

Bangor University

DOCTOR OF PHILOSOPHY

Visualisation, optimisation and Machine Learning: Application in PET Reconstruction and Pea segmentation in MRI Images

Al-Maliki, Shatha

Award date:
2021

Awarding institution:
Bangor University

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.



PRIFYSGOL
BANGOR
UNIVERSITY

School of Computer Science and Electronic Engineering
College of Environmental Sciences and Engineering

**Visualisation, optimisation and Machine
Learning: Application in PET
Reconstruction and Pea segmentation in
MRI Images**

Shatha F. Al-Maliki

Submitted in partial satisfaction of the requirements for the
Degree of Doctor of Philosophy
in Computer Science

Supervisor Dr. Franck P. Vidal

December 2020

Statement of Originality

The work presented in this thesis/dissertation is entirely from the studies of the individual student, except where otherwise stated. Where derivations are presented and the origin of the work is either wholly or in part from other sources, then full reference is given to the original author. This work has not been presented previously for any degree, nor is it at present under consideration by any other degree awarding body.

Student:

Shatha F. Al-Maliki

Statement of Availability

I hereby acknowledge the availability of any part of this thesis/dissertation for viewing, photocopying or incorporation into future studies, providing that full reference is given to the origins of any information contained herein. I further give permission for a copy of this work to be deposited with the Bangor University Institutional Digital Repository, the British Library ETHOS system, and/or in any other repository authorised for use by Bangor University and where necessary have gained the required permissions for the use of third party material. I acknowledge that Bangor University may make the title and a summary of this thesis/dissertation freely available.

Student:

Shatha F. Al-Maliki

Acknowledgements

First and foremost, I would like to express my deep and sincere gratitude to my research supervisor **Dr. Franck Vidal**, for giving me the opportunity to do this research and providing invaluable guidance throughout this study. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the research and to present the research work as clearly as possible. It was a great privilege and honour to work and study under his guidance. I am extremely grateful for what he has offered me and for his corrections of this thesis. I would also like to thank him for his friendship and empathy.

Great thanks to **Dr. Évelyne Lutton** from France's National Research Institute for Agriculture, Food and Environment (INRAE) and **Prof. François Boué** from French National Centre for Scientific Research (CNRS) for supplying the Magnetic Resonance Imaging (MRI) image data and made the experiments possible.

I would also like to acknowledge the Ministry of Higher Education and Scientific Research (MOHESR) of Iraq for generosity in funding my PhD study. Great thanks to Basra University for supporting and helping me during my undergraduate, master, and PhD study.

It would not have been possible to write this thesis without the help and support of the kind people around me: my lovely husband **Hayder Al-maneea** for his personal support and great patience at all times. Also, my sweet kids **Fatimah** and **Mohammed** for their kindness and they make my life full with hope without despair. As well, I would like to thank all my family in Iraq: wonderful mom, sisters and brothers for this support and prayers to me. A special thanks go to my wonderful dad who always encourages me and pushes me to do the best. His soul surrounds and inspires me to pass the difficulties of the study.

Last, but not least, I would like to thank all the staff of the School of Computer Science and Electronic Engineering at Bangor University for their help during these years, and Supercomputing Wales (SCW) (<https://www.supercomputing.wales/>) for the use of its supercomputer.

Abstract

This study aims to investigate the behaviour and applications of an Evolutionary Algorithm (EA) based on a particular approach of Cooperative Co-evolution Algorithm (CCEA), the “Parisian approach” where the solution of an optimisation problem is a set of individuals (e.g. the whole population) instead of a single individual (the best one) as in typical EAs. The CCEA we selected is called “Fly algorithm”. It is named after flies, because the individuals are extremely primitive and correspond to three-dimensional (3-D) points. The focus of this study relies on visualisation to examine the Fly Algorithm (FA) to solve complex problems -reconstruction and segmentation of two types of medical imaging modalities, Positron emission tomography (PET) and MRI.

For image reconstruction, we compare the performance of FA with traditional non-cooperative optimisation schemes, such as Real-Coded Genetic Algorithm (RCGA), Particle Swarm Optimization (PSO) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithms on two test cases: A toy problem, the Lamps; and a complex inverse problem, PET reconstruction. This choice is based on three facts: i) FA has been built on top of RCGA, the comparison yield an assessment of the cooperative component that has been added, ii) PSO has been sometimes opposed to FA, and iii) CMA-ES is considered as the state-of-the-art for continuous optimisation. Our experiments highlight some structural differences and experimentally compare these algorithms. In both test cases FA exhibits a better scalability.

In another work, we propose using information visualisation and user interaction techniques to explore the algorithm’s internal data. Our aim is to better understand what happens during the evolutionary loop. Using PET reconstruction, we demonstrate that it is possible to interactively discover when an early termination

could be triggered. It is implemented in a new stopping criterion that reduction of the number of iterations without any loss of accuracy.

This methodology lead to the segmentation where, we combine optimisation, computer vision and visualisation/data exploration to analyse MRI data and detect peas inside the human stomach. We propose to perform the image analysis task as a multi-objective optimisation. We rely on the FA implemented using NSGA-II. The output of the optimisation is a succession of datasets that progressively approximate the “Pareto front”, which needs to be understood and explored by the end-user. Using interactive Information Visualisation (InfoVis) and clustering techniques, peas are then semi-automatically segmented.

Once a labelled dataset became available, we performed a binary classification as a food recognition problem, implemented using a deep Convolutional Neural Network (CNN). The results have been analysed using interactive visualisation. We prove in this work that advances in computer vision and machine learning can be deployed to automatically label the content of the stomach even when the amount of training data is low and the data imbalanced.

To make the work more robust by taking advantage of the labelled data, we compare the performance of some more traditional machine learning classifiers by using online application. Also, we deploy the multi-objective optimisation NSGA-II to use it as classifier and feature selection to improve classification accuracy and reduce the computational complexity. The result of this classifier is then refined using a residual neural network (ResNet). We have presented a fully-automatic segmentation of the peas using a combination of evolutionary computing, machine learning and computer vision techniques. The final results were confirmed by experts.

In conclusion, our investigations confirm that the Fly algorithm works well with a complex search space. We prove the use of a simple but effective visualisation can help to understand the behaviour of the FA and extract early results without need to wait until the optimisation loop finished also, to analyse the output of the FA. We open the door for the researchers to try use the algorithm in segmentation

processing images with consideration the description of the object they want to segment it.

Contents

1	Introduction	1
1.1	Context	1
1.2	Hypothesis	2
1.3	Aim and objectives	2
1.4	List of publications	3
1.5	Contributions	5
1.6	Outline	6
2	Background	8
2.1	Context	8
2.2	Medical Imaging	9
2.2.1	Emission tomography in nuclear medicine	10
2.2.2	Magnetic Resonance Imaging (MRI)	13
2.3	Visualisation in Evolutionary algorithms	13
2.4	Data visualisation	16
2.4.1	Scatterplots	17
2.4.2	Parallel Coordinate Plots	17
2.4.3	Radar Chart	19
3	Parisian evolution and Fly algorithm: A review	21
3.1	Introduction	21
3.2	Evolutionary Computing (Evolutionary Computing (EC))	22
3.3	Solution encoding	25
3.4	Selection operator	26
3.5	Genetic Engine	27
3.5.1	Crossover operator (Recombination)	28
3.5.2	Mutation operator	29
3.6	Particle Swarm Optimization	32
3.7	Cooperative co-evolution and Parisian Evolution	33
3.7.1	A special case of Parisian Evolution: The Fly Algorithm	36
3.7.2	Fly Algorithm for Stereovision	39
3.8	Comparison	40
3.9	Conclusion	41
4	Comparison between the efficiency of FA, GA and PSO	42

4.1	Introduction	42
4.2	Two test cases for the Fly Algorithm	43
4.2.1	Lamps problem	43
4.2.2	PET Tomography Reconstruction	44
4.3	Fly algorithm implementation features	48
4.3.1	Fitness Metrics	48
4.3.2	Steady state vs Generational	50
4.3.3	Threshold Selection	52
4.3.4	Mitosis	52
4.3.5	Mutation	53
4.3.6	Crossover	54
4.4	Experimental analysis	54
4.4.1	Fair Comparison of the Algorithms	54
4.4.2	Quantitative analysis of the results on the Lamps problem .	56
4.4.3	Quantitative analysis of the results on PET Reconstruction	65
	Global fitness	68
	Correlation between the Ground Truth and the reconstruction	71
	Fixed vs Varying Population Size	73
4.5	Conclusion	77
5	Visualisation in evolutionary reconstruction of PET images	79
5.1	Introduction	79
5.2	Evolutionary reconstruction in PET	81
5.3	Data exploration	84
5.4	Results	86
5.5	Conclusion	98
6	Using a multi-objective optimisation algorithm in interactive analysis of MRI gastric images	100
6.1	Introduction	100
6.2	Problem definition	101
6.3	Multi-objective optimisation problem	106
6.4	Visualisation	109
6.5	Conclusion	111
7	Recognising specific foods in MRI scans using CNN and visualisation	114
7.1	Introduction	114
7.2	Object recognition	116
7.3	Machine Learning approach	116
7.4	Deep Learning	117
7.5	Data repository	118
7.6	Recognising Peas in MRI scans of the stomach	121
7.7	Filtering the results using interactive visualisation	125

7.8	Conclusions	131
8	Fully-automated Segmentation of peas using Multi-objective Optimisation and Machine learning	133
8.1	Introduction	133
8.2	Image data statistics	134
8.3	Training traditional classifiers	140
8.4	Training CNN model ResNet-50	143
8.5	Feature selection and classification using NSGA-II	144
8.6	Automatic Keypoint Selection	147
8.7	Discussion and Conclusion	149
9	Discussion and Conclusions	154
9.1	Discussion	154
9.2	Limitations and future work	157
9.3	Conclusion	158
	Acronyms	160
	References	163

List of Figures

2.1	PET-CT examination of a “head and neck” patient in Oncology. In this case tumours were not visible on the anatomic images from CT, but were on the physiological images from PET. Top row: axial plane; middle row: coronal plane; and bottom row: sagittal plane. Source: the Cancer Imaging Archive (http://www.cancerimagingarchive.net/) [34].	11
2.2	PET data acquisition: $1 e^-$ combines with $1 e^+$; it may results in an annihilation reaction, which generates 2γ of 512 keV emitted at about 180° ; the line joining the pair of detectors activated by this pair of γ is called LOR; the system records many LORs [58].	12
2.3	Conversion of LOR data into a sinogram [58].	12
2.4	Example of a typical slice from our MRI dataset #2.	14
2.5	Example for scatterplots. Plot the points around the x-axes and y-axes for the value in rang (0-135) and (0-75).	18
2.6	Example for scatterplots on MRI showing the final results for Fly algorithm implementation presented in Chapter 6.	18
2.7	Example for Parallel coordinate plot that we used to analyse the output generated by NSGA-II for pea classification in MRI images (see Chapter 6).	19
2.8	Same Parallel coordinate plot as in Figure 2.7 when data filtering has been applied using brushing (some of the data has been selected and remained in the same colour as previously, the excluded data is shown in grey to make it less visible).	19
2.9	Example for Radar chart. The axes represent some statistics on the images we used in the ML in Chapter 7.	20
3.1	Different classes of search methods. The branch corresponding to evolutionary computing is highlighted in grey.	23
3.2	Classical EAs for optimisation: a solutions is encoded into an individual of the population. The population evolves a series of potential solutions and concentrates on the “best” areas of the search space (here a maximisation example).	24
3.3	Schemes of Genetic Encoding.	26
3.4	Genetic Operators for Evolutionary Algorithm.	28
3.5	Different strategies for crossover operator.	30

3.6	Cooperative Co-evolution scheme or Parisian EAs: several individuals (sometimes the whole population) represent a solution to the problem. The loop is similar to classical optimisation applications but embeds an additional step in the main loop that aggregates individuals to build a solution, evaluate it and distribute reward to individuals.	36
3.7	The flies are initialised randomly within the intersection of the cameras' 3-D fields of view [84].	39
3.8	Projections b_1 and b_2 of fly B which lies on the surface of an object, will have identical gray levels, unlike pixels a_1 and a_2 . This holds true independently of camera geometry.	40
4.1	Arrangement of a set of 4 lamps to enlighten a square.	45
4.2	Test case data: Known projections (c) are provided by the imaging system. The tomography reconstruction provides an estimate of the unknown radioactive concentration (a).	46
4.3	From a population of flies (\hat{f}) to estimated projections (\hat{Y}).	49
4.4	Example of projections created by the population of flies. For illustration purposes only two projections and two flies are considered.	51
4.5	Mean global fitness for the Lamps problem for the different algorithms for different problem sizes. For problem sizes 500 and 1000, jobs were killed on the supercomputer as they did not complete within 3 days (the maximum limit allowed).	62
4.6	Simulated lamp configuration for each algorithm for each problem size. For PSO and RCGA, 100 particles or individuals were used. Good images should be gray and homogeneous. Black and white pixels correspond to a lack of illumination and to an overlap respectively. The image for each optimisation algorithm corresponds to the median result in terms of global fitness over 101 runs, except for problem size 1,000 where only 21 runs were used. We use the median result rather than the best result for each algorithm to allow a fair comparison. . .	63
4.7	Efficiency and effectiveness comparison of all the algorithmic approaches for the different problem sizes. The horizontal axis of the scatter plots represents the number of lamps created before acceptance of the solution; the vertical axis the average global fitness. Numerical values were averaged over 101 runs. The best algorithms are in the top left corners of the plots (left for smallest computational requirements, i.e., efficiency, and top for highest global fitness values, i.e., effectiveness).	64

4.8	Fitness value for Problem size 100 as a function of the number of lamps created before acceptance of the solution, which is linearly related to computational requirements. For each optimisation algorithm, the curve corresponds to the median result in terms of final global fitness over 101 runs.	64
4.9	ℓ^2 -norm (global fitness) between Y (known projections) and \hat{Y} (projections estimated by the optimisation algorithm). Each marker represents a run for a given algorithm. There are 15 runs per algorithm. Best algorithms are located at the bottom-left corner of the plot. Note that the problem answer considered here for the Fly Algorithm is the concentration of flies, i.e., including bad flies.	70
4.10	Reconstructed images using Particle Swarm Optimisation and real-coded genetic algorithm with three different selection operators. The image for each optimisation algorithm corresponds to the median result in terms of ℓ^2 -norm between Y and \hat{Y} (global fitness) over 15 runs.	72
4.11	Box plots ofZNCC (or similarities) between f (noise-free ground truth) and \hat{f} (tomographic reconstruction corresponding to the concentration of flies) before elimination of bad flies, for 15 runs of Fly Algorithm with various operators setups.	72
4.12	Images reconstructed using the Fly Algorithm: with both good and bad flies (top row), and without bad flies (bottom row). The images correspond to median result in terms of ℓ^2 -norm between Y and \hat{Y} (global fitness) over 15 runs.	74
4.13	ℓ^2 -norm (global fitness) between Y (known projections) and \hat{Y} (projections simulated by the fly population) before elimination of bad flies. More flies have been used to further evaluate the mitosis. . . .	75
4.14	TV-norm of the reconstructed slices (\hat{f}) before elimination of bad flies. More flies have been used to further evaluate the mitosis.	75
5.1	Initial prototype Parallel Coordinate Plot showing an initial run of the evolutionary process. Objects are coloured according to their iteration number, included as the first axis. This is a screen-shot captured from the tool itself, the labels are clearer in the tool.	85
5.2	The same dataset as in Figure 5.1, with Brushing active on axes 3 and 5 (dEuclid_sinogram and TV_reconstruction). The ranges selected are shown by the tinted rectangle overlaid on those axes. This is a screen-shot captured from the tool itself, the labels are clearer in the tool.	85
5.3	The coordinated scatterplot for Figure 5.2. Items are coloured as in the original figure, a smaller circle represents denotes that no image is available for that point and a larger square does. This is a screen-shot captured from the tool itself, the points and labels are clearer in the tool.	86

5.4	Sinograms.	87
5.5	Reconstructed images.	87
5.6	Combined evolution of the global fitness and the ZNCC between the ground-truth and the reconstructed image.	90
5.7	Combined evolution of the global fitness and the TV of the reconstructed image.	90
5.8	Evolution of the global fitness. For each iteration plot the value of $\ Y - \hat{Y}\ _2^2$	92
5.9	Evolution of the total variation seminorm.	93
5.10	Manual selection of a good candidate solution based total variation seminorm and duration.	94
5.11	Intensity profiles int the ground-truth and in the reconstructions presented in Figure 5.5.	95
5.12	Reconstruction of Phantoms 1, 2, and 3 without and with the new stopping criterion.	97
6.1	MRI slice of a human body, the selected region of interest (Stomach), and the selected pea.	103
6.2	Objectives. For visibility objective functions are displayed in negative (low intensities appear bright; high intensities appear dark).	104
6.3	Convolution kernel $ROI_C(I, x, y, R)$	104
6.4	Example of a Pareto frontier (in red), the set of Pareto optimal solutions (those that are not dominated by any other feasible solutions). The boxed points represent feasible choices, and smaller values are preferred to larger ones. Point C is not on the Pareto frontier because it is dominated by both point A and point B. Points A and B are not strictly dominated by any other, and hence do lie on the frontier. Credit: image and caption by Johann Dréo, distributed under a CC BY-SA 3.0 license.	107
6.5	The main loop for NSGA-II algorithm. Where P_t is the population of parents, Q_t is the population of offspring, R_t is the combination of parent and offspring population and $F = (F1, F2, \dots)$ all non-dominated front R_t [42].	108
6.6	Scatterplots and parallel coordinates plots of successive generations. All solutions (flies) are plotted in red by default. When the user selects an area in the scatterplot, a specific colour is assigned to this area and linked to the corresponding lines in the parallel coordinate plot.	110
6.7	Candidate solution clusters.	111
6.8	Final result. The purple colour represent the points in the right location, and the red colour represent the wrong location,	112
7.1	The steps for preparing data by the experts and save the labelled data in database.	120

7.2	MRI slice of a human stomach and the manual segmentation for the peas.	121
7.3	Examples of peas with different levels of confidence (confidence between 0-3) in the selected ROI.	122
7.4	Automatic selection of non-pea samples in the ROI.	122
7.5	The main steps for implementing supervised learning.	124
7.6	Parallel coordinate plots of the CNN performance metrics for various image sizes and number of epochs.	128
7.7	Radar charts of the CNN performance metrics for various image sizes and number of epochs.	128
7.8	Same as Figure 7.6, but with brushing to filter the data.	129
7.9	Same as Figure 7.7, but with the data filtered using brushing in Figure 7.8.	129
7.10	Peas after resampling of slices at different image resolutions (see corresponding images in Figure 7.11).	130
7.11	Visual representation of the classification results. Circles in magenta depict TPs, crosses in magenta TNs, blue circles depict FPs, and blue crosses FNs.	131
8.1	Parallel coordinate for all the selected points (peas) and all the random located points (non-peas) for all features from all the datasets.	135
8.2	Image statistics. Sample from slice #17 of Dataset #5 for all image statistics and objective functions to describe the problem. Each image represents one statistic or one objective function.	136
8.3	The MRI image for slice #17 of Dataset #5.	141
8.4	Traditional classifiers. Sample from slice #17 of Dataset #5 for all classifiers.	142
8.5	Visual representation of the ResNet-50 classifier the best result (image size 128×128 pixels and 256 epochs). Circles in green TPs, crosses in green TNs and red circles depict FPs. Sample from slice #17 of Dataset #5.	145
8.6	Visual representation of the best result of the NSGA-II based classifier (64 generations of a population made of 1,024 individuals). Circles in magenta TPs, crosses in magenta TNs, and blue crosses FNs. Sample from slice #17 of Dataset #5.	147
8.7	Overall flowchart of our steps to extract the keypoints.	149
8.8	Samples from slice #17 of Dataset #5 for combining the features that selected using computer vision techniques.	150
8.9	Samples from slice #17 of Dataset #5 after apply a threshold.	150
8.10	Samples from slice #17 of Dataset #5 for the filtering the selection data by NSGA-II.	151
8.11	Samples for the final results from slice #17 of Dataset #5. Circles in magenta TPs, crosses in magenta TNs, and blue circles FPs	151

8.12 Final analysis from the experts. The visualisation shows the MRI image with the final results circles in magenta TPs, crosses in magenta TNs, and blue circles FPs. Yellow circles represent the experts confirmation 153

List of Tables

3.1	The similar features of Real-Coded GA, PSO, and FA. Consider here a problem with a k -D search space.	40
4.1	Search space dimension for PSO and real-coded GA for each problem size. There are 3 values (x , y and on/off) per lamp. $3 \times$ problem size is the number of lamps per individual recommended in the original Lamps problem article [136].	58
4.2	Parameters used in all Lamps problems.	59
4.3	Additional parameters used in the Lamps problems with the Fly Algorithm.	59
4.4	Additional parameters used in the Lamps problems with a real-coded GA.	60
4.5	Additional parameters used in the Lamps problems with PSO. . . .	60
4.6	Performance of the different algorithms on the Lamps problem with 7 different problem sizes (3, 5, 10, 20, 100, 500 and 1000). Numerical values correspond to average values and standard deviations over 101 runs, except for problem size 1000 where only 21 runs were performed due to computational constraints. The global fitness is given in %. This is a value that is maximised. Values for algorithms marked in bold are significantly better ($p < 0.01$) than the others for the same problem size.	60
4.7	Parameters used in all PET reconstructions.	67
4.8	Additional parameters used in the PET reconstructions with the Fly Algorithm.	67
4.9	Additional parameters in the PET reconstructions with PSO. . . .	67
4.10	Performance of the different optimisation algorithm configurations. Numerical values correspond to average values and standard deviations over 15 runs. The best algorithms for SSE projections and Reconstruction ZNCC are highlighted as follows: i) Values for algorithms marked in bold are significantly better ($p < 0.01$) than the others, and ii) Values with * show cases where the best performance is equally achieved by two or more algorithms (non separable, with $p > 0.05$). Note that the problem answer considered here for the Fly Algorithm is the concentration of flies, i.e., including bad flies. . . .	70

5.1	Initial parameters of the Evolutionary Algorithm.	88
5.2	Performance of reconstructions at different iterations. The best result for each metrics is in red, the second best in green, and the third best in blue. Iteration #113,401 has been selected by hand using the visualisation tool; #269,301 corresponds to the lowest global fitness; #199,101 corresponds to the first occurrence of the lowest TV; #274,101 corresponds to the last occurrence of the lowest TV; #284,250 corresponds to the last iteration.	89
5.3	Performance comparison between the algorithm with and without the new stopping criterion using 3 test cases. Each reconstruction has been performed 10 times.	96
6.1	Summary of the objectives. All these objectives are numerical values that need to be minimised.	102
7.1	Data extracted from all the datasets.	119
7.2	Performance evaluation of the CNN for various image sizes and number of epochs.	126
7.3	Classification results when using 16×16 images with a CNN trained over 256 epochs. The last row provides the accuracy, precision, recall and F_1 score when all the datasets are considered as a whole, i.e. these are not average values over the ten datasets, i.e. applying Eqs 7.1 to 7.4 with 346, 2217, 33 and 55.	127
8.1	Performance evaluation for all the traditional classifiers. Each value represents the overall values from all the datasets for each classifier. In other words, TP, TN FP and FN below include all the datasets. Accuracy, precision, recall and F1 score are computed using these values. The best performance for each column is highlighted in red colour.	143
8.2	Performance evaluation for ResNet-50 classifier. The best evaluation result is highlighted in red colour.	144
8.3	Performance evaluation for the NSGA2-based classifier. The best performance for each column is highlighted in red colour	148
8.4	The best classifier from the traditional, ResNet-50 and NSGA-II classifiers	149
8.5	Classification results.	150

Chapter 1

Introduction

1.1 Context

An Evolutionary Algorithm (EA) is a nature-inspired Optimisation algorithm, which refers to an algorithm that improves a problem solution through many iterations (known in this context as generations). Typically, each iteration is called a generation, and each generation is composed of a number of individuals. This group of individuals is called a population. In addition, EAs are known as random search methods that simulate the theory of natural biological evolution and/or the social behaviour of creatures. To mimic the practical behaviour of these varieties, various researchers have developed computational systems that require fast and robust solutions to complex optimisation problems [48]. There are many different types of optimisation algorithms. In this thesis, we focus on population-based nature-inspired heuristics, such as Evolutionary Algorithms and Particle Swarm Optimisation. In general, Optimisation algorithms are used to solve complex problems, including image processing, analysing, reconstructing, segmentation, and understanding digital images. Optimisation algorithms are often considered as black-box optimisation methods, where the user is interested only in the final solution provided by the optimisation algorithm. All the data generated by the Optimisation algorithms is used by the Optimisation algorithms only, and is discarded during or at the end the optimisation. With this approach, it is difficult, if not impossible, to improve or finely tune the Optimisation algorithms.

In this thesis, we focus on some large nature-inspired optimisation problems related to image reconstruction and image analysis. We aim to demonstrate how interactive visualisation can help to understand the behaviour of the algorithm

(e.g. to improve it), and help choose a “good” solution. We rely on the Fly Algorithm [82] for three-dimensional (3-D) reconstructions in nuclear medicine (see Chapter 5) where, interactive visualisation method is deployed to understand the behaviour of the algorithm. In another application, we use a multi-objective Fly Algorithm (FA) and interactive visualisation to segment Magnetic Resonance Imaging (MRI) images (see Chapter 6). We also use interactive visualisation and high performance capturing to test Convolutional Neural Network (CNN) models and select hyper-parameters (see Chapter 7). Finally, we use a multi-objective optimisation as a classifier to detect specific objects in a medical images (MRI) (see Chapter 8).

1.2 Hypothesis

In black-box optimisation, people trust the algorithm to provide a good solution. The user does not monitor the data between the beginning and the end of the optimisation process (the data in each generation). Still, the algorithm will run until it reaches a stopping criteria. In this case, and for Evolutionary Computing, the data between each generation will be discarded. However, black-box optimisation is not (always) a panacea. Traditional algorithms may fail in some cases, or may require an unreasonable amount of resources (CPU time, or amount of main memory). In this context, our hypothesis is as follows:

For complex and large problems where traditional algorithms are not suitable, ad-hoc interactive visualisation can be deployed to considerably improve either i) an optimisation algorithm, or ii) the selection of the problem solution on the “Pareto” front to use as a solution.

1.3 Aim and objectives

We aim to validate or invalidate our hypothesis. Therefore, the objectives of the study are as follows:

1. To identify a problem or class of problems where traditional black-box optimisation approaches are not suitable (Chapter 3 and Chapter 4). The study will illustrate how the FA compare in terms of efficiency and effectiveness to a few other traditional optimisation approaches.
2. To select two case studies to validate or invalidate our hypothesis, i) reconstruction in nuclear medicine (Positron emission tomography (PET)), and ii) segmentation of small regions in another type of medical images (here MRI).
3. To use interactive visualisation to analyse the internal data of FA to extract the best possible solution of the reconstructed PET image (Chapter 5).
4. To modify the FA to speed-up the reconstruction process without loss of accuracy (Chapter 5). The two most common stopping criteria in EAs are i) the total number of generations, and ii) stagnation (no further significant improvement in terms of fitness value). Learning from the previous analysis, we could introduce an alternative stopping criterion specific to the application.
5. To analyse the content of the stomach from MRI using multi-objective FA to find small regions on this image (Chapter 6).
6. To demonstrate that it is possible to train a CNN despite a very limited and imbalanced database of cases and, using an interactive visualisation to find the best suitable combinations of hyper-parameters (Chapter 7).
7. To use multi-objective optimisation (NSGA-II) as a classifier and compare it with common classifiers (Chapter 8).

1.4 List of publications

A list of published articles that present most parts of the work can be seen below in reverse chronological order:

1. J. Gardner, **S. Al-Maliki**, É. Lutton *et al.*, ‘Recognising specific foods in MRI scans using CNN and visualisation,’ English, in *Proceedings of Computer Graphics and Visual Computing 2020 (CGVC 2020)*, The Eurographics Association, Jul. 2020, pp. 11–18, ISBN: 978-3-03868-122-9. doi: 10.2312/cgvc.20201145
2. T. Wen, R. P. Mihail, **S. Al-Maliki** *et al.*, ‘Registration of 3D Triangular Models to 2D X-ray Projections Using Black-box Optimisation and X-ray Simulation,’ in *Computer Graphics and Visual Computing (CGVC)*, F. P. Vidal, G. K. L. Tam and J. C. Roberts, Eds., Bangor University, United Kingdom: Eurographics Association, 2019, pp. 105–113, ISBN: 978-3-03868-096-3. doi: 10.2312/cgvc.20191265
3. **S. Al-Maliki** and F. P. Vidal, ‘Evolutionary interactive analysis of MRI gastric images,’ in *First CoESE PhD Conference*, Bangor, UK, Jan. 2019
4. **S. Al-Maliki**, É. Lutton, F. Boué *et al.*, ‘Evolutionary Interactive Analysis of MRI Gastric Images Using a Multiobjective Cooperative-coevolution Scheme,’ in *Computer Graphics and Visual Computing (CGVC)*, The Eurographics Association, 2018, ISBN: 978-3-03868-071-0. doi: 10.2312/cgvc.20181216
5. **S. Al-Maliki**, É. Lutton, F. Boué *et al.*, ‘MRI gastric images processing using a multiobjective fly algorithm,’ English, in *Fifteenth International Conference on Parallel Problem Solving from Nature (PPSN XV)*, ser. Workshop on Evolutionary Machine Learning (EvoML), Sep. 2018. [Online]. Available: <http://ppsn2018.dei.uc.pt/index.php/workshops/>
6. C. C. Gray, **S. Al-Maliki** and F. P. Vidal, ‘Data exploration in evolutionary reconstruction of pet images,’ *Genetic Programming and Evolvable Machines*, vol. 19, no. 3, pp. 391–419, Sep. 2018, ISSN: 1573-7632. doi: 10.1007/s10710-018-9330-7

1.5 Contributions

Given the previous sections, this thesis investigates the problems involved in EAs and ML, and the use of Information Visualisation (InfoVis). It will focus on the major contribution of this work which include:

Open-source implementation of optimisation algorithms: We implement global optimisation methods (Pure Random Search (PRS), Simulated Annealing (SA), FA, and Particle Swarm Optimization (PSO)) as open-source. The Python implementation of all these algorithms and the test data are provided on GitHub ¹. We use some of these algorithms to investigate the ability of FA by comparing it with PSO and PRS (see Chapter 3). Mr. Tianci Wen (PhD student in SCSEE at Bangor University) which is studying the registration of 3D triangular models onto 2D X-ray projections uses three of these optimisation methods (PRS, SA, and FA) to address his problem. The registration framework is successful when using a suitable optimisation algorithm. This work can be found in publications no. [154].

Make Particle Swarm Optimisation cooperate: PSO is based on social interactions. The emerging collective behaviour results from a balance between following a leader and following an individual focus, thanks to inter-individual communications. This mechanism is different from EAs that rely on genetic transmission and natural selection analogies (birth, death and inheritance within a population). Important differences between them are how they manage diversity and how they communicate, making them best fitted to different optimisation tasks. The focus of contribution is fine-grained cooperation. By analogy with an evolutionary scheme that has long proved effective, the FAs, we design a cooperative PSO (coPSO), and a PSO-flavoured fly algorithm. Experiments run on a benchmark, the Lamp problem, show that fine-grained cooperation based on marginal fitness evaluations and steady-state schemes outperforms classical techniques when the dimension of the problem increases.

¹<https://github.com/Shatha1978/Optimisation-algorithm-examples>

The importance of the data visualisation: We recommend the use InfoVis and data exploration to understand some of the behaviours of an FA to extract early best solution and find early stopping. Also, InfoVis helps to understand the output of an FA implementation and enhancement these output to extract the optimal solution. This work can be found in publications no. [58], [114]–[116].

Finding food inside the human stomach: This contribution is part of a large project focused on the understanding of the influence of food structure on digestion. The long-term objective of this study is to help to model food interaction with the human digestive system. Advanced imaging techniques allow observation the digestion process at different scales. We use MRI that provides *in vivo* information at the large scale (stomach and duodenum of healthy human volunteers). We propose a framework to analyse the content of stomach (frozen green peas and pasta) from MRI: i) We use a “Fly Algorithm” to detect peas in these MRI images. The Fly Algorithm has been turned into a multi-objective cooperative-coevolution algorithm, and expert knowledge has been integrated through simple InfoVis techniques. ii) We consider a binary classification task to recognise a small region (in this case the peas) in MRI images. We use CNN as a feature classifier. These studies help the researchers who work in studying food structure to track and follow any type of food. This work can be found in publications no. [53], [114]–[116]

1.6 Outline

This thesis is organised into nine chapters. The first chapter gives outlines about this research, including motivations; it highlights the main contributions of this study and gives a list of published papers. Chapter 2 presents the scientific context related to this work and the main applications that have been used in this thesis. Chapter 3 gives a a brief overview for Fly algorithm (FA) and traditional nature-inspired optimisation approaches (Genetic Algorithm (GA)) and Particle Swarm Optimisation (PSO). Also it shows the main differences between them. Chapter 4 shows the comparison between the efficiency of FA and traditional nature-inspired optimisation approaches GA, PSO and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) by using two problems (Lamp problem and PET reconstruction).

The next chapter (Chapter 5) is a collaborative work. Our contribution relates to the use of a special type of Cooperative Co-evolution Algorithm (CCEA), which is the Parisian approach (specifically the Fly algorithm) in PET reconstruction, and Mr. Gray² contribution focuses on implementing the InfoVis components based on our specifications. We exploited her tool to explore the data and understand some of the behaviours of theFA. Chapter 6 proposes the first multi-objective implementation of the FA to segment small regions (peas) on the MRI. Our collaborators (Dr. Évelyne Lutton³ and Prof. François Boué⁴) provide the MRI data. The outcomes of the algorithm show that FA is suitable for the segmentation if the problem cannot be identified by a single equation. After obtaining more data and labelled data, it became possible to train Machine Learning (ML) algorithms, and this is presented in the Chapter 7. This Chapter was another collaborative work. Mr. Gardner⁵ prepared the data to train a CNN, and conducted some preliminary experiments with ML algorithms. With the help of our expert collaborators, these experiments were successful and developed further in this thesis. The output from the FA contains some undesirable points, due to the accuracy of the images used, so we used multi-objective optimisation problem to filter the outputs and this is explained in Chapter 8 explains. The last chapter discusses the work carried out and provides some possible directions for future work.

²PhD student in Computer Science at Bangor University at the time of this study

³Research director at the French National Institute of Agricultural Research (INRAe)

⁴Professor at the French National Centre for Scientific Research (CNRS) and French Alternative Energies and Atomic Energy Commission (CEA)

⁵BSc student in Computer Science at Bangor University at the time of this study

Chapter 2

Background

2.1 Context

The long-term objective of this collaborative study is to aid the modelling of food interactions with the human digestive system. More precisely, exploring, understanding and modelling the influence of food structure on the nutrients release kinetics during digestion. The digestion process is not just simply food decomposition and the absorption of macronutrients, e.g. carbohydrates, proteins and lipids, as well as micronutrients, e.g. vitamins, minerals, and other food components. Two relevant views are advanced, the kinetics of digestion and food structure. Hence work on food under various structures being assessed for differences of their behaviour, under the action of gastric and intestinal enzymes, has been initiated [1], [43], [45], [51], [163].

On the one hand, our international collaborators have developed observation of *in vitro* digestion at low scales (1 mm down to 10 nm) using microscopy (1 mm down to 10 μ m [43], [51]) and radiation scattering (1 μ m down to 10 nm [43], [86], [100]). Simple proteins were chosen because they form gels which display a huge variety of structure at those scales, and are a crucial food component for health [7]. On the other hand, they have conducted *in vivo* digestion experiments using MRI imaging [52] to study, in relation to the measurement of satiation, gastric emptying.

In particular, the food appearing in image of the stomachs of volunteers contained an easy to digest food (cooked pasta or bread), as well as a few raw peas, introduced the meal as flow tracers (around 20 peas in one stomach for the current experimental data. The peas can reveal the motion of the stomach, which generates a stirring

and gentle “trituration” of the food. This trituration induces in turn the motion of the pasta, which is the food bolus (or “chime”), to facilitate the action of the gastric juice. Being able to follow the peas in these images is important as it reveals how the food bolus is stirred inside the stomach. More generally understanding the food flow is very important since it influences the regions where food is located, and the duration of such localisation, and therefore the kinetics of food digestion. Put simply, the thesis is about image analysis used to detect such peas in medical image processing and advanced Computer Vision (CV) methods.

Formerly, in early 2018 [114], a first image analysis was conducted using the “Fly Algorithm”, an evolutionary optimisation algorithm [115], [134]. The Fly Algorithm, contrary to classical image processing techniques, computes from a model of the object or the surface chosen by the user, an image produced by “flies” located on these model objects or surfaces. This allows for the “fit” of a large variety of various features. Here the peas were considered as circular areas with a rather uniform grey colour surrounded by contrasting, irregular areas. Since we need to detect more than one pea, we used a multi-modal extension of the fly algorithm. Moreover, to decide whether a point was the centre of a pea required several criteria to be fulfilled, which was difficult to embed in a single fitness function. A solution was multi-objective EAs converging toward the Pareto front, through the widely used method, NSGA-II [115].

This chapter presents the general application context related to this thesis, beginning with a brief introduction of medical imaging and in Section 2.2, including a definition of the main type of the medical images used in this thesis PET and MRI. Section 2.3 gives a brief explanation for the visualisation of an Evolutionary Algorithm, and how it helps to understand the behaviour of its implementation. Furthermore, this section presents the data visualisation styles used in this study (scatterplots, parallel coordinate plots, and radar charts).

2.2 Medical Imaging

Medical imaging not only helps with the diagnosis of some diseases, but it also provides help with the planning and monitoring of treatments. The development

of medical imaging has continuously improved over the past four decades. For computer scientists, what is perhaps the most obvious about this progress in medical imaging is how it keeps challenging us to provide significant innovation in computational techniques for nearly all aspects of image processing. There are many different types of medical images, e.g. X-ray radiographs and ultrasound. These types are called modalities. The utilisation of multiple imaging modalities on a single patient, for example, MRI and PET requires intelligent algorithms for image registration and pattern recognition [15]. In this thesis, we make use of PET and MRI images.

2.2.1 Emission tomography in nuclear medicine

In nuclear medicine, a radio-pharmaceutical (i.e. radioactive substance) is administered to the patient. The radioisotope is fixed on a given molecule that is going to be absorbed by the body in relation to a targeted physiological process, such as tumour growth, bone fracture, or reduced blood flow in the heart. Imaging in this context is a type of *molecular and functional* imaging. In other words, a physiological function is targeted by the radioactive molecule. In Oncology, the molecule will be fixed by tumours because of the growth of cancerous cells. This is why tumours are highlighted in Figure 2.1. The radioactive concentration is much higher in tumours than in healthy tissues, which leads to more emission from the tumours. For this reason tomography in nuclear medicine is called emission tomography (ET): The source of radiation is within the patient. There are two main techniques: Single-Photon Emission Computed Tomography (SPECT) and PET. They both produce a stack of 2-D cross-sections through the human body, which corresponds to a 3-D map of the radioactive concentration within the patient (see Figure 2.1b). In this study, we use PET (Chapter 5) as an application example to initially investigate how data can be explored using interactive visualisation to better understand the outcome of the evolutionary optimisation.

Figure 2.2 shows the principle of the PET data acquisition chain. Images produced in ET have a relatively low resolution (typically 128×128 pixels) and signal-to-noise ratio (SNR). Well-known tomography techniques used in radiology departments, such as Computed Tomography (CT) and MRI generate images with a much higher

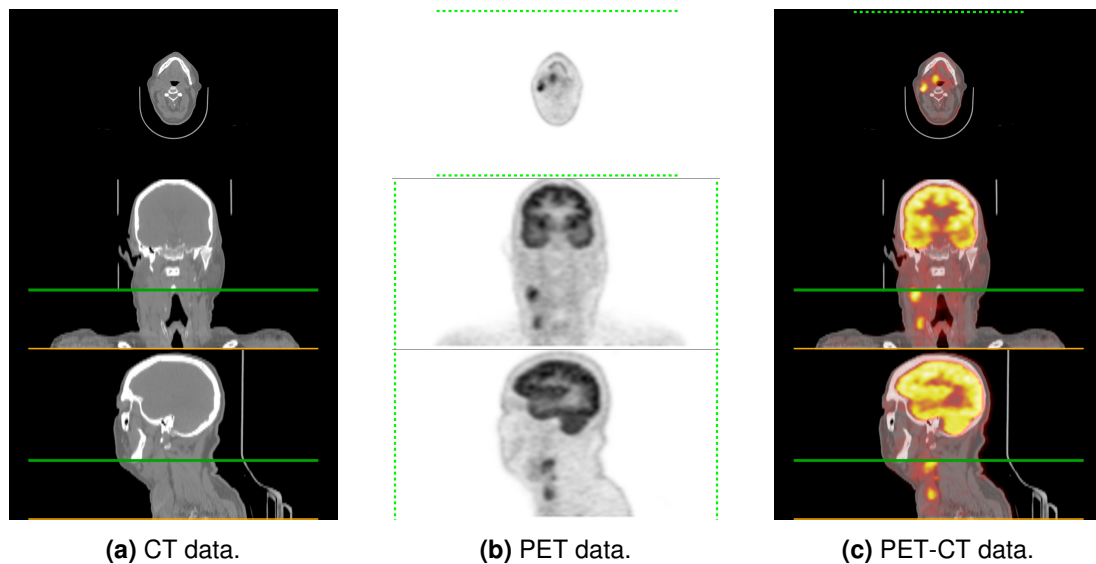


Figure 2.1: PET-CT examination of a “head and neck” patient in Oncology. In this case tumours were not visible on the anatomic images from CT, but were on the physiological images from PET. Top row: axial plane; middle row: coronal plane; and bottom row: sagittal plane. Source: the Cancer Imaging Archive (<http://www.cancerimagingarchive.net/>) [34].

resolution (typically 512×512 pixels) and SNR (see Figure 2.1a). They are used to visualise anatomical structures. Modern medical scanners now combine PET with either CT or MRI to provide collocated physiological and anatomical image datasets (see Figure 2.1c).

In PET, the positron (often shortened as e^+ or β^+) is the type of ionising radiation that is used. When a positron collides with an electron (e^-), an annihilation reaction may occur. In this case, two photons (γ) are emitted at almost 180° of each other with a kinetic energy of 512 kiloelectron volts (keV). Note that photons are the elementary particle of light. Pairs of annihilation photons are detected in coincidence, i.e. at almost the same time, by a dedicated scanner. The line joining the two detectors (see red parallelepipeds in Figure 2.2) that caught the photons of the same pair is called the line of response (LOR) (see red line in Figure 2.2). Each detector of the PET scanner has a unique identifier. All the pairs of detectors corresponding to the LORs are recorded by the system. However, the exact locations of the annihilation reaction are unknown. LOR data can also be converted into sinograms as this is a common data representation [49] that stores a set of 1-D projections at successive angles in a 2-D image (see Figure 2.3).

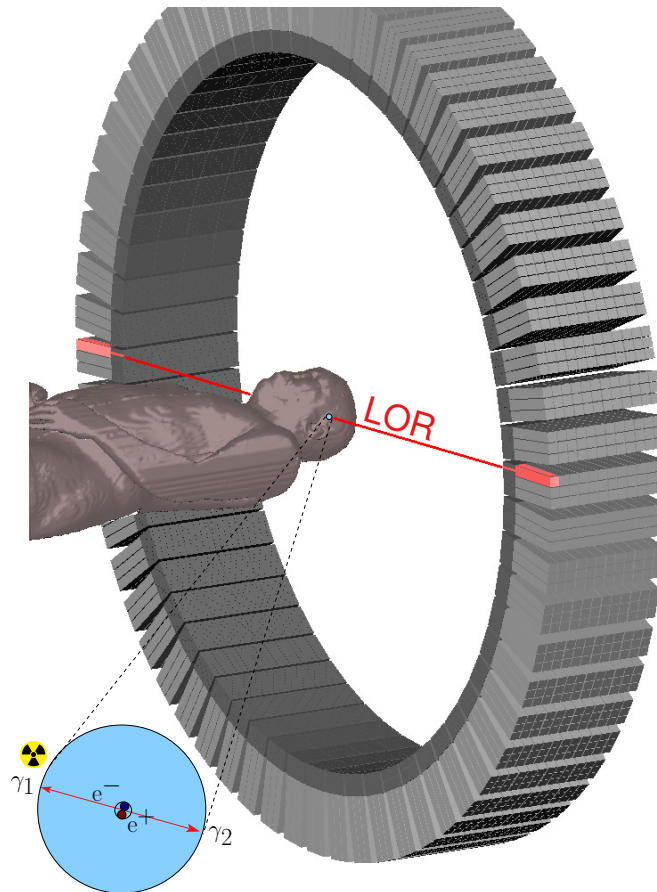


Figure 2.2: PET data acquisition: 1 e^- combines with 1 e^+ ; it may results in an annihilation reaction, which generates 2 γ of 512 keV emitted at about 180° ; the line joining the pair of detectors activated by this pair of γ is called LOR; the system records many LORs [58].

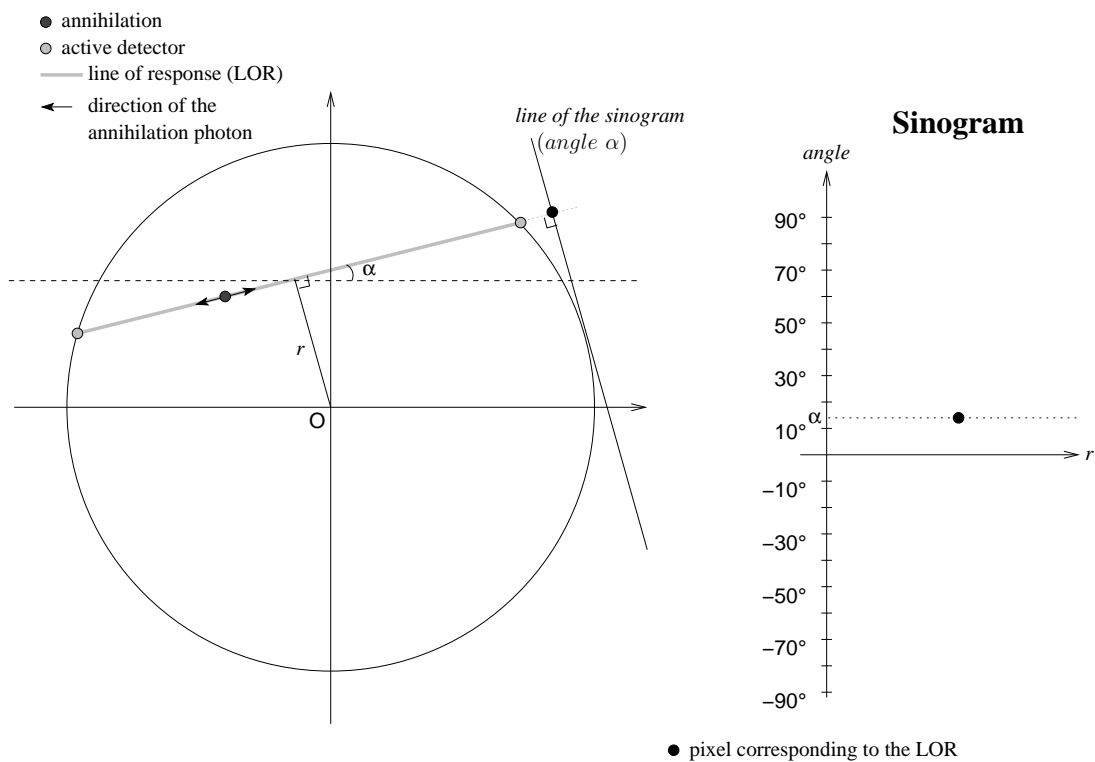


Figure 2.3: Conversion of LOR data into a sinogram [58].

To convert a LOR into a point into the sinogram, the angle between the LOR and the horizontal axis is computed (see α in Figure 2.3). The shortest distance between the LOR and the origin of the system is also computed (see r in Figure 2.3). The intensity of pixel(r, α) in the sinogram is derived from the number of corresponding LOR events detected by the PET scanner. Noise is actually a major concern in ET.

2.2.2 Magnetic Resonance Imaging (MRI)

The second application example relies on Magnetic Resonance Imaging (see Chapters 6). We further rely on the visualisation of the output data produced by the evolutionary algorithm to extract the *best* solution(s).

Contrary to emission tomography techniques, Magnetic Resonance Imaging can show small anatomical details. For example it is the most relevant imaging modality for diagnosis and classification of early axial spondyloarthritis as it highlights small defects in tissues [113]. Each MRI dataset is made of one or more cross-sectional images (called slices) through the human body [138] (see Figure 2.4).

The principle of the magnetic resonance mechanism is the measurement of radio-frequency radiation-produced from transitions induced between nuclear spin states of tissue hydrogen atoms (protons) in the existence of a robust external magnetic field. Whilst CT relies on X-ray reduction by tissues, which are a function of electron density and atomic number, relative pixel intensities in MRI are tissue relaxation times and function of proton densities [74].

2.3 Visualisation in Evolutionary algorithms

Optimisation, including Evolutionary Computing (EC), is often used in engineering as a black box optimisation where the final solution provided by the optimisation algorithm is used as the solution to the optimisation problem. A lot of data is generated during the evolution process, in particular data based on error metrics and correlation measurements. Traditionally all this data is discarded at the end of the evolutionary process, as only the final population is considered. It is however, possible to retain this information for future processing and analysis. The combination of visualisation and evolutionary computing is still a relatively



Figure 2.4: Example of a typical slice from our MRI dataset #2.

overlooked field. The emergence of information visualisation and data analytics is opening new doors for its use in the evolutionary computing domain. Two different approaches can be distinguished:

- visualisation to understand an Evolutionary Algorithm behaviour [73], [103], [157], [158], or outputs [26];
- interactive artificial evolution to improve the visualisation [12], [59], [85].

First attempts were reported at the end of the 90s. Early visualisations used relatively basic techniques that mostly relied on plotting with limited or no interactivity. Visualisations are most effective when they not only show information but allow a user to answer their own questions by interacting with the data [79].

To assist in tuning the Evolutionary Algorithm, the complex interplay of each metric needs to be understood. Examining the raw data in numeric form is often error-prone and limited by the exact process employed by the researcher. There are other methods, such as writing bespoke analysis programs or the use of summary statistics in a spreadsheet application. These methods are then limited by the capability of the tools, and results are still provided in text form which can be harder to reason with. The field of Visual Analytics provides another alternative, exploiting the visual processing and reasoning abilities of the human being [62]. Systems built for visual analytics can be expanded to use multiple views [112] of the same data set highlighting deeper relationships and patterns. Therefore, the exact metric value substituted with a graphical and/or spatial surrogate. In this form, relation of metrics becomes an easier task requiring less mathematical and domain-specific knowledge.

Off-the-shelf computer programs, such as Tableau [132] or Grapheur [22], [30], can be readily used to perform the visualisation of Comma-Separated Values (CSV) files using Parallel Coordinate Plots and scatterplots. However, off-the-shelf computer programs are generalist and obviously not dedicated to a given problem. In which case, a dedicated data visualisation program customised to perform a given task may be required.

2.4 Data visualisation

Data visualisation concerns the processing of sampled and computed data for comprehensive display. The aim of the visualisation is to provide to the user a deeper understanding of the data, in addition, to understand relationship within the data. Also, it is important to make the visualisation more interactive by allowing the user to rapidly discard useless information, focus on necessary information, and comprehend the science behind the data [14].

The use of InfoVis in evolutionary computing is rising. Jean-Daniel Fekete, leader of the Visual Analytics project (AVIZ) at Inria, gave a keynote talk on data visualisation at the Artificial Evolution 2017 conference to promote the use of these technology in evolutionary computing [135]. Also, Genetic Programming and Evolvable Machines (journal published by Springer Nature Switzerland AG) published a special issue in 2018 on “Genetic Programming, Evolutionary Computation and Visualization” [25]. These two examples have partly motivated our research.

There are different methods of data visualisation, here the study focuses on the methods that will help us to understand the performance and the behaviour of the EA, as well to find the link between the hyper-parameters in ML. Firstly, we aimed to find small regions in a medical image by locating points on it. For that purpose, using scatterplots was suitable. Also, to understand the implementation of the Fly algorithm we proposed a simple visualisation method relying on parallel coordinate plots. Parallel coordinate plots were useful to determine the best solution and early stopping criteria. In addition, we display different values for each group (peas and non-peas) that result from the multi-objective optimisation and we need to know the connection between them to find an appropriate threshold for cleaning the outputs to reach the best solution. Parallel Coordinate Plots gave us a simple and efficiency method to analyse the outputs from optimisation algorithm we used. Finally, we implemented ML as a classifier and for that we used various image sizes and number of epochs in training and testing. It was difficult to determine which combination of image sizes and number of epochs provides the best classification result. Radar chart provided an interactive visualisation to identify which combination of number of pixels/number of epochs gives the best possible solution. Below are brief of the

methods used in this research, namely scatterplots, parallel coordinate plots, and radar charts.

To implement these components we used two different implementation technologies. The first method is produced using a browser-based library, D3.js [24], which produces Structured Vector Graphics (SVG) images. The library is written in JavaScript, with the visualisation code also written in JavaScript. D3 includes multiple methods for loading and processing data. This system processes the CSV log files produced by the evolutionary process into JavaScript arrays. This system used to visualise the implementation of the FA only in chapter(Chapter 5). The second technology is depended on Python libraries Panda, Matplotlib and plotly, due to our code was written in Python. Python language has become one of the fastest-growing languages, because it is easy to learn, and flexible. Python's evolving libraries make it a good choice for data analysis [97].

2.4.1 Scatterplots

Scatterplots are one of the most widely used and most powerful techniques for data visualisation. They use a group of points located using Cartesian Coordinates to present values from two variables [94]. By showing a variable in each axis, the user can recognise if a relationship or correlation between the two variables exists. They are perfect for paired numerical data, when the user wants to know if one variable impacts the other [153]. An example of scatterplots can be seen in Figure 2.5. Moreover, we used scatterplots on the image to visualise the best solution from the implementation of the FA (see Figure 2.6).

