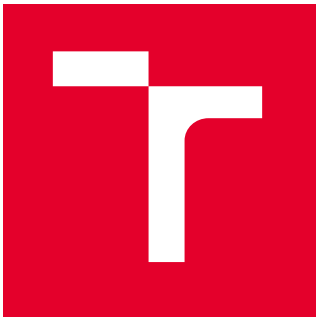

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY

DEPARTMENT OF RADIO ELECTRONICS

NÁSTROJ PRO PLÁNOVÁNÍ ROBOTICKÉ NEUROCHIRURGIE

ROBOTIC NEUROSURGERY PLANNING TOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Simona Sadleková**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Kadlec, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Elektronika a komunikační technologie**

Ústav radioelektroniky

Studentka: Bc. Simona Sadleková ***ID:*** 174528

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Nástroj pro plánování robotické neurochirurgie

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou plánování robotických operací. Implementujte vhodný algoritmus více-kriteriální optimalizace z množiny heuristických algoritmů (MOPSO, GDE3, NSGA-II, MOSOMA aj.). Algoritmus upravte tak, aby byl schopen pracovat s různými dimenzemi problému. Algoritmus vyzkoušejte na sadě testovacích úloh např. z [3].

Získejte 2D řezy 3D geometrických objektů, které představují nejčastější tvary tumorů. Jednotlivé objekty budou v řezu reprezentovány jako obecné polygony. Aplikujte vytvořený optimalizační nástroj pro plánování vhodného postupu operace pro několik vybraných 2D útvarů typických pro nádorová onemocnění. Diskutujte vliv nastavení parametrů využitého optimalizačního algoritmu na dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] GRANNA, Josephine, Arya NABAVI a Jessica BURGNER-KAHRS. Computer-assisted planning for a concentric tube robotic system in neurosurgery. International Journal of Computer Assisted Radiology and Surgery [online]. 2019, 14(2), 335-344 [cit. 2021-5-24]. ISSN 1861-6410. Dostupné z: doi:10.1007/s11548-018-1890-8.

[2] DEB, Kalyanmoy. Multi-objective optimization using evolutionary algorithms. New York: John Wiley, c2001. ISBN 047187339X.

[3] MAREK, Martin a Petr KADLEC. Another evolution of generalized differential evolution: variable number of dimensions. Engineering Optimization [online]. , 1-20 [cit. 2021-5-28]. ISSN 0305-215X. Dostupné z: doi:10.1080/0305215X.2020.1853714.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 22.5.2023

Vedoucí práce: doc. Ing. Petr Kadlec, Ph.D.

doc. Ing. Lucie Hudcová, Ph.D.
předseda rady studijního programu

ABSTRACT

This thesis deals with the planning of robotic operations for neurosurgery and the optimization algorithms used to propose the surgical procedures performed by them, primarily the VNDMOPSO algorithm. The introductory chapter explains the basic principle of performing robotic operations and describes a concentric tube robot that can be used to perform these operations. Furthermore, the thesis deals with a general description of optimization problems and algorithms solving them, and then the VNDMOPSO algorithm, selected for the given optimization problem, is described in detail. For this algorithm, a function in the MATLAB is created, which is subsequently tested on several benchmark problems. In the following sections, its functionality is verified on real tumor shapes and a graphical user interface, which serves as a robotic neurosurgery planning tool, is presented. In the final part of the thesis, the influence of setting individual parameters of the algorithm on the optimization results is evaluated.

KEYWORDS

Ablation, benchmark problems, CTR, GUI, optimization algorithms, planning of robotic neurosurgery, test metrics, tumor, VNDMOPSO

ABSTRAKT

Táto diplomová práca sa zaoberá problematikou plánovania robotických operácií pre neurochirurgiu a optimalizačnými algoritmami používanými na návrh operačných zákrokov nimi prevádzaných, predovšetkým algoritmom VNDMOPSO. V úvodnej kapitole je vysvetlený základný princíp prevádzania robotických operácií a popísaný koncentrický trubicový robot používaný pre tento účel. Ďalej sa práca zaoberá všeobecným popisom optimalizačných problémov a algoritmov ich riešiacich a následne je podrobne popisovaný algoritmus VNDMOPSO, vybraný pre daný optimalizačný problém. Pre tento algoritmus je vytvorená v prostredí MATLAB funkcia, ktorá je následne testovaná na viacerých testovacích úlohách. V nasledujúcich častiach je overovaná jej funkčnosť na reálnych tvaroch nádorov a je predstavené grafické užívateľské prostredie, ktoré slúži ako nástroj pre plánovanie neurochirurgických robotických operácií. V záverečnej časti práce je vyhodnotený vplyv nastavenia jednotlivých parametrov algoritmu na výsledky optimalizácie.

KĽÚČOVÉ SLOVÁ

Ablácia, CTR, GUI, nádor, optimalizačné algoritmy, plánovanie robotickej neurochirurgie, testovacie úlohy, testovacie metriky, VNDMOPSO

ROZŠÍRENÝ ABSTRAKT

V súčasnosti je čoraz väčšia pozornosť upriamená na vývoj využitia robotiky v chirurgickej praxi z dôvodu zvyšujúcich sa požiadaviek na minimálnu invazivitu prevádzania operačných zákrokov. Začlenenie robotov do lekárskej praxe umožňuje chirurgom posúvať hranice svojich technických schopností. Operácie, ktoré predtým znamenali veľkú záťaž pre pacienta aj lekársky tím, je teraz možné vykonávať rýchlejšie a s oveľa menším počtom pooperačných komplikácií a kratším časom rekonvalescencie [1].

Oblasť neurochirurgie sa dá považovať za jednu z najdôležitejších oblastí, v ktorej by mal byť využívaný minimálne invazívny prístup. Napriek tomu, že anatomická zložitosť neurologických štruktúr znamenala značný problém pri začlenení robotiky do tejto oblasti, bolo vyvinutých niekoľko robotov umožňujúcich odstránenie mozgového nádorového tkaniva [1]. Medzi ne patrí aj koncentrický trubicový robotický systém, ktorý dokáže pomocou metódy laserom indukovanej termoterapie denaturovať nádorové tkanivo zvnútra. Táto metóda má veľký potenciál, pretože ide o minimálne invazívny prístup k liečbe mozgových nádorov za použitia magnetickej rezonancie [3].

Proces plánovania zohráva pri tomto type operácií kľúčovú úlohu. Na základe neho je umožnené spoľahlivé odstránenie najväčšej možnej časti nádoru s minimálnym poškodením zdravého okolitého tkaniva. Pre tieto účely sa používajú multikriteriálne optimalizačné algoritmy, ktoré dokážu nájsť súbor najlepších riešení, tvoriaci tzv. Paretovo čelo, z množiny možných riešení tak, aby spĺňali požiadavky definované kriteriálnymi funkciami [10]. Medzi tieto patrí aj algoritmus Multi-objective Particle Swarm Optimization pre premenný počet dimenzií (VNDMOPSO), ktorý bol v tejto práci zvolený na optimalizáciu úlohy plánovania takto vykonávaných operácií.

Teoretická časť práce sa zaoberá popisom koncentrického trubicového systému a metódy laserovo indukovanej termálnej ablácie, ktorá sa používa na denaturáciu tkaniva, ako aj popisom zvoleného optimalizačného algoritmu.

V praktickej časti práce je popisovaná funkcia `VNDMOPSO`, vytvorená v prostredí MATLAB pre implementáciu tohoto algoritmu, ktorá je následne testovaná na viacerých testovacích problémoch. Výsledky sú následne vyhodnocované pomocou troch testovacích metrík, ktoré umožňujú kvalitatívne posúdiť vytvorený optimalizačný nástroj.

Ďalšia časť práce je venovaná testovaniu funkcie na 2D rezoch 3D geometrických objektov, reprezentovaných vo forme obecného polygónu. Tieto objekty predstavujú tvary nádorov často sa vyskytujúcich v medicínskej praxi. Pre vizualizáciu výsledkov testovania bolo vytvorené grafické užívateľské prostredie, ktoré slúži ako nástroj

pre plánovanie robotických neurochirurgických operácií. Tento nástroj umožňuje zobraziť jednotlivé riešenia v Paretovom čele a rozmiestnenie ablačných objektov v oblasti nádoru zodpovedajúce vybranému riešeniu.

Výsledky ukazujú, že algoritmus VNDMOPSO dokáže nájsť množinu riešení vhodných pre vykonávanie tohoto typu operácií, pričom úspešnosť nájdenia ideálnej množiny riešení sa líšila v závislosti od nastavenia viacerých parametrov. Najväčší vplyv malo nastavenie typu ohraničenia priestoru riešení. Algoritmus dokázal nájsť najlepšie riešenia v prípade steny typu absorbing, zatiaľ čo v prípade steny typu reflecting boli výsledky podobné. Pri voľbe neviditeľného typu steny boli výsledky neuspokojivé. Na výsledky optimalizácie mali vplyv aj rôzne kombinácie nastavenia počtu agentov a iterácií. Ukázalo sa, že pri zachovaní množstva výpočtov kriteriálnej funkcie sa oplatí nastaviť skôr viac agentov ako iterácií.

Nakoniec bol optimalizátor s premenným počtom dimenzií porovnaný s optimalizátorom s pevným počtom dimenzií. Pomocou metódy Dominance Ranking bolo vyhodnotené, že verzia algoritmu VND je výrazne lepšia pri použití steny typu absorbing a reflecting. Pri použití neviditeľnej steny bola úspešnejšia verzia s pevným počtom dimenzií.

SADLEKOVÁ, Simona. *Robotic Neurosurgery Planning Tool*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Radio Electronics, 2023, 81 p. Master's Thesis. Advised by doc. Ing. Petr Kadlec, Ph.D.

Author's Declaration

Author:	Bc. Simona Sadleková
Author's ID:	174528
Paper type:	Master's Thesis
Academic year:	2022/23
Topic:	Robotic Neurosurgery Planning Tool

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank my supervisor doc. Ing. Petr Kadlec, Ph.D. for professional guidance, patience and all useful advice and suggestions for my thesis.

Contents

List of Figures

List of Tables

Introduction

Nowadays, more and more attention is paid to medical procedures being minimally invasive. Therefore, in recent decades, the idea of using robotics in surgical practice has been widely developed. Incorporating robots into medical practice thus allows surgeons to push the limits of their technical skills. Surgeries, which meant a great burden for both the patient and the medical team, can now be performed faster and with much fewer post-operative complications and a shorter recovery time [1].

Perhaps the most important area in which a minimally invasive approach is required is the field of neurosurgery. The anatomical complexity of neurological structures meant a considerable problem in the incorporation of robotics in this field [1]. Despite this, several robots have been developed, enabling the removal of tumor tissue from the brain. One of them is a concentric tube robotic system that can denature tumor tissue from the inside using the Laser-Induced Thermotherapy method. This method has great potential because it is a minimally invasive approach to the treatment of brain tumors, thanks to the assistance of magnetic resonance imaging [3].

For the successful removal of tumor tissue, the procedure of each operation must be carefully planned for the specific shape of the tumor. Multi-objective optimization algorithms are used to achieve this task. These include e.g. Multi-objective Particle Swarm Optimization (MOPSO), Multi-Objective Self-Organizing Migrating Algorithm (MOSOMA), Generalized Differential Evolution (GDE3), Non-dominated Sorting Genetic Algorithm (NSGA-II) and others [3].

The theoretical part of this thesis deals with the description of the concentric tube robot and the ablation method that is used for tissue denaturation, as well as the description of the multi-objective optimization algorithm MOPSO, which was chosen to optimize the task performed by the robot.

Subsequently, in the practical part, a function for the implementation of this algorithm is created in MATLAB, which is then tested on several benchmark problems. The validity of the results is subsequently verified using three test metrics. In the next part of the thesis, its functionality is tested on several real shapes of tumors, represented in the form of general polygons in two-dimensional space. A graphical user interface created for visualization of the results, which represents a tool for planning robotic neurosurgical operations, is also presented. The final part of the thesis is devoted to the evaluation of the optimization results and the influence of the setting of individual parameters of the algorithm on them.

1 Robotics in Neurosurgery

In recent years, robotics in the medical field has become one of the main subjects of interest for developers, which has caused the rapid development of this discipline. The first robot used in neurosurgery to perform a brain biopsy was introduced in 1985, and the first robot which was marked the first robotic device approved by the FDA for neurosurgical procedures was developed in the 1990s. Since then, many advances in artificial intelligence and machine learning enabled to evolve various robots. They are now an integral part of medical practise around the world [5] [6].

This chapter describes the method of brain tumor treatment, called laser-induced thermotherapy [7], which makes it possible to remove even an irregularly shaped tumor as well as the robotic surgery system capable to perform such a complex task.

1.1 Medical Motivation

Brain tumors are usually treated with surgery. However, there are cases when it is not possible to completely remove the tumor without damaging the surrounding vital brain tissues. Mostly, these are tumors of irregular shape. In such cases, other treatment approaches are considered, such as chemotherapy, radiotherapy or even a method called laser-induced thermotherapy (LITT), described below [2].

1.1.1 Laser-Induced Thermotherapy Method

Laser-induced thermotherapy is nowadays a widely used method for removing brain tumors. This method consists of several thermal ablation applications in the volume of the tumor, as shown in Fig. 1.1.

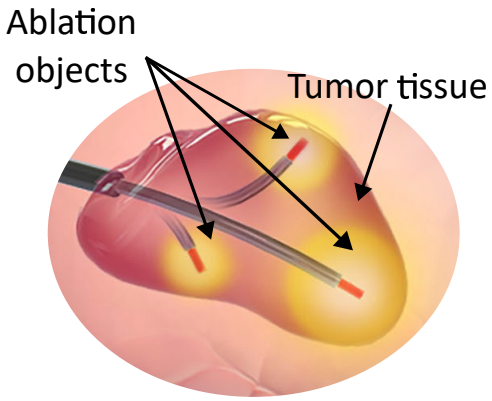

Fig. 1.1: Thermal ablation applied in vitro of tumor tissue [3].

The application is performed using a laser producing non-ionizing radiation. Laser tissue destruction occurs by absorption and scattering. Absorption is the process of converting laser energy into heat after the collision of photons and molecules of the target tissue (so-called chromophores). This heat causes photothermal heating, which destroys diseased cells, as illustrated in Fig. 1.2A. Scattering of light on particles in the tissue then causes an increase in the spatial distribution of light (Fig. 1.2B).

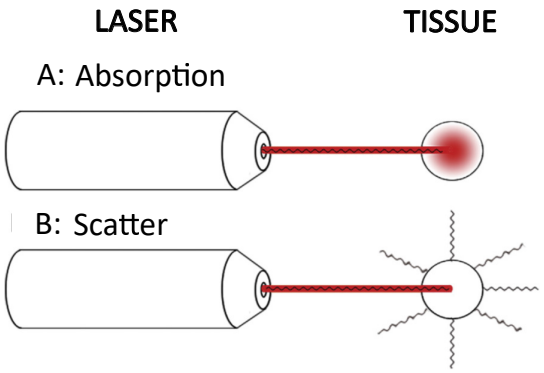

Fig. 1.2: Two main types of interaction between tissue and laser light: A: absorption, and B: scattering [8].

The success of tissue removal depends on the specific properties of the tissue (e.g. conductivity, density, perfusion...) and also on the choice of the radiation wavelength so that tissue heating and light penetration are optimized [7]. The use of this method allows removing tumor tissue with minimal postoperative complications and represents a minimally invasive treatment approach (more detailed information about LITT can be found in [7]).

1.2 Planning Robotic Operations

In order to enable optimal tissue removal by the LITT method, a concentric tube robot (CTR) was designed that can induce thermal energy into the tissue and thus effectively destroy it. The robot is optimized using a computer-assisted planning process that includes optimization of the ablation objects and their placement in the tumor volume (task-specific planning) and optimization of the parameters of the robot itself (robot-specific planning) [3].

1.2.1 Task-Specific Planning

The main goal of task-specific planning procedure is to ensure that the treatment is as minimally invasive as possible. Therefore, the optimization of the placement of ablation objects is very important. It is requisite to remove as much of the tumor as possible, on the other hand, the vital surrounding tissues must not be overheated. These two criteria contradict each other, so it is important to find a compromise between them, using multi-objective optimization methods [3].

Task-specific planning involves the calculation of optimal parameters of ablation objects (number, size, position), considering these two criteria. Achieving optimal distribution can be viewed as an unequal sphere packing problem to place a number of objects into a volume. The method called bin packing, illustrated in Fig. 1.3 (left), could be used for solving this problem. However, in order to remove the largest part of the tumor, we must consider the possibility of the ablation objects overlapping each other (Fig. 1.3 (right)). That's why bin packing method can't be used for this type of optimization problem and parameters of ablation objects must be optimized by some kind of available multi-objective optimization methods (e.g. MOPSO, GDE3, MOSOMA, NSGA-II) [3].

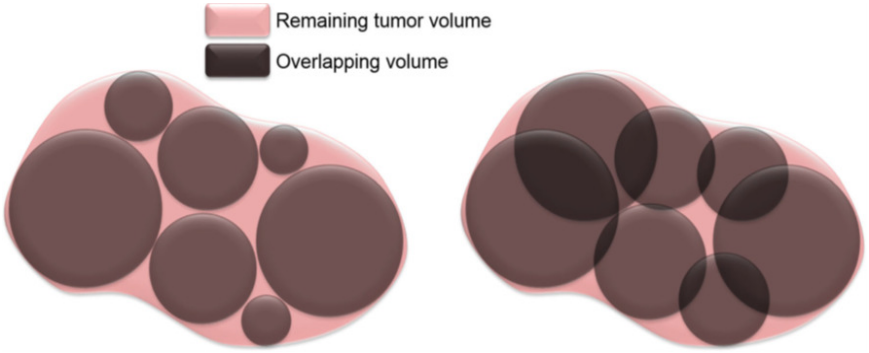

Fig. 1.3: Regular bin packing problem (left) and packing problem for robot-assisted LITT with overlapping objects (right) [3].

1.2.2 Robot-Specific Planning

The parameters of the ablation objects selected by this algorithm are then the input values for the optimization of the robot. This means that CTR optimization is closely related to the results of the task-specific planning process. Robot-specific planning procedure than includes the optimization of the parameters of the robot itself, such as the curvature and length of the tube, as well as the computation of the most suitable trajectories into the tumor volume. [3].

1.2.3 Problem Description

It should be noted that a real ablation object would not have a uniform thermal effect over its entire surface, but the heat would spread progressively from the ablation needle to the edge of the object. The propagation of the heat produced by the laser would depend on the material properties of surrounding tissues. However, tumor tissue has different properties than normal, physiological one, so it would be very difficult to calculate the parameters of heat propagation throughout the ablation object. For the sake of simplicity, the thermal propagation from the ablation injection is simulated as a circle with uniform thermal propagation in this thesis.

The objectives of the optimization problem can be expressed by the percentage of the tumor volume not covered by the ablation objects and the percentage of the tumor volume where the ablation objects overlap or protrude from the tumor volume. This problem can be defined using the following objective functions f_1 and f_2:

$$\min_{\vec{x}} f_1(\vec{x}) = \bigcup_{i=1}^{N_\mathrm{s}} r_i, \tag{1.1}$$

$$\min_{\vec{x}} f_2(\vec{x}) = \bigcap_{i=1}^{N_\mathrm{s}} s_i + \bigcap_{i=1}^{N_\mathrm{s}} o_i, \tag{1.2}$$

where $\vec{x}$ defines the decision vector (solution), r_i represents the area where the i-th ablation object does not cover part of the tumor volume, s_i represents the area where the i-th ablation object covers part of the tumor volume and o_i represents the part of the i-th ablation object that protrudes from the tumor and extends into healthy tissue. The intersection area of individual surfaces is calculated over all N_s ablation objects.

The minimization of functions f_1 and f_2 is the goal of task-specific planning, which is one of the tasks of this thesis [3].

1.3 Concentric Tube Robots

Concentric tube robots form a special class of so-called continuum robots, which are mainly used in interventional medicine. They consist of superelastic and very thin concentric tubes, the size of which can be compared to the size of a catheter. This fact makes them different from classic rigid robots with high mechanical rigidity and a limited degrees-of-freedom (DOFs) [4]. These properties predispose them to be used for many surgical applications.

The concentric tube robotic system consists of two types of tubes: the outer, so-called delivery tube, and internal, so-called ablation guide tube. The delivery tube represents the straight and rigid part of the robot, while the ablation guide tube

is less rigid and consists of a straight part and a curved end part [3]. Figure 1.4A illustrates the design of the tubes of the CTR system. Both types of tubes can be positioned. The outer tube can be positioned by translation, the inner tube by translation and rotation, as shown in Fig. 1.4B. During surgery operation, the activation unit for positioning the tubes is mounted outside of the patient's body. The laser probe is placed in the inner ablation tube. The laser fiber directed in this way then creates an ablation object that destroys the target tumor tissue. In order for the operation to be successful, it is necessary to monitor the thermometry, the ablated tumor tissue and the position of the robot in the brain, which is provided by an MRI scanner [3]. Therefore, this robotic system is compatible with MR. The design of the manual control unit is described in detail in [9].

