Universiteit Leiden
Opleiding Informatica

Indicator-Based Evolutionary Level Set Approximation:

Foundations and Empirical Studies

Name:	Lai-yee Liu
Date:	26/07/2018
1st supervisor:	Dr. Michael T.M. Emmerich
2nd supervisor:	Dr. André H. Deutz
Other assessor:	Dr. Vitor Basto-Fernandes

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The aim of the Black Box Level Set Approximation Problem is to find an approximation of the set of inputs of a function that give rise to a targeted output below a threshold. The motivation for solving it lies in its relation to other science and engineering problems in both theoretical and practical domain. The Evolutionary Level Set Approximation algorithm (ELSA) is an Indicator-Based Evolutionary Algorithm which is tailored to solve this problem. It was tested on 2-dimensional level set benchmarks in its original paper. To gain insights on its behaviour and limitations in varied cases, we conduct an elaborate empirical study of ELSA on a high dimensional black box level set benchmark suite. The influences of algorithm parameters and characteristics of quality indicators are investigated alongside. Furthermore, we introduce a variant on the ELSA algorithm called ELSA-SR which includes a self-adaptative mutation step-size that is modified based on the one-fifth success rule method. Objectives for ranking the algorithms consist of measuring the amount of evaluations to yield a whole, feasible approximation set, the diversity according a quality indicator, and the coverage of the approximation set that describes the properties of its distribution. Comparisons between the algorithms reveal that ELSA-SR is a successful improvement over ELSA, particularly when it comes to solving high dimensional level set benchmarks which are failed to be solved by ELSA.

Contents

1 Introduction and related work

This introductory section of this thesis will cover several topics. First the main research problem of this thesis is highlighted in Section 1.1. Then, examples from related work are provided which include an overview of practical problems that is closely linked to the main problem (Section 1.2) and a comparison of the algorithms that have been used to solve these problems (Section 1.3). Our contributions for this thesis are presented in Section 1.4, followed by a section overview in Section 1.5.

1.1 Problem definition

The optimization problem is a classical problem in computer science which has propelled the development of evolutionary algorithms to solve it. However, there might be circumstances where one is not interested in the optimal solutions, but rather a set of (near-optimal) solutions that are considered acceptable if they satisfy the constraints. An approach to achieve such a set is through solving the Black Box Level Set Approximation Problem which is defined in Definition 1. The aim of the Black Box Level Set Approximation is to find an approximation of the set of inputs of a function that give rise to a targeted output below a threshold ϵ. The approximation set is technically a multiset as there exists no enforcement that all its elements have to be unique, when the definition of a set requires the elements to be all unique. For the sake of simplicity, we refer to multisets simply as 'sets' throughout this thesis. This problem is the main focus of this thesis.

Definition 1. Black Box Level Set Approximation Problem:
Given a black box function $f : S \to \mathbb{R}$, with $S \subset \mathbb{R}^n$, and a threshold $\epsilon \in \mathbb{R}$, we search for the set $L(f \leq \epsilon)$ which is defined as:

$$L(f \leq \epsilon) := \{\mathbf{x} \in S | f(\mathbf{x}) \leq \epsilon\}$$

Solutions in L will be termed feasible solutions.

1.2 Motivation and related practical problems

The importance of solving the Black Box Level Set Approximation Problem lies in its relation to other science and engineering problems in both theoretical and practical domain. Some examples of fields are listed in combination with related work that tackles a specific practical problem.

1. **Geometric Modeling**:
   Determine the points of an n-dimensional geometric shape that is implicitly defined by a complex (in)equality.

2. **Fault Diagnosis**:
   An inverse problem applied to fault diagnosis, where a system model and the observed output of the model (indication that a fault has occurred) are provided, but the set of inputs that produced the output (source of the fault) is initially unknown and is therefore aimed to be found. Zechman et al. defined the EAGA algorithm (which stands for Evolutionary Algorithm to Generate Alternatives). It is applied for the purpose of finding the possible source locations of pollutant leakage given a model of a water distribution system [9]. It provides a real-time response on a dynamic process, and at the beginning of time when the amount of observations are limited, the same output usually fit to multiple different inputs that predict similarly well. To address this non-uniqueness, an algorithm

should be able to maintain a set of alternative solutions representing different non-unique solutions.

3. **Parameter Identification of Complex Systems**:
   Given a partially specified system's model with some observations of its input output behaviour but some unknown parameters, find all parameter settings of a system's model that can explain an observed behaviour or represent the observations with low prediction error. An example of a practical problem for parameter identification is finding the unknown reaction rates (propensities) in a stochastic gene regulatory network (SGRN) model that match the given gene activation time series [12].

4. **Design Engineering and Constraint-based Optimisation**:
   Find all possible designs that comply with a described behaviour, where relaxed optimality such as satisfying some threshold is encouraged rather than finding the most optimal design. This is important in design engineering where constraints are imposed to a design, but generating different near-optimal solutions to an optimisation problem allows for choosing alternative solutions.

   For instance, the need of finding alternative solutions is applied in solving the Regional Wastewater Treatment Network Design Problem (in DuPage County) with EAGA [20], De Novo drug discovery with ELSA (which stands for Evolutionary Level Set Approximation) [16], and in bridge construction with the NOAH [15]. Zechman et al. [20] stresses another importance of alternative solutions, namely that system models can be flawed in representing reality or the real problem when they are incompletely defined due to unquantifiable issues that are important for decision-making, and therefore optimal solutions might be found unrealistic or incorrect for the real problem. Alternative near-optimal solutions are expected to perform differently with respect to the unmodeled issues, providing valuable choices when making decisions. In literature this is known as the modeling to generate alternatives approach (MGA).

1.3 Related algorithms

It is desirable for the aformentioned problems in Section 1.2 to yield an approximation set with maximum diversity, to which diversity optimisation algorithms are specifically tailored for this purpose. The differences between EAGA, NOAH, and ELSA will be highlighted.

EAGA uses the niching scheme [20], while NOAH and ELSA are Indicator-Based Evolutionary Algorithms (IBEA).
The final population in EAGA consists of the best individual of each subpopulation. One subpopulation focuses on locating the optimal solution to the problem and the rest of them are niches to maintain maximum diversity. Emphasis is put on the distances between a solution in one subpopulation to the centroids of other subpopulations, whereby d_{niches}, the minimum of these distances, is used as one way to rank the fitness of an individual solution. Distinction between feasible and infeasible solutions is made, where a solution is considered feasible when it satisfies the threshold of the allowed objective function value. Under the assumption of the ideal case where the best individuals are all feasible, the final population consists of one solution that is optimal with regards to the objective function value and the rest are solutions in a respective subpopulation that have the largest d_{niches} value. EAGA can solve both single-objective and multi-objective problems.

NOAH and ELSA do not incorporate any niching scheme, but as an indicator-based approach,

a quality indicator is applied on the entire population so that different sets of solutions are compared and selected on the basis of the diversity. IBEAs are not unique to level set approximation and are previously found in literature for approximating Pareto fronts in multi-objective optimisation where a quality indicator is a function that maps Pareto set approximations to a real number. While the Pareto front and level set approximation problems are not the same, they have their similarities and principles of multi-objective algorithm design can serve as inspiration for designing level set approximation approaches.

The NOAH algorithm is the first proposed IBEA for the diversity optimisation of the solution set in single- and multi-objective problems. Its aim is to determine a maximally diverse set of solutions whose objective values are within a provided threshold. This is achieved by iteratively switching between the objective value optimisation and diversity optimisation phases while automatically adapting a bound on the objective value until it reaches the threshold. The bound value is initialised on infinity and the new value for the bound is adapted after every objective value optimisation phase, such that the bound is gradually lowered. The new bound is set to the minimal value where at least r solutions are still on or below it, where r is a constant chosen by the user. This means that the feasibility of solutions in NOAH is controlled by the user's wishes, depending how strict or lax they are. NOAH uses the Solow-Polasky Indicator (SPI) as the diversity quality indicator. This quality indicator has several favorable theoretical properties but also requires the choice of a correlation parameter in its definition. Results on standard evolutionary problems such as the NK Landscapes Problem and the 3-SAT Problem and a more realistic bridge construction problem have shown that the algorithm is able to produce high quality solutions with a significantly higher structural diversity than standard evolutionary algorithms. NOAH can solve both single-objective and multi-objective problems.

The ELSA algorithm is designed with the Black Box Level Set Approximation Problem in mind. For this reason, the intended quality indicator to be used for ELSA is the Average Distance Indicator (ADI) which is based on the Average Hausdorff Distance principle. ADI requires a reference set to be offered as a parameter. Ideally for precise representation the target level set itself should be offered as a reference, but the issue is that the level set is unknown in for instance the black box level set problem. A way to circumvent this is using the entire search space that encapsulates the level set, however the calculated ADI value is not an accurate representation of what is intended with ADI. Another issue is that studies show that its extremely high complexity makes it impractical for actual use. The proposed quality indicators for ELSA to replace ADI are the Solow-Polasky Indicator (similarly to NOAH) and three Gap Indicators which have low complexity. Unlike NOAH, ELSA assumes feasibility to have the highest priority even over diversity. This is reflected by the use of an augmentation function, that is superposed on the quality indicator, to penalise the occurrence of infeasible solutions in population sets. The augmented quality indicators have been tested on the sphere function and the Branke's Multipeak function in 2-dimensional space in [5] and in 2D fattened Gielis formulas in the research project prior to this thesis [11]. The black box level set benchmarks that have been tested on ELSA are single-objective problems, but ELSA has also been adapted to multi-objective problems, such as for example the practical Drug Discovery problem [16].

NOAH and ELSA undergo steady-state schemes, but each algorithm approaches the generation of children in a different way. Children are generated from each subpopulation per generation in EAGA. NOAH can generate multiple children in one generation, while for ELSA there is strictly one child per generation. ELSA is the most basic one with a simple flow of operations that are found in any skeleton of a generic Evolutionary Algorithm. On top of that, it has a very limited amount of algorithm parameters which makes it easier to configure the parameter values

for a specific problem. It is not always clear what the recommended settings are for algorithm parameters in NOAH and EAGA. It is therefore challenging to do a fair comparison of these algorithms on a different problem than what have originally been tested for them. While it has been brought up earlier, it should be noted that these algorithms have a different perspective on the optimal solution to the optimisation problem. EAGA considers the optimal solution to be highly important as it has a dedicated subpopulation that aims to find and preserve it. NOAH has diversity optimisation as one of its algorithm phases, although it is not guaranteed that the optimal solution is preserved as the diversity optimisation phase comes right after it. ELSA does not put any priority to the optimal solution as every solution that has an objective function value within the threshold are ranked equally and their diversity contribution is what determines their survival.

1.4 Contributions of this thesis

The main contributions of this thesis are as follows:

1. **Time complexity reduction in ELSA code**:
   The original implementation of ELSA and its corresponding quality indicators are written in MATLAB 9.0 as found in [8]. MATLAB allows for visualisation of graphs and graphical shapes with its built-in functions. Due to the fact that MATLAB is a scripting language, it heavily depends on the code implementation to run efficiently. The old implementation of ELSA suffers from long runtimes even for relatively small populations and evaluation budget. The thesis will briefly touch upon the changes in the redesigned version of ELSA with lower time complexity, while it simultaneously preserves the original algorithm's behaviour and output.

2. **High dimensional black box level set approximation benchmarks**:
   Eight black box level set benchmarks have been defined for 2D, 3D, 10D and 30D. As ELSA has only been tested on 2D black box benchmarks, its performance on higher dimensions is unknown. For comparison of algorithms and to test their strengths and weaknesses, an extensive benchmark suite is required that includes a great variety of difficulties in solvability and cover different characteristics of level sets such as for example disjoint versus non-disjoint sets and acute versus smooth corners. The level set benchmarks are split up into two categories: basic shapes that are based on the sphere function and complex shapes that are derived from functions that are relevant to engineering or practical problems.

3. **ELSA-SR: Modified ELSA algorithm with self-adaptative mutation step-size**:
   Self-adaptation of the mutation step-size is a technique in the design of EAs to deal with the unknown function landscape. The first attempt on a self-adaptation for ELSA is based on the One-Fifth Success Rule which is one of the earliest proposed methods of self-adaptation. This modified algorithm will be dubbed as ELSA-SR.

4. **Recommendation of parameter configurations for ELSA and ELSA-SR**:
   The black box level set benchmarks are to be used to determine what influences the values of algorithm parameters have on the results. The results are evaluated to see whether it is possible to find a combination that yields overall decent results for the level set benchmarks in the suite (and ideally for any black box level set benchmark as well).

1.5 Section overview

An overview of the structure of this thesis will now be provided. Section 2 is about the algorithms for solving the black box level set approximation problem, which includes details on the steps of both ELSA and ELSA-SR, formal definitions of the quality indicators, and current implementation of ELSA in regards to time complexity reduction. The black box level set benchmarks that will be used for the experiments in this thesis are defined in Section 3. Section 4 revolves around the set-up of experiments, the corresponding results and observations. The experiments can be divided into three parts: (1) recommendation of parameter settings for the ELSA and ELSA-SR algorithms, (2) comparison of the quality indicators on the entire level set benchmark suite, and (3) comparison of different algorithm configurations on the entire level set benchmark suite. Finally, the thesis is concluded in Section 5 with a summary on the discoveries and an outlook for future work.

2 Algorithms for black box level set approximation

This section revolves around the details concerning the implementation of the algorithms to solve the level set approximation problem. Section 2.1 discusses the computational steps in ELSA and Section 2.2 the steps in ELSA-SR. The formal definitions of the quality indicators to rate the diversity of a population are introduced in Section 2.3. Finally, in Section 2.4 a brief summary is provided about the major changes in the current implementation of ELSA compared to the original one in [8] for time complexity reduction in Section 2.4.

2.1 ELSA algorithm

ELSA is a relatively novel, simple in design evolutionary algorithm (EA) for level set approximation. The algorithm is dependent on a quality indicator that rates the fitness of a population set. ELSA is a $(\mu + 1)$-EA, which means that ELSA produces one child per generation and only one solution cannot survive to the next generation. Steady-state selection schemes are commonly adopted by indicator-based EAs (IBEAs) to circumvent computationally expensive subset selection problems [4]. The steps of the ELSA algorithm are shown in Algorithm 1. As we deal with a continuous domain in the level set approximation problem, the described operations in ELSA resemble those in Evolutionary Strategies (ES).

Algorithm 1 Indicator-Based Evolutionary Level Set Approximation (ELSA)

1: $P_0 \leftarrow$ init() {Initialise population}
2: $t \leftarrow 0$
3: **while** not terminate **do**
4: $u \sim \text{rand}(0, 1)$ {Draw uniform number between 0 and 1}
5: **if** $u \leq \nu$ **then**
6: $\mathbf{q} \leftarrow \text{mutate}(P_t, \sigma)$ {create new child solution by parent-based mutation}
7: **else**
8: $\mathbf{q} \leftarrow$ reinitialise(S) {create new child solution by random reinitialisation}
9: **end if**
10: $P'_t \leftarrow P_t \cup \{\mathbf{q}\}$
11: $\mathbf{r} = \arg\min_{\mathbf{p} \in P'_t}(\Delta_{QI}(\mathbf{p}, P'_t))$ {Select solution that contributes least to QIC}
12: $P_{t+1} \leftarrow P'_t \setminus \{\mathbf{r}\}$
13: $t \leftarrow t + 1$
14: **end while**
15: **return** P_t

P_t is the population of the approximation set in generation t. The initial population P_0 is generated by drawing random points from a uniform distribution in the search space S.

The first step in the evolutionary loop is to create a novel child solution $\mathbf{q} \in S$ (steps 4-9). The ELSA algorithm omits some techniques from a traditional ES such as recombination from multiple parents and elitist selection schemes for breeding. Instead, it adopts a mixed mutation strategy [10]: the algorithm either creates a child by randomly picking a new point in S (random reinitialisation), or by adding a perturbation to a random solution in P_t (parent-based multi-point mutation). In the parent-based mutation, each variable value in vector $\mathbf{q}$ is generated by drawing a random real number in a normal distribution where the respective variable in the parent solution corresponds to the mean and a user-defined variable called σ corresponds to the standard deviation. It is possible for parent-based mutation to generate an initial child solution that falls outside the search space, and an extra function operation is

required to limit this solution to the interior of S. In case the solution is outside of S, ELSA reflects the solution vector $\mathbf{q}$ with the boundaries of the search space defined by the lower and upper bounds of the search space as axes. It is randomly decided to use which mutation, but the user-defined constant variable $\nu \in \{0, 1\}$ dictates the probability for a parent-based mutation to happen. For $0 < \nu < 1$, ELSA is considered to be a mixed mutation strategy. As a random value $u \in \{0, 1\}$ is drawn from an uniform distribution, a child is produced from parent-based mutation if $u \leq \nu$ holds, otherwise the child is randomly generated.

P'_t is the temporary new population that includes child solution $\mathbf{q}$ (step 10). To keep the population size constant, the least contributing solution $\mathbf{r}$ in P'_t needs to be eliminated. This is decided by ranking all the solutions in P'_t by their quality indicator contribution (QIC) which is defined in Definition 2 (step 11). QIC is to be maximised, similarly to a QI which is to be maximised as well, therefore the least contributing solution has the lowest QIC value in the population. After discarding $\mathbf{r}$ from P'_t, the new parent population P_{t+1} is formed (step 12).
The algorithm is terminated when the number of evaluations exceeds the user-defined evaluation budget. The evaluation refers to the examination of an individual solution, such as calculating its corresponding objective function value.

Definition 2. Quality Indicator Contribution:
Let QI be a quality indicator defined on a set of solutions. These sets could have any finite cardinality. Given a set of solutions P, we define the individual quality indicator contribution $\Delta_{QI}(\mathbf{p}, P)$ of a solution $\mathbf{p} \in P$ with respect to P as:

$$\Delta_{\mathrm{QI}}(\mathbf{p}, P) \leftarrow \mathrm{QI}(P) - \mathrm{QI}(P \backslash \{\mathbf{p}\}) \tag{1}$$

The rationale behind the mixed mutation strategy is that EAs normally combine two types of searches: global search and local search. Global search is to find areas of interest (also known as exploration) and is achieved by having a large σ step-size in an ES. Thanks to exploration in the search space, the algorithm is less likely to be trapped in a local optima and it promotes the search of different level set components in the case of disjoint level sets. Whereas local search is to improve solutions (exploitation) and requires a small σ step-size in an ES. Using a very large σ step-size produces similar effects to random initialisation in the search space, hence the two ways of producing children in ELSA can be considered forms of applying global and local search in the algorithm.
When adapting ELSA for specific continuous problems, it is sometimes recommended to modify the method of producing a child. Such as in the Gene Regulatory Network problem a modified ELSA is used where a child can be created by random initialisation, random recombination, or single and multiple point mutations [12].

2.2 ELSA with self-adaptative mutation step-size: ELSA-SR algorithm

ELSA-SR is a variation on ELSA where an adaptive σ step-size is applied which is modified according to the method described in one-fifth success rule [2]. The steps in the algorithm are shown in Algorithm 2. ELSA-SR has the same algorithm parameters as ELSA and several exclusive parts derived from the one-fifth success rule method which are as follows:

- learning rate (α): learning rate to adapt the σ mutation step-size. It needs to be in the following range: $0 < \alpha < 1$.
- measurement range (β): the amount of evaluations over which the success rate is measured where a child is generated from parental mutation. Success is defined as the ability to produce a child that improves the diversity of the population according its quality indicator.

- target success rate (γ): the success rule dictates that the optimal progress rate can be maintained by adapting the mutation strength σ in such way that the success rate resembles the target success rate. As the name states, γ is 0.2 in the one-fifth success rule because this probability is found to be optimal for the specific tested problems in Rechenberg's research.

Algorithm 2 Indicator-Based Evolutionary Level Set Approximation with Success Rule (ELSA-SR)

1: $P_0 \leftarrow$ init() {Initialise population}
2: $t \leftarrow 0$
3: $ChosenCount \leftarrow 0$
4: $SuccessCount \leftarrow 0$
5: $\sigma \leftarrow$ init() {Initialise σ step-size}
6: **while** not terminate **do**
7: $u \sim \text{rand}(0, 1)$ {Draw uniform number between 0 and 1}
8: **if** $u \leq \nu$ **then**
9: $\mathbf{q} \leftarrow \text{mutate}(P_t, \sigma)$ {create new child solution by parent-based mutation}
10: $ChosenCount \leftarrow ChosenCount + 1$
11: **else**
12: $\mathbf{q} \leftarrow$ reinitialise(S) {create new child solution by random reinitialisation}
13: **end if**
14: $P'_t \leftarrow P_t \cup \{\mathbf{q}\}$
15: $\mathbf{r} = \arg\min_{\mathbf{p} \in P'_t}(\Delta_{QI}(\mathbf{p}, P'_t))$ {Select solution that contributes least to QIC}
16: $P_{t+1} \leftarrow P'_t \setminus \{\mathbf{r}\}$
17: $t \leftarrow t + 1$
18: **if** $u \leq \nu$ **then**
19: **if** $\mathbf{q} \neq \mathbf{r}$ **then**
20: $SuccessCount \leftarrow SuccessCount + 1$
21: **end if**
22: **if** $ChosenCount \bmod \beta = 0$ **then**
23: **if** $SuccessCount/\beta > \gamma$ **then**
24: $\sigma \leftarrow \sigma/\alpha$ {increase σ}
25: **else if** $SuccessCount/\beta < \gamma$ **then**
26: $\sigma \leftarrow \sigma \cdot \alpha$ {decrease σ}
27: **end if**
28: $SuccessCount \leftarrow 0$
29: **end if**
30: **end if**
31: **end while**
32: **return** P_t

The algorithm flow of ELSA-SR is very similar to ELSA, so only the additional steps will be highlighted. In the initialisation phase before entering the evolutionary loop, two variables called *ChosenCount* and *SuccessCount* are initialised to 0 (step 3 and 4). *ChosenCount* is to keep track the number of times the parent-based mutation is chosen (incremented in step 10). *SuccessCount* counts the number of times a child is produced that is not eliminated from the population set (incremented in step 20). Steps 18-30 are the important steps added for the step-size adaptation. First the algorithm checks whether a new child has been born from parent-based mutation (step 18). If this is not the case, steps 19-30 are skipped, since children from random re-initialisation should not affect the step-size adaptation process. However, assuming

the child is born from a parent, it then checks whether this child has survived the elimination process. If it is the case, *SuccessCount* will be incremented, otherwise it stays the same value. The next check is comparing *ChosenCount* with the chosen measurement range β to decide whether it is time to adapt the σ step-size (step 22). If it is the case, then $SuccessCount/\beta$ is compared with the target success rate γ and the decision of what to do with σ depends on the following options:

- $SuccessCount/\beta > \gamma$: increment σ step-size by dividing σ with α
- $SuccessCount/\beta < \gamma$: reduce σ step-size by multiplying σ with α
- $SuccessCount/\beta = \gamma$: σ step-size remains the same

After adapting σ, *SuccessCount* is resetted to 0 (step 28).

To explain the motivation for using a success rule for adaptation, a brief history about the original 1/5th success rule is given. The limitations of using a fixed σ mutation step-size was documented by Rechenberg and Schwefel in 1973 where they used a (1+1)-ES with isotropic Gaussian mutations of $\sigma = 1$ on the optimisation of the 10D sphere function [13]. This function has only one optimum, so in an (1+1)-ES with only one parental state, the algorithm's progress rate towards the optimum can be determined. It is apparent that this strategy becomes extremely slow after a period of improvements, as the system has lost its evolvability. Evolvability is the idea that the mutations should be generated in such a way that improvement steps are likely, thus building a smooth evolutionary random path through the fitness landscape towards the optimum solution. Both the success probability (for finding a child that is better than its parent) and the progress rate are dependent on the chosen σ, as the progress rate scales with σ. Using a very small $\sigma \to 0$ mutation strength ensures a high degree of evolvability, a success probability of 0.5, but it leads to a low progress rate. The opposite case of using a large $\sigma \to \infty$ results into a low evolvability, success probability and progress rate. The evolution window describes the bandwidth between the two extrema of σ where a σ step-size within this window can guarantee an optimal progress rate. Plotting out the success probability and the normalised progress rate against σ in the sphere model in the asymptotic limit case $n \to \infty$ (where n is the dimension), it can be found which success probability corresponds to the optimal normalised progress rate by reading off which σ value they both match to on the two graphs. The optimal success probability on the sphere model is approximately 0.27. The same test is done on the corridor model which yielded an optimal success probability of approximately 0.184. As both models are regarded as typical models of objective functions, the compromise success probability was set to 0.2. The 1/5th success then states that the σ step-size should be tuned in such a way that a success rate of approximately 0.2 is achieved in order to obtain nearly optimal (local) performance of the (1+1)-ES in real-valued search spaces.

It should be taken into consideration that there are many crucial differences between ELSA and the originally tested strategy for the 1/5th success rule. ELSA is a (μ+1)-ES and not a (1+1)-ES. The problem sets for 1/5th success rule all have one optimum while we deal with the level set approximation problem that is an entirely different problem altogether where there are often an infinite amount of sets corresponding to a level set approximation of maximum diversity. Therefore the details of the theory on which the 1/5th success rule is based on are likely to not transfer to the level set approximation level set problem or on an algorithm like ELSA. However using a success rule allows the adaptation of the σ step-size in a controlled manner. It checks whether sufficient improvements are found after a certain number of iterations, and if not, a smaller σ step-size is tried.

2.3 Quality indicators for level set approximation

To evaluate the diversity in an approximation set for the level set approximation problem, quality indicators (QI) are used in the Indicator-Based Evolutionary Algorithms. Quality indicators are fitness functions that assign a scalar value to an approximation set A $\subset S$, so that different approximation sets can be ranked on their quality. Qualities to be considered are for instance how many points in A belong to the level set, how well the points are distributed and whether the points provide a good coverage of the level set. The (ELSA) algorithm is to find an approximation set that maximises the quality indicator value.

Some terms need to be addressed before going into the discussion of the different quality indicators that are used in this thesis for ELSA and its modified variant. In particular, the distinction between diversity and coverage, which have been mentioned to be desirable qualities for a quality indicator. Diversity and coverage should be regarded as concepts that encapsulate a specific goal. The formalisation of both concepts was first attempted in [5] and for better understanding of the level set approximation problem it is necessary to recite them.
Diversity refers to finding a distribution of a set with points that are maximally different to each other where the target set is treated as the allowed space where these points can reside in. There is no formal definition that is universally agreed upon to what it entails for 'the points in a set to be maximally diverse from each other', hence different quality indicators can be used to define how the diversity should be calculated.
Coverage refers to finding an approximation set that represents the target level set with maximal resemblance. Similarly to diversity, there is no formal definition of what 'the resemblance between an approximation set and target set' means. The larger the approximation set the better the ability to express the shape of the level set accurately.
An informal example to illustrate the difference when prioritising diversity or coverage as a goal is given: take a ball and an approximation set of small size, prioritising diversity describes a set with solutions that are more inclined to reside on the boundary of the ball (because spread drives solutions away from each other), while prioritising coverage describes a set that also fills the inside of the ball. As a level set consists more than its boundary, having the solutions of the approximation set reside only on its boundary is misleading in terms of resemblance between the approximation set and the target level set.

The remainder of this section is structured as follows: Section 2.3.1 to Section 2.3.4 present the definitions of the quality indicators. These definitions are under the assumption that all solutions in the approximation set are feasible, namely they all are elements within the target level set. Naturally this is not applicable to realistic situations, so an augmentation function is superposed on the quality indicator. Section 2.3.5 describe the employed measures to make the quality indicators work on approximation sets that may include infeasible solutions and compliant to the requirements of ELSA (such as maximisation of the quality indicator contribution).
The Weitzman Indicator (WI, in Section 2.3.1) was motivated with a list of axioms for what diversity entails. This quality indicator is not actually used by ELSA because of its impractically high complexity, but is included for its significance for defining diversity. The quality indicators that are used in this thesis are the three Gap Indicators (GI) (in Section 2.3.2): Minimal Gap Indicator (GI_N), Arithmetic Mean Gap Indicator (GI_Σ), Geometric Mean Gap Indicator (GI_Π); Solow Polasky Indicator (SPI, in Section 2.3.3); and Average Distance Indicator (ADI, in Section 2.3.4). WI, GI, and SPI are designed to measure diversity, while ADI is designed to measure coverage, therefore despite they are all used as quality indicators their differences lie in their prioritisation.

2.3.1 Weitzman Indicator (WI)

Weitzman [19] made the first attempt to define axioms to describe properties of diversity, so that there exists a guideline for designing quality indicators. He came up with an indicator, which has its original main purpose in measuring biodiversity in populations. Let A be the approximation set, then the Weitzman Indicator (WI) is defined in Formula 2.

$$\mathrm{WI}(A) = \max_{x \in A}\big\{\mathrm{WI}(A\backslash\{x\}) + d(x, A\backslash\{x\})\big\} \tag{2}$$

The d function is a distance-dissimilarity measure. In the case of measuring the difference between any pair of points (i, j), where $i \in A$ and $j \in A$, the following properties need to be satisfied:

$$d(i, j) \geq 0 \tag{3}$$

$$d(i, i) = 0 \tag{4}$$

$$d(i, j) = d(j, i) \tag{5}$$

Weitzman states that the nature of the problem should determine the appropriate distance-dissimilarity measure. The Euclidean distance is used for the black box level set approximation problem in this thesis. Furthermore the distance-dissimilarity d can be expanded for the measurement of the distance between a point and a set which is defined in Definition 3. This will be used in multiple quality indicators.

Definition 3. Solution-Set Distance:
Given a solution x and a set B, the distance d from x to B, where $x \notin B$, is defined by:

$$d(x, B) = \min_{y \in B} d(x, y), \tag{6}$$

This can be seen as the distance between x and its nearest neighbour in B.

Unfortunately the Weitzman Indicator has an exponential time complexity, namely $O(2^n)$, which makes it unattractive to be used as a quality indicator.

Regardless, it is important to examine the list of properties for a diversity indicator which has been suggested by Weitzman [19]. D in this case refers to a generic quality indicator and not necessarily the Weitzman Indicator itself, but the axioms were proven to hold for the Weitzman Indicator. The term 'species' is analogous to a solution in a population or a point in a set in our context.
The diversity properties to be highlighted are: Monotonicity in species property (Definition 4), Twin property (Definition 6), and Monotonicity in distances property (Definition 8).

Definition 4. Monotonicity in species property:
If solution x is added to collection A, then $D(A \cup \{x\}) \geq D(A) + d(x, A)$, $\forall A, \forall x \notin A$.

Definition 5. Link property:
For all A, $|A| \geq 2$, there exists at least one species $x \in A$, called the *link species*, that satisfies $D(A) = D(A\backslash\{x\}) + d(x, A\backslash\{x\})$.

Definition 6. Twin property:
Suppose that some species y outside of A is identical to some species x belonging to A, meaning for some $x \in A, y \notin A$ that $d(x, y) = 0$, $\forall z \in A$, $d(x, z) = d(y, z)$.
Then if y is added to A, there is no change in diversity: $D(A \cup y) = D(A)$.

Definition 7. Continuity in distances property:
Let $|A| = |A'|$ and let $\phi : A \to A'$ be a bijection mapping A to A'.
Then, $\forall \psi > 0$, $\exists \delta > 0$ such that $\sum_{x \in A, y \in A} |d(x, y) - d(\phi(x), \phi(y))| < \delta$ implies that $|D(A) - D(A')| < \psi$.

Definition 8. Monotonicity in distances property:
Let $|A| = |A'| \geq 2$ and let $\phi : A \to A'$ be a bijection mapping A to A'.
Suppose that, $d(\phi(x), \phi(y)) \geq d(x, y)$, $\forall x \in A$, $\forall y \in A$, $x \neq y$. Then $D(A') \geq D(A)$.

Definition 9. Maximum diversity that can be added by a species property:
If species y is added to collection A, then $D(A \cup y) \leq D(A) + \max_{x \in A}\{d(x, y)\}$

These properties are known to be shared by some of the other quality indicators described in this section and are also qualities that are desired for the approximation of a level set. The Monotonicity in species property dictates that the diversity of a set A should increase when a solution $x \notin A$ is added to it. The Twin property enforces that the diversity should remain constant when adding a duplicate solution y to A. The intention of including this property is to not reward the occurrence of duplicate solutions. The Monotonicity in distances property guarantees that if every pair of solutions in A' has a larger or equal distance than its mapped pair in A, then the diversity of A' should be higher or equal compared to A. The Maximum diversity added species property relates the distance quantitatively to the diversity.

2.3.2 Gap Indicators (GI)

All three Gap Indicators are low complexity quality indicators which are to be maximised. The computational complexity of Minimal Gap is $O(n \log n)$ and the other two are at most $O(n^2)$. The Minimal Gap Indicator focuses on the smallest Solution-Set Distance found in an approximation set. Due to that it only takes one Solution-Set Distance into account, it can only detect local improvement. Both Arithmetic Mean Gap Indicator and Geometric Mean Gap Indicator take all the Solution-Set Distances into account and give them all equal weight.

Minimal Gap:

$$\mathrm{GI}_N(A) = \min_{x \in A} d(x, A \backslash \{x\}) \tag{7}$$

Arithmetic Mean Gap:

$$\mathrm{GI}_\Sigma(A) = \frac{1}{|A|} \sum_{x \in A} d(x, A \backslash \{x\}) \tag{8}$$

Geometric Mean Gap:

$$\mathrm{GI}_\Pi(A) = \Big(\prod_{x \in A} d(x, A \backslash \{x\}) \Big)^{\frac{1}{|A|}} \tag{9}$$

Of the diversity properties, the following holds for all three of the Gap Indicators:

- Monotonicity in species property: no. The augmentation of the Gap Indicators addresses this issue this in Section 2.3.5.
- Twin property: no, however some additional remarks should be added to this. The distance between a solution and itself is 0, which results into a diversity of 0 for all three Gap Indicators when a duplicate solution occurs. Consider a population A that does not contain duplicates, therefore every pairwise distance in it has a value higher than 0. Adding a duplicate solution to A results into a smaller diversity. The detection of duplicates is possible when a Gap Indicator is used for the calculation of the quality indicator contribution

iff a population solely consists of non-duplicates and a duplicate solution is added to it (the duplicate solution that added to it will be rooted out as the least contributing). But if the population already contains at least one pair of duplicates, using a Gap Indicator for the calculation of the quality indicator contribution results in failure of pinpointing the duplicate solutions to be least contributing. It will be disadvantageous to add duplicates in most cases.

- Monotonicity in distances property: yes

2.3.3 Solow Polasky Indicator (SPI)

The Solow Polasky Indicator inspired the Weitzman Indicator and has most of its axiomatic properties but not all of them. This indicator is to be interpreted as the number of species in a population (it was also introduced for the purpose of biodiversity quantification) and has a value that ranges from 1 to the set size. The Solow Polasky Indicator value is also to be maximised.
Let us for a given approximation set $A = \{x_1, \ldots, x_\mu\}$ define matrix M with $m_{i,j} = \exp(-\theta d_{x_i,x_j})$, $\forall i, j = 1, \ldots, \mu$ for a positive parameter θ that scales the distance matrix $d_{x_i,x_j}, \forall i, j = 1, \ldots, \mu$. In case M is non-singular, define $C = M^{-1}$ and denote the entries of C by $c_{i,j}$. Then the Solow Polasky Indicator, denoted by SPI_θ, is defined as follows:

$$\mathrm{SPI}_\theta(A) = \sum_{i=1}^{\mu} \sum_{j=1}^{\mu} c_{i,j} \tag{10}$$

$\theta = 10$ is recommended [15] and sometimes the setting of θ is left out for convenience when denoting SPI as we always use $\theta = 10$. [15] also described an efficient implementation of the Solow Polasky Indicator for quality indicator contribution with a computational complexity of $\mathrm{O}(n^3)$. However, it has to be remarked that the setting of θ can critically influence the behaviour of SPI and in problems that are differently scaled than the benchmarks in this study, the setting of θ might have to be changed. The traditional calculation of SPI for quality indicator contribution has a complexity of $\mathrm{O}(n^5)$. It is a more favourable quality indicator to be used over the Weitzman Indicator, as it has lower complexity in large sets and contains most of its properties. The following diversity properties are fulfilled by the Solow Polasky Indicator:

- Monotonicity in species property: yes
- Twin property: yes
- Monotonicity in distances property: yes

2.3.4 Average Distance Indicator (ADI)

The Average Distance Indicator is based on the Average Hausdorff Distance principle, which is a measurement for the differences between two set [5]. ADI is a high complexity indicator. The complexity is dependent on the dimensions of the search space. For 2-dimension space ADI is $O(n \log n)$, however the complexity is unknown for higher dimensions (it is estimated to be exponential). Despite having the same asymptotic time complexity as Minimal Gap, ADI in general still takes more time to compute than the other mentioned quality indicators in practical situations.
Unlike the quality indicators that have been covered prior, Average Distance Indicator aims for finding an approximation set that is representative of the target level set L, rather than aiming for diversity optimisation. Here, the goal is to provide a near neighbour for each solution in the approximation set, where the average distance of points in the target level set L to their

nearest neighbours is minimised.
ADI requires an explicitly known reference set R, that is the target set which the approximation set is supposed to cover. Due to that the level set is not explicitly known, it cannot be taken as the reference set. However, since the level set is a subset in the search space, we take the search space as the reference set. When improving the ADI value for the reference set it also improves the value for the level set. There is a strong correlation between improving ADI and the Average Hausdorff Distance to the actual level set.
The Average Distance Indicator value is to be minimised. This is in conflict with the maximisation of a quality indicator in the description of ELSA, therefore the specific changes to the ADI is addressed in its augmentation function in Section 2.3.5.

The Average Distance Indicator (ADI) is defined in Formula 11:

$$\mathrm{ADI}_R(A) = \frac{1}{\mathrm{Volume}(R)} \int_{x \in R} d(x, A) \mathrm{d}x \tag{11}$$

The formulation of ADI implementation in 2D search space to calculate its exact value is defined in Formula 12. [5] includes a formulation of the calculation of ADI for 1D. For higher dimensions, the calculation of ADI has not yet been formulated and its ADI value is expected to be approximated instead.