2.4.2 Parallel Coordinate Plots

Parallel Coordinate Plots [66], first popularised in computerised form by Alfred Inselberg, visualise high-dimensional geometry and analyse data in the form of multiple linked axes on one graph. These axes are scaled such that each domain is represented in the same length. These axes represent different measurements, or facets, of the objects in the dataset. Objects, known as instances, are plotted as a traditional straight line graph on these co-measurable axes. These plots are used to identify clusters [162] and identify properties of those clusters/subsets [11]. The

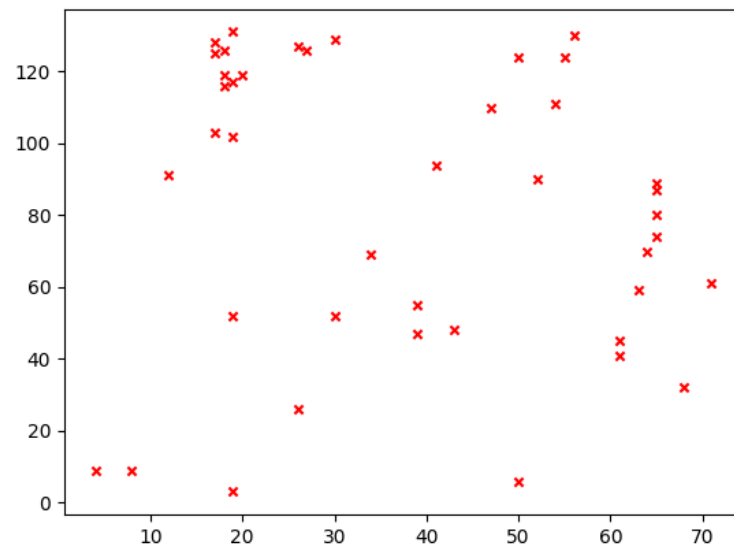


Figure 2.5: Example for scatterplots. Plot the points around the x-axes and y-axes for the value in rang (0-135) and (0-75).

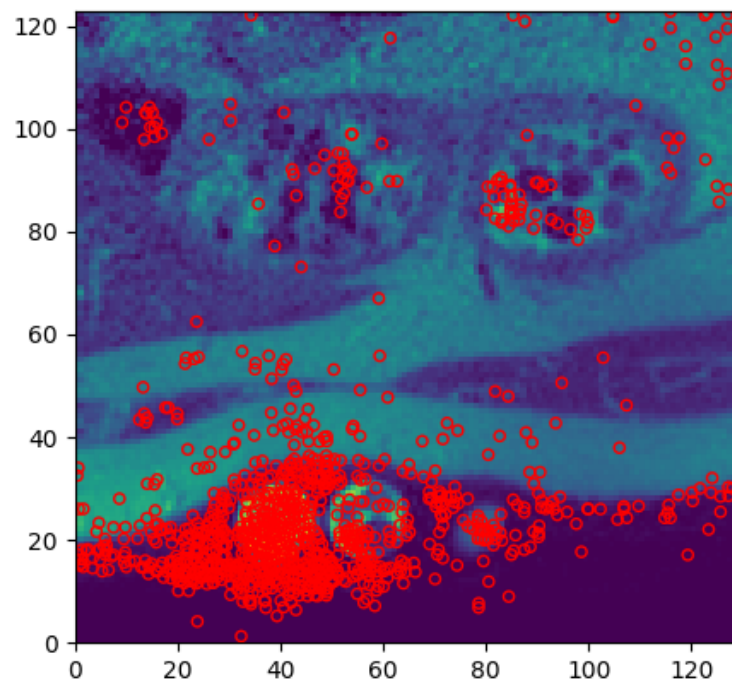


Figure 2.6: Example for scatterplots on MRI showing the final results for Fly algorithm implementation presented in Chapter 6.

tool uses different colour space, this results in the perceived difference in plot colour being proportional to the Euclidean distance of the colouring metric, i.e. items close to each other in the metric space will be similarly coloured in the plot. An example of this version can be seen in Figure 2.7.

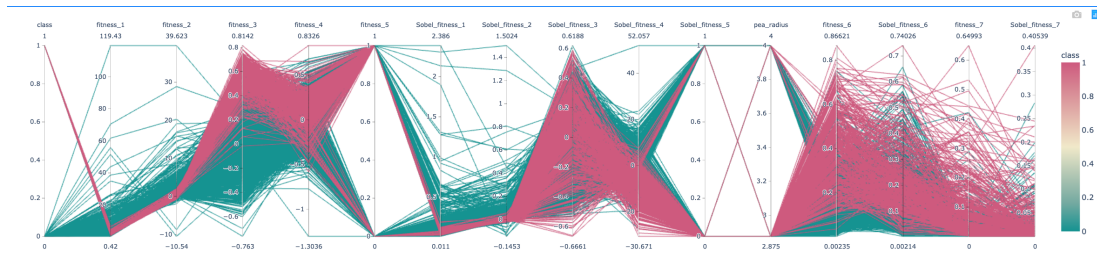


Figure 2.7: Example for Parallel coordinate plot that we used to analyse the output generated by NSGA-II for pea classification in MRI images (see Chapter 6).

Parallel Coordinate Plots most often deal with ranges rather than individuals. When selecting ranges of data, interactivity using the Brushing technique [89]. This allows the user to select multiple items in one stroke as if they were being painted with a brush. This implementation fades un-brushed lines to grey and leaving those of interest in their original colour. An example of this version can be seen in Figure 2.8.

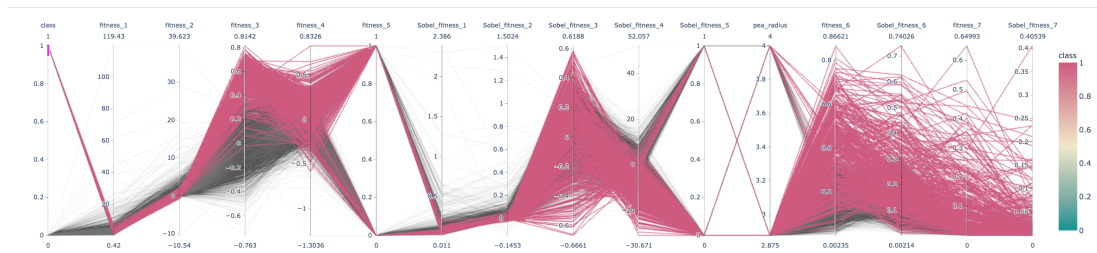


Figure 2.8: Same Parallel coordinate plot as in Figure 2.7 when data filtering has been applied using brushing (some of the data has been selected and remained in the same colour as previously, the excluded data is shown in grey to make it less visible).

2.4.3 Radar Chart

The radar chart, also known as a Spider chart, Web Chart, Polar Chart and Star Plots, is a data visualisation approach to visualise the multidimensional data in the two-dimensional scale Fig. 2.9 is an example of radar chart. This makes them helpful for understanding which variables have similar values or if there are any outliers amongst each variable [161]. Each variable is represented with an axis that starts originally from the centre point. All axes are spread out radially, with equivalent distances between each other, while keeping the same scale between all axes. To connect from axis-to-axis grid line served which are also used as a guide.

The variable values from the dataset are plotted, each value along its individual axis, and they are linked together to form a polygon [95]. Below is an example using a radar chart in this research:

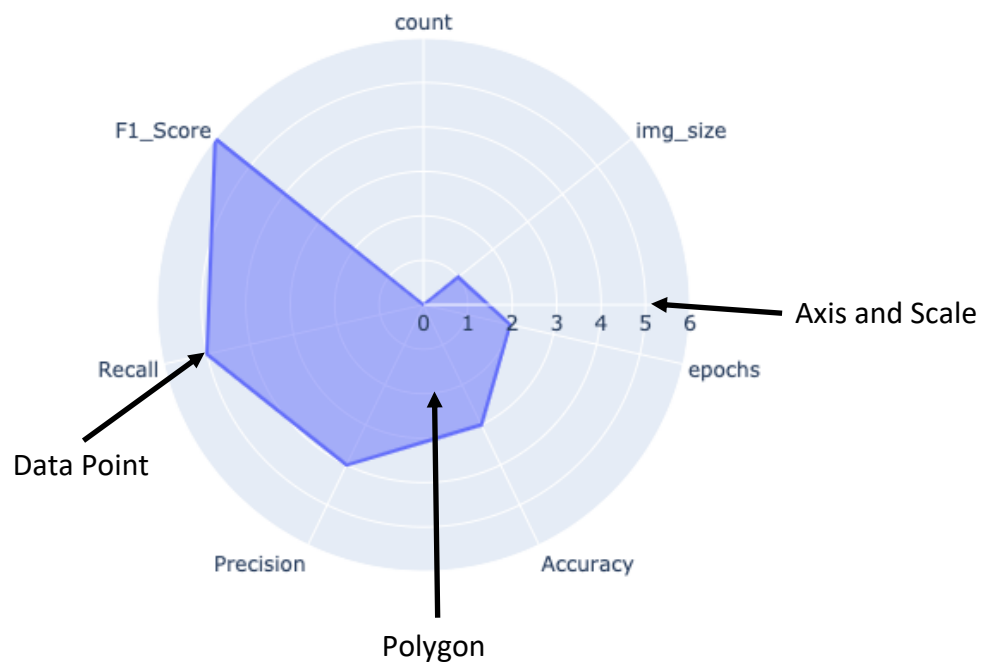


Figure 2.9: Example for Radar chart. The axes represent some statistics on the images we used in the ML in Chapter 7.

Chapter 3

Parisian evolution and Fly algorithm: A review

3.1 Introduction

This chapter presents an optimisation approach called *Fly Algorithm (FA)*, which embeds the searched solution into the whole population, only assigning a small part of the solution to each individual, compared with a few other traditional optimisation approaches that encode a solution into a single individual in a population. The Fly Algorithm was originally designed for a CV task (stereovision) based on 2-D or 3-D geometrical features. Its efficiency has been explained as being due to a parsimonious use of the evolutionary components “The collective search ability of a population”. Contrary to classical optimisation approaches (genetic algorithms for optimisation) that encode a solution into a single individual in a population.

This work introduces the co-evolutionary paradigm that can be in general classified into two main classes namely, competitive co-evolution and cooperative co-evolution. For competitive co-evolution, the different groups of individuals will always fight to get an advantage over the others. However, for the cooperative co-evolution, groups of individuals will replace information within each other through the evolutionary process [56]. This thesis focuses on the cooperative co-evolution algorithms. The Fly Algorithm –and more generally Parisian approaches / cooperative co-evolution– adopt a distributed approach involving several individuals to collectively represent the searched solution [36].

In the last few years, the heuristic and meta-heuristic optimisation communities have started an in-depth cost-benefit analysis on the proliferation of new techniques [129]. This chapter is highlighting its significant differences with popular techniques that might seem similar at a first glance, such as PSO and Real-Coded Genetic Algorithm (RCGA), and showcasing a few selected case studies on which the Fly Algorithm is able to outperform PSO and RCGA as well as the state-of-the-art CMA-ES.

The Fly Algorithm is implemented directly on top of a RCGA on which a ‘global’ fitness function was added to assess the performance of the population as a whole. To some extent, the FA also shares common features with Swarm Intelligence approaches, in particular with PSO as it relies on collective animal-like behaviour. However the study will see in the sequel that there are some major algorithmic differences, besides the fact that FA uses generational mechanisms while PSO uses swarm communication. Also will show several examples of Fly Algorithm implementations and systematic comparisons with other algorithms (including PSO-types and RCGA), to give some clues about the fundamental mechanisms involved, and recommendations for an efficient implementation. The Python implementation of all these algorithms and the test data provide on GitHub (<https://github.com/Shatha1978/Optimisation-algorithm-examples>).

The next section introduce the EC, highlights EA approach and reviews in detail how EAs can be built. Section 3.6 describes PSO and its algorithm. Section 3.7 introduces Cooperative co-evolution and Parisian Evolution, the two classes of optimisation schemes on which the Fly Algorithm is based. Section 3.7.1 provides details about the Fly Algorithm, including a brief history of its development. Section 3.7.2 discusses the Fly Algorithm’s initial application, stereovision. To highlight the different between the three algorithms see Section 3.8. The conclusion of this work can see in Section 3.9.

3.2 Evolutionary Computing (EC)

EC is a large class of stochastic search algorithms. It is widely used in diverse application domains. Fig. 3.1 shows a possible classification for the main search paradigm and highlights EAs [13]. They heavily rely on the theory of biological

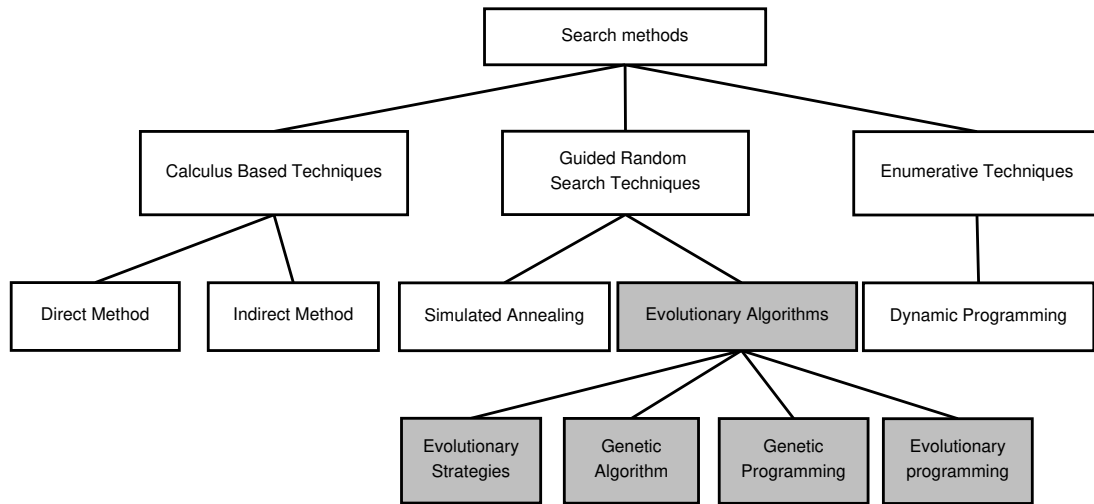


Figure 3.1: Different classes of search methods. The branch corresponding to evolutionary computing is highlighted in grey.

evolution proposed by Charles Darwin published in 1859 [39]. EAs can find an optimum solution from set solutions generated from the search space problem. EAs are considered as metaheuristic optimisation algorithms. Metaheuristics are the higher-level procedures purposed to find, produce, or select a lower-level procedures or heuristics which may perform a partial search. They are suitable for various optimisation problems with limited computation capability and having insufficient or imperfect information. In such situations, it provides an adequately good solution. An EA works on a generic population of individuals. Evolution strategies, genetic algorithms, evolutionary programming, all these techniques have essential commonality: generation, random variation, competition and selection [144].

The principles of EAs emerged in the mid-1950s, when researchers were following various approaches to mimic different concepts in natural evolution, but all of them relied on the principles of natural behaviors from Darwin's theory of evolution initially published in 1859 [47]. Due to the lack of computing power the field of EAs has been pioneered later in the 60s and 70s. Lawrence J. Fogel introduced Evolutionary Programming (EP) in the 60s [32], while John Henry Holland called his method GA [63] and Ingo Rechenberg and Hans-Paul Schwefel [107], [121] proposed Evolution Strategies (ES).

All the evolution techniques mentioned above depend on the natural evolution of a population through generations. Fig. 3.2 illustrates the general loop of these

algorithms. Basically, it manipulates a population, which involves a number of individuals, the parents. It means that parents are going to breed to produce children. In turn the children would become parents and reproduce, and so on.

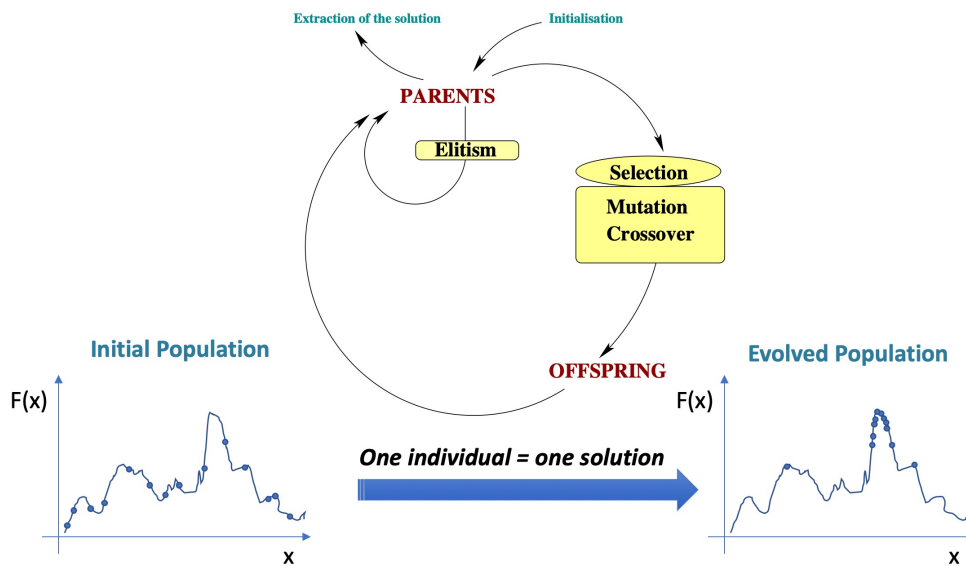


Figure 3.2: Classical EAs for optimisation: a solutions is encoded into an individual of the population. The population evolves a series of potential solutions and concentrates on the “best” areas of the search space (here a maximisation example).

There are three essential procedures for all EAs. The first procedure is to create an initial population of individuals, which is randomly generated. Each individual is defined by a sequence of genes (see Section 3.3). The way genes are encoded is problem-dependant. A quality function, which is called “fitness function” in EC, is used to evaluate every individual of the population. This is the second procedure. It implies that EAs are written as maximisation problems. The best individuals have a higher fitness value than the weakest individuals. Candidates for reproduction will be selected based on the fitness values (see Section 3.4). It means that the strongest individuals have more chance to survive and produce the offspring, whilst, the weakest individuals will died out, and it used for the purpose of selection. The third procedure is developing the population toward an optimum solution. It is done by repeatedly applying a set of genetic operators to each individual of the population at each generation to produce a new population (see Section 3.5). The evolutionary loop will stop when a stopping criterion is met. It can be static or dynamic. Commonly used static criteria are the total number of new generations created or the total number of times the fitness function was computed. Computation

time is another popular static criterion. Stagnation, on the other hand, is a dynamic criterion. Stagnation occurs when there is no chance to achieve important changes in fitness value in the next generations [117].

The algorithm presented in (see Alg. 1) is a generational implementation, which corresponds to the scheme of the *Simple Genetic Algorithm*. This algorithm uses non-overlapping populations and an optional elitism mechanism. The population is replaced at each generation with new individuals. Another way to implement an EA is *the steady-state Genetic Algorithm*, which uses an overlapping population. In this implementation, a portion of parents should be replaced by new individuals (bad parents replaced by good offspring) [148].

3.3 Solution encoding

The population usually starts with random individuals. There are various ways to represent the solution depending on the problem to be solved [105]. Fig. 3.3 shows examples of encoding schemes for EAs.

Binary String: A typical representation of the individuals is as an array of bits. This representation uses a sequence of 0s and 1s to represent each individual (chromosome). This encoding can be found in the traditional “Genetic Algorithm”.

Permutation or Integer Encoding: To symbolise the individuals in these encoding model, a sequence of integer numbers is used. It is useful for ordering problems such as the Travelling Salesman Problem (TSP).

Real (floating-point) Encoding: Each individual is a sequence of real value, e.g., 0.05, 2.5, 0.9, . . . , etc. This model is called “real-valued Genetic Algorithm” in EAs.

Value Encoding: Each individual is represented as a combination of various values such as integer, real, string and object, which depend on the search space of the problem to solve [130].

Tree encoding: The individual is encoded as a tree of objects. It is commonly used with problems that involve genetic programming [57], [87].

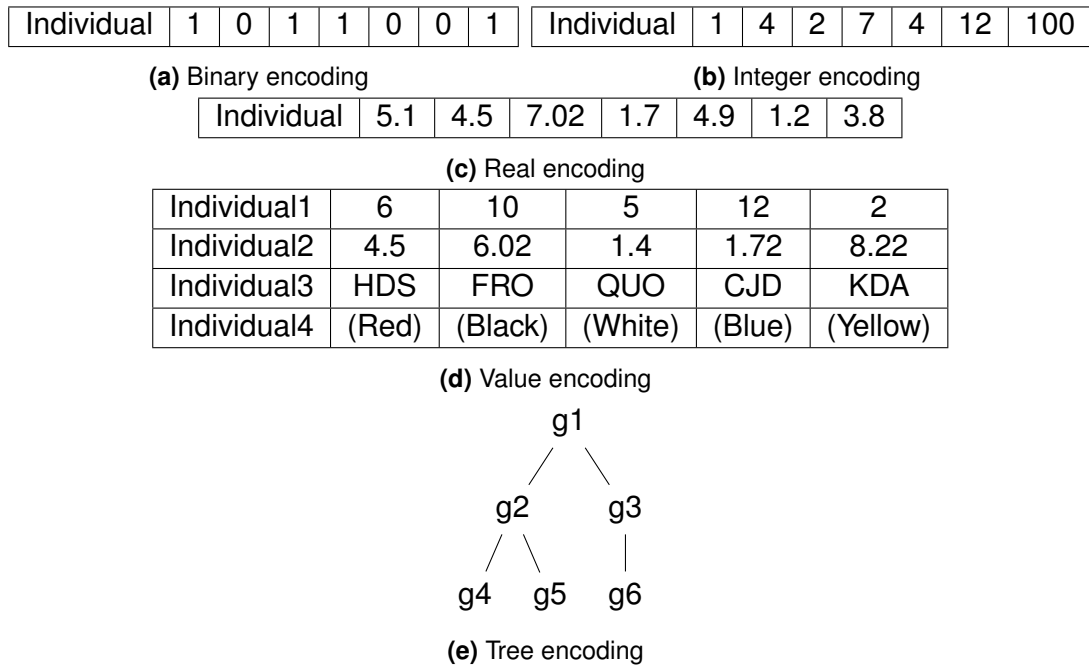


Figure 3.3: Schemes of Genetic Encoding.

3.4 Selection operator

The selection operator statistically selects good solutions for building the next generation, with a bias toward better solutions. Population size is usually kept constant. The selection operator chooses the individuals (parents) in the population with highest probability to survive and generate the new offspring for the next generation [99]. The selection operator uses the fitness of each individual [118], to provide a probability distribution. If maximisation is considered, then individuals with the highest fitness will be more likely to be selected. There are five main traditional selection methods.

Tournament selection In this selection operator, some individuals are randomly picked up from the current population. The best individual (based on the fitness values) wins the competition and will be selected. The number of individuals involved in each selection iteration is called the “tournament size” [106].

Proportional selection or roulette wheel selection This selection operator is considered as one of the most popular and easiest techniques to implement. The individuals with the highest fitness values have the greatest chance to survive and

to be selected as parent. Each individual i in current population has a probability $P(i)$ of being selected relative to its fitness $fitness(i)$. This probability is given by:

$$P(i) = \frac{fitness(i)}{\sum_{j=1}^N fitness(j)} \quad (3.1)$$

where N is the population size [67].

Rank-based selection In this technique the individuals are sorted into a list base on their fitness value. The rank N represents the best individual and the worst individual gets rank 1. The selection probability of an individual is given by:

$$P(i) = \frac{rank(i)}{N \times (N - 1)} \quad (3.2)$$

where N is the population size [67].

Threshold selection This method is widely used for image processing. It can be categorised as bi-level threshold and multilevel threshold. Bi-level threshold classifies the pixels into two groups. Multilevel threshold divides the pixels into several groups [160].

Elitist strategy the best individuals should always be part of the reproduction. However, there is no guaranty that it will be the case, and if it occurs, these individuals may be destroyed by the crossover or mutation operators. To address this deficiency, elitism makes sure that few of the best individuals are copied into the new population [123]. Firstly, the individuals are arranged in decreasing order. After that apply the selection with each two individuals in arrange set. In this method selection is between strong individuals or weak individuals. This means there is no need to apply GA between week and strong individuals, because the best individuals can pass to next generation directly [99].

3.5 Genetic Engine

Genetic operators are used to produce a new individuals in each generation, such as crossover (also called recombination), mutation, possibly elitism and a diversity

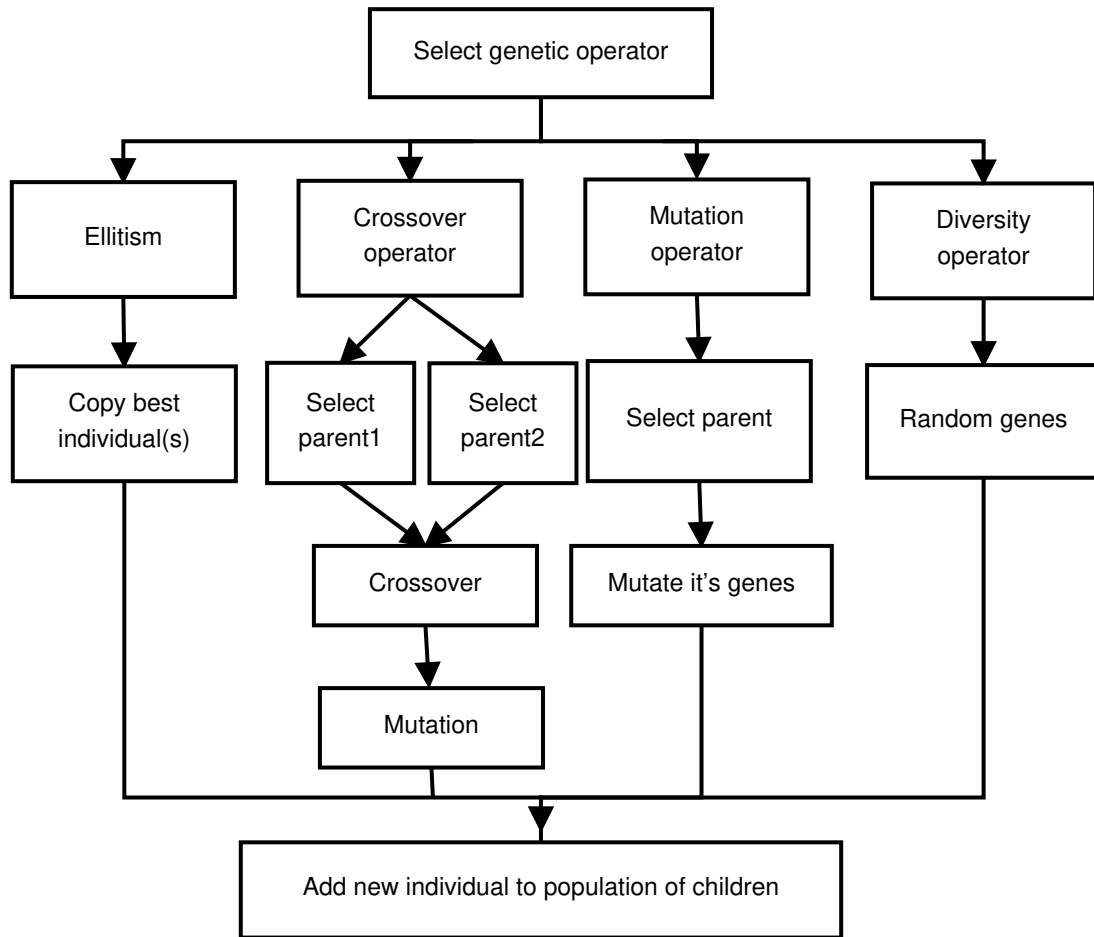


Figure 3.4: Genetic Operators for Evolutionary Algorithm.

mechanism (new blood) [130]. Fig. 3.4 shows some of the most common basic operators used in EA.

3.5.1 Crossover operator (Recombination)

This is an important and random operator in EAs. The occupation of the crossover operator is to produce new “child” individual from two “parent” individuals by merging the information extracted from the parents [152]. Its probability of occurring (P_c) usually ranges between 0 to the maximum height of the task. Pairs of individuals are randomly selected to produce two new individuals. The two parents swap some of their genes depending on crossover points [64]. To determine these points, there are several possible mechanisms. Fig. 3.5 demonstrates the different crossover strategies.

Single-point crossover (1x) In this operator, one integer position k is randomly chosen from the parents and it is between 1 and the parent length l . Then, all genes

values between $k + 1$ and l position are swapped to produce a new individuals. The selected position k can be among any gene position from the first through the last gene. This strategy is very fast, but it has the problem of reducing diversity especially when some individuals are already similar in the population.

Two-points crossover (2x) In this strategy, two cut-points are randomly chosen k_1 and k_2 instead of one. Then the sequence of genes values between them are switched to produce a new individual.

Uniform crossover (U_x) In uniform crossover, a gene value from the parents is assigned to the first a new individual and the second individual with a probability value (P_c). That means randomly mix the genes of the parents by probability value (P_c) to produce two new individuals [88].

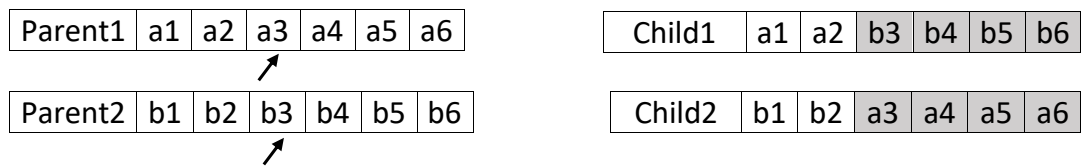
Tree-encoding crossover In this technique the two parent tree are combined to produce two new individuals trees. In theory take subtree from random position in one parent tree and exchange it with the random subtree from the other parent tree. There are no constraints on the subtrees selection. It can choose any node from the tree even the whole tree can be chosen as a crossover point [54].

3.5.2 Mutation operator

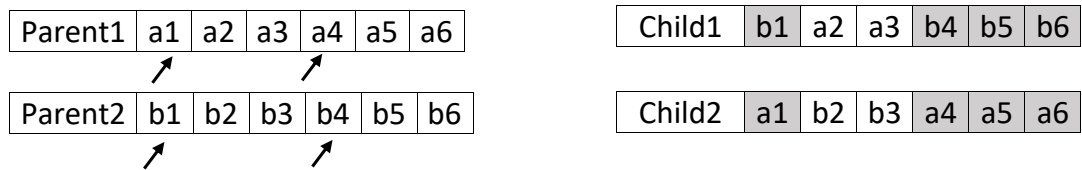
Mutation operators are used as a mechanism for keeping variety in the population and helps ensure that no point in the search space has a zero probability. It can be considered as an occasional (with a small probability P_m typically 0.001) random variation on the value of one or more gene values for one individual at a time [19]. Basically, mutation impacts the whole population because it is increasing the diversity within it. Therefore, it inhibits stagnation in the convergence of the optimisation technique. The general form of mutation can be written as:

$$x' = x + M \quad (3.3)$$

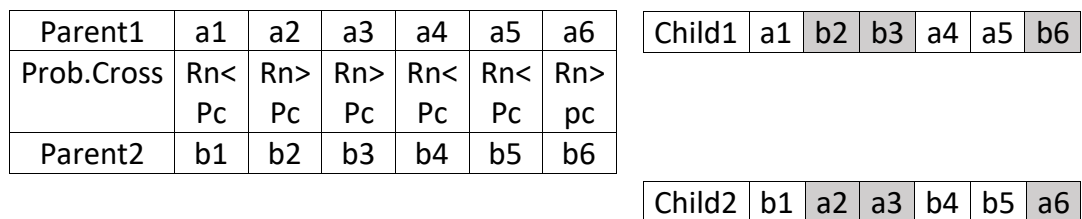
where x is the parent vector, M is a random variable (mutation rate) and x' is the offspring vector.



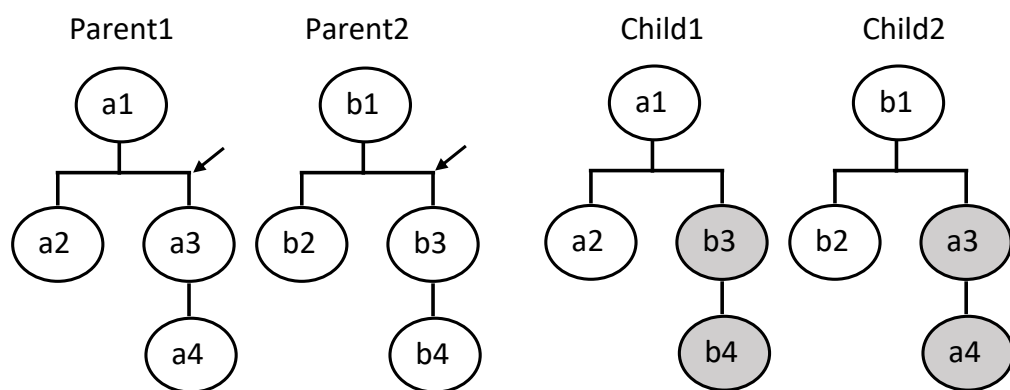
(a) Single-point



(b) Two-points



(c) Uniform



(d) Tree encoding

Figure 3.5: Different strategies for crossover operator.

There are different type of mutation operators [6]:

Flip Bit (Single-point) mutation A single gene value in the individual at a selected mutation point is changed into other value in the range of this gene. When the gene has a binary encoding, the value is flipped from “1” to “0” and *vice versa*.

Uniform mutation This is a classic technique in which each gene of an individual has an equal likelihood to be mutated by any value in the solution space [44].

Boundary mutation This is a special case of uniform mutation. It is used with integer or floating-point number encoding. The newly generated allele z_i is either the upper bound (UB) or the lower bound (LB) of the domain, with equal probability:

$$z_i = \begin{cases} UB & \text{if } x > 0.5 \\ LB & \text{otherwise} \end{cases} \quad (3.4)$$

where x is a random number between 0 and 1.

Tree encoding mutation This mutates selected nodes of the tree, with a new sub tree, to create new offspring.

Gaussian mutation This strategy uses a mutation operator to change an individual value using a random number drawn from a Gaussian distribution. The selected individual changes using the function:

$$x' = x(1 + \text{Gaussian}(\sigma)) \quad (3.5)$$

where $\text{Gaussian}(\sigma)$ returns a random number brought form a Gaussian distribution with a standard deviation of σ [10].

3.6 Particle Swarm Optimization

PSO was originally proposed by J. Kennedy and R. Eberhart in 1995 [72] and was very popular for simulating social behaviour and movements of groups of organisms (flocks of birds, shoal of fish) [125]. This approach, as well as Ant Colony Algorithms, Artificial Bee Colony Algorithms and Bacterial Foragings [69], are examples of the so-called *swarm intelligence*. The swarm as a whole collectively accomplishes a task, like a flock of birds searching for food. It is possible to turn this model into an optimisation algorithm (i.e., search for an optimum over a search space).

PSO is similar to EAs in that it iteratively manipulates a population, starting from a population of random solutions [46]. But its evolution occurs at a different scale, in terms of interpretation: It relies on a simulation of the movements of the swarm instead of a death and birth cycle as in EAs. Each possible solution of a PSO, called particle, has a position in space x_i and a velocity v_i that determines a randomised movement till the next iteration. At each iteration velocity and position are updated using some rules, keeping in memory the best-so-far local and collective solutions, respectively $pbest$ and $gbest$, continuously updated along iterations. The swarm stabilises on optimal areas of the search space [127]. An estimation of the optimum is given by the best particle of the evolved swarm [72]. PSO's main steps are as follows:

1. Initialise each particle with a random position in the problem search space limits and a random velocity.
2. For each time step (t) , record $gbest(t)$ for the swarm and for each Particle i record $pbest_i(t)$.
3. Update position x_i and velocity v_i of each particle i using:

$$v_i(t+1) = \omega v_i(t) + \varphi_p r_p (pbest_i(t) - x_i) + \varphi_g r_g (gbest(t) - x_i) \quad (3.6)$$

$$x_i(t+1) = x_i(t) + v_i(t) \quad (3.7)$$

where r_p and r_g are random values, uniformly distributed in $[0, 1]$, $pbest_i(t)$ is the best known position for Particle i , $gbest(t)$ best known position for the whole swarm [126]. The first term of Equation 3.6 only depends on the particle's current velocity. ω is a parameter that corresponds to the inertia weight. The second term depends on the distance between the current position of the particle and its own optimal position ($pbest_i(t)$). φ_p is the cognitive learning factor. The third term is the social contribution, which takes into account information from the swarm. It depends on the distance between the current position of the particle and the current best position of the swarm ($gbest(t)$). φ_g is the social learning factor.

4. Repeat from Step (2) until a stopping criterion is reached (e.g., a given level of fitness or a maximum number of iterations).

For a given function and search space, the performance of PSO depends on the parameter setting: ω , φ_p and φ_g [150]. This scheme, called *gbest strategy*, is the most common one. Particles are “fully informed” as they are aware of the state of the whole population. There also exist various so-called *lbest strategies* where each particle only have access to local information [104]. The update rule in Equation 3.6) is the same except that $lbest_i(t)$, a best *local* position, is used instead of $gbest(t)$. These schemes rely on the definition of a neighbourhood distance in the space of the particles. A balance should be found between the benefits of such a refined local approach with respect to the additional computations it requires. Algorithms 2 displays the PSO algorithm.

3.7 Cooperative co-evolution and Parisian Evolution

Instead of representing a solution as a single individual, it is possible to distribute the potential solution over several individuals in the population. In this case the individuals have to co-operate to build a solution to the problem. The quality (or fitness) of an individual then relies on its relationship with rest of the population. This strategy is known as “co-evolution”. Apart from its use for theoretical and empirical studies about population behaviour, early examples of this technique are the famous classifier systems, or “Michigan approach” [156]. The original idea,

ALGORITHM 1: Generational real-coded Genetic Algorithm. We consider here a problem with a k -D search space.

```

    // Read problem specific data
    // Set the algorithm
Initialization
// Create the initial population
  of  $n$  individuals
repeat  $n$  times
  | Create an individual with  $k$  random
  |   genes;
  |
  | Add the individual to the population;
end

repeat           // Optimization loop
  | foreach individual  $i \in \text{population}$  do
  |   Compute fitness function;
  |
  | end
  | Initialize empty list of offspring;
  | for  $i = 0$  to  $n - 1$  do
  |   | Select parents from population;
  |   |  $child \leftarrow$  Crossover between
  |   |   selected parents;
  |   | Mutate  $child$ ;
  |   | Add  $child$  to list of offspring;
  | end
  | population  $\leftarrow$  offspring;
until Convergence;
Extract best individual;

```

ALGORITHM 2: PSO. We consider here a problem with a k -D search space.

```

    // Read problem specific data
    // Set the algorithm
Initialization
// Create the initial swarm of  $n$ 
  particle
repeat  $n$  times
  | Create a particle  $i$  at a random
  |   position  $x_i = (x_{i,1}, \dots, x_{i,k})$ ;
  |   Initialize the particle's velocity
  |    $v_i = (v_{i,1}, \dots, v_{i,k})$ ;
  |   Initialize the particle's best known
  |   position  $pbest_i$ ;
  |   Add the particle to the swarm;
end

repeat           // Optimization loop
  | foreach Particle  $p_i \in \text{Swarm}$  do
  |    $f(p_i) \leftarrow$   $p_i$ 's fitness value;
  |   if  $f(p_i) > f(gbest)$  then
  |     |  $gbest \leftarrow p_i$ 
  |   end
  |   if  $f(p_i) > f(pbest_i)$  then
  |     |  $pbest_i \leftarrow p_i$ 
  |   end
  | end
  | foreach Particle  $p_i \in \text{Swarm}$  do
  |   | Update  $p_i$ 's velocity;
  |   |  $x_i \leftarrow x_i + v_i$  // Update  $p_i$ 's
  |   |   position
  | end
until Convergence;
Extract best particle;

```

applied to rule based machine learning, was to evolve a set of individuals, each being a rule, that collectively achieve a given task.

Since this pioneering work, the way co-evolution is structured and exploited in optimisation varied widely in literature. A first distinction can be made according to the interacting behaviour, competitive versus cooperative, depending on the nature of the reward (positive or negative) when individuals interact. Competitive models have been widely studied, but in the last two decades, one can notice an increased interest in cooperation to tackle difficult optimisation problems by means of problem decomposition [18], [23], [36], [41], [143], [155].

Cooperative strategies can also be divided into approaches using a single population of interbreeding individuals, or maintaining multiple interacting populations [98].

The Parisian approach (see Fig. 3.6) is a representative of the Single-Population-Cooperative-Co-evolution category. The idea is to exploit the evolution mechanism in a more parsimonious way: where a traditional EA only keeps the best individual as an optimum solution at the end of the evolution (forgetting all precious information gathered by the population during its exploration of the search space). A Parisian approach tries to capitalise the full potential of an evolved population. In this model, the population is thus considered as a collective in which individuals collaborate with a common goal. This is implemented using a classical EA with all the usual features (e.g., mutation, crossover, and selection), but with two possible levels of fitness:

A local fitness to assess the performance of a single individual (partial evaluation).

It is used during the selection process. For an individual, improving its local fitness means increasing its chances of breeding.

A global fitness to assess the collective performance of the whole population.

Improving (maximising or minimising) the global fitness is the goal of the population.

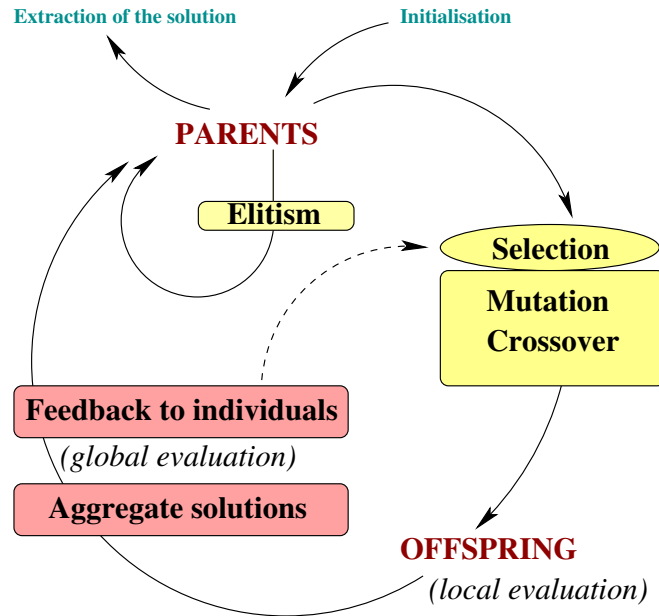


Figure 3.6: Cooperative Co-evolution scheme or Parisian EAs: several individuals (sometimes the whole population) represent a solution to the problem. The loop is similar to classical optimisation applications but embeds an additional step in the main loop that aggregates individuals to build a solution, evaluate it and distribute reward to individuals.

In addition, a diversity mechanism is required to avoid degenerate solutions, where individuals gather in only a few areas of the search space. A further difference between classical EA and Parisian Evolution resides in the extraction of the solution once the evolutionary loop terminates. All the individuals (or individuals of a sub-group of the population) are collated to build the problem solution. The way the fitness functions are constructed and the way the solution is extracted, are problem-dependent.

Parisian Evolution has been successfully applied to various optimisation problems, such as text-mining [77], hand gesture recognition [71], complex interaction modelling in industrial agrifood processes [16], [17], and imaging problems [35] such as computer stereo vision in robotics [82] and tomography reconstruction in medical physics [8].

3.7.1 A special case of Parisian Evolution: The Fly Algorithm

The Fly Algorithm is a good example of Parisian Evolution [27]. It was initially proposed in computer stereo vision (see more details in Section 3.7.2) to extract 3-D information from pairs of digital images. The algorithm is a fast evolutionary

algorithm that can be used to detect the location of obstacles [82], [83]. It is used in autonomous robot navigation to avoid collision with objects and walls.

The Fly Algorithm evolves a population of flies. Each fly is defined as a 3-D point with coordinates (x, y, z) in the solution space. A set of 3-D points is often called *point cloud* in the literature. Flies are projected to compute the local fitness function. This projection operator is problem-specific. In stereo vision applications, each fly is projected twice: once on the image taken from the left camera and once on the image taken from the right camera [84]. When a fly is located on the surface of an object, the pixel neighbourhood of its two projections will match; when a fly is not located on the surface of an object, the pixel neighbourhood of its two projections will be significantly different. The fitness function is designed to take advantage of this fact: The fitness of a fly measures the consistency between its two projections. The algorithm will optimise the 3-D position of the flies so that their projections on the left-hand side and right-hand side 2-D images are similar.

The Fly Algorithm is implemented as any other EA. It starts with a population of randomly generated individuals. They are the *parents*. Then, depending on the genetic operators (selection, mutation, new blood, etc.) that are applied to the parents, a population of new individuals, the *offspring*, is produced. Selection is used to pick up candidate parents for breeding. Mutation is used to randomly alter the genes of an individual. New blood corresponds to creating a randomly generated individual. This simple, but yet effective, operator preserves diversity in the population. Note that crossover is not generally used in the Fly Algorithm because if there are two good flies on different objects, creating a new one in between is likely to produce a bad fly. The fitness function determines the validity of a fly's position and it is calculated during the selection process. The new generation of offspring eventually becomes parents. The same operations are repeated until a stopping criterion is reached. This approach is called 'Generational Fly Algorithm' [120].

A Steady-State approach is also possible. Algorithm 3 provides the pseudo code of the steady state Fly algorithm using i) a marginal fitness for the selection of individuals, ii) immigration, and iii) mutation. Using this approach, a bad fly is

selected at each iteration and replaced by a new one. The rationale is that the new one is likely to be better and there is no reason to delay using it [120].

ALGORITHM 3: Steady state Fly algorithm using i) a marginal fitness for the selection of individuals, ii) immigration, and iii) mutation.

```

Initialization                                // e.g., read problem specific data, and
                                              // set the algorithm's parameters

// Create the initial population of  $n$  individuals
for  $i = 0$  to  $n - 1$  do
    Create a fly at a random position in the search space;
    Add the fly to the population;
    Add the fly's contribution to the population's;           // optional
end

Compute the global fitness;                     // optional

repeat                                       // Optimization loop
    repeat                                   // Select a bad fly
         $i \leftarrow \text{Random}(0, n - 1)$ ;
         $\text{MF}(i) \leftarrow \text{Marginal fitness of Fly } i$ ;
    until  $\text{MF}(i) \leq 0$ ;
    Remove Fly( $i$ )'s contribution from the population's;      // optional
    Compute the global fitness;                             // optional

    Select genetic operator;
    if Genetic operator is immigration then
        Replace Fly( $i$ ) with a fly at a random position in the search space;
    else                                       // Mutation is used
        repeat                               // Select a good fly
             $j \leftarrow \text{Random}(0, n - 1)$ ;
             $\text{MF}(j) \leftarrow \text{Marginal fitness of Fly } j$ ;
        until  $\text{MF}(j) > 0$ ;
        Copy Fly( $j$ )'s genes into Fly( $i$ )'s;
        Randomly alter Fly( $i$ )'s genes by mutation;
    end
    Add Fly( $i$ )'s contribution to the population's;           // optional
    Compute the global fitness;                             // optional

until Convergence;

Iteratively eliminate bad flies from the population;        // optional

Convert the population of flies into problem specific answer;

```

A decade after the initial developments of the Fly Algorithm in robotics, it was adapted to SPECT reconstruction [28], then to PET [3], [8], [139]–[142]. It has also be used in filtering to generate artistic effects on images [2], [4], [5].

3.7.2 Fly Algorithm for Stereovision

Conventional approaches to stereovision use extensive calculation in the 2-D image space to perform primitive extraction, followed by matching and calculation of disparities in order to get 3-D information. The original Fly Algorithm uses an “inverse problem” approach that directly evolves a population of individuals, the “flies”, in the 3-D space [82], [83], in a way similar to the “prediction-verification paradigm”. The population of flies is initialised randomly in the field of view common to at least two cameras (Fig. 3.7). An evolution strategy is used to evolve the population of flies so that they will concentrate on the visible surfaces of the objects present in the scene.

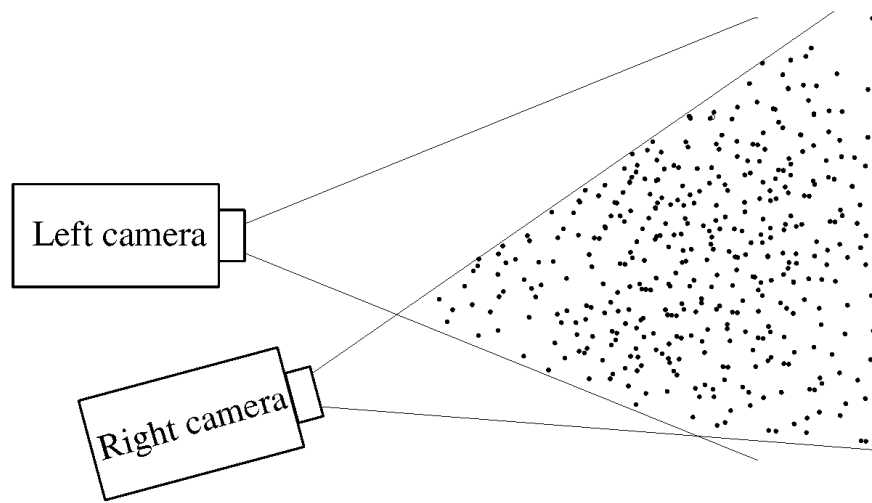


Figure 3.7: The flies are initialised randomly within the intersection of the cameras' 3-D fields of view [84].

Evolution is guided by the flies' fitness values. Each fly is projected onto the image planes as defined by the cameras (see Fig. 3.8). If the fly lies on the surface of an object, then the pixel values of its projections are usually similar; otherwise, if the fly is not on an object's surface, the pixels corresponding to its projections will not represent the radiance and colour of the same physical point in the scene, and therefore will be likely different (Fig. 3.8). The fitness of a fly will reflect the degree of similarity of its projections into the images given by the cameras: flies on visible objects surfaces will thus get higher fitness values. About any camera configuration can be used; however, if the distance from the cameras to the object is large enough compared to the distance between cameras, a correlation or a texture comparison can be introduced into the fitness function.

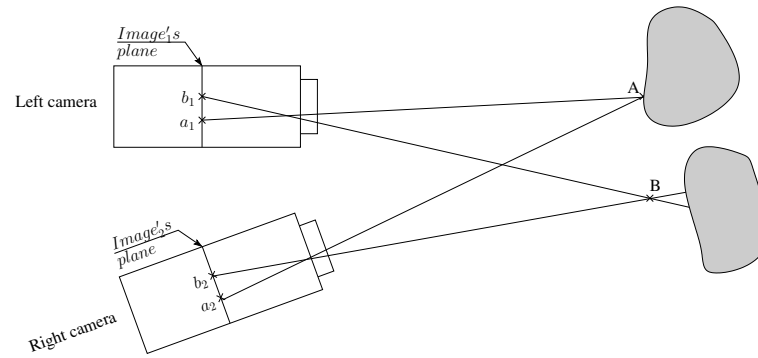


Figure 3.8: Projections b_1 and b_2 of fly B which lies on the surface of an object, will have identical gray levels, unlike pixels a_1 and a_2 . This holds true independently of camera geometry.

3.8 Comparison

A key difference between genetic algorithms/Fly Algorithms and PSO lies in the evolution mechanism: Death and birth of individuals versus movements of particles and intra-swarm information transmission. In algorithmic terms, a parallel can be found between mutations and movements, but this actually leads to a different balance between diversification and intensification [68]. In particular, selection is not used in canonical PSO, which participates explicitly to intensification. Additionally, diversity preservation mechanisms are more explicit and tunable in Fly Algorithms thanks to the “immigration” operator (which explicitly maintains a random – or “novelty” – part into the population). This feature of the Fly Algorithm is important and efficient, as experimentally proven on a variety of inverse problems. Indeed, if the initial population does not cover all the objects that need to be found (e.g., in stereovision or PET reconstruction), bad flies will be destroyed and replaced by new flies in the vicinity of existing good flies. In this case it is likely that some of the small objects will be missed. The introduction of a suitable diversity preservation mechanism, such as immigration, addresses this problem.

Table 3.1 highlights the salient features, similarities, differences and the test case of the three algorithms, when considering a problem with a k -D search space.

Table 3.1: The similar features of Real-Coded GA, PSO, and FA. Consider here a problem with a k -D search space.

Algorithm's features	RCGA	PSO	Fly
SIMILARITIES			
Initialisation	Random population.	Random particles.	Random flies.

Iterations	Repeat genetic loop until a stopping criterion is reached.	Update positions and velocity until a stopping criterion is reached.	Mutate flies until a stopping criterion is reached.
Algorithm's features	RCGA	PSO	Fly
DIFFERENCES			
Size of an individual or a particle	k	$2k$	$s = k/M$
Representation of the problem	An individual is a sequence of k genes.	A particle i has a position $x_i = (x_{i,1}, \dots, x_{i,k})$ and a velocity $v_i = (v_{i,1}, \dots, v_{i,k})$.	A fly i has a position $x_i = (x_{i,1}, \dots, x_{i,s})$ with s much smaller than k .
Population size	Free	Free	Constrained: At least M flies in the population, with $M \times s = k$.
Information sharing mechanism	Generational inheritance. Solutions are ranked according to fitness values. Recombination of 2 selected parents plus mutation for producing offspring.	Information communication inside the swarm through Eq. 3.6. Particles follow the swarm's best one by updating their velocity and position.	Generational inheritance + global fitness. The marginal fitness determines the flies that survive. Offspring are produced by mutation.
Intensification	Selection and crossover.	Use of $pbest_i$ and $gbest$, tuned by cognitive and social learning factors.	Selection, e.g., thresholding.
Diversification	Achieved mainly by the mutation operator (but also by the use of random generators in selection and crossover).	Random terms in the update of velocities for each particle.	Mutation and immigration operators.
Problem solution	One single individual, the best one.	One single particle, the best one.	Aggregation of a set of individuals.
Algorithm's features	RCGA	PSO	Fly
TEST CASES			
Lamp problem with N lamps	Each individual is made of $3 \times N$ values (x , y and on/off for each lamp, the radius being fixed).	Each particle is defined with $2 \times 3 \times N$ values.	Each individual is made of 3 values only and there are N individuals in the population.
2-D PET reconstruction with P emission points	Each individual is made of $2 \times P$ values (x and y for each emission point).	Each particle is made of $2 \times P$ values.	Each individual is made of 2 values only and there are P individuals in the population.

3.9 Conclusion

This chapter presented an overview of a cooperative-coevolution algorithm, the Fly Algorithm. At first glance, it may seem similar to popular techniques such as Real-Coded Genetic Algorithm and Particle Swarm Optimization. The Fly Algorithm is built on the top of a Real-Coded Genetic Algorithm, but uses two levels of fitness functions: A global one to assess the quality of the population as a whole and a local one to gauge the quality of individuals during the selection process. We also highlighted Fly Algorithm's common features and differences with swarm intelligence approaches: Despite their similar biological inspiration (behaviour of populations of insects), the mechanisms are rather different. FA uses generational mechanisms while PSO uses swarm communication.

Chapter 4

Comparison between the efficiency of FA, GA and PSO

4.1 Introduction

Cooperative co-evolution approaches have been successfully used to solve complex problems in various domains. They are based on a representation of the problem to be solved as a cooperative task, where individuals interact (cooperate or compete) to build a solution. Many strategies have been built depending on the way co-evolution occurs: How a solution to the problem is split between individuals, and how individual and global evaluations are balanced. The Fly Algorithm is a particular case of cooperative co-evolution, with added geometrical features. It is suitable for a range of problems where solutions can be represented by a collection of 2-D or 3-D points, or even points in higher dimensions.

Here we compare the performance of FA with traditional non-cooperative optimisation schemes, such as RCGA, PSO and CMA-ES, where a solution is represented by a single individual of the population. The comparison of these algorithms on two test cases: A toy problem, the Lamps; and a complex inverse problem, PET reconstruction. This choice is based on three facts: FA has been built on top of RCGA, the comparison yield an assessment of the cooperative component that has been added, PSO has been sometimes opposed to FA, and CMA-ES is considered as the state-of-the-art for continuous optimisation.