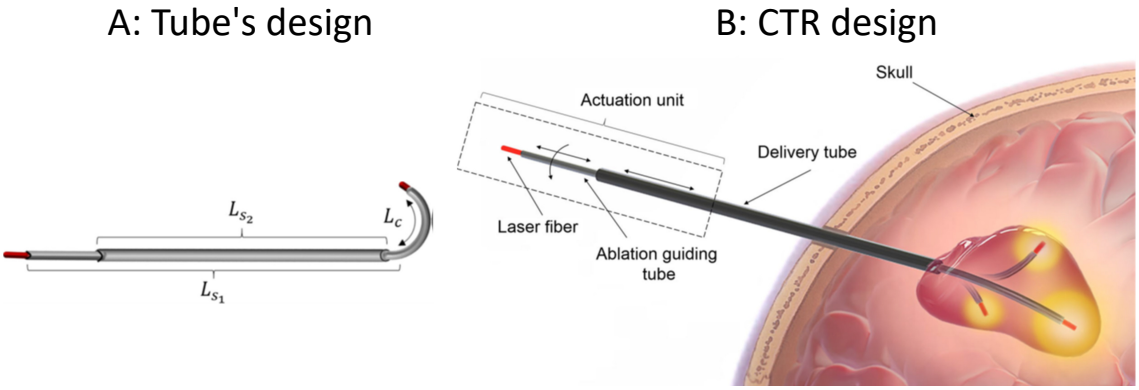

Fig. 1.4: CTR design [3].

As already mentioned, the distribution of ablation objects in the tumor volume is key to achieving good results of the surgical operation with minimal postoperative complications. A multi-objective version of the Particle Swarm Optimization algorithm, described in Chapter 2, was chosen for this task.

2 Optimization Algorithms

Optimization algorithms represent methods for finding the best solution from a set of possible solutions. In the beginning, a fitness (objective) function is defined that evaluates how appropriate the proposed solution is. The optimized system is defined by the so-called decision variables on which the fitness function depends. These can be e.g. profit, cost, production quantity, etc. The optimization process aims to find the decision space vector for that either the maximum or the minimum of the fitness function is reached [10].

The optimization problem can be either formulated as single-objective (SOOP) or multi-objective (MOOP). If the optimization problem is described by only one fitness function, it is a single-objective optimization problem, if it is described by two or more fitness functions, the problem is called a multi-objective optimization problem. In the case of MOOP, it is common for fitness functions to conflict with each other. This means that if the optimization process aims to reach the minimum of these functions, moving closer to the minimum of one function will cause the movement away from the minimum of the other function. In order for MOOP to be optimized, a trade-off is found between individual objectives. The result of the optimization process is then a set of these trade-off solutions. The full set of trade-off solutions is called the Pareto-front [10].

Today, many researchers are engaged in solving optimization problems, and many types of optimization algorithms are already known. These algorithms are generally denoted as evolutionary algorithms. The name is derived from an analogy to Darwin's theory of evolution [20], which is based on the survival of the fittest species. They are divided into genetic algorithms (such as GDE3 [11], NSGA-II [12]) and swarm intelligence algorithms (e.g. MOPSO [21] and MOSOMA [13]) [14]. The choice of the algorithm that will be used for a given optimization task depends on several factors, such as its implementation and computational complexity, the degree of reliability of finding the best solution, different experiences of users with a particular method, etc.

This thesis aims to optimize the robotic system for neurosurgery, described in Section 1.3, for which the multi-objective version of the Particle Swarm Optimization algorithm was chosen based on the decision criteria mentioned above. This chapter contains a description of the single-objective version of this algorithm (PSO), its multi-objective version (MOPSO), and finally the MOPSO algorithm for a variable number of dimensions (VNDMOPSO). At the end of this chapter, there is a clear table (Tab. 2.1) that summarizes the differences between the individual variants of this algorithm.

2.1 Single-Objective Particle Swarm Optimization

This optimization technique was developed by social psychologist James Kennedy and electrical engineer Russell C. Eberhart in 1995 [16]. It is a stochastic method that simulates the social behavior of animals, typically a swarm of bees or a school of fish. Most often, this method is explained by the behavior of a swarm of bees in search of flowers for pollination.

The task of the bees in the swarm is to find the place with the largest concentration of flowers. Along the way from the beehive to the field of flowers, each bee evaluates several positions as the potentially most densely flowered. Subsequently, bee cooperation plays an important role - each bee changes its search method according to its own experiences and the experiences of other members of the swarm. All the bees explore the entire field and then return to the places they have judged during their research so far to be the places with the greatest concentration of flowers. With increasing experience, the number of these places decreases and eventually leads to one location where the most flowers are located, as shown in Fig. 2.1 [16].

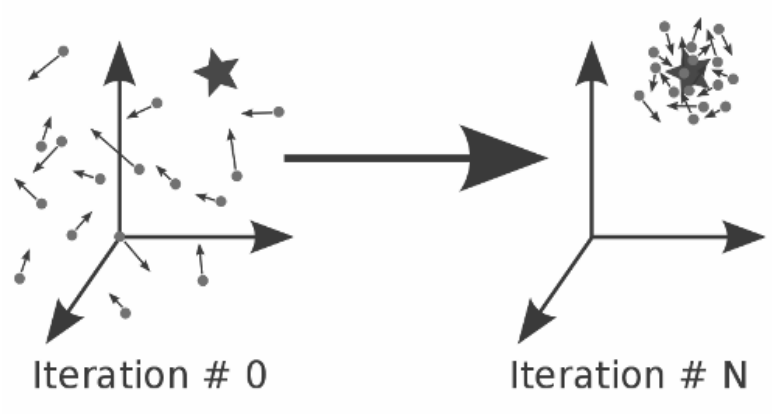

Fig. 2.1: The basic principle of the PSO algorithm [15].

The analogy with the behavior of a swarm of bees can easily be applied to the PSO algorithm. Bees represent so-called *particles* (or *agents*), a field of flowers denotes the *solution space*, the density of flowers on the field is a *fitness function*, the places that one bee evaluated as the most concentrated are called *personal bests*, and the experiences of other swarm members are called *global bests*. Each bee (particle) is located at some current *position* and has an assigned *velocity*. Exploring, exchanging experiences and returning to already explored places condition the position change in individual iterations of algorithm [16].

In each iteration i, the position and velocity vector of all agents are updated. At the beginning of the first iteration, each particle has its own randomly generated

position $\vec{x}_1$. During every iteration, its position is updated by adding the velocity vector of the given particle $\vec{v}_i$ according to equation [16]:

$$\vec{x}_i = \vec{x}_{i-1} + \vec{v}_i \tag{2.1}$$

Therefore, the velocity vector $\vec{v}_i$ must be updated first, which is calculated according to the formula [16]:

$$\vec{v}_i = w\vec{v}_i + c_1 r_1[\vec{x}_{\text{pbest},i} - \vec{x}_i] + c_2 r_2[\vec{x}_{\text{gbest},i} - \vec{x}_i] \tag{2.2}$$

It can be seen from equation (2.2) that the velocity vector of a particle is influenced by the location of the personal best of the i-th particle, but it is also influenced by the rest of the swarm, which represents the global best in the equation. Figure 2.2 shows the graphics computation of the velocity vector [16].

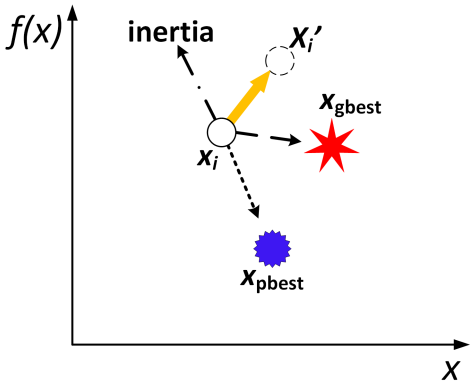

Fig. 2.2: Visual demonstration of updating the velocity vector [16].

The variables c_1 and c_2 are so-called scaling factors whose value describes how strongly the particle is attracted towards $\vec{x}_{\text{pbest},i}$ (c_1) and $\vec{x}_{\text{gbest},i}$ (c_2). The random numbers r_1 and r_2 take values from 0 to 1 and their task is to simulate to some extent the unpredictability that naturally occurs in the swarm. The variable w indicates how much the contribution of the old velocity vector will be to the new velocity vector [16]. This number describes the will of the particle to stick to its original direction.

2.1.1 PSO Algorithm

Figure 2.3 shows the pseudocode of the PSO algorithm. The user initially generates a population consisting of individual particles. These are uniquely determined by their position and velocity vector, which are randomly generated before the first iteration.

Constraint conditions are also set that the generated positions and velocity vectors must satisfy. Individual iterations of the code then consist of updating the velocity and position vectors. Subsequently, the current positions are compared with the best positions of individuals. If the current position is better than the last updated $\vec{x}_{\text{pbest},i}$ from the previous iteration, this $\vec{x}_{\text{pbest},i}$ is replaced with the new position. If not, the last $\vec{x}_{\text{pbest},i}$ is left as $\vec{x}_{\text{pbest},i}$. Finally, the position of the global best is updated, which is the same process as the $\vec{x}_{\text{pbest},i}$ update. So, if the position of the particle's new $\vec{x}_{\text{pbest},i}$ position is better than $\vec{x}_{\text{gbest},i}$, then the corresponding position is replaced by this current position [16].

Step1: Initialization
Initialize fitness function;
Initialize x_{gbest} to the maximum value of the fitness function;
for i = 1 : population size
Generate initial (random) particle position P_i. Take into account the lower and upper bounds of the search space;
Initialize the velocity vector taking into account the boundaries of the solution space.
Evaluate fitness function;
Initialize x_{pbest} to its initial position $x_{pbest}(i) = P_i$;
end
Step2: Main Loop of algorithm
for i = 1 : number of iterations
for j = 1 : population size
Update velocity;
Update position;
Update x_{pbest}: If fitness(P_j) < fitness(x_{pbest})
$x_{pbest} = P_j$;
Update x_{gbest}: If fitness(x_{pbest}) < fitness(x_{gbest})
$x_{gbest} = x_{pbest}$;
end
end
Step3: Output x_{gbest} which holds the best-found solution

Fig. 2.3: Pseudocode of the conventional PSO algorithm [16].

2.1.2 Boundary Conditions

In engineering applications, the realism of the solutions sought is often required. That is why the so-called boundary conditions to limit the solution space are established.

Three types of walls are used to delimit the space in PSO algorithm:

1. *Absorbing walls*: If a particle hits the boundary of the solution space, the absorbing type of wall will absorb the particle's energy in a specific dimension and the particle's velocity will be zeroed in that dimension. In the solution space, it then looks as if the particle would "slide" along the boundary of the space after hitting the wall, which is shown in Fig. 2.4A) [16].

2. *Reflecting walls*: After a particle hits a reflecting type of wall in one of the dimensions, the particle is reflected back into the solution space in the direction of the negative velocity vector, as shown in Fig. 2.4B). From a mathematical point of view, it only changes the sign of the corresponding velocity vector component to the opposite [16].

3. *Invisible walls*: If the solution space is not limited in any way, we are talking about the so-called invisible walls. This type of wall works on the principle of evaluating only the "feasible" position of the particles. It means that particles whose positions are already outside of the feasible decision space are not taken into account. Figure 2.4C) denotes the principle of an invisible wall [16].

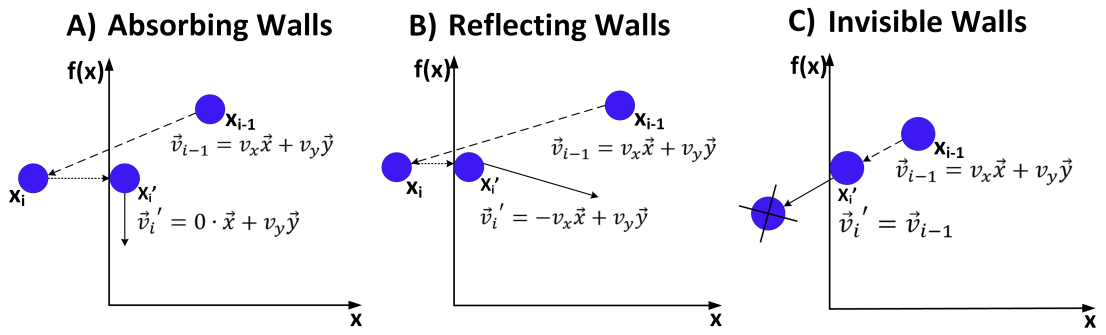

Fig. 2.4: Three types of boundary walls [17].

2.2 Multi-Objective PSO

As already mentioned at the beginning of Chapter 2, the result of a multi-objective optimization problem is not only one solution, called global best, but a set of trade-off (so-called non-dominated) solutions (created Pareto-front). Therefore, a modified, multi-objective, version of this algorithm is used instead of a conventional PSO algorithm for MOOPs.

The main goals to be achieved by the MOPSO algorithm are:

1. Ensure a sufficient number of elements of the Pareto-front,

2. Minimize the difference between the members of the found non-dominated set, which is result from the algorithm and the true Pareto-front,
3. Maximize the allocating of the found solutions along the Pareto-front so that the distribution of vectors is as smooth and uniform as possible [21].

To achieve these goals, non-dominated solutions must be found and stored in an external archive. The maximum number of archive members should be equal to the number of population members (particles). Therefore, in each iteration in which there are more non-dominated solutions in the external archive than the number of particles in the population, it must be decided which non-dominated solutions are better than the others. For the sake of the algorithms' efficiency, a certain diversity must be preserved in the non-dominated set. Solutions that are too close to each other do not carry as much information as those that are further away. Therefore, they can be removed from the archive. Finally, only the least crowded solutions remain in the external archive, and they are then considered global bests, which form the Pareto-front. For this selection, a pruning method called crowding distance is used, which is shown in Fig. 2.5 [22].

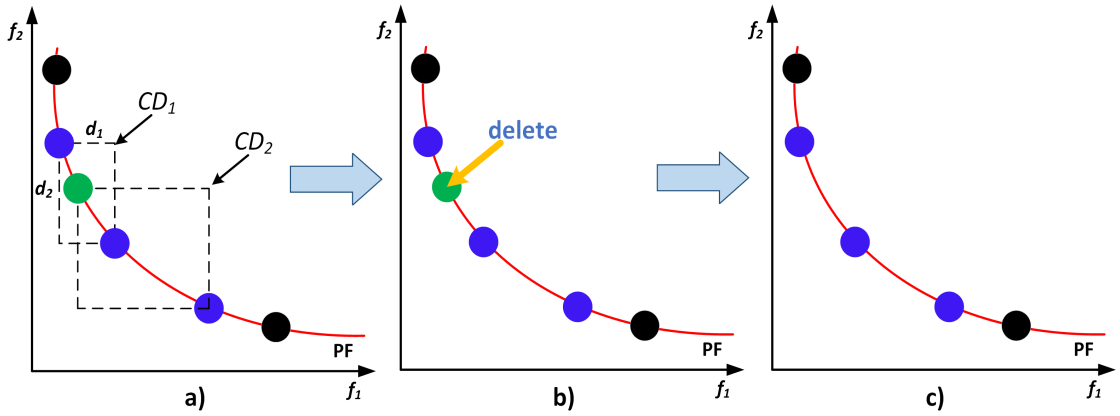

Fig. 2.5: Crowding distance method [24].

The method consists in calculating the Euclidean distances d_1 and d_2 between the solution and its two nearest neighbors. These two Euclidean distances from the nearest neighbors are multiplied by each other, and the solution that is more crowded (closer to the considered solution) is removed from the external archive [10]. Crowding distance can be calculated according to the formula 2.3 [25]:

$$\mathrm{CD} = \frac{d_1}{f_1^{\max} - f_1^{\min}} + \frac{d_2}{f_2^{\max} - f_2^{\min}} \tag{2.3}$$

However, the crowding distance method is reliably applicable only to a two-objective optimization problem. In the case of optimization with three or more

objectives, it is converted using the Equal-average Nearest Neighbor Search (ENNS) method (described in [23]) into a two-objective one, for which the crowding distance method can already be used [10], as shown in Fig. 2.6.

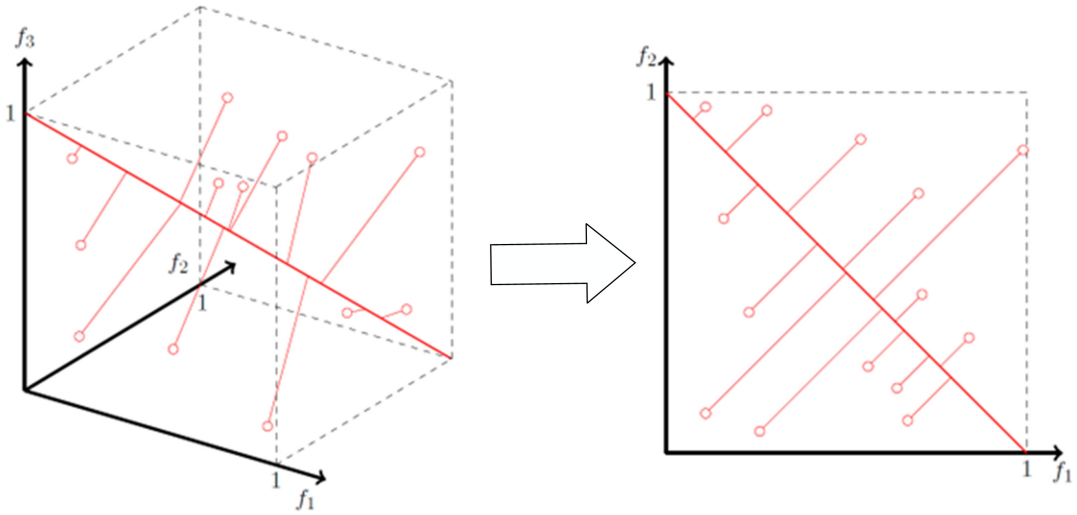

Fig. 2.6: Principle of Equal-average Nearest Neighbor Search method [10].

2.2.1 MOPSO Algorithm

After the population is created, the external archive is filled with non-dominated solutions. Then, in each iteration, a global best is chosen for each particle, to whose position it will be attracted. The following is an update of the velocity and position vector. The positions must be treated with a boundary condition and then the fitness function recalculated and the personal best updated. Next, the members of the external archive must be updated. After the first iteration, old members can be dominated by new members and they will replace the old ones, and new non-dominated solutions have also been created and should be added to the external archive. After the last iteration, the result is the contents of the external archive [26]. In Fig. 2.7 the MOPSO pseudocode is described.

2.3 MOPSO for Variable Number of Dimensions

A conventional MOPSO algorithm works only with a population of particles having the same number of dimensions. However, in some cases of optimization problems,

```
Step1: Initialization
    Initialize Population
    Initialize an external archive
Step2: Main Loop of Algorithm
    for i = 1 : Number of iterations
        for j = 1 : Population Size
            Choose a global best
            Update position
            Apply of boundary conditions
            Calculate values of criterion functions
            Calculate new personal bests
        end
        Update the contents of the external archive
    end
Step3: The result is the content of the external archive
```

Fig. 2.7: Pseudocode of the main MOPSO algorithm [26].

it is necessary that the algorithm can find the best solution even if the particles have a different number of dimensions. Therefore, the conventional MOPSO algorithm was extended to solve problems with a variable number of dimensions [18] (implementation details can be found in [19]).

The VNDMOPSO algorithm is basically the same as the conventional one, but the difference is in the number of dimensions of individual agents in the population and in the computation of the velocity vector. The number of agent's dimensions is randomly generated during population generation from a defined list of dimensions. This list must contain dimensions feasible for the given optimization problem. In the conventional MOPSO algorithm, the velocity vectors of all particles are updated in each iteration. The velocity vector of one particle depends on its position ($\vec{x}_i$), personal best ($\vec{x}_{\text{pbest}}$), and global best ($\vec{x}_{\text{gbest}}$). But the global best is always calculated from all particles, so it can have a different number of dimensions than the personal best and the position of some particles. Therefore, in the VNDMOPSO algorithm, the original MOPSO algorithm is extended by a routine that determines with which number of dimensions to work with [18].

When it is taken into account that each of the vectors $\vec{x}_i$, $\vec{x}_{\text{pbest}}$ and $\vec{x}_{\text{gbest}}$ can have a different number of dimensions (N_x, N_{pbest}, N_{gbest}), it is necessary to determine which vector will have which "weight" in the decision. For this, probabilities are chosen for the mentioned three dimensions: p_1 (for N_x), p_2 (for N_{pbest}) and p_3 (for N_{gbest}). If the vectors differ in the number of dimensions, a random number r is generated, which can take on values between 0 and 1. The number r is then compared with the numbers p_1, p_2 and p_3. In the next iteration, the N_{gbest} number

of dimensions will be used if r is less than or equal to p_1. With the number of dimensions that $\vec{x}_{\text{pbest}}$ has (N_{pbest}) will be counted if $r \in (p_1,\, p_1 + p_2\rangle$. If r is greater than the sum of probabilities p_1 and p_2, then the number of dimensions corresponding to the particle is further considered, as described in [18]. The aforementioned procedure can be expressed as follows:

$$N_x = \begin{cases} N_{\text{gbest}}, & \text{if } 0 < r \leq p_1 \\ N_{\text{pbest}}, & \text{if } p_1 < r \leq p_1 + p_2 \\ N_x, & \text{if } p_1 + p_2 < r \end{cases} \tag{2.4}$$

Thus, symbol N_x denotes a new vector dimension, and all vectors that have a different number of dimensions are either shortened or completed with random numbers. Sometimes these vectors from $\vec{x}_{\text{pbest}}$ are added to speed up the convergence of the algorithm. But the condition is that the $\vec{x}_{\text{pbest}}$ has at least as many dimensions as the new N_x.