Voronoi tessellation is applied on reference set R, which forms Voronoi cells around the solutions $x \in A$. The procedure assumes a multiset with unique elements as input, as it is not defined for elements that are identical to each other [1]. The purpose of the Voronoi cells is to describe which solution is nearest to an arbitrary point in R by looking up in which cell the point resides in (a point that resides on a cell border means that it has more than one nearest neighbour). While normally Voronoi cells at the border can be indefinitely large, we bound them to the border of R in this case. The next step is making it possible to calculate the average distances in the reference set surrounded by a Voronoi cell towards the corresponding solution. This is done by applying triangulation on the Voronoi cells where triangles are formed by adding edges from the solution vertex to all the vertices of its Voronoi cell. A single triangle is denoted with $\Delta(a, b, c)$, where c is the vertice of a solution so that a and b are vertices in an original edge from its respective Voronoi cell. VT denotes the set of all the triangles in area R. The calculation of ADI on point {c} with triangle $\Delta(a, b, c)$ as reference set is defined in Formula 13.

$$\mathrm{ADI}_R(A) = \frac{1}{\mathrm{Area}(R)} \sum_{\Delta(a,b,c) \in VT} \mathrm{Area}\big(\Delta(a, b, c)\big) \mathrm{ADI}_{\Delta(a,b,c)}(\{c\}) \tag{12}$$

$$\mathrm{ADI}_{\Delta(a,b,c)(\{c\})} = \frac{\mathrm{d}(a, b)}{6} \Big(u(1 + v^2) + \frac{1}{2} \cdot (1 - u^2) \cdot (1 - v^2) \cdot \log \frac{u - 1}{u + 1} \Big) \tag{13}$$

$$u = \frac{d(a, c) + d(b, c)}{d(a, b)} \tag{14}$$

$$v = \frac{d(a, c) - d(b, c)}{d(a, b)} \tag{15}$$

Concerning which of the Weitzman diversity properties apply to ADI, the following can be said (note that some of the operators in the Weitzman axioms need to be reversed to its opposite, such as 'greater than' turned into 'smaller than' due to the maximisation and minimisation discrepancy for the QI) [5]:

- Monotonicity in species property: yes
- Twin property: yes

- Monotonicity in distances property: no

2.3.5 Augmented quality indicators

Each quality indicator can be extended to an augmented quality indicator, which in turn can be used as a quality indicator in a steady-state selection. The augmentation of a quality indicator adds a penalty to the occurrence of undesirable solutions in the approximation set. Its generic formula is defined in Formula 16, where *ind* is the quality indicator in its original format as defined in Section 2.3.2 to Section 2.3.4 and A is a population representing the approximation set. This augmented quality indicator is to be maximised. Throughout this thesis, augmented quality indicators are denoted with a $^+$ suffix, for example SPI becomes SPI$^+$.

$$ind^+(A) \leftarrow ind(A \cap L) - penalty^{ind}(A \backslash L) \tag{16}$$

The augmentation of ADI is defined in Formula 17 which allows a quality indicator that is to be minimised to function for maximisation.

$$ind^+(A) \leftarrow -ind(A \cap L) - penalty^{ind}(A \backslash L) \tag{17}$$

The augmentation has multiple purposes that are outlined.

1. It distinguishes between the feasible and infeasible solutions from the approximation set, because we ultimately care more about the feasible subset. For example in Constraint-based Optimisation that is mentioned in Section 1.2, infeasible solutions are unusable for engineering designs where strict criteria hold. The quality indicator in their original format is only applied to the feasible subset.

2. It prioritises a high feasibility rate among the solutions in the approximation set when infeasible solutions are penalised. Adding feasible solutions to the set is rewarded, as infeasible solutions are removed from the approximation set in tandem. Therefore it is not an issue anymore where some quality indicators lack the Monotonicity in species property, when the added penalty function to infeasible solutions is chosen in such a way that an approximation set with more feasible solutions has a better augmented quality indicator ranking than an approximation set with less feasible solutions. Thanks to the augmentation it can be said that the Monotonicity in species property hold for the Augmented Gap Indicators.

3. The penalty function *penalty* is able to rank an infeasible set that has a shorter distance to the target level set as better than a set with a larger distance. It is to measure the closeness of the approximation set towards feasibility.

4. Although it is not included in the definition of Formula 16, it is possible for Augmented Gap Indicators to promote the removal of duplicate solutions. An example of such implementation is an extra penalty function that is evaluated over all solutions in A, where the unique set of A should be determined first, and the subset to be submitted to *ind* consists of the intersection of feasible and unique sets, and an adequate penalty is enforced on each element that is identical to the unique set. It is regarded as a situational augmentation because depending on the problem and settings of the Indicator-Based EA the generation of an exact replica of another solution in the population should not be a frequent occurrence at all.

The penalty functions are defined for a corresponding quality indicator, where f is the level set function. All the GI$^+$ indicators share the same function for *penalty* in Formula 18, while

SPI$^+$ and ADI$^+$ share a different function for *penalty* in Formula 19.
The upper bound for a Solution-Set Distance value is the diameter of the search space S. By multiplying the maximum achievable distance with the number of infeasible solutions, this part of the penalty function ensures that the Monotonicity in species property hold for GI$^+$ indicators. The cumulative difference between the function values of the infeasible solutions and threshold ϵ is to compare which of the infeasible sets is closer to feasibility.

$$penalty^{\mathrm{GI}} = \sum_{\mathbf{x} \in A \setminus L} \big(\mathrm{diameter}(S) + (f(\mathbf{x}) - \epsilon)\big) \tag{18}$$

$$penalty^{\mathrm{SPI}} = penalty^{\mathrm{ADI}} = \sum_{\mathbf{x} \in A \setminus L} \big(f(\mathbf{x}) - \epsilon\big) \tag{19}$$

There are cases when the approximation set is too small to measure a diversity for it, such as the Gap Indicators rely on distances between solutions for measurement. In that case these approximation sets are assigned a default value. For Gap Indicators $D = 0$ if $|A| < 2$; for SPI $D = 0$ if $|A| < 1$; and for ADI $D = \mathrm{diameter}(S)$ if $|A| < 1$.

2.4 Modified approach: Time complexity reduction in ELSA

The initial MATLAB ELSA implementation presented several performance constraints, including for ordinary algorithm parameters (e.g. population size of 100 and evaluation budget of 10K evaluations). Some modifications on the original source code implementation in MATLAB have been done with the intention of reducing the complexity when running the ELSA algorithm.

Section 2.4.1 pinpoints the bottlenecks in the original implementation of ELSA.
The rest of Section 2.4 presents different parts of the implementation changes that make up the modified approach. Section 2.4.2 describes a reformulation of the quality indicator contribution function that splits it up into two simpler functions that are to be used depending on whether the removed solution from the population is feasible or infeasible. Section 2.4.3 focuses on the distance matrix, that includes an implementation to calculate a distance matrix in shorter runtime than previously, and introduces a method to deal with the reuse of the distance matrix. Section 2.4.4 to Section 2.4.6 are about the incremental updating algorithms for the augmented quality indicators: GI$^+$ in Section 2.4.4, SPI$^+$ in Section 2.4.4 and ADI$^+$ in Section 2.4.6.
Results from tests in Section 2.4.7 reflect massive speed-up in the modified approach compared to the original implementation.

2.4.1 Original implementation in ELSA

The most computation-heavy aspect from ELSA has little to do with the algorithm itself, but concerns the calculation of quality indicator contribution from the respective augmented quality indicators.

Quality Indicator Contribution function:

The structure of the original MATLAB implementation for QIC calculation in ELSA as found in the original code [8] is a literal implementation of Formula 1 made up of a generic function which requires a function handle that describes the quality indicator. The function for the quality indicator is called $|P'|+1$ times in total where population P' represents the main population with a new child. First, QI(P') is calculated and its resulting value stored, then followed by entering a for-loop to remove one solution p_i from P' for the calculation of QI($P' \setminus p_i$) (to achieve the QIC of removing p_i), where i is the index of a solution. It should be noted that the only

stored data structures are the population matrix P' and the corresponding function values of the solutions in matrix $\mathbf{f}$. Any operation within a quality indicator are to be repeated upon every function call which means for example that a distance matrix or a Voronoi structure has to be calculated for every different set of solutions.

The new approach includes writing a specific QIC function tailored for an augmented quality indicator. It still derives the same values as the original approach, but by using less expensive operations, reducing the amount of function calls, or avoiding recalculations when possible.

ELSA algorithm:

Another slight modification to the ELSA algorithm (and its derivative ELSA-SR) is the inclusion of an if-statement to allow steps 10-12 in Algorithm 1 to be skipped. The operations described in these steps concern adding a generated child to the population, the calculation of QIC for every solution, and the removal of the least contributing solution. This occurs when an infeasible child is generated when the entire population already consists of feasible solutions. All the augmented quality indicators are designed in accordance that ELSA prioritises feasibility, so from the function value it can be determined that the child does not survive to the next generation.

2.4.2 QIC reformulation

Recall that ELSA and its variants only create one child in each generation. Using this information, some simplifications can be made on the functions for calculating the quality indicator contribution. Given that P is the population of the approximation set, p a solution in P, ind a quality indicator and L the target level set. Formula 16 (augmented quality indicator) in combination with Formula 1 (quality indicator contribution) is equivalent to the following:

$$\Delta_{ind^+}(\mathbf{p}, P) \leftarrow \Big(ind(P \cap L) - ind\big((P \backslash \{\mathbf{p}\}) \cap L\big)\Big) - \Big(penalty^{ind}(P \backslash L) - penalty^{ind}\big((P \backslash \{\mathbf{p}\}) \backslash L\big)\Big) \tag{20}$$

This format allows one to see more easily that some terms cancel each other when we have the knowledge of whether we are dealing with the removal of a feasible or infeasible solution $\mathbf{p}$.

In the case of feasible solution $\mathbf{p}$, we only calculate:

$$\Delta_{ind^+}(\mathbf{p}_{feasible}, P) \leftarrow ind(P \cap L) - ind\big((P \backslash \{\mathbf{p}_{feasible}\}) \cap L\big) \tag{21}$$

In the case of infeasible solution $\mathbf{p}$, we only calculate:

$$\Delta_{ind^+}(\mathbf{p}_{infeasible}, P) \leftarrow penalty^{ind}(P \backslash L) - penalty^{ind}\big((P \backslash \{\mathbf{p}_{infeasible}\}) \backslash L)\big) \tag{22}$$

Furthermore, Formula 22 is equivalent to:

$$\Delta_{ind^+}(\mathbf{p}_{infeasible}, P) \leftarrow penalty^{ind}\big(\{\mathbf{p}_{infeasible}\}\big) \tag{23}$$

If we go back to the penalty functions in Section 2.3, the following remain:

$$\Delta_{\mathrm{GI}^+}(\mathbf{p}_{infeasible}, P) = -\Big(\mathrm{diameter}(S) + (f(\mathbf{p}_{infeasible}) - \epsilon)\Big) \tag{24}$$

and

$$\Delta_{\mathrm{SPI}^+}(\mathbf{p}_{infeasible}, P) = \Delta_{\mathrm{ADI}^+}(\mathbf{p}_{infeasible}, P) = -\Big(f(\mathbf{p}_{infeasible}) - \epsilon\Big) \tag{25}$$

To determine whether Formula 21 or Formula 23 is to be used for the calculation of the quality indicator contribution, a check on whether the solution is feasible or infeasible is required.

2.4.3 Distance matrix

A similarity that is shared among the implementation of the GI^+ and SPI^+ indicators is that they rely on the calculation of a distance matrix.

More efficient distance matrix calculation:

Adapting MATLAB specialised matrix computations is more preferred over ad-hoc matrix operations for speed complexity reasons. Algorithm 3 describes how the Euclidean distance matrix is calculated in the former approach, where P is the matrix of the approximation set containing the solutions p in each column while the rows represent the dimensions. Note the use of double for-loops.

Algorithm 3 Original implementation to calculate Euclidean distance matrix

1: $distanceMatrix \leftarrow$ initialise matrix of size $|P| \times |P|$
2: **for** $i = 1 : |P|$ **do**
3: **for** $j = 1 : |P|$ **do**
4: $distanceMatrix_{i,j} \leftarrow \text{EuclideanDistance}(p_{*,i}, p_{*,j})$
5: **end for**
6: **end for**
7: **return** $distanceMatrix$

The new approach has reduced the amount of for-loops to only one that traverses over the n dimensions of the search space. It divides the Euclidean distance calculation in smaller pieces for each dimension to be processed and then recombines the results. The MATLAB function $[X, Y] = \texttt{meshgrid}(x, y)$ in step 3 of Algorithm 4 returns 2D grid coordinates based on the coordinates contained in vectors x and y: X is a matrix where each row is a copy of x, and Y is a matrix where each column is a copy of y. Laying X and Y over each other is essentially to simulate the same comparisons that is yield from the use of double for-loops and it sets up the structure of the distance matrix.

Algorithm 4 New implementation to calculate Euclidean Distance Matrix: `EuclDistFast`

1: $total \leftarrow 0$
2: **for** $i = 1 : n$ **do**
3: $[X, Y] \leftarrow \texttt{meshgrid}(p_{i,*}, p_{i,*})$
4: $total \leftarrow total + (X - Y)^2$
5: **end for**
6: $distanceMatrix \leftarrow \sqrt{total}$
7: **return** $distanceMatrix$

A time comparison between the two methods for calculating the distance matrix has been made in Section 2.4.7.

Distance Matrix Reuse:

With regards to the calculation of QIC: a population with one solution removed contains much overlap with the original population, which allows the re-use of data in incremental updating. The distance matrix of a set with one solution removed from the total population is the same as the distance matrix of the total population minus the row and column of the removed corresponding solution. In this case the distance matrix can be calculated once for P', the population of the approximation set that includes a child.

SPI^+ uses the distance matrix in a different way than the three GI^+ indicators. For the SPI^+ indicator, every single element in the distance matrix has a contribution to the calculation of the quality indicator value, while for the GI^+ indicators a distance matrix on the solution set is calculated to determine the Solution-Set Distances vector, which is composed from the smallest elements found in each column of the distance matrix (or row given that a distance matrix is symmetric, but for consistency we keep on saying the columns). The distance matrix in the GI^+ Indicators has its diagonal elements set to infinity for the reasons of ensuring that these will not be picked when looking for the smallest distance for the Solution-Set Distances vector.

The following method is used for retrieving the Solution-Set Distances from a population with one solution removed: We need the information of the smallest and second-smallest values found in each column of the distance matrix. The smallest values and second-smallest values including their indices in each column of the distance matrix are retrieved through linear search and stored. By comparing the saved indices of the minimum values with the index of the removed solution, it can be checked whether a Solution-Set Distance element belongs to a removed solution, so that the Solution-Set Distance element will be updated to the second minimum in the column.

2.4.4 Incremental update for GI^+

The GI_N^+ indicator bases its value on the minimal Solution-Set Distance in the distance matrix. If we remove a solution from the population set that is not one of the points that span the minimal Solution-Set Distance, the GI_N^+ value remains the same, and the QIC value will be 0. Therefore the only solutions worth checking are the points that span the minimal Solution-Set, which are easily retrieved through the indices in the distance matrix where the minimal values are found. There always exist at least two of such solutions, and in the usual case there are exactly two of them, unless there exist non-unique minimal Solution-Set Distances or there exist duplicate solutions.

There have not been any specific incremental updates derived for the GI_Σ^+ and GI_Π^+ indicators, on top of the mentioned distance matrix reuse described in Section 2.4.3.

2.4.5 Incremental update for SPI^+

The complexity reduction for SPI^+ is achieved by applying the derivation described by Ulrich in the NOAH algorithm, see [15] for the exact details. Recall that $C = \left(\exp(-\theta \cdot d_{p_i,p_j})\right)^{-1}$, $\forall i, j = 1, \ldots, \mu$ for a positive parameter θ that scales the distance matrix. Formula 26 describes the more efficient quality indicator contribution for SPI^+, where P is the feasible subset of an approximation set and p_k a solution of index k that is removed from P.

$$\Delta_{\mathrm{SPI}^+}(p_k, P) = \frac{\left(\sum_{i=1}^{\mu} c_{i,k}\right)^2}{c_{k,k}} \tag{26}$$

A possible unwanted side-effect from using this implementation is that it does not guarantee the Twin property of Solow Polasky. However the calculated QIC value from Formula 26 for non-duplicates are identical to the results from of applying the traditional Solow Polasky Indicator described in Formula 10 for QIC calculation. Therefore an extra system that takes duplicates into account is required to remain faithful to the description of the Solow Polasky Indicator.

The handling of duplicates is described next. Only feasible duplicates should be treated differently, therefore the description is specifically about the calculation of the SPI diversity.

1. The SPI diversity is first calculated over the feasible unique subset of the population with the extra child. In other words, the 'clones' are removed from approximation set. A distinction is made between 'clones' and 'duplicates', where clones are seen as extra copies from some original solution, while duplicates are solutions that are identical to another in the approximation set.

2. When the QIC calculation involves the removal of a feasible duplicate solution, its resulting QIC value in regards to the SPI diversity is 0 due to the Twin property.

3. What remains is the QIC calculation for the removal of a feasible non-duplicate.

2.4.6 Incremental update for ADI$^+$

Incremental update for ADI$^+$ is applied through reusing Voronoi cells that have not been changed. The implementation is specifically written for the 2D version of ADI$^+$. For the QIC calculation, the following changes can be noticed in the Voronoi Diagram of the new population where one solution has been removed: the Voronoi cells that were adjacent to the cell corresponding to the removed solution change shape, while the rest of the Voronoi cells remain the same (shown in Figure 1a and Figure 1b). Therefore, the triangle contributions from the changed Voronoi cells should be recalculated. The cumulative triangle contributions from one Voronoi Cell in the total population is stored in a table so that it can be retrieved if the triangle contributions from this Voronoi cell can be reused. There may still exist some overlap between some triangles in the original Voronoi (shown in Figure 1c and Figure 1d), so that even more recalculations can be avoided if it is known which triangles have changed shape, but it requires extra work to implement the system that allows such bookkeeping, so this is not taken into consideration.

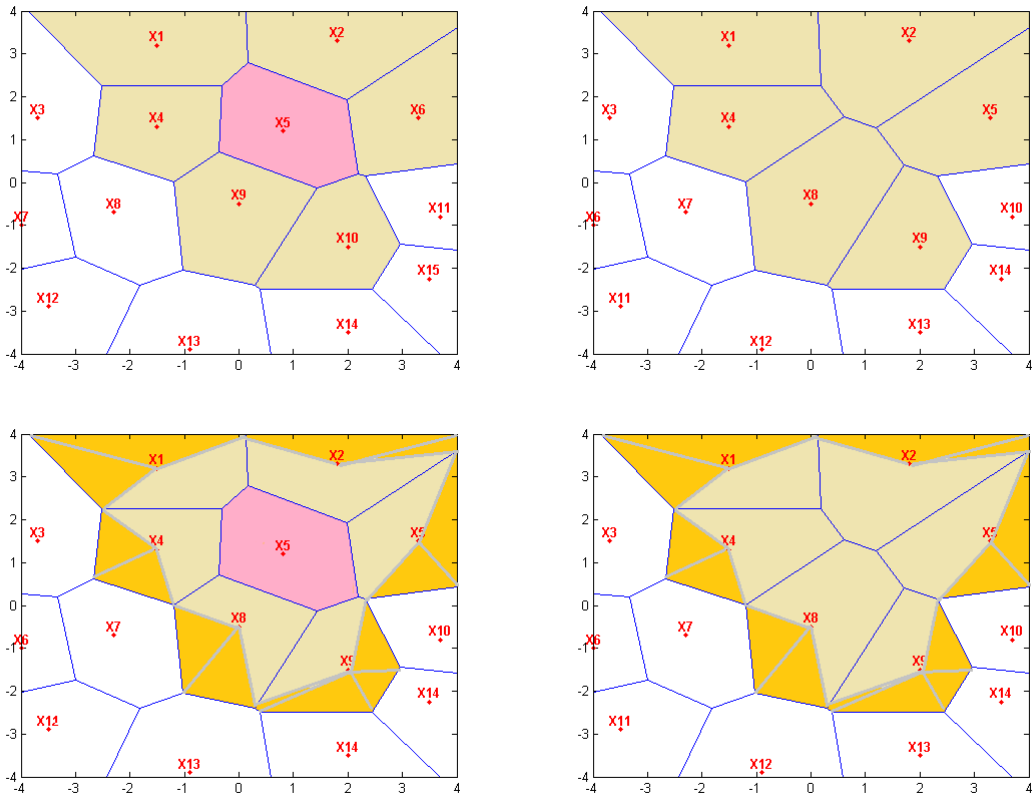

Figure 1: An example to illustrate the change of the structure of Voronoi cells when removing one solution. Read the images from left to right as labeled with alphabetical ordering.
(a) This is the Voronoi diagram of the total population that consists of 15 solutions labeled as $X1 \ldots X15$. Suppose we remove solution $X5$. Its Voronoi cell is marked in pink and its neighbouring cells in brown.
(b) This is the resulting Voronoi diagram after removing solution $X5$. The white non-neighbouring cells remained the same.
(c) and (d) The triangles in the Voronoi cells that have not changed shape are depicted in orange.

Both the implementation in the original approach and new approach for ADI$^+$ uses the default MATLAB functions designed to calculate the Voronoi topology for the population set. First the old approach for 2D ADI$^+$ is described, followed by the new approach.

The first step is ensuring that after applying Voronoi, the cells are bounded by the reference set by using the mirror approach. The population set P is mirrored for the four directions of top, left, right, and bottom, which essentially creates a new population P^+ with 5 times as many solutions and Voronoi is applied to P^+. We only care about the solutions and their cells in the original population, and the mirrored solutions serve no further use anymore.

In the original 2D ADI implementation, the MATLAB function `voronoi()` is used, which fetches the edges that make up a Voronoi cell. The algorithm then traverses through all the edges and first checks whether this edge resides within the search space. If this holds, then it determines to which Voronoi cell(s) each edge belongs to by performing nearest neighbour search on all the solutions in P with the centre of the edge as reference. If the edge falls in the border of the reference set, then it finds one nearest neighbour solution, while in other cases it finds two neighbours. Triangle contribution is calculated over the edge and its nearest neighbour solution.

The mirrored population approach remains to be used in the new approach. The major differ-

ence is that it now uses the MATLAB function `voronoin()` which fetches the vertices of the Voronoi cells in a cell array data structure. A single cell array that corresponds to a Voronoi cell consists of elements, where each element corresponds to one vertex, and adjacent elements suggest that an edge connects them, with an edge as well between the first and last element in the array. The new algorithm traverses through the Voronoi cell of each solution in P, and cuts it up into triangles to calculate the contribution. With this new method, the search of nearest neighbour solutions when encountering an edge in the old method is avoided.
`delauney()` is the MATLAB function to perform Delauney triangulation on P^+, where the circumcircle of each triangle contains no other points than those three from the triangle (delauney can also be considered as the dual graph of Voronoi). This information is used to construct a list of all the nearest neighbours for each Voronoi cell.
For the incremental update of ADI$^+$ when removing a solution, Voronoi is calculated again over a mirrored set that only includes the first and second degree neighbours of the removed solution. The second degree neighbours are included so that the shape of the cluster of the first degree neighbouring cells is preserved.

The ADI$^+$ indicator also has the Twin property and uses the same duplicate handling scheme described for the SPI$^+$ indicator in Section 2.4.5.

2.4.7 Experiments and results on time complexity

The runtimes of the old implementation in ELSA have been compared with the current implementation. The first test compares the previous implementation of generating the distance matrix with the current method, and its results are depicted in Table 1. The second test compares the previous implementation of the QIC calculation for a specific quality indicator with the current methods, and its results are depicted in Table 2.

The MATLAB functions `tic` and `toc` are used for the measurement of time in seconds. For the distance matrix comparisons, a random population of size $\mu = 100$ is generated which is assumed to contain feasible and unique solutions exclusively. For the QIC comparisons, the size of the 2D population is set to 101 to recreate the situation of a population with an extra child $(\mu + 1)$. All the randomly generated values for the population are uniformly drawn from the range of $[0, 1]$. The results are based on the average runtime from 1000 runs.

The results in Table 1 reflect that the new method for calculation is faster than the original method for all tested dimensions 2D, 10D, 30D, and 100D. The new method for generating a distance matrix is approximately 50 times faster than the previous method for a 2D population.

	old	**current**
2D	$1.2 \cdot 10^{-2}$	$2.4 \cdot 10^{-4}$
10D	$1.3 \cdot 10^{-2}$	$6.7 \cdot 10^{-4}$
30D	$1.4 \cdot 10^{-2}$	$2.0 \cdot 10^{-3}$
100D	$1.7 \cdot 10^{-2}$	$5.7 \cdot 10^{-3}$

Table 1: The runtimes of the distance matrix implementations for various dimensions.

Table 2 depicts the improvements of using the new methods for QIC calculation over the previous method. The runtime for the calculation of QIC for any tested quality indicator takes approximately 1 second in the original method, where ADI$^+$ takes the longest with almost 2 seconds. Replacing the original method of distance matrix calculation yields decent time improvement, but most significant improvements are achieved through combining all of the

operations described for incremental updating. With the current methods QIC with GI^+_N is approximately 3408 times faster, QIC with GI^+_Σ 788 times, QIC with GI^+_Π 1197 times, QIC with SPI^+ 215 times, and QIC with ADI^+ 8 times. The largest gains are achieved for the calculation of QIC with Augmented Gap Indicators, while the calculation of QIC with ADI^+ is hardly improved even with the new method.

	old	**EuclDistFast**	**current**
$\mathbf{GI}^+_N$	$9.1 \cdot 10^{-1}$	$1.8 \cdot 10^{-2}$	$2.7 \cdot 10^{-4}$
$\mathbf{GI}^+_\Sigma$	$9.5 \cdot 10^{-1}$	$2.1 \cdot 10^{-2}$	$1.2 \cdot 10^{-3}$
$\mathbf{GI}^+_\Pi$	$9.4 \cdot 10^{-1}$	$2.1 \cdot 10^{-2}$	$7.8 \cdot 10^{-4}$
$\mathbf{SPI}^+$	$1.2 \cdot 10^{0}$	$2.6 \cdot 10^{-1}$	$5.5 \cdot 10^{-3}$
$\mathbf{ADI}^+$	$1.8 \cdot 10^{0}$	-	$2.2 \cdot 10^{-1}$

Table 2: The runtimes of the QIC implementations. The first column with label **old** contains the results from using the generic QIC function with a specific QI. The second column with label **EuclDistFast** contains the results yield from a method that is very similar to the original, except the distance matrix is generated according to the current method. The third column with the label **current** is the result from using a specifically written QIC function dedicated for a QI, which is the accumulation of every single change that is described in Section 2.4.

3 Black box level set approximation problem benchmark suite

One of the aims of this research is to devise a high dimensional level set benchmark suite that is capable of evaluating the various strengths and weaknesses of an Indicator-Based Evolutionary Algorithm that is designed for the purpose of solving the Level Set Approximation Problem. The guidelines for the design of a level set benchmark are outlined in Section 3.1. As the ELSA algorithm has only been tested on level set benchmarks of up to two dimensions in previous research, the dimensions to be covered by the black box level set benchmarks are: 2D, 3D, 10D, and 30D. Visualisation of higher dimensional level sets is difficult, but 3D can still be presented in a human readable format to provide some insights how shapes evolve when adding one extra dimension to 2D.

The black box level set benchmarks consist of geometrical shapes generated from taking the sublevel sets of a mathematical function. The general form of a level set L has already been defined in Formula 1, but is restated below.

$$L = \{\mathbf{x} \in S | f(\mathbf{x}) \leq \epsilon\}$$

For a given vector $\mathbf{x}$ in the Cartesian coordinate system, let $f(\mathbf{x})$ be the level set shape function, S the search space, and ϵ the chosen threshold constant. The search space S is a compact domain that is defined by the box constraints described in Formula 27.

$$S = [\mathbf{x}_1^{min}, \mathbf{x}_1^{max}] \times \cdots \times [\mathbf{x}_n^{min}, \mathbf{x}_n^{max}] \tag{27}$$

The benchmarks are divided into two categories: simple and complex shapes. Simple shapes refer to basic mathematical geometrical objects, such as generalisations of spheres and shapes based on the sphere function. Certain properties of a sphere are well documented in literature, which allows for accurate calculation of the volumes of the level sets.

Simple shape benchmarks:

1. Lamé (Section 3.2)
2. Ellipsoid (Section 3.3)
3. Hollow Sphere (Section 3.4)
4. Double Sphere (Section 3.5)

Complex shapes refer to engineering relevant shapes described by functions more complex than those found in the simple shapes benchmark category. The shape functions that are used for the complex shape benchmarks are taken from mathematical functions in multimodal optimisation problems. They are used to show the performance of the Indicator-Based Evolutionary Algorithms on more realistic landscapes in terms of practical test problems.

Complex shape benchmarks:

1. Branke's Multipeak (Section 3.6)
2. Rastrigin (Section 3.7)
3. Schaffer (Section 3.8)

4. Vincent (Section 3.9)

The multimodal optimisation problems typically come with defined search spaces and can be served as black box benchmarks where the same experiments (with different algorithms) can be replicated. However, level set approximation problem is not the same as multimodal optimisation problem, and some liberties have been taken with the search space. Some are kept the same while others have been adapted differently to specify a level set with certain properties.

3.1 Guidelines for the level set benchmark design

The main guideline for designing the level set benchmarks in that the level set should strike a balance between difficulty and solvability, where the population on the level set should be able to be visualised.

The difficulty of the level set benchmark depends on the level set shapes and their sizes. In general objects that are small are more difficult to locate, which therefore leads to an increase in the difficulty for the benchmark. The same applies for thin parts or acute angles in an object, which adds a challenge for diversity optimisation after finding a whole, feasible population.

Visualisation of the approximation set on the level set is necessary, so that the quality of coverage in a level set can be determined by human judgment. By having a large area or volume percentage for the level set inside the search space, by for instance choosing a high threshold value, it allows for more freedom to settle the solutions and the slightest differences between populations are more pronounced for the viewer.

If a level set has disjoint parts, it is expected that the algorithm should settle at least one solution on each disjoint part, unless the approximation set size is smaller than the amount of disjoint parts. Many of the chosen complex shape level sets have an exponential growth in the amount of disjoint components when increasing the dimensionality of the level set benchmark. Having large distances between the disjoint parts can serve as a way to test the global search capabilities of the algorithm.

In the end, the difficulty of the level set benchmarks has been tailored accordingly to the philosophy of the ELSA algorithm where it is preferred that the entire approximation set is feasible. This provides another argument for having a level set with large volume besides for visualisation reasonings, as the feasible set can be found easier to leave room for the diversity optimisation with the rest of the evaluation budget.

The visualisation of the level set:

The rendering of the level sets is done by first taking an evenly spaced grid of points $\mathbf{x}$ in the search space where the objective function $f(\mathbf{x})$ for each point is calculated. This is followed by the generation of a filled contour plot of $f(\mathbf{x})$ with only one contour line that is set to the value of ϵ. The area where $f(\mathbf{x}) < \epsilon$ holds is depicted as grey and the area where $f(\mathbf{x}) > \epsilon$ holds is depicted as white. In the visualisation for the 2D level set benchmarks, the $f(\mathbf{x}) = \epsilon$ level set is outlined as black, but for the case of 3D the outline of the level set is turned off and is therefore depicted as white (indistinguishable from parts not belonging to the level set). Another difference is that the visualisation of 3D benchmarks has the level sets as transparent.

The refinement of the visualisation depends on the amount of points in the grid and due

to memory constraints intricate details may fail to be rendered. The resolution of one axis for the 2D level set benchmarks consists of 2000 grid points. Respectively for the 3D level set benchmarks this number is set to 25 for Lamé, Ellipsoid, Hollow Sphere, Double Sphere, and Branke's Multipeak, but 200 for Rastrigin, Schaffer and Vincent.

The approximation of the level set size:

The level set sizes for the complex level set benchmarks are gained through approximation. A uniform grid is taken over the search space to divide it up into cubes of equal size (in the case of 2D: replace the term cube with square and the term volume with area), where the sum of volume is taken of the cubes that 'belongs' to the level set. This is decided through taking the centre of each cube as the reference point, which is to be checked whether this point belongs to the level set and consequently treat the entire cube as part of it. One million squares have been used for the approximation of the 2D level set benchmarks. While 8 million cubes have been used for the 3D level set benchmarks, so that the resolution for 2D is higher than for 3D. The approximation had not been done for 10D and 30D because these require too many points in a grid for accurate approximation.

3.2 Lamé shapes

A Lamé curve, which is named after the French mathematician Gabriel Lamé is a geometrical figure that is defined in 2D space [18]. Given a vector $\mathbf{x}$, the definition of a Lamé curve in the Cartesian coordinate system is described in Formula 28.

$$\left|\frac{x_1}{a}\right|^p + \left|\frac{x_2}{b}\right|^p = 1 \tag{28}$$

The positive, continuous numbers a and b are semi-diameters that scale the size of the Lamé curve along each axis, so that for example x_1 falls in the range of $[-a, a]$ in Formula 28. p is a positive, continuous number that controls the curvature of the shape.

We use the Lamé function for higher dimensions in our benchmarks and simplify it by using one value r for all the semi-diameters. This leads to Formula 29.

$$f_{Lamé}(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|} \left|\frac{x_i}{r}\right|^p - 1 \tag{29}$$

$$L_{Lamé} = \left\{\mathbf{x} \in [-3, 3]^n \middle| f_{Lamé}(\mathbf{x}) \leq \epsilon\right\} \tag{30}$$

The parameter values used in the Lamé level set benchmark in Formula 30 for each respective dimension are shown in Table 3.

Dimension	**r**	**p**	ϵ
2D	2	0.5	0
3D	3	0.5	0
10D	3	0.5	0
30D	3	0.5	0

Table 3: The parameter settings for the Lamé level set benchmarks

These parameters lead to a non-disjoint level set consisting of a round symmetrical shape that has acute edges. This level set is located at the centre of the search space. With $p = 0.5$,

the Lamé shapes can be considered a specific form of superellipse in 2D and superellipsoid in dimensions higher than 2. Figure 2 depicts the 2D and 3D Lamé level set benchmarks.

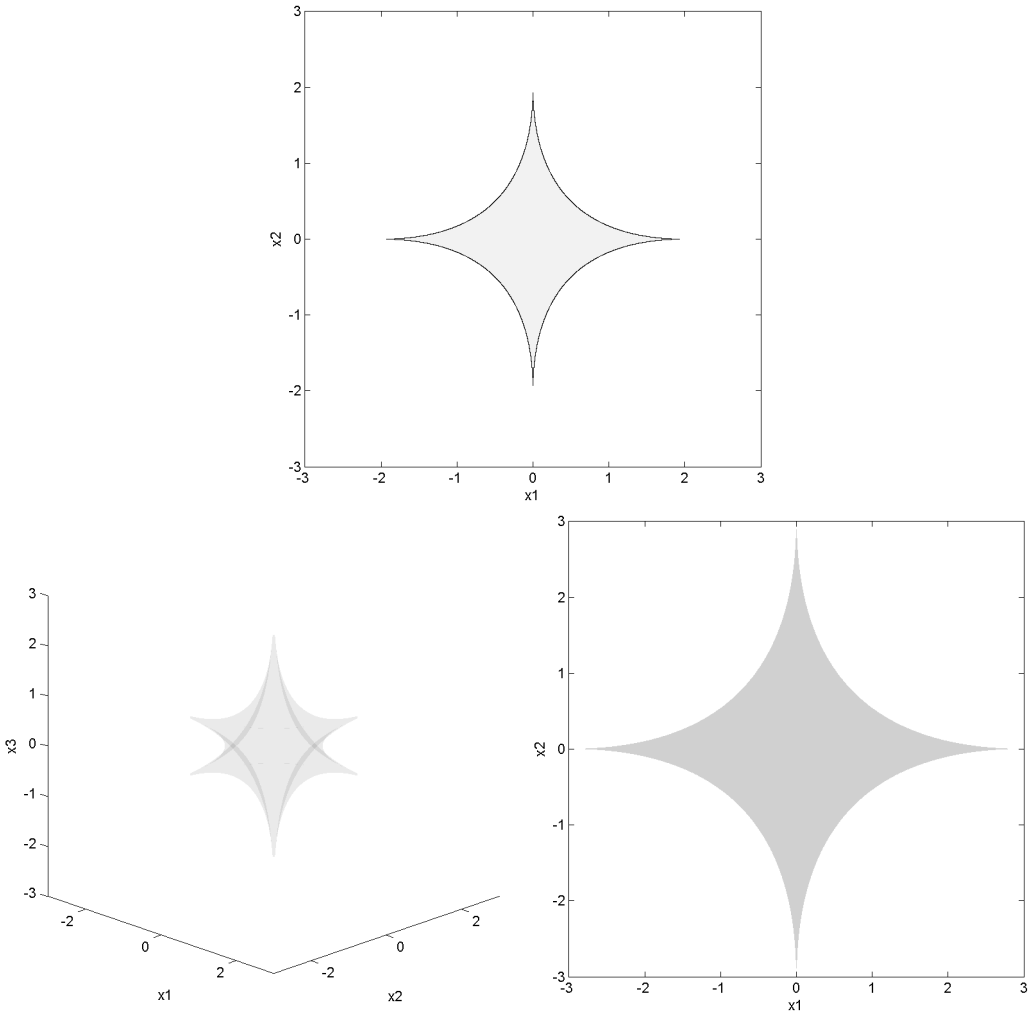

Figure 2: At the top: 2D Lamé level set benchmark; At the bottom left: 3D Lamé level set benchmark; At the bottom right: intersection from the 3D level set benchmark where $x_3 = 0$

The value of r has been increased in the 3D shape in comparison to 2D, because the level set volume divided by search space ratio becomes much smaller when going to higher dimensions for the same parameters. To ensure that the search space covers the entire Lamé level set, when preserving the same axis ranges for each dimension, the maximum allowed value for r is 3.
The area of the 2D Lamé level set benchmark defined in Formula 32 is derived from the known formula for the area of a superellipse defined in Formula 31 [18]. The formula for superellipse is the same as Lamé curve in Formula 28. The 3D Lamé function can be derived from the 3D superellipsoid function in Formula 34 when assuming $p = \frac{2}{p_1} = \frac{2}{p_2}$ and $r = a = b = c$. Formula 35 describes the function for calculating the volume of a 3D superellipsoid [6], which is rewritten into Formula 36 for calculating the volume of the 3D Lamé function. The level set to search space ratio for 2D and 3D Lamé level set benchmark are depicted in Table 4.

area 2D Lamé	2.6667
area/search space ratio	$7.4074 \cdot 10^{-2}$
volume 3D Lamé	2.4000
volume/search space ratio	$1.1111 \cdot 10^{-2}$

Table 4: The table depicts the level set to search space ratio for the 2D and 3D Lamé level set benchmark.

$$Area_{Superellipse} = \frac{4^{1-\frac{1}{p}} ab \cdot \sqrt{\pi} \cdot \Gamma(1+\frac{1}{p})}{\Gamma(\frac{1}{2}+\frac{1}{p})} \tag{31}$$

$$Area_{\text{Lamé}_{2D}} = \frac{4^{1-\frac{1}{p}} r^2 \cdot \sqrt{\pi} \cdot \Gamma(1+\frac{1}{p})}{\Gamma(\frac{1}{2}+\frac{1}{p})} \tag{32}$$

$$\Gamma(n) = (n-1)! \tag{33}$$

$$Superellipsoid_{3D}(x,y,z) = \left(\left(\frac{x}{a}\right)^{\frac{2}{p_2}} + \left(\frac{y}{b}\right)^{\frac{2}{p_2}}\right)^{\frac{p_2}{p_1}} + \left(\frac{z}{c}\right)^{\frac{2}{p_1}} \tag{34}$$

$$Volume_{Superellipsoid_{3D}} = 2abcp_1p_2 \cdot \beta\Big(\frac{p_1}{2}+1, p_1\Big) \cdot \beta\Big(\frac{p_2}{2}, \frac{p_2}{2}\Big) \tag{35}$$

$$Volume_{\text{Lamé}_{3D}} = \frac{8r^3}{p^2} \cdot \beta\Big(\frac{1}{p}+1, \frac{2}{p}\Big) \cdot \beta\Big(\frac{1}{p}, \frac{1}{p}\Big) \tag{36}$$

$$\beta(n,m) = \frac{\Gamma(m) \cdot \Gamma(n)}{\Gamma(m+n)} \tag{37}$$

3.3 Ellipsoid

Given a vector $\mathbf{x}$, the definition of an ellipsoid in the Cartesian coordinate system is described in Formula 38, where $\mathbf{C}$ is a vector that defines the length of the semi-principal axes of the ellipsoid. Formula 39 describes the Ellipsoid level set benchmark.

$$f_{Ellipsoid}(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|} \left(\frac{x_i}{C_i}\right)^2 - 1 \tag{38}$$

$$L_{Ellipsoid} = \left\{\mathbf{x} \in [-3,3]^n \middle| f_{Ellipsoid}(\mathbf{x}) \leq \epsilon\right\} \tag{39}$$

The parameter values used in the Ellipsoid level set benchmark in Formula 39 for each respective dimension are shown in Table 5.

