wind ← particle.getWindPrediction()
8  angle ← abs( getAngle( pso, wind ) )
9  if angle > maxAngle then
10 |  rotationAngle = - angle - maxAngle
11 |  pso ← rotateVector( pso, rotationAngle )
12 particle.velocity = pso
```

E.2. Neighborhoods

E.2.1. GBest

Algorithm 4: GBest Neighbors

```
1 initialization swarm
2 for particle in swarm do
3 |   particle.neighbors ← getParticlesInRadius( particle, swarm )
4 end
```

E.2.2. Ring

Algorithm 5: Ring Neighbors

```
1 initialization swarm
2 for particle in swarm do
3 |   list ← getParticlesInRadius( particle, swarm )
4 |   leftNeighbor ← getFirstNextLeftParticle( list )
5 |   rightNeighbor ← getFirstNextRightParticle( list )
6 |   particle.neighbors = [ leftNeighbor, rightNeighbor ]
7 end
```

E.2.3. Star

Algorithm 6: Star Neighbors

```
1  initialization swarm
2  masterParticle = swarm[ 0 ]
3  for particle in swarm do
4  |   list ← getParticlesInRadius( particle, swarm )
5  |   if particle is masterParticle then
6  |   |   particle.neighbors = list
7  |   else if masterParticle is in list then
8  |   |   particle.neighbors = [ masterParticle ]
9  |   else
10 |   |   particle.neighbors = [ ]
11 end
```

E.2.4. Von Neumann

Algorithm 7: Von Neumann Neighbors

```
1 initialization swarm
2 for particle in swarm do
3 |   list ← getParticlesInRadius( particle, swarm )
4 |   leftNeighbor ← getSecondNextLeftParticle( list )
5 |   rightNeighbor ← getSecondNextRightParticle( list )
6 |   particle.neighbors = [leftNeighbor, rightNeighbor]
7 end
```

E.2.5. Payoff

Algorithm 8: Payoff Neighbors

```
1  initialization swarm
2  initialization payoffMemory
3  for particle in swarm do
4  |   particle.update( payoffMemory )
5  |   radiusList ← getParticlesInRadius( particle, swarm )
6  |   radiusList ← getReducedRandomParticles( radiusList, 5 )
7  |   coalitionList ← getSubsets( radiusList )
8  |   bestCoalition ← getBestCoalition( coalitionList, payoffMemory )
9  |   p.neighbors = bestCoalition
10 end
```

E.2.6. Probability

Algorithm 9: Probability Neighbors

```
1  initialization swarm
2  initialization probabilityMemory
3  for particle in swarm do
4  |  particle.update( probabilityMemory )
5  |  radiusList ← getParticlesInRadius( particle, swarm )
6  |  probList ← getProbabilitylist( radiusList )
7  |  if probList is empty then
8  |  |  randomParticle ← getRandomParticle( radiusList )
9  |  |  p.neighbors = [ randomParticle ]
10 |  else
11 |  |  p.neighbors = probList
12 end
```

E.2.7. Payoff Probability

Algorithm 10: Payoff Probability Neighbors

```
1  initialization swarm
2  initialization payoffMemory
3  initialization probabilityMemory
4  for particle in swarm do
5  |  particle.update( payoffMemory )
6  |  particle.update( probabilityMemory )
7  |  radiusList ← getParticlesInRadius( particle, swarm )
8  |  radiusList ← getReducedRandomParticles( radiusList, 5 )
9  |  coalitionList ← getSubsets( radiusList )
10 |  bestCoalition ← getBestCoalition( coalitionList, payoffMemory )
11 |  probList ← getProbabilitylist( bestCoalition )
12 |  if probList is empty then
13 |  |  randomParticle ← getRandomParticle( radiusList )
14 |  |  p.neighbors = [ randomParticle ]
15 |  else
16 |  |  p.neighbors = probList
17 end
```

E.2.8. Payoff Switch

Algorithm 11: Payoff Switch Neighbors