Tab. 2.1: Differences between individual variants of algorithm

Variant of algorithm	**Result**	**Agent's dimensionality**	**pbest**	**gbest**
SOPSO	single solution	fixed	position with min f value	minimum of all pbest
MOPSO	set of Pareto-optimal solutions	fixed	position with min f value	a trade-off solutions among all solutions
VNDMOPSO	set of Pareto-optimal solutions	variable	position with min f value	a trade-off solutions among all solutions

3 Function for VNDMOPSO in MATLAB

This chapter is devoted to the description of the function for the implementation of the VNDMOPSO algorithm in MATLAB. As a basis, a simple PSO algorithm was created, which was gradually supplemented with boundary conditions, a version for a variable number of dimensions of population members, and finally modified to a multi-objective PSO algorithm. The chapter describes and explains individual parts of the created algorithm. In the following text, the names of files (black), functions (violet), scripts (red), structures (olive), structure's fields (orange), vector variables (pink), scalar variables (green) and string or logic variables (blue) will be color-coded for better orientation.

3.1 Basic Structure

The main function in the `MOPSO` folder is `VNDMOPSO`, which implements the PSO algorithm for a multi-objective optimization problem. The function is called from `main.m`, in which the individual parameters used in the function are set.

At the beginning of the `main.m` script, the `getProblem` function is called, which defines the `problem` structure. This structure contains the fitness function and the limits for the decision space variables. Other parameters, such as the maximum number of iterations, the number of population members, scaling factors, etc., are defined in `main.m` in the `params` structure. After setting the parameters, it calls the `VNDMOPSO` function and finally plots the optimization result. Figure 3.1 shows a flowchart that clearly describes the basic structure.

The function `VNDMOPSO` for performing the optimization algorithm is divided into 4 main parts: 1) Problem definition, 2) Parameters of VNDMOPSO, 3) Initialization and 4) Main loop of VNDMOPSO.

3.1.1 Problem Definition and Parameters Section

The input variables of the function are the structures `problem` and `params`. In the Problem Definition section, the fitness function called `DensityFunction`, is loaded from the `problem` structure, and the number of decision variables is defined. The Parameters of VNDMOPSO section stores the parameters set in the `params` structure in variables.

3.1.2 Initialization Section

Next, the initialization part of the function creates the population. An empty structure `empty_particle` is made, which collects all information about the particle:

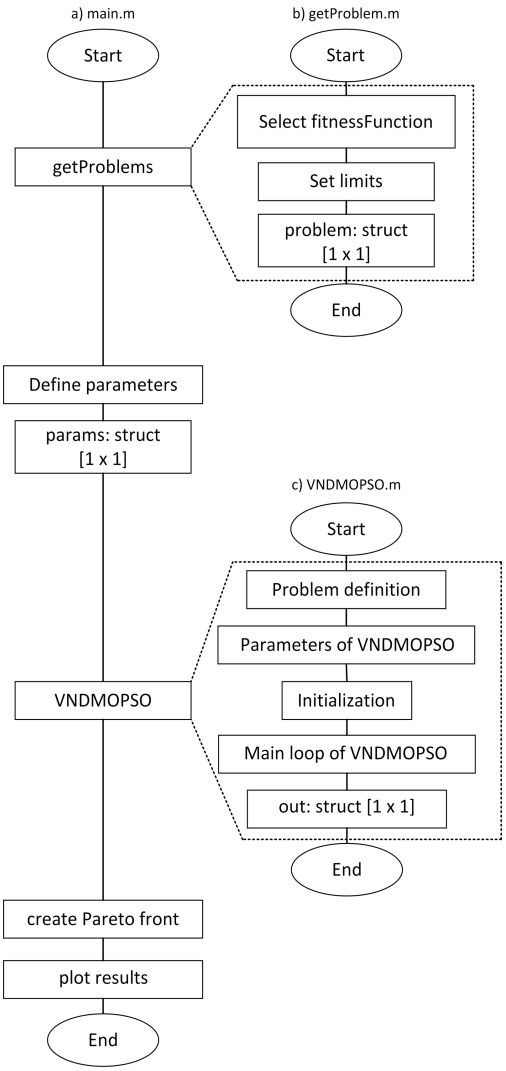

Fig. 3.1: Flowchart of a)main.m, b)getProblem.m and c)VNDMOPSO.m.

`Position`, `Velocity`, `Density` and `Best`. `Best` is also a structure itself that carries information about the personal best of the particle. It contains the best's `Position` and the best's `Density`. Initially, these parts of the structure are empty. Since the `empty_particle` structure defines a single particle and all particles are defined the same way, copies were made using the `repmat` function and thus the entire population was created. The particles copied in this way were stored in the `particle` variable, which is therefore also a structure. The global best fitness value is initialized to infinity in the next step. Subsequently, using the `initParticles` function, a random position and velocity are generated for each particle, the fitness function is evaluated, and the first personal best is created.

As explained in section 2.3, the main modification of a simple optimization algorithm to an algorithm for solving multi-objective problems consists in storing

non-dominated solutions (global bests) in the so-called external archive. So at the end of the initialization part of the function, an external archive is created.

External Archive

First of all, it is necessary to obtain non-dominated solutions, which will be stored in an external archive. This is done by the `kungEtAl` function, which sorts the solutions by the first objective function and then calls the `front` function. The `front` function divides the population into top and bottom halves so that the top half contains dominated solutions and the bottom half contains non-dominated solutions. The function decides which bottom half solutions are dominated by the top half solutions and finally returns the indices where the non-dominated solutions are stored. The `kungEtAl` function then returns these indices and the external archive is then filled with the values of the fitness function for decision space vectors with that indices using the `fillNonDomSol` function. The non-dominated solutions found in this way are then stored using the same function in the `GlobalBest` structure [29]. It contains the global best's `Position` and the global best's `Density`.

3.1.3 Main Loop of VNDMOPSO

The main part of the `VNDMOPSO.m` is the main loop of VNDMOPSO which is a for loop through all iterations, in which the velocity vector (2.2) and the position (2.1) of all particles are updated, then the fitness function is re-evaluated, and then personal best and external archive members are updated. An extension for solving problems with a variable number of dimensions and bounding the solution space using boundary conditions is also implemented in this section.

VNDMOPSO-related Variables

The code contains the extension to solve optimization problems in which the members of the generated population can have a different number of dimensions. In the initialization section, a number from the list of feasible dimensions is randomly assigned to individual members of the population using the `randperm` function. This list called `nVarsList`, can be defined by the user in `main.m` for a specific objective function. There are also chosen probabilities, used in the decision algorithm. The decision algorithm was created according to [18] described in section 2.3.

First, the variables `p1`, `p2` and `p3` are created, indicating the user-defined probabilities `GlobalBest.Position`, `Best.Position` and `Position`. The variables `newSize` and `tempV`, `tempP`, `tempGB` and `tempPB` (hereafter `tempX`) are also created. These will be used after deciding which dimension will continue to be worked

with. In the next procedure, the number of dimensions N_{gbest}, N_{pbest}, and N_{x} are compared, and if they are not the same, a random number `r` is generated, compared with `p1`, `p2` and `p3` and according to equation (2.4) a new vector of dimensions is created and stored to the `newSize` variable.

Next comes the part where the `GlobalBest.Position`, `Best.Position` and `Position` vectors must be adjusted to this new size. Prepared auxiliary variables `tempX` are used for this in the code. If the vector is smaller, a new one (`tempX`) will be created, which will have the size of the `newSize` value. The vector is filled with random numbers (with respect to the limits) and the front positions are replaced by the old vector. If the size of the old vector is larger than the `newSize` value, the new vector (`tempX`) will be the old vector, trimmed to the size of the `newSize` value. Due to less complexity, this algorithm includes only the method of padding with random numbers with respect to the limits.

So far it has been only talked about resizing the three vectors `GlobalBest.Position`, `Best.Position` and `Position`. But the position vector is updated, according to relation (2.1), by adding the velocity vector. This means that if the size of this vector was not adapted to the size of the `newSize` value, it would not be possible to calculate the new position. Therefore, in case of changing the size of the particle's position vector, the size of the velocity vector must also be adjusted.

Boundary Conditions

The user can choose which of the three types of walls - absorbing, reflecting, or invisible (see Subsection 2.1.2) - will be used to delimit the space, by selecting the `Boundary` variable in the `params` structure in `main.m`. Boundary conditions are applied in the `checkBoundaries` function, which uses a switch case condition. A check is made to see if the limit has been exceeded in any of the dimensions. The following procedure varies according to the type of walls.

For absorbing walls, if the limit has been exceeded, it sets a flag at a specific index and returns the value of the index. In the dimension with the given index, the position is then changed to the boundary and the function returns the corrected position for which the value of the fitness function is calculated. In the case of reflective walls, the procedure is similar, but the new position of the particle will not be the position of the boundary of the space, but the particle will return to the feasible space in a position shifted by the distance it flew beyond the boundary. An auxiliary variable `temp` is created in the code for this purpose, in which the distance between the old position and the space limit is stored. This distance is then subtracted from the limit position and thus the new position of the particle

is calculated. Subsequently, the position of the particle shifted in this way is the output of the function and the values of the fitness functions are calculated for it. For invisible walls, extremely high (essentially unrealistic) values of the fitness functions were simply assigned for these cases. However, this type of border wall is not very reliable.

Update of External Archive

After updating the velocity and position vector of all population members, the external archive is updated. New `combX` and `combF` vectors will be created, which will contain both old positions (`combX`) and values of the fitness function (`combF`) and new positions and values of the fitness function from the external archive. The `kungEtAl` function is then applied to the `combF` vector, which selects new non-dominated solutions and fills the external archive with them. At this moment, there can be more non-dominated solutions in the external archive than the size of the population. In that case, it has to be cut. For this, the crowding distance technique, described in section 2.2, was used. The algorithm was created according to [23]:

1. The objective space is normalized: The minimum and maximum of the function are found, and then the normalized function is calculated according to the formula:
$$f_{\text{norm}} = \frac{f - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}}. \tag{3.1}$$
2. The f_{norm} values are sorted according to the first dimension.
3. An ascending heap with f_{norm} values is created.
4. The crowding distance is calculated: The Euclidean distances from the two nearest neighbors ED_1 and ED_2 are calculated and multiplied together.
5. As long as the heap size is larger than the population size, the element with the minimum crowding distance is removed and then the heap is updated.

Finally, the `GlobalBest` will be filled with the content of the external archive. The output is stored in the `out` structure, from which an array with values for the Pareto-front is created in `main.m`. The decision space vectors of the Pareto-front are then plotted on a graph.

4 Validation

The created VNDMOPSO function should be used for planning robotic operations. So it is necessary that it can be used to solve as many and ideally as complex real-world problems as possible. Therefore, the VNDMOPSO function was tested on several test tasks, for which the so-called benchmark problems described in this chapter were used. To make the testing complete, the correctness of the found solution should be verified, e.g. by comparing the location of the found non-dominated solutions and the true ones, by checking the uniformity of the distribution of the found non-dominated solutions along the Pareto-front and other techniques. To validate the solutions found by the VNDMOPSO function, three basic test metrics were used, the description of which can also be found in this chapter.

4.1 Benchmark Problems for VNDMOPSO

To verify the functionality of the algorithm, benchmark problems from the FOPS Package [28] were used. The folder MOPSO contains five two-objective and four three-objective test problems selected from this package. For each two-objective problem, a function for calculating the true Pareto-front is also created. All test problems are described in detail in [29]. These problems are modified test problems from Deb et al.'s test suite [30] based on the method presented in [31].

This method describes how the position of the solution can be defined using two angles ($\theta(x)$ and $\theta_{\max}$). The solution is located between the extremum points of the Pareto-front. An extremum point is a point on a PF, where one of the objective functions has its minimum, while another has its maximum. Symbol $\theta_{\max}$ indicates the angle between the maximum and minimum point of the Pareto-front and $\theta(x)$ indicates the angle between the position of the solution in the objective space and the maximum point of the front [31], as shown in Fig. 4.1.

These test problems use the function getNOpt (for two-objective problems) or getNOpt3D (for three-objective problems). Each problem has the optimal number of dimensions defined by the nOptList vector, which the members of the Pareto-front should have. The functions getNOpt and getNOpt3D divide the Pareto-front into nParts parts, where in each part the order variable decides whether the dimensions are ordered either in ascending order (order = false) or in descending order (order = true) [29], as shown in Fig. 4.2. Dividing the Pareto-front into individual parts makes it possible to have a smaller angle $\theta_{\max}$. In general, parts with small values of θ converge more easily than those with large values of θ [31].

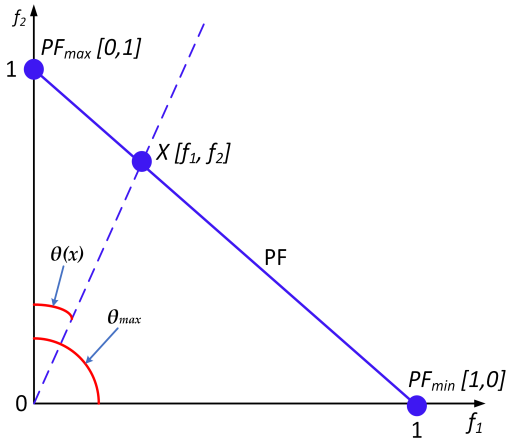

Fig. 4.1: Angle method [31].

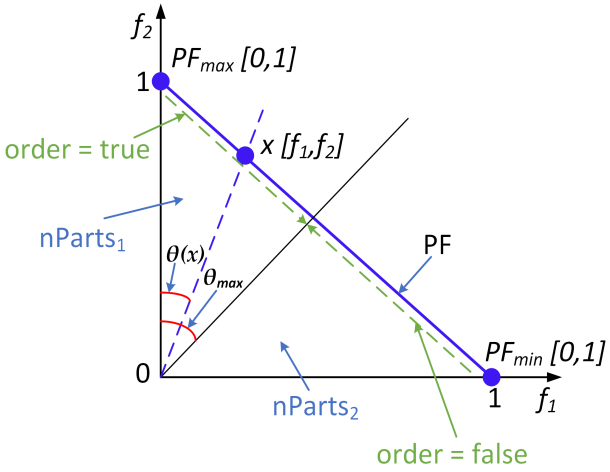

Fig. 4.2: Explanation of `getNOpt` function parameters [31].

4.2 Test Metrics

In an ideal case, the created function for solving multi-objective optimization problems should be able to find solutions as close as possible to the true Pareto-front. At the same time, it should also find solutions uniformly spread over the entire Pareto-front. To verify the correctness of the solutions found by the `VNDMOPSO` function, three test metrics were used: Generational Distance, Spread, and Hypervolume. Detailed information on individual metrics can be found in [32].

4.2.1 Generational Distance

This metric compares the position of the found Pareto-front with the position of the true one. To determine the difference in the positions of the Pareto-front solutions, the average Euclidean distance is calculated according to the equation [29]:

$$\mathrm{GD} = \frac{\sum_{i=1}^{|Q|} d_i}{|Q|}, \tag{4.1}$$

where Q describes the set of found solutions and the parameter d_i indicates the distance between the i-th solution from the found Pareto-front and the closest solution from the true Pareto-front, which is calculated as the Euclidean distance according to [32]:

$$d_i = min_{k=1}^{|P^*|} \sqrt{\sum_{m=1}^{M} \left(f_m^{(i)} - f_m^{*(k)} \right)^2}, \tag{4.2}$$

where the symbol P^* describes the set of Pareto-optimal solutions (members of the true Pareto-front), $f_m^{(i)}$ is the value of the m-th objective function of the i-th member of the set Q and $f_m^{*(k)}$ indicates the value of the m-th objective function of the k-th member of the set P^* [29]. This metric is visualized in Fig. 4.3 for a better understanding.

The metric requires knowledge of the members of the true Pareto-front. The set of members of both Pareto-fronts must be large enough to reliably compare their distance [26].

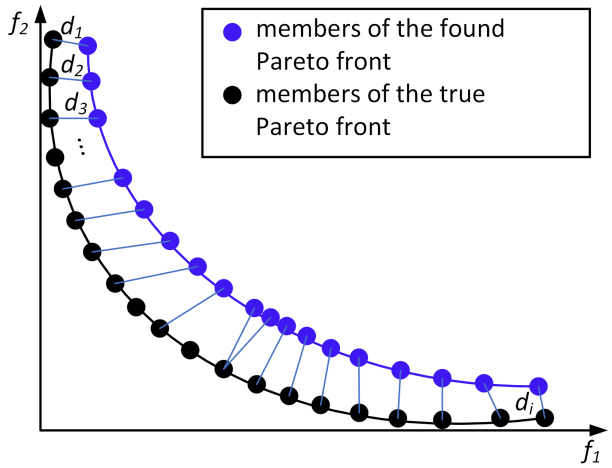

Fig. 4.3: Demonstration of the Generation Distance metric for calculating the distance of members of Pareto-fronts [34].

4.2.2 Spread (Δ)

This metric evaluates the allocation of the members of the found set of non-dominated solutions along the Pareto-front based on the mutual Euclidean distance and the distance of the extremes of the found front from the extremes of the true front. So it is necessary to know at least the extremes of the true Pareto-front. This distribution of the found solutions along the Pareto-front is defined as [29]:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} |d_i - \overline{d}|}{\sum_{m=1}^{M} d_m^e + |Q| \cdot \overline{d}}, \tag{4.3}$$

where d_i is the Euclidean distance between two neighboring solutions, $\overline{d}$ is the mean value of these distances and d_m^e is the distance between the extremes of the found and true Pareto-front of the m-th objective function. An example of the use of the metric is in Fig. 4.4.

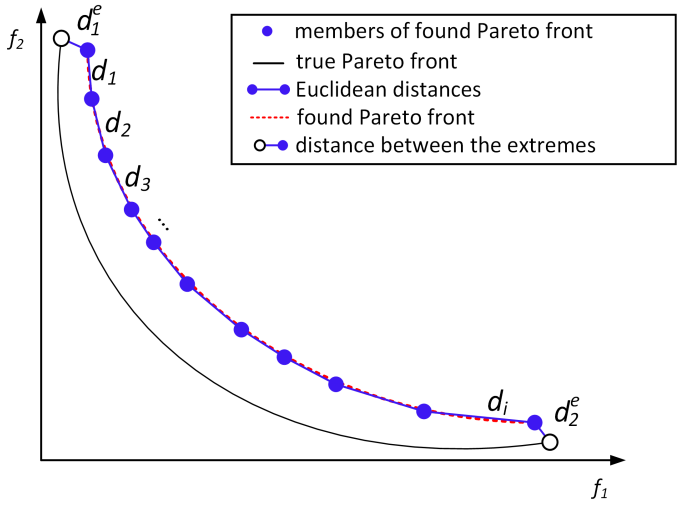

Fig. 4.4: An example of the Spread metric to calculate the distance between the members of the found Pareto-front [35].

4.2.3 Hypervolume (HV)

The Hypervolume metric combines the advantages of the two previous methods. It can compare the locations of the found and the true Pareto-front, and at the same time it also takes into account the allocation of the members of the found non-dominated set along the true Pareto-front.

The metric consists in calculating the volume (HV) of the objective space dominated by the found Pareto-front with respect to the so-called reference point. Individual hypercubes are counted, while each hypercube is given by two points - a

reference point and some member of the set of solutions of the found Pareto-front. Finally, the entire volume is calculated by the union of these hypercubes according to formula [32]:

$$\mathrm{HV} = \mathrm{volume}(\bigcup_{i=1}^{|Q|} v_i), \tag{4.4}$$

Figure 4.5 shows the total volume calculated by uniting individual hypercubes. The point $[f_{1_{\max}}, f_{2_{\max}}, \cdots, f_{N_{\max}}]$ for the N-objective optimization problem is usually called as the reference point. The larger the total calculated HV, the better the set of non-dominated solutions, forming the given Pareto-front, was found [29].

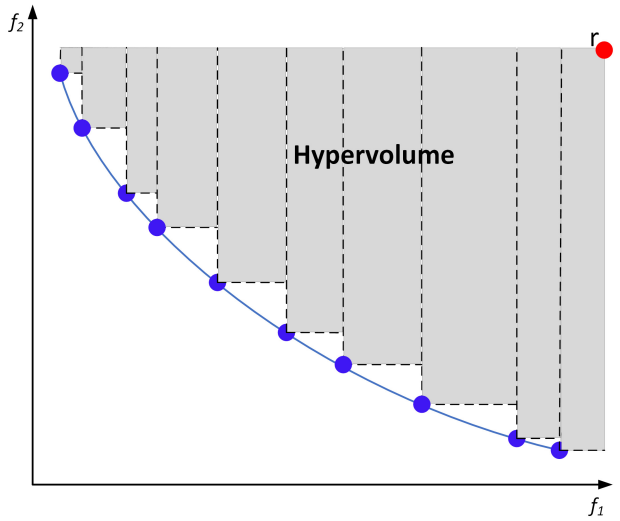

Fig. 4.5: Calculation of the entire volume from individual hypercubes above the Pareto-front using the Hypervolume metric [36].