One of the purposes of this study is also to highlight the properties of cooperative schemes in comparison to classical ones. This is why it chooses to restrain the

study benchmarks to algorithms belonging to similar classes of complexities. There exist of course various more complex versions of the different algorithms tested below, able to better deal with large size problems, for instance for CMA-ES [137] or Fly Algorithm [9], [84].

The next Section 4.2, explains the two test cases used to evaluate the relative performance of a traditional PSO and a RCGA against the Fly Algorithm. They are used to illustrate the inappropriateness of some of the traditional operators and how new operators were designed (see Section 4.3). Quantitative results on the two test cases are provided in Section 4.4. Finally, Section 4.5 presents the conclusion of this study.

4.2 Two test cases for the Fly Algorithm

Cooperative co-evolution approaches have been successfully used to solve complex problems in various domains. They are based on a representation of the problem to be solved as a cooperative task, where individuals interact (cooperate or compete) to build a solution. Many strategies have been built depending on the way co-evolution occurs: How a solution to the problem is split between individuals, and how individual and global evaluations are balanced. This collaboration considered two test cases in experimental analysis (Section 4.4). The idea is to illustrate how a few implementation choices (e.g., generational vs. steady state, selection methods and genetic operators) can be made to take advantage of the Parisian Evolution paradigm. The first test case is called the “Lamps problem”, which is a toy problem. It is used to demonstrate how to implement the FA from the implementation of the Real-Coded Genetic Algorithm. The second test case is Positron emission tomography reconstruction, which is an example of real application in medical physics.

4.2.1 Lamps problem

The *Lamps problem* is an optimisation benchmark originally designed for cooperative co-evolution algorithms [136]. The basic idea of the benchmark is to cover a square area, representing a field, with optimally placed circles of same fixed radius, representing lamps. The evaluation function provides a reward to

each lamp, and a global reward that depends on the relative placement of all lamps. Interestingly, while each single lamp can be placed optimally, sometimes it might be better to accept an individual sub-optimal placement to improve the global reward. Fig. 4.1 shows an example where four lamps completely cover a square (see Fig. 4.1a), but part of their area is outside of the square itself; and a second situation (see Fig. 4.1b) where one of the lamps is completely inside the square, but the global solution is unable to completely cover the square [119].

The benchmark problem depends on only one parameter, the ratio between the radius of a lamp and a side of the square [136], or equivalently the ratio between the surface of a light versus the surface of the room:

$$problem_size = \frac{area_room}{area_lamp} \quad (4.1)$$

As this ratio becomes smaller, more lamps are required to cover the squared surface, the number of possible relative placements increase, and the problem becomes harder to solve for optimisation algorithms. The fitness of a candidate solution is directly related to the total area enlightened with a penalty for overlap, making the problem even harder to solve. A weight (W) tunes the relative importance of the two terms:

$$\begin{aligned} fitness &= \frac{area_enlightened}{total_area} - W \cdot \frac{area_overlap}{total_area} \\ &= \frac{area_enlightened - W \cdot area_overlap}{total_area} \end{aligned} \quad (4.2)$$

Best solutions maximise the illuminated area whilst minimising the number of lamps to cover the whole area. Tonda *et al* showed that traditional approaches based on genetic operators are competitive when the search space is relatively small, i.e., for Lamps problem size less than 10 (see Eq. 4.1) [136]. For more complex problems, cooperative-coevolution (or Parisian approach) outperformed the other algorithms they tested.

4.2.2 PET Tomography Reconstruction

Positron emission tomography is a nuclear medicine imaging technique. A radioactive tracer, which can be injected, inhaled or digested, is inserted into the

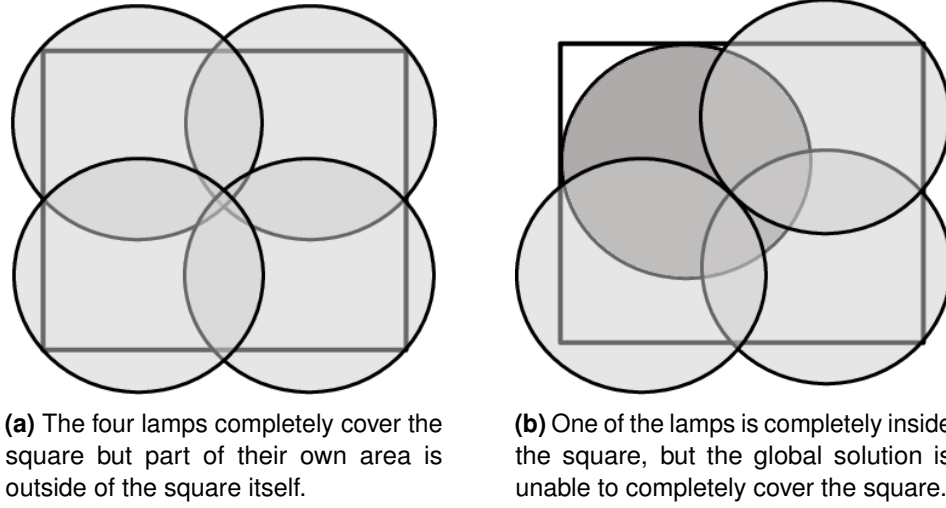


Figure 4.1: Arrangement of a set of 4 lamps to enlighten a square.

body of a patient. PET is a functional imaging technique that allows the visualisation and measurements of various physiological processes (metabolism, blood flow, tumor growth, bone fracture, chemical composition and absorption), depending on the tracer that is used. The radioactive concentration is proportional to the physiological process of interest.

The tracer's decomposition emits positrons. Gamma rays, resulting from the combination of positrons and neighbouring electrons, are then detected on a ring of detectors placed around the area to be observed (e.g., brain, stomach, full body, etc.). From this set of projections, a 3D image of the tracer distribution can then be computed as an inverse process using "tomography reconstruction". It is a complex inverse problem that is often ill-posed due to missing data and/or noise. The answer to the inverse problem is not unique, and in case of extreme noise level it may not even exist. The data acquisition in tomography can be modelled as:

$$Y = P[f] + \epsilon \quad (4.3)$$

where P is the system matrix or projection operator and ϵ corresponds to some Poisson noise. In this case the reconstruction corresponds to the inversion of the data acquisition model:

$$f = P^{-1}[Y] \quad (4.4)$$

Y is measured by the imaging system. It corresponds to projections at successive angles. In real applications, f is an unknown distribution of radioactive concentration.

Iterative tomography reconstruction aims to generate an estimate \hat{f} of f from the known projections Y . Fig. 4.2 shows the test case data that used.

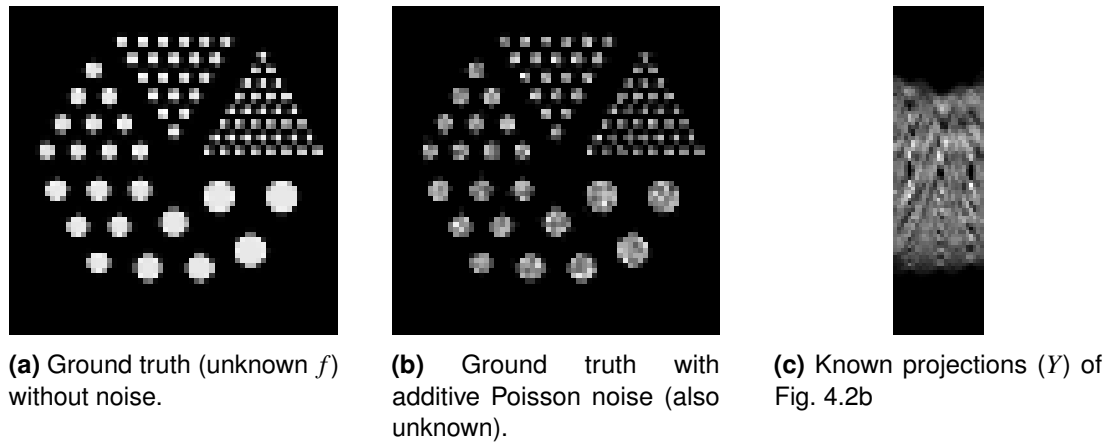


Figure 4.2: Test case data: Known projections (c) are provided by the imaging system. The tomography reconstruction provides an estimate of the unknown radioactive concentration (a).

Note that P^{-1} in reconstruction algorithms can account for noise, acquisition geometry, etc. Reconstruction methods used in clinical routine, namely Maximum-Likelihood Expectation-Maximization (MLEM) and Ordered Subset Expectation-Maximization (OSEM), are iterative correction algorithms [65], [124]. Iterative methods in tomographic reconstruction are relatively easy to model:

$$\hat{f} = \arg \min \mathcal{E}(Y, \hat{Y}) \quad (4.5)$$

where \hat{f} is an estimate of f that minimise an error metrics \mathcal{E} between Y and \hat{Y} . Note that a regularisation term can be introduced to prevent over-fitting and to smooth noise whilst preserving edges. In the Fly Algorithm \hat{f} corresponds the concentration of flies, and \hat{Y} to the image created when the flies are projected. Iterative methods are often implemented as follows:

1. The reconstruction starts using an initial estimate of the image (generally a constant image);
2. Projection data is computed from this image;
3. The estimated projections are compared with the measured projections;
4. Corrections are made to modify the estimated image; and

5. The algorithm iterates until convergence of the estimated and measured projection sets.

The Fly Algorithm strictly follows the iterative reconstruction paradigm:

1. The reconstruction starts with a population of flies (\hat{f}) uniformly distributed in the search space (here the flies are located within the space that corresponds to the imaging scanner);
2. Each fly is projected:
 - Each fly keeps track of its own projections;
 - The projections of all the flies are aggregated to produce the estimated projections \hat{Y} ;
3. The estimated projections are compared with the measured projections using the global fitness $\mathcal{E}(Y, \hat{Y})$, such as:

$$\mathcal{E}(Y, \hat{Y}) = \|Y - \hat{Y}\|_2^2 \quad (4.6)$$

4. The genetic operators are repetitively applied to minimise $\mathcal{E}(Y, \hat{Y})$ by adjusting the flies' position:
 - Selection of bad flies to kill (their projection data is removed from the population data);
 - Selection of good flies to create new flies that replace the killed ones;
 - The new flies are mutated;
 - The projections of the new flies are added to the population data;
5. The algorithm iterates until a stopping criterion is met, e.g., a maximum number of iterations is reached or convergence of the estimated and

measured projection sets (i.e., when \mathcal{E} stops decreasing between successive iterations).

In the case study below, we propose to reconstruct the synthetic data (Fig. 4.2c) based on the Derenzo phantom which is one of the most popular quality monitoring phantoms for nuclear medicine imaging. It is typically built by using two reservoirs of positron-emitting isotopes in a solution related by channels making up the Derenzo design [37] from Fig. 4.2b, which corresponds to the known projections of the ground truth image after Poisson noise was added to Fig. 4.2a. In this test case, there are 25 projections of 91 pixels. The algorithm aims to optimize the 2-D position of 1,840 emission points to minimise an error metric between Y and \hat{Y} . The fitness function used here is Sum of Squared Error (SSE) (Eq. 4.6), also known as ℓ^2 -norm, but other error metrics could be used.

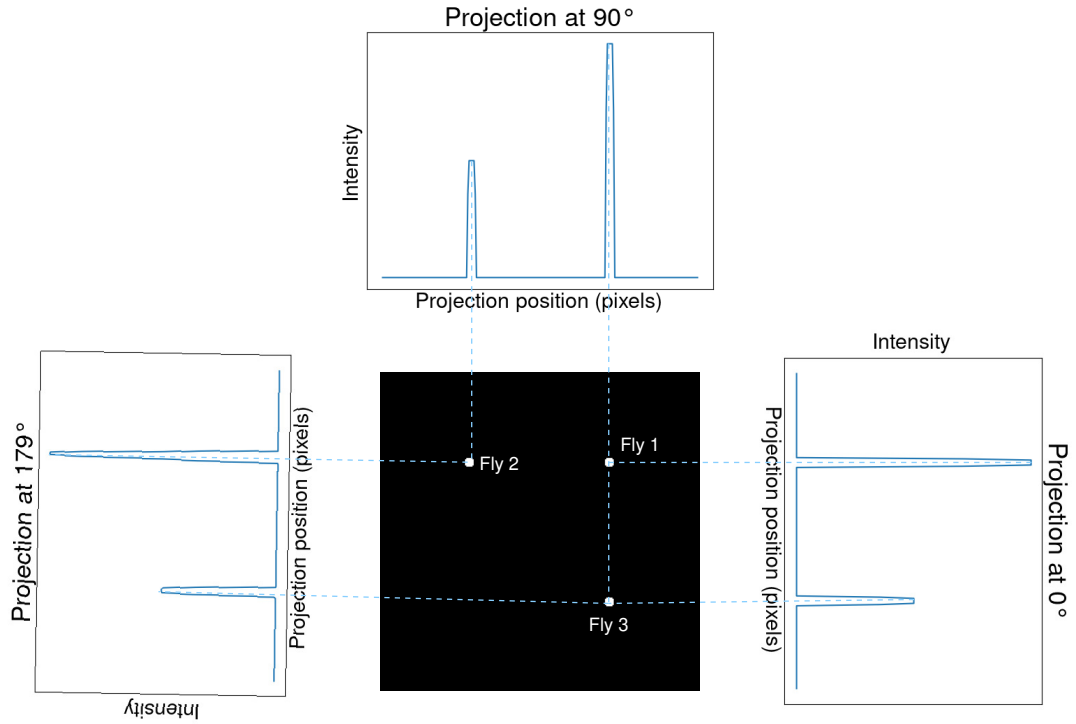
4.3 Fly algorithm implementation features

It has been experienced that cooperative co-evolution schemes often provide very efficient and scalable algorithms, but at the cost of a more complex design phase. Splitting a problem into a set of interdependent sub-problems, able to co-evolve within a single population, is not necessarily simple. This section details the specific features of the Fly Algorithm: A specific fitness function, a careful choice of appropriate operators and strategies.

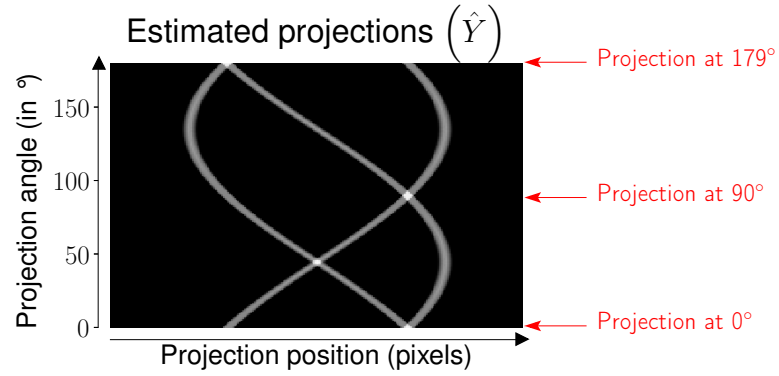
4.3.1 Fitness Metrics

In Parisian evolution two level of fitness functions can be used: (i) a local fitness for evaluating each individual, and (ii) a global fitness for the whole population, which is re-distributed on individual. Some problems may not need this specificity, in which case only the local fitness is used.

As in any other EA, the local fitness is used in the selection process to choose good individuals to breed. In the steady state implementation, it is also used to kill bad individuals. It tends to optimize this local fitness. In the same time, the algorithm is designed so that the population tends to improve its global fitness, which usually represents the searched solution. The choice of local and global fitness functions is



(a) Each fly of the population (\hat{f}) is projected at successive angles.



(b) Put together, these projections produce the estimated projections (\hat{Y}) used to compute the global fitness.

Figure 4.3: From a population of flies (\hat{f}) to estimated projections (\hat{Y}).

thus crucial and finding the right balance between local and global fitness may be delicate.

Bousquet et al. proposed a Fly Algorithm to reconstruct SPECT images¹ [28]. They initially used a simple local fitness function called “Bonus fitness”: If the projection of a fly is on a point of the detector that received some radiation, the fly gets a positive bonus (+1), else it receives a mauls (-1). Their experiments showed however that small objects of low intensity were not picked up by the algorithm and did not show up in the reconstructed images. To address this deficiency they proposed a “marginal” fitness, which relies on the leave-one-out cross-validation principle where the local fitness is computed from the global fitness. The marginal fitness $MF(i)$ of Fly i is the difference of the global fitness with and without i :

$$MF(i) = GF(pop \setminus \{i\}) - GF(pop) \quad \text{for a minimisation} \quad (4.7)$$

$$MF(i) = GF(pop) - GF(pop \setminus \{i\}) \quad \text{for a maximisation} \quad (4.8)$$

where $GF(pop)$ is the global fitness of the whole population and $GF(pop \setminus \{i\})$ is the global fitness of the population without Fly i .

One of the consequences is that if $MF(i)$ is negative, then Fly i is a “bad fly” that negatively contributes to the population. It can be killed. If it is positive, then it is a “good fly”. It can be selected to produce new children.

Another consequence is that the local fitness of an individual relies on its own contribution to the population as well as the contribution of other individuals. Each time an individual is killed and its contribution removed from the population, the global fitness changes. Therefore the local fitness of all the remaining individuals change too. This is also true when a new individual is created.

4.3.2 Steady state vs Generational

The example in Fig. 4.4 explains how the local fitness and global fitness depend on each other: Suppose there are two flies in the population (X and +). Both flies correspond to a point that emits photons (see Section 4.2.2).

¹SPECT uses a similar principle as PET but with gamma-emitting radioisotopes.

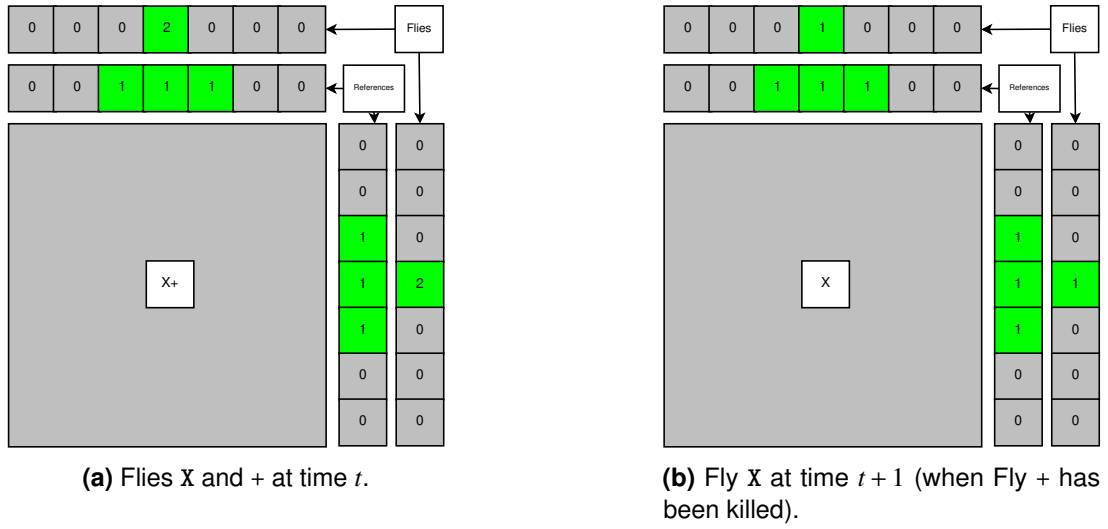


Figure 4.4: Example of projections created by the population of flies. For illustration purposes only two projections and two flies are considered.

In this example, they are purposely placed at the same location (on the same pixel, see Fig. 4.4-a). In this case they have the same fitness value. If consider the SSE (also known ℓ^2 -norm) as global fitness (see Eq. 4.6), it is:

$$GF(pop) = (0-1)^2 + (2-1)^2 + (0-1)^2 + (0-1)^2 + (2-1)^2 + (0-1)^2 = 6 \quad (4.9)$$

$$GF(pop \setminus \{X\}) = (0-1)^2 + (1-1)^2 + (0-1)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 = 4 \quad (4.10)$$

$GF(pop \setminus \{+\})$ is also 4. As a minimise the global fitness (Eq. 4.7), the marginal fitness of X becomes $MF(X) = 4 - 6 = -2$. $MF(+)$ is also -2. In a generational scheme, both are considered as bad flies and should be removed from the next generation. In a steady state context, if one of the flies, for instance $+$, is killed, the global fitness has to be re-evaluated, and the fitness of X also changes:

$$GF(pop) = (0-1)^2 + (1-1)^2 + (0-1)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 = 4 \quad (4.11)$$

$$GF(pop \setminus \{X\}) = (0-1)^2 + (0-1)^2 + (0-1)^2 + (0-1)^2 + (0-1)^2 + (0-1)^2 = 6 \quad (4.12)$$

$MF(X)$ is now equal to 2, which is a positive contribution.

This example highlights that the leave-one-out cross-validation principle used in the marginal fitness is better exploited by a steady state scheme. This assumption is confirmed by experiments of Section 4.4: Steady state and generational Fly Algorithms are compared on the two test cases presented in Section 4.2.

4.3.3 Threshold Selection

The selection operator chooses a random fly, which is duplicated and mutated to create a new fly. It is biased towards reproducing good flies and eliminating bad flies. Common selection operators in EA are tournament, roulette wheel, rank and elitism. Roulette wheel, rank selection and elitism have actually a relatively high maintenance cost: The local fitness value of all the individuals must be known at any time, which is not suitable for a steady state implementation.

Vidal et al. showed that it is possible to take advantage of the marginal fitness to provide a “threshold selection operator” [142]. The marginal fitness values quantify how much each fly improves or degrades the population’s performance (global fitness). They also showed that it provided an effective stopping criterion as it is able to characterise stagnation. The algorithm does not maintain the fitness value of the flies, which is compatible with a steady state implementation. It only computes the fitness of a fly when it is randomly picked up. When there are not enough bad flies left in the population, the threshold selection struggles to find flies to kill. As a consequence the optimisation either ends or triggers a new operator, such as the mitosis defined in Section 4.3.4.

In the experiments presented in Section 4.4, a simple, yet effective, early stopping criterion has been implemented as follows: The early stopping criterion is triggered if the number of good flies at iteration t is lower than at iteration $t - 1$ (i.e., the algorithm failed to increase the number of good flies).

4.3.4 Mitosis

Due to the distributed nature of the Fly Algorithm, it is straightforward to implement a varying population size. To get a high image resolution in the PET reconstruction problem, it is essential to use a large population of flies. This is because the

Fly population estimates a radioactive concentration. However it will affect the processing time, which is quasi proportional to the number of flies. To avoid this effect, the algorithm starts with a small population. Then, when stagnation is detected, a mitosis² process is initiated: Each fly is duplicated to double the size of the population, then each new fly is mutated. It has been experimentally shown that it increases the probability to find flies with positive fitness [141]. When the population stops improving the global fitness despite mitosis being triggered, the evolution stops as adding new flies is no longer beneficial.

Section 4.4 will also investigate how the population size (fixed vs. varying) affects the performance of the Fly Algorithm for the PET reconstruction problem.

4.3.5 Mutation

The Fly Algorithm maintains diversity in its populations thanks to the immigration and mutation operators. Usually in Evolutionary Algorithms, mutation operators are made more effective thanks to a variation of the mutation radius. It is large at the beginning of the evolution to ensure a good exploration of the search space. It then gradually decreases to concentrate the search. However, when a varying population size is used, the final population size might be unknown at the start of the algorithm. If the Fly Algorithm is implemented as a continuous steady state process, the number of generations might also be unknown. As a consequence, adaptive mutation operators with low maintenance cost are preferred. Ali Abbood and Vidal already showed that such dedicated mutation operators can significantly improve the performance of the Fly Algorithm in terms of both accuracy of the results and computational requirements [9]. In particular they proposed a “directed mutation” where each mutated fly keeps track of the performance of its ancestors. This operator is biased towards the position of good ancestors, and away from the position of bad ancestors. This research will rely on Gaussian mutation in the experiments presented in Section 4.4.

²Named by analogy with cell biology. Mitosis corresponds to a cell division process that gives rise to genetically identical cells.

4.3.6 Crossover

The crossover (also known as recombination, where two parents exchange part of their information to create new offspring) is not fundamental in the Fly Algorithm. It can actually be detrimental for some reconstruction problems such as stereovision and PET reconstruction. The crossover operator in the Fly Algorithm creates a new fly located in the space between the two selected parents. If consider two good flies on the surface of two different objects in stereovision or within two different radioactive regions in PET, creating a new fly in between is not justified. It is actually likely to lead to a bad fly. As a consequence, in the experiments of Section 4.4, if crossover is used in the RCGA, it is not used in FA.

4.4 Experimental analysis

4.4.1 Fair Comparison of the Algorithms

Comparing optimisation algorithms in a fair manner is not trivial. Benchmarking is often used for this purpose. Its aim is often to help researchers compare the performance of a (new) algorithm against established algorithms, or to help users select the most suitable optimisation method for their problems. This section below follow the ‘Best practices for comparing optimisation algorithms’ provided by Beiranvand et al [20]. For example they highlighted the need to consider more than one problem, and the importance of having easy problems and hard ones. Some problems must include known solutions. For tests that include starting points, new starting points can be randomly generated. It is also known that specific parameters required by the algorithms, such as stopping criteria and genetic operator probabilities, can dramatically influence the outcome of an algorithm. For this reason this collaboration will use evolutionary algorithms with various selection mechanisms. For algorithms with a fixed population or swarm size, several sizes are considered.

A fix number of fitness evaluations is often used as stopping criteria in benchmarking. This methodology is, however, not necessarily suited for the Parisian approach for two reasons. For example, in PET reconstruction, computing the fitness function is not the computational bottleneck. Adding a new emission point and computing its

projections at different angles is the real bottleneck. With PSO, RCGA, and CMA-ES, there is a single fitness function evaluation even for a large number of points. With FA, there is a fitness function evaluation per point. In this case, considering the number of emission points that have been created is more appropriate than the number of evaluations of the fitness function. This is why the following subsections used the number of lamps, or emitting points, created as timeline in graphs.

Also, looking only at the total number of fitness function evaluations does not take into account the maintenance cost of the various algorithms, e.g. in terms of time requirements or memory usage. For example see in section on *Quantitative analysis of the results on the Lamps problem*, Page 59. This is why we choose to restrain our benchmarks to algorithms belonging to similar classes of complexities. There exist of course various more complex versions of the different algorithms tested below, able to better deal with large size problems, for instance for CMA-ES [137] or Fly Algorithm [9], [84].

As all the algorithms considered here are iterative, the lack of improvement over the last N iterations is our main stopping criterion. To compare the algorithms' performance, the quality of the problem answer as well as the total number of lamps or emission points created will be accessed to allow a fairer comparison (as opposed to using the number of fitness evaluations in most benchmarking tools).

In benchmarking, the different algorithms will be ran on each problem and the benchmark will generate performance metrics for each algorithm for each problem. For each stochastic algorithm, for each problem, the test must be executed several times to gather statistically meaningful data. This is important as all the algorithms considered in our research are stochastic. In general, performance metrics are divided into three categories: i) efficiency (including scalability), ii) reliability, and iii) quality of algorithmic output. Due to the nature of the problems considered here, for the efficiency the collaboration rely on the number of lamps or emission points evaluated by the different algorithms to reach the stopping criterion. For the quality of algorithmic output, the average global fitness over N runs is used for both the Lamps and PET. In addition, the study relies on the zero-normalised cross-correlation (ZNCC) of the reconstructed image for PET. For the reliability, use

the standard deviation of the global fitness and ZNCC if applicable. The benchmark then provides a report on the results to ease the comparison. It can be in a tabular format or/and a visual format. This comparison used both.

For the visual comparison, it is possible to present actual images of the reconstructed data for both the Lamps and PET reconstruction problems. For each algorithm an odd number of runs were performed, 101 runs for the Lamps and 15 for PET. Each run for the same algorithm will provide a different image and a different final global fitness value. The performance measure is to select which image to present for each algorithm. Presenting the image reconstructed by the best run is unacceptable as cannot guaranty that the corresponding image is a representative outcome of the algorithm: It is possible the algorithm was just very 'lucky' for that run, in which case the image will correspond to an outlier. In statistical analysis, the mean (or average) is a popular metrics to show the central tendency of a statistical data set. It is possible to average the images of all the runs. However, such average images i) will be largely influenced by extreme (i.e., lucky and unlucky runs compared to the rest of runs), and ii) will not be the actual output of a run. Also, the distribution of final global fitness values for the runs corresponding to a given algorithm is unknown and cannot assume that it follows a normal distribution. In this case the median is more robust than the mean to derive the central tendency of a possibly skewed distribution. For these reasons, the images presented below correspond to the median result for each algorithm in terms of global fitness for the 101 runs for the Lamps or the 15 runs for PET.

4.4.2 Quantitative analysis of the results on the Lamps problem

The Lamps problem is a convenient benchmark aimed at better understanding a few features of the Fly Algorithm by comparison with more traditional approaches. Tonda et al, observed that it is not beneficial to use the cooperative co-evolution approach for problems with a search space of small size [136]. This collaboration aim to reproduce their observations. Here, two versions of the Fly Algorithm – steady state and generational FA, with tournament (duel here) and threshold selection – have been compared with RCGA with ranking, roulette wheel and tournament selection. The assumption is that in a steady state loop, the level of cooperation between

individuals is somehow much higher because the contribution of individuals toward the solution (i.e., global fitness) is constantly changing as individuals die and new ones are created. In other words, for every death or birth in the population, the fitness of all the individuals change.

Note that the RCGA and FA implementations use exactly the same Python class. The main differences are: (i) The Fly Algorithm includes the global fitness, the RCGA does not; (ii) The way the problem solution is extracted is different, as RCGA returns the individual with the highest fitness, while the Fly Algorithm returns all individuals in its final population. A varying population size is used in FA. Every time stagnation is detected, alternate slaughtering and mitosis. In the slaughtering step, bad flies are iteratively eliminated. In addition this comparison used PSO and CMA-ES.

The PSO, RCGA, CMA-ES and FA are all minimising Eq. 4.2 with $W = 1$. In PSO, this equation is the objective function used to evaluate particles. In RCGA and CMA-ES, Eq. 4.2 is the fitness used to evaluate individuals. In the Fly Algorithm, Eq. 4.2 is the global fitness used to evaluate the population as a whole. The marginal fitness is used in the Fly Algorithm to compute the fitness value of every individual (local fitness). Note that the real-coded genetic algorithm is tested with duel, ranking and roulette wheel selection. The Fly Algorithm is used as a generational algorithm and in steady state with duel and threshold selection. For PSO and RCGA, two sizes of swarm or population were used, 20 and 100. In total 15 different optimisation strategies were compared.

Seven problems of increasing complexity (3, 5, 10, 20, 100, 500 and 1,000) have been used to assess the performance and scalability of the different optimisation algorithms. Table 4.1 shows the search space dimension of the corresponding optimisation problems. Here consider search spaces from 27 D up to 9,000 D. Following Tonda et al's recommendations, there are $3 \times$ problem size lamps per

individual [136]. Each individual is made of 3 floating point numbers, x and y for the lamp position, and an on/off status so that:

$$0 \leq x \leq w - 1 \quad (4.13)$$

$$0 \leq y \leq h - 1 \quad (4.14)$$

$$0 \leq on/off \leq 1 \quad (4.15)$$

with w and h the room size along the x and y axis respectively. If $on/off \geq 0.5$ then the lamp is on, else it is off.

Table 4.1: Search space dimension for PSO and real-coded GA for each problem size. There are 3 values (x , y and on/off) per lamp. $3 \times$ problem size is the number of lamps per individual recommended in the original Lamps problem article [136].

Problem size	Dimension of search space
3	$3 \times 3 \times 3 = 27$
5	$3 \times 3 \times 5 = 45$
10	$3 \times 3 \times 10 = 90$
20	$3 \times 3 \times 20 = 180$
100	$3 \times 3 \times 100 = 900$
500	$3 \times 3 \times 500 = 4,500$
1,000	$3 \times 3 \times 1,000 = 9,000$

Each optimisation has been repeated 101 times, except for problem size 1000 where only 21 runs were performed due to computational constraints, to gather statistically meaningful results. In total $15 \times 6 \times 101 + 15 \times 1 \times 21 = 9,405$ optimisations were performed on the supercomputer environment.

The source code and Bash scripts are provided in a GitHub repository³. It enables readers to reproduce the results. In addition, it gives an example of how to turn a RCGA into a Fly Algorithm. Tables 4.2 to 4.5 give a summary of the main parameters of all the algorithms that have been evaluated in this section. Note that the stopping criterion is stagnation: No improvement over the last 5 generations for the Fly Algorithms, and 50 generations/iterations for the other algorithms. When using a traditional approach, a much larger number was empirically defined to avoid prematurely exits of the optimisation loop.

³<https://github.com/Shatha1978/Optimisation-algorithm-examples>

Table 4.2: Parameters used in all Lamps problems.

Lamp radius:	8 pixels
Stopping criterion:	500 generations max

Table 4.3: Additional parameters used in the Lamps problems with the Fly Algorithm.

Initial population size:	$3 \times \text{problem_size}$ flies
Crossover probability:	0%
Immigration probability:	30%
Gaussian mutation probability:	70%
Initial mutation factor:	16 pixels
Decrease of mutation factor:	0.016 pixel per generation
Additional stopping criterion:	No improvement over the last 5 generations
Solution extraction:	Whole population after iterative elimination of bad flies

When the tournament selection operator is used, the tournament size is equal to 2; when the threshold selection operator is used, the threshold is equal to 0.0.

Table 4.6 summarises the performance of all algorithms for various problem sizes: 3, 5, 10, 20, 100, 500, and 1,000. When the problem size is relatively small (3 and 5), all the approaches, except the generational Fly Algorithms, are successful in minimising the global fitness. Traditional approaches actually lead to better results than the Fly Algorithms. When the problem size reaches 10, the real-coded Genetic Algorithms start to fail, whereas PSO, CMA-ES and the steady state Fly Algorithms maintain their ability to deal with the global fitness. When the problem size is higher (20, 100, 500 and 1,000), real-coded Genetic Algorithms fail whereas the steady state Fly Algorithms remain successful. These results are consistent with Tonda's [136]. In this experiments, PSO is able to cope with problem sizes up to 10. CMA-ES is able to cope larger problem sizes, but at the expense of both extremely large memory requirements and computing times. In the supercomputing environment that used, jobs are killed after 3 days, which was not enough to complete the optimisation for CMA-ES with problem sizes 500 and 1,000. The amount of main memory used to compute the covariance matrix exceeded what is available on a desktop computer.

Table 4.4: Additional parameters used in the Lamps problems with a real-coded GA.

Population size:	20 & 100 individuals
Elitism probability:	9%
Crossover probability:	18%
Gaussian mutation probability:	73%
Additional stopping criterion:	No improvement over the last 50 generations
Solution extraction:	Best individual

When the tournament selection operator is used, the tournament size is equal to 2.
Other selection operators tested are: roulette wheel and rank selection.

Table 4.5: Additional parameters used in the Lamps problems with PSO.

Swarm size:	20 & 100 particles
Additional stopping criterion:	No improvement over the last 50 generations
Solution extraction:	Best particle

Table 4.6: Performance of the different algorithms on the Lamps problem with 7 different problem sizes (3, 5, 10, 20, 100, 500 and 1000). Numerical values correspond to average values and standard deviations over 101 runs, except for problem size 1000 where only 21 runs were performed due to computational constraints. The global fitness is given in %. This is a value that is maximised. Values for algorithms marked in **bold** are significantly better ($p < 0.01$) than the others for the same problem size.

Problem size	Evolution	Selection operator	Population/Swarm Size	Global fitness	Number of lamps	Lamps created before acceptance
3	PSO-20		20	91.85 % \pm 4.44	3.82 \pm 0.38	9.30e+03 \pm 4.00e+03
3	PSO-100		100	93.80 % \pm 1.58	3.94 \pm 0.24	3.29e+04 \pm 1.31e+04
3	RCGA-20 generational	roulette	20	76.82 % \pm 3.35	4.67 \pm 0.68	6.13e+03 \pm 5.35e+03
3	RCGA-20 generational	ranking	20	75.85 % \pm 2.89	4.86 \pm 0.72	7.00e+03 \pm 5.30e+03
3	RCGA-20 generational	tournament	20	76.22 % \pm 3.28	4.74 \pm 0.76	5.38e+03 \pm 4.14e+03
3	RCGA-100 generational	roulette	100	79.88 % \pm 2.55	4.71 \pm 0.68	3.47e+04 \pm 2.63e+04
3	RCGA-100 generational	ranking	100	79.41 % \pm 2.26	4.70 \pm 0.69	3.05e+04 \pm 2.71e+04
3	RCGA-100 generational	tournament	100	79.96 % \pm 2.16	4.56 \pm 0.70	2.72e+04 \pm 2.28e+04
3	CMAES		13	86.06 % \pm 8.51	3.82 \pm 0.54	9.72e+03 \pm 4.39e+03
3	FA generational	tournament	5	61.86 % \pm 7.10	3.47 \pm 0.83	6.41e+01 \pm 5.61e+01
3	FA generational	threshold	5	59.57 % \pm 6.82	3.55 \pm 1.01	6.58e+01 \pm 7.02e+01
3	FA steady state	tournament	9	71.10 % \pm 5.59	4.29 \pm 0.80	2.47e+02 \pm 2.43e+02
3	FA steady state	threshold	7	75.81 % \pm 4.22	4.53 \pm 0.72	1.32e+02 \pm 1.01e+02
5	PSO-20		20	84.46 % \pm 3.57	5.23 \pm 0.89	2.29e+04 \pm 7.84e+03
5	PSO-100		100	87.29 % \pm 3.24	5.52 \pm 0.81	1.02e+05 \pm 4.73e+04
5	RCGA-20 generational	roulette	20	70.17 % \pm 2.90	7.41 \pm 1.27	9.84e+03 \pm 8.28e+03
5	RCGA-20 generational	ranking	20	70.04 % \pm 2.81	7.50 \pm 1.11	1.16e+04 \pm 8.65e+03
5	RCGA-20 generational	tournament	20	69.87 % \pm 2.87	7.69 \pm 1.14	1.07e+04 \pm 8.14e+03
5	RCGA-100 generational	roulette	100	73.03 % \pm 2.40	7.61 \pm 1.03	5.53e+04 \pm 4.94e+04
5	RCGA-100 generational	ranking	100	73.85 % \pm 2.85	7.42 \pm 1.00	6.53e+04 \pm 5.15e+04
5	RCGA-100 generational	tournament	100	73.07 % \pm 2.32	7.57 \pm 0.85	5.05e+04 \pm 3.64e+04
5	CMAES		15	80.17 % \pm 4.90	5.84 \pm 0.72	3.01e+04 \pm 1.25e+04
5	FA generational	tournament	8	55.00 % \pm 5.84	5.38 \pm 1.38	1.08e+02 \pm 9.23e+01
5	FA generational	threshold	9	53.53 % \pm 5.97	5.50 \pm 1.26	9.82e+01 \pm 8.46e+01
5	FA steady state	tournament	14	68.11 % \pm 5.97	6.55 \pm 1.12	3.84e+02 \pm 3.34e+02
5	FA steady state	threshold	10	72.71 % \pm 3.51	7.01 \pm 1.06	2.49e+02 \pm 1.38e+02
10	PSO-20		20	82.09 % \pm 2.89	9.88 \pm 0.99	6.56e+04 \pm 1.68e+04
10	PSO-100		100	86.30 % \pm 2.08	10.20 \pm 0.92	3.28e+05 \pm 9.60e+04
10	RCGA-20 generational	roulette	20	61.00 % \pm 2.39	14.12 \pm 1.97	2.32e+04 \pm 1.73e+04
10	RCGA-20 generational	ranking	20	60.76 % \pm 2.53	14.11 \pm 2.17	2.05e+04 \pm 1.92e+04
10	RCGA-20 generational	tournament	20	61.61 % \pm 2.07	14.13 \pm 1.84	2.61e+04 \pm 1.90e+04
10	RCGA-100 generational	roulette	100	63.77 % \pm 1.87	14.20 \pm 2.02	9.74e+04 \pm 8.19e+04

Continuation of Table 4.6						
10	RCGA-100 generational	ranking	100	63.93 % \pm 1.65	14.23 \pm 1.83	1.22e+05 \pm 9.55e+04
10	RCGA-100 generational	tournament	100	64.15 % \pm 2.43	14.21 \pm 1.89	1.08e+05 \pm 9.25e+04
10	CMAES		17	78.15 % \pm 4.34	10.78 \pm 1.06	1.24e+05 \pm 3.47e+04
10	FA generational	tournament	16	47.23 % \pm 5.25	9.61 \pm 2.57	2.00e+02 \pm 1.38e+02
10	FA generational	threshold	16	47.16 % \pm 5.22	9.66 \pm 2.59	1.82e+02 \pm 1.30e+02
10	FA steady state	tournament	31	65.17 % \pm 4.99	11.44 \pm 1.65	9.98e+02 \pm 5.83e+02
10	FA steady state	threshold	18	69.11 % \pm 3.28	12.26 \pm 1.52	5.15e+02 \pm 3.02e+02
20	PSO-20		20	77.00 % \pm 2.83	18.63 \pm 1.70	1.72e+05 \pm 5.10e+04
20	PSO-100		100	82.47 % \pm 2.36	19.27 \pm 1.42	9.66e+05 \pm 2.65e+05
20	RCGA-20 generational	roulette	20	54.26 % \pm 1.75	28.37 \pm 2.80	3.85e+04 \pm 2.61e+04
20	RCGA-20 generational	ranking	20	54.15 % \pm 1.87	27.66 \pm 2.95	4.74e+04 \pm 3.68e+04
20	RCGA-20 generational	tournament	20	54.04 % \pm 1.73	27.66 \pm 3.12	4.33e+04 \pm 3.28e+04
20	RCGA-100 generational	roulette	100	56.67 % \pm 1.73	27.75 \pm 2.79	2.26e+05 \pm 1.73e+05
20	RCGA-100 generational	ranking	100	56.20 % \pm 1.34	27.28 \pm 3.31	2.26e+05 \pm 1.90e+05
20	RCGA-100 generational	tournament	100	56.09 % \pm 1.59	27.16 \pm 2.78	2.38e+05 \pm 2.13e+05
20	CMAES		19	77.90 % \pm 2.21	20.39 \pm 1.26	5.58e+05 \pm 1.17e+05
20	FA generational	tournament	33	43.29 % \pm 5.83	17.92 \pm 4.92	4.03e+02 \pm 2.04e+02
20	FA generational	threshold	34	42.65 % \pm 6.70	18.97 \pm 5.72	3.84e+02 \pm 2.22e+02
20	FA steady state	tournament	61	63.90 % \pm 3.52	20.74 \pm 2.14	2.10e+03 \pm 1.03e+03
20	FA steady state	threshold	32	66.39 % \pm 2.56	20.93 \pm 1.90	9.97e+02 \pm 5.36e+02
100	PSO-20		20	60.87 % \pm 2.04	93.25 \pm 8.20	1.46e+06 \pm 3.04e+05
100	PSO-100		100	68.31 % \pm 1.68	94.05 \pm 7.31	9.83e+06 \pm 2.42e+06
100	RCGA-20 generational	roulette	20	43.46 % \pm 0.93	141.56 \pm 8.02	2.34e+05 \pm 1.64e+05
100	RCGA-20 generational	ranking	20	43.14 % \pm 0.78	142.33 \pm 7.07	2.18e+05 \pm 1.72e+05
100	RCGA-20 generational	tournament	20	43.32 % \pm 0.72	141.95 \pm 7.57	2.39e+05 \pm 1.75e+05
100	RCGA-100 generational	roulette	100	44.23 % \pm 0.68	142.24 \pm 7.90	1.22e+06 \pm 8.89e+05
100	RCGA-100 generational	ranking	100	44.07 % \pm 0.84	140.33 \pm 7.80	1.03e+06 \pm 7.77e+05
100	RCGA-100 generational	tournament	100	44.33 % \pm 0.75	140.49 \pm 7.51	1.13e+06 \pm 7.91e+05
100	CMAES		24	77.86 % \pm 1.62	101.20 \pm 2.62	2.58e+07 \pm 2.86e+06
100	FA generational	tournament	171	36.82 % \pm 9.66	88.78 \pm 30.33	1.87e+03 \pm 8.38e+02
100	FA generational	threshold	169	36.04 % \pm 9.95	88.66 \pm 31.28	1.83e+03 \pm 9.32e+02
100	FA steady state	tournament	277	62.39 % \pm 5.61	90.04 \pm 8.96	9.36e+03 \pm 3.51e+03
100	FA steady state	threshold	144	64.47 % \pm 1.11	92.42 \pm 3.00	4.33e+03 \pm 1.66e+03
500	PSO-20		20	48.66 % \pm 1.94	463.77 \pm 32.22	1.15e+07 \pm 2.87e+06
500	PSO-100		100	54.00 % \pm 1.09	461.00 \pm 23.41	7.10e+07 \pm 7.81e+06
500	RCGA-20 generational	roulette	20	38.21 % \pm 0.33	731.64 \pm 17.22	1.18e+06 \pm 8.62e+05
500	RCGA-20 generational	ranking	20	38.10 % \pm 0.41	728.02 \pm 17.50	1.15e+06 \pm 8.43e+05
500	RCGA-20 generational	tournament	20	38.23 % \pm 0.37	725.99 \pm 18.29	1.16e+06 \pm 8.56e+05
500	RCGA-100 generational	roulette	100	38.71 % \pm 0.36	726.33 \pm 17.05	5.34e+06 \pm 3.98e+06
500	RCGA-100 generational	ranking	100	38.51 % \pm 0.37	727.73 \pm 17.77	6.15e+06 \pm 4.54e+06
500	RCGA-100 generational	tournament	100	38.76 % \pm 0.38	725.94 \pm 20.32	5.89e+06 \pm 4.06e+06
500	FA generational	tournament	885	34.06 % \pm 11.31	443.79 \pm 154.97	9.23e+03 \pm 4.70e+03
500	FA generational	threshold	854	33.72 % \pm 11.27	439.50 \pm 163.33	9.02e+03 \pm 4.77e+03
500	FA steady state	tournament	1369	62.63 % \pm 0.64	427.67 \pm 6.09	4.28e+04 \pm 8.00e+03
500	FA steady state	threshold	687	63.73 % \pm 0.56	433.84 \pm 5.82	2.22e+04 \pm 7.00e+03
1,000	PSO-20		20	45.72 % \pm 1.08	940.22 \pm 35.69	2.61e+07 \pm 4.08e+06
1,000	PSO-100		100	48.82 % \pm 2.43	917.14 \pm 51.12	1.43e+08 \pm 2.82e+07
1,000	RCGA-20 generational	roulette	20	36.98 % \pm 0.21	1456.10 \pm 27.00	2.80e+06 \pm 2.05e+06
1,000	RCGA-20 generational	ranking	20	37.08 % \pm 0.33	1468.83 \pm 22.15	2.92e+06 \pm 1.81e+06
1,000	RCGA-20 generational	tournament	20	36.99 % \pm 0.26	1466.95 \pm 27.56	1.68e+06 \pm 1.66e+06
1,000	RCGA-100 generational	roulette	100	37.35 % \pm 0.27	1462.33 \pm 32.25	1.46e+07 \pm 1.16e+07
1,000	RCGA-100 generational	ranking	100	37.27 % \pm 0.34	1465.56 \pm 31.45	1.22e+07 \pm 1.09e+07
1,000	RCGA-100 generational	tournament	100	37.46 % \pm 0.10	1464.30 \pm 26.76	1.03e+07 \pm 5.14e+06
1,000	FA generational	tournament	1,716	32.54 % \pm 13.22	857.10 \pm 355.15	1.70e+04 \pm 1.06e+04
1,000	FA generational	threshold	2,000	37.21 % \pm 0.44	1002.62 \pm 21.57	2.07e+04 \pm 1.02e+04
1,000	FA steady state	tournament	2,729	62.44 % \pm 0.45	840.79 \pm 9.25	8.54e+04 \pm 1.64e+04
1,000	FA steady state	threshold	1,346	63.74 % \pm 0.29	854.71 \pm 7.93	4.33e+04 \pm 1.31e+04

To visually compare the relative performance of the algorithms in terms of maximisation of the objective function, the collaborative plotted the average global

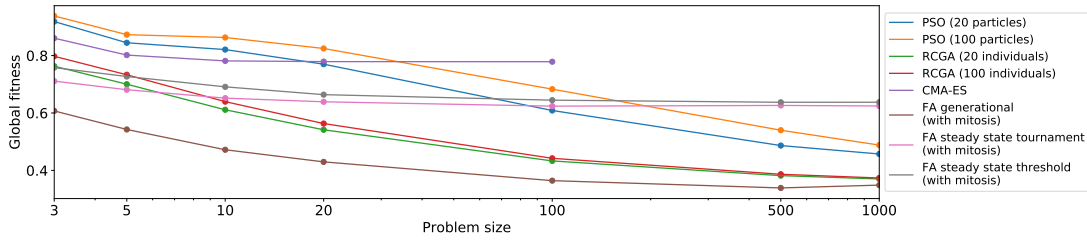


Figure 4.5: Mean global fitness for the Lamps problem for the different algorithms for different problem sizes. For problem sizes 500 and 1000, jobs were killed on the supercomputer as they did not complete within 3 days (the maximum limit allowed).

fitness for each algorithm for each problem size in Fig. 4.5. Table 4.6 shows that the choice of selection operator in the RCGA does not impact the performance of the algorithm. To improve the readability of the graph, the three corresponding curves (i.e., RCGA with duel, ranking, and roulette selection) replaced with a single one that averages their values. For the same reason the same is done with the generational Fly Algorithm. Fig. 4.5 clearly shows that traditional approaches are competitive until the problem size reaches 10, i.e., a 90-D search space (see Table 4.1). When the size of the search space increases further, the curves decline rapidly. The graph also shows that the steady state Fly Algorithm retains its ability to maximise the global fitness: The curves reach a plateau rather than decline. In addition, Figure 4.6 shows the lamp configurations for each algorithm for each problem size. The image for each optimisation algorithm corresponded to the median result in terms of global fitness over 101 runs, except for problem size 1,000 where only 21 runs were used. In this study used the median result rather than the best result for each algorithm to allow a fair comparison. It clearly shows that PSO and RCGA are successful in limiting the overlaps for Problem sizes 3 and 5. Problem size 10 is a transition where these algorithms are still competitive. For larger problem size, only the steady state Fly Algorithm is successful in both maximising the illumination (i.e, homogeneous gray images) and minimising overlaps whereas other algorithms fail on both counts (i.e, patchy images with both black and white areas). These results indicate that the steady state Fly Algorithm scales well when the problem size increases. The same is true with CMA-ES. However, memory usage and computing times may make its use prohibitive with the largest problem. In this respect the steady state Fly Algorithm remained competitive.

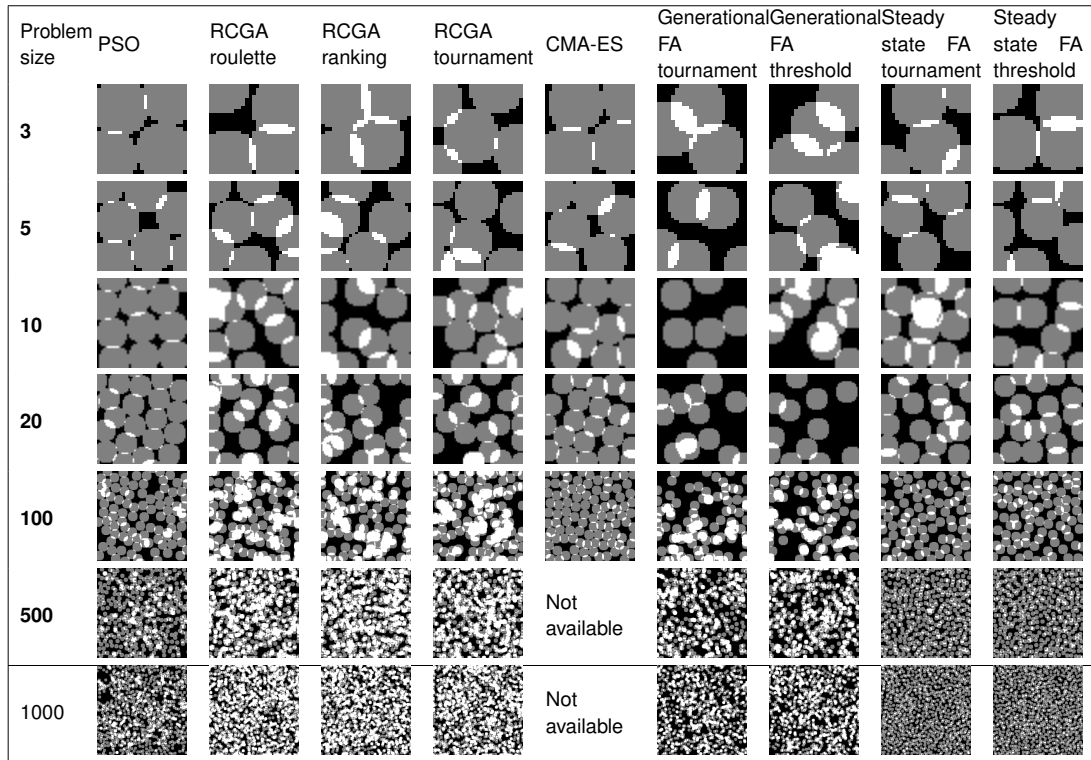


Figure 4.6: Simulated lamp configuration for each algorithm for each problem size. For PSO and RCGA, 100 particles or individuals were used. Good images should be gray and homogeneous. Black and white pixels correspond to a lack of illumination and to an overlap respectively. The image for each optimisation algorithm corresponds to the median result in terms of global fitness over 101 runs, except for problem size 1,000 where only 21 runs were used. We use the median result rather than the best result for each algorithm to allow a fair comparison.

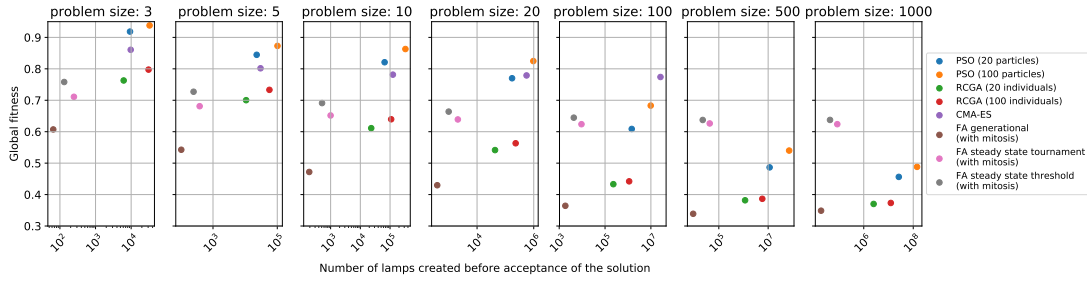


Figure 4.7: Efficiency and effectiveness comparison of all the algorithmic approaches for the different problem sizes. The horizontal axis of the scatter plots represents the number of lamps created before acceptance of the solution; the vertical axis the average global fitness. Numerical values were averaged over 101 runs. The best algorithms are in the top left corners of the plots (left for smallest computational requirements, i.e., efficiency, and top for highest global fitness values, i.e., effectiveness).