Dimension	**C**	ϵ	Notes
2D	(1, 2)	0	
3D	(1, 2, 2.5)	0	
10D	(1, 2, 2.5, 1, 2, 2.5, 1, 2, 2.5, 1)	0	
30D	(1, 2, 2.5, ..., 1, 2, 2.5)	0	**C** is a repetition of 1, 2, 2.5 as values

Table 5: The parameter settings for the Ellipsoid level set benchmarks

The parameters are chosen in such a way that the Ellipsoid function creates a non-disjoint level set with symmetry around its axes but the semi-principal axes should have sufficient variation

so that the ellipsoids do not resemble spheres. The geometrical shape has no sharp edges of any kind. This level set is also located at the centre of the search space. Figure 3 depicts the 2D and 3D Ellipsoid level set benchmarks.

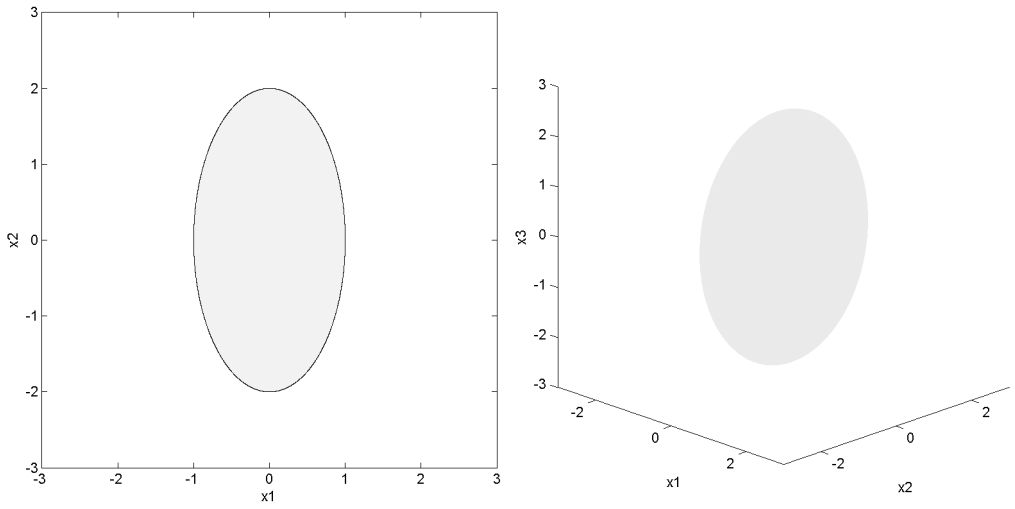

Figure 3: On the left: 2D Ellipsoid level set benchmark; On the right: 3D Ellipsoid level set benchmark

The volume of an nD ellipsoid is calculated with Formula 41, which is a generalization of the formula to calculate the volume of a nD sphere with radius r denoted in Formula 40 [17].

$$Volume_{Sphere_{nD}}(r) = \frac{2\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)} \cdot \frac{1}{n} \cdot r^n \tag{40}$$

$$Volume_{Ellipsoid_{nD}}(\mathbf{C}) = \frac{2\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)} \cdot \frac{1}{n} \cdot \prod_{i=1}^{|\mathbf{C}|} C_i \tag{41}$$

The volume of the 2D, 3D, 10D and 30D Ellipsoid level set benchmarks and their respective ratio to the search space that are calculated with Formula 41 are shown in Table 6.

area 2D Ellipsoid (Ellipse)	6.2832
area/search space ratio	$1.7453 \cdot 10^{-1}$
volume 3D Ellipsoid	20.9440
volume/search space ratio	$9.6963 \cdot 10^{-2}$
volume 10D Ellipsoid	318.7705
volume/search space ratio	$5.2719 \cdot 10^{-6}$
volume 30D Ellipsoid	214.0171
volume/search space ratio	$9.6808 \cdot 10^{-22}$

Table 6: The table depicts the level set to search space ratio for the 2D, 3D, 10D and 30D Ellipsoid level set benchmark.

3.4 Hollow Sphere

The Hollow Sphere level set benchmark is described in Formula 43, which is created through fattening the sphere function with ϵ on both sides. Formula 42 depicts the sphere function of radius r with an absolute operation wrapped around it, so that it can be fattened.

$$f_{HollowSphere}(\mathbf{x}) = \left| \sqrt{\sum_{i=1}^{|\mathbf{x}|} x_i^2} - r \right| \tag{42}$$

$$L_{HollowSphere} = \left\{ \mathbf{x} \in [-3,3]^n \middle| f_{HollowSphere}(\mathbf{x}) \leq \epsilon \right\} \tag{43}$$

The parameter values used in the Hollow Sphere level set benchmark in Formula 43 for each respective dimension are shown in Table 7.

Dimension	**r**	ϵ
2D	1.5	0.3
3D	1.5	0.3
10D	1.5	0.3
30D	1.5	0.3

Table 7: The parameter settings for the Hollow Sphere level set benchmarks

The parameters are chosen in such a way that the Hollow Sphere function creates a non-disjoint level set with symmetry around its axes. This shape has a hole in it so it has an overall characteristically thinner surface compared to the other simple level set benchmarks. Figure 4 depicts the 2D and 3D Hollow Sphere level set benchmarks.

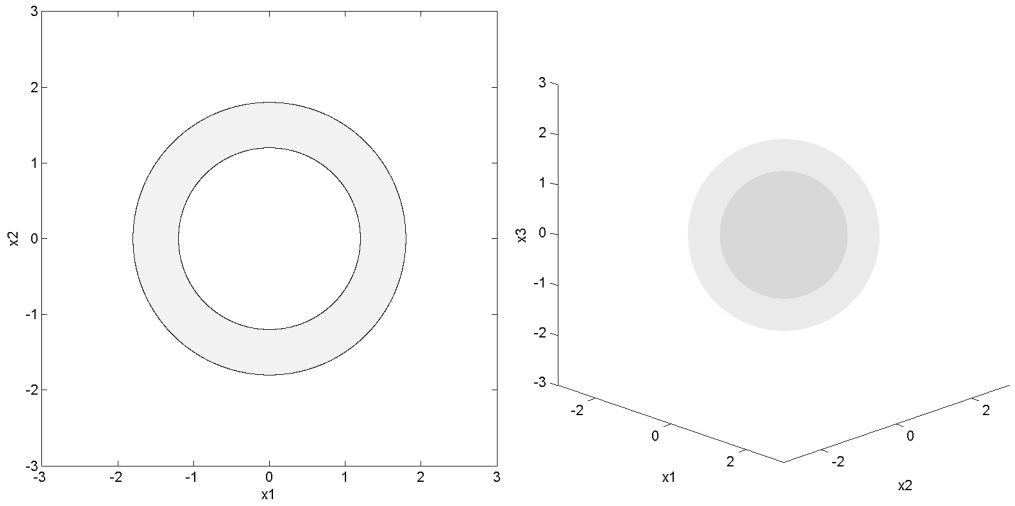

Figure 4: On the left: 2D Hollow Sphere level set benchmark; On the right: 3D Hollow Sphere level set benchmark

The volume of an nD Hollow Sphere is calculated with Formula 44.

$$Volume_{HollowSphere_{nD}}(r,\epsilon) = \frac{2\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)} \cdot \frac{1}{n} \cdot \left((r+\epsilon)^n - (r-\epsilon)^n \right) \tag{44}$$

The volume of the 2D, 3D, 10D and 30D Hollow Sphere level set benchmarks and their respective ratio to the search space that are calculated with Formula 44 are shown in Table 8.

area 2D Hollow Sphere	5.6549
area/search space ratio	$1.5708 \cdot 10^{-1}$
volume 3D Hollow Sphere	17.1908
volume/search space ratio	$7.9587 \cdot 10^{-2}$
volume 10D Hollow Sphere	894.7378
volume/search space ratio	$1.4797 \cdot 10^{-5}$
volume 30D Hollow Sphere	997.5194
volume/search space ratio	$4.5122 \cdot 10^{-21}$

Table 8: The table depicts the level set to search space ratio for the 2D, 3D, 10D, and 30D Hollow Sphere level set benchmark.

3.5 Double Sphere

Formula 46 describes the Double Sphere level set benchmark, which consists of two spheres as defined in Formula 45. Both the spheres share the same radius of length r and their centres are determined by the positive constant c so that one centre is located at $(-c, \ldots, -c)^n$ and the other at $(c, \ldots, c)^n$. There exists a bound for the radius to guarantee the condition that the two spheres do not touch each other.

$$f_{DoubleSphere}(\mathbf{x}) = \left(\sqrt{\sum_{i=1}^{|\mathbf{x}|}(x_i + c)^2} - r\right) \cdot \left(\sqrt{\sum_{i=1}^{|\mathbf{x}|}(x_i - c)^2} - r\right) \quad \text{under the condition: } r < \frac{1}{2} \cdot \sqrt{|\mathbf{x}| \cdot (2c)^2} \tag{45}$$

$$L_{DoubleSphere} = \left\{\mathbf{x} \in [-3, 3]^n \middle| f_{DoubleSphere}(\mathbf{x}) \leq \epsilon\right\} \tag{46}$$

The parameter values used in the Double Sphere level set benchmark in Formula 46 for each respective dimension are shown in Table 9.

Dimension	**r**	**c**	ϵ
2D	1	1	0
3D	1	1	0
10D	1	1	0
30D	1	1	0

Table 9: The parameter settings for the Double Sphere level set benchmarks

The parameters are chosen in such a way that the Double Sphere function creates a disjoint level set consisting of two spheres that are located diagonally from each other in the search space. As the spheres are of equal size, the benchmark can serve as a test whether the algorithm can evenly spread out its population within the two spheres. Figure 5 depicts the 2D and 3D Double Sphere level set benchmarks.

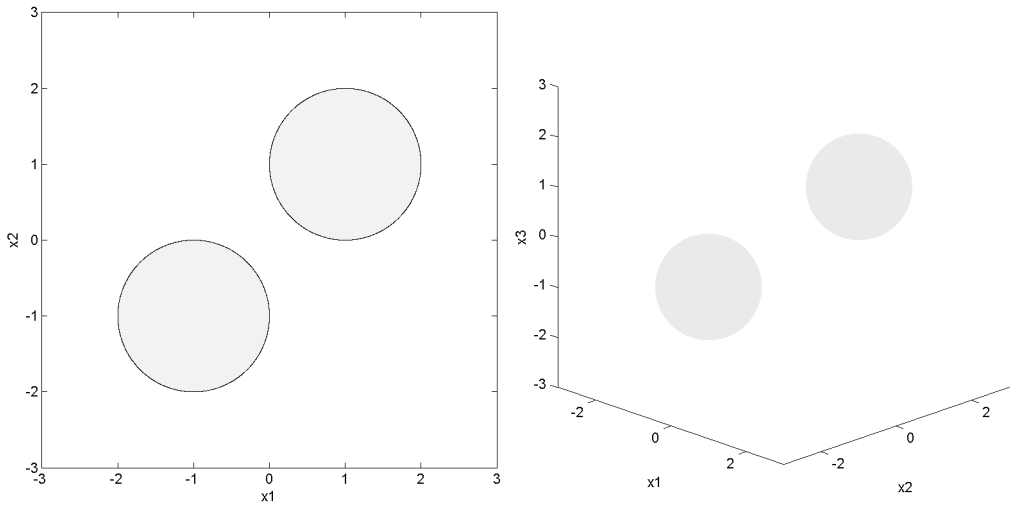

Figure 5: On the left: 2D Double Sphere level set benchmark; On the right: 3D Double Sphere level set benchmark

The volume of the nD Double Sphere level set is calculated with Formula 47.

$$Volume_{DoubleSphere_{nD}}(r) = 2 \cdot \frac{2\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)} \cdot \frac{1}{n} \cdot r^n \tag{47}$$

The volume of the 2D, 3D, 10D, and 30D Double Spheres level set benchmarks and their respective ratio to the search space that are calculated with Formula 47 are shown in Table 10.

area 2D Double Spheres	6.2832
area/search space ratio	$1.7453 \cdot 10^{-1}$
volume 3D Double Spheres	8.3776
volume/search space ratio	$3.8785 \cdot 10^{-2}$
volume 10D Double Spheres	5.1003
volume/search space ratio	$8.4350 \cdot 10^{-8}$
volume 30D Double Spheres	$4.3831 \cdot 10^{-5}$
volume/search space ratio	$1.9826 \cdot 10^{-28}$

Table 10: The table depicts the level set to search space ratio for the 2D, 3D, 10D, and 30D Double Sphere level set benchmark.

3.6 Branke's Multipeak

Formula 49 describes the Branke's Multipeak level set benchmark. This benchmark has been used in the previous ELSA research from Emmerich et al. [5] and its definition is also found in [7] where Kruisselbrink made a compilation of benchmarks suitable for testing evolution strategies on robust optimisation.

$$\begin{aligned} f_{BrankeMultipeak}(\mathbf{x}) &= \frac{1}{n} \sum_{i=1}^{n} \Big(c - g(x_i) \Big) \\ g(x_i) &= \begin{cases} -(x_i + 1)^2 + 1 & \text{if } -2 \leq x_i < 0 \\ c \cdot 2^{-8|x_i - 1|} & \text{if } 0 \leq x_i \leq 2 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{48}$$

$$L_{BrankeMultipeak} = \Big\{ \mathbf{x} \in [-2, 2]^n \Big| f_{BrankeMultipeak}(\mathbf{x}) \leq \epsilon \Big\} \tag{49}$$

The parameter values used in the Branke Multipeak level set benchmark in Formula 49 for each respective dimension are shown in Table 11.

Dimension	**c**	ϵ
2D	1.3	0.6
3D	1.3	0.4

Table 11: The parameter settings for the Branke's Multipeak level set benchmarks

The configuration for the 2D Branke's Multipeak benchmark including its search space size is chosen the same as the benchmark used in [5]. The local optima of Branke's Multipeak function are located at $(m, \ldots, m)$ where $m = -1$ or $m = 1$ holds and the global optima is located at (1,...,1). This corresponds to 2^n disjoint level set parts in the nD benchmark if the value for ϵ is chosen sufficiently small. This criteria is satisfied for the 2D and 3D Branke's Multipeak level set benchmarks, however the same does not necessarily hold true for higher dimensions, because the values for ϵ were determined through the visualisation of the level set. The Branke's Multipeak level set is not defined for dimensions higher than 3, because due the way Formula 48 is written it is not always possible to find a combination of values for c and ϵ that corresponds to exactly 2^n disjoint level set parts in the benchmark. When ϵ is set too low, some level set components are not expressed at all in the benchmark, while when ϵ is set too large the level set components conjoin.
Figure 6 depicts the 2D and 3D Branke's Multipeak level set benchmarks. The disjoint level set parts come in varying shapes, which include round objects and objects with pointiness, where the shape located at $(1, \ldots, 1)$ is even reminiscent of a Lamé shape.

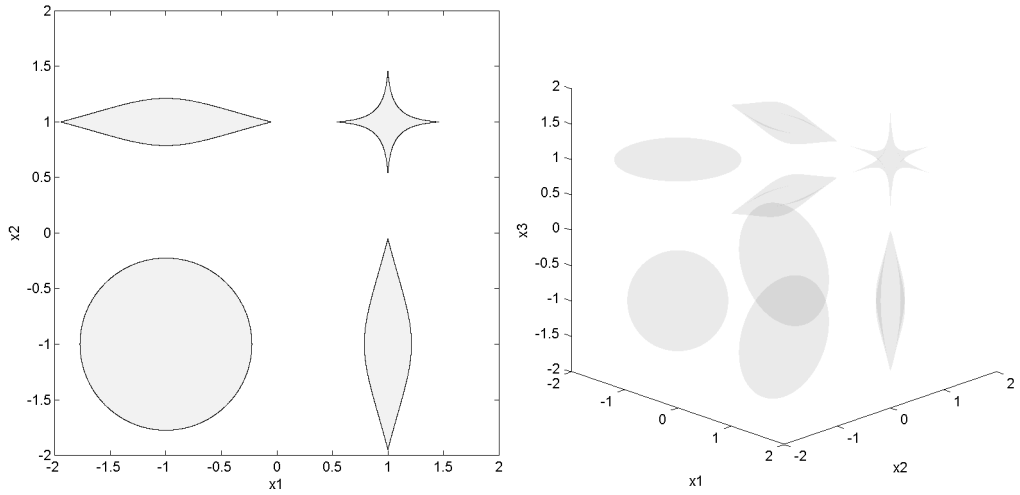

Figure 6: On the left: 2D Branke's Multipeak level set benchmark; On the right: 3D Branke's Multipeak level set benchmark

The approximated volume of the 2D and 3D Branke's Multipeak level set benchmarks and their respective ratio to the search space are shown in Table 12.

area 2D Branke's Multipeak	2.9736
area/search space ratio	$1.8585 \cdot 10^{-1}$
volume 3D Branke's Multipeak	1.4207
volume/search space ratio	$2.1980 \cdot 10^{-2}$

Table 12: The table depicts the level set to search space ratio for the 2D and 3D Branke's Multipeak level set benchmark.

3.7 Rastrigin

Formula 51 describes the Rastrigin level set benchmark. The Rastrigin function defined in Formula 50 [7] is a multimodal function with a large amount of local optima.

$$f_{Rastrigin}(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos(2\pi x_i) \right) \tag{50}$$

$$L_{Rastrigin} = \left\{ \mathbf{x} \in [-4.5, 4.5]^n \middle| f_{Rastrigin}(\mathbf{x}) \leq \epsilon \right\} \tag{51}$$

The parameter values used in the Rastrigin level set benchmark in Formula 51 for each respective dimension are shown in Table 13.

Dimension	ϵ
2D	29
3D	29
10D	29
30D	29

Table 13: The parameter settings for the Rastrigin level set benchmarks

The Rastrigin level set covers up a large space of the search space which makes it relatively easy to find feasible solutions for this problem. This benchmark has a large amount of disjoint level set parts, which serves as a test for the algorithm to locate them all. The 2D Rastrigin level set benchmark with $\epsilon = 29$ contains 40 disjoint level set parts on the rim. Lowering the threshold value ϵ can lead to the disintegration of the larger level set centre piece into smaller parts and the disappearance of the small disjoint level set parts on the rim. Figure 7 depicts the 2D and 3D Rastrigin level set benchmarks.

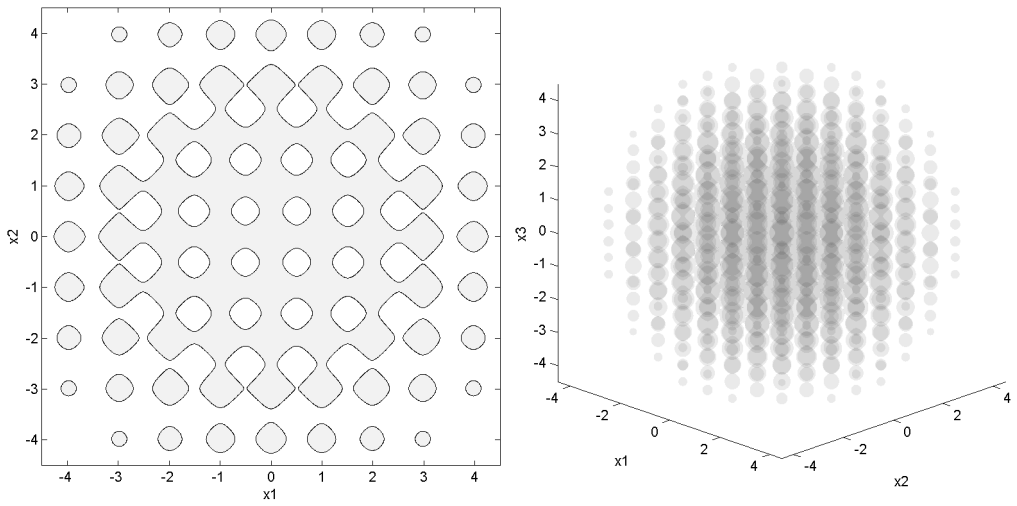

Figure 7: On the left: 2D Rastrigin level set benchmark; On the right: 3D Rastrigin level set benchmark

The approximated volume of the 2D and 3D Rastrigin level set benchmarks and their respective ratio to the search space are shown in Table 14.

area 2D Rastrigin	30.9044
area/search space ratio	$3.8154 \cdot 10^{-1}$
volume 3D Rastrigin	69.3337
volume/search space ratio	$9.5108 \cdot 10^{-2}$

Table 14: The table depicts the level set to search space ratio for the 2D and 3D Rastrigin level set benchmark.

3.8 Schaffer

Formula 53 describes the Schaffer level set benchmark, which uses the Schaffer F7 function defined for n dimensions in Formula 52 [7].

$$f_{Schaffer}(\mathbf{x}) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2)^{0.25} \cdot \left(\sin^2\left(50 \cdot (x_i^2 + x_{i+1}^2)^{0.1}\right) + 1 \right) \tag{52}$$

$$L_{Schaffer} = \left\{ \mathbf{x} \in [-2.5, 2.5]^n \middle| f_{Schaffer}(\mathbf{x}) \leq \epsilon \right\} \tag{53}$$

The parameter values used in the Schaffer level set benchmark in Formula 53 for each respective dimension are shown in Table 15.

Dimension	ϵ
2D	1.7
3D	2
10D	2
30D	2

Table 15: The parameter settings for the Schaffer level set benchmarks

The Schaffer level set benchmark consists of a complex level set shape that is based on waves. The 2D Schaffer level set has a relatively straightforward appearance with a circle that is surrounded by thin rings. The level set for this dimension also covers a large percentage of the search space. The challenge is to test whether an algorithm can spread the solutions evenly on each ring and within the circle. The 3D Schaffer level set consists of many ring objects that are ordered in a structured way that vaguely resembles a cross. Figure 8 depicts the 2D and 3D Schaffer level set benchmarks.

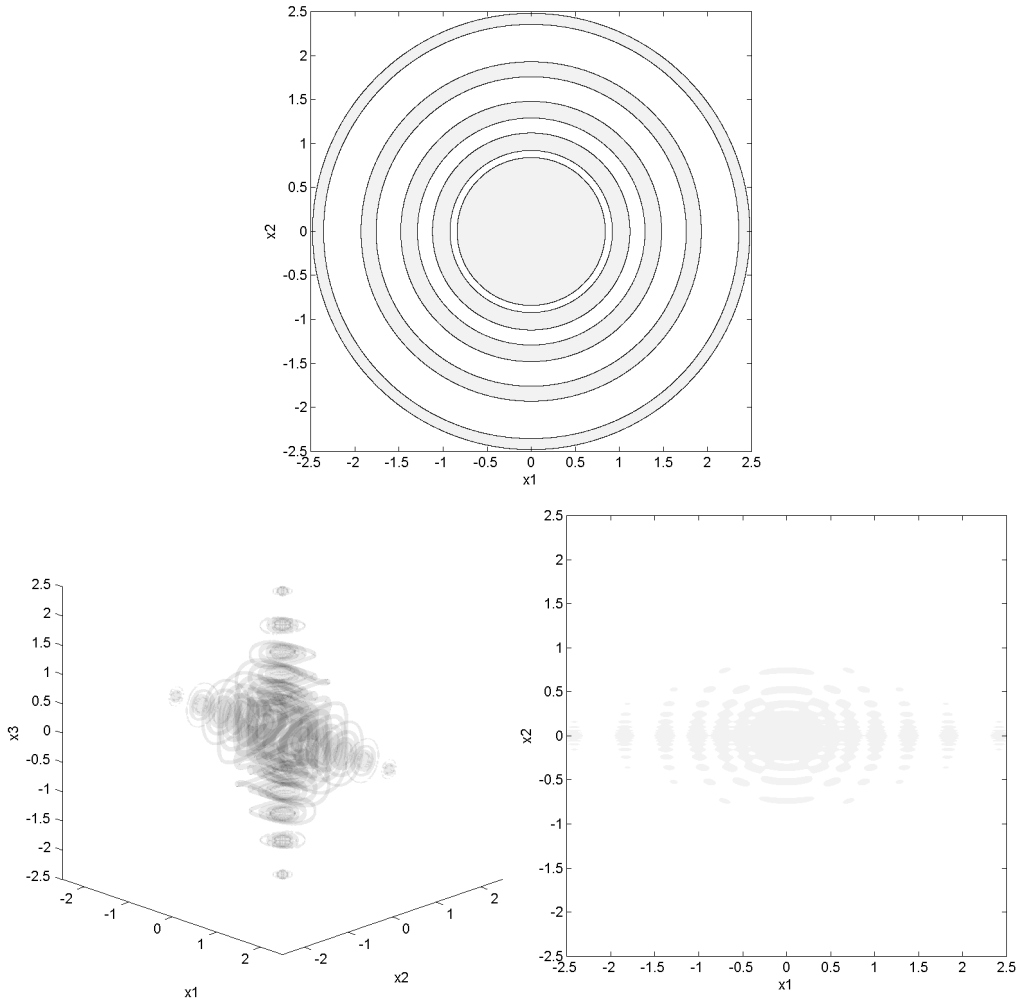

Figure 8: At the top: 2D Schaffer level set benchmark; At the bottom left: 3D Schaffer level set benchmark; At the bottom right: intersection from the 3D level set benchmark where $x_3 = 0$

The approximated volume of the 2D and 3D Schaffer level set benchmarks and their respective ratio to the search space are shown in Table 16.

area 2D Schaffer	9.0491
area/search space ratio	$3.6196 \cdot 10^{-1}$
volume 3D Schaffer	1.9897
volume/search space ratio	$1.5918 \cdot 10^{-2}$

Table 16: The table depicts the level set to search space ratio for the 2D and 3D Schaffer level set benchmark.

3.9 Vincent

Formula 55 describes the Vincent level set benchmark. The Vincent function in Formula 54 is taken from [14]. It is a sine function with a decreasing frequency.

$$f_{Vincent}(\mathbf{x}) = -\frac{1}{n}\sum_{i=1}^{n}\sin\Big(10 \cdot \ln(x_i)\Big) \tag{54}$$

$$L_{Vincent} = \Big\{\mathbf{x} \in [0.5, 5]^n \Big| f_{Vincent}(\mathbf{x}) \leq \epsilon\Big\} \tag{55}$$

The parameter values used in the Vincent level set benchmark in Formula 55 for each respective dimension are shown in Table 17.

Dimension	ϵ
2D	-0.8
3D	-0.8
10D	-0.8
30D	-0.8

Table 17: The parameter settings for the Vincent level set benchmarks

The chosen values for ϵ and nD search space size in the Vincent level set benchmark yield a level set that consists of 4^n disjoint parts in the shapes of round objects of varying elongation. The centres of the level set parts are located on a rectilinear grid with increasing unit lengths, so that this benchmark is suitable to an algorithm on its global search capabilities. Figure 9 depicts the 2D and 3D Vincent level set benchmarks.

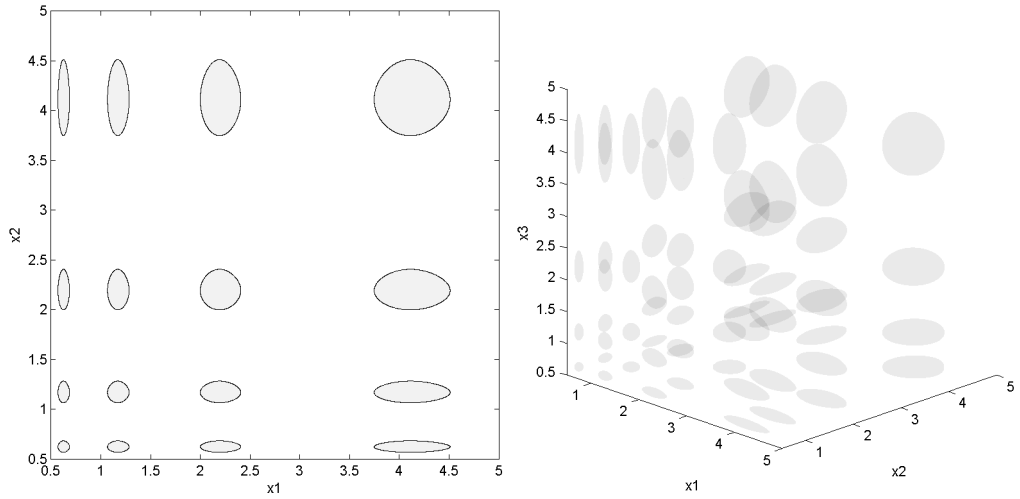

Figure 9: On the left: 2D Vincent level set benchmark; On the right: 3D Vincent level set benchmark

The approximated volume of the 2D and 3D Vincent level set benchmarks and their respective ratio to the search space are shown in Table 18.

area 2D Vincent	1.7424
area/search space ratio	$8.6043 \cdot 10^{-2}$
volume 3D Vincent	3.2368
volume/search space ratio	$3.5521 \cdot 10^{-2}$

Table 18: The table depicts the level set to search space ratio for the 2D and 3D Vincent level set benchmark.

4 Empirical studies with ELSA and ELSA-SR

This section describes the empirical study of ELSA and ELSA-SR. First, the standard of objectives for ranking different configurations is defined in Section 4.1 as they re-occur in multiple experiments. This is followed by experiments to determine recommended settings for the algorithm parameters, where Section 4.2 specifies them for ELSA and Section 4.3 for ELSA-SR. Besides the purpose of choosing recommended values, a study is done on the influences of each algorithm parameter. Section 4.4 reports the experiments and results with the recommended configurations of ELSA and ELSA-SR on the entire black box level set approximation problem benchmark suite. This includes an elaborate comparison between the different quality indicators and a comparison between the different algorithm configurations. Based on the findings in Section 4.4, Section 4.5 provides a commentary on whether the level set benchmark suite meets the criteria imposed by the level set design guidelines in Section 3.1.

The default pseudo-random number generator from MATLAB 11 was used in this work. The population standard deviation is used as the standard deviation in the results.

4.1 Objectives for the comparison of algorithm configurations

For the comparison of the results to decide that a configuration is preferred over another, the following objectives are used:

- **Eval**$_{Feasible}$: the amount of evaluations it takes to yield a population that solely consists of feasible solutions. Eval$_{Feasible}$ is both used as a measurement to calculate the speed of an algorithm in finding a feasible population and as a measurement to estimate the difficulty of a level set benchmark to be solved. In the cases where no value for Eval$_{Feasible}$ is found (within the allowed evaluation budget) two extra terms are used which are described below.
  - **Population Feasibility Rate**: percentage of the tested runs that generated a whole, feasible final population.
  - **Solution Feasibility Rate**: percentage of feasible solutions in the population. Given a level set benchmark, the Solution Feasibility Rate is measured as the average percentage of feasible solutions in populations that contain infeasible solutions. The purpose of this indicator is to understand the algorithm's behaviour in the cases where it does not find a whole, feasible population.
- **Diversity**: the quality indicator measure for the final population. In general the diversity is measured with the quality indicator that has been used to generate the population, but in some cases multiple quality indicators are used to describe the different aspects of the final population.
- **Coverage**: the coverage of the final population on the level set determined by human inspection on the visualised results. This objective is further divided into multiple subcriteria to look into when describing the distribution of the population.
  - **Boundary Concentration**: the tendency that a solution is aligned on the boundary of a level set which is defined by ϵ. The ratio of the solutions that resides on boundary compared to those that are closer to the interior of level sets are taken into consideration. A solution is considered to reside on the boundary if its marker is located on top of the level set boundaries in the generated figures in the Appendix.
  - **Clustering Interior**: the tendency of solutions to form clusters within the level set.

- **Even Distribution**: the tendency of solution to be evenly distributed in the level set.
- **Incomplete Coverage**: whether a level set with multiple disjoint components has one or more components without a solution situated on them. Incomplete coverage can be further specified by counting the amount of missing level set components that are not covered by the population. This does not require human input as the counting process can be automatised.

The objectives are closely related to the guidelines in the design of a level set benchmark in Section 3.1.

4.2 Experiments on ELSA parameter configurations

The ELSA algorithm only contains three vital configurable algorithm parameters for controlling its behaviour: the chosen quality indicator function, and the two static continuous parameters σ (mutation step-size) and ν (probability of which type of mutation to occur). The effects from using different values for σ and ν are examined with the aim to derive a recommended configuration for the black box level set benchmarks. Different σ step-sizes are tested for ELSA in Section 4.2.1, whereas Section 4.2.2 shows the results of ELSA with different ν values. Section 4.2.3 derives a conclusion from the results gathered in Section 4.2.1 and Section 4.2.2 and provides a recommended configuration for ELSA. Furthermore, analysing different values for σ and ν provides us knowledge about what is necessary for the approximation of black box level set benchmarks.

4.2.1 Experiments and results on σ step-size

This section first covers the set-up for the experiments to compare different σ values for ELSA and shows the results afterwards.

General set-up:

2D level set benchmarks have been already covered for ELSA in previous work, hence 3D level set benchmarks will be used for the testing of the algorithm parameters. Two 3D level set benchmarks are used for the experiment: Ellipsoid and Vincent. These are chosen because they represent the core opposites of having a non-disjoint level set with Ellipsoid versus the multi-segmented Vincent level set. The evaluation budget for a single ELSA run is set to 10K. Population size $\mu = 100$ is used for all experiments. GI_{Π}^{+} and SPI^{+} are used as quality indicators. Each configuration had been run 40 times. $\mathrm{Eval}_{Feasible}$ and diversity are used as measurements to rank the algorithm configurations.

Additionally, the 2D Vincent level set benchmark is used for the purpose of analysing the Incomplete Coverage from using ELSA with equivalent configurations. Using 3D Vincent as a benchmark is a possibility for this purpose, however Incomplete Coverage occurs in all the cases and then we rely on counting the amount of missing level set components to compare the performance between the different settings. This does not showcase the robustness of whether the algorithm configuration can achieve finding all level set components.

Configurations:

Several σ step-sizes are to be defined to use in ELSA. The generic form for the σ step-size

with ω normalisation is described in Formula 56, where n is the dimension of the search space. The term $\sqrt{n}$ is derived from the longest diagonal of an n-dimensional hypercube.

$$\sigma = \frac{\omega \cdot \text{mean}(\mathbf{x}^{max} - \mathbf{x}^{min})}{\sqrt{n}} \tag{56}$$

The chosen ω values for the σ step-sizes are 1, 0.1, 0.01, and 0.001 (different magnitudes of 10). At $\nu = 0$, the child is a random reinitialised point in the search space, and at $\nu = 1$, the child is created from perturbation of a random parent. Monte Carlo Search (MCS), which can be derived with $\nu = 0$ in ELSA (the value of σ is irrelevant as the algorithm never produce children through parent perturbation), is also additionally run as a reference algorithm for comparison. For the rest of the configurations of this experiment $\nu = 0.5$ is chosen.