For two-objective optimization problems, this method is easy to use because it only means the union of the rectangular areas. But for three- and more-objective problems, the calculation becomes more complicated. Therefore, the WFG method described in [33] is used, which can simplify the calculation and thus speed it up [29].

4.3 Summary of Results

The proposed algorithm was tested on all optimization benchmark problems from the `MOPSO` folder and the non-dominated solutions found were plotted in a graph and compared with the true Pareto-front (see Appendix A). Using each metric, the value of the deviation of found Pareto-front from true Pareto-front was calculated.

The setting of individual parameters of the algorithm for testing is summarized in Table 4.1. Tables 4.2, 4.3, and 4.4 contain a comparison of deviations calculated using individual metrics.

Tab. 4.1: Setting parameters for testing

Agents:	1000	**w:**	1
Iterations:	100	**Boundary:**	absorbing
c1:	2	**p3:**	0,99
c2:	2	**p1, p2:**	0,005

Tab. 4.2: Values of deviations calculated using the Generational Distance metric

Problem	**Max Value**	**Min Value**	**Average Value**
MOLI1	0,0082	0,0015	0,0041
MOLI2	0,0147	0,0042	0,0063
MOLZ3	0,0062	0,0046	0,0053
MOZDT2	$1,9808 \cdot 10^{-4}$	$2,7562 \cdot 10^{-5}$	$5,1809 \cdot 10^{-5}$
MOZDT3	$5,7930 \cdot 10^{-5}$	$4,5966 \cdot 10^{-5}$	$5,0480 \cdot 10^{-5}$

Tab. 4.3: Values of deviations calculated using the Spread metric

Problem	**Max Value**	**Min Value**	**Average Value**
MOLI1	1,0842	0,7612	0,9048
MOLI2	1,0640	0,6394	0,7635
MOLZ3	1,4793	1,4656	1,4645
MOZDT2	1,4519	1,1733	1,3499
MOZDT3	1,5450	1,4068	1,4714

The largest average deviation, calculated using the Generational Distance metric, was 0,0063. This metric calculates the distance of found Pareto-front members from true Pareto-front members. It follows that the larger the value of the deviation, the worse is the group of found non-dominated solutions. In this case, this group of non-dominated solutions was the worst found for the VNDMOLI2Fitness function.

The average deviation calculated using the Spread metric was the highest for the VNDMOZDT3Fitness benchmark problem with a value of 1,4714. For the Spread

Tab. 4.4: Values of deviations calculated using the Hypervolume metric

Problem	Max Value	Min Value	Average Value
MOLI1	0,4766	0,4376	0,4622
MOLI2	0,4853	0,4617	0,4735
MOLZ3	0,5738	0,4968	0,5353
MOZDT2	0,3300	0,3213	0,3253
MOZDT3	0,7799	0,7662	0,7775

metric, the smaller the deviation value, the better. It expresses how much the distribution of found Pareto-front members differs from the distribution with mutual Euclidean distance equal to the mean value, which represents the optimal distribution of members along the Pareto-front.

For the Hypervolume metric, the largest average value was 0,7775. For this metric, the larger the HV, the better the set of solutions was found. In this case, it was found for the test function VNDMOZDT3Fitness.

4.3.1 Effect of Probability of Dimension Change

One of the main parameters that could affect the result of the optimization is the probability that the number of decision variables of the particle will change or not. As mentioned in Section 2.3, this probability is described using three variables - p_1, p_2 and p_3.

For ten different values of the variable p_3, the values of the fitness function of the test problem were calculated for 20 iterations. Subsequently, metrics were calculated for each probability value and their deviations were plotted in the form of standard boxplots. Examples of boxplots of the parameter p_3 are shown in Figures 4.6 (for the Generational Distance metric), 4.7 (for the Spread metric) and 4.8 (for the Hypervolume metric), when the parameter was tested on the VNDMOZDT3Fitness. The effect of changing this parameter was tested on all test problems. The results are presented in Appendix B.

The results of testing the probability parameter showed a minimal impact of setting this parameter on the optimization results. Testing of influence was performed for different settings of type of wall as well as different numbers of agents and iterations. It can be seen from the boxplot figures that the results are very comparable for both low and high numbers of agents and iterations. An absorbing wall was used when rendering the images below, as well as those in Appendix B.

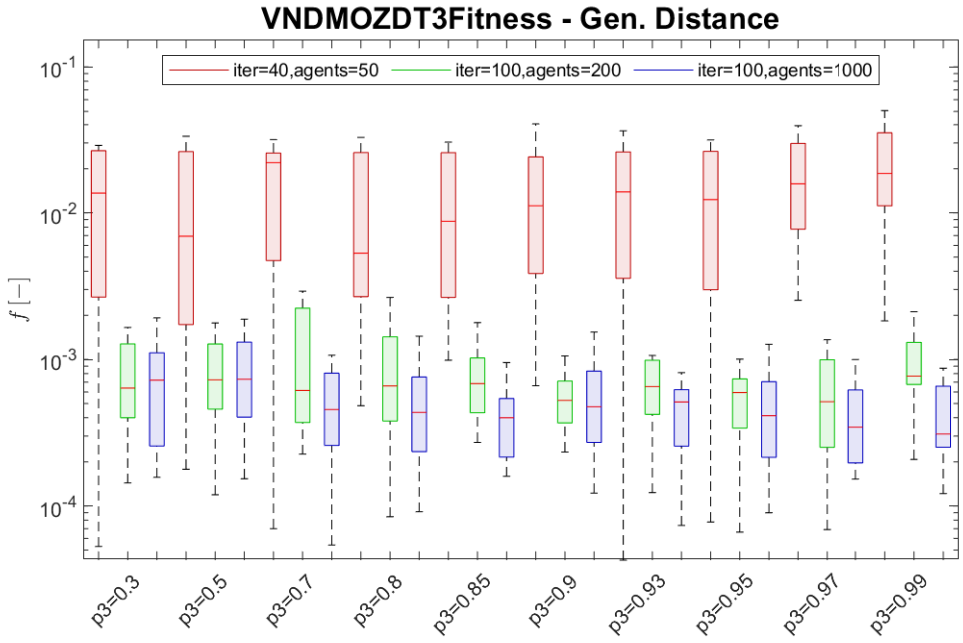

Fig. 4.6: Standard boxplots for ten different settings of the p_3 parameter for the Generational Distance metric.

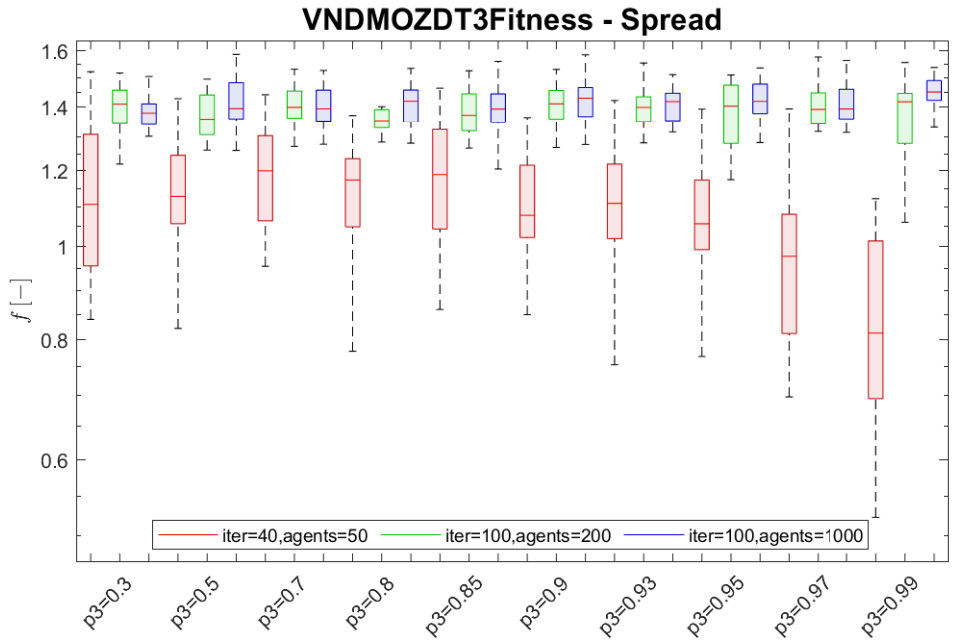

Fig. 4.7: Standard boxplots for ten different settings of the p_3 parameter for the Spread metric.

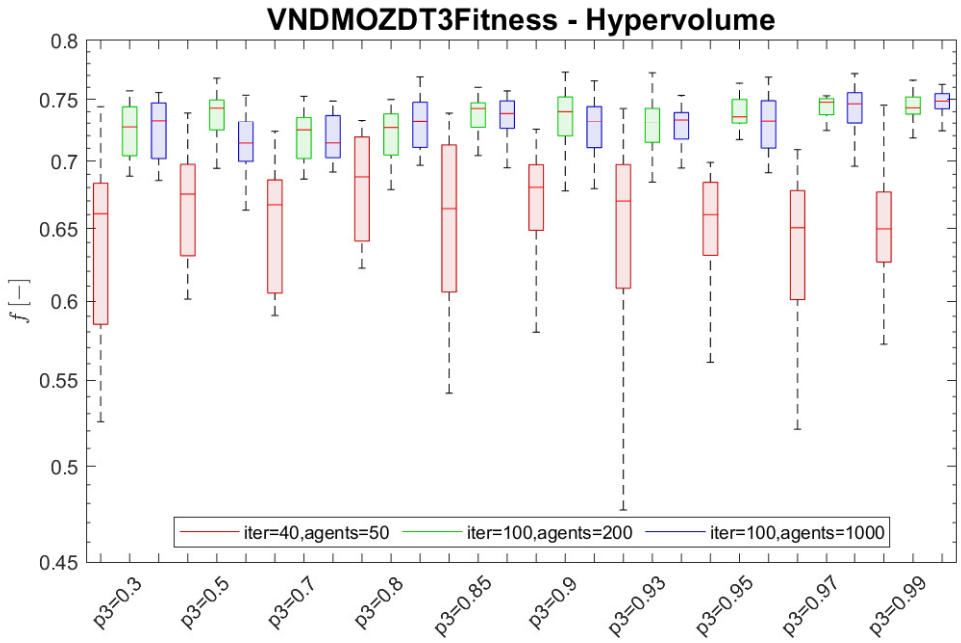

Fig. 4.8: Standard boxplots for ten different settings of the p_3 parameter for the Hypervolume metric.

5 Implementation of Real Problems

In the real world, tumors usually do not have a regular shape that can be easily removed. Therefore, the created `VNDMOPSO` function should be able to find the best possible solution for the removal of an irregularly shaped tumor. For the purpose of testing the functionality of the programmed algorithm, ten different tumor shapes were selected, for which a new fitness function `polygonFitness` was created based on the mathematical definition of the solved problem (see Section 1.2.3). Since finding a solution in 3D space would be very difficult and beyond the scope of this thesis, their representation in two-dimensional space was created. This chapter describes the selected method of obtaining 2D cross-sections of 3D geometric objects that represent real examples of brain tumors. Subsequently, the method for calculating the area in the created fitness function is described.

5.1 Geometric Representation of Tumors

Preparing the patient for the type of surgery being considered involves a magnetic resonance (MR) examination. This non-invasive imaging technology produces three-dimensional detailed anatomical images [37]. It is then possible to detect a tumor from the brain images obtained in this way. The operator can then delineate the part of the brain affected by tumor growth.

5.1.1 Delineation of the Tumor from the MRI Image

A simple script `ginputToImage.m` was created to obtain a 2D cross-section from individual MRI images of brain tumors. This script allows the user to delineate any shape on the selected image using the `ginput` function. Using this function, it is possible to get the coordinates of all the points that the user clicked on. The object selected in this way is then stored in the `polygon` variable and saved as a *.mat file with the corresponding name. All objects created by the described method are therefore represented as general polygons and stored in the `3DObjects` folder.

However, this method of obtaining the shape of the tumor introduces a certain deficiency into the solution due to the necessary approximation. It is logical that the more points used to define the given shape, the smaller this approximation error will be, and the more accurate the solution will be. On the other hand, the complexity of the calculation of fitness functions increases significantly with the number of points, and the improvement in calculation accuracy is insignificant. For this reason, it is possible to simplify the given problem in the mentioned way. A representation of

the approximate shape of the tumor obtained using the `ginput` function is shown in Fig. 5.1.

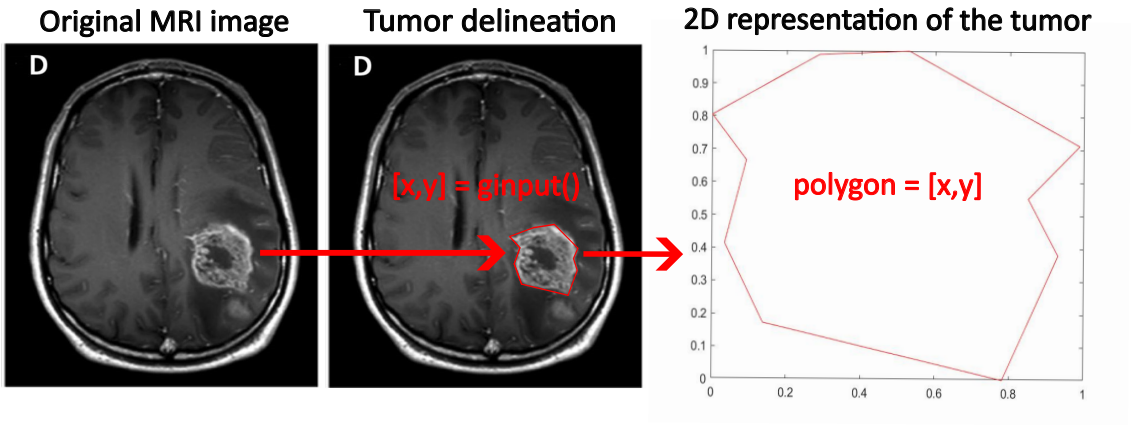

Fig. 5.1: Obtaining a 2D representation of the tumor using ginputToImage.m [41].

5.2 Fitness Function

The `polygonFitness` function is an objective function for calculating the areas by which the solved problem is described - covered/remaining and overlapping. Since the polygon is actually a 2D cross-section of the real shape of the tumor, 2D ablation objects, i.e. circles, are also considered. In order to meet the considered criteria, it is necessary to determine the area of the tumor, the area of the circles, the intersection of the area of the tumor and of the circles, and the area of the individual circles with each other. The Monte Carlo integration method, described below, is used to calculate the areas.

5.2.1 Computation of Fitness Function

The method consists of covering a certain bounded space with a large number of randomly distributed testing points. Subsequently, it is determined which testing points lie in the desired part of the space (in the tumor, outside the tumor area, in the intersection of the circles, etc.).

In the case of a circle, calculating the area it covers is simple - just find out the distance of the selected point to the center of the circle and compare it with its radius. For this, the function `euclideanDistanceBetweenTwoSets` is used, which returns a matrix of calculated Euclidean distances between the corresponding testing points and centers of circles. If this distance of a specific testing point to the center

of a specific circle is shorter than its radius, the point lies inside the circle, as is shown in Fig. 5.2.

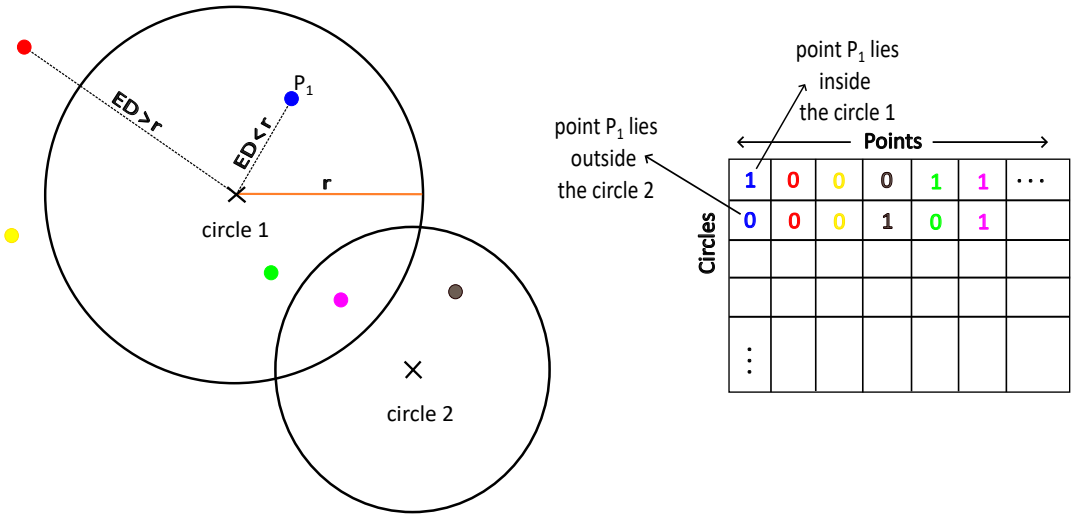

Fig. 5.2: Determining whether or not a point lies in a circle and the corresponding table.

Using the following equation, the area covered by a given object is obtained:

$$\frac{S_\mathrm{p}}{S_\mathrm{b}} \approx \frac{n_{S_\mathrm{p}}}{n_{S_\mathrm{b}}} \tag{5.1}$$

where S_p represents the area of the object, S_b the area of the bounding box, n_{S_p} is the number of points located in the object area and n_{S_b} is the total number of random points.

The larger the selected number of points, the more accurate the area estimate. An example of area calculated this way is shown in Fig. 5.3.

However, the polygon, the shape that represents the tumor, is not any known regular shape, so determining whether or not a point lies inside the polygon is more difficult. The `arePointsInPolygon` function was used for this purpose. It uses one of the two main options for solving this problem - obtaining the value of the so-called *winding number*. If the winding number is non-zero, the point lies inside the polygon. The principle of the algorithm is explained in detail in [38]. The function then returns a vector of ones and zeros depending on whether the given point lies inside the polygon or not. In Figure 5.3 shows the area of the polygon covered by the points detected in this way.

The principles described above are subsequently applied for the calculation:

1. The area of the tumor that the individual ablation objects cover/do not cover (Fig. 5.4 left)

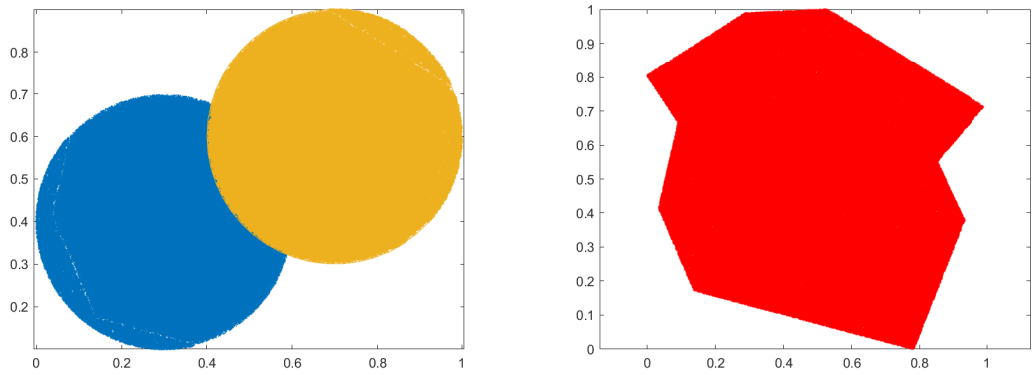

Fig. 5.3: Area of circles and polygon covered with points.

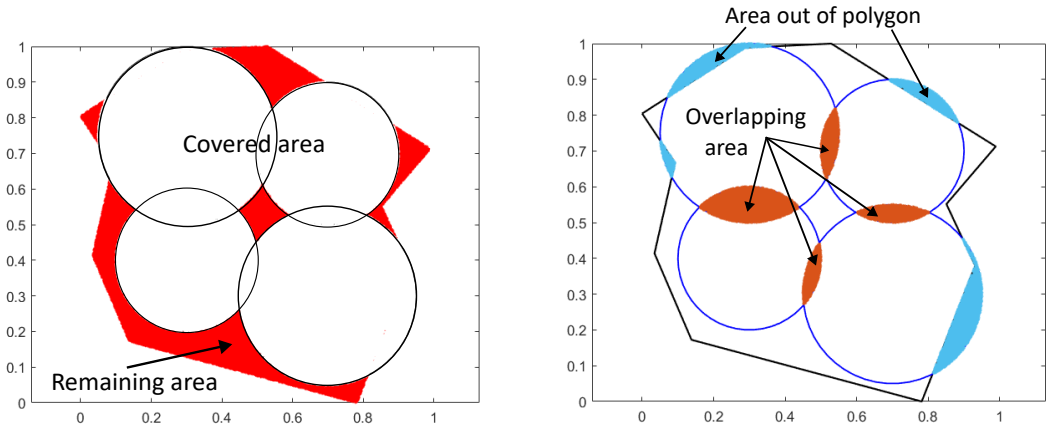

Fig. 5.4: Remaining, overlapping and out of the polygon area.

2. The area of the tumor in which the ablation objects overlap
3. The area of ablation objects that protrude from the tumor area (Fig. 5.4 right)

The task of the created optimization algorithm is then to find the best possible compromise between the values of these two functions.

6 Applying the Algorithm to Various Tumor Shapes

The main goal of this thesis is to apply the chosen optimization algorithm to various 2D structures that could represent a real tumor and to show its functionality in solving these cases as well. This chapter describes the options for setting the optimization function, subsequent processing of the results, and presents a tool for planning neurosurgical operations performed by robots.