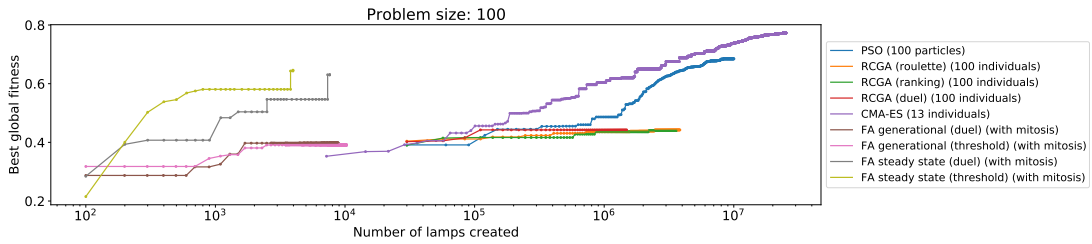


Figure 4.8: Fitness value for Problem size 100 as a function of the number of lamps created before acceptance of the solution, which is linearly related to computational requirements. For each optimisation algorithm, the curve corresponds to the median result in terms of final global fitness over 101 runs.

To study the trade-off between effectiveness (ability to find the best possible solution) and efficiency (minimal amount of time spent to find the best solution), scatter plots are used (see Fig. 4.7). The horizontal axis represents the number of lamps created before acceptance of the solution. The vertical axis represents the average global fitness over 101 runs. The most effective algorithms are shown at the top of the plots. The most efficient algorithms are shown at the left-hand side of the plots. Fig. 4.7 shows that PSO and CMA-ES are effective in finding solutions for problem sizes 3, 5, 10 and 20.

However, it is not very efficient compared to the steady state Fly Algorithms. When the problem size is greater than 20, steady state Fly Algorithms starts to outperform other algorithms. These results show that the cooperative co-evolution principles of the Parisian evolution is scalable.

As stagnation was used as a stopping criterion (5 successive iterations for the Fly Algorithms and 50 iterations for the traditional algorithms), it is important to assess

how the best global solution improves as more lamps are created by the algorithms. This is to make sure the assessment of the efficiency provided above was fair for all the algorithms. Fig. 4.8, compares the median solution in terms of global fitness over 101 runs when the problem size is 100. The plot clearly shows that the steady state Fly Algorithm is able to reach a global maximum extremely quickly compared to other algorithms. In fact, the global fitness is already high even before CMA-ES, PSO and RCGA start their optimisation loop because these algorithms need to create a large pool of solutions, each solution including 300 lamps ($3 \times \text{problem size}$). Hence again, it validates our assumption of the scalability of the steady state Fly Algorithm. This observation validate the hypothesis about the unsuitability, in this case, of the number of fitness function evaluations (or lamp creations in this case) as stopping criteria in benchmarking. Similar patterns were observed for other problem sizes.

A sharp spike can be seen at the end of the evolution for the steady state Fly Algorithm. This is because bad flies are iteratively killed from the population so that the population is made of good flies only. To further refine the results, it is possible to restart the evolution.

In the next section, it will use a more realistic test case, Positron emission tomography reconstruction, where the number of unknown values to estimate is much higher and cannot be easily predicted. For the Lamps problem with a problem size of 100, only about 90 lamps were needed, with 2 coordinates per lamp. That is about 180 unknown values to estimate. In PET reconstruction, thousands or millions of unknowns need to be estimated.

4.4.3 Quantitative analysis of the results on PET Reconstruction

The reconstruction aims at estimating the position of 1,840 emitting points. This number is proportional to the sum of pixel values in the phantom (see Fig. 4.2b), hence proportional to the radioactive concentration.

Once again, a traditional Real-coded genetic algorithm, a traditional PSO and a state-of-the-art CMA-ES are compared again with the steady state and generational

Fly Algorithms. In RCGA, CMA-ES and PSO each individual/particle is made of $1,840 \times 2$ real values (1,840 is the number of emitting points, and there are 2 coordinates, x and y , per point). In the Fly Algorithm individuals are made of 2 real values only. All the algorithms minimise Eq. 4.6.

- In the real-coded genetic algorithm and in CMA-ES, Eq. 4.6 is the fitness used to evaluate individuals. The final solution is a single individual, the best one. Note that the real-coded genetic algorithm is tested with duel, ranking and roulette wheel selection.
- In PSO, Eq. 4.6 is the objective function used to evaluate particles. The final solution is a single particle, the best one.
- In the Fly Algorithm, Eq. 4.6 is the global fitness used to evaluate the population as a whole. The marginal fitness is used in the Fly Algorithm to compute the fitness value of every individual (local fitness).

The experimental analysis aims at evaluating the added value of:

1. The Fly Algorithm to optimize the position of 1,840 individuals made of 2 genes over a real-coded genetic algorithm, PSO and CMA-ES to optimize N individuals/particles made of $2 \times 1,840$ genes;
2. The steady state implementation of the Fly Algorithm over its generational equivalent;
3. The Threshold selection operator over a traditional tournament (here duel) selection operator;
4. An early stopping criterion based on the threshold selection operator (see Section 4.3.3); and
5. An increasing population size using mitosis over a constant population size.

Table 4.7: Parameters used in all PET reconstructions.

Phantom size:	64 × 64 pixels
Projection width:	91 pixels
Number of projections:	25
Final number of emission points:	1,840
Image comparison metrics:	SSE (also called ℓ^2 -norm)
Stopping criterion:	No improvement over the last 5 generations

Table 4.8: Additional parameters used in the PET reconstructions with the Fly Algorithm.

Final population size:	1,840 flies
Crossover probability:	0%
Initial immigration probability:	25%
Decrease of mutation probability:	0.08% per generation
Initial Gaussian mutation probability:	75%
Increase of Gaussian mutation probability:	0.08% per generation
Initial mutation factor:	5 pixels
Decrease of mutation factor:	0.012 pixel per generation
Additional stopping criterion:	250 generations max
Solution extraction:	Whole population
Additional solution extraction:	Whole population after iterative slaughtering of bad flies

When the tournament selection operator is used, the tournament size is equal to 2;
when the threshold selection operator is used, the threshold is equal to 0.0.
When a fixed population size is used, the initial population size is equal to 1,840 flies;
when a variable population size (here mitosis) is used, it is equal to 115 flies.

Table 4.9: Additional parameters in the PET reconstructions with PSO.

Swarm size:	142 particles
Additional stopping criterion:	500 iterations max
Solution extraction:	Best particle

Again, for reproducibility, the source code and data used in our reconstructions are provided in a GitHub repository⁴. In addition, Tables 4.7 to 4.9 give a summary of the main parameters of all the algorithms that have been evaluated. In total 17 optimisation algorithm configurations were run (see Table 4.10). Each configuration was tested 15 times to gather statistically meaningful results. In total 255 evolutionary reconstructions were performed on the supercomputer environment. Note that the numbers of individuals/particles in the Real-coded genetic algorithm and PSO tests are calibrated based on the average number of newly created individuals in the best Fly Algorithm configuration identified in Table 4.10, that is to say $\text{round}(\sqrt{20,240}) = 142$ individuals and particles in the populations and swarms respectively. The main stopping criteria is no improvement over the last 5 generations (or equivalent in steady state). Additionally, a maximum number of generations is considered, 250 for the Fly Algorithm and 500 for PSO and RCGA. As the population size is automatically computed for CMA-ES, the maximum number of generations is calibrated so that as many emitting points as in PSO and RCGA are generated.

Global fitness

For each test, the final value of $\mathcal{E}(Y, \hat{Y})$ and the ZNCC between f and \hat{f} (see Eq. 4.16) are recorded [29]. The ZNCC is used to quantify how similar the reconstruction is from the ground truth. 1 corresponds to a perfect correlation between the two images, -1 an anti-correlation (also known as negative correlation), and 0 a non-correlation (the two images are unrelated to each other).

$$\text{ZNCC}(f, \hat{f}) = \frac{1}{M \times N} \sum_{i=0}^M \sum_{j=0}^N \frac{(f(i, j) - \bar{f}) \times (\hat{f}(i, j) - \bar{\hat{f}})}{\sigma_f \sigma_{\hat{f}}} \quad (4.16)$$

where M and N are the image width and height respectively, \bar{f} and $\bar{\hat{f}}$ are the average pixel value in Images f and \hat{f} respectively, and σ_f and $\sigma_{\hat{f}}$ are the standard deviation of the pixel values in images f and \hat{f} respectively.

For the Fly Algorithm, the number of individual created is recorded as it provides a good estimate of the computing time. The solution extraction is performed twice:

⁴<https://github.com/Shatha1978/Optimisation-algorithm-examples>

With both bad and good flies: The radioactive concentration is the concentration of flies at the end of the evolution process.

With good flies only: Bad flies are iteratively eliminated after the evolution process (after a bad fly is removed, the global fitness is updated, and the whole population is reprocess until there is no bad fly left in the population). The radioactive concentration is then the concentration of the remaining flies.

However, to provide a fair comparison, Table 4.10 displays the concentration of flies including bad flies for the Fly Algorithm. Good reconstruction methods should:

Be consistent: Always provide comparable results;

Minimise the global fitness: SSE between the estimated projections and the real projections;

Maximise the correlation of the reconstructed image with the ground truth;

Limit its computational requirements: Create as few emitting points as possible whilst still verifying the three characteristics above.

Table 4.10 shows that the traditional RCGA and PSO algorithms fail on these four fronts. CMA-ES is slightly better but still fails to deliver a good reconstruction. In fact RCGA, PSO and CMA-ES all fail to converge in the allocated computation time. The generational Fly Algorithm converges quickly enough, but fails to provide results of quality. The steady state Fly Algorithm, with threshold selection in particular, is the best option.

To visually analyse the results, Fig. 4.9 displays the scatter plot of the final values for the 15 runs of $\mathcal{E}(Y, \hat{Y})$ (global fitness) on the vertical axis against the total number of emitting points created by the optimisation algorithm (horizontal axis), which is proportional to the computational requirement. **Good algorithms will therefore concentrate at the bottom (small error) left (small computing time) corner of the plot.**

Table 4.10: Performance of the different optimisation algorithm configurations. Numerical values correspond to average values and standard deviations over 15 runs. The best algorithms for SSE projections and Reconstruction ZNCC are highlighted as follows: i) Values for algorithms marked in **bold** are significantly better ($p < 0.01$) than the others, and ii) Values with * show cases where the best performance is equally achieved by two or more algorithms (non separable, with $p > 0.05$). Note that the problem answer considered here for the Fly Algorithm is the concentration of flies, i.e., including bad flies.

Optimization Algorithm	Selection operator	Initial number of individuals/ particles	Genes per individual/ particle	Early termination	SSE projections (global fitness)	Reconstruction ZNCC (in %)	Number of emitting points created before acceptance of the solution
PSO	N/A	142	$2 \times 1,840$	N/A	$2.23\text{e}+08 \pm 7.59\text{e}+06$	33.7 ± 1.7	$1.31\text{e}+08 \pm 0.00\text{e}+00$
Real-coded GA	ranking	142	$2 \times 1,840$	N/A	$2.78\text{e}+08 \pm 8.28\text{e}+06$	17.3 ± 1.6	$1.30\text{e}+08 \pm 0.00\text{e}+00$
Real-coded GA	roulette	142	$2 \times 1,840$	N/A	$2.24\text{e}+08 \pm 7.55\text{e}+06$	21.8 ± 1.0	$1.30\text{e}+08 \pm 0.00\text{e}+00$
Real-coded GA	tournament	142	$2 \times 1,840$	N/A	$2.18\text{e}+08 \pm 7.79\text{e}+06$	21.4 ± 0.9	$1.30\text{e}+08 \pm 0.00\text{e}+00$
CMA-ES	N/A	28	$2 \times 1,840$	N/A	$1.37\text{e}+08 \pm 8.81\text{e}+06$	60.9 ± 1.8	$1.31\text{e}+08 \pm 0.00\text{e}+00$
Fly Algorithm (generational)	tournament	115	2	no	$2.79\text{e}+08 \pm 1.05\text{e}+07$	11.9 ± 2.0	$3.91\text{e}+04 \pm 7.09\text{e}+03$
Fly Algorithm (generational)	tournament	1,840	2	no	$2.87\text{e}+08 \pm 1.01\text{e}+07$	10.6 ± 1.4	$2.55\text{e}+04 \pm 7.51\text{e}+03$
Fly Algorithm (generational)	threshold	115	2	no	$3.47\text{e}+08 \pm 3.18\text{e}+07$	10.7 ± 1.5	$3.80\text{e}+04 \pm 8.30\text{e}+03$
Fly Algorithm (generational)	threshold	1,840	2	no	$3.21\text{e}+08 \pm 1.92\text{e}+07$	10.8 ± 1.6	$1.70\text{e}+04 \pm 1.63\text{e}+03$
Fly Algorithm (generational)	threshold	115	2	yes	$3.19\text{e}+08 \pm 2.53\text{e}+07$	11.1 ± 1.9	$1.29\text{e}+04 \pm 1.54\text{e}+03$
Fly Algorithm (generational)	threshold	1,840	2	yes	$2.87\text{e}+08 \pm 1.64\text{e}+07$	12.1 ± 1.8	$7.48\text{e}+03 \pm 4.75\text{e}+02$
Fly Algorithm (steady state)	tournament	115	2	no	$9.79\text{e}+07 \pm 4.96\text{e}+06$	52.7 ± 1.3	$5.49\text{e}+04 \pm 1.13\text{e}+04$
Fly Algorithm (steady state)	tournament	1,840	2	no	$1.04\text{e}+08 \pm 3.34\text{e}+06$	51.3 ± 1.1	$3.48\text{e}+04 \pm 7.76\text{e}+03$
Fly Algorithm (steady state)	threshold	115	2	no	$1.75\text{e}+07 \pm 8.66\text{e}+05$	$82.2^* \pm 0.8$	$6.26\text{e}+04 \pm 1.12\text{e}+04$
Fly Algorithm (steady state)	threshold	1,840	2	no	$1.87\text{e}+07 \pm 7.56\text{e}+05$	$81.8^* \pm 0.6$	$4.10\text{e}+04 \pm 8.00\text{e}+03$
Fly Algorithm (steady state)	threshold	115	2	yes	$1.85\text{e}+07 \pm 9.22\text{e}+05$	$81.8^* \pm 0.6$	$2.53\text{e}+04 \pm 3.10\text{e}+03$
Fly Algorithm (steady state)	threshold	1,840	2	yes	$1.90\text{e}+07 \pm 9.52\text{e}+05$	$82.2^* \pm 0.6$	$2.02\text{e}+04 \pm 2.60\text{e}+03$

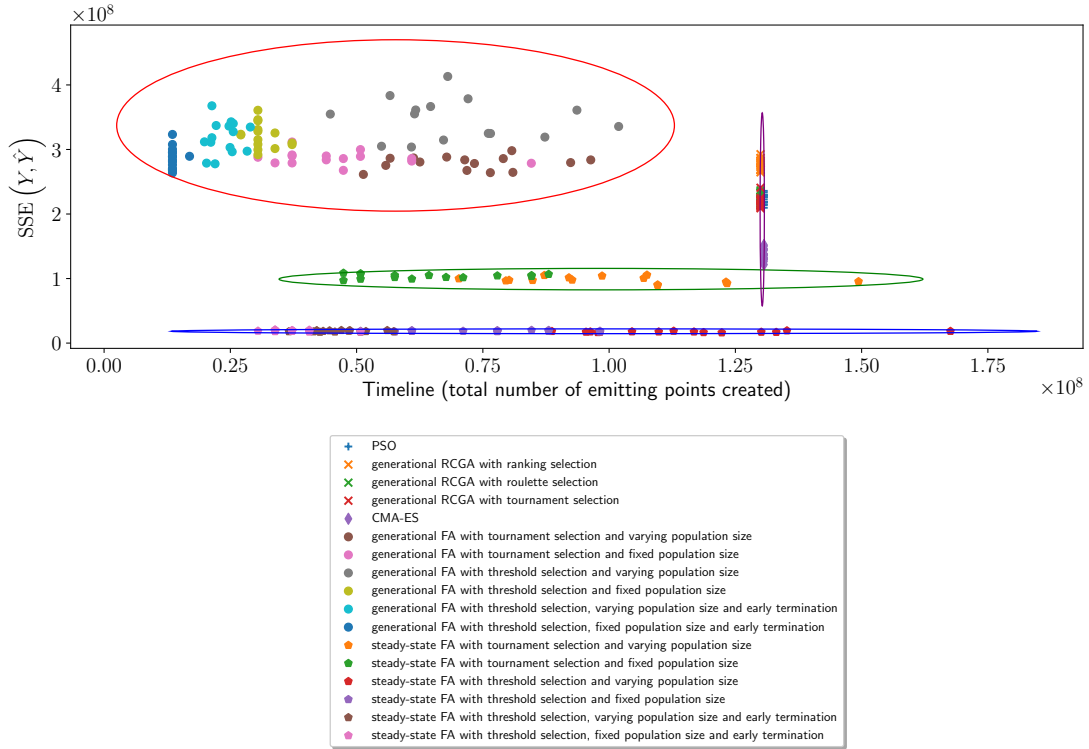


Figure 4.9: ℓ^2 -norm (global fitness) between Y (known projections) and \hat{Y} (projections estimated by the optimisation algorithm). Each marker represents a run for a given algorithm. There are 15 runs per algorithm. Best algorithms are located at the bottom-left corner of the plot. Note that the problem answer considered here for the Fly Algorithm is the concentration of flies, i.e., including bad flies.

It is possible to identify four clusters in the plot:

1. Traditional algorithms (PSO, RCGA and CMA-ES) (see the ellipse in purple);
2. Generational Fly Algorithm (ellipse in red);
3. Steady state Fly Algorithm with tournament selection (ellipse in green); and
4. Steady state Fly Algorithm with threshold selection (ellipse in blue).

Any steady state configuration provides better results than any of the generational configurations. Obviously conclude that the steady state implementation is better at minimising the error between Y and \hat{Y} . This can be explained by the fact that when using marginal fitness, the fitness of every fly is depending on all the population; selection based on marginal fitness only makes sense if the fly killed is immediately barred and the replacement fly re-incorporated into the population before the selection process goes on. Also, it is clear from the scatter plot that the steady state Fly Algorithm provides results of consistent quality (small spread in terms of SSE), which is not the case of the generational Fly Algorithm, CMA-ES, RCGA and PSO.

In addition, there is a clear difference can see between tournament and threshold selection operators in steady state when using a marginal fitness to select flies: Threshold selection is always better than tournament selection. This is because the tournament selection may choose i) to reproduce a bad fly, or ii) to kill a good fly. This behaviour is just impossible with the threshold selection (see Sections 4.3.2 and 4.3.3). The plot also shows that the early stopping criterion implemented using the threshold selection's internal data (see Section 4.3.3) considerably decreases computation time without affecting the quality of the output.

Correlation between the Ground Truth and the reconstruction

The quality of reconstruction is measured by the ZNCC, which quantifies the similarity between reconstructed and ground truth data. When ZNCC is high (100%), the reconstructed image is close to the noise-free ground truth (see Fig 4.2a). As mentioned in Section 4.4.1, for each algorithm displayed the median result over 15 runs rather than the best run. This is to allow a fair comparison.

Figures 4.10 and 4.12 show such results with images that correspond to the median result in terms of ℓ^2 -norm between Y and \hat{Y} (global fitness). It is clear from this visual inspection that the traditional optimisation methods are highly ineffective in reconstructing tomographic images: The reconstructed images are far from the ground truth.

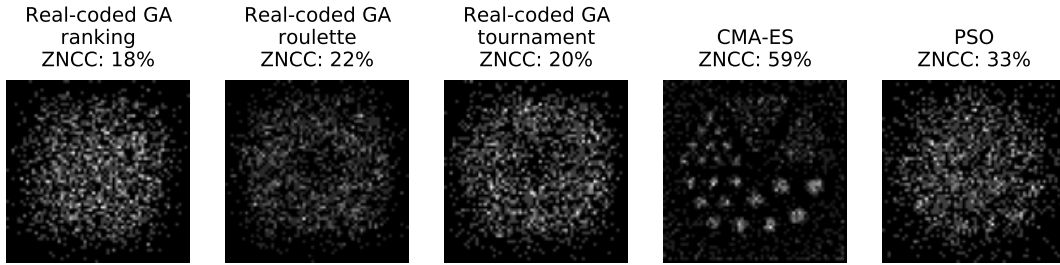


Figure 4.10: Reconstructed images using Particle Swarm Optimisation and real-coded genetic algorithm with three different selection operators. The image for each optimisation algorithm corresponds to the median result in terms of ℓ^2 -norm between Y and \hat{Y} (global fitness) over 15 runs.

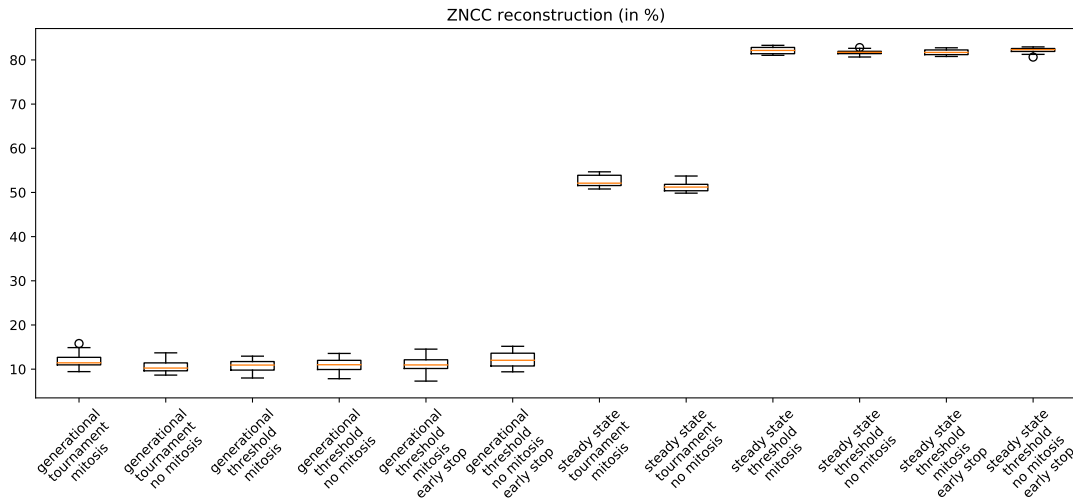


Figure 4.11: Box plots of ZNCC (or similarities) between f (noise-free ground truth) and \hat{f} (tomographic reconstruction corresponding to the concentration of flies) before elimination of bad flies, for 15 runs of Fly Algorithm with various operators setups.

The steady state and generational FAs are compared with various combinations of selection operators, initial population sizes (1,840 individuals for fixed population size, and 115 for varying population size), and an early stopping criteria (based on the threshold selection's internal data). Fig. 4.11 compares the final ZNCC of 15 runs of the Fly Algorithms. This figure highlights that:

1. The steady state implementation is always superior to its generational counterpart,

2. For steady state, threshold selection is always superior to the tournament selection, and
3. For steady state with threshold selection, there is no major difference in terms of ZNCC between fixed and varying population sizes strategies.

These conclusions are confirmed by a visual examination of the reconstructed images in Fig. 4.12. The top row shows the final results with the whole population (i.e., including both good and bad flies). The bottom row shows the final results after iterative slaughtering of bad flies: Images without bad flies look much clearer and ZNCCs are also much higher.

As a conclusion:

- Only the steady state Fly Algorithm was always successful in reconstructing PET images (PSO, RCGA, generational FA consistently failed);
- Threshold selection always outperformed the tournament selection in the steady state Fly Algorithm;
- Iterative slaughtering of bad flies at the end of the evolution helps to further improve the quality of the reconstruction.

Fixed vs Varying Population Size

The main difficulty with the fixed population size is to determine how many individuals are required. It may actually be impossible to achieve in a deterministic way. A trial and error approach can of course be used, but this is unrealistic for a clinical setting. This is why a varying population size is favoured, but the final number of point needs itself to be optimised. This section focuses on the steady state Fly Algorithm with threshold selection and early stopping criterion as it provided the best performance in both efficiency and effectiveness.

generational tournament		generational threshold		generational tournament		generational threshold		steady state tournament		steady state threshold		steady state tournament		steady state threshold	
no mitosis	mitosis	no mitosis	mitosis	no mitosis	mitosis	no mitosis	mitosis	no mitosis	mitosis	no mitosis	mitosis	no mitosis	mitosis	no mitosis	mitosis
ZNCC: 14%	ZNCC: 11%	ZNCC: 7%	ZNCC: 11%	ZNCC: 11%	ZNCC: 11%	ZNCC: 13%	ZNCC: 11%	ZNCC: 52%	ZNCC: 51%	ZNCC: 81%	ZNCC: 81%	ZNCC: 81%	ZNCC: 81%	ZNCC: 81%	ZNCC: 81%
ZNCC: 37%	ZNCC: 31%	ZNCC: 26%	ZNCC: 22%	ZNCC: 30%	ZNCC: 27%	ZNCC: 68%	ZNCC: 67%	ZNCC: 85%	ZNCC: 85%	ZNCC: 84%	ZNCC: 85%	ZNCC: 85%	ZNCC: 85%	ZNCC: 85%	ZNCC: 85%

Figure 4.12: Images reconstructed using the Fly Algorithm: with both good and bad flies (top row), and without bad flies (bottom row). The images correspond to median result in terms of ℓ^2 -norm between Y and \hat{Y} (global fitness) over 15 runs.

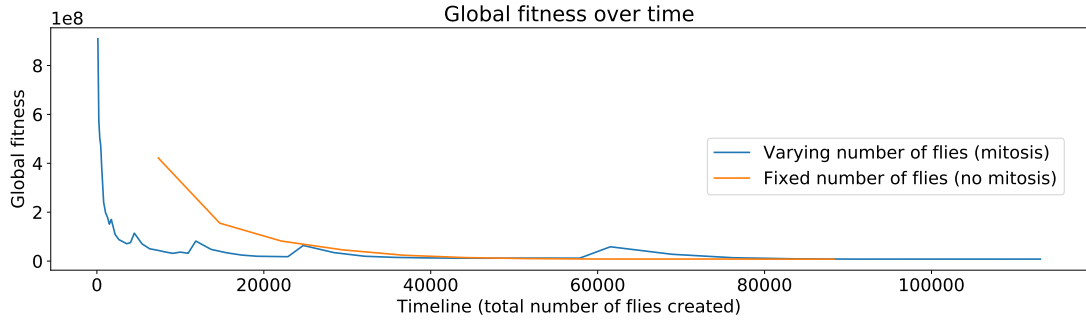


Figure 4.13: ℓ^2 -norm (global fitness) between Y (known projections) and \hat{Y} (projections simulated by the fly population) before elimination of bad flies. More flies have been used to further evaluate the mitosis.

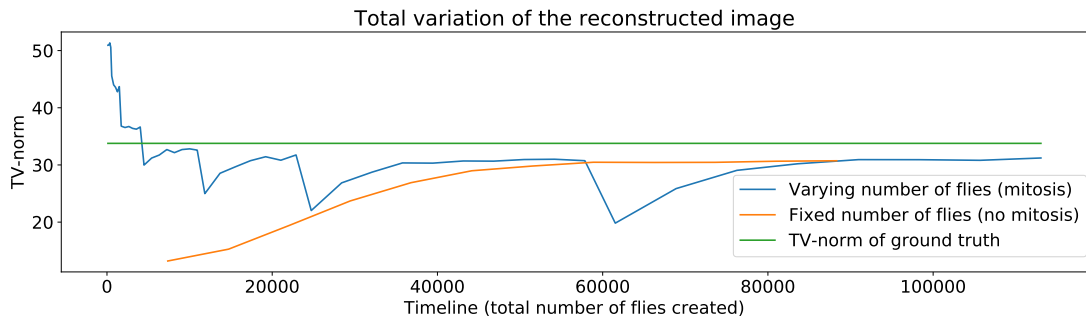


Figure 4.14: TV-norm of the reconstructed slices (\hat{f}) before elimination of bad flies. More flies have been used to further evaluate the mitosis.

When increased the final number of flies up to 7,360, and monitor the global fitness in function of time (i.e., number of flies created), see Fig. 4.13. Having multiple mitosis occurring in quick succession at the beginning of the evolutionary process helped to lower the global fitness very quickly. It was $9.09\text{E}+8$ at the beginning. It is $3.18\text{E}+07$ when the number of created flies is 10,925. By the time the varying population size scheme has created as many flies as there are in the fixed population size (i.e., 7,360 flies), its global fitness is already much lower ($1.70\text{E}+08$ for the varying population size compared to $4.22\text{E}+08$ for the fixed population size). The fixed population size scheme caught up with the varying population size scheme at about 45,000 created flies.

For each mitosis triggered after 10,925 new flies, the decrease of global fitness relative to the previous mitosis is not as significant as it previously was. The mitosis operator actually implements a multi-resolution tomography reconstruction. It can be used to let the algorithm determine the optimal number of flies. When the global fitness at two successive mitosis points is relatively similar, it means that adding

more flies does not help improving the global fitness anymore. It could stop adding new flies and extract the solution.

However, other image metrics calculated on \hat{f} rather than \hat{Y} may provide additional information. Gray et al. used visualisation to better understand the behaviour of the Fly Algorithm in PET reconstruction [58]. They showed in particular that a relatively good reconstruction is achieved quickly in terms of global fitness between Y and \hat{Y} but that other image metrics on \hat{f} , e.g., smoothness, may be more relevant to decide when to stop adding more flies and actually stop the reconstruction. This study will rely here on the smoothness of the reconstructed images as defined by the following total variation (TV) norm:

$$\text{TV}(\hat{f}) = \frac{1}{M \times N} \sum_{i=0}^M \sum_{j=0}^N |\hat{f}(i-1, j) - \hat{f}(i+1, j)| + |\hat{f}(i, j-1) - \hat{f}(i, j+1)| \quad (4.17)$$

where \hat{f} is the reconstructed image, M and N are its width and height in number of pixels respectively. Noisy images have a large TV-norm whilst homogeneous images have a TV-norm close to 0. This metrics is used to assess the quality of the reconstructed image. Fig. 4.14 shows how the TV-norm of the reconstructed slices evolves over time. It was chosen as the mitosis acts as a low-pass filter: When a mitosis occurs, every fly is duplicated and every new fly undergoes a mutation, hence blurring the reconstructed image. This is why the TV-norm decreases after each mitosis, then increases and finally reaches a plateau. With the varying population size, the reconstruction starts with a relatively small number of flies that are scattered in the image space: The image is quite noisy. The TV-norm is therefore high. With the fixed population size, the reconstruction starts with a much higher number of flies that are scattered in the image space: The image is more homogeneous. The TV-norm is therefore low. Again, the fixed population size takes a lot more time to stabilise compared to the varying population size. Also, if the global fitness for the fixed population size stopped significantly decreasing after 45,000 new flies, the TV-norm has not reached its final value. If the TV-norm of the reconstructed tomographic slice does not decrease, it means that adding more flies does not help improving the image quality anymore. For the varying population size, as both the global fitness and TV-norm do not significantly vary between their values at 22, 885 and 57,845 new flies, we could have stopped the evolution with

a population of 3,680 flies rather than 7,360, which was the strategy proposed by Gray et al to implement a more effective stopping criteria that limits the number of mitosis.

4.5 Conclusion

The study experimentally explored the scalability of FA compared to the classical *gbest* PSO strategy and RCGA, as well as one of today's most popular evolutionary algorithms for global optimisation, CMA-ES. For this purpose we used two problems, a toy problem (the Lamps) and an actual problem (Positron emission tomography reconstruction). PSO provided better results for the Lamps with problems of a low number of dimensions (up to 180-D). When the number of dimensions increased further, its performance collapsed rapidly. CMA-ES delivered more consistent results, but failed to converge in an acceptable time. For instances of CMA-ES dealing with problems with 4,500 and 9,000 dimensions, the optimisation was stopped before the end, as they reached the maximum duration allocated by the supercomputer. In other words, 3 full days of computations were not sufficient to complete the task. In addition to a high computational demand, CMA-ES also required a much larger amount of main memory because of the estimation of the distribution step, which increases considerably with the dimensionality of the problem. The Fly Algorithm was, however, successful in preserving its ability to find suitable solutions in an acceptable time, with no decrease of quality when the number of dimensions increased. In PET reconstruction, only the Fly Algorithm was successful in finding good solutions. PSO, RCGA and CMA-ES failed by a large margin. This study experiments highlighted that, while PSO, RCGA and CMA-ES are better at finding good solutions for problems with a low dimensionality, the Fly Algorithm scales better in higher dimensionality.

Its efficiency and remarkable scalability is a counterpart of its rather delicate design phase. The study provided some guidelines for future usage of the Fly Algorithm on other applications. Its scope of application is larger than expected. Initially designed for geometrical inverse problems (reconstruction from projections), it can be extended to other problems involving a large number of similar items. The Lamps benchmark is a typical example of such problems.

In conclusion, there is no free lunch in optimisation: Each algorithm is adapted to a range of problems. The study experiments demonstrated that the Fly Algorithm was better on large scale problems made of small interconnected subproblems of the same nature. As computer systems see a significant increase in both computational power and memory availability, trends in optimisation, as well as machine learning in general, show an increase in the scale of the problem they have to solve, e.g., big data is arguably its most visible phenomenon. In this respect, the importance of the optimisation algorithm scalability is likely to increase as well.

Chapter 5

Visualisation in evolutionary reconstruction of PET images

5.1 Introduction

This chapter is based on another collaborative study [58] that investigates the use of InfoVis and data exploration to understand some of the behaviours of an FA. It is related to the use of evolutionary computing in nuclear medicine, more particularly PET reconstruction. In particular, it wants to assess if the algorithm could have been stopped earlier to get a reasonable solution instead of waiting until the algorithm ends and using the final solution as the problem answer. During the evolutionary PET reconstruction, multiple time series are recorded hundreds of thousands of times. Comparing these time series by hand using typical scatterplots and line charts with no interactivity is not practically feasible:

- The order of magnitude of each time series is different. They would need to be independently normalised before plotting.
- Trial and error would be needed to choose the axis of interest because it is not necessarily straightforward to do so without a deep *a priori* understanding of the data.
- Adjusting the data range visualised in the scatterplots would also need to be performed with trial and error.
- Displaying selected images would need to be done manually.

The use of Parallel Coordinate Plot (PCP) is very popular to visualise high-dimensional geometry and analyse multivariate data [66], which is the type of data considered here. Interactivity using the brushing technique [89] makes it feasible to easily explore parts of this high-dimensional space and visually analyse this complex multivariate dataset.

The contribution of this study demonstrates how simple interactive visualisation techniques such as Parallel Coordinate Plots, scatterplot and image display can be used to analyse complex datasets generated using the temporal internal data of the evolutionary algorithm. Visual observations can then be used to improve the performance of the algorithm. The study illustrates how can be used to analyse the behaviour of the algorithm over time. This task would be extremely difficult without interactive visualisation. Well-designed user interaction and effective visualisation make it relatively easy. They can be used to analyse the performance of the population over time. When using stagnation as the stopping criterion, the final population is not necessarily the best one due to oscillations around the minimal fitness value.

A simple, but yet effective, visualisation framework has been purposely developed to explore data embedded in the log file and display the intermediate results based upon user interactions. It is used to assess the behaviour of the evolution process over time. The relationship between different properties of the reconstruction can also be examined. Using a case study, it helped to ascertain that allocating more computation time to the reconstruction algorithm did not lead to a significant improvement in accuracy. The research proposed an alternative early stopping criterion that looks at both the global fitness of the population and the smoothness of the reconstructed image over the last 500 iterations.

Section 5.2 describes the application to tomography reconstruction PET used in Fly Algorithm. The next Section 5.3 develops the information visualisation techniques that used to explore the internal data generated by successive iterations of the Fly Algorithm. Section 5.4 deals with PET reconstruction using the Fly algorithm. The purpose is to understand the behaviour of the algorithm, looking at three metrics, the global fitness (the minimisation of the Euclidean distance between the input

projections Y , and the projections \hat{Y} estimated with the Fly algorithm), the ZNCC between the ground truth f and the reconstructed PET image \hat{f} corresponding to the concentration of flies, and the TV norm of \hat{f} . It was exploited to extract which occurrence of \hat{f} that minimises both the Euclidean distance and the TV norm. It was then used to implement a much more effective stopping criteria that takes into account both the global fitness and the TV norm. It is followed by a conclusion that summarises the contributions.

5.2 Evolutionary reconstruction in PET

This is the approach adopted here to develop an integrated visualisation framework dedicated to evolutionary PET reconstruction algorithm. In typical evolutionary algorithms, the best individual of the final population is the solution of the optimisation problem. This algorithm relies on the Parisian approach where the solution to the problem is a group of individuals, e.g. the whole population or a subset of the population. The population size progressively increases to improve the resolution of the output image. The algorithm is launched with input parameters such as the initial number of individuals, the final number of individuals, the probability of operators, etc., the final solution is extracted at the end of the optimisation process then converted into a problem-specific answer.

In past implementations [3], [8], [139]–[142], the final result given by the last iteration is considered as the reconstructed image. But, when using stagnation as the stopping criterion, the final population is not necessarily the best one due to oscillations around the minimal fitness value. In such a case, past generations will have to be accessible. Also, reaching the targeted number of individuals may not be necessary if the reconstructed image stops improving. Offline analysis of intermediate results makes it possible to look at quality metrics other than the fitness value, e.g. smoothness of the reconstructed image. The initial goal is to extract the best possible solution in terms of fitness function and smoothness of the reconstructed image. The aim is to identify the smallest population that could be used as the solution instead of the final population to reduce the computing time as much as possible without compromising the quality of the reconstructed PET image.

The data acquisition in PET can be described as:

$$Y = Pf \quad (5.1)$$

where f is the radioactive concentration, which is unknown; Y the observations (known data as measured by the scanner); and P the system matrix or projection operator. The initial guess is a population \hat{f} of flies randomly located within the object space. Projections \hat{Y} are computed from the population and are compared with the data Y from the medical scanner. To that effect, an error metric between the two images is measured (see Eq. 5.2), this is the global fitness.

$$\|Y - \hat{Y}\|_2^2 = \sqrt{\sum_{y=0}^{y < h} \sum_{x=0}^{x < w} [Y(x, y) - \hat{Y}(x, y)]^2} \quad (5.2)$$

It is the numerical value that the optimisation algorithm will minimise. Errors are corrected using the application of genetic operators (mainly selection, mutation, new blood, and mitosis). The aim is to optimise the position of each fly so that the projection data of the whole population closely matches the one from the real radioactive concentration. The process is repeated until a stopping criterion is met. After convergence, the point cloud made by the flies is an estimate of the real radioactive concentration. The point cloud is then sampled to produce voxel data [8].

The study implemented a steady-state FA where, at each iteration, a bad fly is selected for death and replaced using a genetic operator (mutation or new blood). To evaluate the performance of a single individual (Fly i), it used the *marginal fitness* ($F_m(i)$) [28]. It relies on the global fitness with the leave-one-out cross-validation principle.

$$F_m(i) = \left\| Y - (\hat{Y} \setminus \{i\}) \right\|_2^2 - \|Y - \hat{Y}\|_2^2 \quad (5.3)$$

where $\hat{Y} \setminus \{i\}$ is the estimated projections without the photons simulated by Fly i . The idea behind the leave-one-out cross-validation is to assess the error metric twice: once with Fly i in the population, and once without it. By comparing the two values (the subtraction in Eq. 5.3) it can determine if having Fly i is beneficial or not for the population. If F_m is positive, the error is smaller when the fly is included: the fly has a positive impact on the population's performance. It is a good fly, i.e. a good candidate for reproduction. If F_m is negative, the error is larger when the fly is included: the fly has a negative impact on the population's performance. It is a bad fly, i.e. a good candidate for death. F_m is, therefore, a measure maximised by the algorithm. The study used this principle to define the 'threshold-selection' operator [141], [142]: to choose a fly to kill, find a fly with $F_m \leq 0$; and to choose a fly to reproduce, find a fly with $F_m > 0$. When the number of bad flies is low, the threshold-selection will struggle to find flies to kill. It provides a good stopping criterion.

The implementation start with a low number of flies (e.g. 25). When convergence is detected, each fly is duplicated to double the population size (see mitosis operator in [141], [142]). Each new fly is then mutated using mutation operators are available in [3]. Then the evolutionary process carries on until convergence is detected again. When the number of flies reaches a limit set by the user, and when convergence is detected, the reconstruction process ends. In the test cases presented below, it will use the new blood, basic mutation, and adaptive mutation operators. Also, note that the implementation is fully adaptive: operator probabilities are encoded by flies and undergo mutations.

Several stopping criteria can be used. Stagnation can be detected if the threshold selection operator struggles to find a bad fly several times in a row. The goal of the population is to minimise the global fitness as it is an error measurement. Stagnation can also be detected if the global fitness stops decreasing over a given number of iterations.

5.3 Data exploration

The Fly implementation produces a multivariate output, which may or may not be interrelated. In order to achieve the goal of inferring those relationships, our design choices are limited to multivariate relationship techniques. The common options in this situation are Heatmaps, Parallel Coordinate Plots, Scatterplots, Radar Charts, and Venn diagrams [111]. As Heatmaps is limited in the number of variables it can display [153], and Venn diagrams become difficult to read beyond three variables; these options must discount. Radar Charts are able to handle a larger number of variables, limited by the sweep angle between each axis. In theory without needing actual scale values, the chart could support 360 different axes; however, in practice the limit is substantially lower. An additional factor is that individuals (results in this case) are plotted over each other. Even with opacity effects it becomes more difficult to visually separate the individuals or extract patterns.

This requirements analysis leaves PCPs as the logical choice [159]. PCPs will suffer from over-plotting where results share equal/similar values. The tool allows the axes to be re-positioned and re-ordered to make any relationships more clear. This tool allows as many brushed ranges as there are axes, allowing users to precisely select items of interest, removing or fading unrelated data from the view. The system processes the CSV log files produced by the evolutionary process into JavaScript arrays. Time series were recorded over the evolution process. Each row of the file contains the data as follows: time stamp, population size, global fitness, corresponding images saved flag, common error/similarity metrics between Y and \hat{Y} as well as between f and \hat{f} (namely mean absolute error (MAE), mean squared error (MSE), Euclidean distance, root mean squared error (RMSE), ZNCC, SNR, peak signal-to-noise ratio (PSNR), structural similarity (SSIM), structural dissimilarity (DSSIM)), smoothness of \hat{Y} and \hat{f} using total variation, and internal states of the evolutionary algorithm (e.g. probability of the various genetic operators). The visualisation code selects user-specified columns (metrics) to make available as axes in the Parallel Coordinate Plot. Users are also offered the option to colour the lines produced according to another column (whether plotted or not). The values of that column are converted into a linear range between two user-specified colours. The tool uses the LAB colour space and HCL interpolation [60]. This results in

the perceived difference in plot colour being proportional to the Euclidean distance of the colouring metric, i.e. items close to each other in the metric space will be similarly coloured in the plot. An example of this version can be seen in Figure 5.1. A subsequent version added Brushing capability to the system. An example of this version can be seen in Figure 5.2.

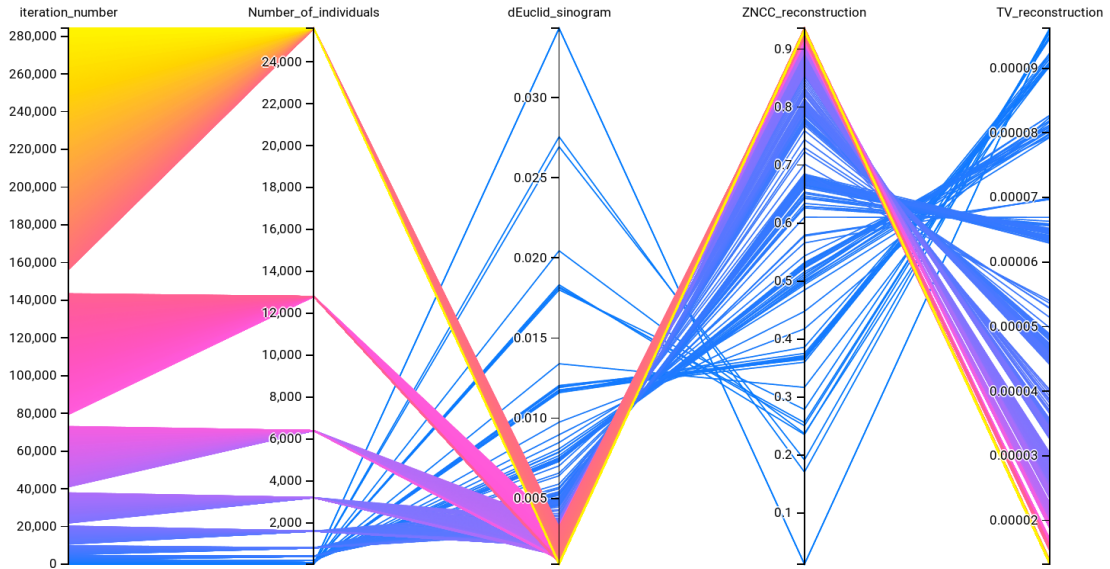


Figure 5.1: Initial prototype Parallel Coordinate Plot showing an initial run of the evolutionary process. Objects are coloured according to their iteration number, included as the first axis. This is a screen-shot captured from the tool itself, the labels are clearer in the tool.

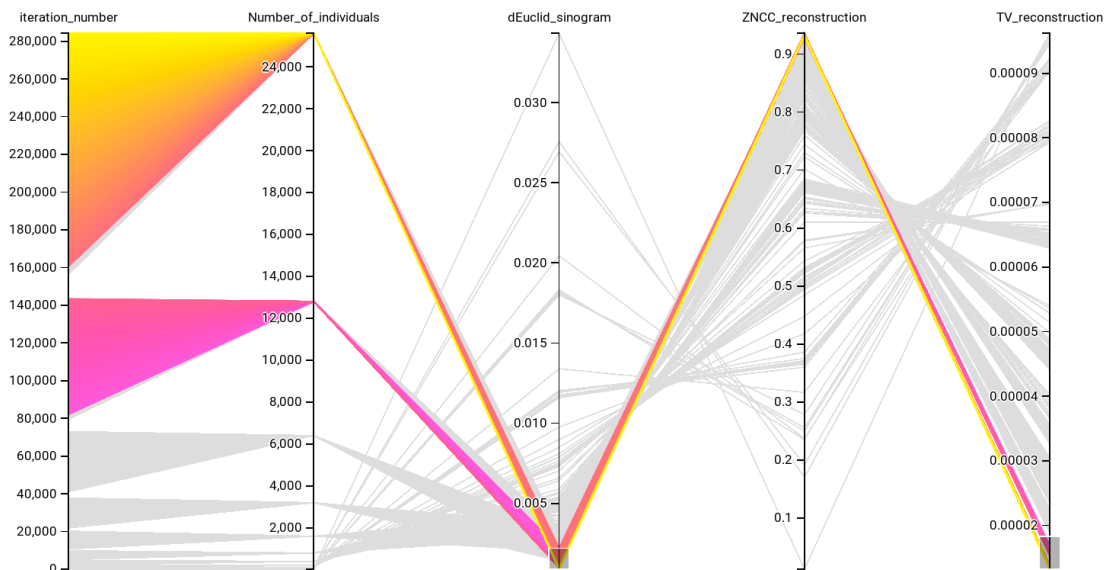


Figure 5.2: The same dataset as in Figure 5.1, with Brushing active on axes 3 and 5 (dEuclid_sinogram and TV_reconstruction). The ranges selected are shown by the tinted rectangle overlaid on those axes. This is a screen-shot captured from the tool itself, the labels are clearer in the tool.

When (exactly) two axes are brushed, the coordinated scatterplot is also drawn. The scatterplot uses the range of the two brushed axes and only plots selected data. The Y-axis represents the lowest numbered (leftmost) axis. The colouring

from the main plot is also maintained. As columns may not be in the desired order, the Parallel Coordinate Plot allows axes to be dragged left and right into the order required. The corresponding scatterplot to Figure 5.2 is shown in Figure 5.3.

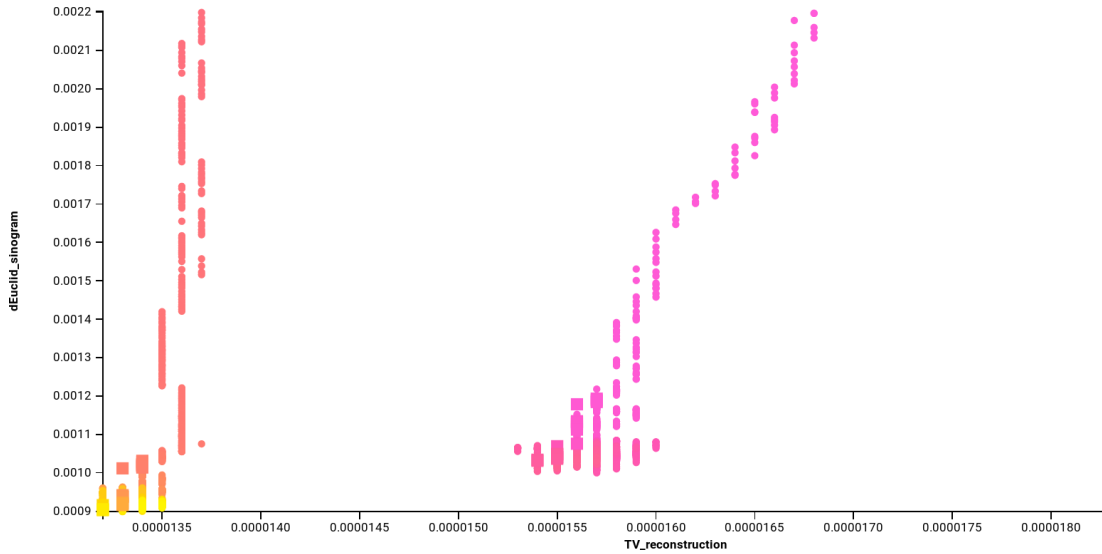


Figure 5.3: The coordinated scatterplot for Figure 5.2. Items are coloured as in the original figure, a smaller circle represents denotes that no image is available for that point and a larger square does. This is a screen-shot captured from the tool itself, the points and labels are clearer in the tool.

Combining these techniques, produced a powerful exploratory tool. It allows researchers and practitioners to gain insight into the performance of their algorithms in an intuitive visual way. The Parallel Coordinate Plots unveil potentially masked correlations and relationships within a dataset, and the scatterplot allows reasoning about efficiency and potential tuning options. A demonstration can be seen with a modern web-browser at <http://fly4pet.fpvidal.net/visualisation/>.

5.4 Results

The Fly Algorithm implementation consider the final result given by the end of the optimisation process. It provides a simple way to extract the answer of the optimisation problem, but it is not certain that it is the best answer that the evolutionary process provided. The initial goal of this study with the visualisation tool was to gain an understanding of what happens during the evolutionary process. A subsequent goal was to identify a ‘good’ reconstruction as quickly as possible. The ultimate goal was to develop stopping criterion dedicated to the Fly Algorithm in tomography reconstruction to automatically limit the reconstruction duration

to its minimum level whilst still preserving the accuracy of the results. A good reconstruction is when the error between the simulated projection data (\hat{Y}) and the input data (Y) is extremely low and when the noise levels in the reconstructed volume (\hat{f}) are low.

The initial assertion was that the huge amount of data generated by the evolutionary loop should not be discarded as it has the potential to actually be extremely useful to understand the reconstruction algorithm. The initial goals were to extract the best possible solution rather than simply take the final one and to determine if any other comparable solution could have been extracted earlier on to speed-up the reconstruction time. For this purpose, we performed a reconstruction using a controlled test case and analyse the results using this visualisation. The observation data (i.e. known data) is presented in Figure 5.4a. The ground-truth (i.e. unknown data) is presented in Figure 5.5a.

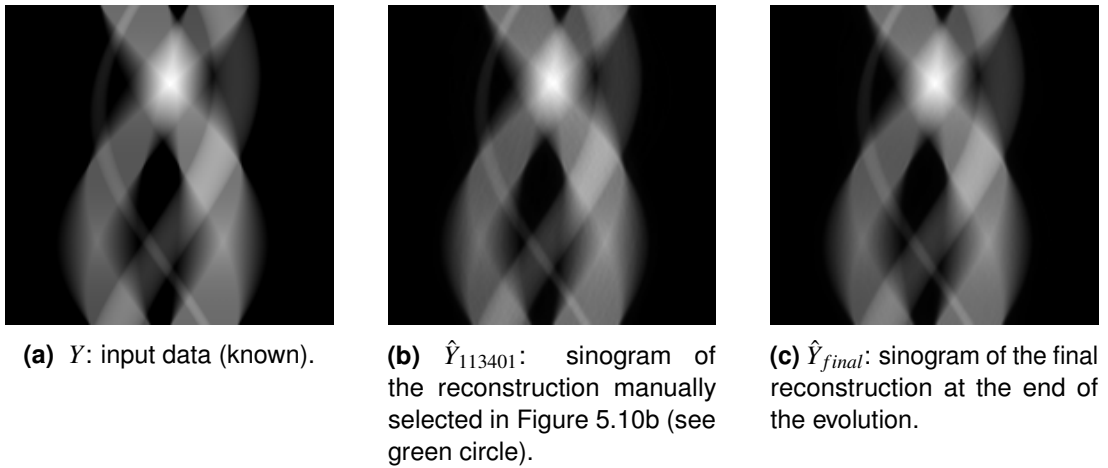


Figure 5.4: Sinograms.

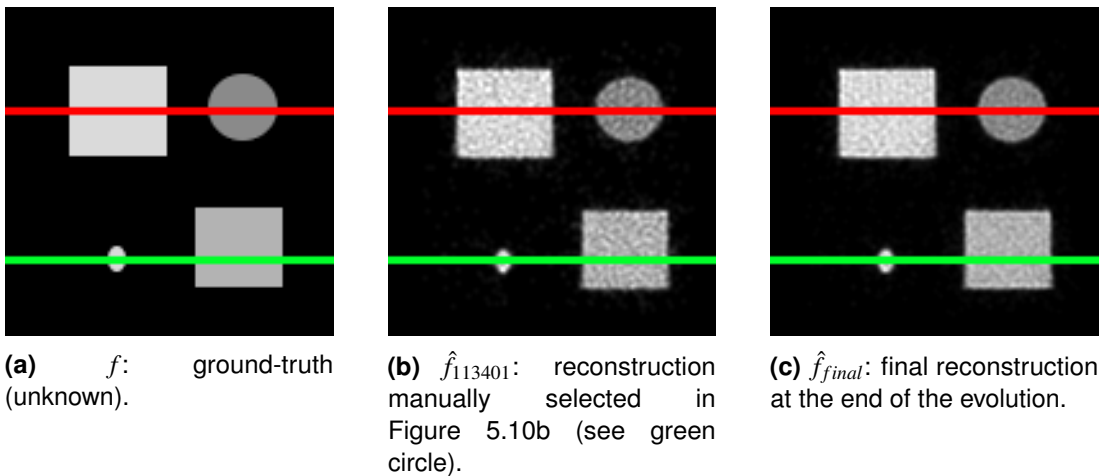


Figure 5.5: Reconstructed images.