Results:

Table 19 depicts the average $\text{Eval}_{Feasible}$ and average diversity from the configurations that have been tested. The best configuration is marked with bolded text in the table (this applies for all subsequent tables encountered in this thesis). Regardless of the σ step-size, all of the configurations manage to find populations that solely consist of feasible solutions in the results for the two 3D level set benchmarks within the evaluation budget. A correlation can be found that the smaller σ step-size, the faster on average the population converges to a population consisting of solely feasible solutions, however the results from $\omega = 0.01$ and $\omega = 0.001$ do not differ significantly. In theory, if σ is chosen too small the population may not find a whole, feasible population, but it is not the case for the tested σ step-sizes on the two benchmarks. That Monte Carlo Search can find decent results indicates that these level set benchmarks are not extremely difficult to solve.

		MCS		$\omega = 1$		$\omega = 0.1$		$\omega = 0.01$		$\omega = 0.001$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Ellipsoid	$\mathbf{GI}_{\Pi}^{+}$	0.592	0.004	0.593	0.006	0.625	0.004	**0.647**	0.005	0.594	0.007
	$\mathbf{SPI}^{+}$	99.198	0.029	99.192	0.035	99.381	0.016	**99.498**	0.016	99.170	0.045
	$\mathbf{Eval}_{Feasible}$	1030	101	1044	93	468	40	**454**	49	456	49
Vincent	$\mathbf{GI}_{\Pi}^{+}$	0.518	0.015	0.514	0.019	0.525	0.016	**0.541**	0.019	0.444	0.022
	$\mathbf{SPI}^{+}$	98.736	0.243	98.716	0.212	98.920	0.202	**98.948**	0.232	97.132	0.571
	$\mathbf{Eval}_{Feasible}$	2852	309	2843	283	1086	91	659	83	**656**	77

Table 19: The diversity of the final populations from the tested algorithm configurations with different ω normalised step-sizes on the selected 3D level set benchmarks.

Figure 10 depicts the robustness of the configurations by visualising the distribution of the quality indicator values from the final populations in box plots. In both the 3D Ellipsoid and Vincent level set benchmark the performance from ELSA is gradually better when a smaller σ step-size until $\omega = 0.001$ is used: $\omega = 0.01$ is therefore optimal of the tested configurations. For 3D Ellipsoid there is no overlap in diversity of the populations generated from $\omega = 1$, $\omega = 0.1$, and $\omega = 0.01$, while for 3D Vincent they overlap and the boxplot spreads resemble each other more. The configuration of $\omega = 0.001$ is proven to be too small to be used as a σ step-size in both level set benchmarks as the results are comparable or slightly worse than the results from MCS for Ellipsoid, but significantly worse for Vincent. The results from the configuration with $\omega = 1$ also resemble the results from Monte Carlo Search in both level set benchmarks, which may imply that this step-size is considered to be too large that the generation of children tends to resemble a random reinitialisation. Similar patterns in the results occur for the two different quality indicators in terms of how the configurations compare to each other.

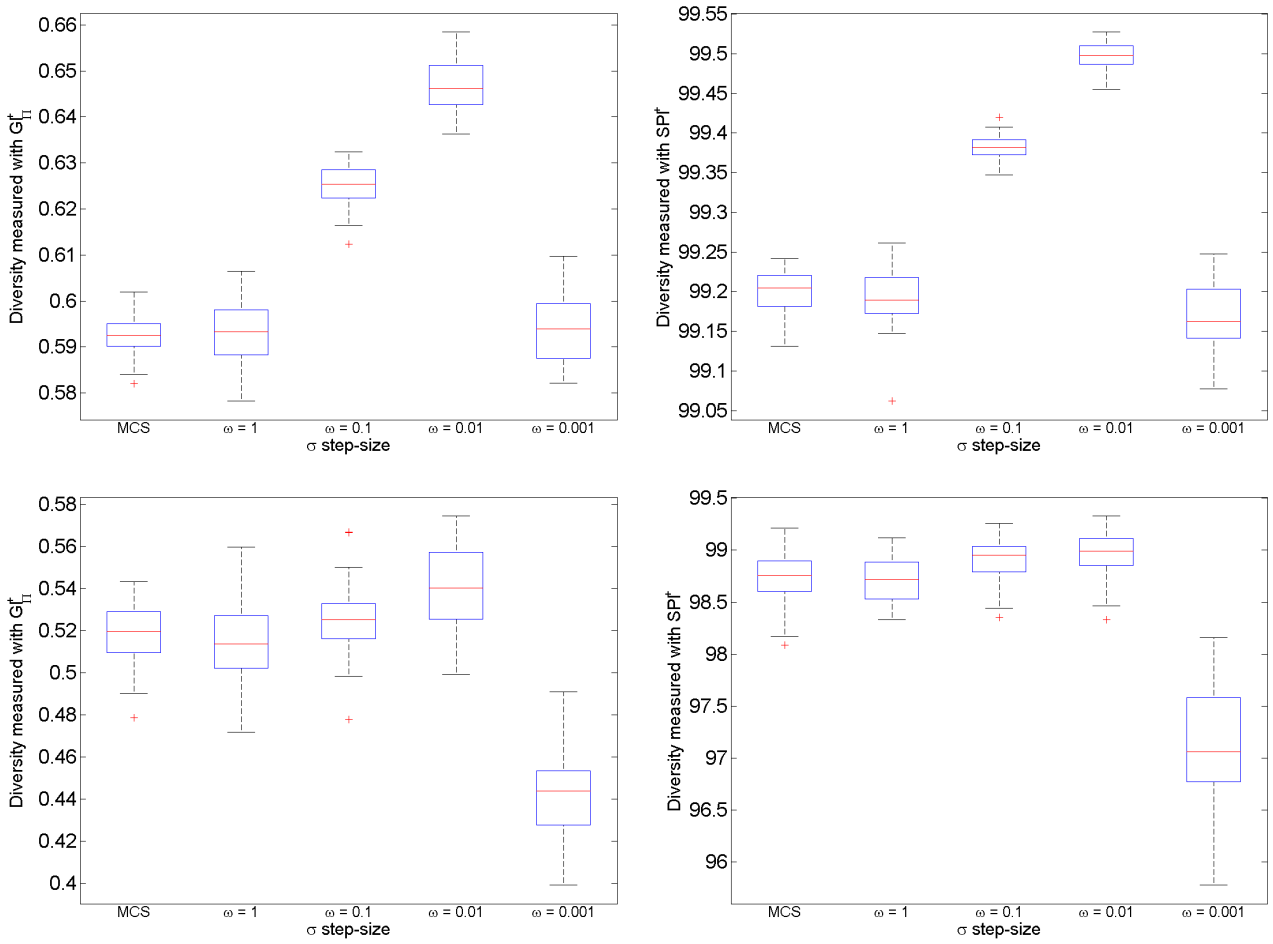

Figure 10: The box plots depict the distribution of the diversity of the final populations found from the different σ configurations, where a red cross signifies an outlier. The top row represents the results for the 3D Ellipsoid level set benchmark and the bottom row for the 3D Vincent level set benchmark. The left column represents the results ran with the GI_{Π}^{+} quality indicator and the right column the results from SPI^{+}.

Table 20 provides information on the Incomplete Coverage of the populations on the 2D Vincent level set benchmark by counting the number of times that a configuration generates a population that does not cover every disjoint level set part. Of all the encountered cases, only one level set part is not covered, which is usually the smallest level set part of Vincent located at the bottom left.

	MCS	$\omega = 1$	$\omega = 0.1$	$\omega = 0.01$	$\omega = 0.001$
$\mathbf{GI_{\Pi}^{+}}$	**0**	**0**	**0**	5	3
$\mathbf{SPI^{+}}$	**0**	**0**	3	3	8

Table 20: The number of occurrences in 40 runs where the final population does not cover every level set part from the 2D Vincent level set benchmark.

4.2.2 Experiments and results on ν

That ν parameter has an influence to the ELSA algorithm is already seen in the experiments with different σ step-sizes in Section 4.2.1, as Monte Carlo Search is identical to any configuration with $\nu = 0$ regardless of σ value and differences are found in the results between Monte Carlo Search and the configurations with $\nu = 0.5$. This section examines the results with other ν values. The set-up for this experiment is very similar to the experiment with different σ values

in Section 4.2.1. What is specific for the ν experiment however is that results from ELSA with $\omega = 0.1$ and $\omega = 0.01$ are examined with the following ν values: 0, 0.25, 0.5, 0.75, and 1. Comparing two different σ step-sizes provide us knowledge on whether the effects of ν are consistent.

Results:

Table 21 depicts the average $\text{Eval}_{Feasible}$ and average diversity from the configurations with different ν values. It shows that using a larger ν value leads to less evaluations needed to reach a population that solely consists of feasible solutions.

The ν patterns from the two diffferent step-sizes are not completely similar for the 3D Ellipsoid benchmark. It can be observed for $\omega = 0.1$ that using a higher ν value yields higher diversity for both GI_{Π}^{+} and SPI^{+} on the 3D Ellipsoid benchmark. Since it is a single non-disjoint level set, ELSA benefits from step-size mutations rather than random initialisation. However when $\omega = 0.01$ is used, the results from $\nu = 1$ are even worse than MCS on the same benchmark, which indicates that picking a high ν value expresses the limitations of using a σ step-size that is too small. This is an indication that merely focusing on exploitation is not practical for ELSA. Another small difference is that for GI_{Π}^{+} with $\omega = 0.01$ the diversity of the population from $\nu = 0.5$ is slightly better than the population from $\nu = 0.75$.

The ν patterns for 3D Vincent benchmark is consistent for both quality indicators with the different step-sizes. The results from ν ordered from best to worst diversity are as follows: 0.25, 0.5, 0, 0.75, 1. The results found for $\nu = 0.75$ and $\nu = 1$ are significantly worse than those for smaller ν, and it is better to allow some parent mutations instead of completely relying on MCS even for a multi-segmented level set problem. This is because exploration is necessary for finding new level set components while exploitation is needed for distributing the solutions within a level set component.

In conclusion the experiments indicate that extreme values for ν are suboptimal for ELSA in terms of final diversity. Therefore a mixed mutation strategy employing both exploration and exploitation is recommended for ELSA when dealing with both non-disjoint and multi-segmented level set benchmarks. For a non-disjoint level set benchmark like Ellipsoid a high ν value is preferred, while a lower ν value is preferred for multi-segmented level set benchmark like Vincent. The configuration of $\nu = 0.5$ proves to be a balanced middle ground for both level set benchmarks with satisfactory results.

		$\nu = 0$		$\nu = 0.25$		$\nu = 0.5$		$\nu = 0.75$		$\nu = 1$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Ellipsoid	$\mathbf{GI}^+_\Pi$	0.592	0.004	0.613	0.005	0.625	0.004	0.631	0.005	**0.637**	0.004
($\omega = 0.1$)	$\mathbf{SPI}^+$	99.198	0.029	99.327	0.021	99.381	0.016	99.413	0.015	**99.439**	0.012
	$\mathbf{Eval}_{Feasible}$	1030	101	607	52	468	40	392	36	**352**	33
Vincent	$\mathbf{GI}^+_\Pi$	0.518	0.015	**0.532**	0.013	0.525	0.016	0.485	0.021	0.355	0.030
($\omega = 0.1$)	$\mathbf{SPI}^+$	98.736	0.243	**98.953**	0.125	98.920	0.202	98.665	0.379	94.613	2.098
	$\mathbf{Eval}_{Feasible}$	2852	309	1530	134	1086	91	855	86	**710**	74
Ellipsoid	$\mathbf{GI}^+_\Pi$	0.592	0.004	0.637	0.006	**0.647**	0.005	0.641	0.007	0.504	0.050
($\omega = 0.01$)	$\mathbf{SPI}^+$	99.198	0.029	99.444	0.022	99.498	0.016	**99.510**	0.017	99.091	0.411
	$\mathbf{Eval}_{Feasible}$	1030	101	580	62	454	49	397	51	**346**	44
Vincent	$\mathbf{GI}^+_\Pi$	0.518	0.015	**0.562**	0.015	0.541	0.019	0.440	0.025	0.178	0.028
($\omega = 0.01$)	$\mathbf{SPI}^+$	98.736	0.243	**99.088**	0.182	98.948	0.232	97.665	0.621	70.054	9.182
	$\mathbf{Eval}_{Feasible}$	2852	309	949	96	659	83	539	69	**451**	65

Table 21: The diversity of the final populations from ELSA with $\omega = 0.1$ and $\omega = 0.01$ for σ step-size and different ν values on selected 3D level set benchmarks.

Table 22 shows the Incomplete Coverage of the tested ELSA configurations with different ν values on the 2D Vincent level set benchmark.

The number of occurrences where a population does not cover all level set components in the 2D Vincent level set benchmark is higher when using $\omega = 0.01$ than with $\omega = 0.1$. The populations from $\nu = 0$ and $\nu = 0.25$ cover all components, while the populations from $\nu = 0.5$ miss one level set component at most for all the tested configurations. For $\omega = 0.1$, populations from $\nu = 0.75$ also miss one component at most, however instances with two components missing are found for $\omega = 0.01$. Using $\nu = 1$ is shown to perform bad in terms of Incomplete Coverage where the most extreme case with $\omega = 0.1$ is missing as many as 8 level set components and for $\omega = 0.01$ it even goes as far as 13 components.

An inspection of the final diversity of the populations depicted in Table 23 reveals that the diversity yield from using $\omega = 0.01$ are better than the diversity yield with $\omega = 0.1$ for equivalent ν configurations, with the exception of $\nu = 1$, despite the Incomplete Coverage is worse in the case of $\omega = 0.01$. This means that neither GI^+_Π nor SPI^+ are reflective as indicators for Incomplete Coverage. Although comparing the population from one specific configuration with each other, additional statements can be made about the relationship between diversity and Incomplete Coverage: populations of one specific configuration from $\omega = 0.1$ that miss more level set parts roughly translate to a lower diversity measure, while this dependency is absent for populations from the smaller σ step-size with $\omega = 0.01$. It should be noted that counting the amount of missing level set components does not take topography into account where some level set component may be more essential than another to be located for high diversity.

		$\nu = 0$	$\nu = 0.25$	$\nu = 0.5$	$\nu = 0.75$	$\nu = 1$
$\omega = 0.1$	$\mathbf{GI}^+_\Pi$	**0**	**0**	**0**	2	26
	$\mathbf{SPI}^+$	**0**	**0**	3	5	23
$\omega = 0.01$	$\mathbf{GI}^+_\Pi$	**0**	**0**	5	15	40
	$\mathbf{SPI}^+$	**0**	**0**	3	16	40

Table 22: The number of occurrences in 40 runs where the final population does not cover every level set part from the 2D Vincent level set benchmark for $\omega = 0.1$ and $\omega = 0.01$ with different ν values.

		$\nu = 0$		$\nu = 0.25$		$\nu = 0.5$		$\nu = 0.75$		$\nu = 1$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
$\omega = 0.1$	$\mathbf{GI}^+_\Pi$	0.163	0.002	0.166	0.002	**0.167**	0.002	0.166	0.003	0.157	0.010
	$\mathbf{SPI}^+$	64.268	0.403	64.676	0.312	65.044	0.402	**65.319**	0.461	62.243	3.556
$\omega = 0.01$	$\mathbf{GI}^+_\Pi$	0.163	0.002	0.174	0.002	**0.175**	0.003	0.167	0.004	0.114	0.015
	$\mathbf{SPI}^+$	64.268	0.403	66.736	0.226	**67.344**	0.306	66.952	0.795	44.403	6.655

Table 23: The diversity of the final populations from ELSA with $\omega = 0.1$ and $\omega = 0.01$ for σ step-size and different ν values on 2D Vincent level set benchmark.

4.2.3 Conclusion on ELSA parameter configuration

A configuration that performs optimally on all of the three aspects concerning $\text{Eval}_{Feasible}$, diversity and Incomplete Coverage has not been found. Nevertheless $\omega = 0.1$ as σ step-size and $\nu = 0.5$ are chosen to be the default configuration for the experiments in Section 4.4. Populations with $\omega = 0.01$ yields overall higher diversity than with $\omega = 0.1$ for the tested 3D level set benchmarks, however when a σ step-size is chosen too small the algorithm is more likely to miss level set components in a multi-segmented level set benchmark. A mixed mutation strategy with $\nu = 0.5$ emphasises on equal probability for random initialisation and parent mutation to occur, which is shown to be the most balanced choice for black box level set benchmarks.

4.3 Experiments on ELSA-SR parameter configurations

ELSA-SR is a variation on ELSA where an adaptive σ step-size is applied, which is modified according to the method described in the one-fifth success rule. The algorithm parameters that are found in ELSA-SR and not in ELSA are the α learning rate, β measurement range, and γ target success rate.

The optimal value of the factor α depends on the objective function to be optimized, the dimensionality n of the search space, and on the value of β. Schwefel [2] made the following recommendations: If n is sufficiently large $n \geq 30$, $\beta = n$ is a reasonable choice and under this condition $0.85 \leq \alpha < 1$ is advised. The tested dimensions for the level set benchmarks in this thesis are small, but we made the assumption that an α value in that range is a reasonable choice. $\alpha = 0.95$ is used for all the ELSA-SR configurations.

This leaves β and γ as undecided variables, which are examined in Section 4.3.1 and Section 4.3.2 respectively. Furthermore Section 4.3.3 describes to what degree ELSA-SR can adapt the initial chosen σ, specifically whether it can adjust σ step-sizes properly that are either chosen too large or too small, and whether the mixed mutation strategy can be replaced with a full mutation strategy. Section 4.3.4 concludes the findings with a recommended configuration for the algorithm parameters of ELSA-SR.

4.3.1 Experiments and results on target success rate γ

This section determines whether $\gamma = 0.2$ is also applicable for the level set approximation problem, as we postulate that this probability is determined from a specific set of optimisation problems possibly unrelated to ours. It first covers the set-up for the experiments to compare five different γ values for ELSA-SR and shows the results afterwards. The γ values include the standard 0.2, two larger success probabilities, and two smaller success probabilities.

General set-up:

Two 3D level set benchmarks are used for the experiment: Ellipsoid and Vincent. The evaluation budget for a single ELSA-SR run is set to 10K. Population size $\mu = 100$ is used. GI_N^+, GI_Π^+ and SPI^+ are used as quality indicators. Each configuration had been run 40 times. The average diversity and $Eval_{Feasible}$ of the final populations are used as measures for comparison. $Eval_{Feasible}$ is measured over all the three quality indicators that were used in this experiment.

Configurations:

The algorithm parameters in ELSA-SR that also occur in ELSA are chosen the same values described in Section 4.2.3 ($\nu = 0.5$ and $\omega = 0.1$ for σ). The γ values that have been compared are: 0.4, 0.3, 0.2, 0.1, 0.02. All configurations use $\beta = 50$ where σ changes 200 times at maximum, which in practice is approximately 100 times given that the probability to generate a solution from parent mutation is 0.5 out of an evaluation budget of 10K.

Results:

Table 24 displays the diversity of the final populations from the configurations with different γ. For GI_N^+ the diversity is better the smaller the γ value is, and closer examination shows that this quality indicator is unconstructive to be used in combination with the adaptive σ step-size with success rule method. The measured success rate is low and this triggers a positive feedback loop where the σ step-size is perpetually reduced. This is illustrated in Table 26 in Section 4.3.2. It reinforces that a quality indicator like GI_N^+ that only focuses on local improvements has its limitations. GI_N^+ is included in the rest of the ELSA-SR parameter experiments in Section 4.3, but the focus will not be put on it and instead the results from GI_Π^+ and SPI^+ are compared in the text.

Of the tested configurations for GI_Π^+ and SPI^+, $\gamma = 0.2$ yields the overall best results. For both Ellipsoid and Vincent, the results from GI_Π^+ and SPI^+ have a worse diversity the further away the γ value is from the optimal γ. It can be assumed that $\gamma = 0.2$ is not necessarily limited to specific problems, or at least the results show that it is compatible with level set approximation problems.

		$\gamma = 0.4$		$\gamma = 0.3$		$\gamma = 0.2$		$\gamma = 0.1$		$\gamma = 0.02$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Ellipsoid	$\mathbf{GI_N^+}$	0.507	0.006	0.508	0.007	0.513	0.006	0.519	0.009	**0.528**	0.006
	$\mathbf{GI_\Pi^+}$	0.664	0.005	0.664	0.005	**0.668**	0.006	0.658	0.004	0.619	0.005
	$\mathbf{SPI^+}$	99.544	0.014	99.549	0.013	**99.553**	0.011	99.506	0.016	99.337	0.018
	$\mathbf{Eval_{Feasible}}$	478	45	477	52	**476**	46	488	48	481	42
Vincent	$\mathbf{GI_N^+}$	0.308	0.014	0.312	0.012	0.322	0.013	0.332	0.013	**0.355**	0.014
	$\mathbf{GI_\Pi^+}$	0.549	0.018	0.557	0.019	0.560	0.018	**0.561**	0.015	0.518	0.017
	$\mathbf{SPI^+}$	99.104	0.168	99.175	0.185	**99.215**	0.149	99.150	0.186	98.834	0.232
	$\mathbf{Eval_{Feasible}}$	**1116**	88	1219	110	1330	121	1450	207	1469	249

Table 24: The comparison between the average diversity and $Eval_{Feasible}$ of the populations generated by ELSA-SR with different γ values.

4.3.2 Experiments and results on measurement range β

The level set approximation problem benchmarks do not fit with the conditions in the one-fifth success rule parameter guidelines provided by Schwefel. This section examines different β

values, ranging from few to many σ step-size changes, and from the results a recommended β value for ELSA-SR will be picked.

General set-up:

The same set-up described in Section 4.3.1 is used.

Configurations:

For the remainder of the thesis, $\gamma = 0.2$ is used and in this experiment the following β values have been compared:

- $\beta = 250$ (σ changes 40 times at maximum (approximately 20 in practice))
- $\beta = 100$ (σ changes 100 times at maximum (approximately 50 in practice))
- $\beta = 50$ (σ changes 200 times at maximum (approximately 100 in practice))
- $\beta = 25$ (σ changes 400 times at maximum (approximately 200 in practice))
- $\beta = 5$ (σ changes 2000 times at maximum (approximately 1000 in practice))

Results:

A common pattern for all the configurations is that the success rate always starts high (far above the target success rate) and then rapidly depletes. By the definition of the augmented quality indicators, the behaviour of ELSA and ELSA-SR can be divided into two phases: (1) objective function optimisation to find a feasible population and (2) diversity optimisation. The target success rate is easily met in the initial evaluations in the first phase when any solution that has a better objective value than another solution in the population is considered an improvement, while the second phase has much more stricter requirements. For $\beta = 250$ and $\beta = 100$, the target success rate is not met again after the initial phase, whereas using a smaller β leads to success rate plots that successfully fluctuate around the target success rate. Therefore plotting the σ changes from the configurations shows that the plot first go up into a peak and undergoes a smooth decline when using $\beta = 250$ or $\beta = 100$, but for smaller β the plot is not smooth. The adaption based on success rate inhibits the σ step-size from decreasing down to a certain level (except for GI_N^+ which is not compatible with this adaption method). The overall shape of the σ and success rate plots are similar for each configuration in both Ellipsoid and Vincent. The maximum and minimum σ values from one configuration run are found in Table 26.

Concerning the recommended β value for ELSA-SR, $\beta = 50$ yields the overall best diversity from the tested configurations. $\beta = 50$ yields the best diversities for Ellipsoid with the results from $\beta = 25$ close behind them. $\beta = 25$ yields the best GI_Π^+ diversity and $\beta = 100$ yields the best SPI^+ diversity, however the results from $\beta = 50$ are also close to the best diversity for both quality indicators.

		$\beta = 250$		$\beta = 100$		$\beta = 50$		$\beta = 25$		$\beta = 5$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Ellipsoid	$\mathbf{GI}_N^+$	**0.520**	0.006	0.512	0.007	0.513	0.006	0.496	0.007	0.486	0.007
	$\mathbf{GI}_\Pi^+$	0.631	0.005	0.657	0.005	**0.668**	0.006	0.666	0.005	0.658	0.005
	$\mathbf{SPI}^+$	99.417	0.014	99.518	0.014	**99.553**	0.011	99.551	0.009	99.539	0.014
	$\mathbf{Eval}_{Feasible}$	**473**	37	484	44	476	46	488	44	718	84
Vincent	$\mathbf{GI}_N^+$	**0.340**	0.011	0.334	0.015	0.322	0.013	0.298	0.015	0.290	0.015
	$\mathbf{GI}_\Pi^+$	0.529	0.019	0.558	0.019	0.560	0.018	**0.565**	0.023	0.557	0.017
	$\mathbf{SPI}^+$	99.035	0.172	**99.217**	0.152	99.215	0.149	99.208	0.149	99.160	0.171
	$\mathbf{Eval}_{Feasible}$	**1122**	116	1198	117	1330	121	1500	121	1691	119

Table 25: The comparison between the average diversity and $\text{Eval}_{Feasible}$ of the populations generated by ELSA-SR with $\gamma = 0.2$ and different β values.

		Ellipsoid		**Vincent**	
		max	**min**	**max**	**min**
$\beta = 250$	$\mathbf{GI}_N^+$	0.384	$1.5{\cdot}10^{-1}$	0.288	$1.2{\cdot}10^{-1}$
	$\mathbf{GI}_\Pi^+$	0.384	$1.6{\cdot}10^{-1}$	0.288	$1.2{\cdot}10^{-1}$
	$\mathbf{SPI}^+$	0.365	$1.4{\cdot}10^{-1}$	0.288	$1.2{\cdot}10^{-1}$
$\beta = 100$	$\mathbf{GI}_N^+$	0.404	$3.8{\cdot}10^{-2}$	0.319	$3.7{\cdot}10^{-2}$
	$\mathbf{GI}_\Pi^+$	0.425	$4.2{\cdot}10^{-2}$	0.336	$3.5{\cdot}10^{-2}$
	$\mathbf{SPI}^+$	0.448	$4.9{\cdot}10^{-2}$	0.336	$3.5{\cdot}10^{-2}$
$\beta = 50$	$\mathbf{GI}_N^+$	0.496	$4.9{\cdot}10^{-3}$	0.336	$4.1{\cdot}10^{-3}$
	$\mathbf{GI}_\Pi^+$	0.522	$1.2{\cdot}10^{-2}$	0.372	$1.3{\cdot}10^{-2}$
	$\mathbf{SPI}^+$	0.471	$2.0{\cdot}10^{-2}$	0.392	$1.3{\cdot}10^{-2}$
$\beta = 25$	$\mathbf{GI}_N^+$	0.710	$6.6{\cdot}10^{-5}$	0.457	$7.5{\cdot}10^{-5}$
	$\mathbf{GI}_\Pi^+$	0.748	$6.7{\cdot}10^{-3}$	0.481	$1.4{\cdot}10^{-2}$
	$\mathbf{SPI}^+$	0.675	$1.2{\cdot}10^{-2}$	0.412	$1.3{\cdot}10^{-2}$
$\beta = 5$	$\mathbf{GI}_N^+$	1.614	$5.5{\cdot}10^{-19}$	0.803	$1.5{\cdot}10^{-17}$
	$\mathbf{GI}_\Pi^+$	1.614	$4.0{\cdot}10^{-3}$	1.038	$9.8{\cdot}10^{-3}$
	$\mathbf{SPI}^+$	1.533	$8.2{\cdot}10^{-3}$	0.689	$8.8{\cdot}10^{-3}$

Table 26: The maximum and minimum σ found from one run of each configuration that has yield a diversity that is closest to the average diversity corresponding to the respective configuration. Initial σ value for Ellipsoid is 0.346 and 0.260 for Vincent.

4.3.3 ELSA-SR adaptability

This section deals with examining the capability of ELSA-SR in adapting the σ step-size and is split up into two main experiments. The first experiment focuses on whether ELSA-SR can stabilise the σ step-size when using initial σ step-sizes with different magnitudes of ω normalisation. The second experiment is a comparison whether ELSA-SR can be implemented as a full mutation strategy instead of a mixed mutation strategy.

The same set-up described in Section 4.3.1 is used for both experiments. For the experiment to test different initial σ step-sizes, the following algorithm parameter values for ELSA-SR are used: $\alpha = 0.95$ and $\gamma = 0.2$. The initial σ step-sizes to be tested have been introduced earlier in Section 4.2.1 ($\omega = 1$, $\omega = 0.1$, $\omega = 0.01$, and $\omega = 0.001$). Results have been gathered for $\beta = 50$ and $\beta = 5$, which are shown in Table 27 and Table 28 respectively.

Using $\omega = 1$ for the σ step-size is considered a large step-size and $\omega = 0.001$ for the σ step-size is considered a small step-size for the level set benchmarks according to the ELSA parameters experiments in Section 4.2.1. Using a small β value allows for more σ step-size changes and it is observed in Table 28 that the final diversities from different initial σ step-sizes are in close approximation to each other, especially for Ellipsoid. The results in Table 27 and Table 28

can be considered an indication of functional step-size adaptability in ELSA-SR as the similar comparisons for ELSA in Table 19 show far larger differences in final diversities of populations from different step-sizes. The final diversities from using the different step-sizes with $\beta = 50$ are less close to each other. However, comparing the results from $\beta = 5$ and $\beta = 50$, reveals that using $\beta = 50$ over $\beta = 5$ can reach slightly higher diversities for the σ step-size with $\omega = 0.1$. When choosing the algorithm parameters with the goal of diversity maximization, choosing a small β does not guarantee the optimal results and larger β values might be considered.

		$\omega = 1$		$\omega = 0.1$		$\omega = 0.01$		$\omega = 0.001$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Ellipsoid	$\mathbf{GI}_N^+$	**0.517**	0.007	0.513	0.007	0.467	0.009	0.463	0.008
	$\mathbf{GI}_\Pi^+$	0.647	0.005	**0.666**	0.005	0.656	0.006	0.644	0.007
	$\mathbf{SPI}^+$	99.479	0.015	**99.554**	0.010	99.528	0.015	99.487	0.015
	$\mathbf{Eval}_{Feasible}$	1025	94	477	46	458	50	**448**	48
Vincent	$\mathbf{GI}_N^+$	**0.347**	0.015	0.320	0.011	0.261	0.016	0.254	0.016
	$\mathbf{GI}_\Pi^+$	**0.573**	0.017	0.565	0.018	0.531	0.019	0.535	0.020
	$\mathbf{SPI}^+$	**99.268**	0.117	99.199	0.180	99.079	0.163	98.730	0.274
	$\mathbf{Eval}_{Feasible}$	2785	237	1333	122	661	73	**660**	88

Table 27: Results of experiments on adaptability of ELSA-SR with $\beta = 50$

		$\omega = 1$		$\omega = 0.1$		$\omega = 0.01$		$\omega = 0.001$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Ellipsoid	$\mathbf{GI}_N^+$	0.484	0.007	0.484	0.007	**0.484**	0.008	0.480	0.008
	$\mathbf{GI}_\Pi^+$	0.659	0.006	**0.660**	0.006	0.659	0.006	0.657	0.008
	$\mathbf{SPI}^+$	99.539	0.018	**99.544**	0.013	99.541	0.014	99.536	0.014
	$\mathbf{Eval}_{Feasible}$	1038	94	709	83	**451**	50	455	53
Vincent	$\mathbf{GI}_N^+$	**0.294**	0.016	0.286	0.013	0.291	0.016	0.282	0.015
	$\mathbf{GI}_\Pi^+$	**0.567**	0.016	0.552	0.019	0.542	0.024	0.527	0.020
	$\mathbf{SPI}^+$	**99.225**	0.122	99.147	0.164	99.133	0.160	99.053	0.179
	$\mathbf{Eval}_{Feasible}$	2081	115	1684	125	1076	172	**677**	100

Table 28: Results of experiments on adaptability of ELSA-SR with $\beta = 5$

For the experiment to test different ν values, the following algorithm parameter values for ELSA-SR are used: $\alpha = 0.95$, $\beta = 50$, and $\gamma = 0.2$. The corresponding results are depicted in Table 29. Purely using mutation without random solution yield worse results than mixed mutation strategy. Again, $\nu = 0.5$ is most balanced for a black box level set benchmark, as shown in the similar test on the ν parameter with ELSA. ELSA-SR is therefore incapable to be used as a full mutation strategy despite the presence of an adaptive σ step-size.

		$\nu = 0$		$\nu = 0.25$		$\nu = 0.5$		$\nu = 0.75$		$\nu = 1$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Ellipsoid	$\mathbf{GI}_N^+$	0.498	0.007	0.507	0.006	**0.513**	0.006	0.504	0.007	0.497	0.008
	$\mathbf{GI}_\Pi^+$	0.592	0.004	0.634	0.005	0.668	0.006	**0.674**	0.005	0.674	0.008
	$\mathbf{SPI}^+$	99.198	0.029	99.425	0.018	99.553	0.011	99.578	0.011	**99.588**	0.009
	$\mathbf{Eval}_{Feasible}$	1035	98	610	53	476	46	413	36	**367**	37
Vincent	$\mathbf{GI}_N^+$	**0.347**	0.015	0.345	0.014	0.322	0.013	0.272	0.013	0.223	0.013
	$\mathbf{GI}_\Pi^+$	0.518	0.015	0.559	0.017	**0.560**	0.018	0.510	0.025	0.366	0.032
	$\mathbf{SPI}^+$	98.736	0.243	**99.250**	0.132	99.215	0.149	98.618	0.381	93.445	2.412
	$\mathbf{Eval}_{Feasible}$	2835	303	1716	173	1330	121	1128	120	**974**	120

Table 29: Results of experiments on adaptability of ELSA-SR with different ν values

4.3.4 Conclusion on ELSA-SR parameter configuration

To sum it up, Section 4.3 has examined different values for algorithm parameters in ELSA-SR on 3D level set benchmarks to derive a standard. The proposed values are $\alpha = 0.95$, the value for the measurement range of β is set accordingly for 200 changes at maximum for the σ parameter ($\beta = 50$ for an evaluation budget of 10K), and a target success probability of $\gamma = 0.2$.

4.4 Experiments with ELSA and ELSA-SR on the entire level set benchmark suite

This section revolves around the experiments of ELSA and ELSA-SR on the entire level set benchmark suite with the dimensions 2D, 3D, 10D, and 30D (described in Section 3). It is divided into two parts as different aspects of the results are examined. Section 4.4.1 takes a closer look to reoccurring patterns in the population distributions on the level sets to determine the main characteristics of a quality indicator. After that in Section 4.4.2, the limitations of ELSA and ELSA-SR are analysed for which level set benchmarks they can solve. Predominantly, it is to compare whether the proposed ELSA-SR is an improvement for ELSA.

The following four main algorithm configurations are used:

1. **MCS**: Monte Carlo Search is used as a reference.
2. **ELSA**: ELSA with $\omega = 0.1$ and $\nu = 0.5$.
3. **ELSA-SR**$_{\omega=0.1}$: ELSA-SR with $\gamma = 0.2$, $\alpha = 0.95$, $\omega = 0.1$, and $\nu = 0.5$. The measurement range over which the success rate is calculated is set accordingly that σ can at maximum change 200 times during its run ($\beta = 50$ for 2D and 3D; $\beta = 500$ for 10D and 30D).
4. **ELSA-SR**$_{\omega=1}$: ELSA-SR with algorithm parameters almost equivalent to ELSA-SR$_{\omega=0.1}$ with the exception of the value of ω which is set to 1.

When the configuration is called ELSA-SR in the text, it addresses both the ELSA-SR$_{\omega=0.1}$ and ELSA-SR$_{\omega=1}$ configurations.
Each algorithm configuration with a specific quality indicator is run 40 times on a benchmark. The evaluation budget for the 2D and 3D level set benchmarks is set to 10K and for 10D and 30D benchmarks it is set to 100K.

4.4.1 Comparison between the quality indicators

The main characteristics from the populations associated with a particular quality indicator are examined in this section. The quality indicators in question are Augmented Minimal Gap Indicator (GI_N^+), Augmented Arithmetic Mean Gap Indicator (GI_Σ^+), Augmented Geometric Mean Gap Indicator (GI_Π^+), Augmented Solow Polasky Indicator (SPI^+), and Augmented Average Distance Indicator (ADI^+). The characteristics are divided into two categories: in terms of diversity properties and coverage properties. At the end of Section 4.4.1, the most important findings are summarised.

Quality indicator comparison based on diversity properties:

Besides measuring a population with its own corresponding quality indicator, all the other quality indicators are also taken into account to discover what other spread properties are

brought forth and which quality indicators resemble each other. This also allows alternatives of quality indicators to be found, for example ADI^+ is impractical for actual use due to its high complexity. ADI^+ itself is not regarded as a viable quality indicator due to its inefficiency and its results will not be taken into account when scoring on average which population is scored the best on a quality indicator. Note that the ADI^+ values have been made negative in the results, because maximisation of the quality indicator is used in the ELSA algorithm definition. GI^+_Σ has been left out in dimensions higher than 2D because its ineffectiveness for diversity optimization (as in achieving the criteria we are looking for) are apparent in the 2D results.

Table 30 depicts the average diversity of the populations from MCS on the 2D level set benchmarks. This algorithm configuration is the most unbiased and therefore provides the best idea how a quality indicator behaves in a neutral setting without setting specific algorithms parameters. However, MCS performs extremely poor for 10D level set benchmarks, and observing the results from ELSA is more appropriate to yield insights about quality indicators throughout different dimensions. Table 31 shows the quality indicator for population selection that yields the highest diversity in each quality indicator measurement for results from ELSA with $\nu = 0.5$ and $\omega = 0.1$ for σ step-size on level set benchmarks in 2D, 3D, and 10D.

The observation is that using a particular quality indicator for population selection does not necessarily yield a population that scores the best in its own corresponding quality indicator measurement. This effect is expressed most visibly when comparing level sets of different dimensions as seen in Table 31.