6.1 Algorithm Setting Options

Even before starting the algorithm itself, it is necessary to set the parameters for which the algorithm will perform the calculation. Unlike the functionality testing on benchmark problems (see Chapter 4), the algorithm is not run using the `main.m` script, but from `mainPolygon.m`, where the user can find the parameter settings. Their setting is key to achieving the desired result.

It is mainly:

1. `polygonName` - selection of the shape of the tumor, represented by a polygon
2. `nVarsList` - represents a list (or number) of dimensions. Since the optimization algorithm finds a trade-off between coverage and overlap of circles, `nVarsList` will give the maximum possible number of these circles. Each circle is then described by the x and y coordinates of its center and then its radius. This means that if the user demands that the algorithm find solutions for example for one to ten circles, it is necessary to set `nVarsList` from three to thirty with a step of three because each circle is given by three parameters.
3. `MaxIter` - setting the maximum number of iterations
4. `PopSize` - setting the number of agents (particles)
5. `Boundary` - setting the type of wall (see Subsection 2.1.2) for delimiting the solution space

After setting the necessary parameters, the algorithm can be started to optimize the selected problem. The calculation time may vary depending on the set parameters. By default, two runs of the algorithm are set.

6.2 Results of Optimization

After the completion of the optimization process, the results are stored in the `out` structure. The most important information is contained in the `exArchive` structure, which stores the positions of the agents (in the field `Position`) and the values of

the fitness function (in the variable Density). The values of the fitness function are then stored in the form of a matrix in the paretoFront variable.

The Position variable contains three types of information:

1. x-coordinate of the center (cell centers_x) - 1st, 4th, 7th, etc. position in the Position vector
2. y-coordinate of the center (cell centers_y) - 2nd, 5th, 8th, etc. position in the Position vector
3. radius (cell radius) - 3rd, 6th, 9th, etc. position in the Position vector

The variables paretoFront, centers_x, centers_y and radius are stored in a *.mat file named PF_1 (first run) and PF_2 (second run) in the selected location.

6.2.1 Comparing Results

The two optimization runs are important for the possibility of comparing the results due to the randomness of the input data. So the files PF_1.mat and PF_2.mat are then compared. The script named compareResult.m is used for this. The user chooses polygonName (the name of the polygon) and Boundary (the type of wall used during the calculation). After starting, the optimization results from the first and second runs of the algorithm are compared. The script works on the principle that the values of the fitness function of both solutions are combined, and the non-dominated solutions are selected and stored in the bestPF.mat file, which contains both fitness function values and information about circles. In this way, the best non-dominated solutions across all solutions can be selected. Finally, it is determined how many non-dominated solutions in the new Pareto-front come from the first run and how many from the second run. The selection of the best non-dominated solutions from two Pareto-fronts is shown in Fig. 6.1.

6.3 Tool for Displaying Results

A graphical user interface (GUI) called GUIforTumorCoverage was created to visualize the optimization results, which serves as a tool for planning robotic neurosurgical operations. It allows the user to view the distribution of individual circles in the polygon and thus decide on the robot settings suitable for a particular patient.

At the beginning, polygonName and Boundary are selected again, and after starting, a graphical interface is displayed in which a specific solution can be selected either by choosing an index in the roller in the upper middle or by clicking on the corresponding point in the Pareto-front graph (left graph). In the right figure, the polygon and the distribution of circles in its area will then be displayed. In the middle, the percentage of coverage of the polygon and the percentage of overlap of

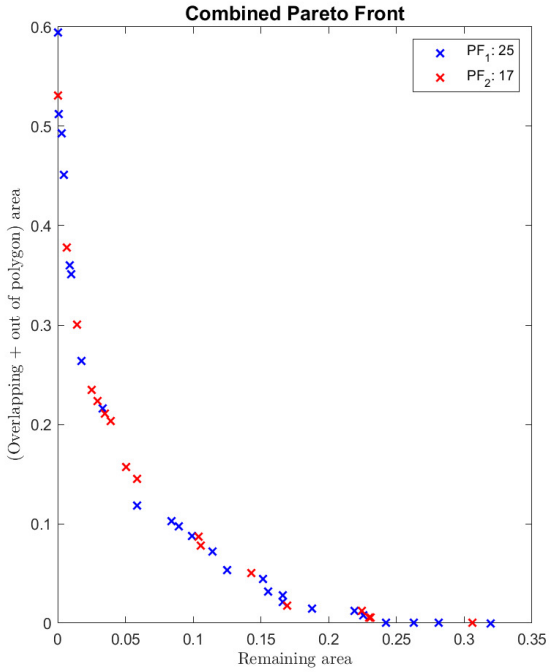

Fig. 6.1: A demonstration of combining two runs of the algorithm and selecting the best solutions from both.

circles or protruding from the polygon are displayed. Information about the location of the individual circles and their radius is then available in the table in the middle below. A description of the workspace and a visual demonstration of the GUI is shown in Fig. 6.2 and Fig. 6.3.

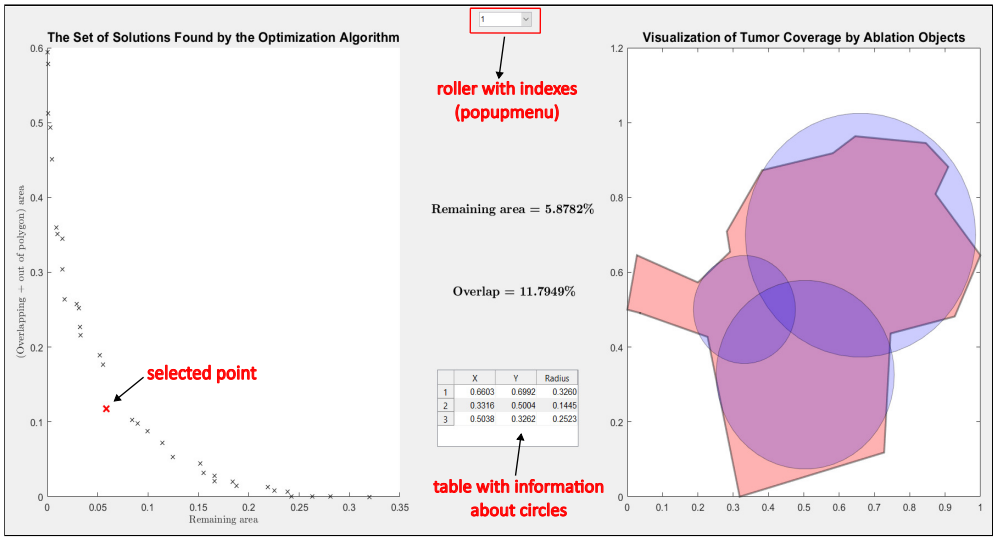

Fig. 6.2: Description of GUI workspace.

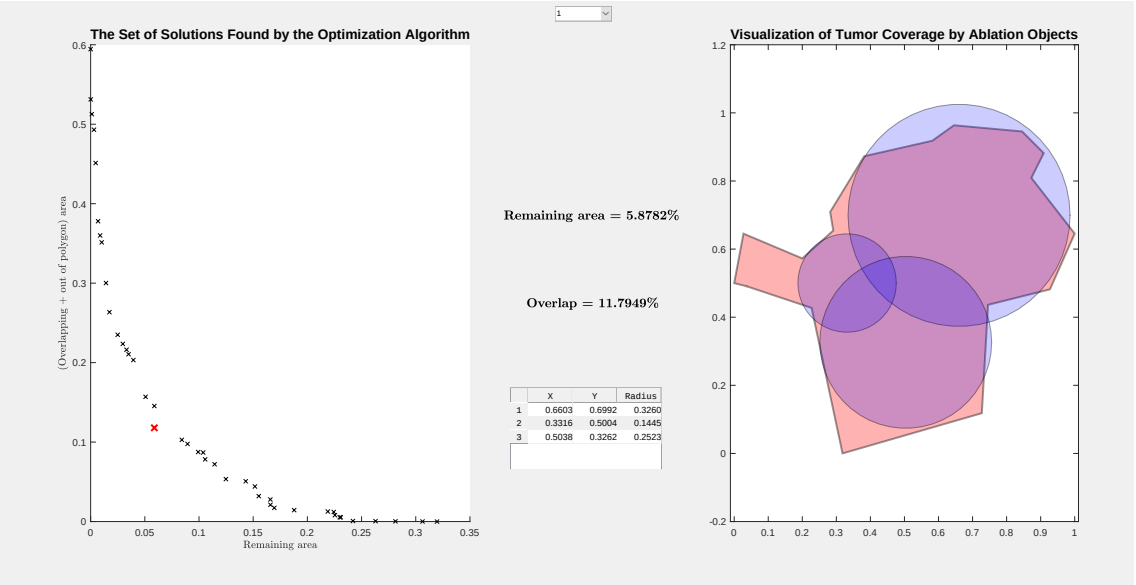

Fig. 6.3: GUI demonstration.

7 Evaluation of Optimization Results

For each optimizer, its quality should be evaluated, i.e. how efficiently it can find a solutions for different settings of the selected parameters for which the optimization will be performed. Some methods can also be used to compare two optimizers against each other.

This chapter is devoted to a general evaluation of the optimization results, observation of the influence of the setting of individual parameters on the results obtained by the created optimizer, and a subsequent discussion regarding the possible improvement of its performance. This VNDMOPSO optimizer is then compared to a simple MOPSO algorithm that would use a fixed dimension setting.

7.1 The Influence of the Type of Boundary of the Solution Space

Each of the three boundary options has its advantages and disadvantages. Based on research, reflecting wall is the best in terms of convergence (i.e. how well the algorithm converges). The results, summarized in the table below (Tab. 7.1), show that the used optimization tool can find the best set of solutions when choosing an absorbing wall. It depends on the specific tested problem, but in general the algorithm was able to find a set of solutions for both absorbing and reflecting wall types very similarly. Images of all tested problems (P1-P10) can be found in the `3DObjects` folder of the electronic appendix.

When evaluating results of this type, however, the user cannot only look at numbers but solutions must also be evaluated visually, i.e. how much is the proposed distribution of circles in the area of the tumor suitable depending on the comfort of the patient, the duration of the procedure, the power that was burned during the procedure, etc.

7.2 Effect of Number of Iterations and Agents

There are several ways to look at obtaining solutions by an algorithm. One of them is obtaining solutions based on the speed of convergence. With two different settings, when the fitness function is calculated the same times, by choosing the ratio of the number of iterations and agents, it is possible to set the algorithm either so that the number of agents exceeds the number of iterations, or, on the contrary, when the setting of the number of iterations is greater than the number of agents. In the first case, the algorithm will search the solution space well using a large number of

Tab. 7.1: The influence of the type of boundary for more agents than iterations

PARAMETERS: Agents = 300, Iterations = 100							
Polygon	Wall	max		min		best	
		f_1[%]	f_2 [%]	f_1 [%]	f_2 [%]	f_1 [%]	f_2 [%]
P1	absorbing	34,80	57,40	0	0	10,05	10,80
	reflecting	37,20	80,18	0	0	12,03	12,71
	invisible	54,50	60,32	0,56	0	17,94	17,13
P2	absorbing	34,40	46,87	0	0	11,02	11
	reflecting	35,15	68,57	0,04	0	11,70	11,60
	invisible	43,97	65,73	0,30	0,01	15,23	12,17
P3	absorbing	23,68	47,77	0,05	0	7,83	7,99
	reflecting	31,96	59,40	0	0	8,94	9,77
	invisible	39,78	72,07	0,10	0	9,60	9,88
P4	absorbing	24,42	56,69	0	0	7,41	7,28
	reflecting	20,69	65,20	0	0	7,87	8,15
	invisible	36,38	52,40	0,91	0,01	10,73	7,80
P5	absorbing	35,85	52,10	0	0	10,28	10,85
	reflecting	42,30	76,30	0,31	0	13,10	12,10
	invisible	48,73	55,80	2,30	0,01	15,59	14,91
P6	absorbing	22,50	35,25	0	0	7,70	7,60
	reflecting	36,28	65,65	0	0	9,57	9,54
	invisible	55,13	69,85	0,38	0,02	12,09	14,42
P7	absorbing	33,42	32,72	0	0	10,23	10,14
	reflecting	33,20	52,47	0	0	10,79	11,39
	invisible	40,70	58,51	0,23	0	14,67	12,90
P8	absorbing	24,50	45,98	0	0	5,70	5,32
	reflecting	22,50	53,83	0	0	6,18	6,68
	invisible	37,40	75,82	0	0	4,98	7,38
P9	absorbing	20,10	31,34	0	0	5,09	5,04
	reflecting	22,46	39,98	0,10	0	7,16	6,62
	invisible	28,62	45,30	0,40	0	8,77	7,77
P10	absorbing	27,66	35,43	0	0	7,84	7,60
	reflecting	26,98	41,89	0	0	9,45	8,80
	invisible	39,41	68,71	0,04	0	7,96	12,30

agents, but it will converge faster, so it will not have enough time to choose a global best of good quality. The second case has the opposite problem when the algorithm

will be able to select the best agent as the global best, but it will not have enough information about the solution space due to the low number of agents.

Some problems did not achieve satisfactory results when the ratio of number of agents and iterations was initially set, so this ratio was reset to exactly the opposite. It can be seen from Table 7.2 that the ratio of percentage of coverage and overlap has improved for some problems. In addition, visual results for very difficult problems, which the algorithm could not solve with the previous settings, were solved relatively more successfully with the new parameter settings, which is also shown in Figure 7.1.

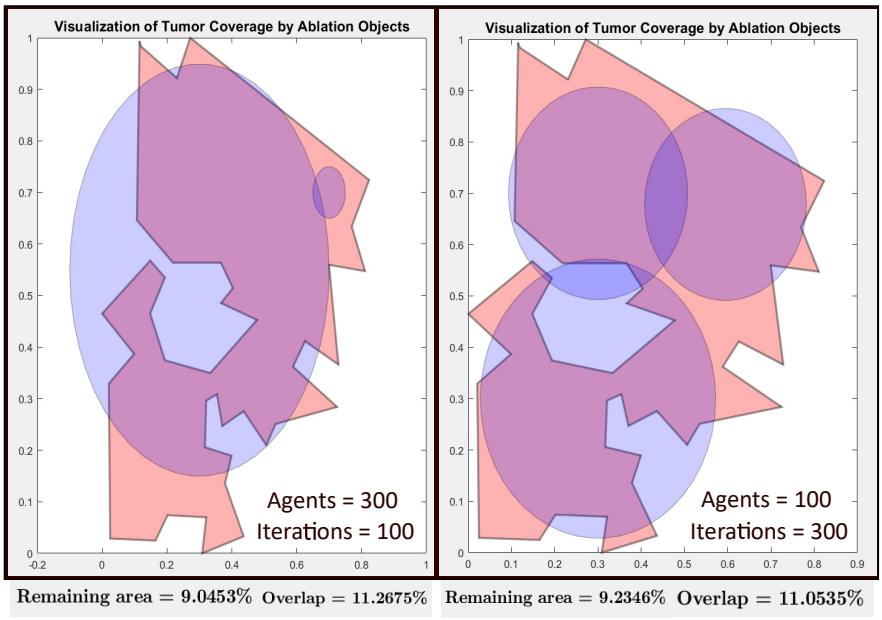

Fig. 7.1: After changing the parameters, it was possible to find good solutions even for very complex tumor shapes, such as P2 (upper) and P5 (lower).

Tab. 7.2: The influence of the type of boundary for more iterations than agents

PARAMETERS: Agents = 100, Iterations = 300							
Polygon	Wall	max		min		best	
		f_1[%]	f_2 [%]	f_1 [%]	f_2 [%]	f_1 [%]	f_2 [%]
P1	absorbing	31,47	66,31	0	0	10,27	10,31
	reflecting	38,70	77,20	0,03	0	12,52	13,00
	invisible	61,14	71,17	1,15	0	14,92	22,30
P2	absorbing	34,73	48,80	0	0	9,98	10,52
	reflecting	33,70	65,18	0,07	0	11,13	10,92
	invisible	43,81	55,92	0,44	0,01	16,80	17,24
P3	absorbing	25,36	50,16	0	0	7,95	7,71
	reflecting	25,10	58,28	0	0	7,49	8,20
	invisible	51,41	58,01	0,98	0,01	12,15	13,25
P4	absorbing	21,67	55,87	0	0	7,43	7,54
	reflecting	21,72	60,95	0	0	7,27	7,94
	invisible	36,82	63,44	0,48	0,01	9,76	11,28
P5	absorbing	38,20	48,14	0	0	9,47	9,88
	reflecting	37,92	70,34	0,19	0	11,54	11,92
	invisible	49,04	63,75	1,71	0,02	18,55	17,79
P6	absorbing	22,48	35,30	0	0	7,54	7,60
	reflecting	32,28	59,75	0	0	8,20	9,85
	invisible	55,34	63,30	3,69	0,01	17,55	18,93
P7	absorbing	31,91	33,28	0	0	10,39	10,52
	reflecting	36,53	50,65	0	0	10,78	10,78
	invisible	46,55	47,17	0,31	0	13,99	15,18
P8	absorbing	23,10	47,80	0	0	5,30	5,43
	reflecting	27,85	60	0	0	5,56	5,50
	invisible	31,55	57,32	0,26	0	10,80	5,93
P9	absorbing	20,40	31,05	0	0	5,18	4,69
	reflecting	22,32	38,42	0,07	0	6,17	6,06
	invisible	28,40	47,40	0,70	0	9,14	9,52
P10	absorbing	27,90	33,74	0	0	7,41	7,53
	reflecting	28,08	38,80	0	0	8,45	8,86
	invisible	39,38	67,45	0,018	0	9,81	7,13

7.3 Impact of Tumor Shape Approximation

As already mentioned in Chapter 5, the real shapes of the tumors were approximated, so the calculation of the area covered by the circles is imprecise. However, the error it introduces into the overall result is small, as shown in Fig. 7.2. For this purpose, a much more accurate tumor model was created, when 61 points were used for delineation. It can be seen that the same solution in the Pareto-front corresponds to a distribution of circles with almost the same coverage and overlap as when the shape was much more simplified.

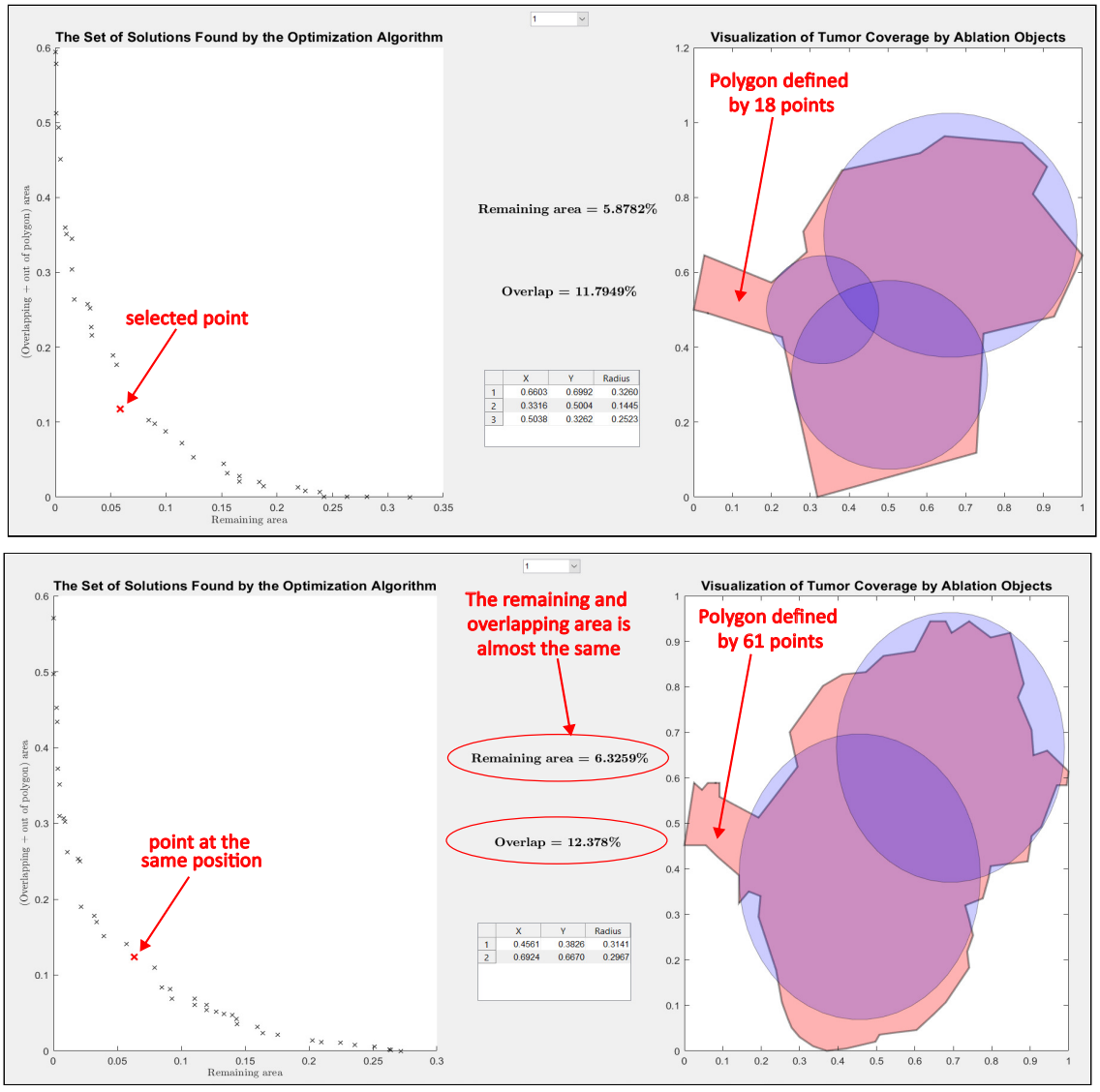

Fig. 7.2: Demonstration of minimal impact of tumor shape approximation on area calculation (tested on P3).