Table 5.1: Initial parameters of the Evolutionary Algorithm.

Global fitness function:	ℓ^2 -norm
Initial population size:	25
Final population size:	25,600
Initial new blood probability:	1/3
Initial basic mutation probability:	1/3
Initial adaptive mutation probability:	1/3
Selection threshold struggle:	5 times in a row
Global fitness stagnation:	5 times in a row ($\varepsilon = 1\text{E-}2$)

Table 5.1 shows the initial parameters of the Fly Algorithm for this test case.

To measure the level of similarity between two images, whether they are f and \hat{f} or Y and \hat{Y} , used the ZNCC. The ZNCC is equal to 1 if the two images are perfectly correlated, 0 if they are totally uncorrelated, and -1 if they are perfectly anticorrelated (one is the negative of the other). The ZNCC is often expressed as a percentage. This image metric is very popular in image-processing and computer vision. To measure the smoothness level of the reconstructed image \hat{f} , used the TV (also known as TV-norm). Noisy images will have a higher TV-norm than smoother images. It can be used to compute a level of quality.

Defining what is the ‘best solution’ is not trivial:

- Traditionally it is the final population after convergence (#284,250).
- A good candidate solution is also the one that provides the lowest global fitness ($\|Y - \hat{Y}\|_2^2$). In this test case, it is #269,301.
- It can also be the population that gives the lowest discrete TV seminorm of the reconstructed image ($\|\hat{f}\|_{TV}$). Its first occurrence is #199,101 and its last is #274,101.
- Ideally, the best solution should provide the highest ZNCC with the ground-truth (f) ($\text{ZNCC}(f, \hat{f})$), but it cannot be assessed in the reconstruction as it is not available in real cases because f is unknown. However, it can be used with test cases to analyse the behaviour of this algorithm.

- Also, a good iteration should, if possible, have a relatively small cumulative computation time up to that iteration.

Table 5.2 summaries the performance of reconstruction at different iterations.

Table 5.2: Performance of reconstructions at different iterations. The best result for each metrics is in red, the second best in green, and the third best in blue. Iteration #113,401 has been selected by hand using the visualisation tool; #269,301 corresponds to the lowest global fitness; #199,101 corresponds to the first occurrence of the lowest TV; #274,101 corresponds to the last occurrence of the lowest TV; #284,250 corresponds to the last iteration.

Iteration #	113,401	199,101	269,301	274,101	284,250
Duration (in min)	7:32	13:15	17:55	18:14	18:55
# of individuals	12,800	25,600	25,600	25,600	25,600
$\ Y - \hat{Y}\ _2^2$	10.69E-4	9.49E-4	8.99E-4	9.03E-4	9.06E-4
$\text{ZNCC}(Y, \hat{Y})$	99.92%	99.94%	99.95%	99.95%	99.94%
$\ \hat{f}\ _{TV}$	1.55E-5	1.32E-5	1.33E-5	1.32E-5	1.34E-5
$\text{ZNCC}(f, \hat{f})$	93.20%	93.38%	93.19%	93.22%	93.16%

It presents the reconstruction cumulative computation time, the global fitness, the ZNCC between the input projections and simulated projections ($\text{ZNCC}(Y, \hat{Y})$), the TV of the reconstructed image and the ZNCC between the ground-truth and the reconstructed image. In terms of global fitness and TV, the results of the 4 iterations we selected seem to be equivalent. To assess if this is the case, we look at $\text{ZNCC}(f, \hat{f})$. The values are within 0.22%. In addition, a plot combining the global fitness and $\text{ZNCC}(f, \hat{f})$ is also presented (see Figure 5.6).

The figure shows barely any improvement, whether it is for the global fitness or ZNCC, when mitosis occurred (see pics in the graph). It can conclude that the results of the 4 iterations it selected are relatively equivalent. As it cannot distinguish between the results of the 4 iterations when looking at the global fitness and TV, it can consider the cumulative computation time (see Figure 5.7).

It can conclude that #199,101 is the ‘best’ iteration among #199,101, #269,301, #274,101, and #284,250 because its duration is the smallest: it lead to one of the best results in the smallest length of time. Spending an extra 6 minutes only marginally improved the results.

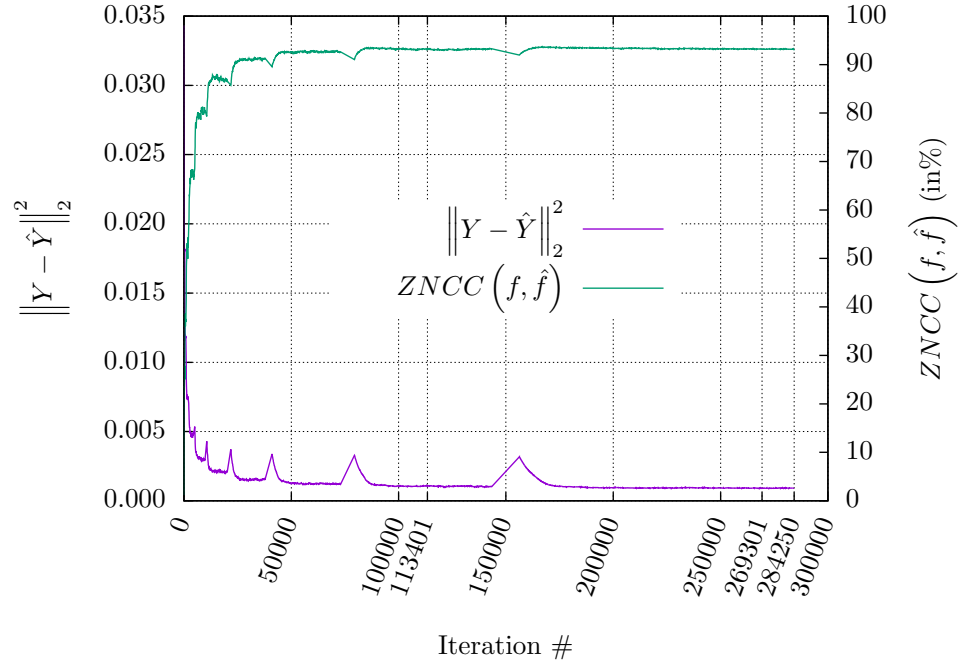


Figure 5.6: Combined evolution of the global fitness and the ZNCC between the ground-truth and the reconstructed image.

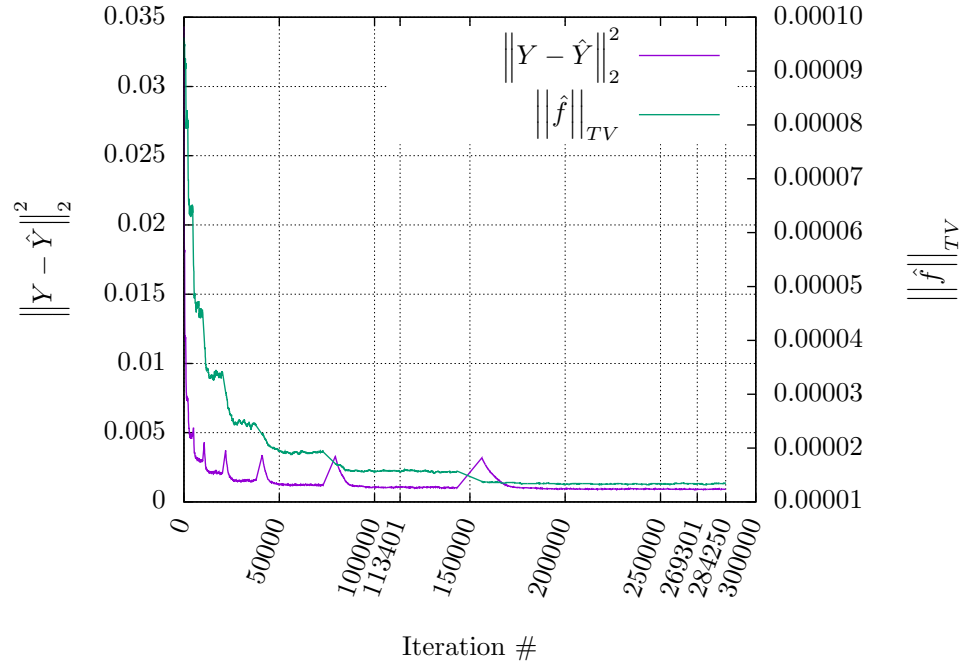


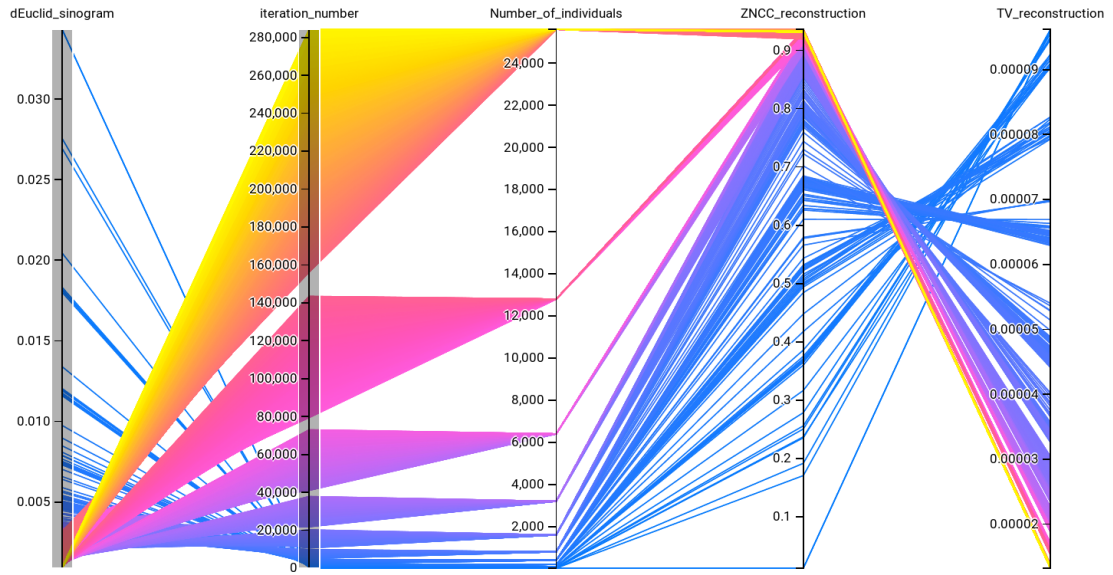
Figure 5.7: Combined evolution of the global fitness and the TV of the reconstructed image.

With the visualisation tool, it expected that the reconstruction time can be further reduced. The initial step is to look at how the global fitness evolves. The same dataset as Figure 5.1 is a plot with Brushing active on 'iteration_number' and 'dEuclid_sinogram' (see Figure 5.8a). Note that we swapped the axes in the figure to ensure that the number of iterations corresponds to the horizontal axis, and dEuclid_sinogram to the vertical axis in the scatterplot (Figure 5.8b). The study observed a rapid decrease of dEuclid_sinogram with upticks when mitosis occurs. It means that the global fitness approaches its minimum at an early stage of the reconstruction process. In other words, a relatively good reconstruction is achieved quickly in terms of data fidelity between \hat{Y} and Y but that other image metrics on \hat{f} may be more relevant to decide when to stop the reconstruction or to pick a better reconstructed volume.

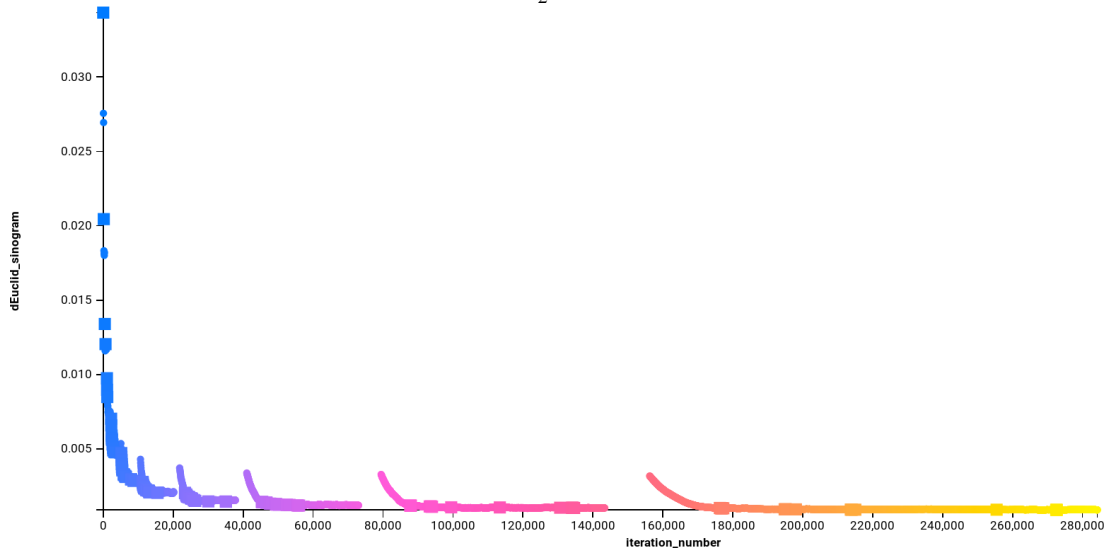
The same experiment is performed using TV_reconstruction rather than dEuclid_sinogram. Figure 5.9 shows that $\|\hat{f}\|_{TV}$ rapidly decreased, but a lot slower than dEuclid_sinogram. This is because the more mitosis happens, the more flies there are, resulting in less noise. However, it observed a plateau, beyond which the TV ceases to decrease significantly. This means that increasing the population size by mitosis would increase the duration without improving much the reconstruction. In this case, further investigation is needed as it indicates that the reconstruction process could have been stopped much earlier, with a lower number of flies.

It refined the brushed region to allow to zoom-in on a low $\|\hat{f}\|_{TV}$ (see Figure 5.10a). The goal is to ascertain that $\|Y - \hat{Y}\|_2^2$ is still low and minimise the duration. Ideally, the best possible candidate solution will be in the lower left corner of the scatterplot (see Figure 5.10b). The study selected a candidate solution that is a good compromise between time and noise levels (as more iterations do not reduce $\|\hat{f}\|_{TV}$ much) (see green circle in Figure 5.10b). It was obtained at 7:32 with 12,800 flies whereas the final candidate was obtained in 18:55 with 25,600 flies (see Table 5.2). It corresponds to a speedup of 2.5X.

To further validate the claim that #113,401 is a good candidate, comparable to the final one (#284,250), it now looks at image data (see Figures 5.4, 5.5 and 5.11). The difference, in terms of ZNCC, for \hat{Y} between #113,401 and #284,250 is 0.02% (see

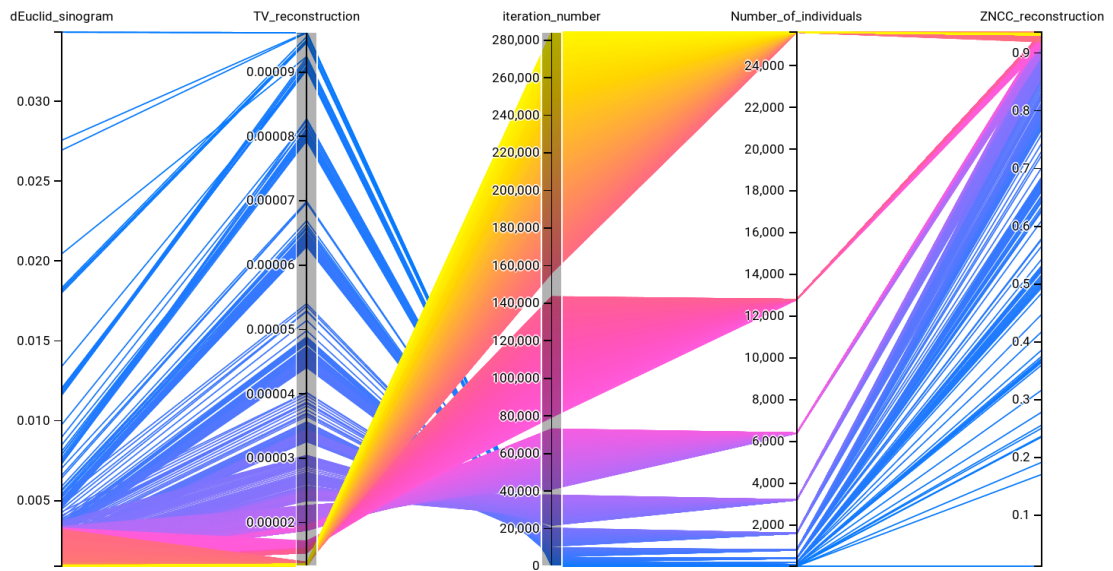


(a) Brushing on $\|Y - \hat{Y}\|_2^2$ and number of iterations.

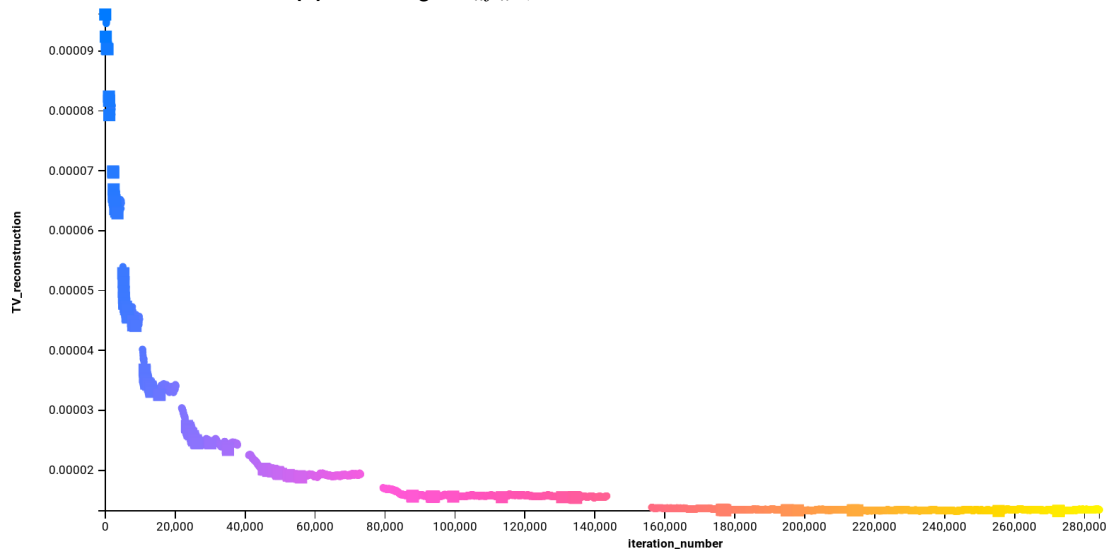


(b) Corresponding scatterplot.

Figure 5.8: Evolution of the global fitness. For each iteration plot the value of $\|Y - \hat{Y}\|_2^2$.

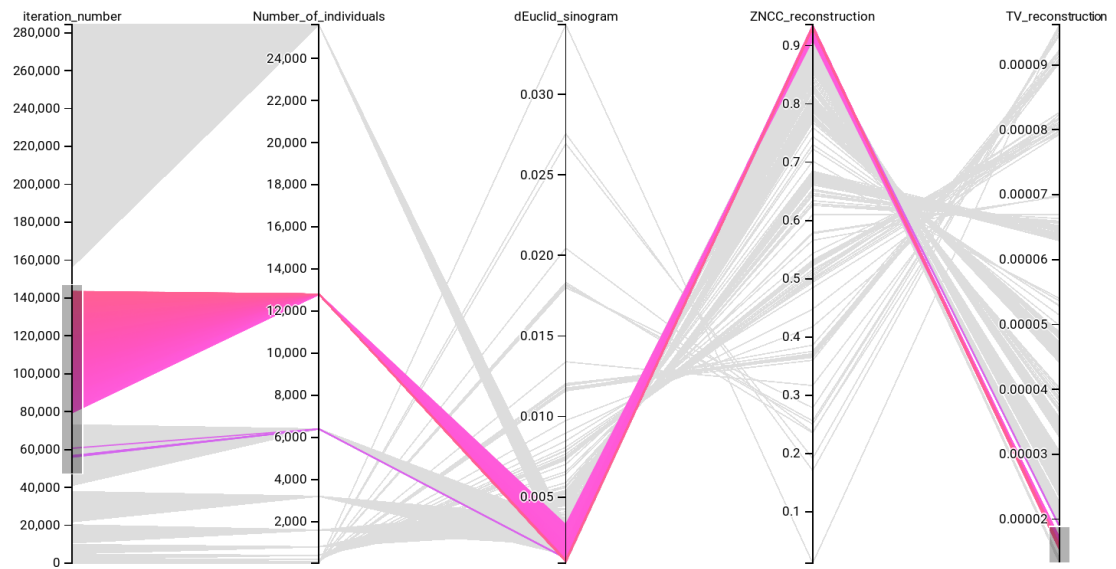


(a) Brushing on $\|\hat{f}\|_{TV}$ and number of iterations.

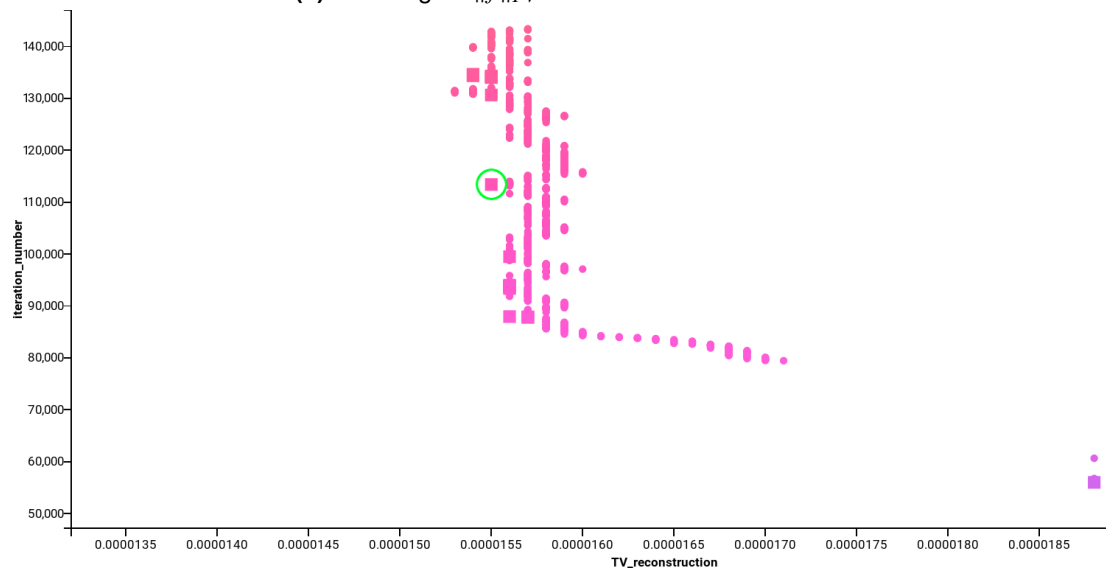


(b) Corresponding scatterplot.

Figure 5.9: Evolution of the total variation seminorm.



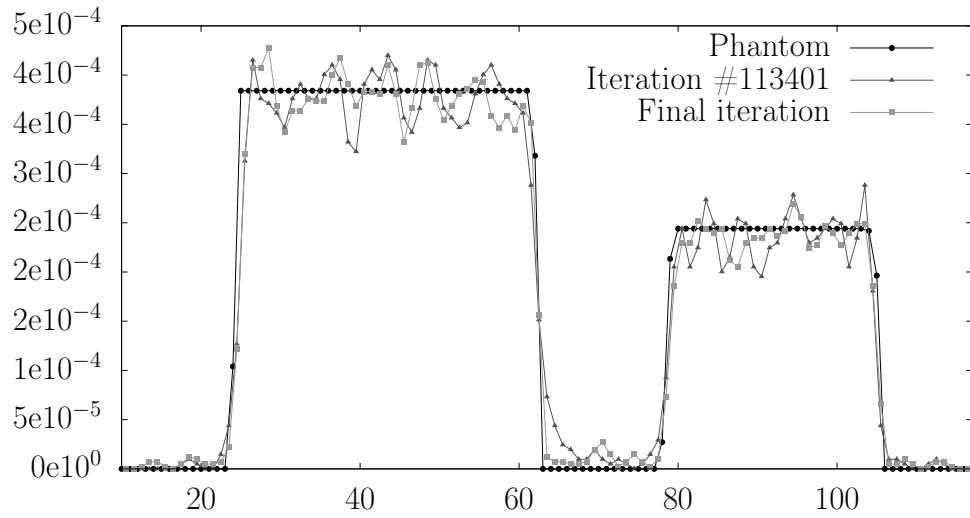
(a) Brushing on $\|\hat{f}\|_{TV}$ and number of iterations.



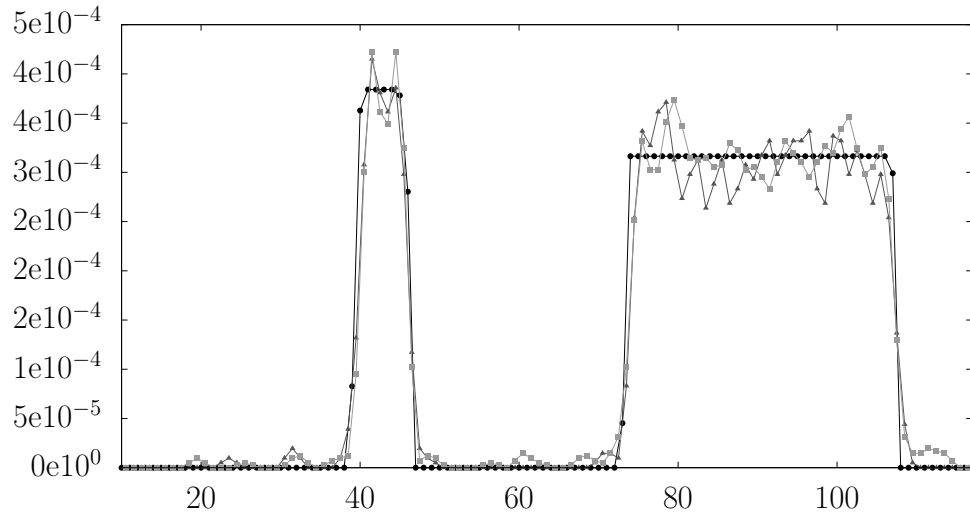
(b) Corresponding scatterplot. A 'good' candidate solution is circled in green.

Figure 5.10: Manual selection of a good candidate solution based total variation seminorm and duration.

Figure 5.4 for the image data). This is negligible. The ZNCC of \hat{f} is actually slightly smaller (by 0.04%) for #113,401 than #284,250 (see Figure 5.5 for the image data). To visually assess the noise levels in #113,401 and #284,250, intensity profiles of interest are extracted. An intensity profile plots the intensity values along a line segment between two points of an image. They are shown in Figure 5.11. The noise levels in #113,401 and #284,250 are very similar. Therefore, conclude that the extra 11:23, after iteration #113,401, did not significantly improve the reconstruction.



(a) Intensity profiles corresponding to the red lines in Figure 5.5.



(b) Intensity profiles corresponding to the green lines in Figure 5.5.

Figure 5.11: Intensity profiles in the ground-truth and in the reconstructions presented in Figure 5.5.

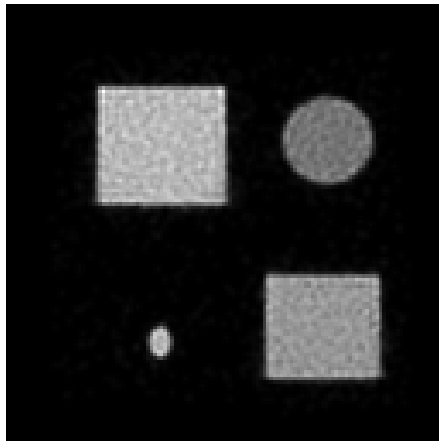
The study further exploited these results by introducing a new stopping criterion that looks at both the global fitness and TV. The global fitness is analysed over the last 500 iterations. Using simple linear regression, the fitness values are reduced to a

Table 5.3: Performance comparison between the algorithm with and without the new stopping criterion using 3 test cases. Each reconstruction has been performed 10 times.

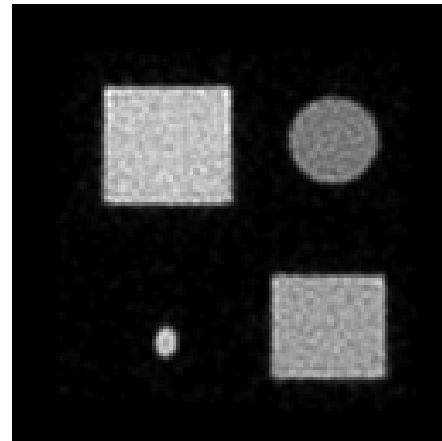
		Phantom 1	Phantom 2	Phantom 3
Without	# of iterations	282590±989	276320±18230	282820±987
	$\ Y - \hat{Y}\ _2^2$	1.09E-03± 5.69E-05	1.46E-03±6.83E-05	8.94E-04±6.88E-05
	$\ \hat{f}\ _{TV}$	1.34E-05±1.43E-07	1.07E-05±1.45E-07	1.60E-05±1.62E-07
	ZNCC(f, \hat{f})	93.23%±0.05%	94.34%±0.04%	92.59%±0.04%
With	# of iterations	89190±5880	90370±14331	117450±58705
	$\ Y - \hat{Y}\ _2^2$	1.32E-03±5.95E-05	1.81E-03±1.30E-04	1.04E-03±9.98E-05
	$\ \hat{f}\ _{TV}$	1.38E-05±1.11E-07	1.13E-05±2.06E-07	1.65E-05±1.79E-07
	ZNCC(f, \hat{f})	93.47%±0.05%	94.66%±0.07%	92.68%±0.06%

single line, and the equation for it is extracted. When the slope is close to zero, the line is almost horizontal. It means that the global fitness has not changed much over the last 500 iterations. This process is repeated using the TV metric, again over the last 500 iterations. If the slope of both lines is below a given threshold, it deems the global fitness and TV to be stagnant. When stagnation occurs, the stopping criterion is met. To provide statistically meaningful results and due to the stochastic nature of the evolutionary algorithm, it performed 10 evolutionary reconstructions with and without the new stopping criterion. It tested this approach using three controlled test cases (see Phantoms 1, 2, and 3 in Figure 5.12), therefore running 60 reconstructions in all. Figure 5.12 shows the reconstructed images corresponding to the median value of the total number of iterations needed for each test case.

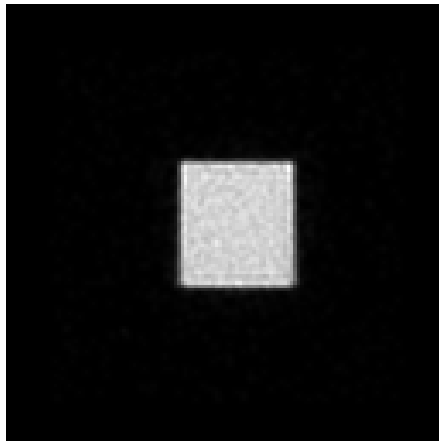
The performance, in terms of the total number of iterations needed, global fitness, TV, and ZNCC between the reconstruction and ground-truth, is summarised in Table 5.3. The total number of iterations have been reduced by 68%, 67%, and 58% on average for Phantom 1, 2 and 3 respectively. It did not lead to any loss of accuracy as the ZNCC between the reconstructions and the ground-truth has marginally improved (by less than 0.5% for the three test cases). The TV metrics of the images reconstructed with and without the new stopping criterion are also consistent. The study can conclude that the data exploration using visualisation has lead to a new stopping criterion that significantly reduces the computing time without any loss of accuracy.



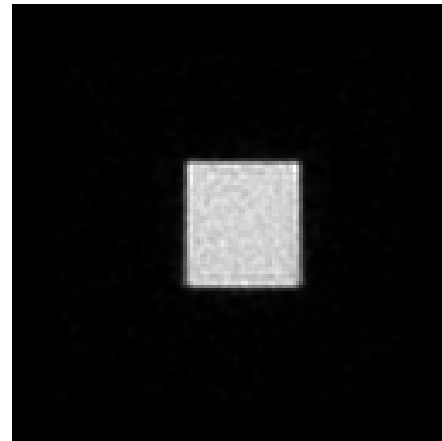
(a) Phantom 1 without.



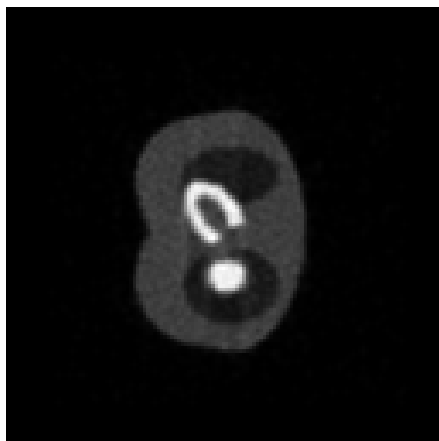
(b) Phantom 1 with.



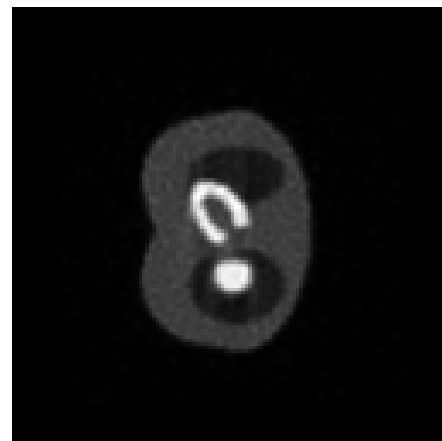
(c) Phantom 2 without.



(d) Phantom 2 with.



(e) Phantom 3 without.



(f) Phantom 3 with.

Figure 5.12: Reconstruction of Phantoms 1, 2, and 3 without and with the new stopping criterion.

5.5 Conclusion

The research presented here relies heavily on a fully adaptive implementation of a Cooperative Co-evolution Algorithm based on the Fly Algorithm. The purpose of this algorithm is to optimise the location of 3-D points. The final set of points corresponds to the solution of the optimisation problem. The study used this algorithm to solve a complex ill-posed inverse problem: tomography reconstruction in nuclear medicine. To date, the solution to the optimisation problem was extracted at the end of the evolutionary loop.

This chapter investigated the use of a simple but effective visualisation. It relies on Parallel Coordinate Plot, scatterplot and image display. The visualisation is used to explore the huge quantity of time series data generated by the algorithm during the optimisation loop. It focused, in particular, on metrics related to image accuracy, smoothness, and reconstruction time. It demonstrated that the final population may not be the most suitable solution and that preceding candidate solutions have to be considered to ensure that the reconstruction is accurate and not too noisy. This was not trivial as smooth images may not be accurate. This investigation allowed to demonstrate that increasing the population size, and hence the computation time, did not necessarily lead to a significant increase in quality of the reconstruction.

This approach can be easily deployed to any evolutionary algorithm (not only Parisian Evolution) where the quality of the solution cannot be measured by a single value (usually the fitness function). It is particularly suited to multi-objective optimisation where several concurrent fitness functions are used to assess the quality of an individual. All the objectives are equally important. Multi-objective optimisation algorithms often output a set of candidate solutions (the Pareto front). Choosing which solution is the best one may not be trivial. The decision maker with expert knowledge may be able to express preferences. An interactive visualisation similar to the one in this research has the potential to help the decision maker decide which solution(s) to pick amongst the candidates proposed by the algorithm.

The study used these results to propose a new stopping criterion. It analyses the local variation in terms of global fitness and smoothness of the reconstructed image

over the last 500 iterations. It allowed us to reduce the total number of iterations by almost 60% or more without any loss of accuracy.

Chapter 6

Using a multi-objective optimisation algorithm in interactive analysis of MRI gastric images

6.1 Introduction

The work presented here has been published in national and international conferences [114]–[116] and it is a contribution to a large project focused on the understanding of the influence of food structure on digestion. The approach is based on advanced imaging techniques to observe phenomena at different scales. It relies on MRI of the gastrointestinal tract (GIT) for capturing *in vitro* large scale information, while smaller scale measurements are performed *in vitro* on large facilities (small-angle neutron scattering (SANS), small-angle X-ray scattering (SAXS), and X-ray imaging) [51], [86]. The observation of *in vivo* digestion using MRI is a recent challenge. Here the focus is on the content of the GIT and not on the GIT itself as in clinical routine [91], [110], [122].

The work is part of a larger multi-disciplinary collaboration where recently acquired experimental MRI data of the stomach and duodenum area of healthy human volunteers. The aim is to analyse the content of the stomach and its evolution. The study focus here on the kinetics of gastric emptying for two species of ingested food: i) progressively and partially digested cooked pasta, and ii) frozen garden peas, which keep their shape in early gastric stages on the kinetics of gastric emptying for two species of ingested food: i) progressively and partially digested cooked pasta, and ii) frozen garden peas, which keep their shape in early gastric stages.

Manually processing this large amount of MRI data sets is not practically feasible. Processing it in a fully-automatic manner is not trivial as appropriate information need first to be collected and analysed to provide suitable models. Here the study showed how a “Fly Algorithm” [140] can be efficiently adapted to analyse these MRI images. The Fly Algorithm, on the contrary to classical image processing techniques, is able to provide a map of various features (e.g. the location of components of the food bolus). This work presented early results on the tracking of peas (around 20 peas in one stomach for the current experimental data), which reveals the motion of the stomach (the food bolus is stirred and gently “trituated” for better action of the gastric juice). For this purpose, the Fly Algorithm has been turned into a multi-objective cooperative-coevolution algorithm, and expert knowledge has been integrated through an interaction/visualisation interface.

Section 6.2 will describe the problem and give a definition for each objective will use in this work. In particular, the multi-objective scheme provides complex time series of Pareto front data, which needs to be understood and explored, Section 6.3 will explain that and produce the multi-objective algorithm (NSGA-II). Section 6.4 shows how simple, but yet effective, InfoVis techniques can be used to display the output of an evolutionary algorithm. Finally, the conclusion will conclude the work.

6.2 Problem definition

The experts selected a typical MRI image containing the stomach full of pasta and peas (see Fig. 6.1a). The MRI slice is manually cropped to only focus on the stomach and its content (see Fig. 6.1b). The study aimed to look for the peas by using FA that is need to describe what a pea looks like (See Figure 6.1c) and how to mathematically model it:

- Peas keep their shape and size in early gastric stages. A pea appear as a circle of a fixed radius of about 4mm, which is equivalent to $R = 8$ pixels.
- The interior of a pea is homogeneous; the outside is not.
- The interior of a pea is darker than the outside.

Table 6.1: Summary of the objectives. All these objectives are numerical values that need to be minimised.

Objective #	Properties	Figure
1	Local pixel intensity homogeneity within a darker circular ROI	Figure 6.2b
2	Heterogeneity outside a circular ROI compared to the homogeneity within the same circular ROI.	Figure 6.2c
3	Heterogeneity and homogeneity within a darker circular ROI and outside a circular ROI	Figure 6.2d
4	The isotropy for the circular ROI in objective 1.	Figure 6.2e
5	The isotropy for the circular ROI in objective 2.	Figure 6.2f
6	The isotropy for the circular ROI in objective 3.	Figure 6.2g
7	A circular ROI is darker than its background.	Figure 6.2h

Table 6.1 provides a summary the different objectives we used, alongside what they actually measure and their visual representation.

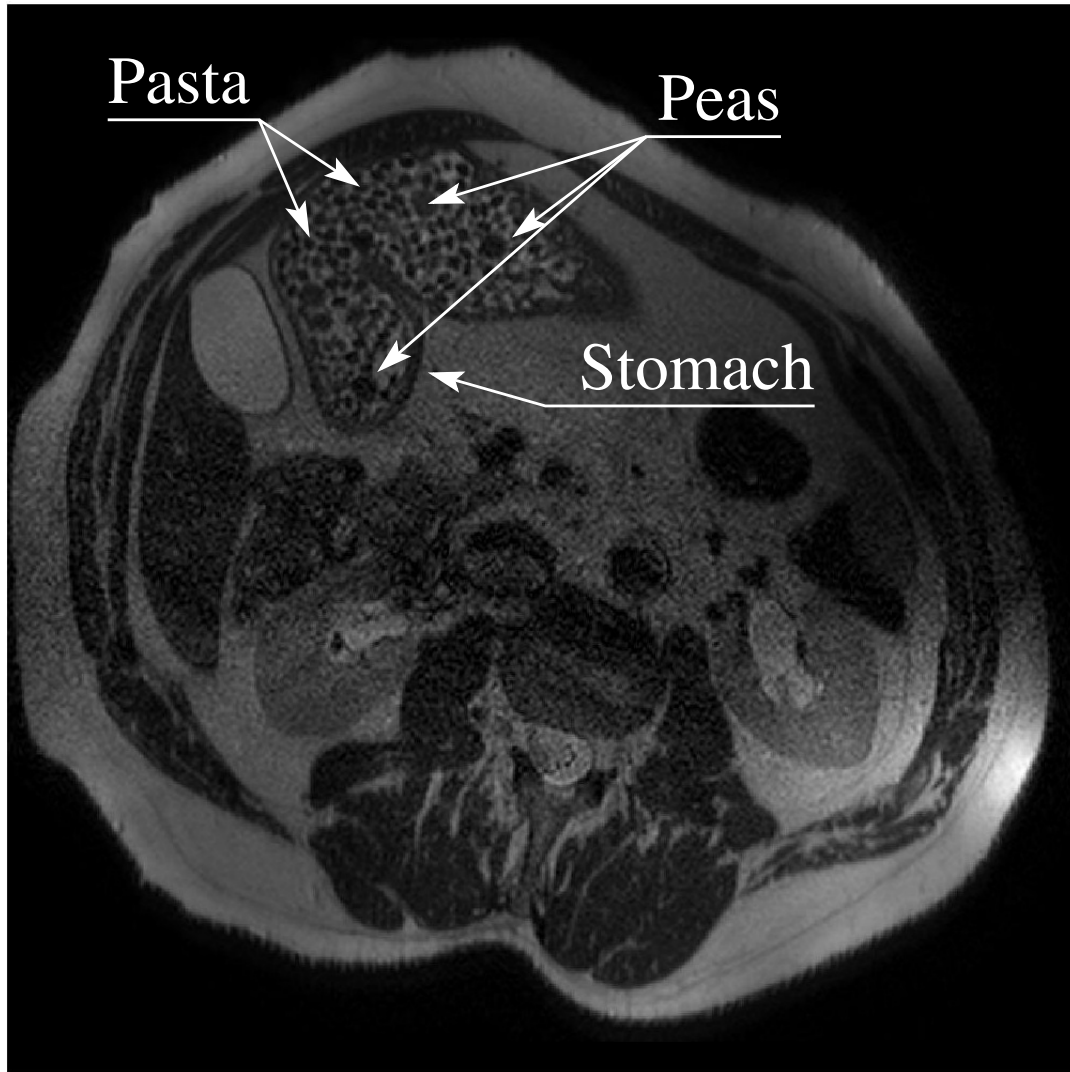
This study proposes to implement this knowledge into 7 objectives to minimise as a multi-objective optimisation problem. The size of the circular region of interest (ROI) depending on the pea radius is calculated as:

$$\text{diameter of } ROI_C(I, x, y, R) = d = 2R + 1 \quad (6.1)$$

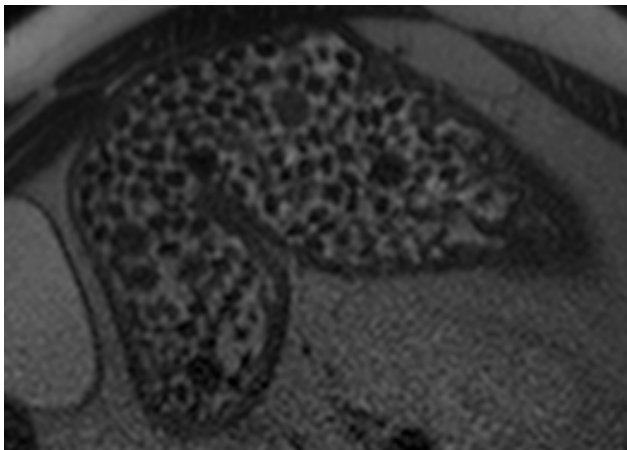
where $ROI_C(I, x, y, R)$ is a circular region of interest in Image I . It is centred on Pixel (x, y) , and its radius is R in number of pixels. The circular ROI can be defined as a Convolution kernel (see Fig. 6.3). We use a circular ROI, i.e. not squared one, because peas a spherical. For example, as peas are darker than their background, it enhances the differences between darker and brighter regions in Objective 1. For each pixel (x, y) of the MRI image, Objective 1 measures the local pixel intensity homogeneity within a circular ROI.

$$obj1(I, x, y, R) = \frac{1}{d \times d} \sqrt{\sum_{i=0}^n ROI_C(I, x, y, R)^2} \quad (6.2)$$

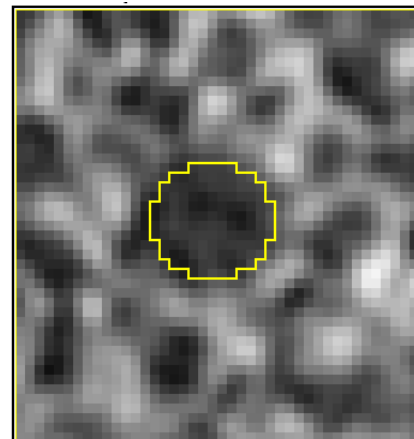
d is the diameter of $ROI_C(I, x, y, R)$, n is the size of $ROI_C(I, x, y, R)$. Fig. 6.2b is an image representation of Objective 1.



(a) MRI slice of a human stomach containing peas and pasta.



(b) Stomach selected ROI .



(c) The pea. Mean pixel value inside pea = 0.202, outside pea = 0.441.

Figure 6.1: MRI slice of a human body, the selected region of interest (Stomach), and the selected pea.

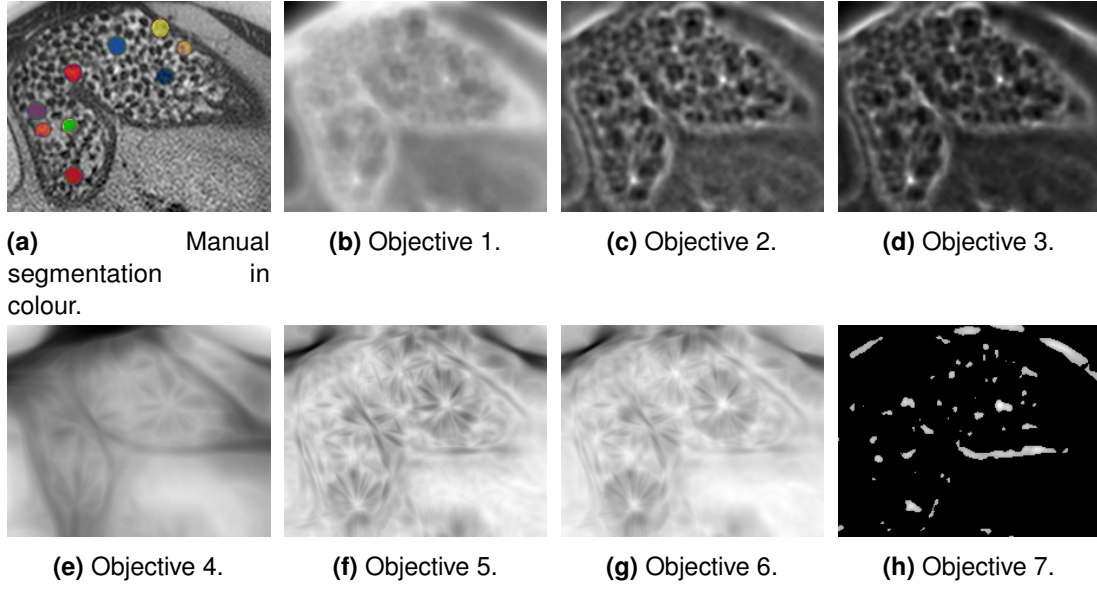


Figure 6.2: Objectives. For visibility objective functions are displayed in negative (low intensities appear bright; high intensities appear dark).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
3	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
4	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
5	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
6	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
13	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
14	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
15	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
16	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
17	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

Figure 6.3: Convolution kernel $ROI_C(I, x, y, R)$.

Objective 2 measures how homogeneous the interior of the circle is and how heterogeneous the outside is (see Fig. 6.2c). It compares Objective 1 with the local pixel intensity standard deviation within a ring region of interest (ROI_R) whose inner radius is R and outer radius $R + 5$:

$$obj2(I, x, y, R) = obj1(I, x, y, R) - obj1(I, x, y, R + 5) \quad (6.3)$$

Objective 3 combines Objectives 1 and 2 (see Fig. 6.2d):

$$obj3(I, x, y, R) = obj1(I, x, y, R) \times obj2(I, x, y, R) \quad (6.4)$$

When a pea is considered in Objectives 1, 2 and 3, the corresponding pattern in Figures 6.2b, 6.2c and 6.2d is isotropic. Each pea corresponds to a small bright dot onto a dark background in these images. Objectives 4, 5 and 6 exploit this property. The study modelled an intensity profile ($defP$) of 30 pixels using a triangular function. It mimics an intensity profile perfectly centred on a pea:

$$defP(i) = \begin{cases} 1 - \frac{|i - (2R - 1)|}{R - 1}, & \forall i \in \left[\frac{30}{2} - R, \frac{30}{2} + R \right] \\ 0, & \text{otherwise} \end{cases} \quad (6.5)$$

For each pixel in Figures 6.2b, 6.2c and 6.2d, The study extracted 89 intensity profiles ($prof_k$) every 2° around that pixel:

$$obj_{iso}(x, y, I) = -\max \left(\sum_{i=0}^{i < 30} (defP(i) - prof_k(I, x, y, i))^2 \right) \quad \forall k \in [0, 89) \quad (6.6)$$

where $prof_k(I, x, y, i)$ is i -th value of the intensity profile in Image I , centred of Pixel x, y at the k -th angle.

$$obj4(x,y) = obj_{iso}(x,y,obj1) \quad (6.7)$$

$$obj5(x,y) = obj_{iso}(x,y,obj2) \quad (6.8)$$

$$obj6(x,y) = obj_{iso}(x,y,obj3) \quad (6.9)$$

Objective 7 assesses that the interior of the circle is darker than the ring around it:

$$f(x,y) = \overline{ROI_C}(I,x,y,R-1) - \overline{ROI_R}(I,x,y,R-1,R+5)$$

$$obj7(x,y) = \begin{cases} f(x,y), & \forall f(x,y) \leq T \\ 0, & \text{otherwise} \end{cases} \quad (6.10)$$

where $\overline{ROI_C}$ is the average pixel value of a given circular ROI, $\overline{ROI_R}$ is the average pixel value of a given ring ROI, and T is a user defined threshold. $obj7(x,y)$ is expected to be negative or null, which is suitable for a minimisation algorithm. T is also negative. It restricts non-null values in $obj7$ to areas where the difference in pixel intensities of the two corresponding ROIs are significantly different, which correspond to the location of peas and stomach wall.

6.3 Multi-objective optimisation problem

Multi-objective optimisation is an approach to make a multiple criteria decision. It can be defined as mathematical optimisation problems concerning a set of the objective functions to be optimised at the same time. It has been utilised in many domains of science, including engineering, economics and logistics where need to make optimal decisions [70]. It provides a set of trade-off between objectives, the Pareto front. The Pareto front is the set of non-dominated solutions, i.e. points of the search space for which one objective function cannot be improved in value without degrading some of the other objectives (see Figure 6.4) [93].

Evolutionary optimisation methods can be adapted to deal with multiple-objectives, and various efficient algorithms are now available. **Non-dominated Sorting Genetic Algorithm (NSGA-II)** [42] is a very popular implementation, that is able to

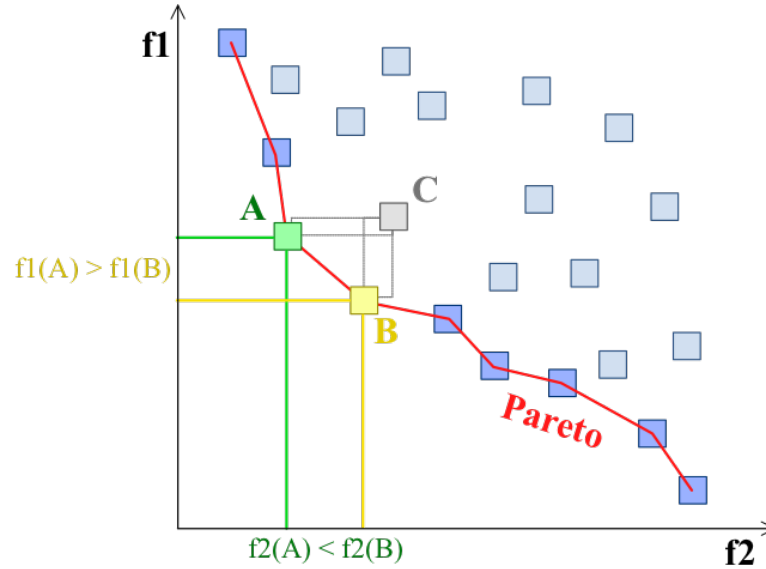


Figure 6.4: Example of a Pareto frontier (in red), the set of Pareto optimal solutions (those that are not dominated by any other feasible solutions). The boxed points represent feasible choices, and smaller values are preferred to larger ones. Point C is not on the Pareto frontier because it is dominated by both point A and point B. Points A and B are not strictly dominated by any other, and hence do lie on the frontier. Credit: image and caption by Johann Dréo, distributed under a CC BY-SA 3.0 license.

produce an efficient sampling of the Pareto front in a single run of the algorithm. All individuals are assigned front number 1 that not dominated by any other individuals. All individuals are assigned front number 2 that only dominated by individuals in front number 1, and so on [96]. NSGA-II is an improved version of NSGA [131]. The main steps for NSGA-II are:

1. Initially, create a random population P_0 .
2. Sort the population based on non-dominated f .
3. Calculate the average distance of two points on all sides of this point along each of the objectives, called the *crowding distance*.
4. Genetic operators (*selection, crossover and mutation*) are used to create a offspring population Q_0 of size N .
5. The elitism is found by comparing the current population with the best non-dominated solutions found previously, and the procedure is different after the initial generation.