All level set benchmarks, with the exception of Ellipsoid (and Branke's Multipeak which has only been defined for 2D and 3D), share identical values for their respective level set function parameter constants and ϵ threshold for 3D, 10D, and 30D. Hollow Sphere, Double Sphere, Rastrigin and Vincent have the same level set function parameter values and ϵ for all dimensions. The Lamé, Ellipsoid, and Branke's Multipeak 2D and 3D level set benchmarks maintain similar characteristics in terms of shape. Only the Schaffer level set displays very different traits, where the 2D Schaffer level set is well-spread and covers a large area of search space while the 3D level set consists of many tiny disjoint level set parts that are clustered at the centre of the search space. Nevertheless the majority of the level set benchmarks can serve as indications how a quality indicator behaves in different dimensions when looking at the same level set function.

2D benchmarks:

In the 2D case the quality indicator patterns and properties witnessed over MCS, ELSA, and ELSA-SR are relatively similar. Using GI^+_Π for population selection in the algorithm often generates a slightly more diverse population measured in GI^+_Σ than using GI^+_Σ itself. Populations from SPI^+ have the most resemblance to ADI^+ of the compared quality indicators, which makes SPI^+ a good alternative for ADI^+ as its time complexity is lower. However, populations from SPI^+ do not surpass the populations from ADI^+ by the measure of their ADI^+ diversity.
The 2D Rastrigin and Schaffer are benchmarks where populations from SPI^+ can outperform the native quality indicators (here SPI^+ even outperforms GI^+_N in the MCS, ELSA, and ELSA-SR runs). These two benchmarks are the only ones where the SPI^+ measurement of the populations have almost reached the optimum of 100 (in a population of size 100). Both the level sets have a large surface area and are spread out in the search space.
GI^+_N is found to be less receptive of small step size mutations compared to the other quality indicators, as it is observed in the comparison between ELSA and ELSA-SR that the other quality indicators gain a diversity improvement when using ELSA-SR, while the results from

GI_N^+ become worse. SPI^+ especially gets the largest benefit.

3D benchmarks:

For 3D level set benchmarks a gradual shift is observed where using SPI^+ for population selection can more often perform better than its native quality indicator. For ELSA, SPI^+ yields higher diversity than populations from GI_N^+ itself for 3D Hollow Sphere, Rastrigin and Vincent. SPI^+ also outperforms GI_Π^+ for Rastrigin and Vincent. The effect is most apparent in ELSA-$SR_{\omega=0.1}$ where SPI^+ is often better than both GI_N^+ and GI_Π^+. ELSA-$SR_{\omega=0.1}$ SPI^+ outperforms all the tested native quality indicators for 3D Ellipsoid and Hollow Sphere. However SPI^+ dominates the least for 3D Lamé and Schaffer (and Double Sphere to lesser degree). The common trait of these level sets is that their overall level set shapes are small and compact within their respective search space (although Lamé does have narrow pointy segments). This might be an effect that can be attributed to the choice of θ parameter in the Solow Polasky diversity. It seems to be well chosen for the typical distances obtained in the level sets of the 3D Ellipsoid and Hollow Sphere, while less appropriate in cases where small distances occur in diverse sets. Overall, for MCS, ELSA, and ELSA-SR, the populations that have the best diversity in the GI_Σ^+ measure for 3D benchmarks have been produced from using GI_Π^+. The quality indicator patterns from the results of ELSA-$SR_{\omega=1}$ for 3D are not particularly distinctive and are a mixture of patterns found for ELSA and ELSA-$SR_{\omega=0.1}$.

10D benchmarks:

Some 10D level set benchmarks prove to be too difficult for ELSA or even ELSA-$SR_{\omega=0.1}$ to solve. The ELSA configuration cannot find feasible populations for 10D Lamé, Rastrigin, and Schaffer, while the ELSA-$SR_{\omega=0.1}$ configuration cannot find feasible populations for Schaffer. The quality indicator patterns for 10D Ellipsoid, Hollow Sphere, Double Sphere, and Vincent are the same for both algorithms. Populations from SPI^+ has taken over as the having the best diversity for each quality indicator category for 10D Ellipsoid and Hollow Sphere. Similar patterns in results are also seen for 10D Vincent, but the populations from GI_Π^+ measure better on average for the GI_Σ^+ category than the populations from SPI^+. For the other 10D level set benchmarks (Lamé, Double Sphere, and Rastrigin), populations from GI_Π^+ has the best diversity for the GI_N^+, GI_Σ^+, and GI_Π^+ measurements, and populations SPI^+ remains the best for its own quality indicator measurements.

		$\mathbf{GI}_N^+$	$\mathbf{GI}_\Sigma^+$	$\mathbf{GI}_\Pi^+$	$\mathbf{SPI}^+$	$\mathbf{ADI}^+$
Lamé	$\mathbf{GI}_N^+$	**0.1255**	0.1467	0.1451	49.6931	-1.1662
	$\mathbf{GI}_\Sigma^+$	0.0978	**0.1676**	0.1542	46.1628	-1.1575
	$\mathbf{GI}_\Pi^+$	0.1174	0.1655	**0.1614**	50.0734	-1.1584
	$\mathbf{SPI}^+$	0.0835	0.1497	0.1467	**52.9408**	**-1.1529**
	$\mathbf{ADI}^+$	0.0373	0.1482	0.1403	52.9437	-1.1460
Ellipsoid	$\mathbf{GI}_N^+$	**0.1997**	0.2210	0.2201	70.9337	-1.0265
	$\mathbf{GI}_\Sigma^+$	0.1680	0.2435	0.2393	71.2754	-1.0123
	$\mathbf{GI}_\Pi^+$	0.1877	**0.2455**	**0.2432**	72.3526	-1.0107
	$\mathbf{SPI}^+$	0.1662	0.2363	0.2347	**74.3128**	**-1.0005**
	$\mathbf{ADI}^+$	0.0851	0.2236	0.2149	73.6042	-0.9967
Hollow	$\mathbf{GI}_N^+$	**0.1973**	0.2211	0.2201	73.6049	-0.7795
	$\mathbf{GI}_\Sigma^+$	0.1684	0.2486	0.2447	75.1824	-0.7640
	$\mathbf{GI}_\Pi^+$	0.1868	**0.2501**	**0.2478**	76.0396	-0.7635
	$\mathbf{SPI}^+$	0.1744	0.2437	0.2417	**78.0884**	**-0.7551**
	$\mathbf{ADI}^+$	0.1023	0.2316	0.2238	77.4898	-0.7511
Double	$\mathbf{GI}_N^+$	**0.2021**	0.2248	0.2238	72.7564	-0.9375
	$\mathbf{GI}_\Sigma^+$	0.1710	0.2505	0.2461	73.5962	-0.9193
	$\mathbf{GI}_\Pi^+$	0.1924	**0.2519**	**0.2494**	74.6060	-0.9186
	$\mathbf{SPI}^+$	0.1662	0.2415	0.2395	**76.7023**	**-0.9084**
	$\mathbf{ADI}^+$	0.0867	0.2278	0.2185	75.9733	-0.9038
Branke's	$\mathbf{GI}_N^+$	**0.1480**	0.1673	0.1661	57.9643	-0.3631
	$\mathbf{GI}_\Sigma^+$	0.1135	0.1945	0.1785	54.1786	-0.3745
	$\mathbf{GI}_\Pi^+$	0.1383	**0.1954**	**0.1892**	59.3840	-0.3489
	$\mathbf{SPI}^+$	0.1131	0.1759	0.1735	**63.2837**	**-0.3420**
	$\mathbf{ADI}^+$	0.0584	0.1740	0.1670	63.0327	-0.3407
Rastrigin	$\mathbf{GI}_N^+$	0.6136	0.7309	0.7223	99.8028	-0.3805
	$\mathbf{GI}_\Sigma^+$	0.5266	0.8261	0.8072	99.7732	-0.3760
	$\mathbf{GI}_\Pi^+$	0.5909	**0.8323**	**0.8192**	99.8466	-0.3721
	$\mathbf{SPI}^+$	**0.6382**	0.8225	0.8156	**99.9064**	**-0.3677**
	$\mathbf{ADI}^+$	0.4951	0.7890	0.7801	99.8777	-0.3638
Schaffer	$\mathbf{GI}_N^+$	0.3533	0.3964	0.3942	95.1282	-0.2358
	$\mathbf{GI}_\Sigma^+$	0.2728	0.4431	0.4284	93.3081	-0.2407
	$\mathbf{GI}_\Pi^+$	0.3169	**0.4484**	**0.4409**	94.9414	-0.2341
	$\mathbf{SPI}^+$	**0.3545**	0.4350	0.4334	**96.2727**	**-0.2276**
	$\mathbf{ADI}^+$	0.2730	0.4245	0.4213	95.9202	-0.2233
Vincent	$\mathbf{GI}_N^+$	**0.1153**	0.1423	0.1380	57.7848	-0.3739
	$\mathbf{GI}_\Sigma^+$	0.0751	**0.1870**	0.1476	49.6033	-0.3841
	$\mathbf{GI}_\Pi^+$	0.0994	0.1793	**0.1626**	58.1731	-0.3630
	$\mathbf{SPI}^+$	0.0754	0.1471	0.1429	**64.2682**	**-0.3550**
	$\mathbf{ADI}^+$	0.0434	0.1425	0.1322	63.1404	-0.3532

Table 30: Average diversity of the final populations yielded with MCS for the 2D level set benchmarks. Rows represent the quality indicators that have been used for population selection; and columns represent the quality of the final population according to a quality indicator measurement. The diversity from a quality indicator that performs best in a quality indicator measurement is bolded, where populations from ADI^+ are exempted from participation.

		$\mathbf{GI}_N^+$	$\mathbf{GI}_\Sigma^+$	$\mathbf{GI}_\Pi^+$	$\mathbf{SPI}^+$	$\mathbf{ADI}^+$
Lamé	**2D**	GI_N^+	GI_Σ^+	GI_Π^+	SPI^+	SPI^+
	3D	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	
	10D	-	-	-	-	
Ellipsoid	**2D**	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	SPI^+
	3D	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	
	10D	SPI^+	SPI^+	SPI^+	SPI^+	
Hollow	**2D**	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	SPI^+
	3D	SPI^+	GI_Π^+	GI_Π^+	SPI^+	
	10D	SPI^+	SPI^+	SPI^+	SPI^+	
Double	**2D**	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	SPI^+
	3D	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	
	10D	GI_Π^+	GI_Π^+	GI_Π^+	SPI^+	
Branke's	**2D**	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	SPI^+
	3D	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	
Rastrigin	**2D**	SPI^+	GI_Π^+	GI_Π^+	SPI^+	SPI^+
	3D	SPI^+	GI_Π^+	SPI^+	SPI^+	
	10D	-	-	-	-	
Schaffer	**2D**	SPI^+	GI_Π^+	GI_Π^+	SPI^+	SPI^+
	3D	GI_N^+	GI_Π^+	GI_Π^+	SPI^+	
	10D	-	-	-	-	
Vincent	**2D**	GI_N^+	GI_Σ^+	GI_Π^+	SPI^+	SPI^+
	3D	SPI^+	GI_Π^+	SPI^+	SPI^+	
	10D	SPI^+	GI_Π^+	SPI^+	SPI^+	

Table 31: The table depicts the quality indicators that on average yield the best results for each quality indicator measurement for ELSA with $\nu = 0.5$ and $\omega = 0.1$ for σ step-size on level set benchmarks of different dimensions. Situations where the algorithm cannot find a whole, feasible population are marked with "-".

Quality indicator comparison based on coverage properties:

What remains to be discussed in this section is an analysis on the visually observed characteristics of a population generated from the use of a specific quality indicator. Figures 13 to 20 in the Appendix show the population which diversity is closest to the average diversity for a quality indicator used in each algorithm configuration on the 2D level set benchmarks. Similarly, the results for 3D can also be found in the Appendix in Figures 21 to 28.

Table 32 shows a summary of the visually observed traits on the level sets. A plus symbol indicates that a certain trait is present, whereas a minus symbol is used when it is not present. The amount of plus symbols reflects the severity of a certain trait for a quality indicator compared to others. The rating of the Incomplete Coverage trait for each quality indicator are based on the results in Table 33 that depicts data for Incomplete Coverage and the average amount of missing level set components.

Even Distribution is a positive trait that is aimed to be achieved for good coverage. Boundary Concentration is not strictly considered to be a desirable or undesirable trait for coverage. Clustering Interior and Incomplete Coverage, however, are negative traits and are aimed to be avoided. Clustering in small degrees is acceptable for a quality indicator, but a quality indicator will be rated poorly when this trait is expressed severely. Clustering Interior and Even Distribution are not treated as mutually exclusive traits because a quality indicator can stimulate even distribution in the population for some benchmark, but promote clustering in a different benchmark.

That SPI^+ has an emphasis to put solutions on the boundary might be a possible explana-

tion why this quality indicator can outperform native quality indicators in their own category as it makes use of the available level set space more efficiently. The results from SPI^+ and ADI^+ might be described to be similar, but they have a significant difference which is more visible in the figures for Ellipsoid, Hollow Sphere, Double Sphere, where the solutions from ADI^+ generated populations have a tendency to be unevenly spread out on the boundaries of the level sets while those from SPI^+ maintain an even distance from each other.
It is remarkable that ADI^+ which is designed as a quality indicator for maximising the representativeness of the approximation set to the target expresses a high degree of Boundary Concentration which is described as an effect from maximising diversity instead of something associated with coverage. A theory to elaborate this effect is that ADI^+ uses the entire search space as the reference set, therefore the result might not be representative to the actual target level set.

QI	Boundary	Clustering	Even Distribution	Incomplete Coverage
$\mathbf{GI}_N^+$	-	+	++	+
$\mathbf{GI}_\Sigma^+$	+	+++	+	+++
$\mathbf{GI}_\Pi^+$	+	+	++	++
$\mathbf{SPI}^+$	++	-*	+++	+
$\mathbf{ADI}^+$	++	-	++	+

Table 32: The table depicts quality indicator characteristics related to coverage that can be observed from the results on the 2D level set benchmarks. (*if θ in SPI^+ is set too small: ++ [5])

		MCS			ELSA			ELSA-SR$_{\omega=0.1}$			ELSA-SR$_{\omega=1}$		
		IC	**mean**	**std**	**IC**	**mean**	**std**	**IC**	**mean**	**std**	**IC**	**mean**	**std**
2D Rastrigin	$\mathbf{GI}_N^+$	**1**	1.00	0.00	**1**	1.00	0.00	5	1.00	0.00	**1**	1.00	0.00
	$\mathbf{GI}_\Sigma^+$	18	**1.33**	0.68	28	2.21	1.29	37	2.59	1.64	**16**	1.50	0.71
	$\mathbf{GI}_\Pi^+$	**2**	1.00	0.00	3	1.00	0.00	23	1.22	0.41	**2**	1.00	0.00
	$\mathbf{SPI}^+$	**0**	0.00	0.00	1	1.00	0.00	2	1.00	0.00	**0**	0.00	0.00
	$\mathbf{ADI}^+$	**0**	0.00	0.00	**0**	0.00	0.00	4	1.00	0.00	**0**	0.00	0.00
2D Vincent	$\mathbf{GI}_N^+$	**0**	0.00	0.00	3	1.00	0.00	2	1.00	0.00	2	1.00	0.00
	$\mathbf{GI}_\Sigma^+$	**4**	4.50	0.50	31	4.35	1.38	26	3.85	1.75	5	**2.40**	1.74
	$\mathbf{GI}_\Pi^+$	**0**	0.00	0.00	**0**	0.00	0.00	4	1.00	0.00	1	1.00	0.00
	$\mathbf{SPI}^+$	**0**	0.00	0.00	3	1.00	0.00	2	1.00	0.00	**0**	0.00	0.00
	$\mathbf{ADI}^+$	1	1.00	0.00	2	1.00	0.00	4	1.00	0.00	**0**	0.00	0.00
3D Branke's	$\mathbf{GI}_N^+$	**5**	1.00	0.00	13	1.15	0.36	12	1.00	0.00	8	1.00	0.00
	$\mathbf{GI}_\Pi^+$	**6**	1.00	0.00	9	1.11	0.31	11	1.18	0.39	11	1.00	0.00
	$\mathbf{SPI}^+$	**3**	1.00	0.00	8	1.00	0.00	14	1.07	0.26	9	1.11	0.31
3D Vincent	$\mathbf{GI}_N^+$	40	**11.75**	2.26	40	16.00	2.64	40	18.78	2.32	40	14.75	2.86
	$\mathbf{GI}_\Pi^+$	40	**12.70**	2.32	40	18.75	2.60	40	20.83	2.85	40	16.48	3.02
	$\mathbf{SPI}^+$	40	**12.18**	2.43	40	16.28	3.52	40	19.30	2.92	40	15.90	2.63
10D Double	$\mathbf{GI}_N^+$	-	-	-	33	1.00	0.00	27	1.00	0.00	**2**	1.00	0.00
	$\mathbf{GI}_\Pi^+$	-	-	-	23	1.00	0.00	28	1.00	0.00	**4**	1.00	0.00
	$\mathbf{SPI}^+$	-	-	-	30	1.00	0.00	30	1.00	0.00	**2**	1.00	0.00
30D Double	$\mathbf{GI}_N^+$	-	-	-	40	1.00	0.00	40	1.00	0.00	**39**	1.00	0.00
	$\mathbf{GI}_\Pi^+$	-	-	-	40	1.00	0.00	40	1.00	0.00	**39**	1.00	0.00
	$\mathbf{SPI}^+$	-	-	-	40	1.00	0.00	40	1.00	0.00	**37**	1.00	0.00

Table 33: The table represents Incomplete Coverage (IC), and of those occurrences, the mean and standard deviation of missing level set components.

Summary of the conclusions found from the quality indicator comparison:

Five quality indicators, which are GI_N^+, GI_Σ^+, GI_Π^+, SPI^+, and ADI^+, have been compared with each other with regards to diversity and coverage properties.

In terms of diversity properties and resemblance, the following pairs of quality indicators for population selection should be considered: SPI^+-ADI^+ (in ADI^+ measure) and GI^+_Π-GI^+_Σ (in GI^+_Σ measure).
Using a particular quality indicator for population selection does not necessarily yield a population that scores the best in its own corresponding quality indicator measurement as seen before (see Table 31 on page 57). Analysing the results throughout the different dimensions, it is observed that it is more common in higher dimensions that populations from GI^+_Π can outperform the other Augmented Gap Indicators in their own category, and in particular populations from SPI^+ can outperform all the other tested quality indicators. It is dependent on the characteristics of the level set benchmark, however since geometric shapes are more difficult to comprehend in higher dimensions, specific comment on them is left out.

In terms of coverage properties we see that some quality indicators distribute the solutions poorly, e.g. GI^+_Σ promotes extreme clustering. Since GI^+_Π distributes solutions similarly except expressing the negative traits for coverage to a lesser degree, it is recommended to be used over GI^+_Σ. The GI^+_Π indicator has the characteristics that solutions reside both on the interior and boundaries of the level set with a relatively even distribution.
The populations from ADI^+ and SPI^+ bear similar coverage traits, except the results from SPI^+ are superior in the Even Distribution trait. Similarly, SPI^+ can replace ADI^+ to achieve certain coverage properties in a population. The traits of SPI^+ is that it has the highest Boundary Concentration and Even Distribution compared to the other quality indicators.
GI^+_N is a quality indicator that expresses low Boundary Concentration and performs decently on Even Distribution. While it does not stand out in diversity comparisons and does not benefit at all from an adaptive step-size with ELSA-SR, GI^+_N should not be ignored as it has coverage properties distinctive to the other tested quality indicators. It should be remarked that the boundary represents the maximum allowed threshold, and for practical problems it might be undesirable to have many solutions far away from the optimum objective value.
None of the quality indicators can avoid Incomplete Coverage in the tests, although the occurrences are relatively low in the 2D benchmarks, while much higher starting from the 3D benchmarks.

4.4.2 Comparison between the algorithms

The results from MCS, ELSA, ELSA-$SR_{\omega=0.1}$, and ELSA-$SR_{\omega=1}$ on the 2D, 3D, 10D and 30D level set benchmarks are compared with each other in this section. The main criteria for algorithm comparison described in Section 4.1 consisting of $Eval_{Feasible}$, diversity, and coverage are used. These criteria split the text into three parts which describe in detail how the algorithms perform in that category. At the end of Section 4.4.2, the most important findings are summarised.

Comparison based on $Eval_{Feasible}$:

Table 34 represents the $Eval_{Feasible}$ values of the results, where $Eval_{Feasible}$ is calculated through taking the average from the average of $Eval_{Feasible}$ measured from the following three quality indicators: GI^+_N, GI^+_Π, and SPI^+.

2D benchmarks:

Experiment results in Table 34 illustrate that the 2D level set benchmarks are easy to solve as

even MCS can find feasible populations within the evaluation budget. The 2D Lamé and Vincent benchmarks require the most evaluations, with slightly over 1000 evaluations; 2D Rastrigin and Schaffer are the easiest to locate with less than 300 evaluations, and the remaining 2D level set benchmarks require approximately 600 evaluations for MCS. The $\text{Eval}_{Feasible}$ results from ELSA-$\text{SR}_{\omega=1}$ resemble those from MCS on 2D benchmarks. But both ELSA and ELSA-$\text{SR}_{\omega=0.1}$ are quicker at finding feasible populations than MCS and the improvements are proportional to the level set difficulties found from MCS, with the largest improvements witnessed for Lamé and Vincent, while the improvements for Rastrigin and Schaffer are almost non-existent. ELSA and ELSA-$\text{SR}_{\omega=0.1}$ share similar $\text{Eval}_{Feasible}$ values for the 2D level set benchmarks.

3D benchmarks:

The four tested algorithms still manage to find diverse feasible populations on 3D level set benchmarks within the evaluation budget of 10K, although 3D Lamé is the only benchmark which MCS does not consistently manage to find a whole, feasible population for. In general it is noticeable that MCS is beginning to struggle with locating level sets for a relatively small dimension like 3D when the needed amount of evaluations has grown in the thousands range, while 1090 is the highest encountered average for ELSA (for 3D Schaffer and Vincent) and 1330 for ELSA-$\text{SR}_{\omega=0.1}$ (for 3D Vincent). The $\text{Eval}_{Feasible}$ values from ELSA-$\text{SR}_{\omega=1}$ are still quite similar to those from MCS, although its results from 3D Lamé, Branke's Multipeak and Schaffer are noticeably lower but not as low as the results from ELSA or ELSA-$\text{SR}_{\omega=0.1}$.

Analysing the feasibility of MCS populations on 3D Lamé in greater detail, it shows a Population Feasibility Rate of 88.3% out of 120 populations and a Solution Feasibility Rate of 93.3%. The average $\text{Eval}_{Feasible}$ value is 8675 with a standard deviation of 696 when measured over the whole, feasible MCS populations for 3D Lamé.

10D benchmarks:

The 10D level set benchmarks prove to be too difficult for MCS where it cannot find any feasible populations at all. The difficulty of the level sets themselves are varied. ELSA can solve some 10D level set benchmarks successfully such as Ellipsoid, Hollow Sphere, Double Sphere, and Vincent, however it cannot find any whole, feasible populations for Lamé, Rastrigin, and Schaffer. ELSA-$\text{SR}_{\omega=0.1}$ can find feasible populations for all 10D benchmarks except for 10D Schaffer, although the populations of this algorithm with SPI^{+} all happen to be feasible. ELSA-$\text{SR}_{\omega=1}$ has found whole, feasible populations for all 10D level set benchmarks.

The Solution Feasibility Rates for MCS are often far lower than 1% or exactly at 0%, with 10D Vincent being a rare case with a slightly higher Solution Feasibility Rate of approximately 15%.
It is also apparent that ELSA is struggling severely with those three 10D benchmarks: the Solution Feasibility Rate for Lamé is lower than 1%, for Rastrigin it is approximately 11%, and for Schaffer it is exactly 0%.
The Population Feasibility Rate for ELSA-$\text{SR}_{\omega=0.1}$ on 10D Schaffer is 96%. The corresponding average $\text{Eval}_{Feasible}$ value is 48897 with a standard deviation of 6804 when measured over the whole, feasible ELSA-$\text{SR}_{\omega=0.1}$ population. The Solution Feasibility Rate of ELSA-$\text{SR}_{\omega=0.1}$ on 10D Schaffer is also exactly 0%.

The 10D benchmarks that can be solved easily (Ellipsoid, Hollow Sphere, Double Sphere, and Vincent) have $\text{Eval}_{Feasible}$ values that are lower than 5000 for ELSA and ELSA-$\text{SR}_{\omega=0.1}$. The

more complex 10D benchmarks take tens of thousands of evaluations for ELSA-SR which justifies the need of a larger evaluation budget for high dimensions. ELSA-SR$_{\omega=1}$ has significantly higher Eval$_{Feasible}$ values with all of them in the ten-thousands.

30D benchmarks:

The 30D level set benchmarks are to challenge the limitations of the level set approximation algorithms on a significantly higher dimension. The easier level set benchmarks like Ellipsoid are shown to be solve easily by ELSA and both ELSA-SR settings on 10D and this remains the case even for 30D as the algorithms are successful at finding diverse feasible populations at all times. The required Eval$_{Feasible}$ has risen up to tens of thousands for ELSA and ELSA-SR, however more than half the evaluations of the evaluation budget are still left for diversity optimisation. It is expected that 30D Lamé, Rastrigin, and Schaffer are even more difficult than their 10D variants, and the results reflect this as none of the algorithms can consistently find feasible populations for them. ELSA has exactly 0 feasible populations on those three level set benchmarks and the Solution Feasibility Rates are also exactly 0%. Both versions of ELSA-SR almost produce similar results, except ELSA-SR$_{\omega=0.1}$ barely manages to find 3 feasible populations for 30D Lamé (Solution Feasibility Rate: 3%), and ELSA-SR$_{\omega=1}$ barely manages to find 7 feasible populations for 30D Rastrigin (Solution Feasibility Rate: 0%).

		MCS		**ELSA**		**ELSA-SR$_{\omega=0.1}$**		**ELSA-SR$_{\omega=1}$**	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
2D	**Lamé**	1352	131	**557**	51	577	51	1343	126
	Ellipsoid	568	51	**360**	31	361	30	570	60
	Hollow	637	53	425	32	**423**	30	644	60
	Double	577	52	**371**	31	373	31	573	45
	Branke's	535	52	**370**	28	379	29	536	47
	Rastrigin	262	19	**242**	17	245	19	259	21
	Schaffer	278	21	256	21	**256**	20	275	21
	Vincent	1177	113	**723**	62	797	93	1167	107
3D	**Lamé**	-	-	**1074**	112	1277	169	4116	199
	Ellipsoid	1035	98	**468**	42	476	46	1025	94
	Hollow	1271	121	**537**	47	559	47	1240	111
	Double	2575	275	**693**	71	726	84	2396	178
	Branke's	4472	434	**961**	94	1093	137	3356	209
	Rastrigin	1055	101	**712**	67	731	73	1043	96
	Schaffer	6353	623	**1090**	118	1266	159	3779	200
	Vincent	2835	303	**1090**	91	1330	121	2785	237
10D	**Lamé**	-	-	-	-	**36216**	1577	73263	539
	Ellipsoid	-	-	**3099**	487	3117	524	26537	438
	Hollow	-	-	2723	343	**2697**	366	23132	414
	Double	-	-	**4329**	492	4634	609	34690	406
	Rastrigin	-	-	-	-	**22106**	1848	62865	859
	Schaffer	-	-	-	-	-	-	**85714**	626
	Vincent	-	-	**3136**	513	3505	521	35253	1307
30D	**Lamé**	-	-	-	-	-	-	-	-
	Ellipsoid	-	-	**16533**	1522	20784	1712	41520	923
	Hollow	-	-	**13561**	771	15938	1291	38765	978
	Double	-	-	**24677**	2043	31927	2523	51927	1186
	Rastrigin	-	-	-	-	-	-	-	-
	Schaffer	-	-	-	-	-	-	-	-
	Vincent	-	-	**10136**	1943	11142	1364	40911	1549

Table 34: The table depicts the Eval$_{Feasible}$ values for MCS, ELSA, ELSA-SR$_{\omega=0.1}$ and ELSA-SR$_{\omega=1}$ on the 2D, 3D, 10D, and 30D level set benchmarks. The best configuration is bolded.

Comparison based on diversity:

The hierarchy among the four algorithm configurations for the best diversity is marginally less clear than the hierarchy for the lowest $\text{Eval}_{Feasible}$, however there are enough patterns in the results to make general statements about them. The populations with the best diversity are bolded in Table 35 (2D results), Table 36 (3D results), Table 37 (10D results), and Table 38 (30D results). The depicted diversity is measured with the quality indicator that corresponds to the one which is used for population selection.

Diversity comparison between algorithms in 2D and 3D benchmarks:

Populations from MCS have the lowest diversity over all the level set benchmarks on their tested dimensions, however it can produce slightly more diverse populations than ELSA for level set benchmarks with large surface areas or volumes such as 2D Rastrigin, 2D Schaffer, and 3D Rastrigin. Results from ELSA sit in the middle as being better than the results from MCS and worse than results from either of the ELSA-SR configurations, however GI_N^+ populations from ELSA often have the best diversity on 2D and 3D level set benchmarks. Which of the ELSA-SR configurations yield the best diversity depends on the level set benchmarks.
For the 2D level set benchmarks (for populations from GI_Σ^+, GI_Π^+, SPI^+, and ADI^+) the most diverse populations are generated by ELSA-$\text{SR}_{\omega=0.1}$.
Regarding the 3D benchmarks, ELSA-$\text{SR}_{\omega=0.1}$ is superseded by ELSA-$\text{SR}_{\omega=1}$ on Rastrigin and Vincent; and ELSA-$\text{SR}_{\omega=0.1}$ is superseded by ELSA on Schaffer for generating the most diverse populations.

Diversity comparison between algorithms in 10D and 30D benchmarks:

Most of the populations from ELSA-$\text{SR}_{\omega=1}$ have the best diversity for 10D level set benchmarks, but ELSA-$\text{SR}_{\omega=0.1}$ performs better on 10D Lamé and 10D Hollow Sphere (except the case for the results with GI_N^+ on 10D Hollow Sphere). If the few infeasible populations from ELSA-$\text{SR}_{\omega=0.1}$ for 10D Schaffer are ignored, then the diversity of the whole, feasible populations from ELSA-$\text{SR}_{\omega=0.1}$ is higher than the diversity from ELSA-$\text{SR}_{\omega=1}$. The step-size normalisation with $\omega = 1$ is considered too large when used for ELSA, however in combination with step-size adaptation in ELSA-SR, it can produce diverse populations for level set benchmarks of high dimension.

None of the algorithms can solve all 30D level set benchmarks, so the ranking of the best diversity refers to both the actual diversity of a whole, feasible population and the closeness of an algorithm at finding a whole, feasible population. For 30D level set benchmarks the populations from ELSA-$\text{SR}_{\omega=0.1}$ have the best diversity for the simple shape level set benchmarks (Lamé, Ellipsoid, Hollow Sphere and Double Sphere) and ELSA-$\text{SR}_{\omega=1}$ have the best diversity for the complex shape level set benchmarks (Rastrigin, Schaffer, and Vincent).

Level set benchmarks which have different algorithm configurations tested as best for 10D and 30D have been encountered, as 10D Ellipsoid and Double Sphere favour ELSA-$\text{SR}_{\omega=1}$, but their 30D variants favour ELSA-$\text{SR}_{\omega=0.1}$. It is likely explained by the phenomenon where the volume of a unit sphere rapidly decreases as compared to the volume of the unit cube from dimension 6 onwards. As seen in Section 3, the volume of 30D Ellipsoid and Double Sphere level set are indeed smaller than their 10D versions. A smaller σ step-size might be more beneficial to locate volumes that are small, and more specifically if the ratio of the volume in search space is extremely small.

Diversity comparison between benchmarks of different dimensions:

Examining the values from the quality indicators over all dimensions show patterns that the algorithms are affected by the changing level sets. Take for example the diversity yield for the populations in 2D level set benchmarks in comparison with the diversities found in 3D level set benchmarks. The results from 3D benchmarks have higher diversity values for all quality indicators than in 2D benchmarks for the most part and the differences between the algorithms for the GI_N^+ and GI_Π^+ categories are larger in 3D benchmarks than in 2D benchmarks. However, the Schaffer benchmark is an exception with a rather unpredictable level set appearance as it drastically changes between dimensions.

Purely looking at the nD rectangular search space which increases its volume when going to higher dimensions, the potential distances between solutions go up, so that the cap of the simple gap indicator value is also raised for level set benchmarks of higher dimensions (SPI^+ also relies on distances for its calculation but it is a measure that represents the amount of sub-species which is always capped by the population size).
The differences between the diversities of algorithms with SPI^+ are not larger in 3D benchmarks than in 2D benchmarks, as the values quickly approach 100 which is already the case for the 2D benchmarks but even more so in the 3D benchmarks. On the other hand when SPI^+ measurements heavily depend on decimal precision to distinguish them, SPI^+ is not a meaningful descriptor on its own that provides any information on what the diversity in the level set is like.

Nevertheless, the quality indicator values are not perpetually increasing in size when going into higher dimension benchmarks. Lamé and Rastrigin both show signs of decreasing quality indicator values from 10D onwards, and Schaffer from 3D onwards. Only Ellipsoid, Double Sphere, Hollow Sphere, and Vincent have perpetually larger GI_Π^+ and SPI^+ diversity values for higher dimension benchmarks. For these benchmarks the GI_N^+ values are mostly increasing as well, but there are cases where they decrease in higher dimension benchmarks. The reason why the decrease occurs might be because the volume of the level sets are decreasing in higher dimensions, which is measured to be the case for some benchmarks: for reference compare the volumes between the 10D and 30D benchmarks in Section 3. Another potential reason is that the ratio between the volume of the level set and the volume of the search space is much smaller for higher dimensions, which makes it more difficult to locate the level set and in turn leaves less evaluations to focus on diversity optimisation.

		MCS		ELSA		ELSA-SR$_{\omega=0.1}$		ELSA-SR$_{\omega=1}$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Lamé	$\mathbf{GI}^+_N$	0.126	0.003	**0.139**	0.002	0.137	0.002	0.136	0.002
	$\mathbf{GI}^+_\Sigma$	0.168	0.003	0.176	0.002	**0.183**	0.002	0.177	0.002
	$\mathbf{GI}^+_\Pi$	0.161	0.002	0.172	0.002	**0.179**	0.002	0.172	0.002
	$\mathbf{SPI}^+$	52.941	0.611	54.068	0.685	**55.131**	0.556	54.565	0.521
	$\mathbf{ADI}^+$	-1.146	0.017	-1.146	0.018	**-1.126**	0.012	-1.136	0.014
Ellipsoid	$\mathbf{GI}^+_N$	0.200	0.003	**0.209**	0.002	0.207	0.003	0.206	0.003
	$\mathbf{GI}^+_\Sigma$	0.243	0.002	0.251	0.002	**0.260**	0.002	0.253	0.002
	$\mathbf{GI}^+_\Pi$	0.243	0.002	0.252	0.002	**0.261**	0.002	0.252	0.002
	$\mathbf{SPI}^+$	74.313	0.164	74.860	0.130	**75.652**	0.088	75.005	0.106
	$\mathbf{ADI}^+$	-0.997	0.001	-0.992	0.001	**-0.988**	0.000	-0.992	0.001
Hollow	$\mathbf{GI}^+_N$	0.197	0.003	**0.206**	0.003	0.205	0.003	0.203	0.003
	$\mathbf{GI}^+_\Sigma$	0.249	0.003	0.253	0.003	**0.265**	0.004	0.258	0.003
	$\mathbf{GI}^+_\Pi$	0.248	0.003	0.256	0.003	**0.269**	0.002	0.258	0.002
	$\mathbf{SPI}^+$	78.088	0.162	78.810	0.135	**79.970**	0.088	79.088	0.125
	$\mathbf{ADI}^+$	-0.751	0.001	-0.747	0.001	**-0.743**	0.000	-0.746	0.001
Double	$\mathbf{GI}^+_N$	0.202	0.003	**0.211**	0.003	0.210	0.003	0.209	0.003
	$\mathbf{GI}^+_\Sigma$	0.251	0.002	0.256	0.002	**0.267**	0.002	0.259	0.002
	$\mathbf{GI}^+_\Pi$	0.249	0.002	0.257	0.002	**0.267**	0.002	0.258	0.002
	$\mathbf{SPI}^+$	76.702	0.206	77.346	0.137	**78.294**	0.072	77.538	0.146
	$\mathbf{ADI}^+$	-0.904	0.002	-0.900	0.001	**-0.895**	0.000	-0.899	0.001
Branke's	$\mathbf{GI}^+_N$	0.148	0.002	**0.152**	0.002	0.151	0.002	0.150	0.002
	$\mathbf{GI}^+_\Sigma$	0.195	0.003	0.197	0.003	**0.206**	0.004	0.200	0.003
	$\mathbf{GI}^+_\Pi$	0.189	0.002	0.192	0.002	**0.202**	0.002	0.195	0.002
	$\mathbf{SPI}^+$	63.284	0.419	63.514	0.306	**65.165**	0.226	64.041	0.326
	$\mathbf{ADI}^+$	-0.341	0.003	-0.340	0.003	**-0.334**	0.002	-0.338	0.002
Rastrigin	$\mathbf{GI}^+_N$	0.614	0.008	**0.620**	0.010	0.619	0.012	0.614	0.012
	$\mathbf{GI}^+_\Sigma$	0.826	0.006	0.822	0.007	**0.843**	0.009	0.837	0.005
	$\mathbf{GI}^+_\Pi$	0.819	0.005	0.818	0.005	**0.842**	0.007	0.830	0.006
	$\mathbf{SPI}^+$	99.906	0.003	99.905	0.004	**99.922**	0.003	99.914	0.004
	$\mathbf{ADI}^+$	-0.364	0.001	-0.364	0.001	**-0.361**	0.002	-0.362	0.001
Schaffer	$\mathbf{GI}^+_N$	0.353	0.005	0.352	0.005	0.352	0.006	**0.354**	0.005
	$\mathbf{GI}^+_\Sigma$	0.443	0.003	0.442	0.004	**0.455**	0.004	0.450	0.003
	$\mathbf{GI}^+_\Pi$	0.441	0.003	0.440	0.003	**0.454**	0.003	0.446	0.003
	$\mathbf{SPI}^+$	96.273	0.038	96.280	0.062	**96.555**	0.032	96.404	0.041
	$\mathbf{ADI}^+$	-0.223	0.000	-0.223	0.000	**-0.221**	0.000	-0.222	0.000
Vincent	$\mathbf{GI}^+_N$	0.115	0.003	0.119	0.003	**0.120**	0.003	0.120	0.002
	$\mathbf{GI}^+_\Sigma$	0.187	0.003	0.193	0.006	**0.204**	0.005	0.197	0.003
	$\mathbf{GI}^+_\Pi$	0.163	0.002	0.167	0.002	**0.180**	0.003	0.173	0.002
	$\mathbf{SPI}^+$	64.268	0.403	65.044	0.402	**67.698**	0.330	65.988	0.289
	$\mathbf{ADI}^+$	-0.353	0.001	-0.350	0.001	**-0.342**	0.001	-0.347	0.001

Table 35: The table depicts the average diversities for 2D benchmarks. Each row represents the quality indicator which is both used to generate the population and the measure for its diversity. Each column represents an algorithm configuration. The algorithm-QI combination that yields the best diversity is bolded.