7.4 Ideal Distribution of Ablation Objects for Complete Removal of the Tumor

For this type of surgery, neurosurgeons have to trust incomplete or inaccurate data produced by imaging techniques such as magnetic resonance or computed tomography, where tumor detection is not always accurate. The images contain information about the brain tumor collected before surgery. Meanwhile, the brain tissue may have moved and the size of the tumor may also have increased [39]. Therefore, it is more appropriate to choose such a distribution of circles, in which the circles protrude to a certain extent from the tumor. In this case, the tumor would definitely be removed entirely. The first priority is to remove the tumor tissue, but it is necessary to consider that the brain tissue does not have the ability to regenerate and its degradation is irreversible [39]. An example of the distribution of ablation objects in the tumor area for the maximum possible removal of tumor tissue is shown in Fig. 7.3. It can be seen in the picture that for the selection of such large and thus positioned ablation objects, essentially 100% coverage would be achieved, while their size is not much larger than the tumor itself. This would make it possible to remove e.g. changes in tumor size.

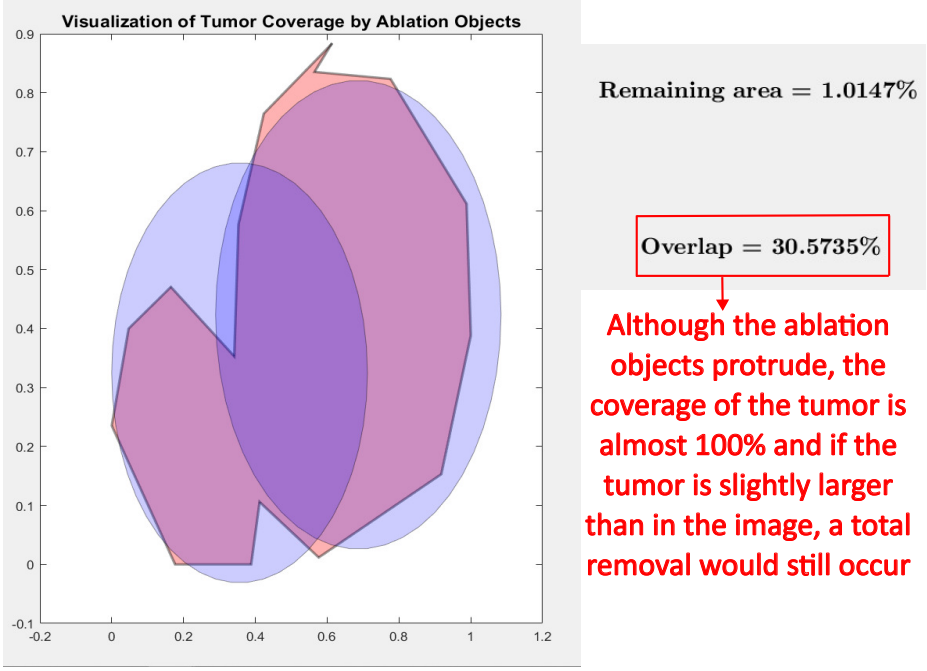

Fig. 7.3: Ideal distribution of ablation objects.

7.5 Comparison of VNDMOPSO and Simple MOPSO Algorithm

Multi-objective PSO for variable number of dimensions is the most complex version and it should be proven that its use was necessary. Therefore, a comparison test of the version of the MOPSO algorithm with a fixed (FND) and variable number of dimensions was performed. The method used for this test is called Dominance Ranking.

This method consists in combining two Pareto-fronts and their dominance. Using both types of optimizer, the values of the fitness function are calculated, then their Pareto-fronts are combined into a common one, and non-dominated solutions are selected from it. According to how many non-dominated solutions from which Pareto-front were selected, it is then decided which of the two original Pareto-fronts has more better solutions [40].

Figure 7.4 shows that the test results were in favor of the VND version of this algorithm for absorbing and reflecting walls. For an invisible wall, the algorithm with fixed dimensions worked better, as shown in Figure 7.5.

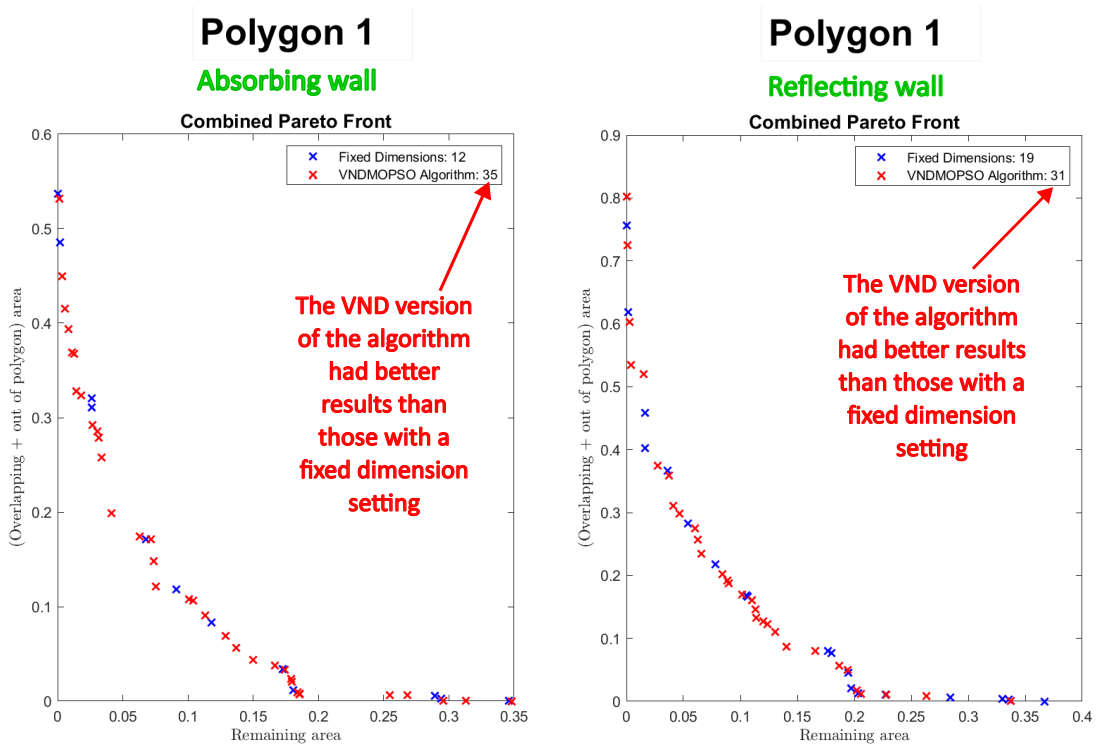

Fig. 7.4: Comparison results of VND and FND versions of the algorithm for absorbing and reflecting wall.

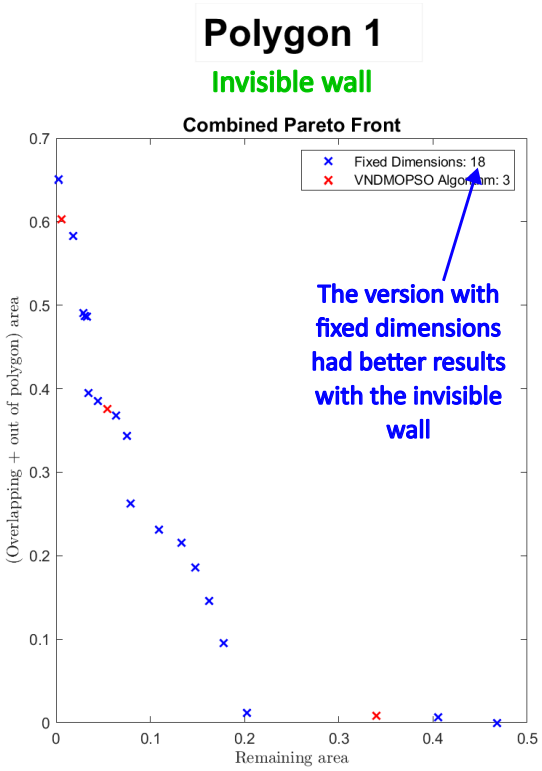

Fig. 7.5: Comparison results of VND and FND versions of the algorithm for invisible wall.

Since the invisible wall is the worst approach in terms of convergence, the MOPSO algorithm with a variable number of dimensions seems to be the right choice, both in terms of computational time and better results. An overview of the number of non-dominated solutions originated from the Pareto-front obtained by the VND variant and the FND variant of the algorithm is available in Tab. 7.3.

Tab. 7.3: Comparison of VND (Variable number of dimensions) and FND (fixed number of dimensions) version of the algorithm

Polygon	Wall	Number of solutions	
		PF_{VND}	PF_{FND}
P1	absorbing	35	12
	reflecting	31	19
	invisible	3	18
P2	absorbing	81	8
	reflecting	42	8
	invisible	6	26
P3	absorbing	50	13
	reflecting	35	9
	invisible	6	23
P4	absorbing	60	21
	reflecting	42	13
	invisible	6	28
P5	absorbing	55	9
	reflecting	35	16
	invisible	7	23
P6	absorbing	88	3
	reflecting	23	11
	invisible	5	17
P7	absorbing	64	15
	reflecting	38	11
	invisible	5	20
P8	absorbing	46	9
	reflecting	26	14
	invisible	6	13
P9	absorbing	97	11
	reflecting	34	9
	invisible	5	27
P10	absorbing	77	3
	reflecting	24	14
	invisible	4	31

Conclusion

The aim of this thesis was to get acquainted with the basic principle of performing robotic operations used in the field of neurosurgery. After an initial study of the issue, a selected algorithm for solving multi-objective optimization problems was described, which could be used to optimize surgical procedures for the removal of brain tumors by the LITT method, used by a concentric tube robot.

The Multi-Objective Particle Swarm Optimization algorithm with a variable number of dimensions turned out to be the most suitable of the available heuristic algorithms, thanks to its simple implementation and the best results in solving multi-objective problems having up to low tens of variables, which includes the problem solved in this thesis.

One of the main tasks of the thesis was the creation of a function in which this algorithm is implemented. This function, created in MATLAB, was tested on several benchmark problems and the results were verified using three test metrics.

The largest average deviation calculated using the Generational Distance metric was found for the VNDMOLI2Fitness function, the average deviation calculated using the Spread metric was the highest for the VNDMOZDT3Fitness function, and the Hypervolume metric calculated the largest average HV for the VNDMOZDT3Fitness test function. The influence of setting the probability whether or not the number of decision variables will be changed was minimal.

The thesis was further devoted to solving real problems. This part of the thesis dealt with obtaining 2D cross-sections of 3D structures, represented as general polygons, and creating a fitness function. This function is formulated to be able to calculate the area that covers or does not cover the tumor, the area of overlapping ablation objects, and the part of the area of the ablation objects that does not cover the tumor.

The created optimization function was subsequently used to optimize the coverage of ten different polygons with ablation objects. To visualize the results, a tool that displays individual solutions in the Pareto-front and the corresponding distribution of ablation objects in the tumor area was created.

The results show that the VNDMOPSO algorithm was able to find a set of solutions that could be suitable for this type of operation, while the success of finding the ideal set of solutions varies based on the setting of several parameters. Setting the solution space boundary type had the greatest impact. The algorithm was able to find the best solutions in the case of an absorbing wall, while it achieved similar success in the case of a reflecting wall. When setting up an invisible wall, the results were unsatisfactory. Different combinations of setting the number of agents and iterations also had an impact on the optimization results. When setting a higher

number of iterations than agents, the optimizer was more successful.

Finally, the optimizer with a variable number of dimensions was compared to the optimizer with a fixed number of dimensions. Using the Dominance Ranking method, it was evaluated that the VND version of the algorithm is significantly better when using an absorbing and reflecting wall. With the invisible wall, the version with a fixed number of dimensions was more successful.

However, several simplifications have been used in achieving the results, which cause them to be somewhat distorted. Of course, these simplifications could be replaced by a more accurate model. However, their usage would increase the time required for the calculation. The use of another programming language, possibly parallel processing of results, etc. could then reduce this increase in time requirements.

Bibliography

[1] *Aansneurosurgeon* [online]. [cit. 2022-12-11]. Available from URL: <https://aansneurosurgeon.org/feature/the-state-of-robotics-in-neurosurgery/>.

[2] *Ucsfhealth* [online]. [cit. 2022-11-20]. Available from URL: <https://www.ucsfhealth.org/conditions/brain-tumor/treatment>.

[3] GRANNA, Josephine, Arya NABAVI a Jessica BURGNER-KAHRS: *Computer-assisted planning for a concentric tube robotic system in neurosurgery. International Journal of Computer Assisted Radiology and Surgery* [online]. 2019, 14(2), 335-344 [cit. 2021-5-24]. ISSN 1861-6410. Available from: doi:10.1007/s11548-018-1890-8.

[4] ALFALAHI, Hessa, Federico RENDA a Cesare STEFANINI. *Concentric Tube Robots for minimally Invasive Surgery: Current Applications and Future Opportunities.* IEEE Transactions on Medical Robotics and Bionics [online]. 2020, 2(3), 410-424 [cit. 2022-11-24]. ISSN 2576-3202. Available from: doi:10.1109/TMRB.2020.3000899

[5] BAGGA, Veejay a Dev BHATTACHARYYA. *Robotics in neurosurgery.* The Annals of The Royal College of Surgeons of England [online]. 2018, 100(6 sup), 23-26 [cit. 2022-11-25]. ISSN 0035-8843. Available from: doi:10.1308/rcsann.supp1.19

[6] *History of Robotics in Medicine: Medical Uses for Robots* [online]. [cit. 2022-11-25]. Available from URL: <https://www.gwsrobotics.com/blog/history-robotics-medicine-medical-uses-robots>.

[7] Omar Ashraf and Nitesh V. Patel and Simon Hanft and Shabbar F. Danish *Laser-Induced Thermal Therapy in Neuro-Oncology: A Review* Available from URL: <https://www.sciencedirect.com/science/article/pii/S1878875018301669>.

[8] Franck P, Henderson PW, Rothaus KO. *Basics of lasers: history, physics, and clinical applications. Clin Plast Surg.* 2016;43:505-513

[9] Granna J, Graf A, Nabavi A, Burgner-Kahrs J (2017) *A manual actuation system for laser induced thermal therapy of malignant brain tumors.* In: Proceedings of the annual meeting of the german society for computer- and robot-assisted surgery, pp 125–130

[10] MAREK, Martin *Multi-objective Optimization of EM Structures With Variable Number of Dimensions: doctoral thesis.* Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Ústav radioelektroniky, 2020. 153 p. Supervised by Ing. Petr Kadlec, Ph.D.

[11] S. Kukkonen and J. Lampinen *GDE3: the third evolution step of generalized differential evolution,* 2005 IEEE Congress on Evolutionary Computation, 2005, pp. 443-450 Vol.1, Available from: doi: 10.1109/CEC.2005.1554717.

[12] Yusliza Yusoff and Mohd Salihin Ngadiman and Azlan Mohd Zain *Overview of NSGA-II for Optimizing Machining Process Parameters* Procedia Engineering journal, CEIS 2011, 15, pp. 3978-3983 Available from URL: <`https://www.sciencedirect.com/science/article/pii/S1877705811022466`>

[13] Kadlec Petr and Zbynek Raida *Multi-Objective Self-Organizing Migrating Algorithm Applied to the Design of Electromagnetic Components.* IEEE Antennas and Propagation Magazine 55.6 (0AD): 50–68. Web. Available from: doi:10.1109/MAP.2013.6781705

[14] Slowik, A., Kwasnicka, H. *Evolutionary algorithms and their applications to engineering problems.* Neural Comput and Applic 32, 12363–12379 (2020). Available from URL: <`https://doi.org/10.1007/s00521-020-04832-8`>.

[15] *Particle Swarm Optimization (PSO)* [online]. [cit. 2022-11-27]. Available from URL: <`https://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html`>.

[16] J. Robinson and Y. Rahmat-Samii *Particle swarm optimization in electromagnetics* in IEEE Transactions on Antennas and Propagation, vol. 52, no. 2, pp. 397-407, Feb. 2004, Available from: doi: 10.1109/TAP.2004.823969.

[17] S. Xu and Y. Rahmat-Samii, *Boundary Conditions in Particle Swarm Optimization Revisited*, in IEEE Transactions on Antennas and Propagation, vol. 55, no. 3, pp. 760-765, March 2007, Available from: doi: 10.1109/TAP.2007.891562.

[18] P. Kadlec et al. *Design of a Linear Antenna Array: Variable Number of Dimensions Approach* 2020 30th International Conference Radioelektronika (RADIOELEKTRONIKA), 2020, pp. 1-6, Available from: doi: 10.1109/RADIOELEKTRONIKA49387.2020.9092422.

[19] P. Kadlec and V. Šeděnka *Particle swarm optimization for problems with variable number of dimensions* Engineering Optimization, vol. 50, no. 3, pp. 382–399, 2018.

[20] Darwin, Charles, and Leonard Kebler *On the origin of species by means of natural selection, or, The preservation of favoured races in the struggle for life.* London: J. Murray, 1859. Pdf. Retrieved from the Library of Congress, Available from URL: <www.loc.gov/item/06017473/>.

[21] Sierra, Margarita Reyes and Coello C.A.C. *Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art.* International Journal of Computational Intelligence Research 2 (2006): 287-308.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan *A fast and elitist multiobjective genetic algorithm: NSGA-II* IEEE transactions on evolutionary computation, vol. 6, no. 2, pp. 182–197, 2002.

[23] S. Kukkonen and K. Deb *A fast and effective method for pruning of non-dominated solutions in many-objective problems* in Parallel Problem Solving from Nature-PPSN IX, pp. 553–562, Springer, 2006.

[24] Li, W., Meng, X., Huang, Y. et al. *Knowledge-guided multiobjective particle swarm optimization with fusion learning strategies.* Complex Intell. Syst. 7, 1223–1239 (2021). Available from: <https://doi.org/10.1007/s40747-020-00263-z

[25] Choi, and Kim *Self-Adaptive Models for Water Distribution System Design Using Single-/Multi-Objective Optimization Approaches.* Water journal, vol.11, p.1293, June 2019. Available from: doi: 10.3390/w11061293

[26] MAREK, Martin *Toolbox pro vícekriteriální optimalizační problémy: diplomová práce.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2016. 81 s. Vedoucí práce Ing. Petr Kadlec, Ph.D.

[27] Martin Marek and Petr Kadlec (2022) *Another evolution of generalized differential evolution: variable number of dimensions, Engineering Optimization*, 54:1, 61-80, Available from: DOI: 10.1080/0305215X.2020.1853714

[28] *Fast Optimization ProcedureS (FOPS)*, Czech Technical University in Prague, 2018, Available from: <www.antennatoolbox.com/fops

[29] Marek, Martin, Petr Kadlec, and Miroslav Cupal. 2020. FOPS Documentation. AToM—Antenna Toolbox for Matlab. <http://antennatoolbox.com/download_file?file=FOPS_documentation.pdf

[30] K. Deb, L. Thiele, M. Laumanns and E. Zitzler, *Scalable multi-objective optimization test problems,* Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), 2002, pp. 825-830 vol.1, Available from: doi: 10.1109/CEC.2002.1007032.

[31] H. Li and K. Deb, *Challenges for evolutionary multiobjective optimization algorithms in solving variable-length problems,* 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 2217-2224, Available from: doi: 10.1109/CEC.2017.7969573.

[32] DEB, Kalyanmoy *Multi-objective optimization using evolutionary algorithms.* Chichester: John Wiley, c2001. Wiley paperback series. ISBN 0-471-87339-x

[33] L. While, L. Bradstreet and L. Barone *A Fast Way of Calculating Exact Hypervolumes*, in IEEE Transactions on Evolutionary Computation, vol. 16, no. 1, pp. 86-95, Feb. 2012, Available from: doi: 10.1109/TEVC.2010.2077298.

[34] Tharwat, Alaa and Houssein, Essam and Ahmed, Mohammed and Hassanien, Aboul Ella and Gabel, Thomas *MOGOA algorithm for constrained and unconstrained multi-objective optimization problems.* Applied Intelligence. 48., (2018) Available from: doi: 10.1007/s10489-017-1074-1.

[35] *Metaheuristics.jl* [online]. [cit. 2022-12-11]. Available from: <`https://docs.juliahub.com/Metaheuristics/aJ70z/3.1.0/indicators/`

[36] Lwin, Khin and Qu, Rong and MacCarthy, Bart. *Mean-VaR Portfolio Optimization: A Nonparametric Approach.* European Journal of Operational Research. 260., (2017) Available from: doi: 10.1016/j.ejor.2017.01.005.