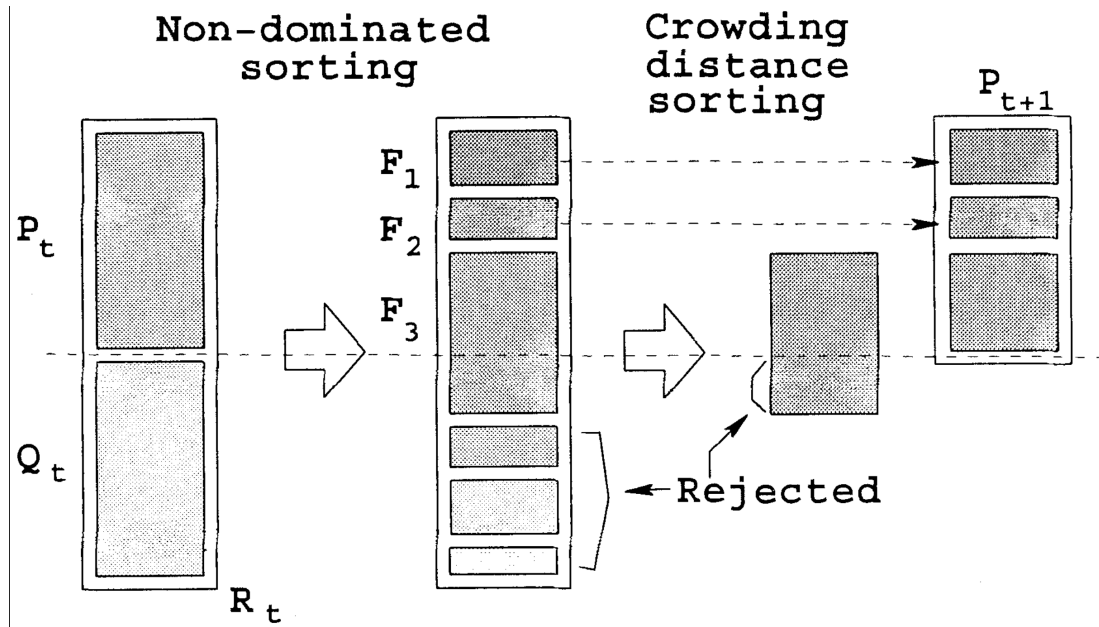


Figure 6.5: The main loop for NSGA-II algorithm. Where P_t is the population of parents, Q_t is the population of offspring, R_t is the combination of parent and offspring population and $F = (F_1, F_2, \dots)$ all non-dominated front R_t [42].

Figure 6.5 shows that the step-by-step procedure in NSGA-II algorithm is simple and straightforward. NSGA-II has the properties of a fast non-dominated sorting procedure an elitist strategy, a parameterless approach and a simple yet efficient constraint-handling method.

Peas detection is actually a non-trivial optimisation problem, as it involves multiple objectives that cannot be merged into a single one using simple rules (example: a circular uniform area, a defined colour, and an irregular background ...) and/or subjective preference weights. Multi-objective optimisation considers all objectives as equally important and non comparable.

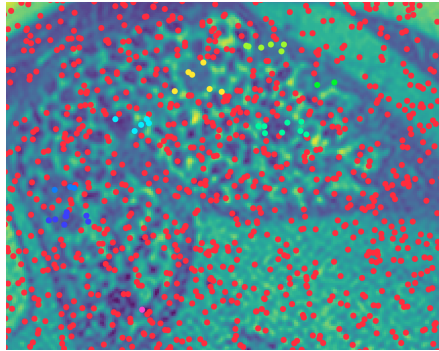
Here, we implement a multi-objective version of the Fly Algorithm based on NSGA-II so that the population of flies stabilises onto a Pareto front. 1000 individuals are used over 25 generations. These parameters were obtained empirically. Once a Pareto front (i.e, a set of possible best solutions) is produced, the decision may be put in the hand of an expert. For this purpose an interactive visualisation tool was developed to enable the expert explore the data produced by NSGA-II.

6.4 Visualisation

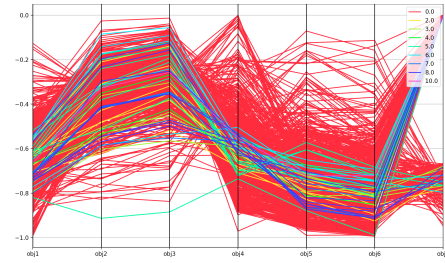
The output is a dataset of 1000×25 samples, i.e. (x,y) positions with 7 associated fitness values. In typical multi-objective EAs, one of the individuals on the Pareto front is a possible answer to the optimisation problem. The Fly Algorithm approach provides a set of points as a solution, each point corresponding not only to a possible pea location, but also to a different objective priority assessment. Several flies of the population may co-exist on the same pea with different objective weights. Automatically extracting the points that really correspond to peas is not trivial. Extracting only one point per pea, the point the closest to the centre of the corresponding pea, is even more complicated. This can be done efficiently through an interactive visualisation interface.

Each generation is displayed twice at a time. A scatterplot is used to display the position of the individuals over the MRI image (see left-hand side column of Fig. 6.6). A parallel coordinate plot is used to display the values of the multiple objective functions corresponding to each individual (see right-hand side column of Fig. 6.6). This plot represents a point in a n -dimensional space as a broken line with $n - 1$ segments, joining its n coordinates located on n vertical axes. The user can easily and interactively select areas of points in the scatterplot that correspond to peas. Each new manually selected area is assigned a new unique colour, which is the same in both plots. With this tool, the behaviour of individuals toward a global optimal solution in each generation can be visually detected. It also helped to understand the relationship between positions in peas and objective functions.

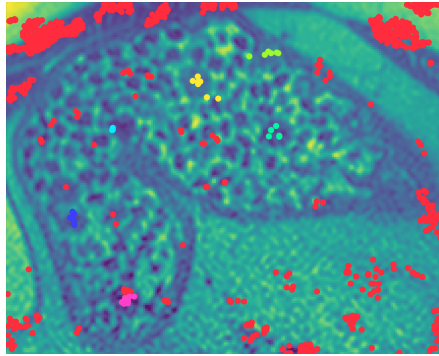
The result is used to define 7 validity ranges (two thresholds per objective) that filter out the 25,000 individuals generated during the evolutionary process. Only the individuals meeting all 7 validity ranges are considered in the following steps, others are discarded. After that, groups of points in the 2-D space are identified using clustering based on a Gaussian Mixture Model (GMM) (see Fig. 6.7a). Clustering aims to group a collection of points into subsets such that those within each cluster are more closely related to one another than points assigned to different clusters. The model GMM is the most widely used in practice. its perspective, each cluster can be mathematically represented by a parametric distribution [61]. Clusters



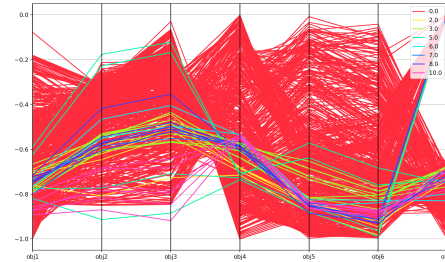
(a) Scatterplot of the initial population.



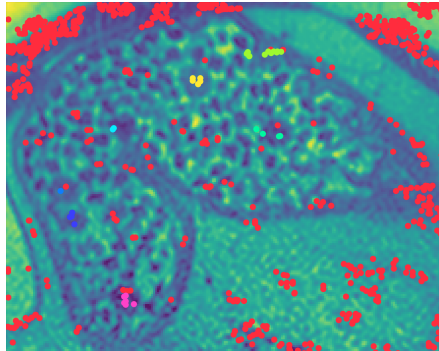
(b) Parallel coordinate plot corresponding to (a).



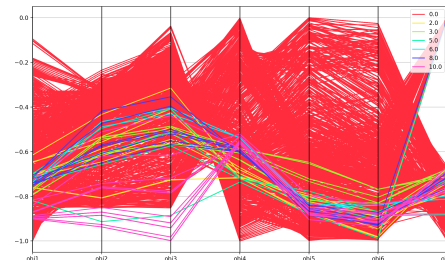
(c) Scatterplot of the 6th generation.



(d) Parallel coordinate plot corresponding to (c).



(e) Scatterplot of the best solution (16th generation).



(f) Parallel coordinate plot corresponding to (e).

Figure 6.6: Scatterplots and parallel coordinates plots of successive generations. All solutions (flies) are plotted in red by default. When the user selects an area in the scatterplot, a specific colour is assigned to this area and linked to the corresponding lines in the parallel coordinate plot.

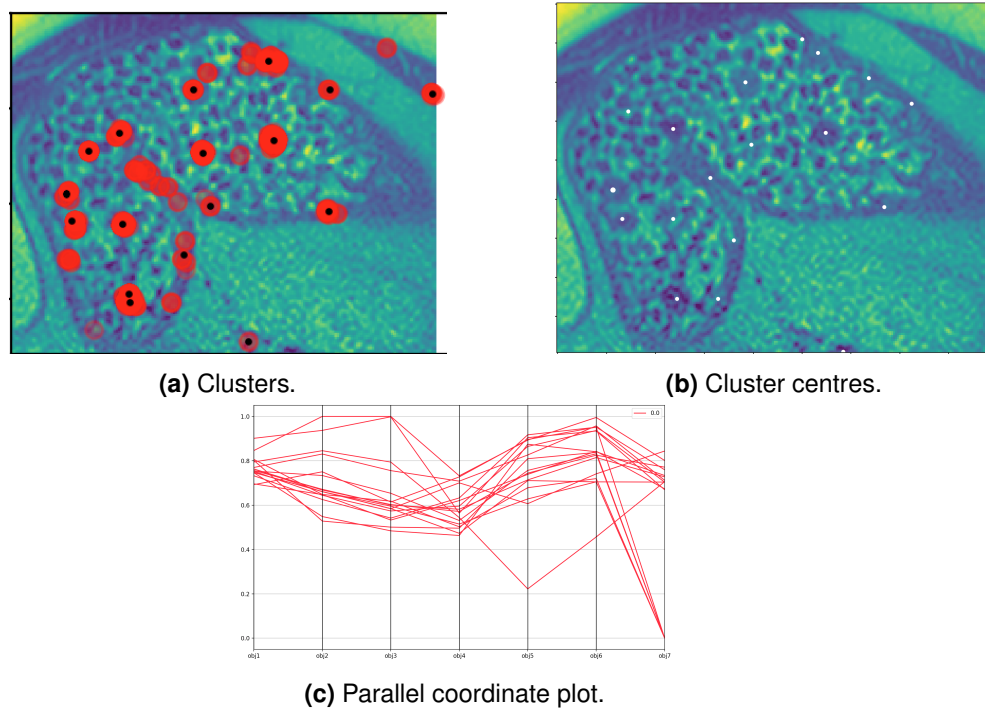


Figure 6.7: Candidate solution clusters.

that are close to each other (e.g, within a pea diameter) are then merged into a single cluster. All the cluster centres are extracted and presented to the user (see Fig. 6.7b). By using another parallel coordinate plot a new set of thresholds is extracted (see Fig. 6.7c). It is used to further refine the results and limit the number of false positive (i.e, points that do not actually correspond to peas).

The last set of thresholds is used so that stronger candidates are highlighted using a purple dot in Fig. 6.8; weaker candidates using a red dot. In total, 19 points were selected. Note that the selection depends on personal preferences and that experts debated whether some of the points near the wall of the stomach correspond to peas or not. It includes 9 points located on peas, and 10 wrongly selected. From the 9 points selected by the experts, 7 peas were highlighted in purple represent right position and 2 peas highlighted in red.

6.5 Conclusion

This study combines computer vision and visualisation/data exploration to analyse and understanding the kinetics of gastric emptying using MRI images of the stomach of healthy volunteers, and detect garden peas inside the human stomach by using Fly Algorithm. Also, it presented some preliminary results of the semi-automatic

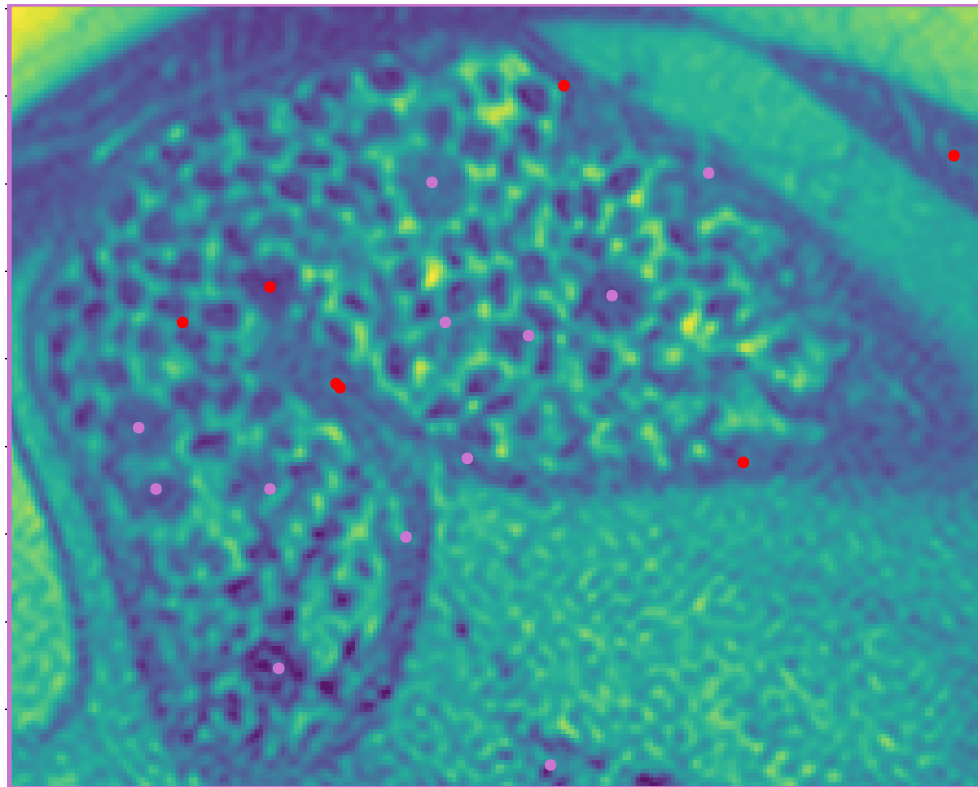


Figure 6.8: Final result. The purple colour represent the points in the right location, and the red colour represent the wrong location,

evolutionary segmentation of garden peas in MRI images. This work proposed multi-objective implementation of the Fly Algorithm where several concurrent fitness functions are used to assess the quality of an individual. Choosing which solution is the best one may not be trivial.

An interactive visualisation, combining image display, scatter plot and parallel coordinate plot and clustering, are used to analyse the output of the evolutionary algorithm. It helps understanding the complex relationship between the objectives and extracting individuals of the Pareto front that correspond to peas. In addition, to segment MRI using NSGA-II, only possible if using an interactive visualisation to exploit the enormous amount of data generated during the search.

This research demonstrated that manual thresholding on the data generated by NSGA-II using interactive visualisation can be used to semi-automatically detect peas in MRI images. In the following chapter, we will aim at deploying state-of-the-art Machine Learning learning techniques (namely deep learning implemented using Convolutional Neural Networks) to replace the manual thresholding. **The Machine**

Learning was not initially feasible due to the lack of labelled training data when this research was performed.

Chapter 7

Recognising specific foods in MRI scans using CNN and visualisation

7.1 Introduction

This chapter presents a collaborative study [53]. The usual focus of GIT medical imaging techniques is on the organs, while in this study we aim at following *in vivo* digestion, thus the evolution of the content of the organs (i.e. food deconstruction). The kinetics of gastric emptying were observed for two combinations of ingested food mixed together: progressively and partially digested cooked pasta (resp. white bread), and frozen garden peas (resp. “petits pois”, smaller), which keep their shape in early gastric stages. New acquisition protocols were thus developed. But adapted image processing are also needed.

As usual for such *in vitro* studies, the size of the cohort is rather small (here 10 volunteers), the development of adapted image recognition algorithms is thus difficult due to the availability of only rather small learning sets. Supervised learning algorithms, in particular CNNs, are known to work best with a huge database of training data. Also, the execution of CNNs, as most Deep Neural Networks (DNNs), is controlled by hyper-parameters. Poor choice of hyper-parameters can inhibit the potential of the model and lead to under fitting or over fitting.

Instead of the optimisation method we used in the previous chapter, we aimed to perform the segmentation as a binary classification task: given a point in the image, is there a pea or not? The previous study was based on a single dataset (named Dataset 1 in this thesis) from a single volunteer. The pixel resolution was

high, but so did the image acquisition time. To shorten this acquisition time when many volunteers contributed to the study, our collaborators had to considerably decrease the pixel resolution (from $0.5 \times 0.5 \times 2.0\text{mm}$ between two successive voxel to $0.98 \times 0.98 \times 2.0\text{mm}$). In addition, due to miscommunication amongst the experimentalists, petit-pois were used in this next experimental campaign. They are much smaller than the garden peas used in Dataset #1. Instead of cooked pasta, white bread was used, making it harder to distinguish petit-pois from their background. Due to these three factors, independent from us, our previous method based on NSGA-II is no longer applicable.

In this study, the database of cases contains 10 volunteer datasets from which 45 MRI slices were manually labelled by domain experts. They extracted 401 peas in total. 2,250 non-peas were automatically extracted. We aimed to demonstrate that it is possible i) to train a deep CNN to recognise peas despite a very limited and imbalanced database of cases, ii) to rely on a leave-one-out cross-validation over the dataset for training and testing, which emulates a more realistic scenario once the approach is deployed, iii) to use an interactive visualisation to find the best suitable combinations of hyper-parameters (here image size and number of epochs) to train the CNN, and iv) to obtain a relatively acceptable binary classification despite the constraints concealed in the database of cases over which we have no control.

The next sections introduce object recognition and how machine learning (including deep learning techniques such as CNN) can be used to detect features in objects. Section 7.5 describes the data used in this study and how it was prepared with object recognition in mind. Section 7.6 details how the approach was implemented and how to evaluate the result of the classification. After training and testing the CNN with various combinations of input parameters, the classification results are difficult to analyse in a tabular format. We show in Section 7.7 how interactive visualisation (a parallel coordinate plot and two radar charts) helped us identify the most promising combination of parameters. We conclude the work in Section 7.8.

7.2 Object recognition

Object recognition aims to identify objects in images or videos. It is a computational challenge, producing an entire research field in computer vision alone. Object recognition traditionally works by the use of a sliding window [55] such as the famous template matching technique [31].

Other traditional techniques include using machine learning algorithms, e.g. classifiers, to detect trends in features extracted in images using image processing techniques. For example Histogram of oriented gradients (HOG) aims to identify people from their shape in the image [92], whereas the Viola-Jones algorithm focuses on detecting specific facial features [146]. However, these two techniques are not always applicable to recognise other objects. CNN, pioneered by Yann LeCun, solves this problem [78]. By training a Neural Network (NN) on a dataset of training images, it can learn to detect, classify and recognise many different object types without making any assumption on the type of features to use. The major drawback is that for the best and most accurate CNN detectors there is often a requirement for large amounts of training data. This training data would have to be hand labelled and can be possibly hard to get, if not impossible. For example training CNNs on medical images, such as the ones available in this project, is often prohibited due to the lack of availability of sufficiently labelled data in sufficient quantity.

7.3 Machine Learning approach

In machine learning, computational statistics and computer vision techniques are used to algorithmically detect features and objects through statistical comparisons at the pixel level, based either on the pixel intensity or on other statistical data.

The Viola Jones object detection framework detects objects by using simple features [146] based on Haar functions [145]. These are very similar to kernels used in computer vision techniques like line detection, filtering, and CNNs. These functions act like a sliding window, going over regions of the image and doing some form of comparison or calculation. In facial recognition, this is done to detect

features such as eyebrows by comparing the estimated region of the eyebrows to the skin around it. It is typically done in grey scale. The eyebrow intensity is lower, or darker in colour terms, than the surrounding skin, which would be whiter (i.e. higher intensity). The same method can be applied to detecting the eyes from the cheeks, and the bridge of the nose. All these features must be detected for the algorithm to determine that the image contains a face.

Histogram of oriented gradients (HOG) differs from the Viola-Jones algorithm and many deep learning methods of object detection: it does not try to detect or determine objects based on specific features or learned weights. HOG encodes a global feature, e.g. the whole shape of a person or car. It relies on the fact that an object's shape is determined by the distribution of intensity gradients. HOG works similarly to edge detection and will detect human silhouettes from its contours. The image is divided into small uniform regions. A histogram of gradient directions is compiled for each region using the sliding window method going over these regions. A descriptor is formed from a combination of histograms [38]. This technique is best suited for human detection where people are upright and mostly visible.

7.4 Deep Learning

Deep learning techniques for object recognition require training a CNN to detect or recognise specific classes in images. This requires collecting a set of data and typically manually classifying the images. For the best and most accurate results, a large dataset of a good ratio of easy and hard examples are required, it is advisable to include lots of negative examples (note that the database of cases used in this project is very limited and imbalanced). There are many open source datasets that exist for training networks such as Common Objects in Context (COCO), which contains a vast amount of pre-classified images [80]. COCO and other datasets such as the PASCAL Visual Object Classes Challenge (VOC), are often used as testing sets to compare object detection frameworks mean Average Precision (mAP). However, COCO and many other datasets are often broad and would not be useful for more specific tasks such as pea recognition in MRI images.

With the popularisation of CNNs in recent years, many research projects are being produced in this field and they often grant public access to their object detection frameworks. The CNN can be controlled its capacity by varying its depth and breadth, and they also make robust and mostly right assumptions about the nature of images (in other words, stationarity of statistics and locality of pixel dependencies) [76]. This gives the work a wide range of trainable networks optimised for object detection, that are constantly being improved upon and bested by other network models. A current popular approach is the use of Google's TensorFlow 2.0, a specialised numerical computation library for deep learning, in Python via Keras, a high-level application programming interface (API) capable of running on top of TensorFlow [21]. A popular alternative is PyTorch, which was initially developed by Facebook [101]. Both approaches support graphics processing units (GPUs) to speed up computations.

Many algorithms rely on parameters, known as hyper-parameters, to control their execution. Neural Networks are not an exception, because CNNs have enormously fewer connections and parameters, and so they are easier to train compared to standard feed forward neural networks with similarly sized layers. Setting the values of hyper-parameters may significantly influence the outcome of the classification e.g. due to under fitting or over fitting. Tuning hyper-parameters is then recommended, but it is complex and is computationally expensive in our case [50].

7.5 Data repository

We saw that supervised learning algorithms, particularly Neural Networks, require a large amount of training data. The data used to train and test the CNN in this study is collected from MRI scans (see Figure 7.2a for an example of MRI slice). Table 7.1 shows a summary of the data selected from all the volunteers. This work build a repository of anonymised medical images that is organised in a very formal way (e.g. where files are and how directories and files are named) to enable batch processing.

Due to the nature of the study, the amount of data that is available is extremely limited (10 datasets only, with 45 slices in total, see Table 7.1) and unbalanced

Table 7.1: Data extracted from all the datasets.

Dataset #	# of slices	Pixel spacing (in mm)	Pea diameter (in mm)	# of peas	# of non peas
1	2	[0.5, 0.5, 2.0]	~ 8	11	50
2	1	[0.98, 0.98, 2.0]	~ 5	0	50
3	7	[0.9, 0.9, 2.0]	~ 5	102	50
4	8	[0.9, 0.9, 2.0]	~ 5	83	50
5	8	[0.9, 0.9, 2.0]	~ 5	87	50
6	5	[0.9, 0.9, 2.0]	~ 5	63	50
7	1	[0.9, 0.9, 2.0]	~ 5	5	50
8	3	[0.83, 0.83, 3.0]	~ 5	20	50
9	4	[0.83, 0.83, 3.0]	~ 5	14	50
10	6	[0.83, 0.83, 3.0]	~ 5	16	50
Total					
10	45	N/A	N/A	401	2250

(401 peas vs 2,250 non-peas). The initial MRI scan use a fine voxel resolution, with garden peas and small Italian pasta. However the scanning time was too long to process the whole cohort of volunteers. Several lower resolutions were then used to shorten the scanning time, but in the case of petits pois, which have a much smaller radius than garden peas, combined with bread whose shape is not clearly visible, some of the data which also had a larger voxel spacing was too challenging to manually segment. Hence the number of slices manually segmented per dataset varied widely.

Fig 7.1 shows the main steps. For each new volunteer dataset, a set of ‘interesting’ slices is manually selected by experts (see Step 5 and Figure 7.2a for an example of MRI slice). They selected slices, where peas were clearly visible and separable from the background (i.e, pasta or bread depending on the scan), and some where they were not. They selected slices with many peas, some with just a few. They also included a dataset with no pea at all as they wanted us to demonstrate the ability of our approach to process MRI slices with no pea as they are likely to occur, e.g, when the digestion is in an advanced stage. For each slice, the expert manually extracts a ROI that contains the stomach (Step 6) and (see Figure 7.2b for an example of ROI). This is done using a Python script. Using the same program the expert then clicks on each pea that he/she can identify (Step 7). For each pea, the expert rates his/her level of confidence from 0 (not sure if it is a pea) to 3 (definitely a pea). Figure 7.3 shows two selected peas, but with a different level of confidence. Once peas are manually segmented, 50 samples are randomly

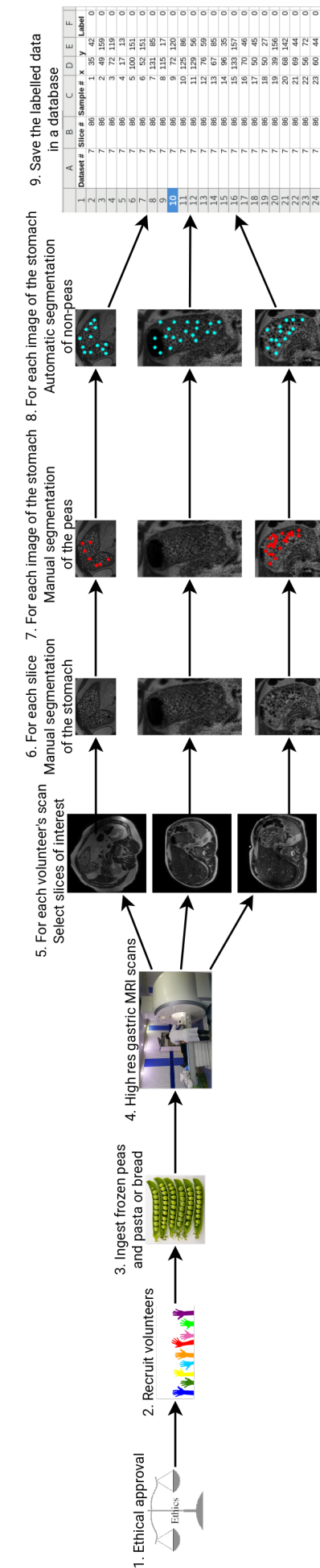
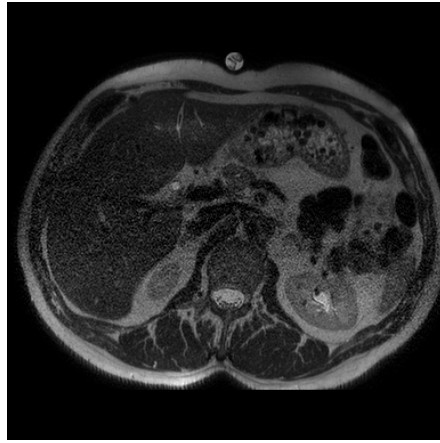
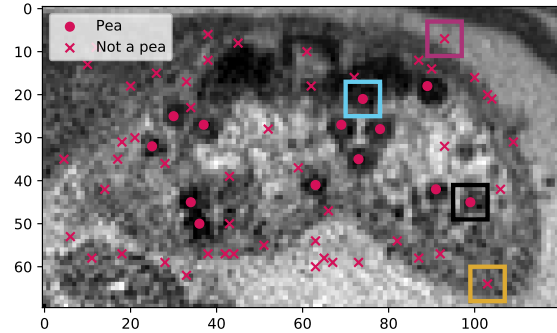


Figure 7.1: The steps for preparing data by the experts and save the labelled data in database.



(a) MRI slice #45 of Dataset #4.



(b) The human stomach extracted from slice #45 of Dataset #4. A circle depicts a sample labelled as a pea, a cross depicts 'not a pea'. ROIs highlighted with square are shown in Figures 7.3 and 7.4..

Figure 7.2: MRI slice of a human stomach and the manual segmentation for the peas.

chosen in the ROI (Step 8). These samples are located outside peas (see crosses in Figure 7.2b and Figure 7.4 for two examples). The manual segmentation of peas and automatic segmentation of non-peas is eventually saved in the database a CSV with its associated ROI image.

7.6 Recognising Peas in MRI scans of the stomach

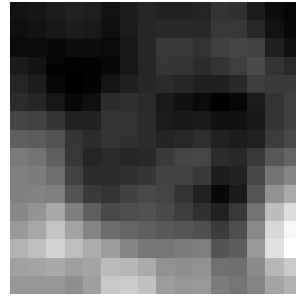
The study relied on a deep Convolutional Neural Network implemented in Python using the Keras deep learning API [33] that runs on the top of TensorFlow [90]. Our implementation is actually based on Keras' deep CNN example provided on GitHub at https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py.

The data is prepared as mentioned in Section 7.5. The data is stored in a CSV file, where each row corresponds to the pixel location of the sample in the given MRI slice in a volunteer dataset. For each sample, the label '1' corresponds to a pea; '0' to not a pea. The combination 'Dataset #' and 'Slice #' corresponds to the location of an image file in the repository.

For suitability with supervised learning, the data is then split into training and testing sets (see Steps 1a and 1b in Figure 7.5). It is often performed randomly. However,

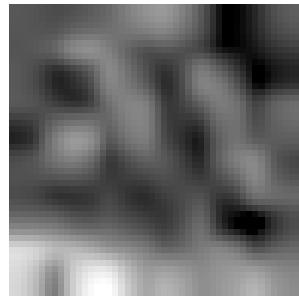


(a) Pea with a high level of confidence (2.94). See black square in Figure 7.2b.

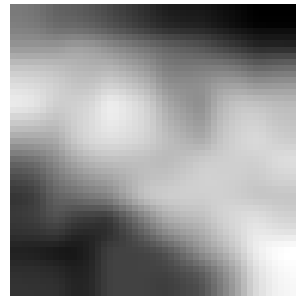


(b) Pea with a low level of confidence (1.02). See cyan square in Figure 7.2b.

Figure 7.3: Examples of peas with different levels of confidence (confidence between 0-3) in the selected ROI.



(a) See yellow square in Figure 7.2b.



(b) See purple square in Figure 7.2b.

Figure 7.4: Automatic selection of non-pea samples in the ROI.

to provide more realistic predictions when the tool is deployed, the work opted for a leave-one-out cross-validation where each volunteer dataset will be tested independently. It means that when the CNN is tested on Dataset #1, the CNN is trained on Datasets #2 to 10; when the CNN is tested on Dataset #2, the CNN is trained on Dataset #1 and Datasets #3 to 10; and so on. It prevents training and testing the CNN with slices from the same dataset, which has the potential to bias results. This way, when a new dataset is added to the repository, we will also know how the CNN should cope. However, built-in hyper-parameter tuning frameworks are not applicable with this leave-one-out cross-validation strategy, which is why we favoured an interactive visualisation. This way, the end-user has a fine control on which of the metrics (accuracy, precision, recall and F_1 score) to put an emphasis.

The CNN is trained using the training data (Step 2). The images used during all the training and testing must have the same size in number of pixels. It is also important that they also cover the same area in mm^2 , which must be large enough to include a pea. However, the voxel spacing in the medical data is different depending on the volunteer. As a consequence the input images are resampled so that the voxel

spacing is consistent over all the volunteer datasets. The number of pixels used in each image used during the training and testing is a parameter that the user can set.

Training CNNs is an iterative process where an iteration is called an epoch. The total number of epochs used to train the CNN is also a parameter that is set by the user. The trained model (Step 3) is used on the test data to provide predictions (Step 4). Predictions in our case are binary, is a sample a pea or not? The outcome for a sample can be:

TP: The sample was manually segmented as a pea by the domain expert, and the CNN predicted that it is a pea: this is a true positive;

FN: It is manually segmented as a pea, and the CNN predicted that it is not a pea: this is a false negative;

FP: It is manually segmented as not a pea, and the CNN predicted that it is a pea: this is a false positive;

TN: It is manually segmented as not a pea, and the CNN predicted that it is not a pea: this is a true negative.

For each test, the number of true positive (*TP*), false negative (*FN*), false positive (*FP*), and true negative (*TN*) are recorded. To evaluate the performance of supervised learning algorithms, metrics such as accuracy, prediction, recall, and F_1 score are often used. They all provide values between 0 and 1, 0 being the worst possible results, and 1 the best.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (7.1)$$

The number of correctly classified data instances over the total number of data instances. It is 1 when both *FP* and *FN* are equal to zero. This metrics is very

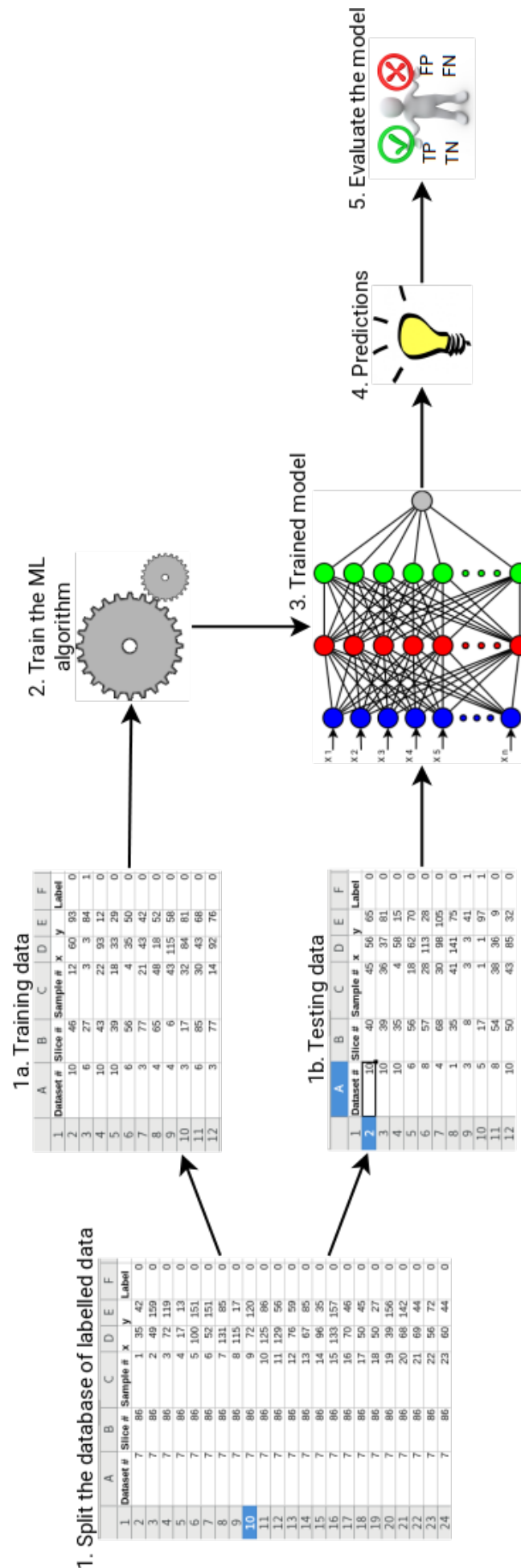


Figure 7.5: The main steps for implementing supervised learning.

popular when the class distribution is similar. However, the class distribution is extremely imbalanced in this study case.

$$Precision = \frac{TP}{TP + FP} \quad (7.2)$$

is 1 when FP is equal to zero. This metric is important when the cost of false positive is high, which is not necessarily the case in our application.

$$Recall = \frac{TP}{TP + FN} \quad (7.3)$$

is 1 when FN is equal to zero. This metric is important when the cost of false negative is high, which is this study case in this application.

Precision and recall are complementary. Ideally a good classifier will have FP and FN close to zero. F_1 score takes into account both Precision and Recall, which is why it considered to be a more robust metric than Accuracy.

$$F_1 \text{ score} = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (7.4)$$

It also makes F_1 score a better metric to use when the class distribution is imbalanced.

If TP is null, recall and precision are null too and F_1 score is equal to infinite. When both TP and FP are null, precision is equal to infinite; when both TP and FN are null, recall is equal to infinite. When both precision and recall are equal to infinite, F_1 score is not a number (NaN).

7.7 Filtering the results using interactive visualisation

The CNN is tested for each volunteer dataset using the leave-one-out cross validation methodology. This work evaluated the CNN using various image sizes (16×16 , 32×32 , 64×64 and 128×128 pixels) and number of epochs (32, 64, 256, 1024, and 2048). In total 200 (10 datasets \times 4 image sizes \times 5 number of epochs) trainings and testings were performed. The aim is to determine which combination

Table 7.2: Performance evaluation of the CNN for various image sizes and number of epochs.

image size	epochs	Accuracy	Precision	Recall	F_1 Score
16 × 16	32	0.961	0.900	0.833	0.865
16 × 16	64	0.960	0.860	0.875	0.868
16 × 16	256	0.967	0.913	0.863	0.887
16 × 16	1024	0.952	0.931	0.736	0.822
16 × 16	2048	0.896	0.931	0.334	0.492
32 × 32	32	0.966	0.912	0.855	0.883
32 × 32	64	0.963	0.909	0.843	0.875
32 × 32	256	0.962	0.877	0.873	0.875
32 × 32	1024	0.959	0.888	0.830	0.858
32 × 32	2048	0.961	0.896	0.838	0.866
64 × 64	32	0.961	0.894	0.840	0.866
64 × 64	64	0.960	0.894	0.838	0.865
64 × 64	256	0.955	0.851	0.853	0.852
64 × 64	1024	—	—	—	—
64 × 64	2048	—	—	—	—
128 × 128	32	0.954	0.841	0.855	0.848
128 × 128	64	0.956	0.851	0.858	0.855
128 × 128	256	—	—	—	—
128 × 128	1024	—	—	—	—
128 × 128	2048	—	—	—	—

of ‘image size’ and ‘number of epochs’ provides the best possible classification outcome. This is not feasible on a desktop computer because weeks of computations would be required. This is why the training and testing were performed on Supercomputing Wales’ supercomputer.

Table 7.2 shows the corresponding classification results in terms of accuracy, precision, recall, and F_1 score. The best two results for each column are highlighted in bold characters. The number of counts corresponds to the number of MRI slices that were tested. In 5 cases, when the image size and the number of epochs are both relatively large, it was not possible to perform the training due to the amount of computations that was required, even on a supercomputer. Increasing the image size, which significantly increases the training time, does not improve performance. Increasing the number of epochs does not seem to often improve the performance either, which indicates a possible over fitting.

Table 7.3: Classification results when using 16×16 images with a CNN trained over 256 epochs. The last row provides the accuracy, precision, recall and F_1 score when all the datasets are considered as a whole, i.e. these are not average values over the ten datasets, i.e. applying Eqs 7.1 to 7.4 with 346, 2217, 33 and 55.

Dataset #	TP	TN	FP	FN	Accuracy	Precision	Recall	F_1 Score
1	0	96	4	11	0.86	0.00	0.00	∞
2	0	50	0	0	1.00	∞	∞	NaN
3	88	347	3	14	0.96	0.97	0.86	0.91
4	80	397	3	3	0.99	0.96	0.96	0.96
5	86	394	6	1	0.99	0.93	0.99	0.96
6	61	245	5	2	0.98	0.92	0.97	0.95
7	5	47	3	0	0.95	0.62	1.00	0.77
8	7	150	0	13	0.92	1.00	0.35	0.52
9	10	194	6	4	0.95	0.62	0.71	0.67
10	9	297	3	7	0.97	0.75	0.56	0.64
Overall	346	2217	33	55	0.97	0.91	0.86	0.89

From the table, it is hard to identify a combination of ‘image size’ and ‘number of epochs’ that perform, the others. To address this issue, we plot each row using a parallel coordinate plot (see Figure 7.6). Figure 7.7 shows the same data, but as a radar chart using a linear scale and a logarithmic scale. However, it is still not possible to identify the best parameters using these three images.

It is possible to select a subset of the data with an input device (mouse) in the parallel coordinate plot by using brushing technique. It is used to interactively filter the data. The purple marks in Figure 7.8 on the ‘Accuracy’ and ‘ F_1 ’ score axis show the value ranges of interest. The radar charts are updated accordingly (see Figure 7.9). Three parameter combinations were selected, i) 16×16 pixels and 256 epochs, ii) 32×32 pixels and 32 epochs, and iii) 32×32 pixels and 64 epochs. Again, with the updated radar chart in linear scale, it is not possible to distinguish which parameter combination provides the best results. However, in the logarithmic scale it is clear that 32×32 pixels and 64 epochs is outperformed by the other two parameter combinations, particularly in terms of recall and F_1 score, and to a lesser extent accuracy and precision. The logarithmic scale radar chart also shows that 16×16 pixels and 256 epochs outperforms 32×32 pixels and 32 epochs for both recall and F_1 score, which are the two most important metrics to consider in this application. The two configurations provide similar accuracy and precision. Using this interactive visualisation enabled the study to identify that 16×16 pixels and 256 epochs should be used over the other 19 possible configurations.

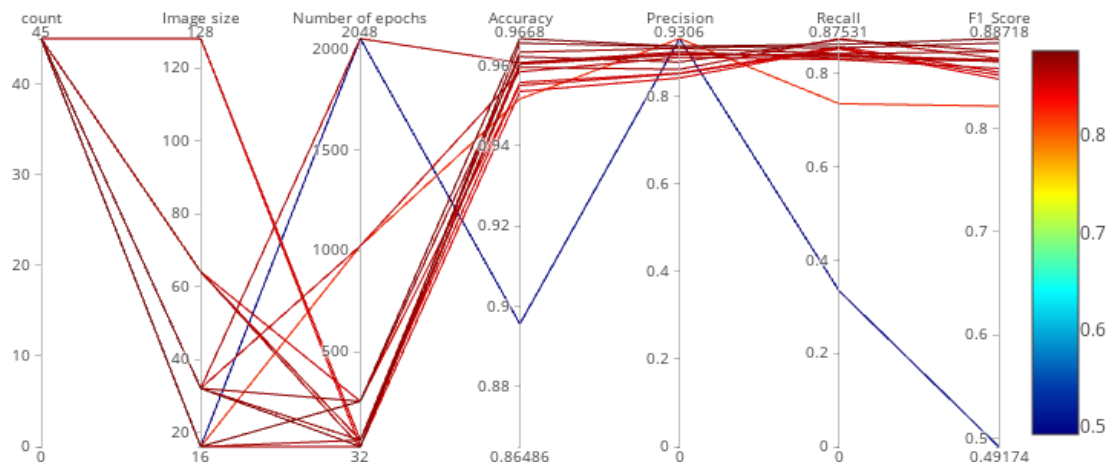
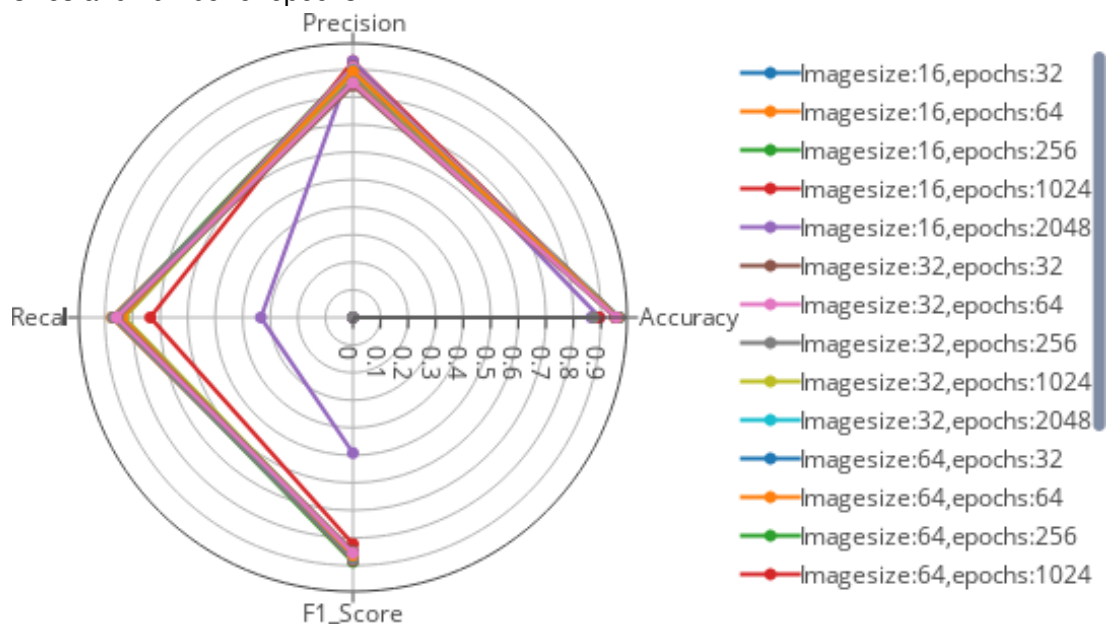
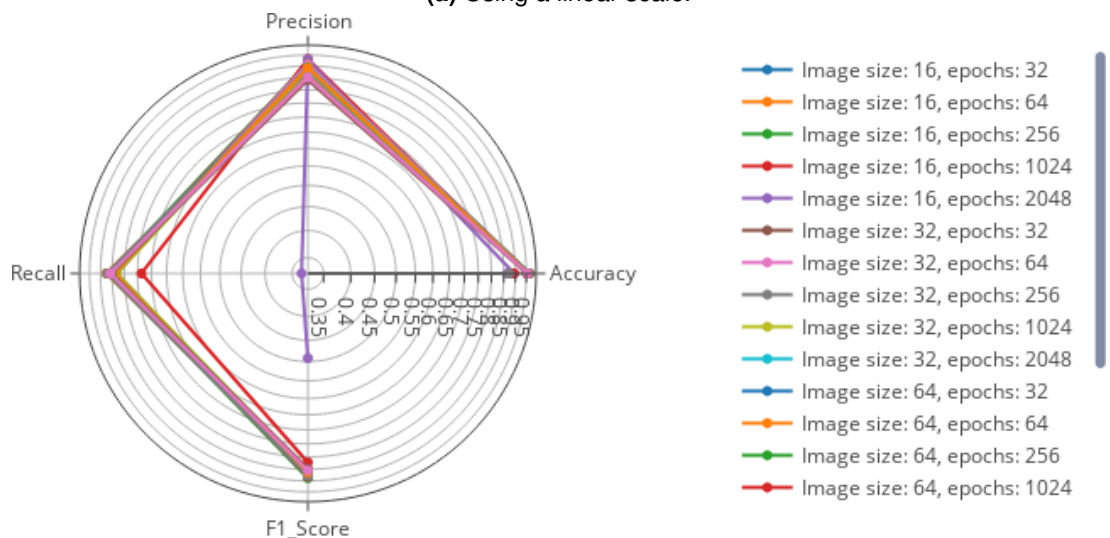


Figure 7.6: Parallel coordinate plots of the CNN performance metrics for various image sizes and number of epochs.



(a) Using a linear scale.



(b) Using a logarithmic scale.

Figure 7.7: Radar charts of the CNN performance metrics for various image sizes and number of epochs.

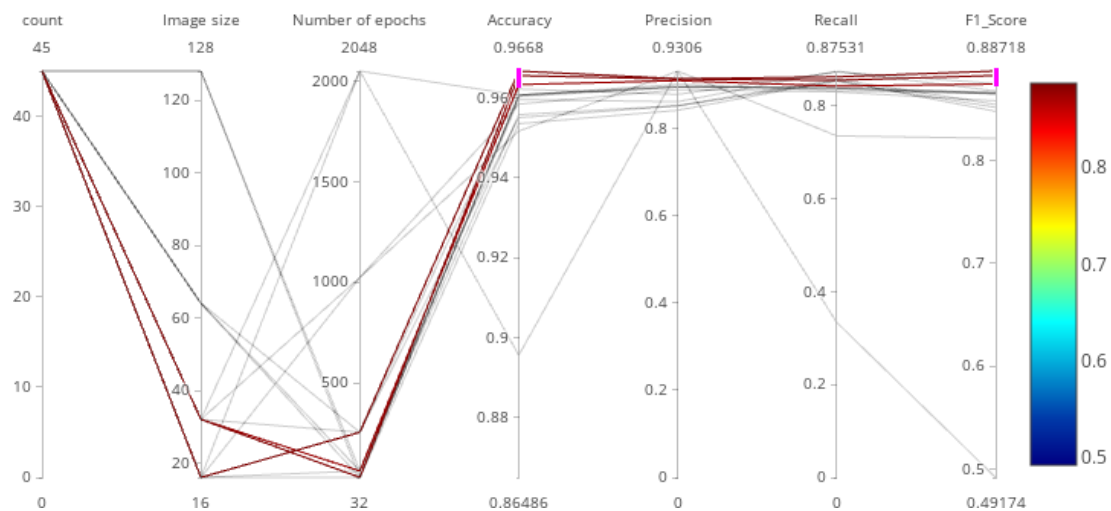
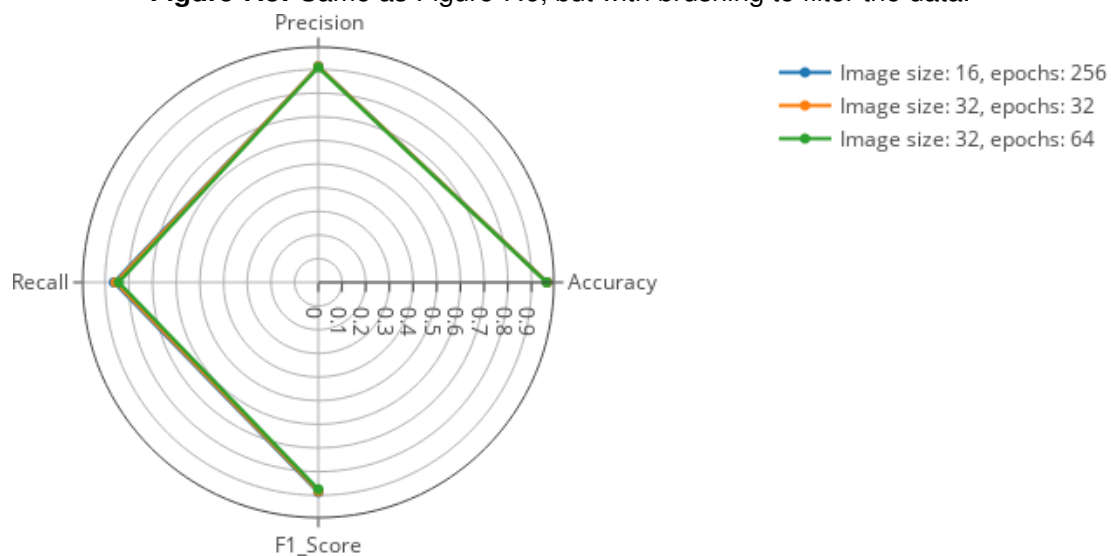
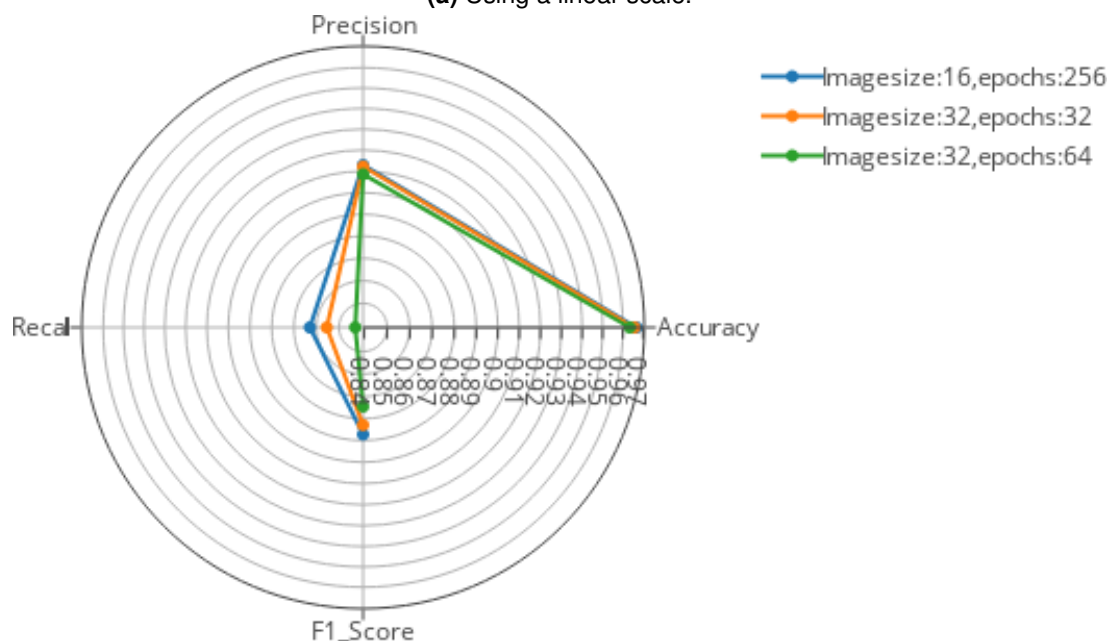


Figure 7.8: Same as Figure 7.6, but with brushing to filter the data.



(a) Using a linear scale.



(b) Using a logarithmic scale.

Figure 7.9: Same as Figure 7.7, but with the data filtered using brushing in Figure 7.8.

The corresponding results for each volunteer dataset are shown in Table 7.3. Dataset #1 was the first volume acquisition of the study. The pixel spacing is the smallest in this dataset (0.5×0.5 mm). However, the data acquisition was too long and the pixel spacing was therefore almost doubled. The pixel spacing in other datasets are 0.98×0.98 , 0.9×0.9 , or 0.83×0.83 mm. Despite that all the images used by the CNN are resampled to have the same number of pixels and the same pixel spacing, results for Dataset #1 are not consistent with others: None of the peas were detected in Dataset #1. The image selected in Dataset #2 did not contain any pea. The classification was perfect as there is no false positive. For other datasets, the proportion of false positives is very low: High precision. The proportion of false negatives is higher: smaller value of recall. Recall is actually relatively low for Datasets #8 and 10, and to some extent Dataset #9.

The similarity amongst Datasets #1, 8, 9, and 10 is that they have a higher MRI image resolution (i.e. smaller pixel spacing) than other datasets. These results could be due to a difference in image quality once resized to the same resolution, which makes the training dataset even more imbalanced. There are 2 MRI slices at pixel resolutions of 0.5×0.5 mm, 13 at 0.83×0.83 mm, 29 at 0.9×0.9 mm, and 1 at 0.98×0.98 mm. Figure 7.10 shows two peas after resampling of slices at different image resolutions. For the MRI scan protocol used during the data acquisition, when the resolution is lower, the data is grainier (see Figure 7.10a); and when the resolution is higher, the data is blurrier (see Figure 7.10b).

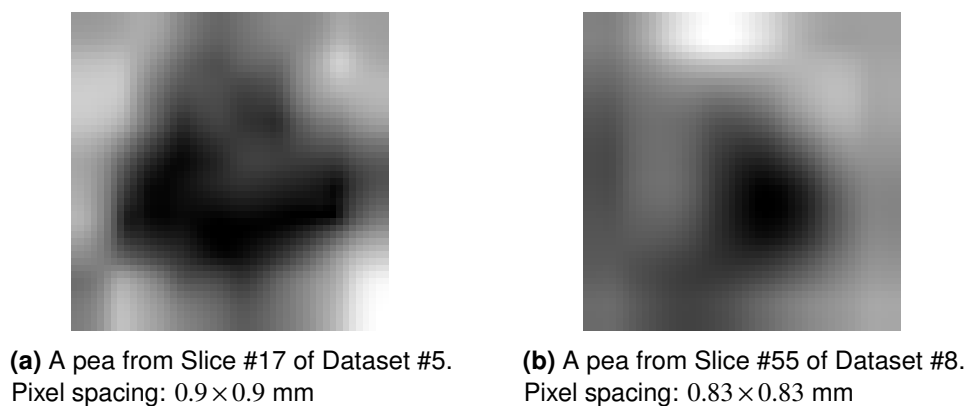


Figure 7.10: Peas after resampling of slices at different image resolutions (see corresponding images in Figure 7.11).