```
1  initialization swarm
2  initialization payoffMemory
3  initialization strategyMemory
4  for particle in swarm do
5  |  particle.update( payoffMemory )
6  |  particle.update( strategyMemory )
7  |  if particle.strategy is Cooperate then
8  |  |  p.neighbors ← getNeighborPayoff()
9  |  else
10 |  |  p.neighbors ← getNeighborRing()
11 end
```

E.2.9. Probability Switch

Algorithm 12: Probability Switch Neighbors

```
1  initialization swarm
2  initialization probabilityMemory
3  initialization strategyMemory
4  for particle in swarm do
5  |  particle.update( probabilityMemory )
6  |  particle.update( strategyMemory )
7  |  if particle.strategy is Cooperate then
8  |  |  p.neighbors ← getNeighborProbability()
9  |  else
10 |  |  p.neighbors ← getNeighborRing()
11 end
```

E.2.10. Payoff Probability Switch

Algorithm 13: Payoff Probability Switch Neighbors

```
1  initialization swarm
2  initialize payoffMemory
3  initialize probabilityMemory
4  initialize strategyMemory
5  for particle in swarm do
6  |   particle.update( payoffMemory )
7  |   particle.update( probabilityMemory )
8  |   particle.update( strategyMemory )
9  |   if particle.strategy is Cooperate then
10 |   |   p.neighbors ← getNeighborPayoffProbability()
11 |   else
12 |   |   p.neighbors ← getNeighborRing()
13 end
```

E.3. Collision Avoidance

E.3.1. Collision Detection

Algorithm 14: Collision Detection

```
1  collisionCount = 1;
2  while collisionCount > 0 do
3      collisionCount = 0
4      for ParticleA in swarm do
5          for ParticleB in swarm do
6              aPos = particleA.position
7              bPos = particleB.position
8              distance ← getDistance( aPos, bPos )
9              mindistance = particleRadius / 2 + particleRepulsion
10             collision = distance < minDistance
11             if collision ) then
12                 collisionCount += 1
13                 solveCollision( ParticleA, ParticleB )
14         end
15     end
16 end
```

E.3.2. Collision Resolution

Algorithm 15: Solve Collision