[37] *Magnetic Resonance Imaging (MRI).* National Institude of Biomedical Imaging and Bioengineering [online]. [cit. 2023-04-04]. Available from: <`https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri`

[38] HORMANN, Kai a Alexander AGATHOS. *The Point in Polygon Problem for Arbitrary Polygons.* [online]. [cit. 2023-04-04] Available from: <`https://www.inf.usi.ch/hormann/papers/Hormann.2001.TPI.pdf`

[39] *Saving brain cells during cancer surgery.* euronews.next [online]. [cit. 2023-04-19]. Available from: <`https://www.euronews.com/next/2015/10/19/saving-brain-cells-during-cancer-surgery`

[40] KNOWLES, Joshua D., Lothar THIELE a Eckart ZITZLER. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers.* TIK Report [online]. ETH Zurich, Computer Engineering and Networks Laboratory, 2006-02, (Revised version) [cit. 2023-04-29]. Available from: <https://doi.org/10.3929/ethz-b-000023822

[41] LEUNG MD, Denise, Xiaosi HAN MD, Tom MIKKELSEN MD, FRCPC, and L. Burt NABORS MD. *Role of MRI in Primary Brain Tumor Evaluation.* Journal of the National Comprehensive Cancer Network [online]. 2014, Nov, Volume 12: Issue 11 [cit. 2023-05-16]. Available from: doi: <https://doi.org/10.6004/jnccn.2014.0156

Symbols and abbreviations

c_1	Cognitive Learning Factor
c_2	Social Learning Factor
CD	Crowding Distance
CTR	Concentric Tube Robot
d_1	Euclidean Distance between the First Solution and Its Nearest Neighbor
d_2	Euclidean Distance between the First Solution and Its Second Nearest Neighbor
d_i	Euclidean Distance of i-th Solution
DOF	Degree-of-Freedom
Δ	Spread
ENNS	Equal-average Nearest Neighbor
FDA	Food and Drug Administration
FOPS	Fast Optimization Procedures
GD	Generational Distance
GDE3	Third version of Generalized Differential Evolution
GUI	Graphical User Interface
HV	Hypervolume
LITT	Laser-Induced Thermotherapy
MOOP	Multi-Objective Optimization Problem
MOPSO	Multi-Objective Particle Swarm Optimization
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
MOSOMA	Multi-Objective Self-Organizing Migrating Algorithm
n_{S_b}	Total number of random points

n_{S_p}	Number of random points located in the object area
N_{pbest}	Number of dimensions of Personal Best
N_{gbest}	Number of dimensions of Global Best
N_x	Number of dimensions of Particle
NSGA-II	Non-dominated Sorting Genetic Algorithm
p_1	Probability of dimension of Particle
p_2	Probability of dimension of Personal Best
p_3	Probability of dimension of Global Best
r	Random number
P^*	Pareto-optimal front
PF	Pareto-front
PSO	Particle Swarm Optimization
Q	Found Pareto-front
S_b	Bounding box area
S_p	Object area
SOOP	Single-Objective Optimization Problem
SOPSO	Single-Objective Particle Swarm Optimization
v_i	Velocity Vector of i-th Particle
VNDMOPSO	MOPSO Algorithm for Variable Number of Dimensions
w	inertial weight
WFG	Weighted Fast Greedy Algorithm
x_i	Position of i-th Particle
x_{gbest}	Global Best
x_{pbest}	Personal Best

List of appendices

A Results of Testing the Function VNDMOPSO

For each test problem, nVarsList, nOptList, the number of parts of the Pareto-front nParts and the order parameter were set to the optimal parameters specified in the FOPS Package. The number of iterations and the population size were adjusted as needed. Since the metrics can only calculate the difference between the found and the true Pareto-front for two-objective problems, the following figures depict the three-objective problems without the true Pareto-front.

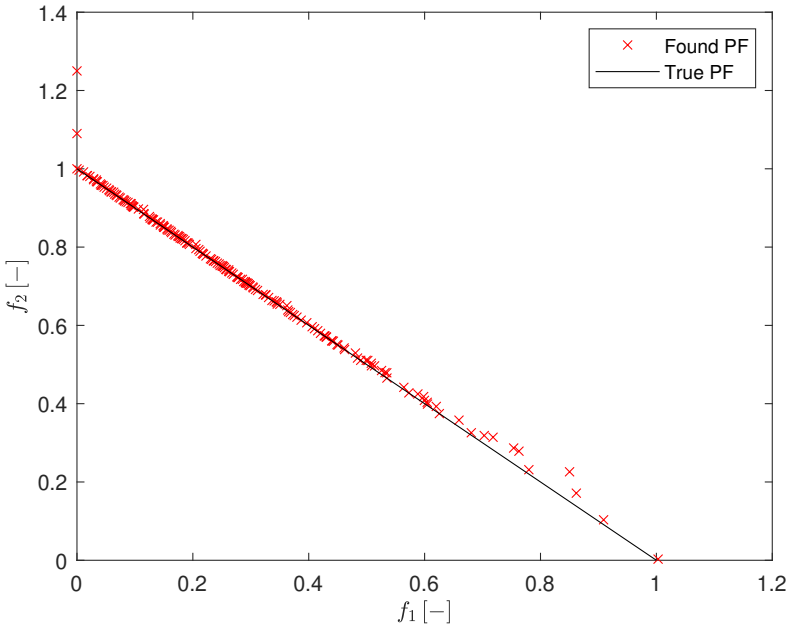

Fig. A.1: VNDMOPSO applied to VNDMOLI1Fitness (MaxIter = 100, PopSize = 1000).

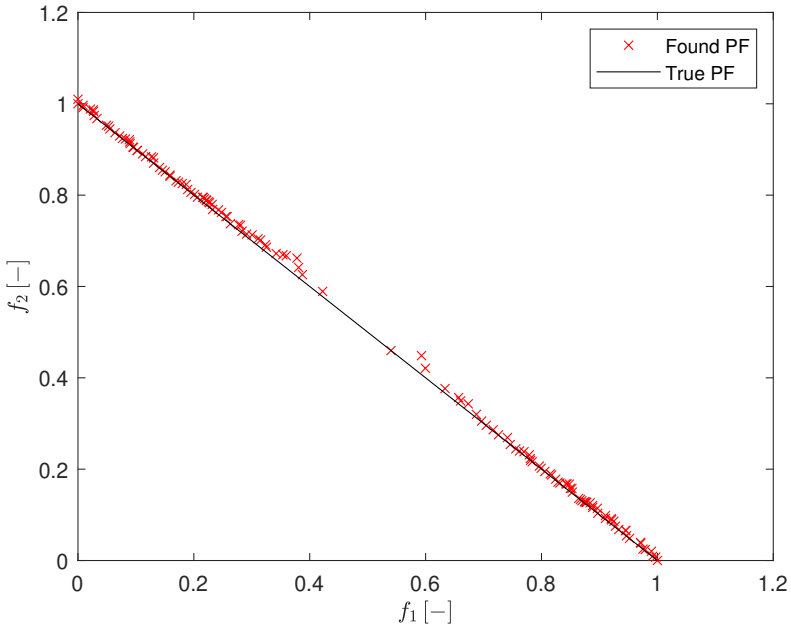

Fig. A.2: VNDMOPSO applied to VNDMOLI2Fitness (MaxIter = 100, PopSize = 1000).

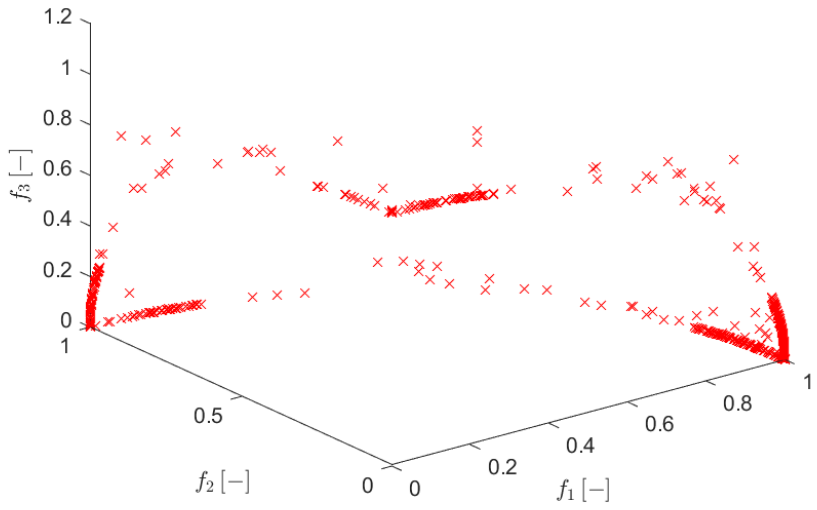

Fig. A.3: VNDMOPSO applied to VNDMODTLZ4Fitness (MaxIter = 200, PopSize = 1000).

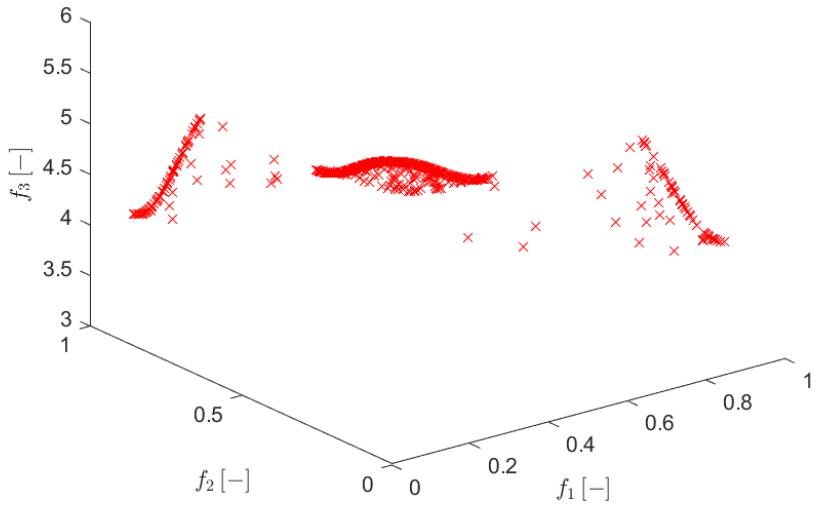

Fig. A.4: VNDMOPSO applied to VNDMODTLZ7Fitness (MaxIter = 200, PopSize = 1000).

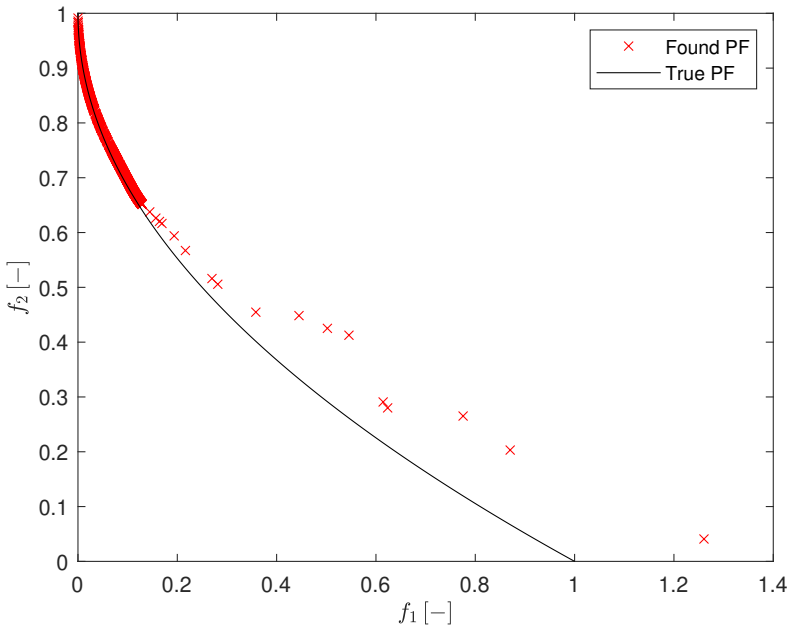

Fig. A.5: VNDMOPSO applied to VNDMOLZ3Fitness (MaxIter = 100, PopSize = 1000).

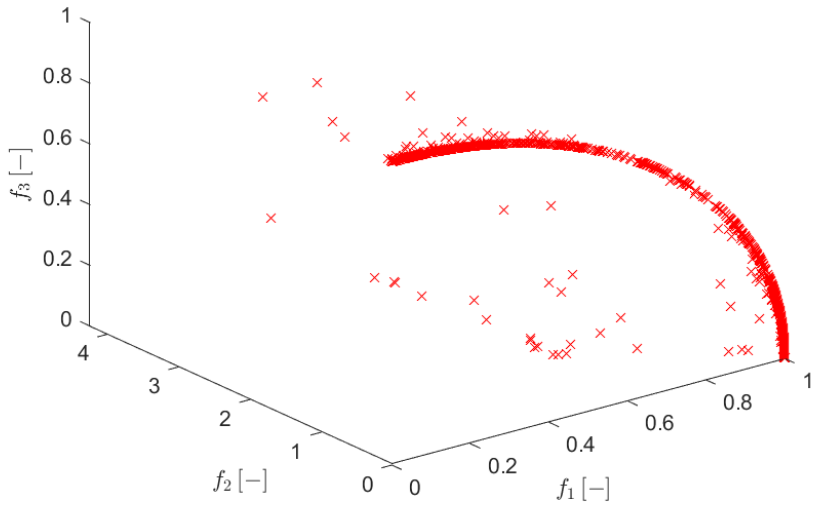

Fig. A.6: VNDMOPSO applied to VNDMOLZ6Fitness (MaxIter = 200, PopSize = 1000).

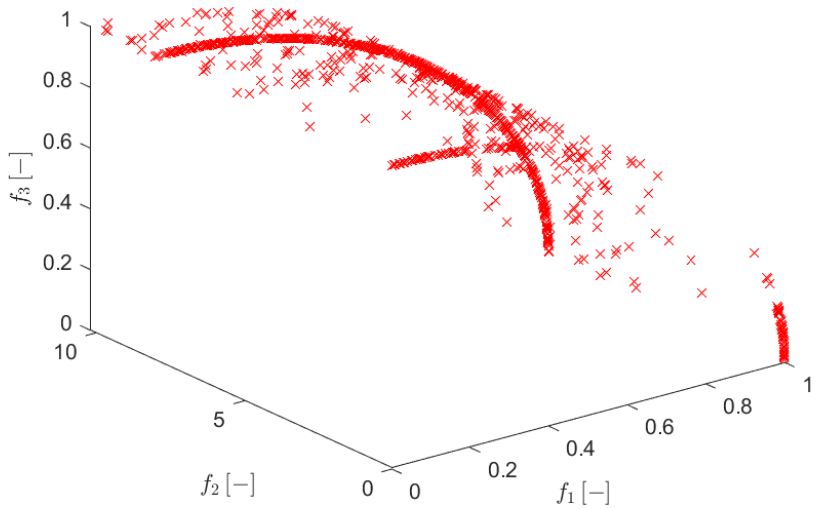

Fig. A.7: VNDMOPSO applied to VNDMOUF10Fitness (MaxIter = 500, PopSize = 1000).

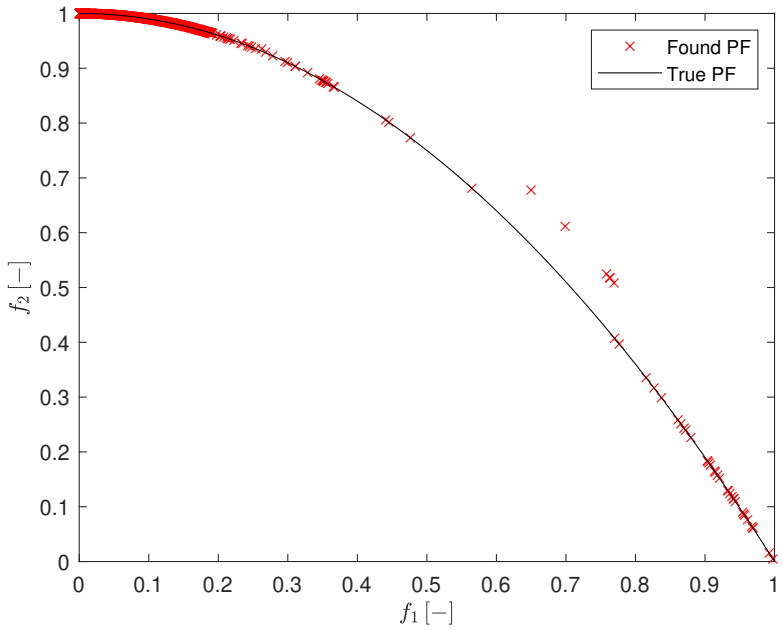

Fig. A.8: VNDMOPSO applied to VNDMOZDT2Fitness (MaxIter = 100, PopSize = 1000).

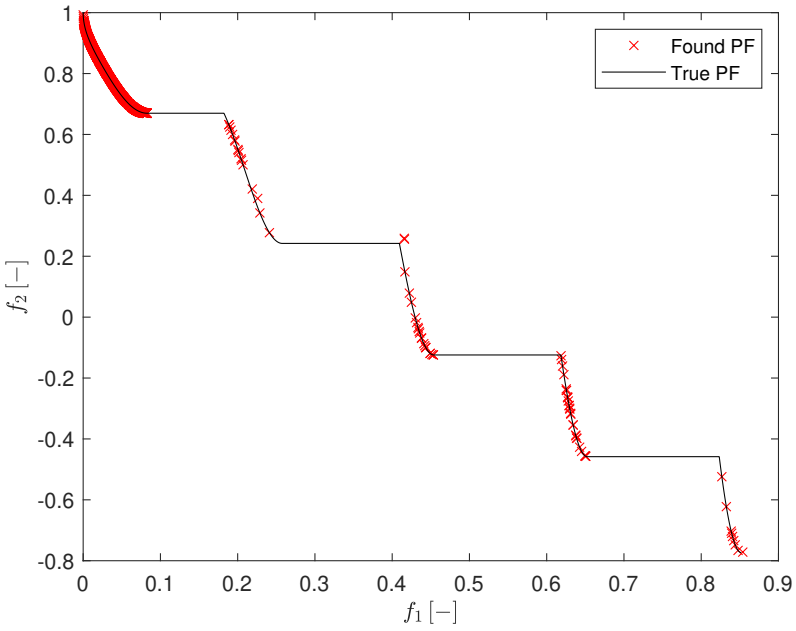

Fig. A.9: VNDMOPSO applied to VNDMOZDT3Fitness (MaxIter = 100, PopSize = 1000).

B Results of Testing the Influence of the Parameter of Probability

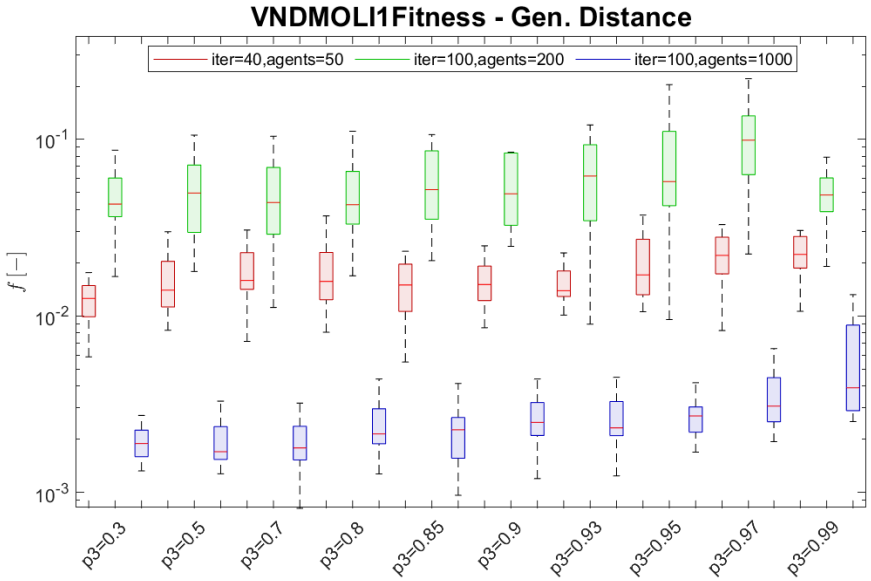

Fig. B.1: Standard boxplots for ten different settings of the p_3 parameter for the Gen.Distance metric.

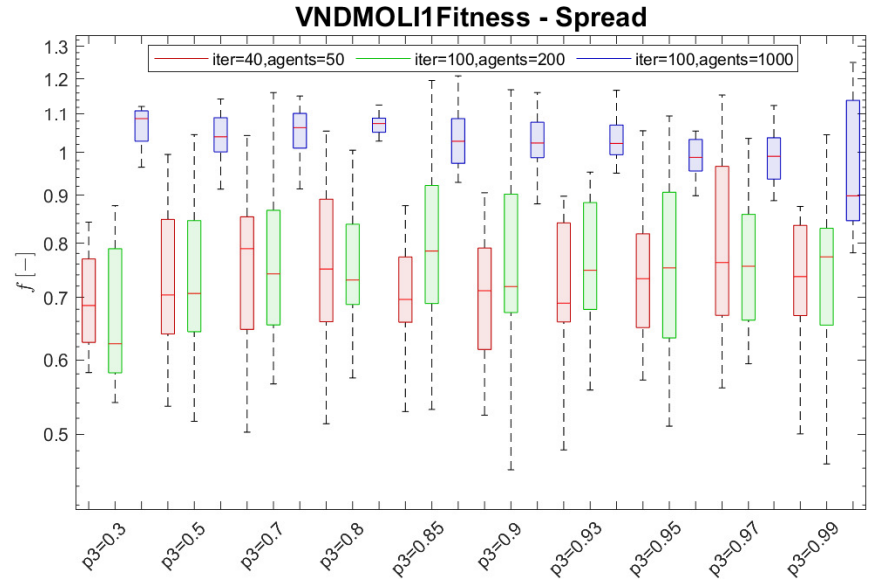

Fig. B.2: Standard boxplots for ten different settings of the p_3 parameter for the Spread metric.