		MCS		ELSA		ELSA-SR$_{\omega=0.1}$		ELSA-SR$_{\omega=1}$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Lamé	$\mathbf{GI}_N^+$	-7.956	24.239	**0.267**	0.004	0.252	0.004	0.249	0.004
	$\mathbf{GI}_\Pi^+$	-15.411	41.893	0.326	0.006	**0.348**	0.005	0.326	0.007
	$\mathbf{SPI}^+$	68.961	2.642	86.442	0.448	**88.932**	0.581	86.607	0.671
Ellipsoid	$\mathbf{GI}_N^+$	0.498	0.007	**0.532**	0.004	0.513	0.006	0.517	0.007
	$\mathbf{GI}_\Pi^+$	0.592	0.004	0.625	0.004	**0.668**	0.006	0.647	0.005
	$\mathbf{SPI}^+$	99.198	0.029	99.381	0.016	**99.553**	0.011	99.479	0.015
Hollow	$\mathbf{GI}_N^+$	0.479	0.008	**0.518**	0.006	0.495	0.008	0.499	0.006
	$\mathbf{GI}_\Pi^+$	0.575	0.007	0.614	0.005	**0.662**	0.005	0.636	0.007
	$\mathbf{SPI}^+$	99.131	0.043	99.362	0.016	**99.554**	0.013	99.469	0.016
Double	$\mathbf{GI}_N^+$	0.334	0.007	**0.391**	0.004	0.378	0.005	0.373	0.005
	$\mathbf{GI}_\Pi^+$	0.413	0.006	0.463	0.003	**0.498**	0.004	0.476	0.003
	$\mathbf{SPI}^+$	95.128	0.203	96.709	0.077	**97.407**	0.040	97.036	0.065
Branke's	$\mathbf{GI}_N^+$	0.185	0.006	**0.230**	0.004	0.211	0.006	0.215	0.005
	$\mathbf{GI}_\Pi^+$	0.254	0.009	0.295	0.007	**0.315**	0.011	0.309	0.008
	$\mathbf{SPI}^+$	80.179	1.267	86.537	0.731	**88.635**	0.991	87.754	0.896
Rastrigin	$\mathbf{GI}_N^+$	**1.317**	0.020	1.249	0.020	1.242	0.022	1.286	0.019
	$\mathbf{GI}_\Pi^+$	1.549	0.014	1.506	0.023	1.585	0.023	**1.605**	0.018
	$\mathbf{SPI}^+$	100.000	0.000	100.000	0.000	100.000	0.000	**100.000**	0.000
Schaffer	$\mathbf{GI}_N^+$	0.211	0.015	**0.322**	0.008	0.289	0.009	0.290	0.009
	$\mathbf{GI}_\Pi^+$	0.302	0.016	**0.395**	0.005	0.393	0.010	0.385	0.011
	$\mathbf{SPI}^+$	84.973	1.839	**93.275**	0.434	92.655	0.702	92.310	0.626
Vincent	$\mathbf{GI}_N^+$	0.347	0.015	**0.361**	0.013	0.322	0.013	0.347	0.015
	$\mathbf{GI}_\Pi^+$	0.518	0.015	0.525	0.016	0.560	0.018	**0.573**	0.017
	$\mathbf{SPI}^+$	98.736	0.243	98.920	0.202	99.215	0.149	**99.268**	0.117

Table 36: The table depicts the average diversities for 3D benchmarks. Each row represents the quality indicator which is both used to generate the population and the measure for its diversity. Each column represents an algorithm configuration. The algorithm-QI combination that yields the best diversity is bolded.

		MCS		ELSA		ELSA-SR$_{\omega=0.1}$		ELSA-SR$_{\omega=1}$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Lamé	$\mathbf{GI}_N^+$	-2199	3	-1925	10	**0.135**	0.002	0.125	0.002
	$\mathbf{GI}_\Pi^+$	-2199	3	-1923	11	**0.237**	0.004	0.179	0.003
	$\mathbf{SPI}^+$	-302	4	-32	1	**53.969**	1.742	31.369	1.467
Ellipsoid	$\mathbf{GI}_N^+$	-2027	13	1.134	0.031	1.080	0.032	**1.398**	0.016
	$\mathbf{GI}_\Pi^+$	-2027	16	1.663	0.014	1.743	0.022	**1.789**	0.008
	$\mathbf{SPI}^+$	-140	5	100.000	0.000	100.000	0.000	**100.000**	0.000
Hollow	$\mathbf{GI}_N^+$	-1936	27	1.265	0.035	1.163	0.042	**1.539**	0.015
	$\mathbf{GI}_\Pi^+$	-1945	20	1.760	0.005	**1.859**	0.007	1.854	0.004
	$\mathbf{SPI}^+$	-68	5	100.000	0.000	**100.000**	0.000	100.000	0.000
Double	$\mathbf{GI}_N^+$	-2584	21	0.828	0.020	0.851	0.027	**0.905**	0.020
	$\mathbf{GI}_\Pi^+$	-2587	12	0.966	0.032	1.060	0.034	**1.091**	0.034
	$\mathbf{SPI}^+$	-692	15	99.948	0.018	99.962	0.013	**99.984**	0.007
Rastrigin	$\mathbf{GI}_N^+$	-7336	82	-3276	443	0.317	0.041	**0.359**	0.017
	$\mathbf{GI}_\Pi^+$	-7356	98	-3096	546	0.419	0.056	**0.516**	0.044
	$\mathbf{SPI}^+$	-4531	71	-601	286	84.655	14.565	**97.247**	1.706
Schaffer	$\mathbf{GI}_N^+$	-2441	8	-1748	31	-170	510	**0.058**	0.001
	$\mathbf{GI}_\Pi^+$	-2440	9	-1744	41	-42	263	**0.074**	0.001
	$\mathbf{SPI}^+$	-859	8	-176	52	**6.720**	0.985	5.458	0.301
Vincent	$\mathbf{GI}_N^+$	-1204	66	0.653	0.029	0.620	0.039	**0.831**	0.028
	$\mathbf{GI}_\Pi^+$	-1211	56	0.943	0.102	1.083	0.107	**1.364**	0.084
	$\mathbf{SPI}^+$	10	4	99.933	0.041	99.953	0.030	**99.996**	0.004

Table 37: The table depicts the average diversities for 10D benchmarks. Each row represents the quality indicator which is both used to generate the population and the measure for its diversity. Each column represents an algorithm configuration. The algorithm-QI combination that yields the best diversity is bolded.

		MCS		ELSA		ELSA-SR$_{\omega=0.1}$		ELSA-SR$_{\omega=1}$	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
Lamé	**GI^+_N**	-4734	5	-3704	42	**-3330**	109	-3368	10
	GI$^+_\Pi$	-4734	4	-3702	48	**-3000**	923	-3372	15
	SPI$^+$	-1449	6	-410	38	**-57**	49	-82	15
Ellipsoid	**GI^+_N**	-4862	25	1.194	0.036	**1.239**	0.050	1.231	0.044
	GI$^+_\Pi$	-4863	21	2.279	0.016	**2.403**	0.018	2.387	0.019
	SPI$^+$	-1573	24	100.000	0.000	**100.000**	0.000	100.000	0.000
Hollow	**GI^+_N**	-3773	3	1.372	0.027	**1.461**	0.050	1.452	0.051
	GI$^+_\Pi$	-3773	3	2.132	0.005	**2.208**	0.006	2.194	0.005
	SPI$^+$	-487	3	100.000	0.000	**100.000**	0.000	100.000	0.000
Double	**GI^+_N**	-8903	43	**0.884**	0.018	0.808	0.027	0.798	0.026
	GI$^+_\Pi$	-8914	34	1.146	0.003	**1.222**	0.003	1.208	0.007
	SPI$^+$	-5625	37	99.985	0.000	**99.987**	0.000	99.987	0.001
Rastrigin	**GI^+_N**	-35527	177	-21395	1348	-11352	1230	**-6103**	1215
	GI$^+_\Pi$	-35556	175	-21175	1120	-10991	1304	**-5726**	1773
	SPI$^+$	-30646	194	-16182	1174	-6221	1216	**-1217**	885
Schaffer	**GI^+_N**	-6992	14	-4928	367	-4552	447	**-3165**	105
	GI$^+_\Pi$	-6991	16	-4968	276	-4455	502	**-3171**	95
	SPI$^+$	-4254	14	-2180	423	-1704	541	**-430**	82
Vincent	**GI^+_N**	-2494	0	0.713	0.060	0.660	0.079	**0.884**	0.074
	GI$^+_\Pi$	-2494	0	1.291	0.124	1.513	0.113	**1.791**	0.093
	SPI$^+$	-29	0	99.998	0.003	100.000	0.000	**100.000**	0.000

Table 38: The table depicts the average diversities for 30D benchmarks. Each row represents the quality indicator which is both used to generate the population and the measure for its diversity. Each column represents an algorithm configuration. The algorithm-QI combination that yields the best diversity is bolded.

Comparison based on coverage:

This section focuses on the appearance of the population distributions on the level set benchmarks by visual inspection and will be further divided into three parts: (1) a general overview on the coverage from populations on the 2D level set benchmarks, (2) a general overview on the coverage of populations on the 3D level set benchmarks, and (3) a detailed overview on the Incomplete Coverage in all the level set benchmarks that contain multiple level set components. In both the overviews of the coverage in 2D and 3D benchmarks first provides a description of the overall appearance of populations on a level set benchmark, followed by a comparison of what differences can be observed between the algorithms. Again, Figures 13 to 20 are referred to for the 2D benchmarks and Figures 21 to 28 are referred to for the 3D benchmarks. The discussion is solely based on those populations depicted in the figures (which have a diversity measure that is closest to the average diversity measure from the runs with that setting) and not over all the populations that have been yield over every run of an algorithm.

Coverage in 2D level set benchmarks:

Overall, all of the tested algorithms with the relevant quality indicators (GI^+_N, GI^+_Π, and SPI^+) yield populations with very good coverage on all the 2D level set benchmarks. All of these populations have the entire surface of non-disjoint level set benchmarks covered, where it is a balanced mix of outlining the boundary of the level set and having solutions in the centre of the level set. As for the level sets benchmarks that consist of multiple level set components, all of the benchmarks have their level set components covered (with the exception for some cases with the GI^+_Σ in 2D Rastrigin). The amount of solutions on each level set component is proportional to the size of the surface, and this is reflected by 2D Double Sphere which has two level set components of equal size and the amount of solutions spread over the two spheres are

approximately the same.

The populations on the 2D level set benchmarks from the different algorithms mostly share similar topographic appearances when comparing them with each other for a specific quality indicator. The text below discusses cases where differences can be seen and where the differences lie. As the tested algorithms can solve the 2D level set benchmarks relatively well, the differences are often not too significant.

For GI_N^+ the algorithms are almost interchangeable and there are no observable differences.

For GI_Π^+ there are no huge differences either. There are however benchmarks where the populations of the algorithms share minor differences, among them are 2D Lamé, Ellipsoid, and Branke's Multipeak. Here, the populations from ELSA-SR$_{\omega=0.1}$ which usually got a higher diversity GI_Π^+ than the other algorithms have a more organised, evenly distributed appearance.
In general 2D Vincent is a benchmark where the populations from algorithms share different appearances, but these are randomly determined depending on which level set components are found during the exploration.
In Lamé, despite none of the four populations depict an extremely organised appearance, the GI_Π^+ population from MCS has visibly more solutions that are denser towards each other while the other algorithms utilise the space better. That the GI_Π^+ population from ELSA-SR$_{\omega=0.1}$ shows slight signs of the Clustering Interior trait by having more of an aversion to put solutions on the acute spaces than the other algorithm seems to be a coincidence, as this occurrence is found in multiple cases from all the tested algorithms regardless of their diversity. It is not clear whether it is more common for high diversity or not, such that optimising the GI_Π^+ would push for this trait.
For Ellipsoid, the population from ELSA-SR$_{\omega=0.1}$ shows a clearer divide of solution markers that are either on the boundary or in the central area of the level set, while with the populations from the other algorithms there are solution markers that are near the boundary but not stationed on it. Here, the population from ELSA-SR$_{\omega=0.1}$ evokes a higher expression of the Even Distribution trait.
In Branke's Multipeak, the populations are very similar and the differences are mostly observed in the circle shape, where the solutions from ELSA-SR$_{\omega=0.1}$ in that level set component appear to be slightly more evenly distributed than the other algorithms.

For SPI^+ the differences between the algorithms are very pronounced in the 2D level set benchmarks except for Rastrigin, Schaffer, and Vincent. The populations with the highest SPI^+ diversity, generated from ELSA-SR$_{\omega=0.1}$ accordingly, have an extremely organised evenly distributed appearance.
The biggest difference between algorithms is seen for 2D Lamé, where the SPI^+ population from MCS is visibly chaotic unlike the neatly organised population from ELSA-SR$_{\omega=0.1}$, and the results from the remaining algorithms ELSA and ELSA-SR$_{\omega=1}$ are relatively similar and come close to ELSA-SR$_{\omega=0.1}$ but are slightly less organised. What has been said about the benchmarks solved with GI_Π^+ applies to the SPI^+ populations as well. Hollow Sphere and Double Sphere are benchmarks where the difference between the SPI^+ populations are slightly more visible than for GI_Π^+.

It is observed that the differences between the algorithms (as seen in results from GI_Π^+ and SPI^+) are most highlighted by level set benchmarks that contains a large non-disjoint surface area, while the populations appear homogeneous for other types of level set benchmarks.
The appearance is directly reflected from the quality indicator measurement, such as if the dif-

ferences from the diversity values are large then the topographical differences in the population are more noticeable, where the population with a higher diversity value gives a more evenly distributed impression.

Coverage in 3D level set benchmarks:

For the 3D benchmarks, each population from an algorithm run with a specific quality indicator is made up from two images. The figure on the top provides a 3D view of the population on the level set which is depicted in light grey and semi-transparent, where the RGB-value of a solution maps to the (x_1, x_2, x_3)-coordinate of the solution with respect to the search space. Infeasible solutions are marked as a red asterisk. The figure on the bottom represents an orthographic 2D projection, where gray scale coding is used to determine the location of a solution with respect to the x_3-axis.

The limitations of the algorithms are made more apparent in the 3D level set benchmarks.
In 3D Lamé, the algorithms have difficulties in evenly distributing solutions in the acute parts of the level set and those areas are noticeably sparser unlike in 2D Lamé. It can be described as such that all the depicted populations do not have solutions with coordinate values lower than approximately -2 or higher than approximately 2 (the 3D Lamé level set allows coordinate values in the range of -3 to 3).
All depicted populations for 3D Branke's Multipeak except one (ELSA-SR$_{\omega=0.1}$ with GI$_N^+$) have all its level set components covered by solutions. The populations from GI$_N^+$ and SPI$^+$ share surprisingly many similarities, and they both have a considerable amount of solutions distributed on each level set component relative to their sizes. It is debatable whether the distribution in the GI$_\Pi^+$ populations that score high in this diversity measure are desirable, such as GI$_\Pi^+$ can show signs of Clustering Interior (resembling GI$_\Sigma^+$) in extreme cases, while such occurrences have not been encountered in the 2D benchmarks.
For 3D Rastrigin the majority of the solutions are settled on the central waffle structure, however all the algorithms are able to settle a few solutions on the outer rim disjoint parts.
3D Schaffer, which has a different shape than 2D Schaffer, bears resemblance to 3D Lamé in terms of general shape with the extra characteristic that this level set benchmark consists of many disjoint components. The same issue to 3D Lamé is encountered for 3D Schaffer as well, however the algorithms manage to distribute solutions on the exterior parts to some degree despite the lack of connectivity.
3D Vincent has 64 disjoint level set parts and none of the populations can cover them all, although the overall diversity is good.
As for the 3D level set benchmarks in which all four algorithm configurations perform well on, they are Ellipsoid, Hollow Sphere and Double Sphere (the simple shape level set benchmarks with the exclusion of Lamé). For all of these benchmarks there is a large variety of colours visible in the 3D view and different greytones in the orthographic view to signify good diversity. The populations of 3D Double Sphere have a roughly equal amount of solutions spread over the two spheres as well.

The difficulty of interpreting 3D space brings challenges to identify the differences between the populations of algorithms. In the 2D analysis, results from quality indicators are classed together and algorithms have been compared in rank-wise for a specific quality indicator, as quality indicators come with reoccurring coverage properties (as described in Table 32 in Section 4.4.1). However in 3D level set benchmarks, sometimes the distribution differences between quality indicators are not clear. Therefore the analysis does not treat quality indicators as main categories and a description is provided for each 3D benchmark that highlight the combina-

tions of an algorithm with a quality indicator that stand out (either positively or negatively). Extra attention is brought to 3D Lamé, Branke's Multipeak, Schaffer, and Vincent in particular.

Overall for 3D Lamé, MCS with any quality indicator appears to be worst compared to the other algorithms. The population of MCS with GI_N^+ has most solutions clustered at the centre of the level set with barely any solution placed on the acute parts. MCS with GI_Π^+ is the sole population in 3D Lamé with infeasible solutions. According to the earlier seen Population Feasibility Rate, there are not that many infeasible populations in MCS, but it is likely explained that the penalty from infeasible solutions weights the diversity done, so that taking a population with a diversity measure that is closest to the mean it will skew towards a population with infeasible solutions. The population of MCS with SPI^+ has similar issues as described for this algorithm with GI_N^+, however it looks more evenly distributed with less clustering and more solutions on the acute parts.
The other three algorithms do not differ significantly from each other in appearance when comparing them for a specific quality indicator. Their populations can be roughly described as: the solutions with coordinate values within the range of $[-1, 1]$ are mostly evenly distributed over the 3D Lamé level set and some other solutions are in the acute points. The ratio of whether a solution falls in the central area or acute point depends on the quality indicator, and SPI^+ for example promotes more evenly distribution than GI_N^+. The populations of ELSA-$SR_{\omega=0.1}$ with GI_Π^+ and SPI^+ stand out slightly more than the other algorithms by having several more solutions in the acute points that are further away from the centre.

The results in 3D Ellipsoid, Hollow Sphere, and Double Sphere show the same patterns and they are treated as one category. In these three level set benchmarks the populations from different quality indicators resemble each other. What distinguish them are the tendency of solutions to lie on the boundary, which can be determined through visual inspection of the orthographic view. In all of these three benchmarks, both ELSA-SR configurations with GI_Π^+ and SPI^+ favour the boundary of the level set. ELSA-$SR_{\omega=0.1}$ expresses this tendency the most, while MCS and ELSA do not have it.

3D Branke's Multipeak is a level set benchmark where ELSA-$SR_{\omega=0.1}$ performs the worst in Incomplete Coverage: its population with GI_N^+ misses an entire level set component, and its GI_Π^+ population is much sparser than the other algorithms with most solutions clustered at the sphere shape, however its population with SPI^+ is not significantly different compared to the other algorithms. In contrary, the populations from MCS show a good population distribution that is comparable with the results from ELSA and ELSA-$SR_{\omega=1}$.

There are no noticeable differences between the results from the different quality indicators or between the different algorithm configurations for 3D Rastrigin.

There is no definitive algorithm that performs the best for 3D Schaffer in combination with all of the quality indicators, which is perhaps related to the observation that it is difficult to identify the differences between the quality indicators. The structure of 3D Schaffer can be described as a complex ring-like structure at the centre of the search space with separate bundles of rings positioned at different points on the x_1 and x_3 axes. In general all the populations cover the different ring bundles quite well barring the ones that are further away from the centre. If the ability to cover as many ring bundles, especially those at the extrema of the x_1 and x_3 axes, is considered desired for appealing distribution, then the algorithm that has the best coverage for each quality indicator are as follows: MCS for GI_N^+, a tie between ELSA and both ELSA-SR for GI_Π^+, and ELSA for SPI^+.

While the populations for 3D Vincent are not identical among the different quality indicators or algorithms, it is difficult to describe patterns in how they differ from each other or rank the populations of which looks better distributed than another. Overall MCS with any of the tested quality indicators seems to cover more level set components than the other algorithms, especially since it has more solutions on the smaller components (identified by dark diamond markers in 3D view). When viewing the orthographic depictions, all of the algorithms have cases of missing an entire layer of level set components in the x_3 axis except for the MCS populations. Therefore it can be said that MCS performs the best on Incomplete Coverage criteria for 3D Vincent.

Incomplete Coverage:

Incomplete Coverage for MCS, ELSA, and ELSA-SR for various multi-segmented level set benchmarks is described in Table 33. Level set benchmarks where no cases of Incomplete Coverage have been found are not depicted in the table like 2D Branke's Multipeak and Double Sphere of lower dimensions. Also, some level set benchmarks contain that many level set components that a population of size $\mu = 100$ can never cover them all, and including them in the table would not reveal any meaningful information. The Incomplete Coverage percentages that are described in the following paragraphs are taken from the algorithm runs with the relevant quality indicators GI_N^+, GI_Π^+, and SPI^+. Table 33 also assumes that populations need to completely consist of feasible solutions, so results from higher dimensions are ruled out for MCS.

All the algorithms are satisfactory at covering all level set components on the 2D level set benchmarks. For 2D Rastrigin, only a few cases of Incomplete Coverage are found in the MCS, ELSA, and ELSA-$SR_{\omega=1}$ runs which measures up to respectively approximately 2.5%, 4.2%, and 2.5%, where only one level component is missing, while ELSA-$SR_{\omega=0.1}$ has an Incomplete Coverage rate of 25.0% for 2D Rastrigin, which is heavily skewed by the results from GI_Π^+. For 2D Vincent, MCS has not found any Incomplete Coverage cases for the relevant quality indicators. The percentages on 2D Vincent from the remaining algorithms are as follows: 5.0% for ELSA, 6.7% for ELSA-$SR_{\omega=0.1}$, and 2.5% for ELSA-$SR_{\omega=1}$. All their cases miss only one level set component.

3D level set benchmarks reveal that the algorithms begin to encounter serious difficulties with Incomplete Coverage. Despite it merely contains 8 level set components, 3D Branke's Multipeak has cases where level set components are not covered, unlike its 2D equivalent. The Incomplete Coverage percentages for this benchmark are 11.7% for MCS, 25.0% for ELSA, 30.8% for ELSA-$SR_{\omega=0.1}$, and 23.3% for ELSA-$SR_{\omega=1}$. None of the algorithm configurations can cover all level set components of 3D Vincent, and the average amount of missing components are 12.2 for MCS, 17.0 for ELSA, 19.6 for ELSA-SR and 15.7 for ELSA-SR.

Double Sphere seems to be an easy level set benchmark with only two components, but it surprisingly poses difficulties for dimensions starting from 10. Few of the algorithms are consistent at locating both spheres, where approximately 71.7% of the ELSA runs does not cover both spheres for 10D Double Sphere and 70.8% of the runs ELSA-$SR_{\omega=0.1}$. The percentage for ELSA-$SR_{\omega=1}$ is however significantly lower with 6.7%. All of the runs from ELSA and ELSA-$SR_\omega = 0.1$ are unable to cover both spheres in 30D, and finally ELSA-$SR_{\omega=1}$ has a very high percentage as well with 95.8%.

The common pattern is that MCS performs most robust for the Incomplete Coverage criteria as

it is most frequent at covering all level set components compared to the other algorithms, and in the occurrences of Incomplete Coverage, it misses the least level set parts on average. This shows that MCS still has merit to be used for level set benchmarks of lower dimensions, especially if they contain multiple level set parts. ELSA-SR$_{\omega=0.1}$ performs the worst for Incomplete Coverage and it should be taken into account that this algorithm might trade a high diversity for Incomplete Coverage. ELSA sits between the two aforementioned algorithm configurations in terms of Incomplete Coverage. A simple fix for ELSA-SR is using a larger mutation step-size, as ELSA-SR$_{\omega=1}$ often has Incomplete Coverage that is closest to MCS and it is the only algorithm configuration in the test that maintains decent results even for 10D Double Sphere. As seen before in Section 4.2.1 results from ELSA with $\omega = 1$ has many similarities to the results of MCS, and ELSA-SR typically reduces the mutation step-size.

Summary of the conclusions found from the algorithm comparison:

Four algorithm configurations which include the Monte Carlo Search, ELSA, two versions of ELSA-SR with different initial step-sizes have been compared with each other on the 2D, 3D, 10D and 30D level set benchmarks. Three criteria have been used to rank the algorithms: Eval$_{Feasible}$, diversity and coverage.

Comparing the algorithm configurations with each other in terms of Eval$_{Feasible}$ over the solvable level set benchmarks of all tested dimensions, the order of requiring Eval$_{Feasible}$ in descending order is: MCS, ELSA-SR$_{\omega=1}$, ELSA-SR$_{\omega=0.1}$, and ELSA. Although ELSA-SR$_{\omega=0.1}$ often needs more evaluations on average than ELSA for finding the first population that completely consists of feasible solutions, the extra amount of evaluations can be considered insignificant in terms of the total evaluation budget used. The same cannot be said for ELSA-SR$_{\omega=1}$, where its Eval$_{Feasible}$ differences with the other algorithm configurations are large on 10D level set benchmarks.
Roughly all of the algorithms can find feasible populations for the 2D and 3D benchmarks (the only exception is MCS with 3D Lamé). MCS cannot find whole, feasible populations for the 10D benchmarks and higher dimensions, while the other algorithms manage to solve them with varying successes. Ellipsoid, Hollow Sphere, Double Sphere and Vincent are the sole level set benchmarks where whole, feasible populations have been found for by ELSA and ELSA-SR regardless of dimension. ELSA-SR can be considered an improvement on ELSA when either these algorithm configurations have managed to find whole, feasible populations for 10D Lamé, Rastrigin and Schaffer level set benchmarks as well as on those problems where ELSA failed.

In terms of diversity, ranking the algorithm configurations from best to worst goes approximately: ELSA-SR, ELSA, and MCS. Whether ELSA-SR$_{\omega=0.1}$ or ELSA-SR$_{\omega=1}$ yields a population of a higher diversity depends on the level set benchmark.

The Even Distribution and Incomplete Coverage sub-categories of coverage were primarily evaluated for the comparison of algorithms. The Boundary Concentration and Clustering Interior traits are also looked into, however algorithm comparisons do not clearly reveal what influences the algorithms have on these coverage traits to make generalisations about them.
The coverage observed on the 2D level set benchmarks is good for all tested algorithms. The results on the 2D benchmarks indicate that algorithms that promote a high diversity also yields a population that scores high on Even Distribution. Therefore the order of the algorithms that yield the best Even Distribution in 2D benchmarks in descending order is: ELSA-SR$_{\omega=0.1}$, ELSA / ELSA-SR$_{\omega=1}$, and MCS. In particular, the 2D populations from ELSA-SR$_{\omega=0.1}$ with SPI^{+} have an extremely evenly distribution.

The results on 3D benchmarks reflect that the tested algorithms that promote a high diversity have the drawback that its Incomplete Coverage trait is more prominent than in the other algorithm configurations. The opposite holds true where MCS has the lowest occurrences of Incomplete Coverage. The ranking of algorithm settings in Incomplete Coverage over the entire benchmark suite from least frequently occurring to most is: MCS, ELSA-SR$_{\omega=1}$, ELSA, ELSA-SR$_{\omega=0.1}$.

There exist limitations that are present in all four of the algorithm configurations. One of them is distributing solutions in acute spaces and it might be the reason why 10D Lamé provides quite a challenge.This is dependent on the exploitation aspect of the algorithm. Examining 3D populations for Lamé in terms of coverage shows preliminary signs of struggling as the distribution is not evenly spread out over the level set.
Incomplete Coverage is another bottleneck. When Incomplete Coverage is not defined as an aim in any of the quality indicators, it should not be fully expected that an algorithm always meet this goal. Such as 3D and higher dimensions Rastrigin, Schaffer, and Vincent have too many level set components for them to be covered or Schaffer has such an irregular shape that makes it near impossible to count whether a level set component is covered. However 3D Branke's Multipeak only has 8 components and Double Sphere only has 2 components, yet the algorithms fail to robustly cover all components of Branke's Multipeak or fail to find both spheres in 10D Double Sphere. This is dependent on the exploration aspect of the algorithm.
Both the limitations indicate that there is still room for improvement on the exploitation and exploration capabilities of ELSA and its variants.

In general to tackle the black-box level set approximation problem, the recommended algorithm configuration is ELSA-SR$_{\omega=1}$. It does not necessarily perform the best in all criteria, however it has less occurrences of Incomplete Coverage in populations than ELSA-SR$_{\omega=0.1}$ and the diversity of populations is almost equally high.

4.5 Commentary on the level set benchmarks

Since the level set benchmarks have not been used before prior to this thesis, it is necessary to reflect back on them and decide whether future improvements are required. Especially if there is a need for black-box level set benchmarks to compare different diversity optimisation algorithms. The commentary on the level set benchmark are based on the findings in Section 4.4.2 which compared the algorithm configurations.

Solvable level set benchmarks:

There are level set benchmarks that are not challenged with respect to finding feasible populations for even high dimensions, which are Ellipsoid, Hollow Sphere, Double Sphere and Vincent.

Ellipsoid and Hollow Sphere:

Ellipsoid and Hollow Sphere are both non-disjoint level set benchmarks where well-diversified populations can be relatively easily found by ELSA and ELSA-SR. Hollow Sphere has constant parameters over all dimensions, which is therefore good representation for the changes of a level set benchmark over all dimensions. It can be argued that the Ellipsoid level set benchmarks might be considered too easy to solve with a relatively large surface or volume level set area, and it might warrant picking smaller semi-principal axes values for extra challenge. On the other hand, while the used 2.5 is fairly large for a semi-principal axis, the ellipsoid shape is less resem-

bling to a sphere when picking a set of semi-principal axes that contain both large and small values. However, neither Ellipsoid nor Hollow Sphere should be seen as trivial either because MCS are not capable to find whole, feasible populations for any of the 10D (and higher) level set benchmarks. So Ellipsoid and Hollow Sphere can be representatives of the bare minimum of what a diversity optimisation algorithm should be able to find whole, feasible populations for.

Double Sphere and Vincent:

Double Sphere and Vincent are disjoint level set benchmarks where the Double Sphere benchmarks only have two level set components while for Vincent the amount of level set components grow exponentially over the dimensions. Double Sphere seems relatively simple, however many of the tested algorithms fail to disperse the population equally over the two spheres in 10D Double Sphere benchmark. Therefore the current version of Double Sphere can serve as a suitable reference to measure an algorithm's ability to achieve an even spread over level set components. Vincent benchmark is a specialised benchmark to test an algorithm's ability of dispersing its population over the highly disjoint available level set space.

Branke's Multipeak:

The Branke's Multipeak level set benchmark has many interesting features and it would be preferable if it can be used as a higher dimension level set benchmark. For future work, the function definition should be rewritten to achieve a level set benchmark that maintains its separability without being as much affected by its level set ϵ threshold value.

Challenging level set benchmarks:

Then there remains the level set benchmarks which are either too challenging to be solved by any of the tested algorithms in all dimensions. This is most apparent for high dimensional Schaffer and to some lesser degrees Lamé and Rastrigin. When one encounters results that are below expectations, it begs the question whether the level set benchmark itself is designed to be too challenging, such as the level set consists of a small feasible volume amidst a large search space, or the problem simply lies in the algorithms that have their limitations. Therefore Lamé, Schaffer and Rastrigin are examined again in more depth.
One of the quickest ways to modify a level set benchmark for lower difficulty is to increment the ϵ threshold values for the level set benchmarks. However as seen in Branke's Multipeak it can lead to undesirable effects such as losing the disjointness of the level set components.

Lamé shapes:

The shape of the Lamé level set benchmark is controlled by its function parameters, and as the maximum length for the semi-diameter r that fits within the search space is already chosen, the only recommended parameter that can be changed is the curvature p. Few indication is found that Lamé sets up impossible standards, and it is theorised that it is linked to the shrinking volume of unit sphere in high dimensions phenomenon [17], combined to the fact that the Lamé shapes are extremely concave compared to the Ellipsoid that is convex. As there are already signs of a preliminary struggle of the algorithms for 3D Lamé, it seems to imply more that the ELSA and its variants are not adept in solving level set benchmarks with acute parts or where the volume of the level set is very small compared to the search space.

Rastrigin:

The Rastrigin level set benchmark has the characteristic that it has a rather large level set volume which has its components spread out over the entire search space. This is the case in the 2D and 3D level set benchmark. Motivated by the fact that the average diversities of the populations in 10D Rastrigin are lower than the average diversities of the populations in 3D Rastrigin, the GI_{Π}^{+} populations from ELSA-$\text{SR}_{\omega=0.1}$ for 10D Rastrigin are examined in more detail as a case study. It is observed that the GI_{Π}^{+} populations from ELSA-$\text{SR}_{\omega=0.1}$ for 10D Rastrigin have $\frac{1}{|P'|}\sum_{P\in P'}(\min_{i=1\ldots n,\mathbf{x}\in P}(x_i)) = -2.16$ and $\frac{1}{|P'|}\sum_{P\in P'}(\max_{i=1\ldots n,\mathbf{x}\in P}(x_i)) = 2.25$, where P' is the set of all GI_{Π}^{+} populations from ELSA-$\text{SR}_{\omega=0.1}$ for 10D Rastrigin. Also, the smallest and largest variable value found in X are not too far off from the averages. If the landscape of 10D Rastrigin remains identical to the 2D and 3D variants, it means that most of the solutions are located on the central waffle shape and rarely on the disjoint parts. The Rastrigin benchmark landscape can be elaborated as such that it consists of peaks and the peaks that are further away from the centre have higher function values. Its partial function $y(x_i) = \left(x_i^2 - 10\cdot\cos(2\pi x_i)\right)$ is depicted in Figure 11, where the entire Rastrigin function is $10n + \sum_{i=1}^{n} y(x_i)$. As visualised in Figure 11, to include extrema variable values in a solution in higher dimension Rastrigin benchmarks requires the compensation from variable values that are small to balance the sum to fit within the allowed ϵ threshold value. Therefore theoretically, both 10D and 30D Rastrigin still contain the outer rim with disjoint level set components when for example a solution that consists of a zero vector with one variable value as -4 or 4 is feasible. Going by this assumption, a well-diversified population is achievable if an algorithm is capable of finding them. While increasing the ϵ threshold value most likely guarantee a benchmark that can be solved easier, some disjointedness in the Rastrigin level set benchmark might get lost such that Rastrigin loses its identity as a level set benchmark with both a mix of interconnected hole-filled central structure and disconnected outer structure.

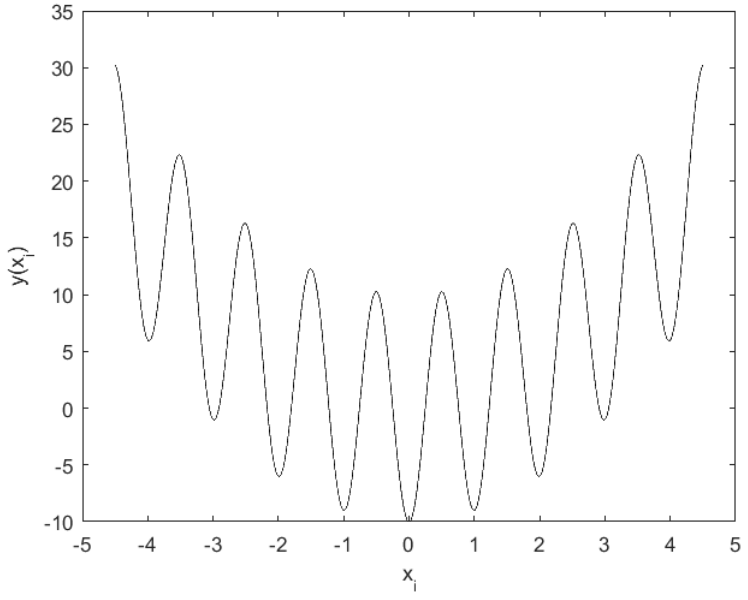

Figure 11: Rastrigin partial function $y(x_i)$ plot

Schaffer:

10D Schaffer (and higher dimensions) might not be a suitable level set benchmark for the purpose of level set approximation with the current level set parameter settings or function definition. Some populations from ELSA-SR configurations for 10D Schaffer might be wholly feasible but their solutions barely have any variation from each other. Reasoning about the Schaffer formula, the first element x_1 and the last element x_n of solution vector $\mathbf{x}$ have less weight than the middle elements in the summation, therefore the middle elements are expected to be close to 0 to not exceed the threshold while x_1 and x_n are allowed to have more extreme values. In 3D Schaffer it can be seen that the level set is located around 0 for the middle vector axis x_2 and the search space is prominently empty at extrema for x_2. Schaffer has comparable traits with Rastrigin (high degree of disjointness) and Lamé (resemblance in shape with the chosen benchmarks parameters in 3D) but is significantly more complex than the other two level set benchmarks. The Schaffer benchmark seems to be in need of a larger ϵ threshold value for each increase of dimension. For example 3D Schaffer with a higher ϵ has larger ring structures that bear resemblance to the overall appearance of the 2D Schaffer benchmark and fills out the search space more evenly as depicted in Figure 12. The decision for the eventually chosen ϵ was made without the foresight that it poses a lot of trouble in higher dimensions, but more study is required to decide what threshold and search space size is suitable to create benchmarks that are both challenging and solvable.