```
1 aPos = particleA.position
2 bPos = particleB.position
3 radius = (particleSize / 2 + particleRepulsion) )
4 pullMag ← abs( distance - radius )
5 c ← substract( bPos, aPos )
6 c.normalize()
7 pull.x = pullMag * c.x
8 pull.y = pullMag * c.y
9 aPos ← substract( aPos, pull )
```

Bibliography

[1] N. A. Ab Aziz and Z. Ibrahim. Asynchronous particle swarm optimization for swarm robotics. *Procedia Engineering*, 41:951–957, 2012.

[2] S. B. Akat and V. Gazi. Particle swarm optimization with dynamic neighborhood topology: Three neighborhood strategies and preliminary results. In *2008 IEEE Swarm Intelligence Symposium*, pages 1–8. IEEE, 2008.

[3] S. Almanasra. Evolutionary model for the iterated n-players prisoners'dilemma based on particle swarm optimization. *Journal of Theoretical and Applied Information Technology*, 97(5), 2019.

[4] N. A. A. Aziz, Z. Ibrahim, M. Mubin, S. W. Nawawi, and M. S. Mohamad. Improving particle swarm optimization via adaptive switching asynchronous–synchronous update. *Applied Soft Computing*, 72:298–311, 2018.

[5] J. C. Barca and Y. A. Sekercioglu. Swarm robotics reviewed. *Robotica*, 31(3):345–359, 2013.

[6] P. Bartashevich, L. Grimaldi, and S. Mostaghim. Pso-based search mechanism in dynamic environments: Swarms in vector fields. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1263–1270. IEEE, 2017.

[7] C. Blum and X. Li. Swarm intelligence in optimization. In *Swarm intelligence*, pages 43–85. Springer, 2008.

[8] Z. Cui, X. Cai, J. Zeng, and G. Sun. Predicted-velocity particle swarm optimization using game-theoretic approach. In *International Conference on Intelligent Computing*, pages 145–154. Springer, 2006.

[9] C. Di Chio, P. Di Chio, and M. Giacobini. An evolutionary game-theoretical approach to particle swarm optimisation. In *Workshops*

on Applications of Evolutionary Computation, pages 575–584. Springer, 2008.

[10] C. M. Fernandes, J. Merelo, and A. C. Rosa. An asynchronous and steady state update strategy for the particle swarm optimization algorithm. In *International Conference on Parallel Problem Solving from Nature*, pages 167–177. Springer, 2016.

[11] Y.-j. Gong and J. Zhang. Small-world particle swarm optimization with topology adaptation. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 25–32. ACM, 2013.

[12] S. Hamdan. Hybrid particle swarm optimiser using multi-neighborhood topologies. *INFOCOMP*, 7(1):36–43, 2008.

[13] S. Helwig, J. Branke, and S. Mostaghim. Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary computation*, 17(2):259–271, 2012.

[14] W. Jatmiko, K. Sekiyama, and T. Fukuda. A pso-based mobile sensor network for odor source localization in dynamic environment: theory, simulation and measurement. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1036–1043. IEEE, 2006.

[15] M. A. O. Junior, C. J. Bastos Filho, and R. Menezes. Using network science to define a dynamic communication topology for particle swarm optimizers. In *Complex Networks*, pages 39–47. Springer, 2013.

[16] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1931–1938. IEEE, 1999.

[17] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[18] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1671–1676. IEEE, 2002.

[19] F. Li and J. Guo. Topology optimization of particle swarm optimization. In *International Conference in Swarm Intelligence*, pages 142–149. Springer, 2014.

[20] H. Matsushita and Y. Nishio. Network-structured particle swarm optimizer with small-world topology. In *Proc. of Int. Symposium on Nonlinear Theory and its Applications*, pages 372–375, 2009.

[21] H. Matsushita, Y. Nishio, and T. Saito. Particle swarm optimization with novel concept of complex network. In *Proc. of Int. Symposium on Nonlinear Theory and its Applications*, pages 197–200, 2010.

[22] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: simpler, maybe better. *IEEE transactions on evolutionary computation*, 8(3):204–210, 2004.

[23] Q. Ni, C. Cao, and X. Yin. A new dynamic probabilistic particle swarm optimization with dynamic random population topology. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1321–1327. IEEE, 2014.

[24] M. J. Osborne et al. *An introduction to game theory*, volume 3. Oxford university press New York, 2004.

[25] J. Penders, L. Alboul, U. Witkowski, A. Naghsh, J. Saez-Pons, S. Herbrechtsmeier, and M. El-Habbal. A robot swarm assisting a human fire-fighter. *Advanced Robotics*, 25(1-2):93–117, 2011.

[26] J. Rada-Vilela, M. Zhang, and W. Seah. A performance study on synchronous and asynchronous updates in particle swarm optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 21–28. ACM, 2011.

[27] J. Rada-Vilela, M. Zhang, and W. Seah. A performance study on synchronicity and neighborhood size in particle swarm optimization. *Soft Computing*, 17(6):1019–1030, 2013.

[28] E. Sahin and A. F. Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2-4):69–72, 2008.

[29] A. K. Saxena and M. Vora. Novel approach for the use of small world theory in particle swarm optimization. In *2008 16th International*

Conference on Advanced Computing and Communications, pages 363–366. IEEE, 2008.

[30] G. Toscano-Pulido, A. J. Reyes-Medina, and J. G. Ramírez-Torres. A statistical study of the effects of neighborhood topologies in particle swarm optimization. In *Computational Intelligence*, pages 179–192. Springer, 2011.

[31] M. Turduev, Y. Ataş, P. Sousa, V. Gazi, and L. Marques. Cooperative chemical concentration map building using decentralized asynchronous particle swarm optimization based search by mobile robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4175–4180. IEEE, 2010.

[32] Y.-X. Wang and Q.-L. Xiang. Particle swarms with dynamic ring topology. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 419–423. IEEE, 2008.

[33] X. Yao and P. J. Darwen. An experimental study of n-person iterated prisoner's dilemma games. *Informatica*, 18(4):435–450, 1994.

[34] A. E. M. Zavala. A comparison study of pso neighborhoods. In *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II*, pages 251–265. Springer, 2013.

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Doreen Körte Magdeburg, Thursday 24th September, 2020