Overall, the CNN exhibits very good classification results, with high value of accuracy, precision, recall and F_1 score. Figures 7.11a and 7.11b show the

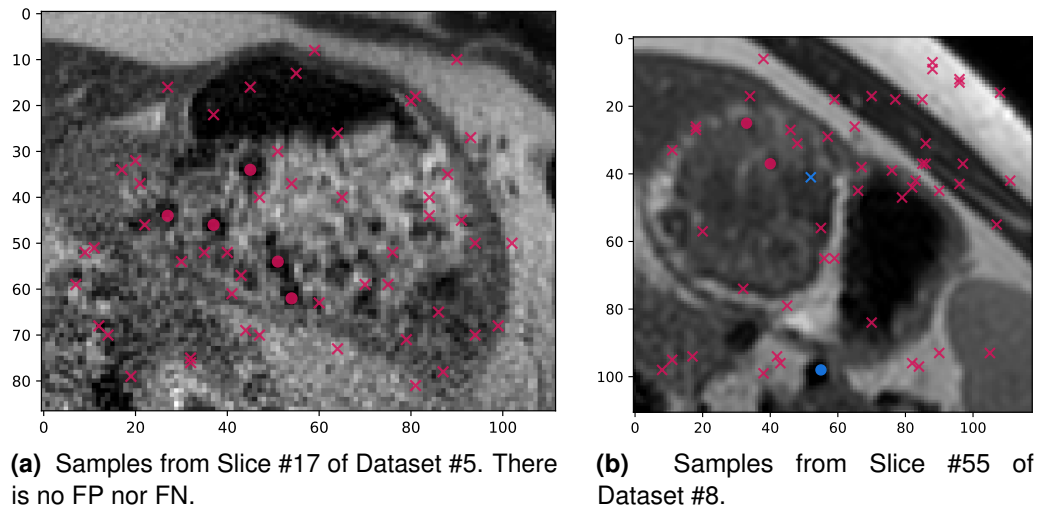


Figure 7.11: Visual representation of the classification results. Circles in magenta depict TPs, crosses in magenta TNs, blue circles depict FPs, and blue crosses FNs.

classification for MRI datasets at different pixel resolutions and with the two most extreme F_1 scores (0.96 and 0.52 respectively).

7.8 Conclusions

This feasibility study demonstrated i) how state-of-art techniques in machine learning (supervised learning and binary classification) and computer vision (object recognition) can be used despite the inherent limitations of the ad hoc database of images that used, and ii) how visual analytics complements them to achieve a suitable level of performance. This approach is directly applicable to other limited and imbalanced databases of images or to compare the performance of various classifiers (e.g. evaluate the performance change due to different combinations of hyper-parameters).

A deep CNN was evaluated with a limited (2,651 images only) and fairly unbalanced dataset (401 peas vs 2,250 non-peas) to determine if it is possible to recognise peas in MRI scans. The study relied on two popular Python packages, namely TensorFlow and Keras, for deep learning on GPUs. Here, the database of cases contain 10 volunteers datasets from which 45 MRI slices in total were manually labelled by domain experts. This task is highly time consuming, which explains why only 45 slices were manually labelled. To provide a meaningful evaluation of the

approach when the tool is deployed, a leave-one-out cross-validation strategy was favoured to train and test the CNN.

The two input hyper-parameters (number of pixels per image, and number of epochs to train the CNN) were exhaustively evaluated thanks to the computational power delivered by a supercomputer. The results were analysed using interactive visualisation to identify which combination of number of pixels/number of epochs provides the best possible outcome. With 16×16 pixels and 256 epochs, the accuracy is 0.97, precision 0.91, recall 0.86 and F_1 score 0.89. There were 346 TP , 2217 TN , 33 FP , and 55 FN , which is very promising considering the inherent limitations of the data we had to use.

These results highlighted that the initial image resolution of the MRI scan may play a role in the quality of the classification, which needs to be investigated further. In the next chapter, the keypoint selector used in previous chapter (evolutionary analysis of MRI gastric images using a multi-objective cooperative-coevolution [115], is integrated to automatically locate points of interest in new MRI datasets. It then feeds the data to the trained CNN model.

Chapter 8

Fully-automated Segmentation of peas using Multi-objective Optimisation and Machine learning

8.1 Introduction

In this chapter we compare the ability of several machine learning classifiers to predict the pea positions. We also develop our own classifier based on NSGA-II. We rely on the same training and testing data as in the previous chapter. In many ways we rely on the same methodology, but use different algorithms with the aim to automatically identify keypoints to label in the image. Our pipeline is as follows:

1. Manual segmentation (training dataset) (see Section 7.5).
2. Extraction of features from the training dataset (local image statistics on ROIs and objective functions) (see Section 8.2).
3. Training classifiers (list all the features) on the data:
 - Support Vector (SVC or SVM), AdaBoost, Decision tree, Gaussian Naive Bayes (GaussianNB), Gaussian process, Nearest neighbours, Quadratic discriminant analysis and Random forest (see Section 8.3),
 - a Deep Residual Network called Resnet-50 (see Section 8.4),

- our own classifier based on a multi-objective evolutionary algorithm, NSGA-II (see Section 8.5).
4. Automatically selecting keypoints in MRI images and final classification (see Section 8.6):
- Identifying the ‘best’ suitable classifier to predict the labels,
 - Applying the best NSGA-II-based model as classifier to generate binary images from features of new MRI images,
 - Deploying computer vision techniques on these binary images to extract keypoints that need to be labelled,
 - Running the selected classifier on these keypoints to refine the output provided by our NSGA2-based model,
 - Visualisation to present the results to the end-user (see Section 8.7).

8.2 Image data statistics

The data used in this research is the same data that collected from MRI scan in the previous chapter (see Chapter 7). In this study we propose to describe the problem using local image statistics and different objective functions “called features” for each selected point (peas and non-peas). Figure 8.1 shows the value of the features for all the points (peas and non-peas) in all the datasets using a parallel coordinate plot. The statistics such as; average, minimum, maximum, median, standard deviation and entropy rely on very few theoretical assumptions, whilst they provide good information to detailed quantitative predictions [128]. We also consider statistical moments (including skewness and kurtosis), which are often used to describe the shape of a distribution, here the distribution of pixel intensities.

Figure 8.2 shows the local image statistics and objective functions for the pixels of Slice #17 of Dataset #5 (see Fig. 8.2a). The measurements (local image statistics and objective functions) are computed once on the input MRI, and once on its

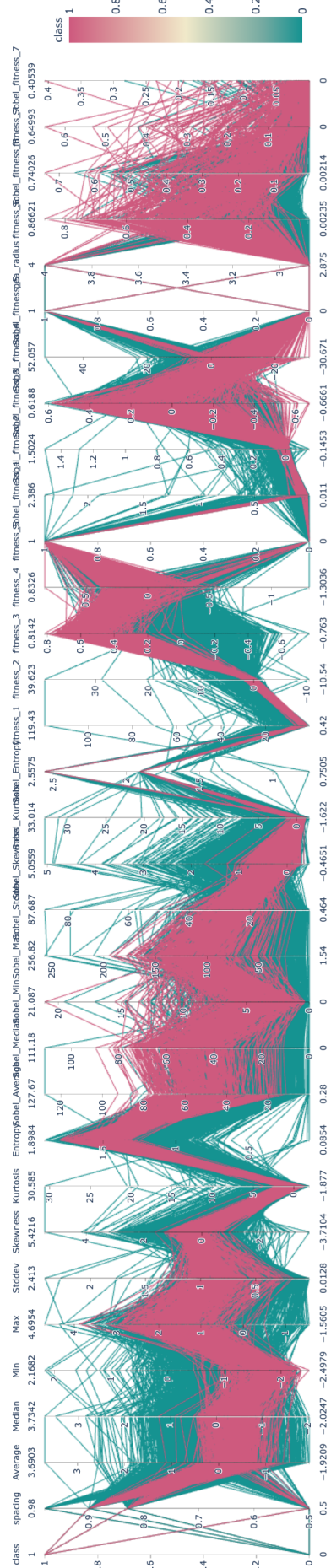


Figure 8.1: Parallel coordinate for all the selected points (peas) and all the random located points (non-peas) for all features from all the datasets.

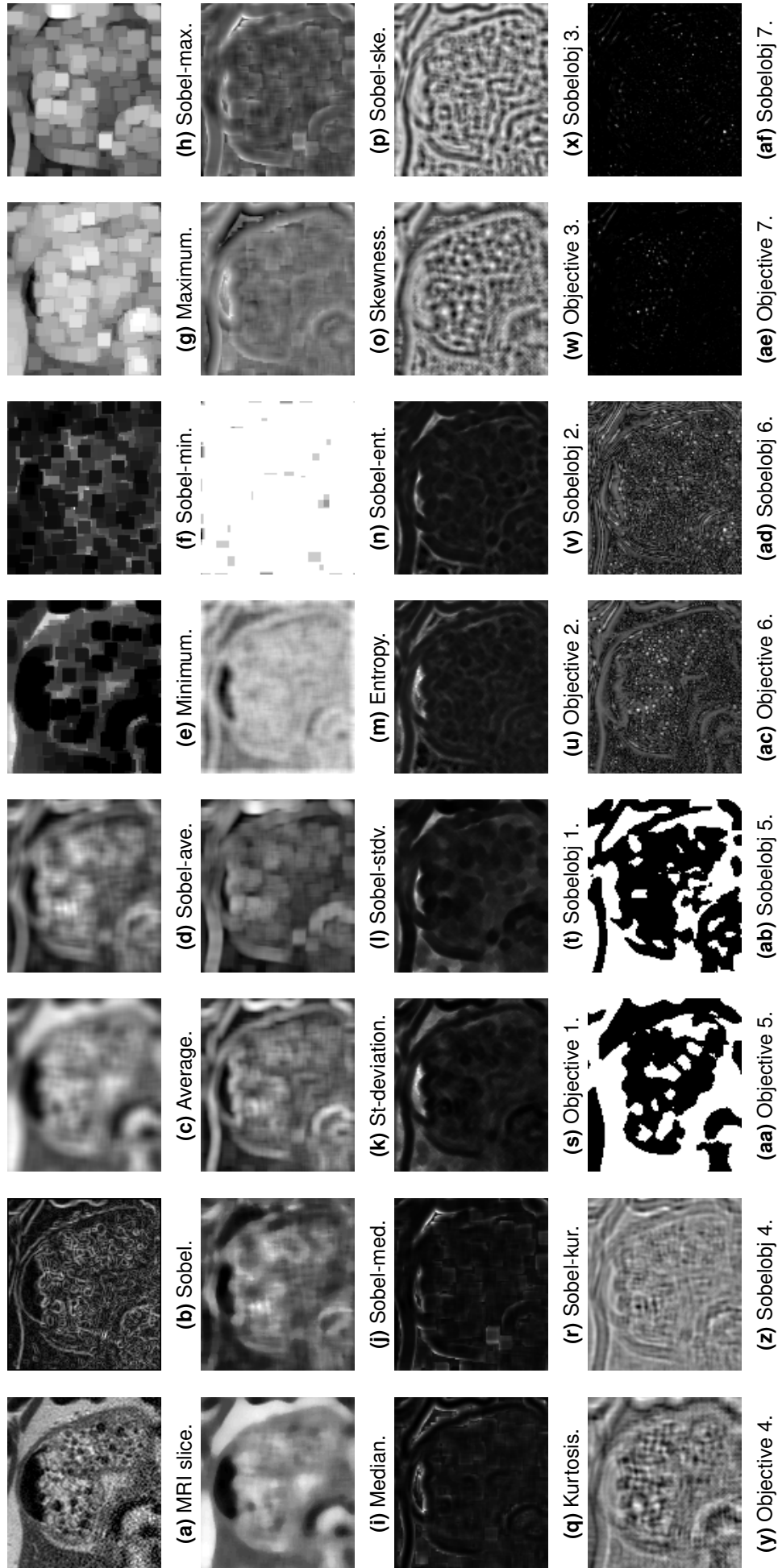


Figure 8.2: Image statistics. Sample from slice #17 of Dataset #5 for all image statistics and objective functions to describe the problem. Each image represents one statistic or one objective function.

gradient magnitude (see Figure 8.2b) (computed with the Sobel operator). Below, $ROI(I, x, y, n)$ is a small squared ROI from Image I , centred on the pixel x, y and the sides of the square have a length of n pixels. n is slightly larger than the diameter of a pea.

Average (μ): Gives the contribution for each pixel intensity (see Fig. 8.2c and Fig. 8.2d). The average is defined as:

$$\mu = \frac{\sum_{i=0}^n \sum_{j=0}^n ROI(I, x, y, n)(i, j)}{n \times n} \quad (8.1)$$

Minimum (min): Updates each pixel based on the minimum pixel surrounding “includes retaining the darker” (see Fig. 8.2e and see Fig. 8.2f). The minimum is defined as:

$$\min(ROI(I, x, y, n)) \quad (8.2)$$

Maximum (max): Updates each pixel based on the maximum pixel surrounding “includes retaining the lighter” (see Fig. 8.2g and Fig. 8.2h). The maximum is defined as:

$$\max(ROI(I, x, y, n)) \quad (8.3)$$

Median (med): Selects the middle-ranked value from a neighbourhood (see Fig. 8.2i and Fig. 8.2j). The median is defined as:

$$\text{med}(ROI(I, x, y, n)) = \begin{cases} \frac{X[\frac{n}{2}] + X[\frac{n+1}{2}]}{2} & \text{if } n \text{ is even.} \\ X[\frac{n+1}{2}] & \text{if } n \text{ is odd.} \end{cases} \quad (8.4)$$

where X is the sorted array of all the pixels of $ROI(I, x, y, n)$.

Standard deviation (σ): Quantifies dispersion of the local pixel intensities (see Fig. 8.2k and Fig. 8.2l). The standard deviation is defined as:

$$\sigma(I, x, y, n) = \sqrt{\frac{1}{n \times n} \sum_{i=0}^n \sum_{j=0}^n (ROI(I, x, y, n)(i, j) - \mu)^2} \quad (8.5)$$

Entropy (H): Measures the image data content “the degree of confusion or doubt in a system” (see Fig. 8.2m and Fig. 8.2n). Let (X, Y) be a discrete random variables from $ROI(I, x, y, n)$ with alphabet $(\mathcal{X}, \mathcal{Y})$ and probability mass function $P(x, y)$. The entropy is defined as [149]:

$$H(X, Y) = - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y) \log_2 P(x, y) \quad (8.6)$$

Skewness (S): Encodes the asymmetry of the distribution (see Fig. 8.2o and Fig. 8.2p). It depends on the normalised k -th central moment m_k , which is defined as:

$$m_k = \sum_j^n \sum_i^n \frac{(ROI(I, x, y, n)(i, j) - c)^k}{n \times n} \quad (8.7)$$

with c a constant. If $c = 0$, Eq. 8.7 computes “raw moments”. If $c = \mu$, then it computes the central moments (i.e. moments centred on the mean). The 1st moment is then equal to the variance. The 3rd moments relates to the Skewness:

$$S = \frac{m_3}{\sigma^3} \quad (8.8)$$

Kurtosis K : encodes the the sharpness of the distribution (see Fig. 8.2q and Fig. 8.2r). The 4rd moments relates to the Kurtosis:

$$K = \frac{m_4}{\sigma^4} \quad (8.9)$$

In addition, we propose specifically defined “objective functions” to encode a pea, **a relatively homogeneous circular region that is much darker than its surrounding**. For example, in this objective functions we will now consider circular regions of interest ($ROI_C(I, x, y, R)$) rather than squared ones, where R is the radius. The objective functions in this chapter are slightly different from the objective functions that used in the chapter 6. The main difference with chapter 6 is that Objective 1 was meant to be standard deviation, but there was a bug, which is now corrected.

Objective 1: Measures the inverse local pixel intensity homogeneity within a circular ROI $ROI_C(I, x, y, R)$. (see Fig. 8.2s and Fig. 8.2t).

$$obj1(I, x, y, R) = \frac{1}{\sigma(I, x, y, R)} \quad (8.10)$$

with $\sigma(I, x, y, R)$ the standard deviation of the pixel intensities within $ROI_C(I, x, y, R)$.

Objective 2: Compares Objective 1 with the inverse local pixel intensity standard deviation within a ring region of interest (ROI_R) whose inner radius is R and outer radius $R + extra$ with a few extra pixels. (see Fig. 8.2u and Fig. 8.2v).

$$obj2(I, x, y, R) = obj1(ROI_C(I, x, y, R)) - obj1(ROI_R(I, x, y, R, R + extra)) \quad (8.11)$$

Objective 3: Measures the mean value for the circular region of interests $ROI_C(I, x, y, R)$ with size n . (see Fig. 8.2w and Fig. 8.2x).

$$obj3(I, x, y, R) = \frac{\sum_{i=0}^n ROI_C(I, x, y, R)}{\pi R^2} \quad (8.12)$$

Objective 4: Compares Objective 3 with mean value within a ring region of interest (ROI_R) whose inner radius is R and outer radius $R + extra$ with extra pixel out of the radius. (see Fig. 8.2y and Fig. 8.2z).

$$obj4(I, x, y, R) = obj3(ROI_C(I, x, y, R)) - obj3(ROI_R(I, x, y, R, R + extra)) \quad (8.13)$$

Objective 5: Uses a threshold on mean value for the circular region of interests $ROI_C(I, x, y, R)$ with size n . The threshold is depending on the mean value for a ring region of interest (ROI_R). (see Fig. 8.2aa and Fig. 8.2ab).

$$obj5(I, x, y, R) = \begin{cases} 0 & \text{if } \mu(ROI_C(I, x, y, R)) \geq \mu(ROI_R(I, x, y, R, R + extra)). \\ 1 & \text{Otherwise.} \end{cases} \quad (8.14)$$

Objective 6: Rotates the circular region of interests $ROI_C(I, x, y, R)$ three times and then averages the ZNCC between the original circular ROI and the rotated ones. This is used to know if the ROI is rotationally symmetric. (see Fig. 8.2ac and Fig. 8.2ad).

$$obj6(I, x, y, R) = \frac{ZNCC(ROI_C(I, x, y, R), ROI_{rot}(I, x, y, R, 90))}{3} + \frac{ZNCC(ROI_C(I, x, y, R), ROI_{rot}(I, x, y, R, 180))}{3} + \frac{ZNCC(ROI_C(I, x, y, R), ROI_{rot}(I, x, y, R, 270))}{3} \quad (8.15)$$

where $ROI_{rot}(I, x, y, R, \alpha)$ is the circular ROI rotated by α degrees.

Objective 7: The same of Objective 6, but use of a product (see Fig. 8.2ae and Fig. 8.2af).

$$obj7(I, x, y, R) = ZNCC(ROI_C(I, x, y, R), ROI_{rot}(I, x, y, R, 90)) \times ZNCC(ROI_C(I, x, y, R), ROI_{rot}(I, x, y, R, 180)) \times ZNCC(ROI_C(I, x, y, R), ROI_{rot}(I, x, y, R, 270)) \quad (8.16)$$

The data is stored a CSV file, just as the same process in the chapter 7 We also record the voxel spacing as it differs across the datasets we were asked to analyse. For appropriateness with supervised learning, the data is then split into training and testing sets. We selected again the leave-one-out cross-validation where each volunteer dataset will be tested individually.

8.3 Training traditional classifiers

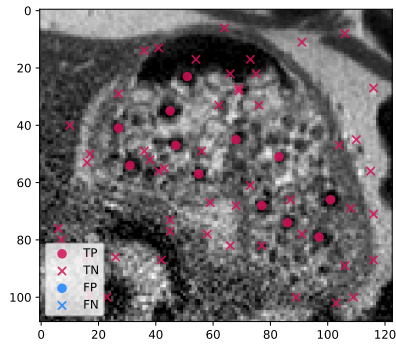
In this study, we investigate whether traditional classifiers are useful for distinguishing between the samples corresponding to peas and non-peas. For this purpose, we use some of the common classifiers implemented in SciPy [147] such as; Support Vector (SVC or SVM), AdaBoost, Decision tree, Gaussian Naive Bayes (GaussianNB), Gaussian process, Nearest neighbours, Quadratic discriminant analysis and Random forest [102]. The classifiers are trained using the training data. It is also important that they also cover the same area in mm^2 , which must be large enough to include a pea. The trained model is used on the test data to

provide predictions. Predictions in our case are binary, is a sample a pea or not? The outcome for a sample can be: TP , FN , FP and TN . For each test, the number of TP s, FN s, FP s, and TN s are recorded. To evaluate the performance of the classifiers, metrics such as accuracy, prediction, recall, and F_1 score are often used. To visualise the results we plot all the outcomes (TP as magenta depict circles, FN as blue crosses, FP as blue circles depict, and TN as magenta crosses) on the MRI slice to compare it with the selected data. Here, we use slice #17 of Dataset #5 (see Fig. 8.3) as a sample to display the final results of each classifier (see Fig. 8.4).

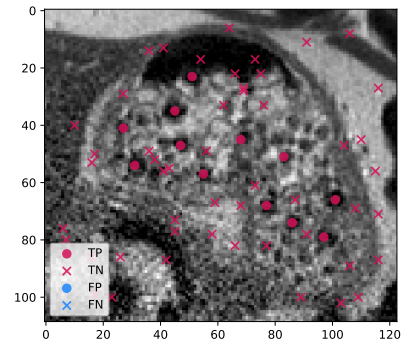


Figure 8.3: The MRI image for slice #17 of Dataset #5.

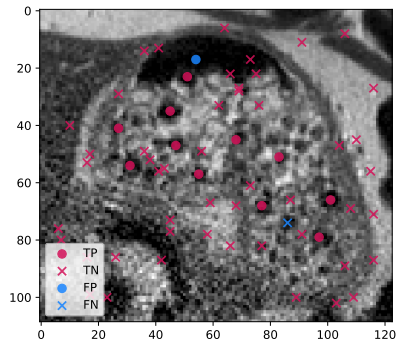
Table 8.1 shows the corresponding classification results in terms of accuracy, precision, recall, and F_1 score. Each value represents the overall values from all datasets for each classifier. The best evaluation result for each classifier is highlighted in red colour. From the table, we can see the accuracy and F_1 score are related to each other where accuracy is high F_1 score is high too. The SVC, Gaussian process and Random forest classifiers are performing very well in term of accuracy and F_1 score. They provide the highest accuracy and F_1 score, but the Gaussian process took more time to train. Figures 8.4a, 8.4e and 8.4h overlay the classification result on top of the original MRI slice. The AdaBoost, Decision tree, Nearest neighbours and Quadratic discriminant analysis classifiers were performing relatively well (see Figures 8.4b, 8.4c, 8.4f). and 8.4g). However, the GaussianNB classifier was not performing very well (see Fig. 8.4d).



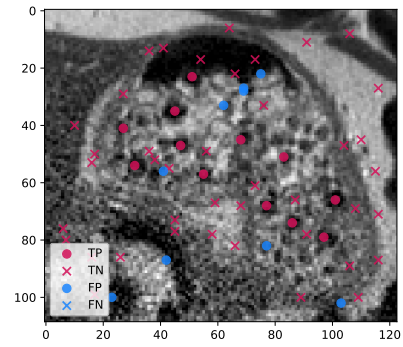
(a) SVC classifier.



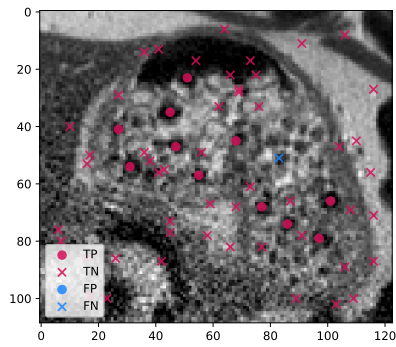
(b) Ada boost classifier.



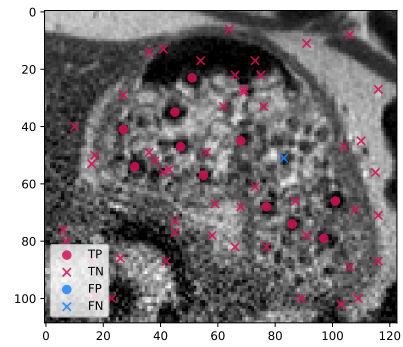
(c) Decision tree classifier.



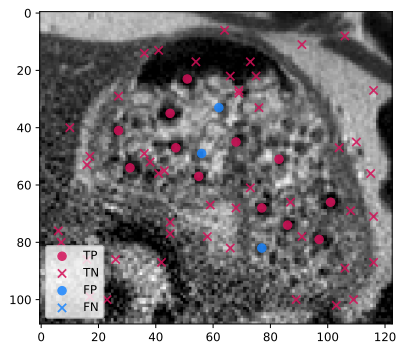
(d) Gaussian nb classifier.



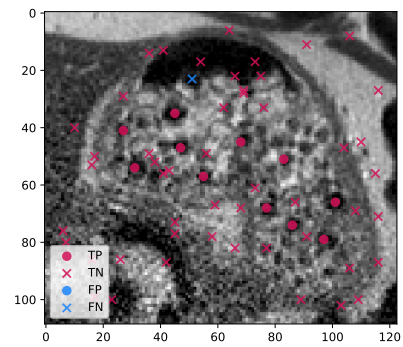
(e) Gaussian process classifier.



(f) Nearest neighbours classifier.



(g) Quadratic discriminant analysis classifier.



(h) Random forest classifier.

Figure 8.4: Traditional classifiers. Sample from slice #17 of Dataset #5 for all classifiers.

Table 8.1: Performance evaluation for all the traditional classifiers. Each value represents the overall values from all the datasets for each classifier. In other words, TP, TN FP and FN below include all the datasets. Accuracy, precision, recall and F1 score are computed using these values. The best performance for each column is highlighted in red colour.

model	runtime (CPU seconds)	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 Score
SVC	82	338	2230	20	63	0.97	0.94	0.84	0.89
ada boost	86	347	2205	45	54	0.96	0.89	0.87	0.88
decision tree	83	337	2181	69	64	0.95	0.83	0.84	0.84
gaussian nb	82	348	1986	264	53	0.88	0.57	0.87	0.69
gaussian process	126	339	2227	23	62	0.97	0.94	0.85	0.89
nearest neighbors	58	335	2220	30	66	0.96	0.92	0.84	0.87
quadratic discriminant analysis	82	356	2157	93	45	0.95	0.79	0.89	0.84
random forest	87	340	2231	19	61	0.97	0.95	0.85	0.89
Max						0.97	0.95	0.89	0.89

8.4 Training CNN model ResNet-50

Over the years deep CNNs have produced series of development in the field of image recognition and classification. However, training deeper CNNs may have some problems due to vanishing when the learning goes deeper and deeper, that will drive the top hidden layers into saturation and is becoming more severe [133]. Another network called “Residual learning” tries to fix this problems. The residual learning model tries to learn some residual, instead of trying to learn features. That means the input will add as a residue to the output of the weight layers and the activation is carried out [108]. ResNet-50 is currently a popular 50 layer Residual network. Our implementation is actually based on keras deep ResNet-50 [90].

We evaluated the ResNet-50 using various image sizes (32×32 , 64×64 , 128×128 and 256×256 pixels) and number of epochs (32, 64, 128, 256, 512 and 1024). In total 240 (10 datasets \times 4 image sizes \times 6 number of epochs) trainings and testings were performed to provide the best possible classification outcome. This is not feasible on a desktop computer because weeks of computations would be required. This is why the training and testing were performed on Supercomputing Wales’ supercomputer.

Table 8.2 shows the corresponding classification results in terms of accuracy, precision, recall, and F_1 score. The best results for each column are highlighted in red colour. In 3 last cases, when the image size and the number of epochs are both relatively large, it was not possible to perform the training due to the amount of computations that was required, even on a supercomputer. Training

time is significantly increases when the image size and the number of epochs are increasing, do not improve performance.

Table 8.2: Performance evaluation for ResNet-50 classifier. The best evaluation result is highlighted in red colour.

img size	epochs	runtime (CPU seconds)	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 Score
32	32	190880	337	2214	36	64	0.96	0.90	0.84	0.87
32	64	367680	312	2206	44	89	0.95	0.88	0.78	0.82
32	128	726840	349	2182	68	52	0.95	0.84	0.87	0.85
32	256	1440640	343	2206	44	58	0.96	0.89	0.86	0.87
32	512	2867640	361	2213	37	40	0.97	0.91	0.90	0.90
32	1024	5750760	354	2220	30	47	0.97	0.92	0.88	0.90
64	32	275040	332	2101	149	69	0.92	0.69	0.83	0.75
64	64	540480	355	2196	54	46	0.96	0.87	0.89	0.88
64	128	1055600	273	2207	43	128	0.94	0.86	0.68	0.76
64	256	2078680	352	2209	41	49	0.97	0.90	0.88	0.89
64	512	3920560	344	2216	34	57	0.97	0.91	0.86	0.88
64	1024	8315360	356	2214	36	45	0.97	0.91	0.89	0.90
128	32	600920	353	2162	88	48	0.95	0.80	0.88	0.84
128	64	1193800	265	2216	34	136	0.94	0.89	0.66	0.76
128	128	2344480	361	2198	52	40	0.97	0.87	0.90	0.89
128	256	4632640	357	2220	30	44	0.97	0.92	0.89	0.91
128	512	9156400	357	2210	40	44	0.97	0.90	0.89	0.89
128	1024	18629240	357	2211	39	44	0.97	0.90	0.89	0.90
256	32	2650440	272	2105	145	129	0.90	0.65	0.68	0.67
256	64	5020680	357	2207	43	44	0.97	0.89	0.89	0.89
256	128	9831360	342	2219	31	59	0.97	0.92	0.85	0.88
256	256	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
256	512	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
256	1024	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Max							0.97	0.92	0.90	0.91

From the table, we identify that 128×128 pixels and 256 epochs was performing well. It had the highest accuracy and F_1 score. Fig. 8.5 visualises the corresponding classifier outcome on top of the MRI slice. Circles in green TP s, crosses in green TN s, red circles depict FP s, and red crosses FN s.

8.5 Feature selection and classification using NSGA-II

From the above, there are 30 features. In this chapter we use NSGA-II as a classifier, that simultaneously minimises the classification error and the number of selected features. In some ways, it also performs some kind of feature selection as a weight will be applied on each feature. Feature selection ideally is found the minimally sized feature subset that is substantial and suitable to the objective concept. Feature selection can improve the classification accuracy and reduce the computational complexity of classification because it will decrease the use of non-effective features [40]. Many traditional feature selection approaches ignore

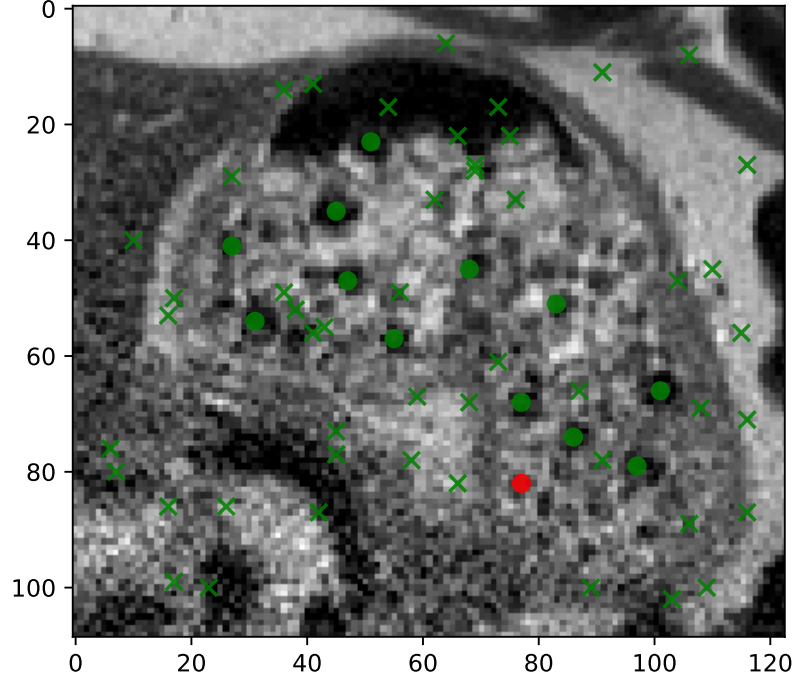


Figure 8.5: Visual representation of the ResNet-50 classifier the best result (image size 128×128 pixels and 256 epochs). Circles in green TPs, crosses in green TNs and red circles depict FPs. Sample from slice #17 of Dataset #5.

that there exist multiple optimal solutions in feature selection. Feature selection in evolutionary learning is a growing topic, with five dedicated presentations in the recent Evo* 2021 conference [75], [81], [109], [151]. Multi-objective optimisation for feature selection in classification is particularly relevant to our problem as there can be multiple different optimal feature subsets that achieve the same or similar classification performance, and this is the approach we adopted. Moreover, when using evolutionary multi-objective optimisation for feature selection, a crowding distance metric is typically used to play a role in environmental selection [151].

An individual \mathbf{I} is made of 30 weights (between 0 and 1), and 30×2 thresholds that is to say 90 floating point numbers per individual. $\mathbf{I}[i * 3]$ is the weight of the i^{th} feature, $\mathbf{I}[i * 3 + 1]$ is the lowest threshold of the i^{th} feature, and $\mathbf{I}[i * 3 + 2]$ is the highest threshold value. Below \mathbf{V} is the feature vector corresponding to a point of the image. For a given individual \mathbf{I} , the weight and thresholds are applied as follows to make a prediction ($\text{prediction}(\mathbf{V}, \mathbf{I})$):

$$prediction(\mathbf{V}, \mathbf{I}) = \sum_{i=0}^{i<30} \begin{cases} +\mathbf{I}[i * 3], & \text{if } \mathbf{I}[i * 3 + 1] \leq \mathbf{V}[i] \leq \mathbf{I}[i * 3 + 2] \\ -\mathbf{I}[i * 3], & \text{otherwise} \end{cases} \quad (8.17)$$

If $prediction(\mathbf{V}, \mathbf{I})$ is greater than or equal to 0, \mathbf{V} is the feature vector of a point that is considered as a pea. If $prediction(\mathbf{V}, \mathbf{I})$ is less than 0, then \mathbf{V} is the feature vector of a point that is considered as a non-pea.

For each individual, the predictions are computed for each sample of the training data. Again, we sort the predictions into TPs , TNs , FPs , and FNs . We derive them into 8 objectives that NSGA-II can minimise:

$$1 - \frac{TP}{TP + TN + FP + FN} \quad (8.18)$$

$$1 - \frac{TN}{TP + TN + FP + FN} \quad (8.19)$$

$$1 - \frac{FP}{TP + TN + FP + FN} \quad (8.20)$$

$$1 - \frac{FN}{TP + TN + FP + FN} \quad (8.21)$$

$$1 - Accuracy \quad (8.22)$$

$$1 - Precision \quad (8.23)$$

$$1 - Recall \quad (8.24)$$

$$1 - F1Score \quad (8.25)$$

Table 8.3 shows the corresponding classification results in terms of accuracy, precision, recall, and F_1 score. We can clearly identify, the classifier was performing well when the generation number is increasing. Population size did not affect the results that much. We select the result in generations: 64 and population size 1,024 as a best results in term of highest accuracy and F_1 score. Fig. 8.6 visualises the corresponding classifier outcome on top of the MRI slice. Circles in magenta TP , crosses in magenta TN , and blue crosses FN .

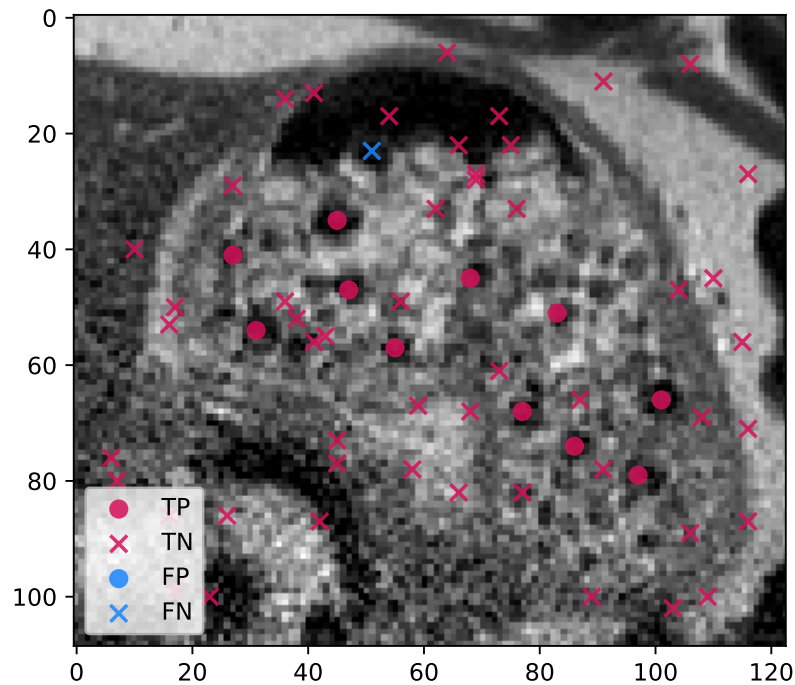


Figure 8.6: Visual representation of the best result of the NSGA-II based classifier (64 generations of a population made of 1,024 individuals). Circles in magenta TPs, crosses in magenta TNs, and blue crosses FNs. Sample from slice #17 of Dataset #5.

8.6 Automatic Keypoint Selection

We extracted the best classifier from the previous three sections in Table 8.4, namely Random forest, ResNet-50 (image size: 128, epochs: 256) and NSGA-II (generations: 64, population size 1,024). They are all equivalent in term of accuracy. NSGA-II is not as precise as the other two. It means that it generates more false

Table 8.3: Performance evaluation for the NSGA2-based classifier. The best performance for each column is highlighted in red colour

generations	pop-size	runtime (CPU seconds)	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 Score
4	4	4275	360	855	1395	41	0.46	0.21	0.90	0.33
4	8	5715	233	1341	909	168	0.59	0.20	0.58	0.30
4	16	8235	278	1561	689	123	0.69	0.29	0.69	0.41
4	32	13275	275	1705	545	126	0.75	0.34	0.69	0.45
4	64	23355	217	1697	553	184	0.72	0.28	0.54	0.37
4	128	42120	284	1799	451	117	0.79	0.39	0.71	0.50
4	256	80910	315	1685	565	86	0.75	0.36	0.79	0.49
4	512	157590	219	1914	336	182	0.80	0.39	0.55	0.46
4	1024	310725	252	1958	292	149	0.83	0.46	0.63	0.53
4	2048	611325	272	1951	299	129	0.84	0.48	0.68	0.56
8	4	5490	229	1583	667	172	0.68	0.26	0.57	0.35
8	8	8505	295	1660	590	106	0.74	0.33	0.74	0.46
8	16	12915	315	1682	568	86	0.75	0.36	0.79	0.49
8	32	22455	288	1926	324	113	0.84	0.47	0.72	0.57
8	64	41670	266	1975	275	135	0.85	0.49	0.66	0.56
8	128	86670	300	1927	323	101	0.84	0.48	0.75	0.59
8	256	171045	291	2054	196	110	0.88	0.60	0.73	0.66
8	512	307665	276	2012	238	125	0.86	0.54	0.69	0.60
8	1024	688725	295	2075	175	106	0.89	0.63	0.74	0.68
8	2048	1399995	277	2110	140	124	0.90	0.66	0.69	0.68
16	4	8730	313	1414	836	88	0.65	0.27	0.78	0.40
16	8	13725	291	1872	378	110	0.82	0.43	0.73	0.54
16	16	24345	264	1987	263	137	0.85	0.50	0.66	0.57
16	32	42030	277	2056	194	124	0.88	0.59	0.69	0.64
16	64	81945	318	2122	128	83	0.92	0.71	0.79	0.75
16	128	172755	314	2090	160	87	0.91	0.66	0.78	0.72
16	256	345285	298	2161	89	103	0.93	0.77	0.74	0.76
16	512	598860	310	2138	112	91	0.92	0.73	0.77	0.75
16	1024	1260855	309	2144	106	92	0.93	0.74	0.77	0.76
16	2048	2774565	329	2139	111	72	0.93	0.75	0.82	0.78
32	4	13680	219	1909	341	182	0.80	0.39	0.55	0.46
32	8	24255	319	2053	197	82	0.89	0.62	0.80	0.70
32	16	39870	302	2083	167	99	0.90	0.64	0.75	0.69
32	32	87795	310	2193	57	91	0.94	0.84	0.77	0.81
32	64	149670	326	2194	56	75	0.95	0.85	0.81	0.83
32	128	299070	335	2188	62	66	0.95	0.84	0.84	0.84
32	256	687195	336	2193	57	65	0.95	0.85	0.84	0.85
32	512	1372815	325	2195	55	76	0.95	0.86	0.81	0.83
32	1024	2466990	336	2195	55	65	0.95	0.86	0.84	0.85
32	2048	2990990	331	2191	59	70	0.95	0.85	0.83	0.84
64	4	23220	313	1888	362	88	0.83	0.46	0.78	0.58
64	8	40410	280	2104	146	121	0.90	0.66	0.70	0.68
64	16	79200	314	2158	92	87	0.93	0.77	0.78	0.78
64	32	153225	336	2205	45	65	0.96	0.88	0.84	0.86
64	64	346815	338	2201	49	63	0.96	0.87	0.84	0.86
64	128	604755	332	2205	45	69	0.96	0.88	0.83	0.85
64	256	1213650	349	2209	41	52	0.96	0.89	0.87	0.88
64	512	2727045	341	2196	54	60	0.96	0.86	0.85	0.86
64	1024	2220775	350	2208	42	51	0.96	0.89	0.87	0.88
64	2048	173070	8	147	3	3	0.96	0.73	0.73	0.73
128	4	46755	295	2074	176	106	0.89	0.63	0.74	0.68
128	8	80910	314	2182	68	87	0.94	0.82	0.78	0.80
128	16	173610	327	2200	50	74	0.95	0.87	0.82	0.84
128	32	287010	342	2198	52	59	0.96	0.87	0.85	0.86
128	64	683460	333	2201	49	68	0.96	0.87	0.83	0.85
128	128	1247940	345	2210	40	56	0.96	0.90	0.86	0.88
128	256	1794085	343	2203	47	58	0.96	0.88	0.86	0.87
128	512	1146643	344	2214	36	57	0.96	0.91	0.86	0.88
128	1024	73390	8	97	3	3	0.95	0.73	0.73	0.73
256	4	88560	317	2181	69	84	0.94	0.82	0.79	0.81
256	8	174510	312	2182	68	89	0.94	0.82	0.78	0.80
256	16	314100	329	2198	52	72	0.95	0.86	0.82	0.84
256	32	625455	342	2212	38	59	0.96	0.90	0.85	0.88
256	64	490920	342	2211	39	59	0.96	0.90	0.85	0.87
256	128	533117	347	2207	43	54	0.96	0.89	0.87	0.88
256	256	1002348	349	2208	42	52	0.96	0.89	0.87	0.88
Max							0.96	0.91	0.90	0.88

positives (FP), which is not necessarily a worry as they can be filtered out by the end-user or as in this pipeline by another classifier. They are all equivalent in term of recall and F1 score. The random forest is the fastest to train, by far, followed by NSGA-II. ResNet-50 is the slowest, by far, but overall probably the best in term of quality of the results.

Table 8.4: The best classifier from the traditional, ResNet-50 and NSGA-II classifiers

model	runtime (CPU seconds)	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 Score
random forest	87	340	2231	19	61	0.97	0.95	0.85	0.89
ResNet-50 (128, 256)	4632640	357	2220	30	44	0.97	0.92	0.89	0.91
NSGA2 (64, 1024)	2220775	350	2208	42	51	0.96	0.89	0.87	0.88

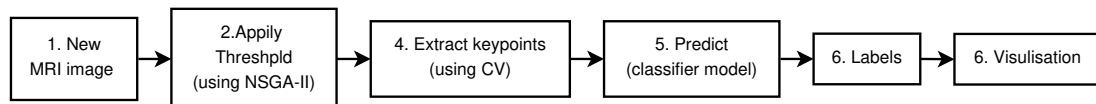


Figure 8.7: Overall flowchart of our steps to extract the keypoints.

Figure 8.7) explains the steps we followed to extract the keypoints. The advantage of NSGA-II over Random forest and ResNet-50 is that we can exploit its output (thresholds on selected features) to generate an image that can be filtered using computer vision techniques. Figure 8.8 is such an image. A threshold is then apply (see Figure 8.9) to extract the darkest regions in Figure 8.8. Islands are identified in the image resulting from the threshold filter (see green lines in Figure 8.10). The centroid of all these islands corresponds to keypoints (see red crosses in Figure 8.10). These keypoints are then labelled using our selected ‘best’ classifier: ResNet-50. Figure 8.11 shows the result of this fully-automatic pipeline on the same image.

8.7 Discussion and Conclusion

We have presented a fully-automatic segmentation of the peas using a combination of evolutionary computing, machine learning and computer vision techniques. The final results (Figure 8.11 and Table 8.5) look disappointing at first sight. However it is important to bear in mind some of the difficulties of this study:

- The problem can only be solved in 2D, not 3D, due to the image acquisition protocol used.

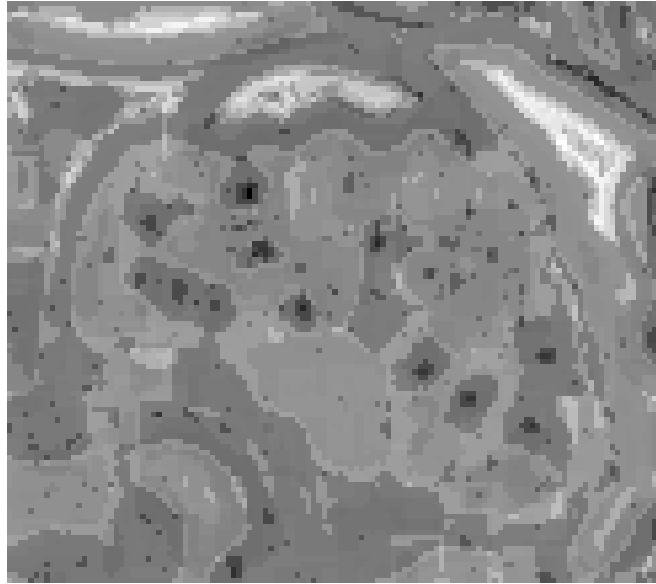


Figure 8.8: Samples from slice #17 of Dataset #5 for combining the features that selected using computer vision techniques.



Figure 8.9: Samples from slice #17 of Dataset #5 after apply a threshold.

Table 8.5: Classification results.

Dataset #	TP	TN	FP	FN	Accuracy	Precision	Recall	F_1 Score
1	4	236	13	5	0.93	0.24	0.44	0.31
2	0	81	6	0	0.93	0.0	inf	nan
3	89	597	105	5	0.86	0.46	0.95	0.62
4	77	355	46	0	0.9	0.63	1.0	0.77
5	79	1001	69	5	0.94	0.53	0.94	0.68
6	46	187	6	13	0.92	0.88	0.78	0.83
7	4	100	5	2	0.94	0.44	0.67	0.53
8	2	156	3	7	0.94	0.4	0.22	0.29
9	9	270	19	4	0.92	0.32	0.69	0.44
10	1	323	16	8	0.93	0.06	0.11	0.08
Overall								
10	311	3306	288	49	0.91	0.52	0.86	0.65

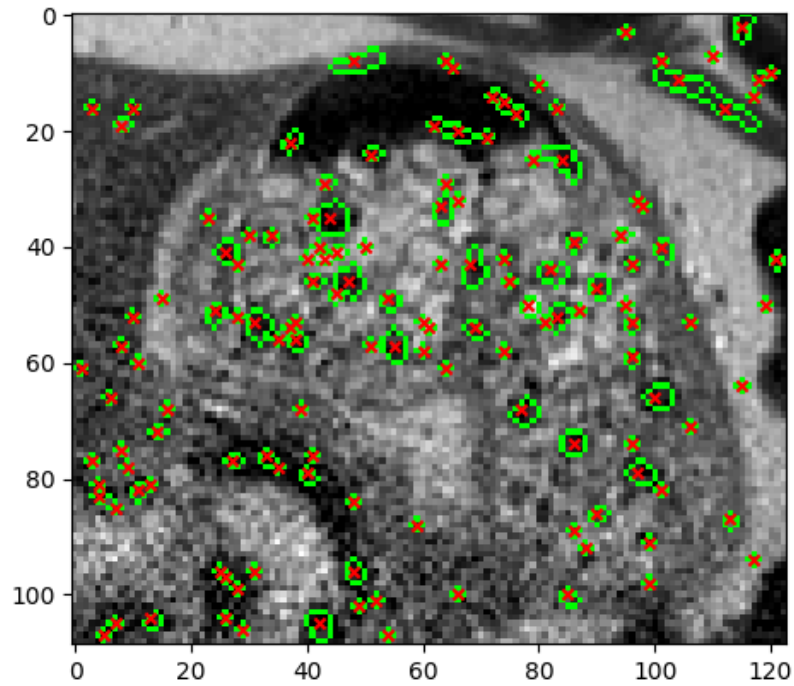


Figure 8.10: Samples from slice #17 of Dataset #5 for the filtering the selection data by NSGA-II.

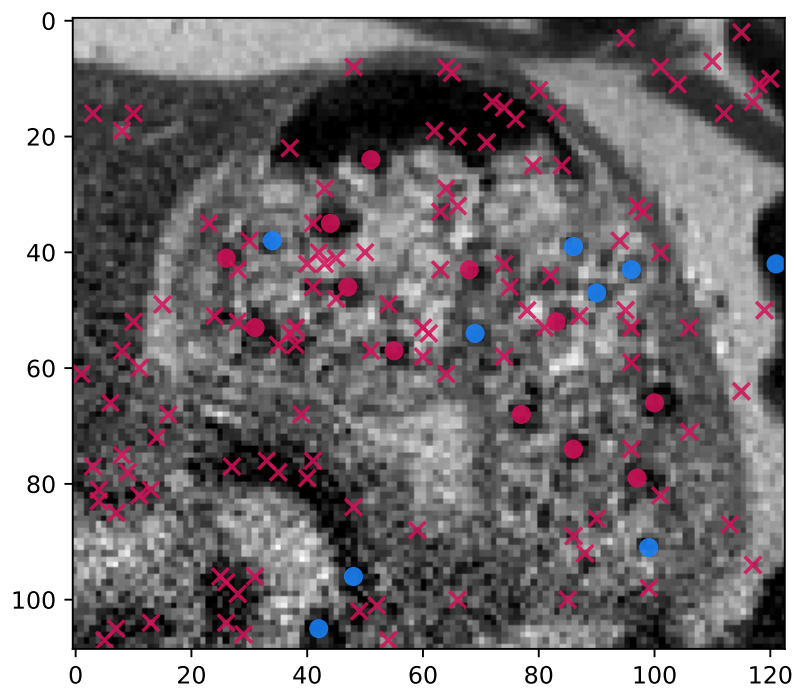


Figure 8.11: Samples for the final results from slice #17 of Dataset #5. Circles in magenta TPs, crosses in magenta TNs, and blue circles FPs

- Petit-pois were used in the end instead of garden peas, i.e. much smaller peas.
- Bread was used instead of pasta, making it harder to differentiate the peas from their background, even for Human beings.
- Poorer image resolution, again, making it harder to differentiate the peas from their background, even for Human beings.
- Limited amount of labelled data.

It is clear that our method produces a relatively high number of false positives. This actually indicates that our framework was able to replicate the same dilemma as what our collaborators experienced when they manually labelled the data: In many cases, Dr Évelyne Lutton and Prof François Boué were not fully sure of how to interpret the images. Some regions looked like peas, but they were not really sure. Also, they were led to assume that the presence of a grey point could be assimilated to statistical error Dr Lutton said:

The algorithm reproduces human evaluations quite well: TP correspond to peas that are quite evident while FP occur on locations where human judgement is not entirely sure due to the imprecision of the image. This is particularly evident on the second part of the dataset (DATASET 8 to DATASET 10) where resolution was poorer and peas less easy to detect due to experimental setup. The algorithm proves to be very useful in this case, as it highlights imprecise peas. In the Figure 8.12, after a second review by the experts, some peas locations proposed by the machine as FP were finally put in the set of peas (circled in yellow). This algorithm could be also used in an interactive manner to help a human expert in making decisions with uncertainty, in correcting some errors due to user fatigue and in progressively providing more and more precise learning sets.

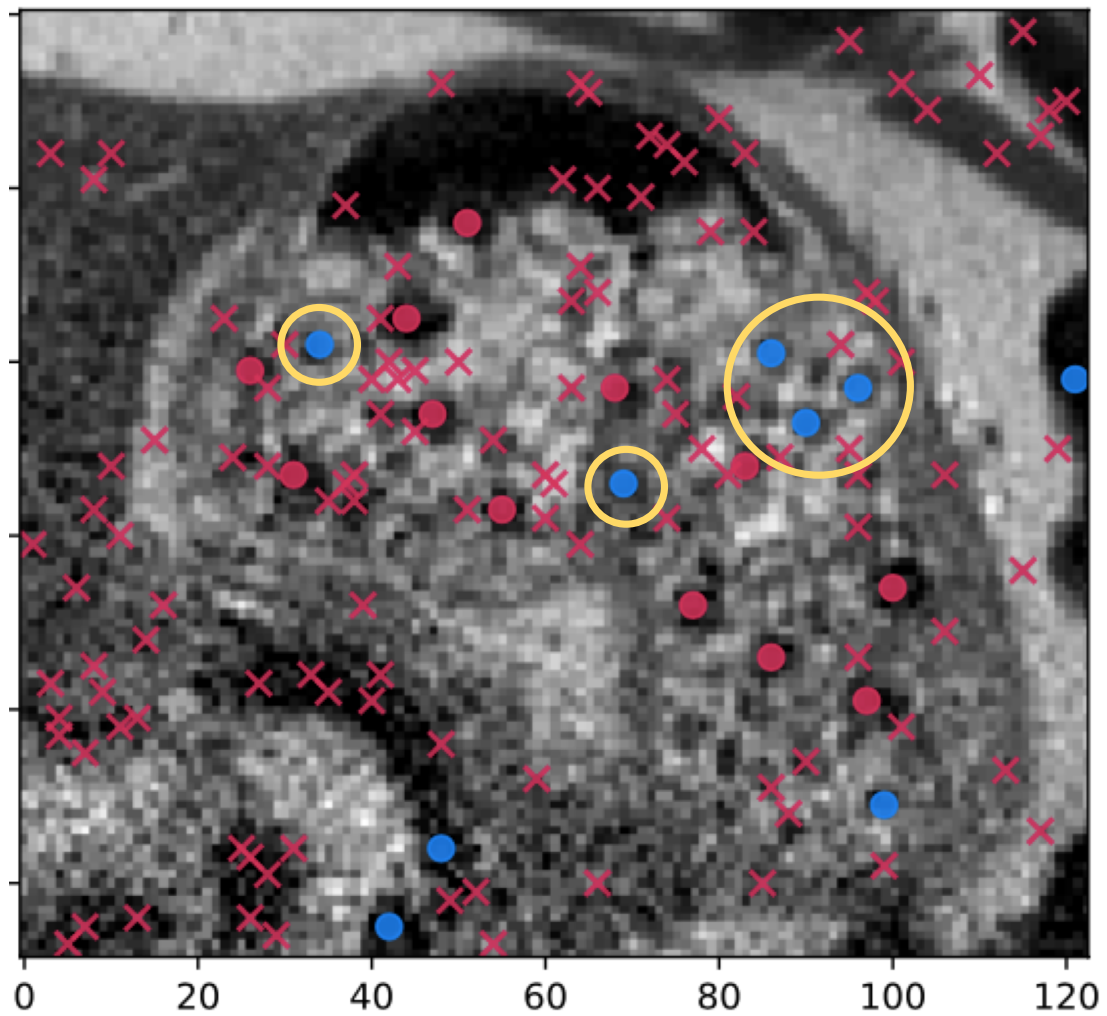


Figure 8.12: Final analysis from the experts. The visualisation shows the MRI image with the final results circles in magenta TPs, crosses in magenta TNs, and blue circles FPs. Yellow circles represent the experts confirmation

Chapter 9

Discussion and Conclusions

9.1 Discussion

Chapter 1 provides an introduction and outlines the hypothesis for this body of thesis. The hypothesis was:

For complex and large problems where traditional algorithms are not suitable, ad-hoc interactive visualisation can be deployed to considerably improve either i) an optimisation algorithm, or ii) the selection of the problem solution on the “Pareto” front to use as a solution.