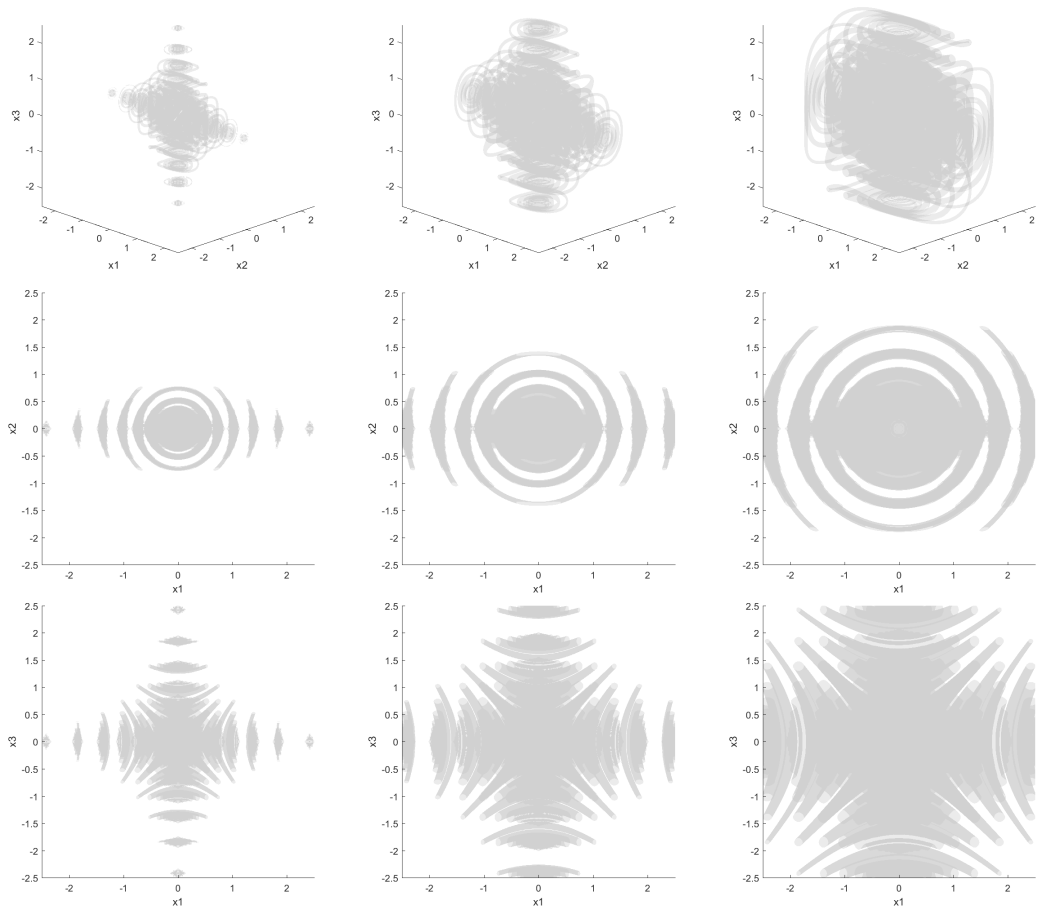

Figure 12: The figure depicts different projections of the 3D Schaffer level set benchmark with a different ϵ threshold value. Each column represents a level set benchmark, where the first corresponds to $\epsilon = 2$ (currently used), the middle corresponds to $\epsilon = 2.5$, and the last corresponds to $\epsilon = 3$.

5 Conclusion and future work

This thesis focuses on the Black Box Level Set Approximation Problem which aim is to find an approximation of the set of inputs of a function that give rise to a targeted output below a threshold ϵ. Problems of level set approximation occur in various disciplines of science and engineering. As maximum diversity is desired for the approximation of the level set, algorithms for diversity optimisation have been designed. They include an Evolutionary Algorithm with niching scheme such as EAGA and Indicator-Based Evolutionary Algorithms such as NOAH and ELSA.

The ELSA algorithm has only previously been applied on a limited set of 2D level set benchmarks in its original papers by Emmerich et al. Additionally ELSA has been adapted to various practical problems where set of inputs of higher dimensions are approximated. The motivation of this thesis is to determine the strengths and limitations of ELSA, especially on higher dimensions which is more relevant to realistic practical problems, through applying an extensive empirical study on ELSA. Furthermore, findings are documented to acquire knowledge on the level set approximation problem to discover what is demanded from algorithms (not necessarily limited to ELSA) to solve this problem.

A summmary is provided about the successfulness of whether the contributions of this thesis met their objectives.

Due to the large bulk of calculations required for the experiments it is expected that the ELSA algorithm can run fast. The implementation of the ELSA algorithm code in MATLAB has been optimised to reduce time complexity, which is described in Section 2.4. Mainly the quality indicator contribution is a costly operation and incremental update is applied to minimise recalculation. Of the quality indicators that are used by ELSA: the three Augmented Gap Indicators (GI^+) gained the largest time speed-up, the Augmented Solow Polasky Indicator (SPI^+) a decent improvement, while the Augmented Average Indicator (ADI^+) has barely been improved.

The proposed black box level set benchmark suite consists of four benchmarks made up from simple geometric shapes (Lamé, Ellipsoid, Hollow Sphere, and Double Sphere) and four benchmarks made up from complex engineering relevant functions (Branke's Multipeak, Rastrigin, Schaffer, and Vincent) for the dimensions 2D, 3D, 10D, and 30D. The guidelines for the design of a level set benchmark are described in Section 3.1. The main guideline is that the level set should strike a balance between difficulty and solvability, where the population on the level set should be able to be visualised (visualisation only applies to the 2D and 3D benchmarks). The benchmarks should take the chosen objectives for measurement and ranking of the algorithms (presented in Section 4.1) into account. They are (1) $\text{Eval}_{Feasible}$: a measurement for the amount of evaluations required to find a whole, feasible population, (2) diversity of population according a quality indicator, and (3) coverage of population, in which coverage is further split up into the following sub-criteria: Boundary Concentration, Clustering Interior, Even Distribution, and Incomplete Coverage.

As previous literature on ELSA did not always recommended algorithm parameter values, we attempt to find a suitable configuration for ELSA by analysing different values for σ step-size and ν mutation probability on selected benchmarks. The step-size is normalised by a parameter called ω (a smaller ω value corresponds to a smaller σ step-size). The 3D Ellipsoid and 3D Vincent benchmarks are chosen to measure $\text{Eval}_{Feasible}$ and diversity. These are a non-disjoint level

set benchmark and a multi-segmented level set benchmark respectively. Additionally 2D Vincent is used to measure Incomplete Coverage.
A high ν value or a low ω value corresponds to less evaluations required to find a whole, feasible population, however they also increase the likelihood to missing level set components in multi-segmented level sets. A configuration that performs optimally on all of the three aspects concerning $\text{Eval}_{Feasible}$, diversity and Incomplete Coverage has not been found. Nevertheless $\omega = 0.1$ as σ step-size and $\nu = 0.5$ are chosen to be the default configuration for the experiments in Section 4.4 on the entire benchmark suite. A mixed mutation strategy with $\nu = 0.5$ emphasises on equal probability for random initialisation (exploration) and parent mutation (exploitation) to occur, which is shown to be the most balanced choice for black box level set benchmarks.

The σ step-size algorithm parameter has influence on the results, and the need to identify and manually pick a suitable step-size is removed when this parameter is self-adaptative. Not only that, depending on the stage within the optimisation process a different step-size might be demanded. Such as the experiments on finding suitable algorithm parameter values for ELSA reveal that a small σ step-size normalised with $\omega = 0.01$ might lead to higher diversity but also higher Incomplete Coverage, while a larger σ (such as with the recommended $\omega = 0.1$) correspond to lower diversity and lower Incomplete Coverage.
We propose a modification on ELSA called ELSA-SR that utilises a self-adaptative σ step-size that changes its value for σ according to the one-fifth success rule scheme. The algorithm parameters of ELSA-SR were analysed to recommend a configuration for this algorithm: $\alpha = 0.95$ for the learning rate, the value for the measurement range of β is set accordingly for 200 changes at maximum for the σ parameter ($\beta = 50$ for an evaluation budget of 10K and $\beta = 500$ for an evaluation budget of 100K), and a target success probability of $\gamma = 0.2$. The algorithm parameters that already exist in ELSA are kept the same for ELSA-SR, which includes $\nu = 0.5$, while two initial ω normalised step-sizes are tested for the entire benchmark suite: $\omega = 1$ and $\omega = 0.1$.
Some of the conclusions derived from the analysis on algorithm parameters for ELSA-SR are highlighted. The one-fifth ratio as a success rule was tailored to the specific problems in Rechenburg's research, however coincidentally this target success rate of 0.2 works very well for ELSA-SR on the level set approximation problem.
The chosen β value for measurement range influences whether the target success rate γ is maintained by the algorithm. Plotting the success rate for all configurations of different β shows a common pattern that it always starts high (objective function phase) far above the selected γ and then rapidly decreases (diversity optimisation phase). For a high β value, the target success rate is not met again after the initial phase, whereas using a smaller β leads to success rate plots that successfully fluctuate around the target success rate. In the latter case, it leads to the effect that the σ step-size is inhibited from decreasing too fast as long as a satisfactory amount of successful children are produced for that step-size.
The adaptability of ELSA-SR has been tested on two aspects: in terms of whether ELSA-SR can fix a poorly chosen initial σ step-size and whether ELSA-SR can be used as a full-mutation strategy. ELSA-SR is capable at stabilising the σ step-size, as the results with $\beta = 5$ show that the final diversities of populations generated from different initial step-sizes are in close approximation with each other and diversity differences between configurations are smaller compared to the similar comparison of different step-sizes with ELSA. However ELSA-SR is incapable to be used as a full mutation strategy, as their populations have the worst diversity among the configurations with different ν values, and mixed mutation strategy produce superior results.

The four algorithm configurations that are tested on the entire level set benchmark suite consist

of the Monte Carlo Search (MCS) as a reference, ELSA, ELSA-SR$_{\omega=0.1}$ and ELSA-SR$_{\omega=0.1}$. The experiments on the entire benchmark suite provides information on quality indicator selection, algorithm selection, and reflection on the level set benchmarks.

The quality indicators have been compared in terms of diversity and coverage to acquire knowledge on their characteristics. Quality indicators of interest are Augmented Minimal Gap Indicator (GI_N^+), Augmented Geometrical Mean Gap Indicator (GI_Π^+), Augmented Solow Polasky Indicator (SPI^+) due to that populations generated from them have different coverage properties. GI_Σ^+ and ADI^+ can be treated as obsolete as they have unfavourable coverage properties and alternative quality indicators that can generate a population of similarly measured diversity and more preferable coverage properties can replace them. GI_Π^+ can serve as a replacement for GI_Σ^+, while SPI^+ serves as a replacement for ADI^+. An interesting observation is that the population generated by SPI^+ which diversity is measured by a different quality indicator can outperform a population from that native quality indicator in terms of diversity. A noticeable drawback concerning GI_N^+ is that its populations do not benefit from the self-adaptative σ stepsize in ELSA-SR.

ELSA-SR has been designed to cover possible shortcomings of ELSA and results prove that it is quite successful. While the comparisons have been measured over $\text{Eval}_{Feasible}$, diversity, and coverage, here we mostly focus on $\text{Eval}_{Feasible}$ as the most crucial differences are found for that objective (for the complete summary, refer to Section 4.4). The 2D and 3D level set benchmarks are considered fairly unchallenging to be solved as mostly all algorithms consistently find whole, feasible populations for them (except for 3D Lamé with MCS). The 10D level set benchmarks is where limitations of the algorithm configurations begin to manifest. MCS cannot find whole, feasible populations for the 10D benchmarks and higher dimensions, while the other algorithms manage to solve them with varying successes. Ellipsoid, Hollow Sphere, Double Sphere and Vincent are the sole level set benchmarks where whole, feasible populations have been found by ELSA and ELSA-SR regardless of dimension. One ELSA-SR algorithm configuration has managed to find whole, feasible populations for 10D Lamé, Rastrigin and Schaffer level set benchmarks which are problems where ELSA failed.
In general ELSA-SR also generates populations of higher diversity than ELSA. The results on the black box level set benchmark suite do not point to any important advantage of using ELSA over ELSA-SR. Either ELSA-SR configuration can perform comparably well or better than ELSA on $\text{Eval}_{Feasible}$, diversity, or coverage.

The difficulty of solving a level set benchmark is determined from the performances of the algorithms. The level set benchmarks that are considered solvable and unchallenging to find feasible solutions for are Ellipsoid, Hollow Sphere, Double Sphere, and Vincent. Ellipsoid and Hollow Sphere are non-disjoint level set benchmarks that can be representatives of the bare minimum of what a diversity optimisation algorithm should be able to find whole, feasible populations for. Double Sphere and Vincent are disjoint level set benchmarks, and while the algorithms manage to find whole, feasible populations for them, it is worthy of mention that many algorithm configurations fail to find both sphere shapes in 10D Double Sphere.
The level set benchmarks that are considered challenging are Lamé, Rastrigin, and Schaffer. The common attribute of these benchmarks is that they either contain acute spaces or extremely small level set components. It is unclear whether 10D Lamé, Rastrigin, and Schaffer are designed too difficult or that the algorithms themselves have limitations that prevent them from solving these benchmarks successfully. While technically an ELSA-SR configuration could find whole, feasible populations for those 10D benchmarks, their diversities are rather poor.
Branke's Multipeak has not been defined for 10D and 30D and its difficulty is therefore not

classified.
Future research might look into rewriting the Branke's Multipeak formulation to yield a benchmark with similar properties but keep the prominent shapes as separate entities regardless of dimension. Secondly more advanced algorithms should be tested on the entire benchmark suite to compare whether diversities and Incomplete Coverage cases can be improved with regards to the results found in this thesis.

The general shortcomings for ELSA and ELSA-SR are conjectured to lie within the methods in which exploration and exploitation are performed. When comparing the development of ELSA and its variants with the history of Evolutionary Strategies (ES) - which the ELSA algorithm applied for the approximation problem share many parallels with - the amount of revisions on the ES is far richer while ELSA is still in its early stages. For example there exist highly specialised ES algorithms such as CMA-ES that adapt the mutation step-sizes in a far more complex manner than what is applied in ELSA. ELSA and its variants uses the mixed mutation strategy to take both exploitation and exploration in consideration, however it does not utilise all available techniques that are found in ES or those that are considered state-of-the-art.

Lamé is a prime example of a benchmark to test the ability of exploitation. For 3D, it is observed that the algorithms struggle with finding populations that distribute the solutions evenly over the entire level set of this benchmark. In particular, the acute spaces are the locations where solutions typically do not reside in.
Incomplete coverage might not be completely avoided when the amount of level set components frequently exceeds the population size. Level set benchmarks with few components (for example Branke's Multipeak and Double Sphere) are expected however to have all its components covered. While minimising Incomplete Coverage is extremely difficult when none of the diversity indicators include this criterion in their formulation, it is speculated that applying exploration might lower the occurrences of Incomplete Coverage. This is based on the results that MCS has the lowest tested cases of Incomplete Coverage, and MCS can be considered an algorithm that fully focuses on the exploration aspect but lacks exploitation.

Ideas for potential improvements on the ELSA algorithm are for example designing new selection and recombination operations for ELSA (cross-over instead of recombination in case of discrete problem domain). ELSA does not utilise recombination at all. However, if applied correctly, recombination might help to improve the level set coverage by closing gaps. However, recombination might increase the production of infeasible children when applying it to parent solutions from different components of the level set. NOAH produces multiple children in a generation and a similar scheme can be applied to ELSA. Moreover, rather than choosing a random solution for mutation, mating selection is an option that can be implemented.
The first attempt of self-adaptation of the σ step-size for ELSA is based on the one-fifth success rule algorithm due to its simple implementation. Contemporary ES algorithms should be used to draw inspiration for adapting σ.

Another vision for future research is to conduct the same experiments on the black box level set benchmark suite with different diversity optimisation algorithms such as NOAH or EAGA. Artificial Immune Systems, in contrast to evolutionary algorithms, feature a variable population size and they also have some inherent mechanisms for diversity maintenance, which as well makes them a promising technique for level set approximation [3].

Finally, there are practical problems where the ELSA algorithm has been applied to or is applicable for. The ELSA-SR algorithm might be applied to these problems to test whether

it can improve on the original results. Moreover, it might be useful in other problem solving domains that involve questions of parameter identification, design option generations, or model-based diagnosis.

Acknowledgements

First I want to dedicate my gratitude to my supervisors: Michael Emmerich for all the constant guidance and support he provided me and all the things I have learned from him while I was working on this Master Thesis, and André Deutz for his helpful feedback in our lengthy discussions. Special credits go to Vitor Basto-Fernandes, Iryna Yevseyeva, and Joost Kok. I am grateful to be granted an opportunity to work with them on the IWINAC conference paper about the ELSA algorithm. Finally, I want to thank my family and friends for always cheering me on to finish the thesis.

References

[1] F. Aurenhammer. Evolution strategies - a comprehensive introduction. *ACM Computing Surveys (CSUR) (September 1991, Volume 23, Issue 3)*, pages 345–405, 1991.

[2] H.G. Beyer and H.P. Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing (March 2002, Volume 1)*, pages 3–52, 2002.

[3] G.P. Coelho and F.J. Von Zuben. omni-ainet: An immune-inspired approach for omni optimization. *In International Conference on Artificial Immune Systems*, pages 294–308, 2006.

[4] M.T.M. Emmerich, N. Beume, and B. Naujoks. An emo algorithm using the hypervolume measure as selection criterion. *International Conference on Evolutionary Multi-Criterion Optimization*, pages 62–76, 2005.

[5] M.T.M. Emmerich, A.H. Deutz, and J.W. Kruisselbrink. On quality indicators for black-box level set approximation. *EVOLVE - A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation Studies in Computational Intelligence*, pages 157–158, 2013.

[6] A. Jaklič, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics: Computational Imaging and Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[7] J.W. Kruisselbrink. Evolution strategies for robust optimization. *PhD Thesis, LIACS, Leiden University*, 2012.

[8] J.W. Kruisselbrink, M.T.M. Emmerich, and A.H. Deutz. Evolutionary level set approximation (matlab) [accessed 10.3.2018]. `http://liacs.leidenuniv.nl/~csnaco/index.php?page=code`, 2012.

[9] L. Liu, E.M. Zechman, E.D. Brill Jr., G. Mahinthakumar, S.R. Ranjithan, and J. Uber. Adaptive contamination source identification in water distribution systems using an evolutionary algorithm-based dynamic optimization procedure. *Water Distribution Systems Analysis Symposium 2006*, pages 1–9, 2008.

[10] L.Y. Liu, V. Basto-Fernandes, I. Yevseyeva, J. Kok, and M.T.M. Emmerich. Indicator-based evolutionary level set approximation: Mixed mutation strategy and extended analysis. *IWINAC 2017: Natural and Artificial Computation for Biomedicine and Neuroscience*, pages 146–159, 2017.

[11] L.Y. Liu, M.T.M. Emmerich, and A.H. Deutz. Diversity optimisation: A case study on the gielis formula. *Research Project, LIACS, Leiden*, 2018.

[12] A. Nezhinsky and M.T.M. Emmerich. Parameter identification of stochastic gene regulation models by indicator-based evolutionary level set approximation. *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, pages 50–64, 2018.

[13] H.P. Schwefel. Evolution and optimum seeking. *Wiley*, 1995.

[14] O.M. Shir. Niching in derandomized evolution strategies and its applications in quantum control. *PhD Thesis, LIACS, Leiden University*, 2008.

[15] T. Ulrich and L. Thiele. Maximizing population diversity in single-objective optimization. *GECCO '11 Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 641–648, 2011.

[16] B. van der Burgh, M.T.M. Emmerich, A.H. Deutz, and I. Yevseyeva. An evolutionary algorithm for finding diverse sets of molecules with user-defined properties. *Internal Report 2013-03, Universiteit Leiden, Opleiding Informatica*, 2013.

[17] E.W. Weisstein. "hypersphere." from mathworld – a wolfram web resource [accessed 18.4.2016]. `http://mathworld.wolfram.com/Hypersphere.html`, 2002.

[18] E.W. Weisstein. "superellipse." from mathworld – a wolfram web resource [accessed 18.4.2016]. `http://mathworld.wolfram.com/Superellipse.html`, 2003.

[19] M.L. Weitzman. On diversity. *The Quaterly Journal of Economics*, pages 363–405, 1992.

[20] E.M. Zechman and S.R. Ranjithan. Generating alternatives using evolutionary algorithms for water resources and environmental management problems. *Journal of water resources planning and management 133(2)*, pages 156–165, 2007.

Appendix

Manual for using ELSA and ELSA-SR:

The MATLAB package for ELSA and ELSA-SR can be downloaded from the `http://moda.liacs.nl` website. It is free software under the GNU General Public License. The most essential files in the package that are required for running the algorithms are listed.

`Algorithms`

- `ELSA.m`
- `ELSASR.m`
- `reflect_bc.m`

`QIC`

- `computeQIC_gap_aug.m`
- `computeQIC_SIGMA_aug.m`
- `computeQIC_PI_aug.m`
- `computeQIC_SPI_aug.m`
- `computeQIC_ADI_aug.m`

`QI`

- `QI_gap.m`
- `QI_gap_augmented.m`
- `QI_SIGMA.m`
- `QI_SIGMA_augmented.m`
- `QI_PI.m`
- `QI_PI_augmented.m`
- `QI_SPI.m`
- `QI_SPI_augmented.m`
- `QI_ADI.m`
- `QI_ADI_augmented.m`

`QI_functions`

- `dmat_RMV.m`
- `euclideanDist.m`
- `findDuplicates.m`

`ELSA` and `ELSASR` are the main algorithm functions and `reflect_bc` is the function that is used by the algorithms to handle a solution that is generated outside the search space.

The files for the black box benchmark level suite and visualisation functions for level sets are found in the `Benchmarks` and `Visualisation` directories respectively. For instructions on how to use these, refer to the `README.txt` file in the package.

The algorithm parameters in ELSA which require the user to initialise its values are:

- `objective_fct`: the function for a level set benchmark (format: columns represent the dimension)
- `threshold`: ϵ, the threshold for level set benchmark
- `N`: dimension of benchmark
- `lb`: lower bound of search space, a $N \times 1$ vector
- `ub`: upper bound of search space, a $N \times 1$ vector
- `stopeval`: evaluation budget
- `mu`: μ population size

Optional algorithms parameters are:

- `QIC_fct`: The quality indicator contribution function that is used, when not specified by user the algorithm will use GI_N^+ for QIC calculation.
- `bch_fct`: the function to handle out-of-bound solutions, by default it is `reflect_fct`.
- `sigma`: σ step-size, the user can pass a numeric value to σ or pass a string that corresponds to a ω normalised step-size from Formula 56, where `sigmaDefault` corresponds to $\omega = 1$, `sigmaSmaller10` corresponds to $\omega = 0.1$, `sigmaSmaller100` corresponds to $\omega = 0.01$, and `sigmaSmaller1000` corresponds to $\omega = 0.001$.
- `nu`: ν, the rate for parent-based mutation
- `T`: the interval of evaluations for which the quality indicators are measured over the current population.

The following four variables are returned by ELSA:

- `P`: final population, a $N \times \mu$ matrix
- `f`: objective function values of population, a $1 \times \mu$ vector
- `QI_plot`: a $5 \times \lfloor$ `stopeval`$/$`T` $\rfloor$ matrix that contains a plot of the QI measurements of the population over `T` intervals. Regardless which QIC function is used, all QIs are used to measure the population. A row corresponds to the following QI: (1) GI_N^+, (2) GI_Σ^+, (3) GI_Π^+, (4) SPI^+, and (5) ADI^+. Exact calculation of ADI^+ is not defined for dimensions other than 2D and the values are denoted with `NaN` instead.
- `evalFeasible`: $\mathrm{eval}_{Feasible}$

An example of a minimal function call to ELSA is as follows:

```
[P,f,QI_plot,evalFeasible] = ELSA(objective_fct, threshold, N, lb, ub, ...
stopeval, mu);
```

Optionally, the user can change the values of algorithm parameters in ELSA:

```
sigma = 'sigmaSmaller10';
QIC = @computeQIC_gap_aug;

[P,f,QI_plot,evalFeasible] = ELSA(objective_fct, threshold, N, lb, ub, ...
stopeval, mu, 'sigma', sigma, 'nu', 0.5, 'QIC_fct', QIC, 'T', 500);
```

Calling ELSA-SR goes in a similar way as ELSA, but it has more optional algorithm parameters:

- `alpha`: α learning rate, needs to be a numeric value in the $0 < \alpha < 1$ range.
- `beta`: β measurement range for which the success rate is measured
- `gamma`: target success rate, needs to be a numeric value in the $0 \leq \gamma \leq 1$ range.

ELSA-SR also returns one extra variable compared to ELSA (in the fifth slot) which is called `SR_plot`. This variable is a vector of unknown length which stores the success rates that have been measured over the intervals. Its length depends on the amount of times σ has been adapted. The sole purpose of this variable is for the analysis of the behaviour of ELSA-SR. The changes that σ undergoes can be derived from the `SR_plot` variable.

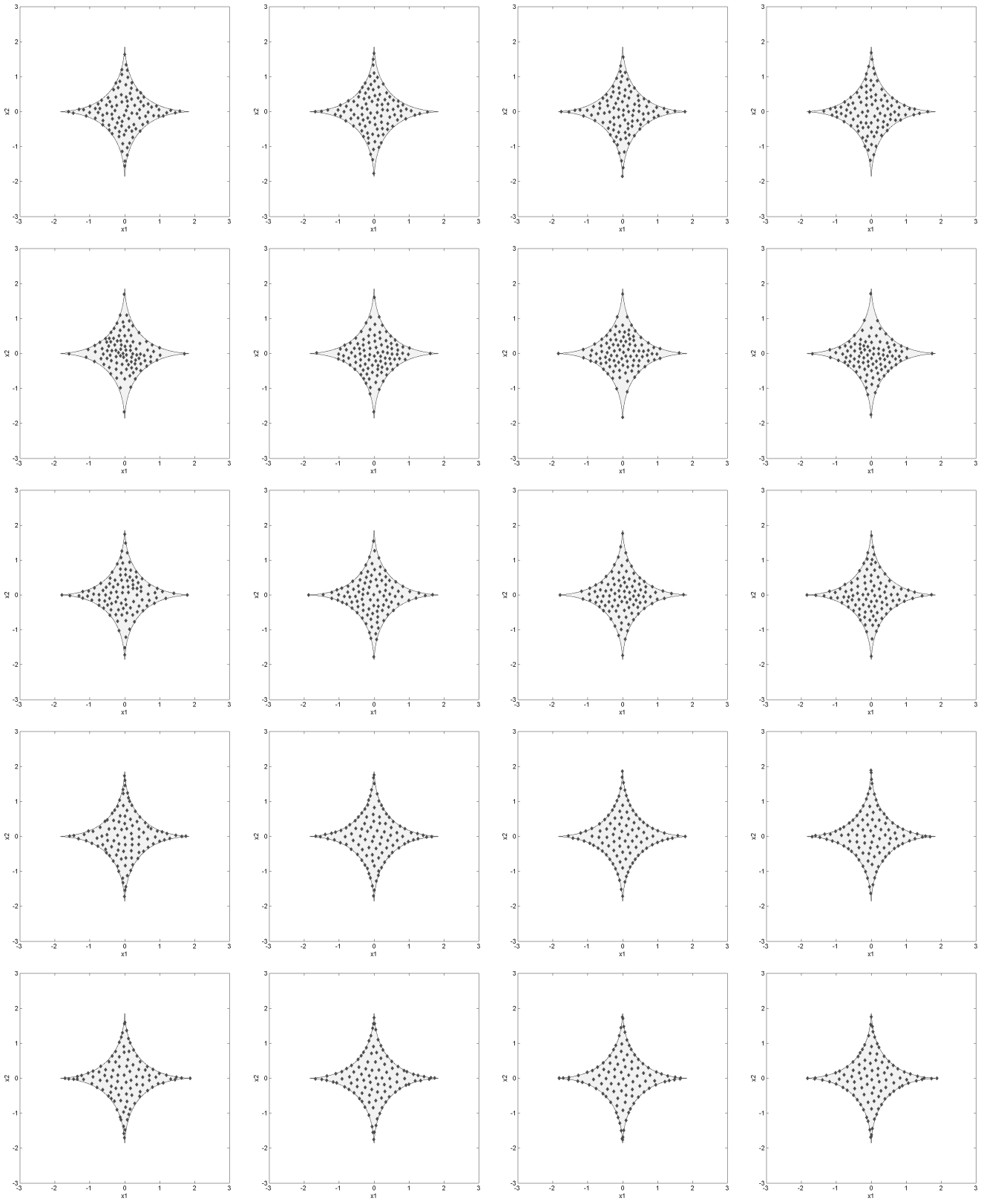

Figure 13: 2D Lamé level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$.

Figure 14: 2D Ellipsoid level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$.

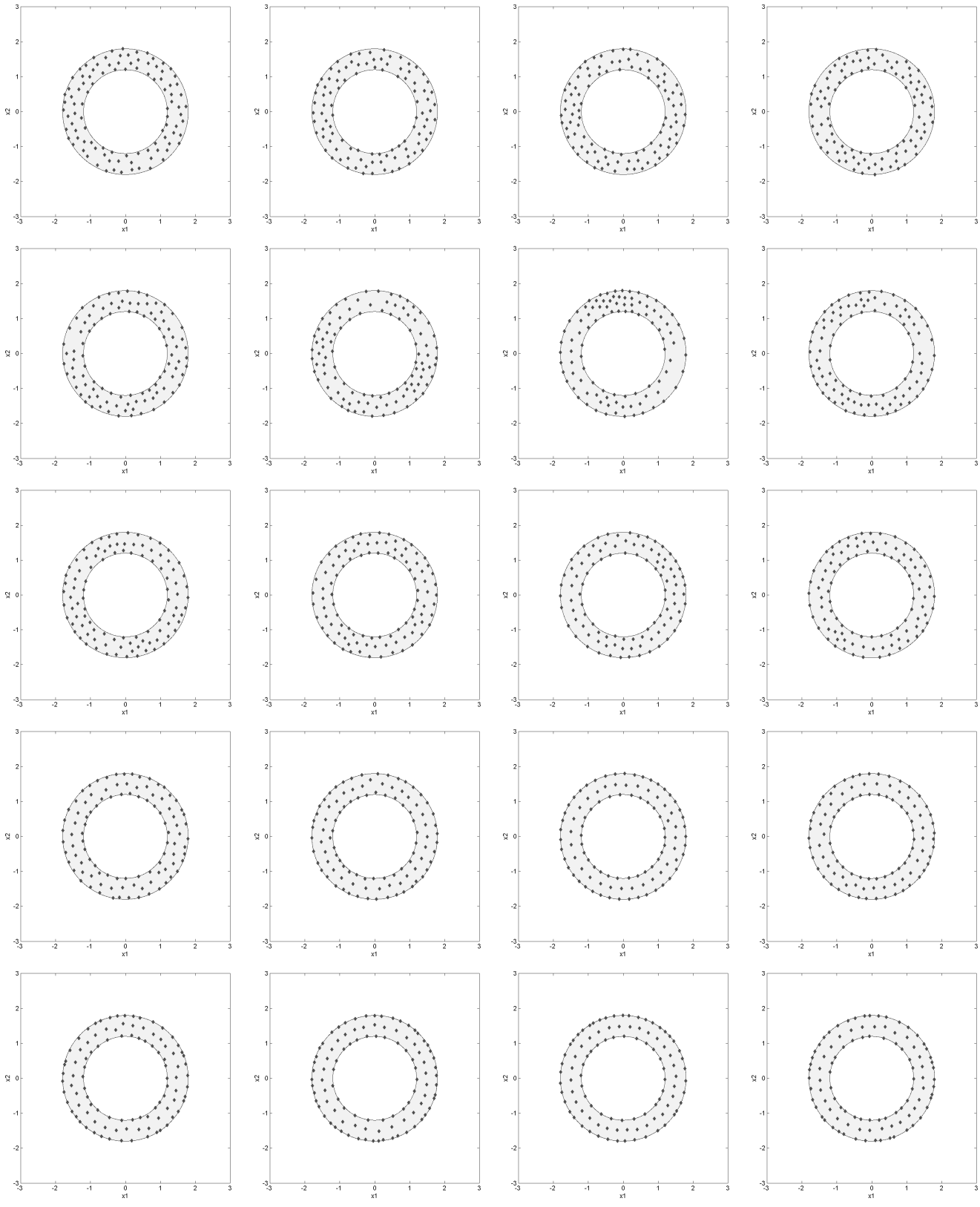

Figure 15: 2D Hollow Sphere level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$.

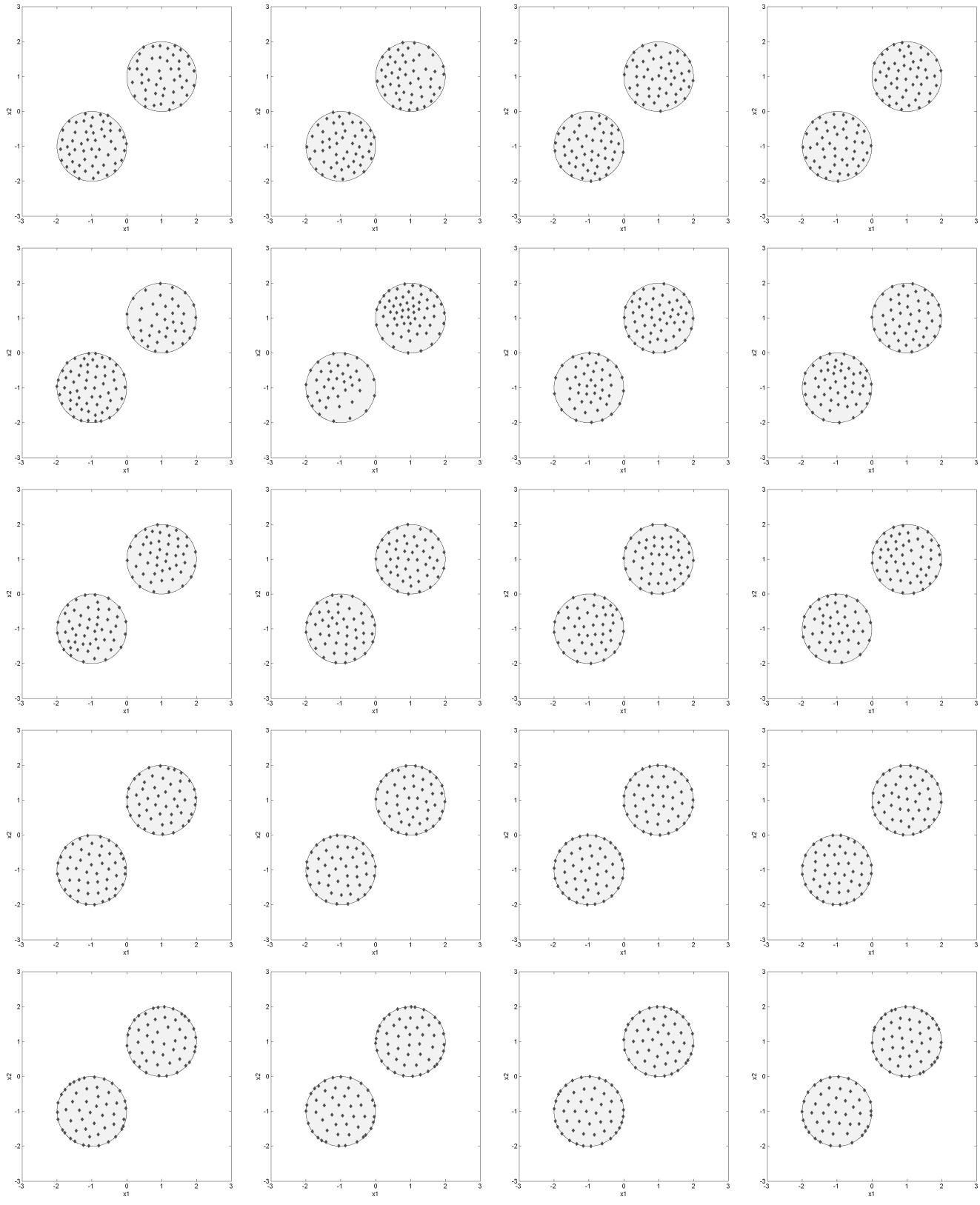

Figure 16: 2D Double Sphere level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$.