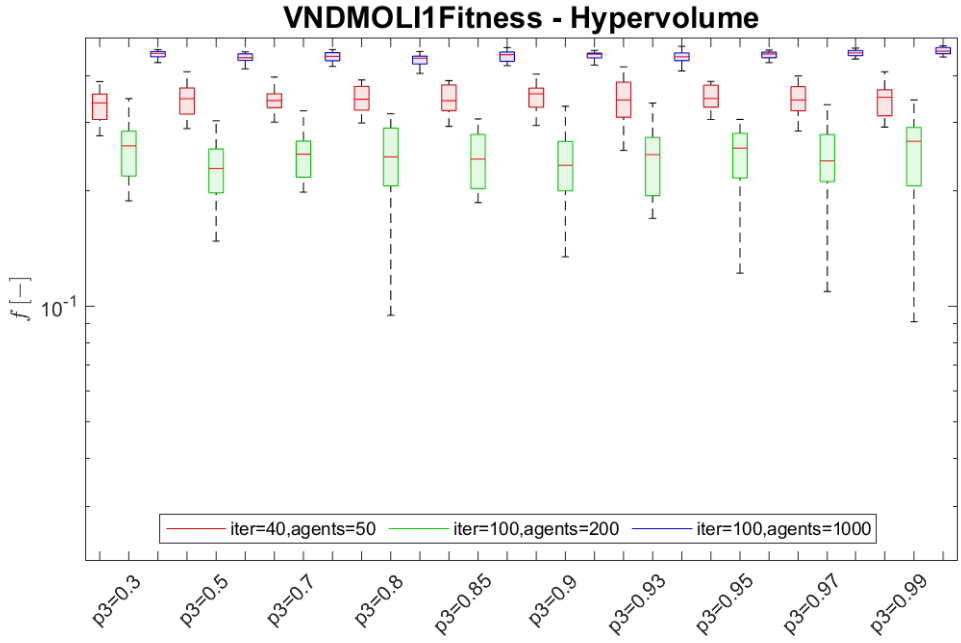

Fig. B.3: Standard boxplots for ten different settings of the p_3 parameter for the Hypervolume metric.

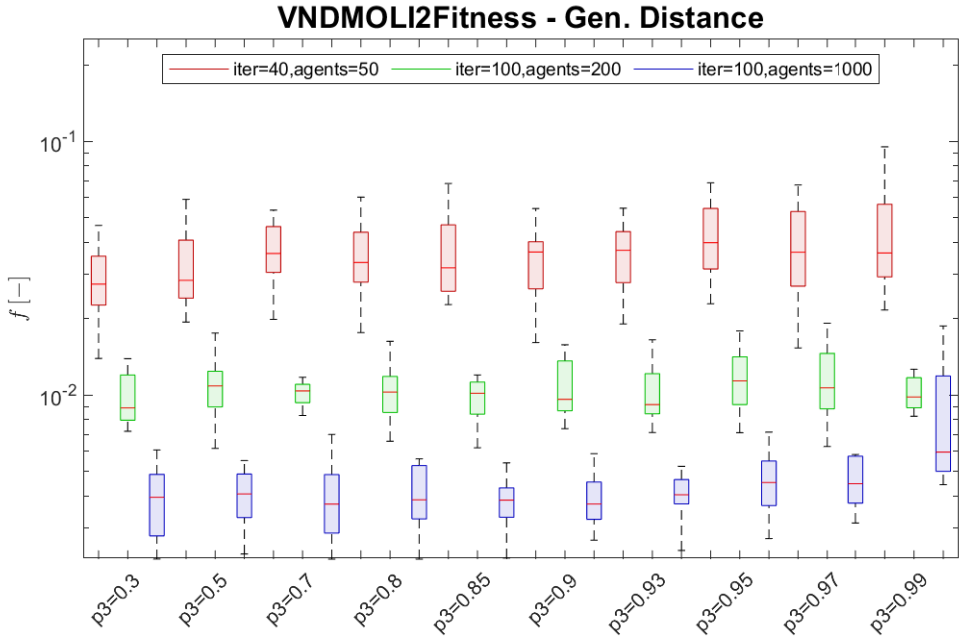

Fig. B.4: Standard boxplots for ten different settings of the p_3 parameter for the Gen.Distance metric.

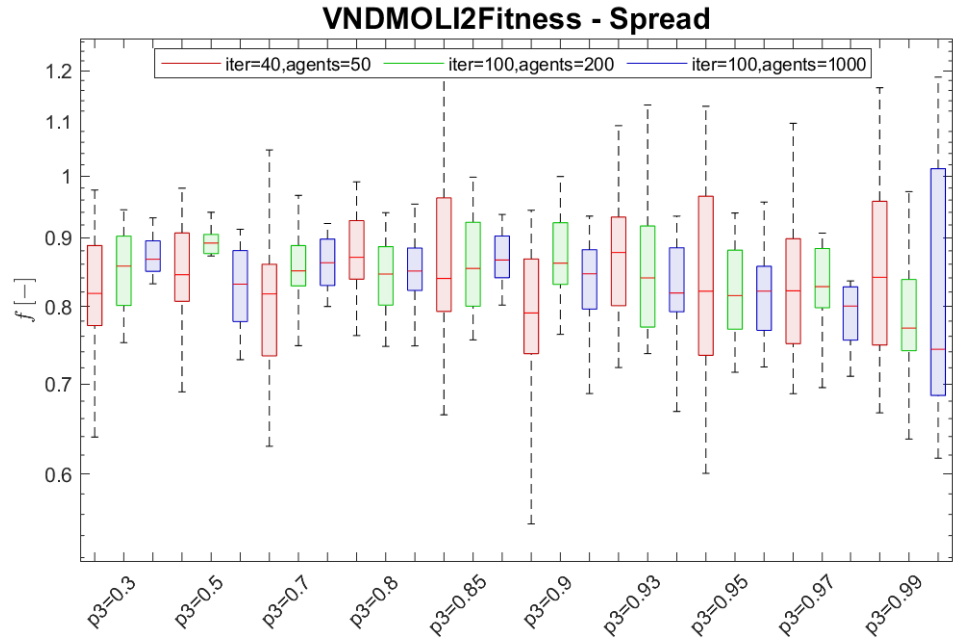

Fig. B.5: Standard boxplots for ten different settings of the p_3 parameter for the Spread metric.

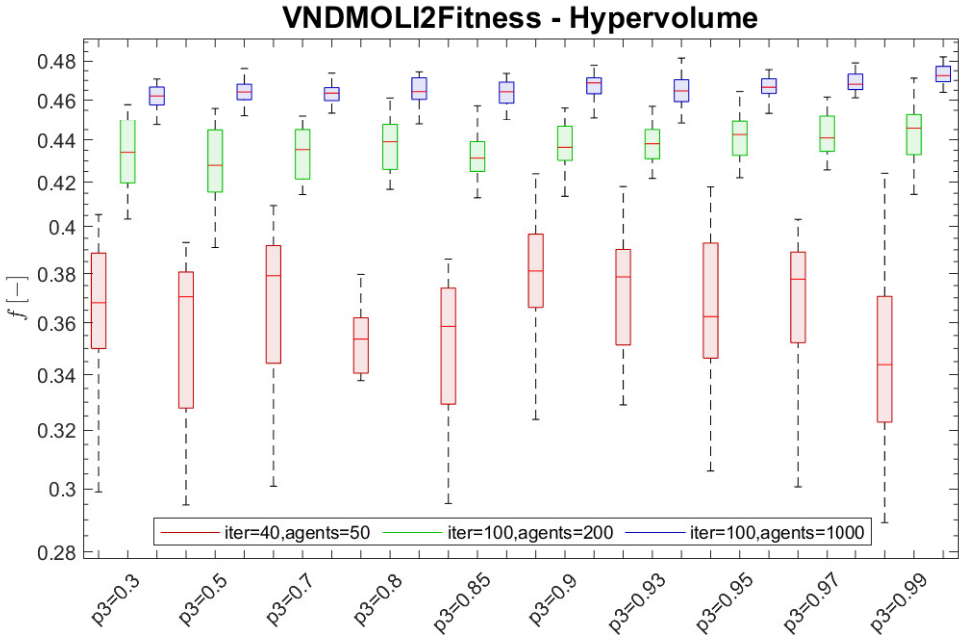

Fig. B.6: Standard boxplots for ten different settings of the p_3 parameter for the Hypervolume metric.

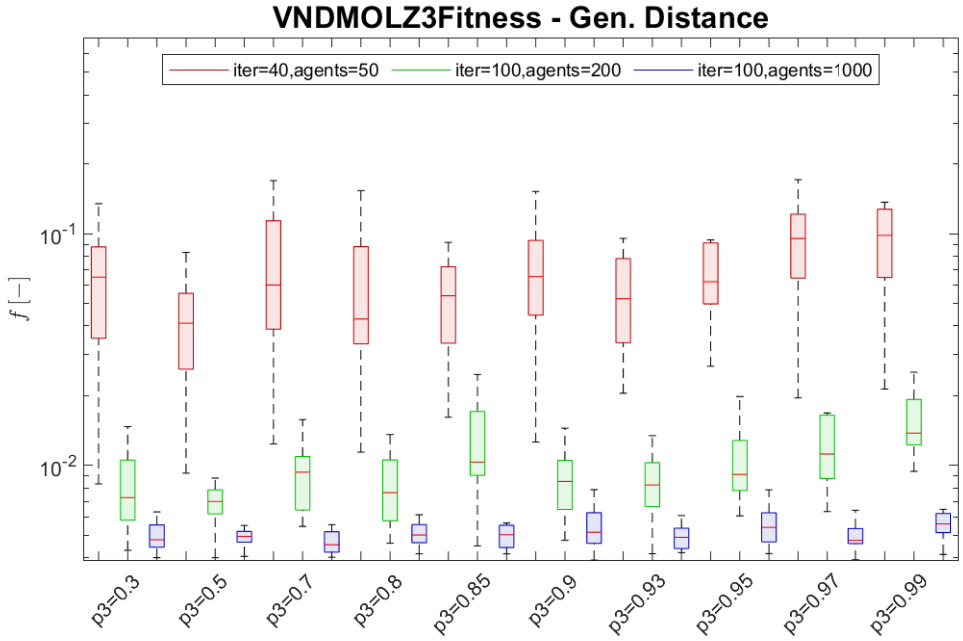

Fig. B.7: Standard boxplots for ten different settings of the p_3 parameter for the Gen.Distance metric.

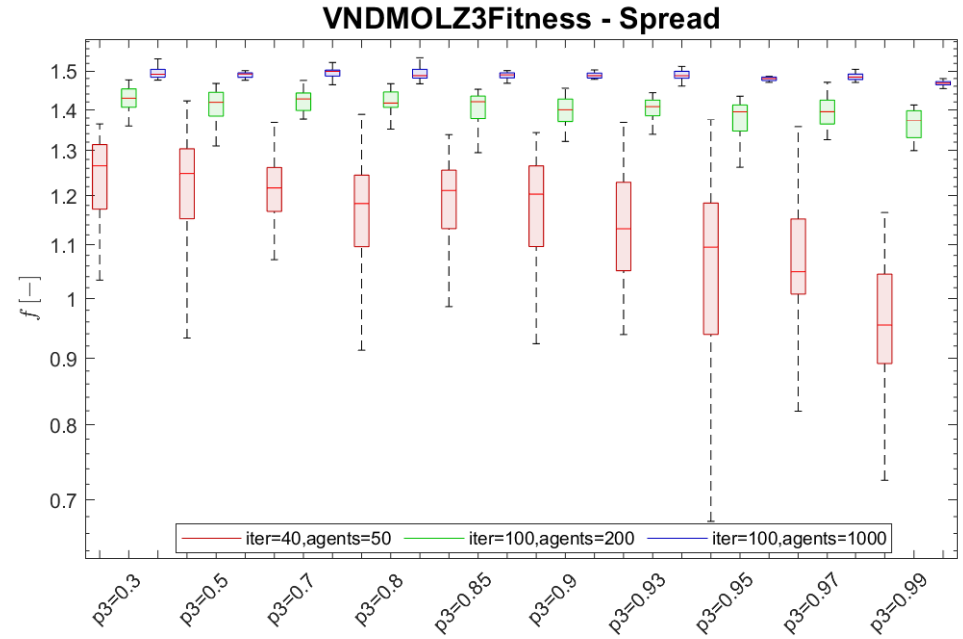

Fig. B.8: Standard boxplots for ten different settings of the p_3 parameter for the Spread metric.

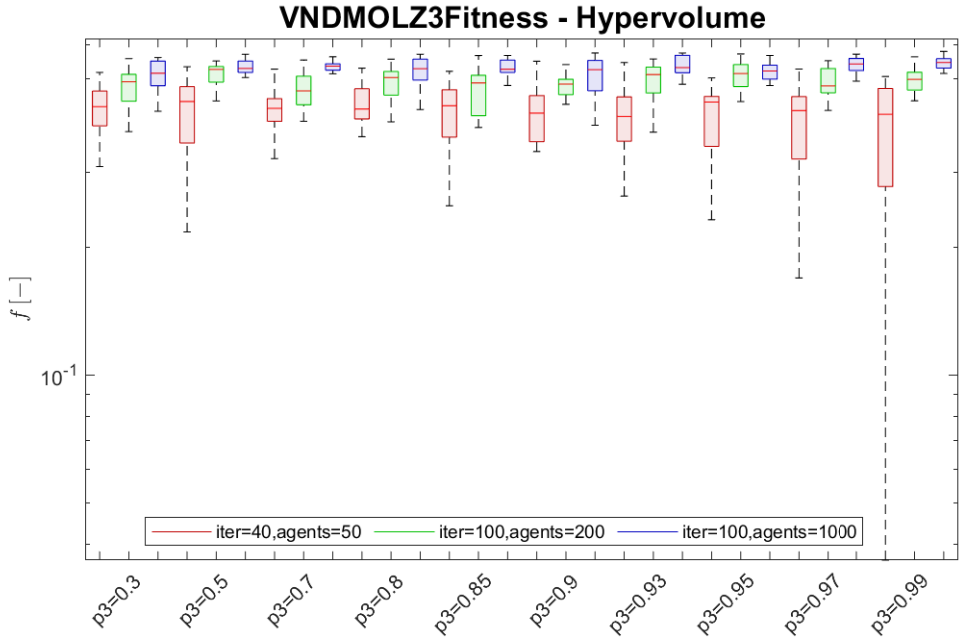

Fig. B.9: Standard boxplots for ten different settings of the p_3 parameter for the Hypervolume metric.

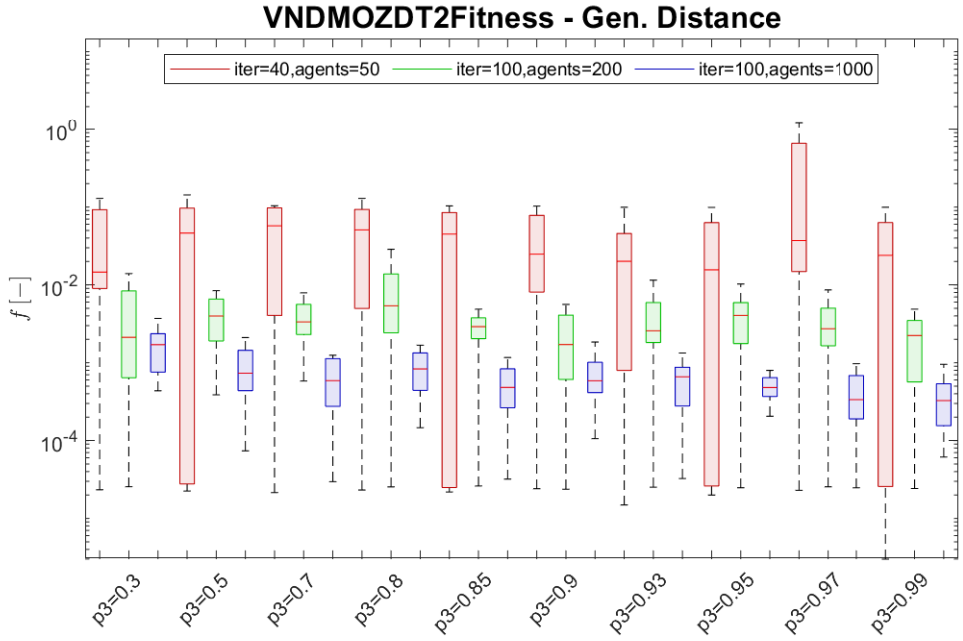

Fig. B.10: Standard boxplots for ten different settings of the p_3 parameter for the Gen.Distance metric.

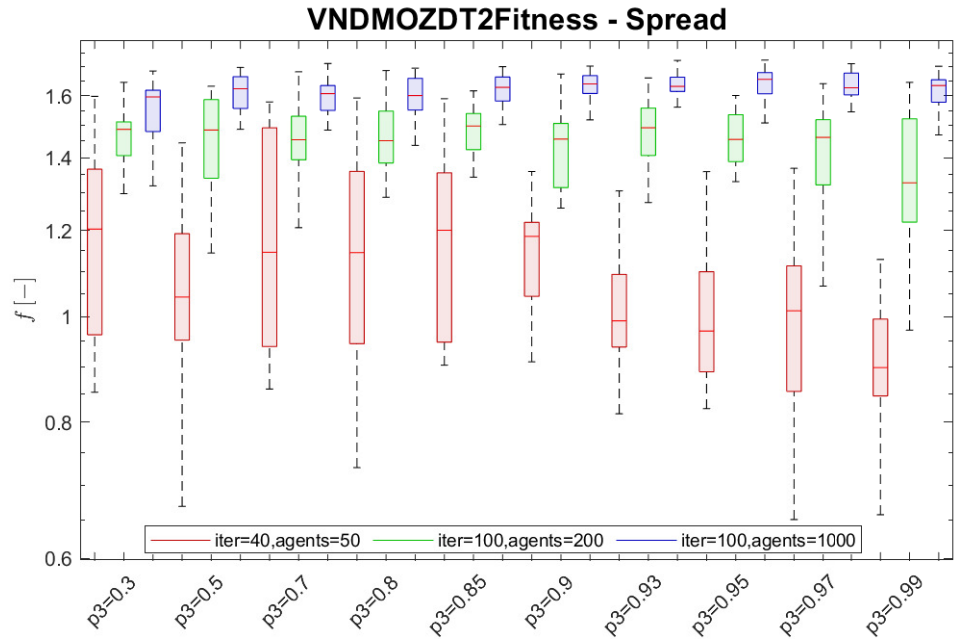

Fig. B.11: Standard boxplots for ten different settings of the p_3 parameter for the Spread metric.

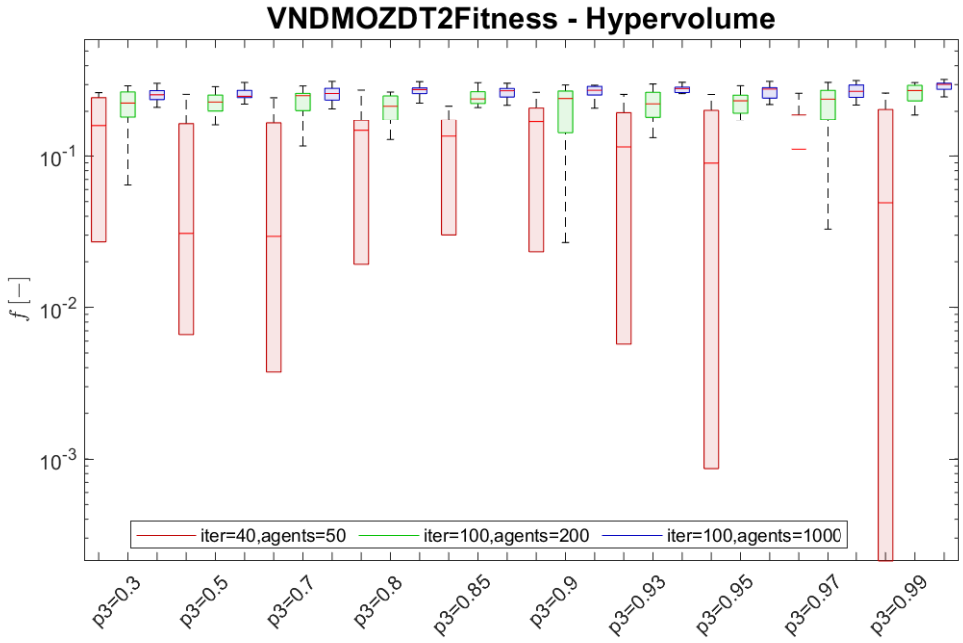

Fig. B.12: Standard boxplots for ten different settings of the p_3 parameter for the Hypervolume metric.