In order to investigate this hypothesis fully, the aims and objectives of the project were:

1. Identify a problem or class of problems where traditional black-box optimisation approaches are not suitable. The study will illustrate how the FA compare in terms of efficiency and effectiveness to a few other traditional optimisation approaches.
2. Select two case studies to validate or invalidate our hypothesis, i) reconstruction in nuclear medicine (PET), and ii) segmentation of small regions in another type of medical images (here MRI).
3. Use interactive visualisation to analyse the internal data of FA to extract the best possible solution of the reconstructed PET image.

4. Modify the FA to speed-up the reconstruction process without loss of accuracy.
The two most common stopping criteria in EAs are i) the total number of generations, and ii) stagnation (no further significant improvement in terms of fitness value).
5. Analyse the content of the stomach from MRI using multi-objective FA to find small regions on this image.
6. Demonstrate that it is possible to train a CNN despite a very limited and imbalanced database of cases and using an interactive visualisation to find the best suitable combinations of hyper-parameters.
7. Use multi-objective optimisation (NSGA-II) as a classifier and compare it with common classifiers.

Chapter 2 presented an overview of the main scientific principle used in this thesis which are medical images and data visualisation. In order to identify gaps within current knowledge, a literature review of optimisation algorithms was undertaken in chapter 3. This explained the common traditional evolutionary algorithms (such as RCGA and PSO) and compared them with parisian evolution which is FA. Chapter 4 identified that FA is superior to other traditional evolutionary algorithms for solving some complex imaging problem (PET reconstruction and lamp problem).

From above, it has been demonstrated that the FA works well in some real-world problems. Therefore, we developed the implementation of FA using interactive visualisation. Chapter 5 relies on Parallel Coordinate Plot, scatterplot and image display. The visualisation helped to extract the final result in a reconstruction problem without need to wait to finish the optimisation loop while maintaining the image accuracy, smoothness, and reconstruction time. That led to reduce implementation time and extract a new stopping criterion. This work has been published in journal Genetic Programming and Evolvable Machines.

Chapter 6 represented a contribution with a large project focused on the understanding of the influence of food structure on digestion. We used MRI for capturing peas in the human stomach. For this purpose, we turned the Fly

Algorithm into a multi-objective cooperative-coevolution algorithm, and expert the results through an interaction/visualisation interface. This helps understanding the complex relationship between the objectives and extracting individuals of the Pareto front that correspond to peas. This work has been published and presented in conference on Parallel Problem Solving from Nature (PPSN XV) and conference on Computer Graphics and Visual Computing (CGVC).

In chapter 7 the experts provided us with a new datasets (10 Datasets) with different voxel spacing. We used supervised learning algorithms (CNNs) as a binary classification task to define a point on the image as peas or non-peas. We split the dataset into training and testing data and used leave-one-out cross-validation to training and testing. Also, we implemented the classifier with different images size and epochs as hyper-parameters. The results of this work show that the performance of machine learning algorithms was good, but it was not possible to distinguish which parameter combination provides the best results. Therefore, we proposed an interactive visualisation (Radar Chart) to identify which is the best combination should be used. This work has been published conference on Computer Graphics and Visual Computing (CGVC) .

To make the work more robust, in chapter 8 we decided to implement several machine learning classifiers to predict the pea positions (traditional classifiers and ResNet-50 classifier). Also, developed the multi-objective optimisation as classifier. We used the output of the multi-objective optimisation as an inputs to some computer vision techniques to extract thresholds on selected features and find the keypoints. Initially, we displayed the results to the experts and they confirmed it.

Throughout this thesis, we implemented global optimisation methods (PRS, SA, FA, and PSO) as open-source. The Python implementation of all these algorithms and the test data provide on GitHub (<https://github.com/Shatha1978/Optimisation-algorithm-examples>). Also, we design a cooperative PSO (coPSO), and a PSO-flavoured fly algorithm. We recommend the use InfoVis and data exploration to understand some of the behaviours of an FA to extract early best solution and find early stopping. Also, the use of InfoVis helps to understand the output of an FA implementation and enhancement these output to extract

the optimal solution. Finally, we proposed a framework to analyse the content of stomach (frozen green peas and pasta). The long-term objective of this study is to help to model food interaction with the human digestive system. This study helped the researchers, who work in studying food structure, to track and follow any type of food.

9.2 Limitations and future work

Some limitations exist in this study. The stomach is manually segmented in MRI volumes, which is time consuming and can only be performed by someone able to interpret MRI data. As with every supervised learning algorithm, the training and testing data must be manually labelled. Again, it is time consuming and can only be performed by the domain experts.

We proposed in this thesis a two-stage approach with 1) NSGA-II computer vision techniques for the keypoints detection and feature selection, and 2) the CNN for the classification. A more exhausted comparison with state-of-art algorithms such as R-CNN and YO-LO can be performed.

At first glance, it seems that there are a lot of false positives, as illustrated by Table 8.5 on Dataset#3 where 105 peas were detected. However, we know from the experimental protocol that only 20 peas were ingested (see Page 101 of expert's PhD student thesis [52]). We also know that successive slices along the z-axis of the Euclidean space were not acquired at the same time but with a significant time delay due to atom relaxation. These time delays may have introduced the possibilities for peas to slightly moved in the stomach, hence being detected several times at different locations. In other words, some (if not most) of these false positives may be true positives but at a different time. Future work needs to be carried out to validate that assumption. A PhD student (Mr Conor Spann) is currently working on the 4D analysis and visualisation of the detected peas.

9.3 Conclusion

The project described in this thesis aimed to demonstrate how interactive visualisation can help understand the behaviour of the algorithm (e.g. to improve it), and help choose a “good” solution. The following results were achieved throughout the duration of this project:

- Proved that the Fly Algorithm was better for solving complex problems. That was through compare the performance of the Fly Algorithm with another traditional optimisation methods.
- Validated our hypothesis through two case studies: i) reconstruction in nuclear medicine (PET), and ii) segmentation of small regions in another type of medical images (here MRI).
- Extracted the best possible solution of the reconstructed PET image by using a simple but effective visualisation.
- Modified the FA to speed-up the reconstruction process without loss of accuracy. The two most common stopping criteria in EAs are i) the total number of generations, and ii) stagnation (no further significant improvement in terms of fitness value).
- Segmented small regions (peas) on MRI by using multi-objective FA.
- Trained a CNN despite a very limited and imbalanced database of cases and used an interactive visualisation to find the best suitable combinations of hyper-parameters.
- Trained a NSGA-II as classifier and presented a fully-automatic segmentation of the peas using a combination of evolutionary computing, machine learning and computer vision techniques.

The results presented in chapters 3 to 6 demonstrated the performance of FA in two cases reconstruction (using reconstructed PET image and lamp problem)

and segmentation a small regions on MRI. The results presented in these chapters highlighted the FA was performing good and was superior to other traditional algorithms in construction problems. Also, the performance of FA as a multi-objective cooperative-coevolution algorithm to segment small region (peas) in MRI images. It extracted individuals of the Pareto front that correspond to peas.

Whilst, in chapters 7 and 8 started to change the direction from the optimisation method to the machine learning due to the labelled data provided from the experts. In chapters 7 we used CNN as a classifier with hyper-parameters (image sizes (16×16 , 32×32 , 64×64 and 128×128 pixels) and number of epochs (32, 64, 256, 1024, and 2048)). We utilised a fine data visualisation which is radar chart to find the suitable combination between the hyper-parameters to give the best solution. Whilst, chapter 8 present another classifier relied on the multi-objective optimisation algorithm namely NSGA-II and compared it with some traditional classifiers. The chapters presented the best results to recognise as much as from the peas.

Acronyms

<i>FN</i>	false negative
<i>FP</i>	false positive
<i>TN</i>	true negative
<i>TP</i>	true positive
3-D	three-dimensional
API	application programming interface
CCEA	Cooperative Co-evolution Algorithm
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CSV	Comma-Separated Values
CT	Computed Tomography
CV	Computer Vision
DNN	Deep Neural Network
DSSIM	structural dissimilarity
EA	Evolutionary Algorithm
EC	Evolutionary Computing
EM	Expectation-Maximization
EP	Evolutionary Programming
ES	Evolution Strategies
ET	emission tomography

FA	Fly Algorithm
GA	Genetic Algorithm
GIT	gastrointestinal tract
GMM	Gaussian Mixture Model
GPU	graphics processing unit
HOG	Histogram of oriented gradients
InfoVis	Information Visualisation
keV	kiloelectron volt
LB	lower bound
LOR	line of response
MAE	mean absolute error
mAP	mean Average Precision
ML	Machine Learning
MLEM	Maximum-Likelihood Expectation-Maximization
MRI	Magnetic Resonance Imaging
MSE	mean squared error
NaN	not a number
NN	Neural Network
OSEM	Ordered Subset Expectation-Maximization
PCP	Parallel Coordinate Plot
PET	Positron emission tomography
PRS	Pure Random Search

PSNR	peak signal-to-noise ratio
PSO	Particle Swarm Optimization
RCGA	Real-Coded Genetic Algorithm
ResNet	residual neural network
RMSE	root mean squared error
ROI	region of interest
SA	Simulated Annealing
SANS	small-angle neutron scattering
SAXS	small-angle X-ray scattering
SCW	Supercomputing Wales
SNR	signal-to-noise ratio
SPECT	Single-Photon Emission Computed Tomography
SSE	Sum of Squared Error
SSIM	structural similarity
SVG	Structured Vector Graphics
TSP	Travelling Salesman Problem
TV	total variation
UB	upper bound
VOC	PASCAL Visual Object Classes Challenge
ZNCC	zero-normalised cross-correlation

References

- [1] *100 results to understand the fate of food in the digestive tract*. [Online]. Available: <http://www.cepia.inra.fr/en/Research/The-fate-of-food-in-the-digestive-tract> (p. 8).
- [2] Z. A. Abbood, O. Amlal and F. P. Vidal, 'Evolutionary art using the fly algorithm,' in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, vol. 10199, Amsterdam, The Netherlands: Springer, Heidelberg, Apr. 2017, pp. 455–470. doi: 10.1007/978-3-319-55849-3_30 (p. 38).
- [3] Z. A. Abbood and F. P. Vidal, 'Basic, dual, adaptive, and directed mutation operators in the Fly algorithm,' in *Biennial International Conference on Artificial Evolution (EA-2017)*, Paris, France, Oct. 2017, pp. 106–119, ISBN: 978-2-9539267-7-4 (pp. 38, 81, 83).
- [4] —, 'Fly4Arts: Evolutionary digital art with the Fly algorithm,' in *Biennial International Conference on Artificial Evolution (EA-2017)*, Paris, France, Oct. 2017, p. 313 (p. 38).
- [5] —, 'Fly4Arts: Evolutionary digital art with the Fly algorithm,' *ISTE Arts & Science*, vol. 17-1, no. 1, pp. 11–16, 2017. doi: 10.21494/ISTE.OP.2017.0177 (p. 38).
- [6] F. Alabsi and R. Naoum, 'Comparison of selection methods and crossover operations using steady state genetic based intrusion detection system,' *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 7, pp. 1053–1058, 2012 (p. 31).
- [7] J. Albert, R. Weisell, W. T. Lee, D. Tomé, A. V. Kurpad and R. Uauy, 'Research Approaches and Methods for Evaluating the Protein Quality of Human Foods Proposed by an FAO Expert Working Group in 2014,' *The Journal of Nutrition*, vol. 146, no. 5, pp. 929–932, Apr. 2016. doi: 10.3945/jn.115.222109 (p. 8).

- [8] Z. Ali Abbood, J. Lavauzelle, É. Lutton, J.-M. Rocchisani, J. Louchet and F. P. Vidal, 'Voxelisation in the 3-D fly algorithm for PET,' *Swarm and Evolutionary Computation*, vol. 36, pp. 91–105, Oct. 2017. doi: 10.1016/j.swevo.2017.04.001 (pp. 36, 38, 81, 82).
- [9] Z. Ali Abbood and F. P. Vidal, 'Basic, dual, adaptive, and directed mutation operators in the fly algorithm,' in *Artificial Evolution*, E. Lutton, P. Legrand, P. Parrend, N. Monmarché and M. Schoenauer, Eds., Cham: Springer International Publishing, 2018, pp. 100–114, ISBN: 978-3-319-78133-4. doi: 10.1007/978-3-319-78133-4_8 (pp. 43, 53, 55).
- [10] P. S. Andrews, 'An investigation into mutation operators for particle swarm optimization,' in *2006 IEEE International Conference on Evolutionary Computation*, Jul. 2006, pp. 1044–1051. doi: 10.1109/CEC.2006.1688424 (p. 31).
- [11] G. Andrienko and N. Andrienko, 'Parallel coordinates for exploring properties of subsets,' in *Coordinated and Multiple Views in Exploratory Visualization, 2004. Proceedings. Second International Conference On*, IEEE, 2004, pp. 93–104 (p. 17).
- [12] B. Bach, A. Spritzer, É. Lutton and J.-D. Fekete, 'Interactive Random Graph Generation with Evolutionary Algorithms,' in *International Symposium on Graph Drawing (GD 2012)*, W. Didimo and M. Patrignani, Eds., ser. Lecture Notes in Computer Science, vol. 7704, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 541–552, ISBN: 978-3-642-36763-2. doi: 10.1007/978-3-642-36763-2_48 (p. 15).
- [13] T. Baeck, D. B. Fogel and Z. Michalewicz, Eds., *Evolutionary Computation 1: Basic Algorithms and Operators*. Taylor & Francis, 2000, ISBN: 978-0750306645 (p. 22).
- [14] C. Bajaj and C. F. J. Wiley, 'Data visualization techniques,' 1998 (p. 16).
- [15] I. Bankman, *Handbook of medical image processing and analysis*. Elsevier, 2008 (p. 10).
- [16] O. Barrière and É. Lutton, 'Experimental analysis of a variable size mono-population cooperative-coevolution strategy,' in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 139–152, ISBN: 978-3-642-03211-0. doi: 10.1007/978-3-642-03211-0_12 (p. 36).

- [17] O. Barrière, É. Lutton and P. Willemin, 'Bayesian network structure learning using cooperative coevolution,' in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '09, Montreal, Québec, Canada: ACM, 2009, pp. 755–762, ISBN: 978-1-60558-325-9. doi: 10.1145/1569901.1570006 (p. 36).
- [18] O. Barrière, E. Lutton and P.-H. Willemin, 'Bayesian network structure learning using cooperative coevolution,' in *Genetic and Evolutionary Computation Conference (GECCO 2009)*, 2009 (p. 35).
- [19] D. Beasley, D. R. Bull and M. R. R., 'An overview of genetic algorithms: Part 1 , fundamentals 1 introduction 2 basic principles,' *University Computing*, vol. 15, no. 2, pp. 58–69, 1993 (p. 29).
- [20] V. Beiranvand, W. Hare and Y. Lucet, 'Best practices for comparing optimization algorithms,' *Optimization and Engineering*, vol. 18, no. 4, pp. 815–848, 2017. doi: 10.1007/s11081-017-9366-1 (p. 54).
- [21] E. Bisong, 'TensorFlow 2.0 and Keras,' in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Berkeley, CA: Apress, 2019, pp. 347–399. doi: 10.1007/978-1-4842-4470-8_30 (p. 118).
- [22] C. Blum and R. Battiti, *Grapheur - Business intelligence and analytics*, Accessed: 2018-06-19. [Online]. Available: <https://www.grapheur.com/> (p. 15).
- [23] J. Bongard and H. Lipson, 'Active coevolutionary learning of deterministic finite automata,' *Journal of Machine Learning Research*, vol. 6, pp. 1651–1678, 2005 (p. 35).
- [24] M. Bostock, *D3.js - Data Driven Documents - v. 5.5.0*, Accessed: 2018-06-19. [Online]. Available: <https://www.d3js.org/> (p. 17).
- [25] N. Boukhelifa and E. Lutton, 'Guest editorial: Special issue on genetic programming, evolutionary computation and visualization,' *Genetic Programming and Evolvable Machines*, vol. 19, no. 3, pp. 313–315, 2018. doi: 10.1007/s10710-018-9333-4 (p. 16).
- [26] N. Boukhelifa, A. TONDA, I.-C. Trelea, N. Perrot and É. LUTTON, 'Interactive knowledge integration in modelling for food sustainability : challenges and prospects,' in *ACM CHI Workshop on Designing Sustainable Food Systems*, NA, France, 2017, np. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-01604947> (p. 15).

- [27] A. M. Boumaza and J. Louchet, 'Mobile robot sensor fusion using flies,' in *Applications of Evolutionary Computing: EvoWorkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM Essex, UK, April 14–16, 2003 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 357–367, ISBN: 978-3-540-36605-8. doi: 10.1007/3-540-36605-9_33 (p. 36).
- [28] A. Bousquet, J. Louchet and J.-M. Rocchisani, 'Fully three-dimensional tomographic evolutionary reconstruction in nuclear medicine,' in *Artificial Evolution: 8th International Conference, Evolution Artificielle, EA 2007, Tours, France, October 29-31, 2007, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 4926, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 231–242, ISBN: 978-3-540-79305-2. doi: 10.1007/978-3-540-79305-2_20 (pp. 38, 50, 82).
- [29] K. Briechle and U. D. Hanebeck, 'Template matching using fast normalized cross correlation,' in *Optical Pattern Recognition XII*, D. P. Casasent and T.-H. Chao, Eds., International Society for Optics and Photonics, vol. 4387, Orlando, FL, United States: SPIE, 2001, pp. 95–102. doi: 10.1117/12.421129 (p. 68).
- [30] M. Brunato and R. Battiti, 'Grapheur: A Software Architecture for Reactive and Interactive Optimization,' in *Learning and Intelligent Optimization*, C. Blum and R. Battiti, Eds., ser. Lecture Notes in Computer Science, vol. 6073, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 232–246, ISBN: 978-3-642-13800-3 (p. 15).
- [31] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley, 2009, p. 348, ISBN: 978-0-470-51706-2 (p. 116).
- [32] G. Burgin, 'In memoriam - reflections on larry fogel at decision science, inc. (1965-1982),' *IEEE Computational Intelligence Magazine*, vol. 3, no. 1, pp. 69–72, Feb. 2008. doi: 10.1109/MCI.2007.913361 (p. 23).
- [33] F. Chollet et al., Keras, <https://keras.io>, 2015 (p. 121).
- [34] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, L. Tarbox and F. Prior, 'The cancer imaging archive (TCIA): Maintaining and operating a public information repository,' *Journal of Digital Imaging*, vol. 26, no. 6, pp. 1045–1057, Dec. 2013. doi: 10.1007/s10278-013-9622-7 (p. 11).

- [35] P. Collet and J. Louchet, 'Applications in the processing of signals and images,' in Wiley, 2010, ch. Chapter 2. Artificial Evolution and the Parisian Approach, pp. 15–44. doi: 10.1002/9780470611319.ch2 (p. 36).
- [36] P. Collet, E. Lutton, F. Raynal and M. Schoenauer, 'Polar ifs + parisian genetic programming = efficient ifs inverse problem solving,' *Genetic Programming and Evolvable Machines Journal*, vol. 1, no. 4, pp. 339–361, 2000, October (pp. 21, 35).
- [37] B. L. Cox, S. A. Graves, M. Farhoud, T. E. Barnhart, J. J. Jeffery, K. W. Eliceiri and R. J. Nickles, 'Development of a novel linearly-filled derenzo micropet phantom,' *American journal of nuclear medicine and molecular imaging*, vol. 6, no. 3, p. 199, 2016 (p. 48).
- [38] N. Dalal and B. Triggs, 'Histograms of oriented gradients for human detection,' in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893. doi: 10.1109/CVPR.2005.177 (p. 117).
- [39] C. Darwin, *1861_OriginNY_F382.pdf*, 1859 (p. 23).
- [40] M. Dash and H. Liu, 'Feature selection for classification,' *Intelligent data analysis*, vol. 1, no. 1-4, pp. 131–156, 1997 (p. 144).
- [41] E. D. De Jong, K. O. Stanley and R. P. Wiegand, 'Introductory tutorial on coevolution,' in *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation, London, United Kingdom, 2007* (p. 35).
- [42] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, 'A fast and elitist multi-objective genetic algorithm: NSGAII,' *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Aug. 2002. doi: 10.1109/4235.996017 (pp. 106, 108).
- [43] Z. Deuscher, J.-M. Bonny, F. Boué, V. Cheynier, S. Clerjon, M.-F. Devaux, J. Meneghel, F. Guillon, F. Jamme, S. L. Feunteun, S. Passot, M. Réfrégiers, H. Rogniaux, D. Ropartz, J. Thévenot, A. Vallverdu-Queralt and F. Canon, 'Selected case studies presenting advanced methodologies to study food and chemical industry materials: From the structural characterization of raw materials to the multisensory integration of food,' *Innovative Food Science & Emerging Technologies*, vol. 46, pp. 29–40, 2018, Food Science and

Technology in France: INRA's contribution to this area. doi: 10.1016/j.ifset.2017.10.003 (p. 8).

- [44] D. Devaraj and B. Yegnanarayana, 'Genetic-algorithm-based optimal power flow for security enhancement,' in *IEE Proceedings - Generation, Transmission and Distribution*, IET, 2005, pp. 899–905. doi: 10.1049/ip-gtd:20045234 (p. 31).
- [45] L. Donato-Capel, C. Garcia-Rodenas, E. Pouteau, U. Lehmann, S. Srichuwong, A. Erkner, E. Kolodziejczyk, E. Hughes, T. Wooster and L. Sagalowicz, 'Chapter 14 - technological means to modulate food digestion and physiological response,' in *Food Structures, Digestion and Health*, M. Boland, M. Golding and H. Singh, Eds., San Diego: Academic Press, 2014, pp. 389–422, ISBN: 978-0-12-404610-8. doi: 10.1016/B978-0-12-404610-8.00014-1 (p. 8).
- [46] Eberhart and Yuhui Shi, 'Particle swarm optimization: Developments, applications and resources,' in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 1, Seoul, South Korea, South Korea: IEEE, May 2001, 81–86 vol. 1. doi: 10.1109/CEC.2001.934374 (p. 32).
- [47] A. E. Eiben and J. E. Smith, 'What is an evolutionary algorithm?' In *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 25–48, ISBN: 978-3-662-44874-8. doi: 10.1007/978-3-662-44874-8_3 (p. 23).
- [48] E. Elbeltagi, T. Hegazy and D. Grierson, 'Comparison among five evolutionary-based optimization algorithms,' *Advanced engineering informatics*, vol. 19, no. 1, pp. 43–53, 2005 (p. 1).
- [49] F. H. Fahey, 'Data acquisition in PET imaging,' *Journal of Nuclear Medicine Technology*, vol. 30, no. 2, pp. 39–49, 2002 (p. 11).
- [50] M. Feurer and F. Hutter, 'Hyperparameter optimization,' in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 3–33, ISBN: 978-3-030-05318-5. doi: 10.1007/978-3-030-05318-5_1 (p. 118).
- [51] J. Floury, T. Bianchi, J. Thévenot, D. Dupont, F. Jamme, É. Lutton, M. Panouillé, F. Boué and S. L. Feunteun, 'Exploring the breakdown of dairy

- protein gels during in vitro gastric digestion using time-lapse synchrotron deep-uv fluorescence microscopy,' *Food Chemistry*, vol. 239, pp. 898–910, 2018. doi: 10.1016/j.foodchem.2017.07.023 (pp. 8, 100).
- [52] D. Freitas, 'Novel insights into starch digestion and the glycaemic response: From in vitro digestions to a human study using magnetic resonance imaging (mri),' Paris-Saclay University, AgroParisTech, Speciality: nutrition science, Doctoral School number 581: Agriculture Alimentation Biologie Environnement Santé (ABIES), Nov. 2018 (pp. 8, 157).
- [53] J. Gardner, **S. Al-Maliki**, É. Lutton, F. Boué and F. Vidal, 'Recognising specific foods in MRI scans using CNN and visualisation,' English, in *Proceedings of Computer Graphics and Visual Computing 2020 (CGVC 2020)*, The Eurographics Association, Jul. 2020, pp. 11–18, ISBN: 978-3-03868-122-9. doi: 10.2312/cgvc.20201145 (pp. 6, 114).
- [54] C. Gathercole and P. Ross, 'An adverse interaction between crossover and restricted tree depth in genetic programming,' in *Proceedings of the 1st Annual Conference on Genetic Programming*, Stanford, California: MIT Press, 1996, pp. 291–296, ISBN: 0262611279 (p. 29).
- [55] N. Glumov, E. Kolomiyetz and V. V. Sergeyev, 'Detection of objects on the image using a sliding window mode,' *Optics & Laser Technology*, vol. 27, no. 4, pp. 241–249, Aug. 1995, Optics and Image Processing in Russia. doi: 10.1016/0030-3992(95)93752-D (p. 116).
- [56] C. Goh, K. Tan, D. Liu and S. Chiam, 'A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design,' *European Journal of Operational Research*, vol. 202, no. 1, pp. 42–54, 2010. doi: <https://doi.org/10.1016/j.ejor.2009.05.005> (p. 21).
- [57] M. Gong and Y. Y-H., 'Multi-resolution stereo matching using genetic algorithm,' *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pp. 21–29, 2001. doi: 10.1109/SMBV.2001.988759 (p. 25).
- [58] C. C. Gray, **S. Al-Maliki** and F. P. Vidal, 'Data exploration in evolutionary reconstruction of pet images,' *Genetic Programming and Evolvable Machines*, vol. 19, no. 3, pp. 391–419, Sep. 2018, ISSN: 1573-7632. doi: 10.1007/s10710-018-9330-7 (pp. 6, 12, 76, 79).

- [59] Z. Halim and T. Muhammad, 'Quantifying and optimizing visualization: An evolutionary computing-based approach,' *Information Sciences*, vol. 385-386, no. Supplement C, pp. 284–313, 2017. doi: 10.1016/j.ins.2016.12.035 (p. 15).
- [60] A. Hanbury, 'Constructing cylindrical coordinate colour spaces,' *Pattern Recognition Letters*, vol. 29, no. 4, pp. 494–500, 2008 (p. 84).
- [61] X. He, D. Cai, Y. Shao, H. Bao and J. Han, 'Laplacian regularized gaussian mixture model for data clustering,' *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1406–1418, 2010 (p. 109).
- [62] J. Heer and B. Shneiderman, 'Interactive dynamics for visual analysis,' *Queue*, vol. 10, no. 2, p. 30, 2012 (p. 15).
- [63] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992, ISBN: 0262082136 (p. 23).
- [64] E. S. H. Hou, N. Ansari and Hong Ren, 'A genetic algorithm for multiprocessor scheduling,' *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 2, pp. 113–120, Feb. 1994. doi: 10.1109/71.265940 (p. 28).
- [65] H. M. Hudson and R. S. Larkin, 'Accelerated image reconstruction using ordered subsets of projection data,' *IEEE Transactions on Medical Imaging*, vol. 13, no. 4, pp. 601–609, Dec. 1994. doi: 10.1109/42.363108 (p. 46).
- [66] A. Inselberg, 'The plane with parallel coordinates,' *The Visual Computer*, vol. 1, no. 2, pp. 69–91, Aug. 1985. doi: 10.1007/BF01898350 (pp. 17, 80).
- [67] K. Jebari, 'Selection methods for genetic algorithms,' *International Journal of Emerging Sciences*, vol. 3, pp. 333–344, Dec. 2013 (p. 27).
- [68] V. Kachitvichyanukul, 'Comparison of three evolutionary algorithms: Ga, pso, and de,' *Industrial Engineering and Management Systems*, vol. 12, pp. 215–223, Sep. 2012. doi: 10.7232/iems.2012.11.3.215 (p. 40).
- [69] D. Karaboga and B. Akay, 'A survey: Algorithms simulating bee swarm intelligence,' *Artif. Intell. Rev.*, vol. 31, no. 1-4, pp. 61–85, Jun. 2009. doi: 10.1007/s10462-009-9127-4 (p. 32).
- [70] M. Karimi and B. Karimi, 'Linear and conic scalarizations for obtaining properly efficient solutions in multiobjective optimization,' *Mathematical*

Sciences, vol. 11, no. 4, pp. 319–325, 2017. doi: 10.1007/s40096-017-0234-0 (p. 106).

- [71] B. Kaufmann, J. Louchet and E. Lutton, 'Hand posture recognition using real-time artificial evolution,' in *Evolutionary Computation in Image Analysis and Signal Processing, EvoApplications 2010, Part I, LNCS 6024, C. Di Chio et al. (Eds.)*, 7th - 9th April, Istanbul Technical University, Istanbul, Turkey, Springer, Apr. 2010, pp. 251–260 (p. 36).
- [72] J. Kennedy and R. Eberhart, 'Particle swarm optimization,' in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Perth, WA, Australia, Australia: IEEE, Nov. 1995, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968 (p. 32).
- [73] A. Kerren and T. Egger, 'EAVis: A visualization tool for evolutionary algorithms,' in *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, Sep. 2005, pp. 299–301. doi: 10.1109/VLHCC.2005.33 (p. 15).
- [74] V. S. Khoo, D. P. Dearnaley, D. J. Finnigan, A. Padhani, S. F. Tanner and M. O. Leach, 'Magnetic resonance imaging (MRI): Considerations and applications in radiotherapy treatment planning,' *Radiotherapy and Oncology*, vol. 42, no. 1, pp. 1–15, 1997. doi: 10.1016/S0167-8140(96)01866-X (p. 13).
- [75] R. A. Khurma, K. E. Sabri, P. A. Castillo and I. Aljarah, 'Salp swarm optimization search based feature selection for enhanced phishing websites detection,' in *Applications of Evolutionary Computation*, Cham: Springer International Publishing, 2021, pp. 146–161. doi: https://doi.org/10.1007/978-3-030-72699-7_10 (p. 145).
- [76] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks,' in *Advances in neural information processing systems*, 2012, pp. 1097–1105 (p. 118).
- [77] Y. Landrin-Schweitzer, P. Collet and É. Lutton, 'Introducing lateral thinking in search engines,' *Genetic Programming and Evolvable Machines*, vol. 7, no. 1, pp. 9–31, Mar. 2006. doi: 10.1007/s10710-006-7008-z (p. 36).
- [78] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, 'Gradient-based learning applied to document recognition,' *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791 (p. 116).

- [79] B. Lee, P. Isenberg, N. H. Riche and S. Carpendale, 'Beyond mouse and keyboard: Expanding design considerations for information visualization interactions,' *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2689–2698, 2012 (p. 15).
- [80] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, *Microsoft coco: Common objects in context*, 2014 (p. 117).
- [81] A. Lopez-Rincon, D. S. Roozendaal, H. M. Spierenburg, A. L. Holm, R. Metcalf, P. Perez-Pardo, A. D. Kraneveld and A. Tonda, 'Modelling asthma patients' responsiveness to treatment using feature selection and evolutionary computation,' in *Applications of Evolutionary Computation*, Cham: Springer International Publishing, 2021, pp. 359–372. doi: https://doi.org/10.1007/978-3-030-72699-7_23 (p. 145).
- [82] J. Louchet, 'Stereo analysis using individual evolution strategy,' in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 1, 2000, 908–911 vol.1. doi: [10.1109/ICPR.2000.905580](https://doi.org/10.1109/ICPR.2000.905580) (pp. 2, 36, 37, 39).
- [83] —, 'Using an individual evolution strategy for stereovision,' *Genetic Programming and Evolvable Machines*, vol. 2, no. 2, pp. 101–109, Jun. 2001. doi: [10.1023/A:1011544128842](https://doi.org/10.1023/A:1011544128842) (pp. 37, 39).
- [84] J. Louchet, M. Guyon, M.-J. Lesot and A. Boumaza, 'Dynamic flies: A new pattern recognition tool applied to stereo sequence processing,' *Pattern Recognition Letters*, vol. 23, no. 1, pp. 335–345, 2002. doi: [10.1016/S0167-8655\(01\)00129-5](https://doi.org/10.1016/S0167-8655(01)00129-5) (pp. 37, 39, 43, 55).
- [85] E. Lutton, N. Perrot and A. Tonda, 'Model analysis and visualization,' in *Evolutionary Algorithms for Food Science and Technology*, John Wiley & Sons, Inc., 2016, pp. 33–55, ISBN: 9781119136828. doi: [10.1002/9781119136828.ch3](https://doi.org/10.1002/9781119136828.ch3) (p. 15).
- [86] E. Lutton, J. Thevenot, S. Le Feunteun, J. Floury, M. Panouille, D. Dupont, P. Roblin and F. Boue, 'Evolution of food gel structures during simulated gastro-intestinal digestion using Small Angle Scattering at SOLEIL synchrotron,' in *5th International Conference on Food Digestion*, Poster, Apr. 2017. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-01581553> (pp. 8, 100).

- [87] J. F. A. Madeira, H. L. Pina and H. C. Rodrigues, 'Ga topology optimization using random keys for tree encoding of structures,' *Structural and Multidisciplinary Optimization*, vol. 40, no. 1, p. 227, Feb. 2009. doi: 10.1007/s00158-008-0353-1 (p. 25).
- [88] R. Malhotra, N. Singh and Y. Singh, 'Genetic algorithms : Concepts , design for optimisation of process controllers,' *Computer and Information Science*, vol. 4, no. 2, pp. 39–54, 2011 (p. 29).
- [89] A. R. Martin and M. O. Ward, 'High dimensional brushing for interactive exploration of multivariate data,' in *Proceedings of the 6th Conference on Visualization'95*, IEEE Computer Society, 1995, p. 271 (pp. 19, 80).
- [90] Martin Abadi, Ashish Agarwal, Paul Barham et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: <https://www.tensorflow.org/> (pp. 121, 143).
- [91] G. Masselli and G. Gualdi, 'Ct and mr enterography in evaluating small bowel diseases: When to use which modality?' *Abdominal Imaging*, vol. 38, no. 2, pp. 249–259, Apr. 2013. doi: 10.1007/s00261-012-9961-8 (p. 100).
- [92] R. K. McConnell, *Method of and apparatus for pattern recognition*, US Patent 4,567,610, Jan. 1986 (p. 116).
- [93] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research & Management Science. Springer US, 1998, vol. 12, ISBN: 978-0-7923-8278-2. doi: 10.1007/978-1-4615-5563-6 (p. 106).
- [94] D. Mindrila, D. Ph, P. Balentyne and M. Ed, 'Scatterplots and Correlation,' 2013 (p. 17).
- [95] H. Mosley and A. Mayer, 'Benchmarking national labour market performance: A radar chart approach,' eng, Berlin, WZB Discussion Paper FS I 99-202, 1999. [Online]. Available: <https://www.econstor.eu/bitstream/10419/43952/1/301154597.pdf> (p. 20).
- [96] P. Murugan, S. Kannan and S. Baskar, 'Nsga-ii algorithm for multi-objective generation expansion planning problem,' *Electric Power Systems Research*, vol. 79, no. 4, pp. 622–628, 2009. doi: 10.1016/j.epsr.2008.09.011 (p. 107).

- [97] A. Nagpal and G. Gabrani, 'Python for data analytics, scientific and technical applications,' in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 140–145. doi: 10.1109/AICAI.2019.8701341 (p. 17).
- [98] G. Ochoa, E. Lutton and E. K. Burke, 'Cooperative royal road functions,' in *Evolution Artificielle, Tours, France, October 29-31, 2007* (p. 35).
- [99] J. Padma and M. Sailaja, 'SELECTION SCHEMES USED IN GENETIC ALGORITHM FOR ATM,' vol. 5, no. 3, pp. 1687–1690, 2018 (pp. 26, 27).
- [100] J. Pasquier, A. Brûlet, A. Boire, F. Jamme, J. Perez, T. Bizien, E. Lutton and F. Boué, 'Monitoring food structure during digestion using small-angle scattering and imaging techniques,' *Coll.Surf.A*, 2019, Accepted. doi: 10.1016/j.colsurfa.2019.02.059 (p. 8).
- [101] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, 'Automatic differentiation in PyTorch,' in *Neural Information Processing Systems 30 (NIPS 2017)*, 2017 (p. 118).
- [102] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, 'Scikit-learn: Machine learning in Python,' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011 (p. 140).
- [103] H. Pohlheim, 'Visualization of evolutionary algorithms - set of standard techniques and multidimensional visualization,' in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, ser. GECCO'99, Orlando, Florida: Morgan Kaufmann Publishers Inc., 1999, pp. 533–540, ISBN: 1-55860-611-4 (p. 15).
- [104] R. Poli, J. Kennedy and T. Blackwell, 'Particle swarm optimization,' *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, Jun. 2007. doi: 10.1007/s11721-007-0002-0 (p. 33).
- [105] H. R. Qodmanan, M. Nasiri and B. Minaei-Bidgoli, 'Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence,' *Expert Systems with Applications*, vol. 38, no. 1, pp. 288–298, 2011. doi: <https://doi.org/10.1016/j.eswa.2010.06.060> (p. 25).
- [106] K. RamaKalyani and D. Umadevi, 'Fraud detection of credit card payment system by genetic algorithm,' 2012 (p. 26).

- [107] I. Rechenberg, U. Küppers, A. Scheel, C. Mattheck and L. Harzheim, 'Evolution und optimierung,' in *Bionik: Grundlagen und Beispiele für Ingenieure und Naturwissenschaftler*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 245–269, ISBN: 978-3-662-06114-5. doi: 10.1007/978-3-662-06114-5_14 (p. 23).
- [108] A. S. B. Reddy and D. S. Juliet, 'Transfer learning with resnet-50 for malaria cell-image classification,' in *2019 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, 2019, pp. 0945–0949. doi: 10.1109/ICCSP.2019.8697909 (p. 143).
- [109] Q. Renau, J. Dreo, C. Doerr and B. Doerr, 'Towards Explainable Exploratory Landscape Analysis: Extreme Feature Selection for Classifying BBOB Functions,' in *Applications of Evolutionary Computation*, Cham: Springer International Publishing, 2021, pp. 17–33. doi: https://doi.org/10.1007/978-3-030-72699-7_2 (p. 145).
- [110] T. Reto, S. Andreas, W. Dominik, G. Oliver, B. Peter, F. Michael and S. Werner, 'Gastric motor function and emptying in the right decubitus and seated body position as assessed by magnetic resonance imaging,' *Journal of Magnetic Resonance Imaging*, vol. 23, no. 3, pp. 331–338, Feb. 2006. doi: 10.1002/jmri.20507 (p. 100).
- [111] S. Ribeca, *The Data Visualisation Catalogue*, Accessed: 2018-06-19. [Online]. Available: <https://www.datavizcatalogue.com> (p. 84).
- [112] J. C. Roberts, 'State of the art: Coordinated & multiple views in exploratory visualization,' in *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV'07. Fifth International Conference on*, IEEE, 2007, pp. 61–71 (p. 15).
- [113] M. Rudwaleit, A.-G. Jurik, K. A. Hermann, R. Landewé, D. van der Heijde, X. Baraliakos, H. Marzo-Ortega, M. Østergaard, J. Braun and J. Sieper, 'Defining active sacroiliitis on magnetic resonance imaging (MRI) for classification of axial spondyloarthritis: A consensual approach by the ASAS/OMERACT MRI group,' *Annals of the rheumatic diseases*, vol. 68, no. 10, pp. 1520–1527, 2009. doi: 10.1136/ard.2009.110767 (p. 13).
- [114] **S. Al-Maliki**, É. Lutton, F. Boué and F. Vidal, 'MRI gastric images processing using a multiobjective fly algorithm,' English, in *Fifteenth International Conference on Parallel Problem Solving from Nature (PPSN*

- XV), ser. Workshop on Evolutionary Machine Learning (EvoML), Sep. 2018. [Online]. Available: <http://ppsn2018.dei.uc.pt/index.php/workshops/> (pp. 6, 9, 100).
- [115] **S. Al-Maliki**, É. Lutton, F. Boué and F. P. Vidal, 'Evolutionary Interactive Analysis of MRI Gastric Images Using a Multiobjective Cooperative-coevolution Scheme,' in *Computer Graphics and Visual Computing (CGVC)*, The Eurographics Association, 2018, ISBN: 978-3-03868-071-0. doi: 10.2312/cgvc.20181216 (pp. 6, 9, 100, 132).
 - [116] **S. Al-Maliki** and F. P. Vidal, 'Evolutionary interactive analysis of MRI gastric images,' in *First CoESE PhD Conference*, Bangor, UK, Jan. 2019 (pp. 6, 100).
 - [117] M. Safe, J. Carballido, I. Ponzoni and N. Brignole, 'On stopping criteria for genetic algorithms,' in *Advances in Artificial Intelligence – SBIA 2004*, A. L. C. Bazzan and S. Labidi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 405–413, ISBN: 978-3-540-28645-5 (p. 25).
 - [118] N. Saini, 'Review of Selection Methods in Genetic Algorithms,' vol. 6, no. 12, pp. 23 261–23 263, 2017. doi: 10.18535/ijecs/v6i12.04 (p. 26).
 - [119] E. Sanchez, G. Squillero and A. Tonda, 'Group evolution: Emerging synergy through a coordinated effort,' Jul. 2011, pp. 2662–2668. doi: 10.1109/CEC.2011.5949951 (p. 44).
 - [120] E. Sapin, J. Louchet and E. Lutton, 'The fly algorithm revisited - adaptation to CMOS image sensors,' in *IJCCI*, 2009, pp. 224–229 (pp. 37, 38).
 - [121] H.-P. Schwefel, 'Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik,' Dipl.-Ing. Thesis, Technical University of Berlin, Hermann Föttinger–Institute for Fluid Dynamics, Mar. 1965 (p. 23).
 - [122] W. Schwizer, A. Steingoetter and M. Fox, 'Magnetic resonance imaging for the assessment of gastrointestinal function,' *Scandinavian Journal of Gastroenterology*, vol. 41, no. 11, pp. 1245–1260, 2006. doi: 10.1080/00365520600827188 (p. 100).
 - [123] P. Sharma, A. Wadhwa and M. Komal, 'Analysis of selection schemes for solving an optimization problem in genetic algorithm,' *International Journal of Computer Applications*, vol. 93, pp. 1–3, May 2014. doi: 10.5120/16256-5714 (p. 27).

- [124] L. A. Shepp and Y. Vardi, 'Maximum likelihood reconstruction for emission tomography,' *IEEE Transactions on Medical Imaging*, vol. 1, no. 2, pp. 113–122, Oct. 1982. doi: 10.1109/TMI.1982.4307558 (p. 46).
- [125] Y. Shi and R. Eberhart, 'A modified particle swarm optimizer,' in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, Anchorage, AK, USA: IEEE, May 1998, pp. 69–73. doi: 10.1109/ICEC.1998.699146 (p. 32).
- [126] Y. Shi and R. C. Eberhart, 'Empirical study of particle swarm optimization,' in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, Washington, DC, USA: IEEE, Jul. 1999, 1945–1950 Vol. 3. doi: 10.1109/CEC.1999.785511 (p. 33).
- [127] Y. Shi and R. C. Eberhart, 'Parameter selection in particle swarm optimization,' in *Evolutionary Programming VII*, V. W. Porto, N. Saravanan, D. Waagen and A. E. Eiben, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 591–600, ISBN: 978-3-540-68515-9 (p. 32).
- [128] E. P. Simoncelli and B. A. Olshausen, 'Natural image statistics and neural representation,' *Annual review of neuroscience*, vol. 24, no. 1, pp. 1193–1216, 2001 (p. 134).
- [129] K. Sörensen, 'Metaheuristics—the metaphor exposed,' *International Transactions in Operational Research*, vol. 22, no. 1, pp. 3–18, 2015 (p. 22).
- [130] M. Srinivas and L. Patnaik, 'Genetic algorithms: A survey,' *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994. doi: 10.1109/2.294849 (pp. 25, 28).
- [131] N. Srinivas and K. Deb, 'Multiobjective optimization using nondominated sorting in genetic algorithms,' *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994 (p. 107).
- [132] Tableau Software, *Business Intelligence and Analytics*, Accessed: 2018-06-19. [Online]. Available: <https://www.Tableau.com/> (p. 15).
- [133] H. H. Tan and K. H. Lim, 'Vanishing gradient mitigation with deep learning neural network optimization,' in *2019 7th International Conference on Smart Computing Communications (ICSCC)*, 2019, pp. 1–4. doi: 10.1109/ICSCC.2019.8843652 (p. 143).
- [134] *The fly algorithm*, https://en.wikipedia.org/wiki/Draft:Fly_algorithm (p. 9).

- [135] S. Thomson, 'Artificial evolution 2017: A report,' *SIGEVolution*, vol. 10, no. 4, pp. 3–5, Jun. 2018. doi: 10.1145/3231555.3231556 (p. 16).
- [136] A. Tonda, E. Lutton and G. Squillero, 'Lamps: A test problem for cooperative coevolution,' in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2011)*, vol. 387, Berlin, Heidelberg: Springer Berlin Heidelberg, Oct. 2011, pp. 101–120, ISBN: 978-3-642-24094-2. doi: 10.1007/978-3-642-24094-2_7 (pp. 43, 44, 56, 58, 59).
- [137] K. Varelas, A. Auger, D. Brockhoff, N. Hansen, O. A. ElHara, Y. Semet, R. Kassab and F. Barbaresco, 'A comparative study of large-scale variants of cma-es,' in *Parallel Problem Solving from Nature – PPSN XV*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete and D. Whitley, Eds., Cham: Springer International Publishing, 2018, pp. 3–15, ISBN: 978-3-319-99253-2 (pp. 43, 55).
- [138] D. R. Varma, 'Managing DICOM images: Tips and tricks for the radiologist,' English, *The Indian journal of radiology & imaging*, vol. 22, no. 1, pp. 4–13, Jan. 2012. doi: 10.4103/0971-3026.95396 (p. 13).
- [139] F. P. Vidal, D. Lazaro-Ponthus, S. Legoupil, J. Louchet, É. Lutton and J.-M. Rocchisani, 'Artificial Evolution for 3D PET Reconstruction,' in *Proceedings of the 9th international conference on Artificial Evolution (EA'09)*, ser. Lecture Notes in Computer Science, vol. 5975, Strasbourg, France: Springer, Heidelberg, Oct. 2009, pp. 37–48. doi: 10.1007/978-3-642-14156-0_4 (pp. 38, 81).
- [140] F. P. Vidal, J. Louchet, É. Lutton and J. Rocchisani, 'PET reconstruction using a cooperative coevolution strategy in LOR space,' in *IEEE Nuclear Science Symposium Conference Record*, IEEE, Oct. 2009, pp. 3363–3366. doi: 10.1109/NSSMIC.2009.5401758 (pp. 38, 81, 101).
- [141] F. P. Vidal, J. Louchet, J. Rocchisani and É. Lutton, 'New genetic operators in the Fly algorithm: Application to medical PET image reconstruction,' in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, vol. 6024, Istanbul, Turkey: Springer, Heidelberg, Apr. 2010, pp. 292–301. doi: 10.1007/978-3-642-12239-2_30 (pp. 38, 53, 81, 83).
- [142] F. P. Vidal, É. Lutton, J. Louchet and J. Rocchisani, 'Threshold selection, mitosis and dual mutation in cooperative coevolution: Application to medical

- 3D tomography,' in *International Conference on Parallel Problem Solving From Nature (PPSN'10)*, ser. Lecture Notes in Computer Science, vol. 6238, Krakow, Poland: Springer, Heidelberg, Sep. 2010, pp. 414–423. doi: 10.1007/978-3-642-15844-5_42 (pp. 38, 52, 81, 83).
- [143] F. P. Vidal, D. Lazaro-Ponthus, S. Legoupil, J. Louchet, É. Lutton and J. M. Rocchisani, 'Pet reconstruction using a cooperative coevolution strategy,' in *Proceedings of the IEEE Medical Imaging Conference 2009*, IEEE, Orlando, Florida, Oct. 2009 (p. 35).
- [144] P. A. Vikhar, 'Evolutionary algorithms: A critical review and its future prospects,' in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, Dec. 2016, pp. 261–265. doi: 10.1109/ICGTSPICC.2016.7955308 (p. 23).
- [145] P. Viola and M. Jones, 'Rapid object detection using a boosted cascade of simple features,' in *Proceedings of Computer Vision and Pattern Recognition*, vol. 1, IEEE Computer Society, 2001, pp. 511–518. doi: 10.1109/CVPR.2001.990517 (p. 116).
- [146] P. Viola and M. J. Jones, 'Robust real-time face detection,' *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004. doi: 10.1023/B:VISI.0000013087.49260.fb (p. 116).
- [147] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Í. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt and SciPy 1.0 Contributors, 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,' *Nature Methods*, vol. 17, pp. 261–272, 2020. doi: 10.1038/s41592-019-0686-2 (p. 140).
- [148] M. Wall, 'Galib: A c++ library of genetic algorithm components (1996),' *Mechanical Engineering Department, MIT* <http://lancet.mit.edu/ga>, 2008 (p. 25).

- [149] C. Wang and H.-W. Shen, 'Information theory in scientific visualization,' *Entropy*, vol. 13, no. 1, pp. 254–273, 2011. doi: 10.3390/e13010254 (p. 138).
- [150] D. Wang, D. Tan and L. Liu, 'Particle swarm optimization algorithm: An overview,' *Soft Comput*, vol. 22, pp. 387–408, 2018. doi: 10.1007/s00500-016-2474-6 (p. 33).
- [151] P. Wang, B. Xue, J. Liang and M. Zhang, 'Improved crowding distance in multi-objective optimization for feature selection in classification,' in *Applications of Evolutionary Computation*, Cham: Springer International Publishing, 2021, pp. 489–505. doi: https://doi.org/10.1007/978-3-030-72699-7_31 (p. 145).
- [152] M. Watanabe, K. Ida and M. Gen, 'A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem,' *Computers & Industrial Engineering*, vol. 48, no. 4, pp. 743–752, 2005, Selected Papers from The 30th International Conference on Computers & Industrial Engineering. doi: <https://doi.org/10.1016/j.cie.2004.12.008> (p. 28).
- [153] S. Wehrend and C. Lewis, 'A problem-oriented classification of visualization techniques,' in *Visualization, 1990. Visualization'90., Proceedings of the First IEEE Conference on*, IEEE, 1990, pp. 139–143 (pp. 17, 84).
- [154] T. Wen, R. P. Mihail, **S. Al-Maliki**, J. M. Létang and F. P. Vidal, 'Registration of 3D Triangular Models to 2D X-ray Projections Using Black-box Optimisation and X-ray Simulation,' in *Computer Graphics and Visual Computing (CGVC)*, F. P. Vidal, G. K. L. Tam and J. C. Roberts, Eds., Bangor University, United Kingdom: Eurographics Association, 2019, pp. 105–113, ISBN: 978-3-03868-096-3. doi: 10.2312/cgvc.20191265 (p. 5).
- [155] R. P. Wiegand and M. A. Potter, 'Robustness in cooperative coevolution,' in *Proceedings of the 8th annual conference on Genetic and evolutionary computation, Seattle, Washington, USA, ser. GECCO '06*, Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 369–376, ISBN: 1595931864. doi: 10.1145/1143997.1144063 (p. 35).
- [156] S. T. Wilson and D. E. Goldberg, 'A Critical Review of Classifier Systems,' in *Third International Conference on Genetic Algorithms*, 1989, pp. 244–255 (p. 33).

- [157] A. S. Wu, K. A. D. Jong, D. S. Burke, J. J. Grefenstette and C. L. Ramsey, 'Visual analysis of evolutionary algorithms,' in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 2, 1999, 1425 Vol. 2. doi: 10.1109/CEC.1999.782649 (p. 15).
- [158] H.-C. Wu, C.-T. Sun and S.-S. Lee, 'Visualization of evolutionary computation processes: From the perspective of population,' in *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788)*, vol. 3, Jun. 2004, pp. 2077–2081. doi: 10.1109/WCICA.2004.1341950 (p. 15).
- [159] Z. Xie, S. Huang, M. O. Ward and E. A. Rundensteiner, 'Exploratory visualization of multivariate data with variable quality,' in *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, IEEE, 2006, pp. 183–190 (p. 84).
- [160] P.-Y. Yin, 'A fast scheme for optimal thresholding using genetic algorithms,' *Signal Processing*, vol. 72, no. 2, pp. 85–95, 1999. doi: [https://doi.org/10.1016/S0165-1684\(98\)00167-4](https://doi.org/10.1016/S0165-1684(98)00167-4) (p. 27).
- [161] H. Zhang, Y. Hou, J. Zhang, X. Qi and F. Wang, 'A new method for nondestructive quality evaluation of the resistance spot welding based on the radar chart method and the decision tree classifier,' *The International Journal of Advanced Manufacturing Technology*, vol. 78, no. 5, pp. 841–851, 2015. doi: 10.1007/s00170-014-6654-1 (p. 19).
- [162] H. Zhou, X. Yuan, H. Qu, W. Cui and B. Chen, 'Visual clustering in parallel coordinates,' in *Computer Graphics Forum*, Wiley Online Library, vol. 27, 2008, pp. 1047–1054 (p. 17).
- [163] Y. Zhu, W. H. Hsu and J. H. Hollis, 'The impact of food viscosity on eating rate, subjective appetite, glycemic response and gastric emptying rate,' *PLOS ONE*, vol. 8, no. 6, pp. 1–6, Jun. 2013. doi: 10.1371/journal.pone.0067482 (p. 8).