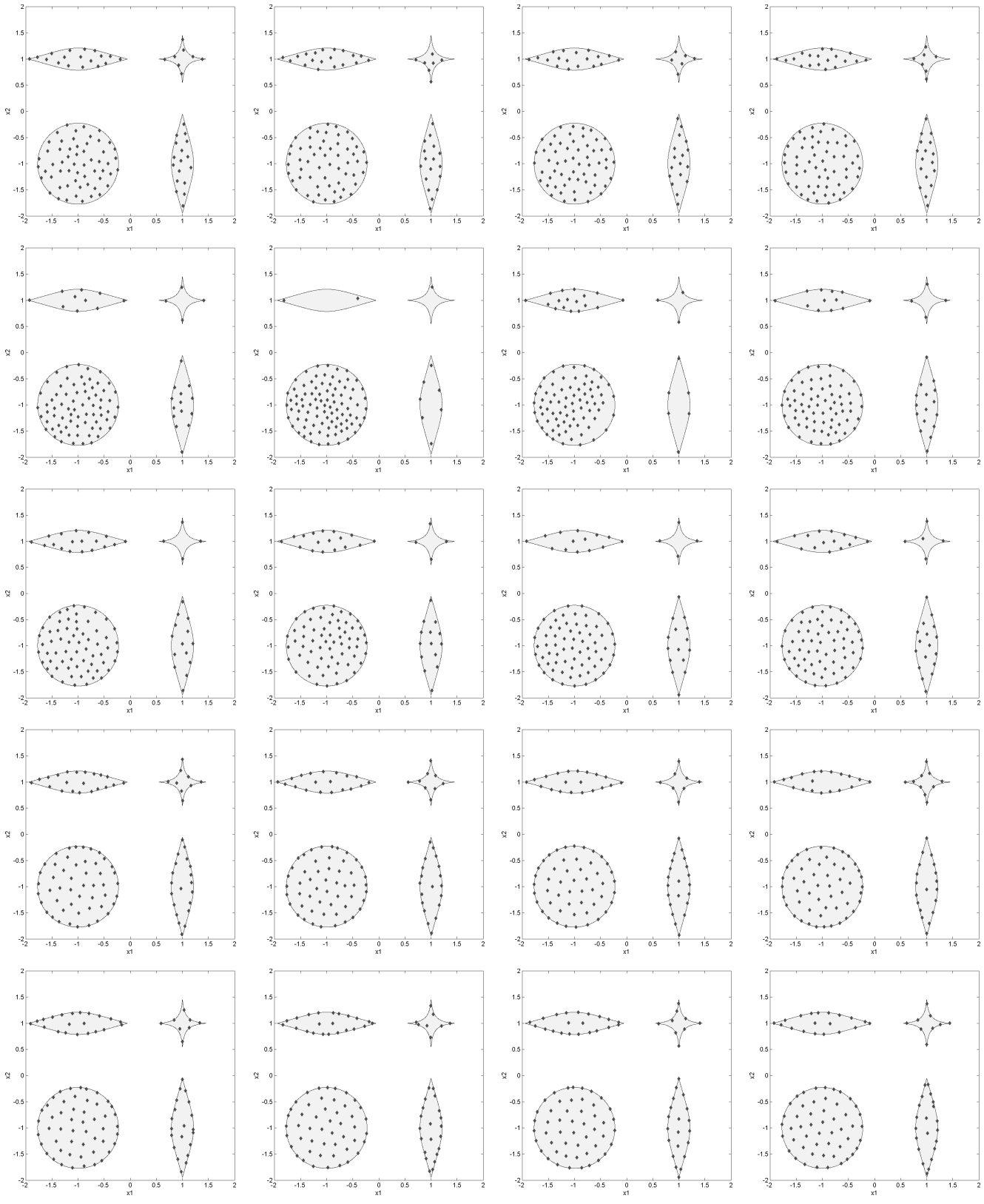

Figure 17: 2D Branke's Multipeak level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$.

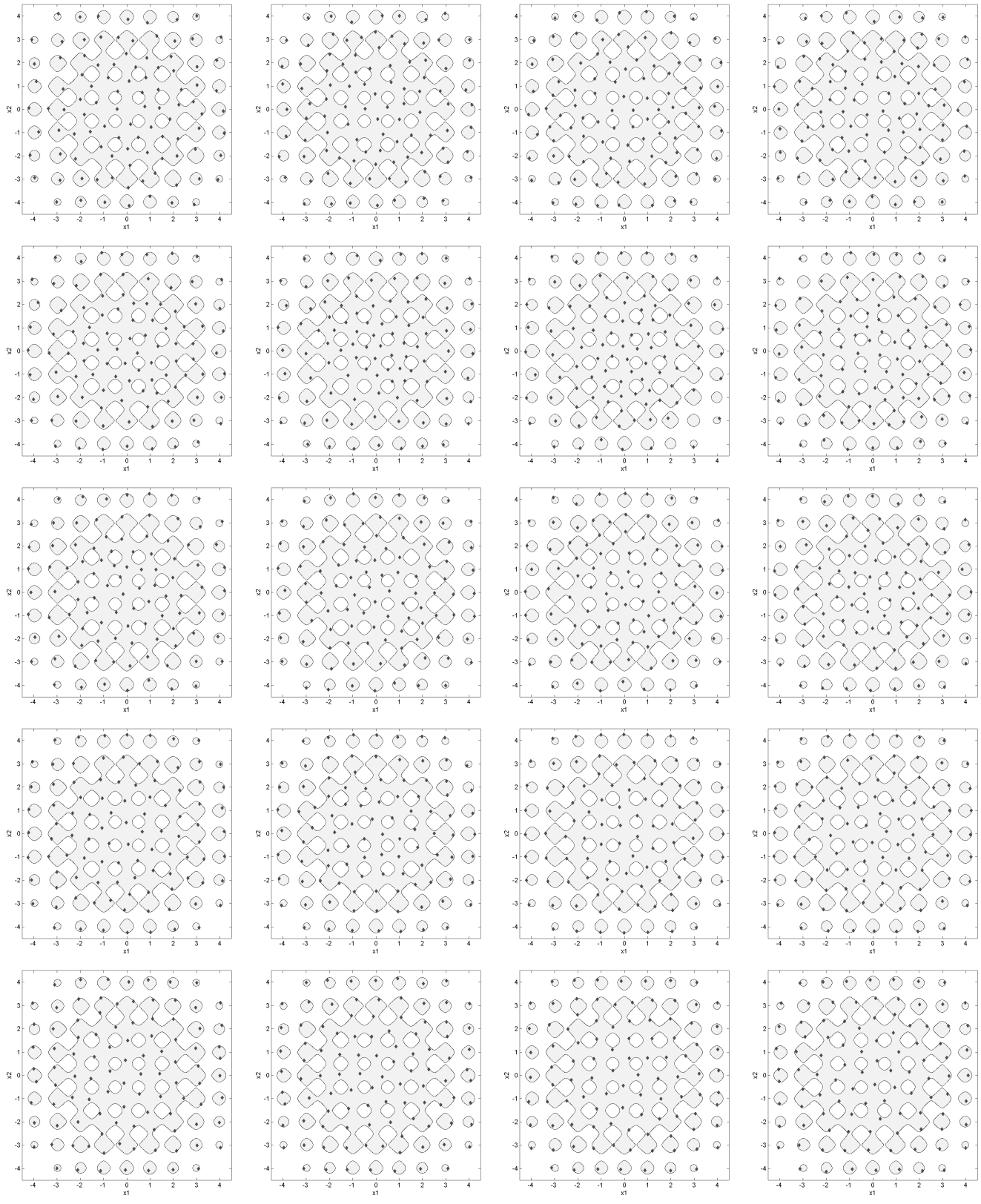

Figure 18: 2D Rastrigin level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$.

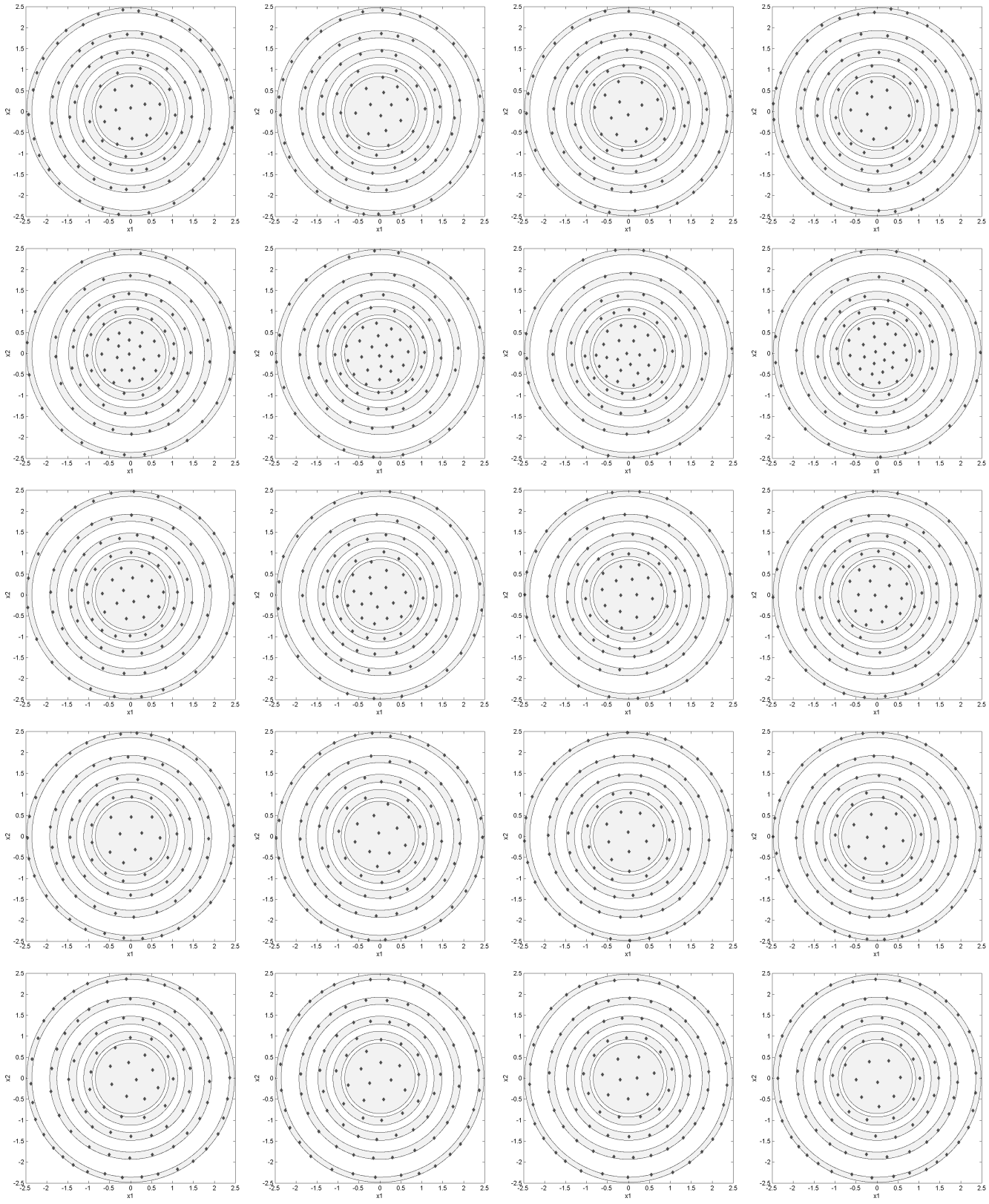

Figure 19: 2D Schaffer level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$\mathrm{SR}_{\omega=0.1}$; fourth column: ELSA-$\mathrm{SR}_{\omega=1}$.

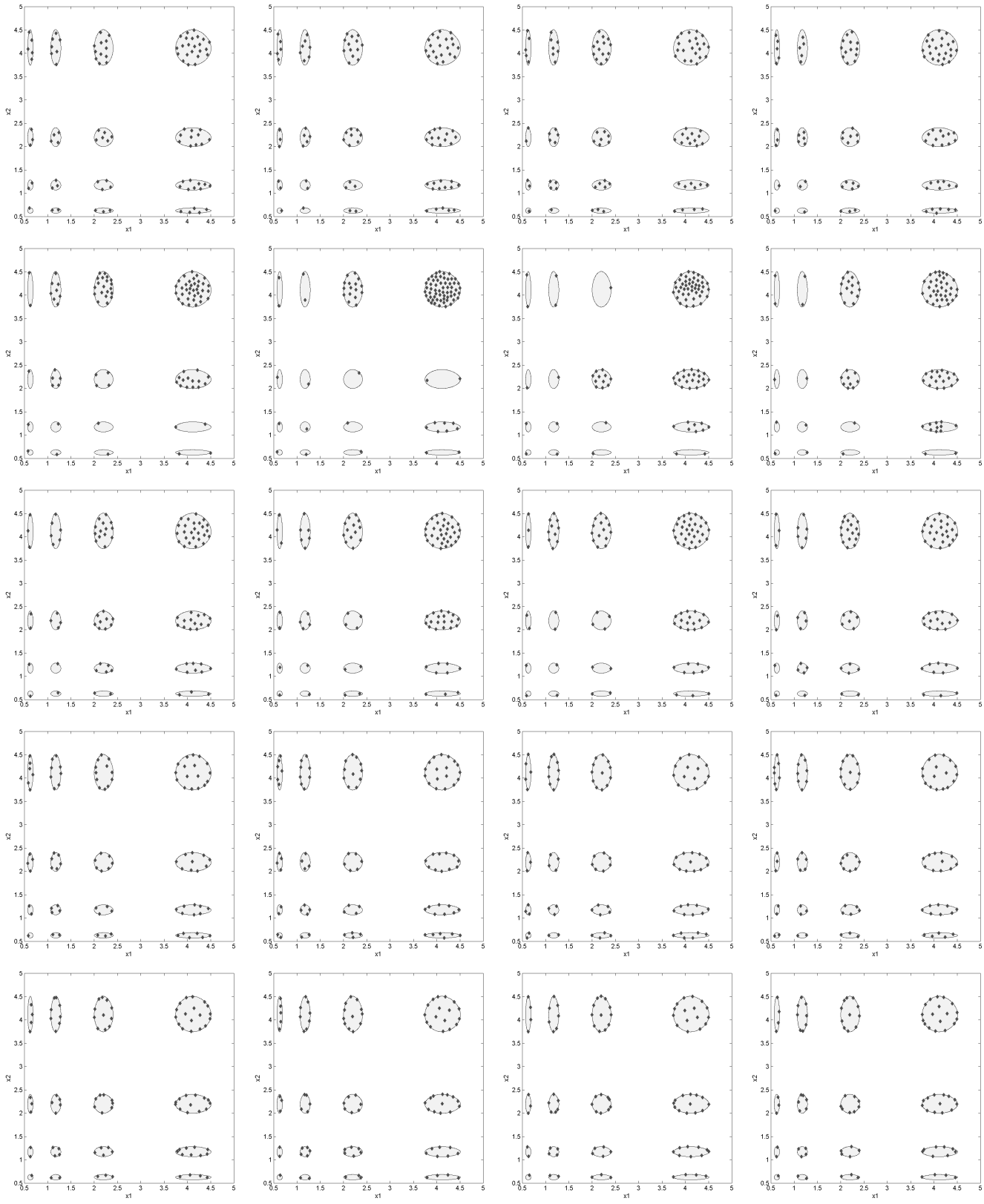

Figure 20: 2D Vincent level set with populations generated from the following quality indicators: first row: GI_N^+; second row: GI_Σ^+; third row: GI_Π^+; fourth row: SPI^+; last row: ADI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$.

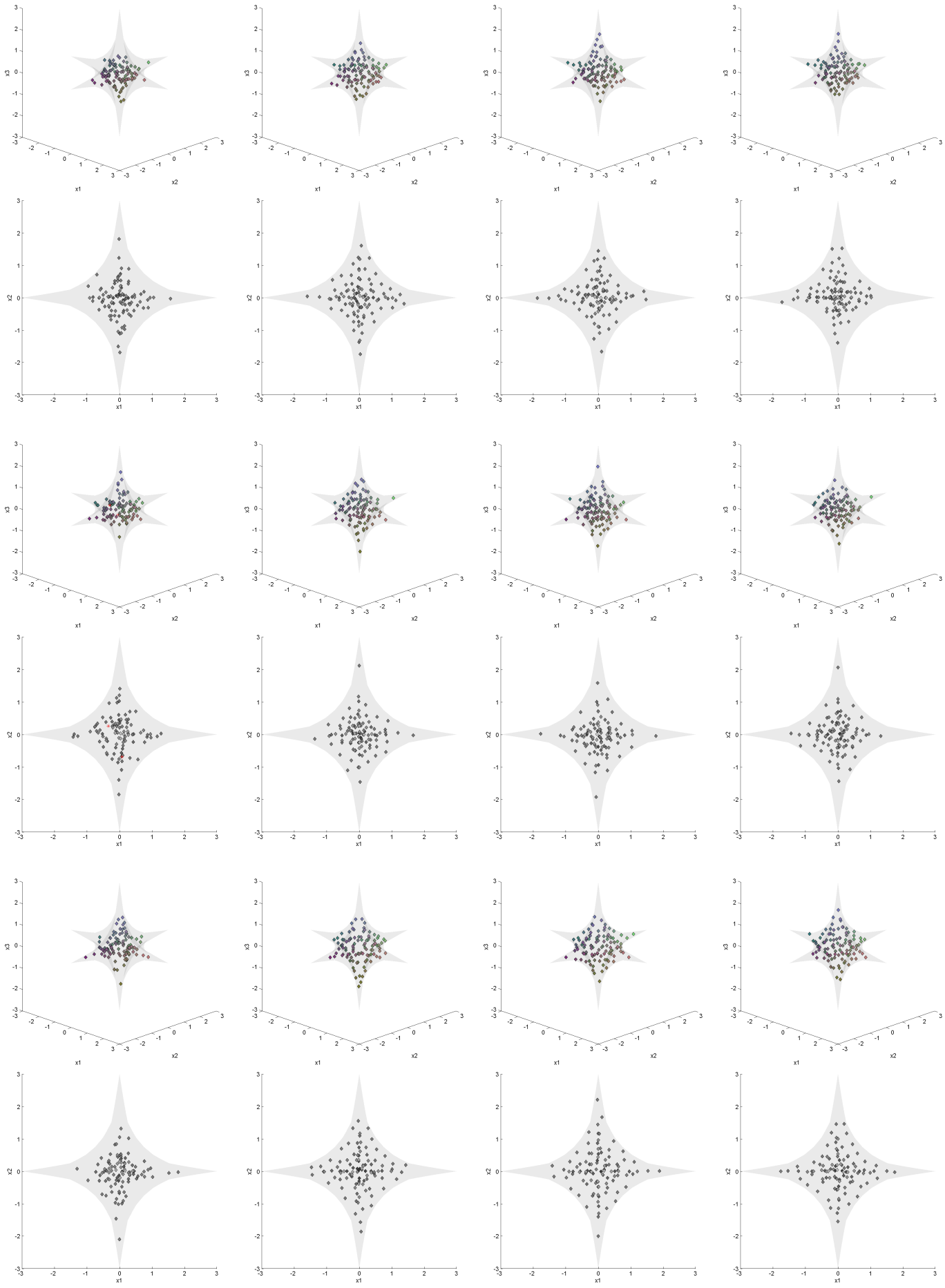

Figure 21: 3D Lamé level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$

Figure 22: 3D Ellipsoid level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$

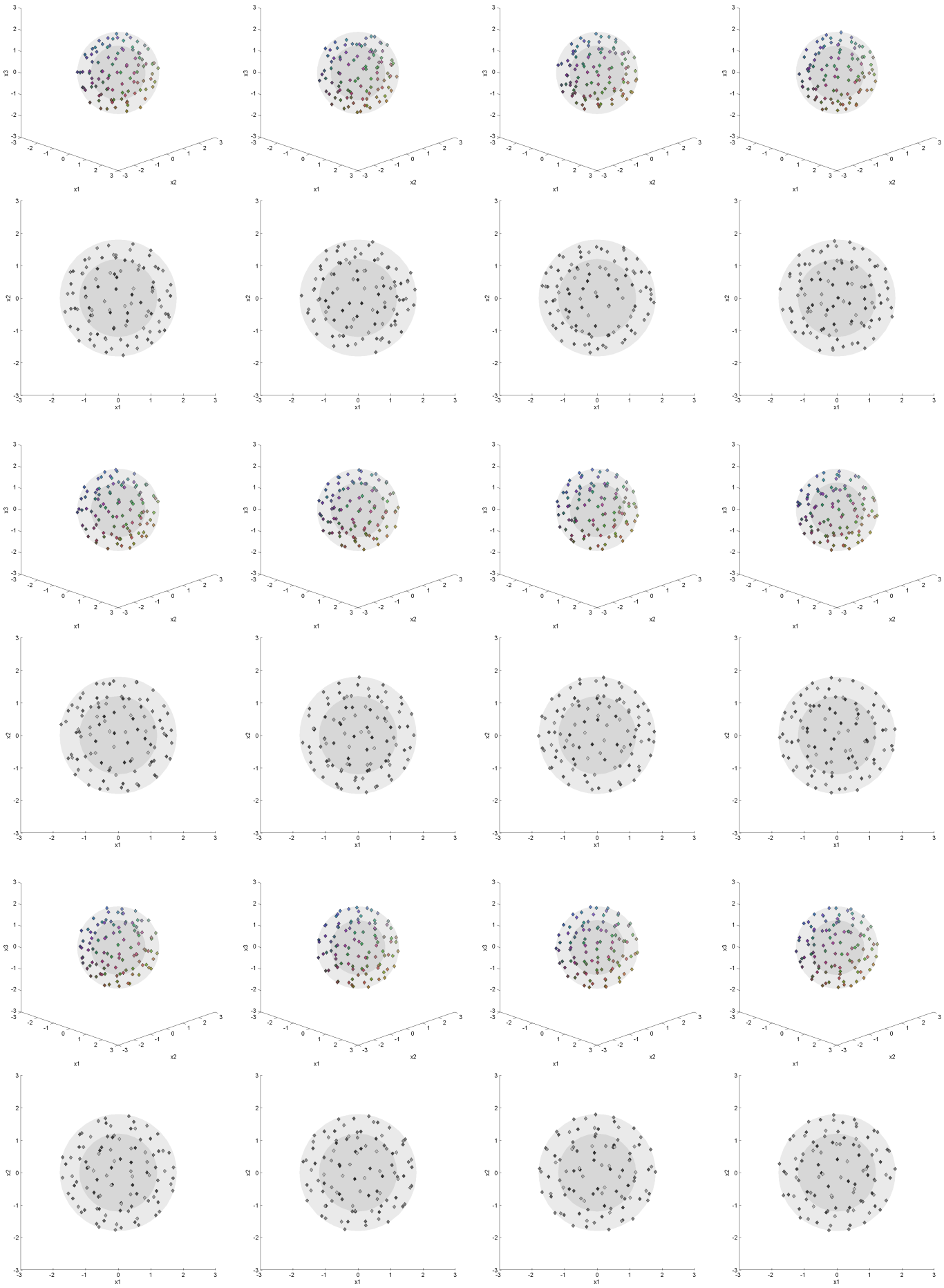

Figure 23: 3D Hollow Sphere level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-SR$_{\omega=0.1}$; fourth column: ELSA-SR$_{\omega=1}$

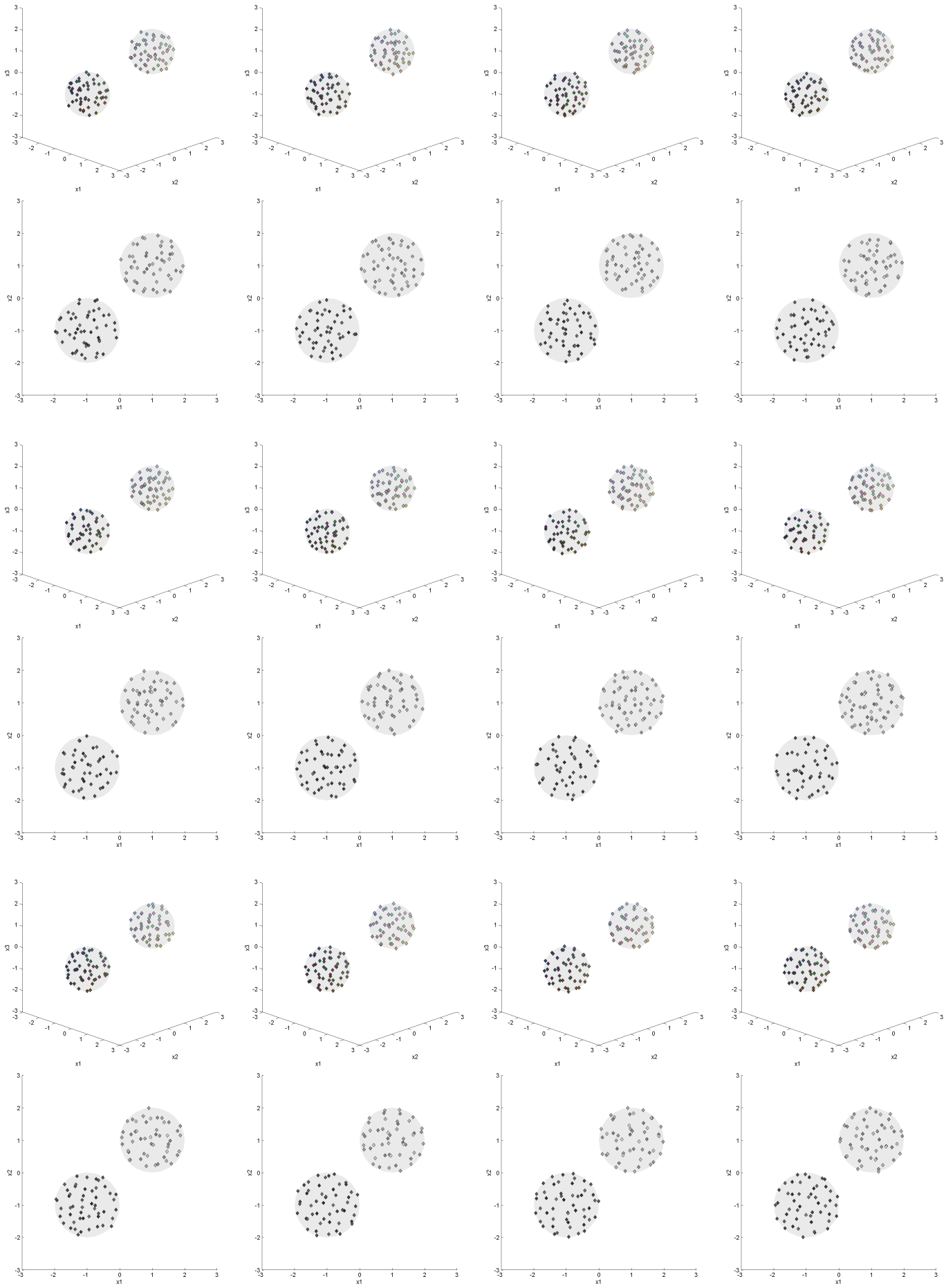

Figure 24: 3D Double Sphere level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$

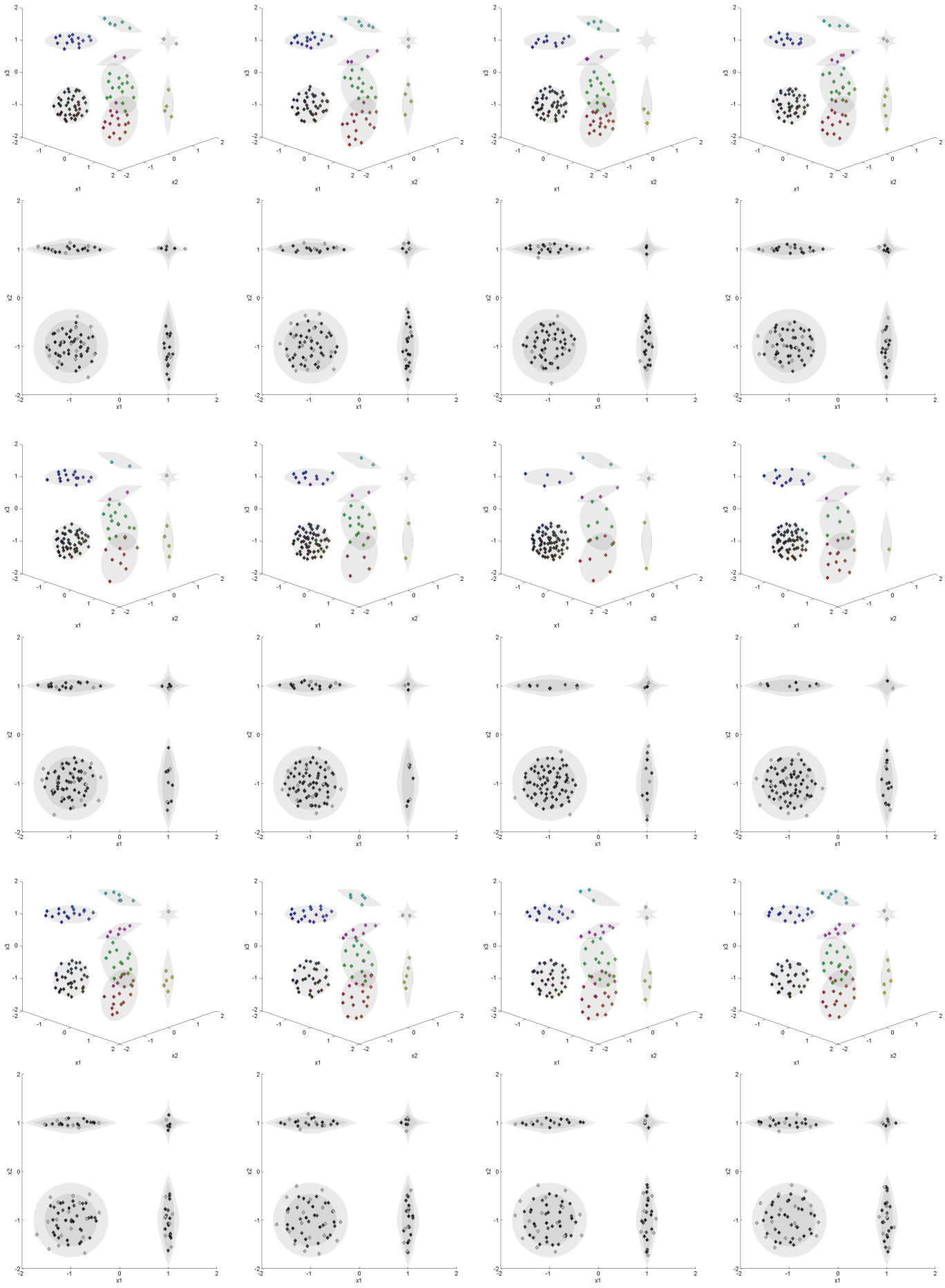

Figure 25: 3D Branke's Multipeak level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-$SR_{\omega=0.1}$; fourth column: ELSA-$SR_{\omega=1}$

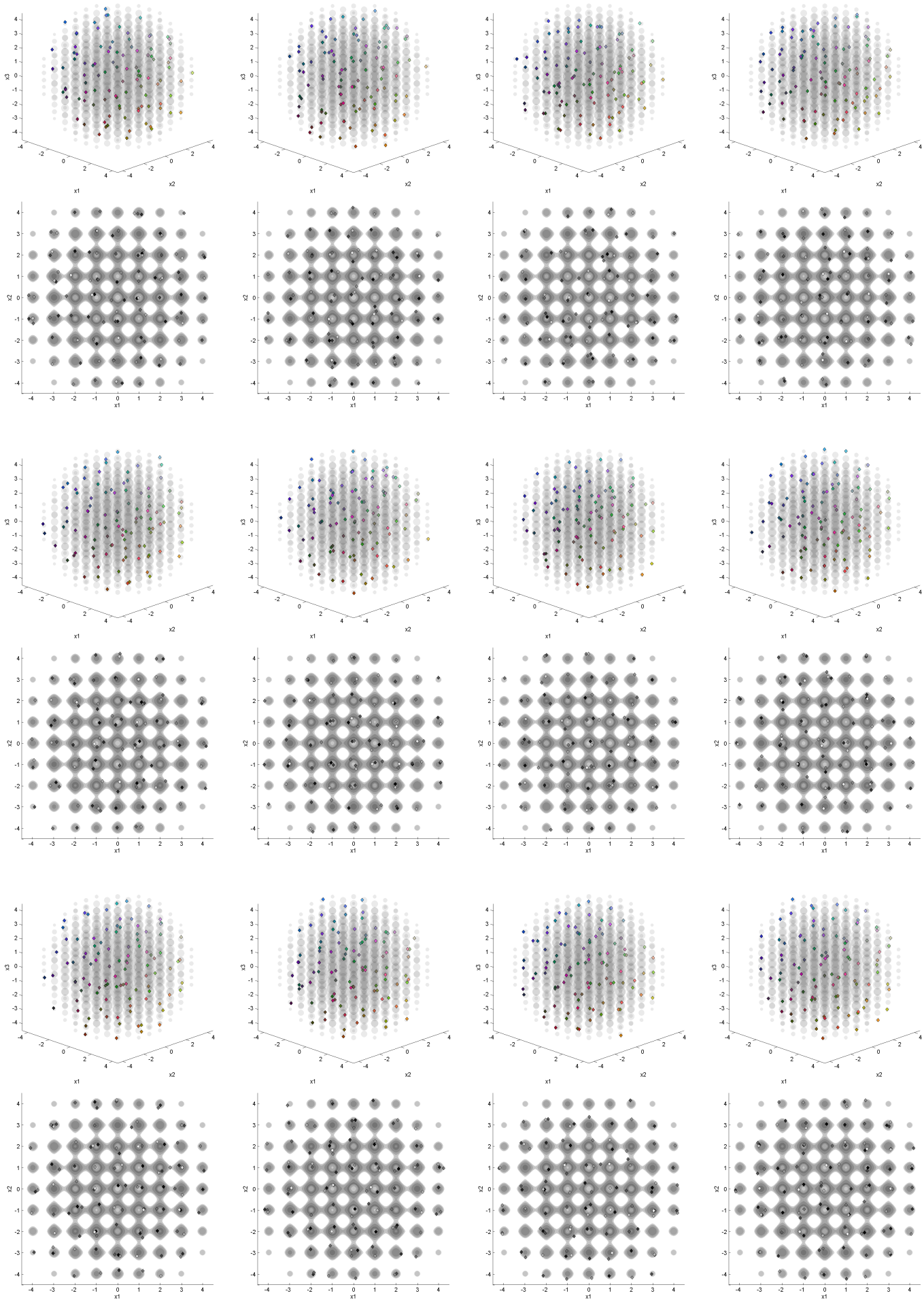

Figure 26: 3D Rastrigin level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-SR$_{\omega=0.1}$; fourth column: ELSA-SR$_{\omega=1}$

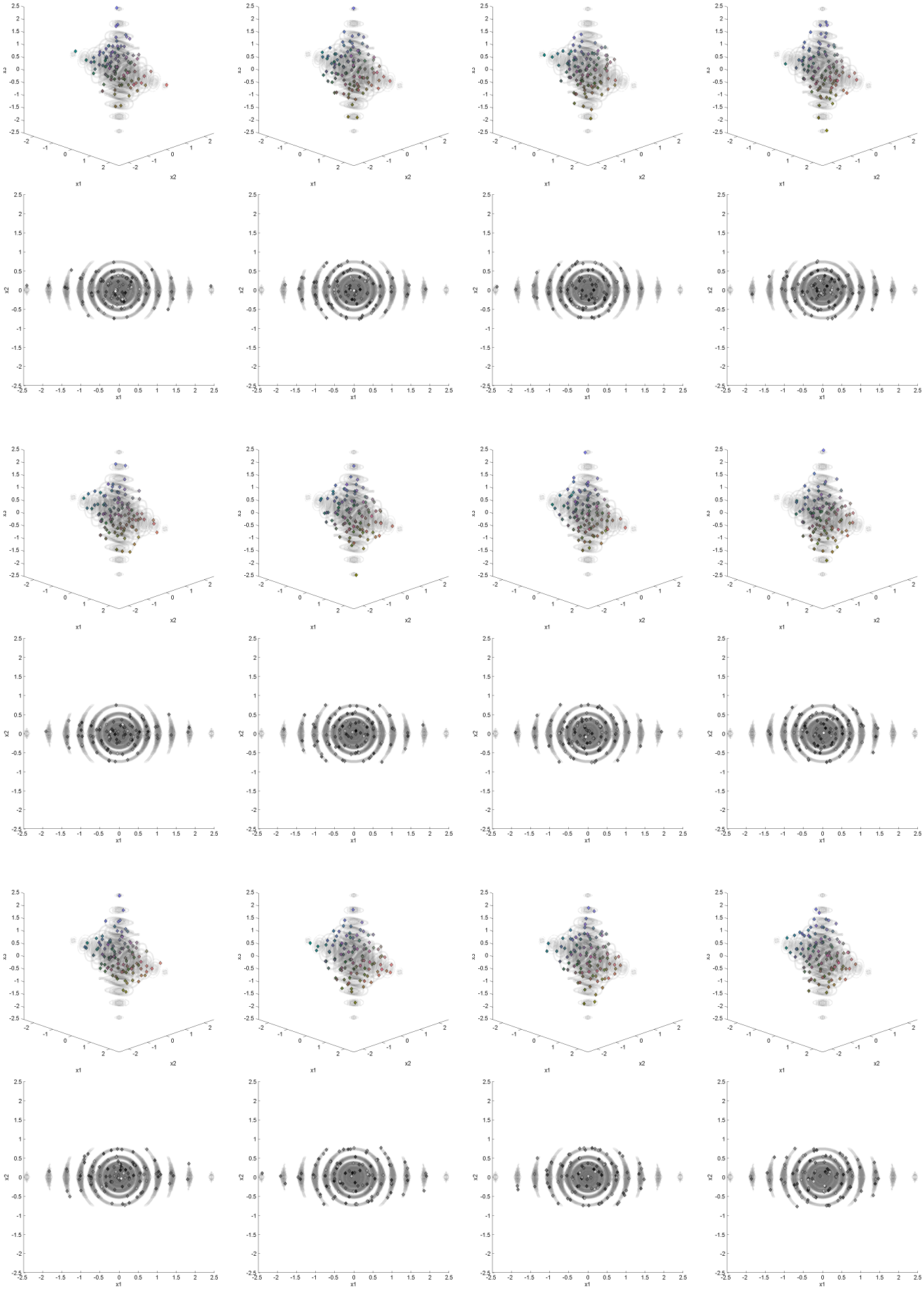

Figure 27: 3D Schaffer level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-SR$_{\omega=0.1}$; fourth column: ELSA-SR$_{\omega=1}$

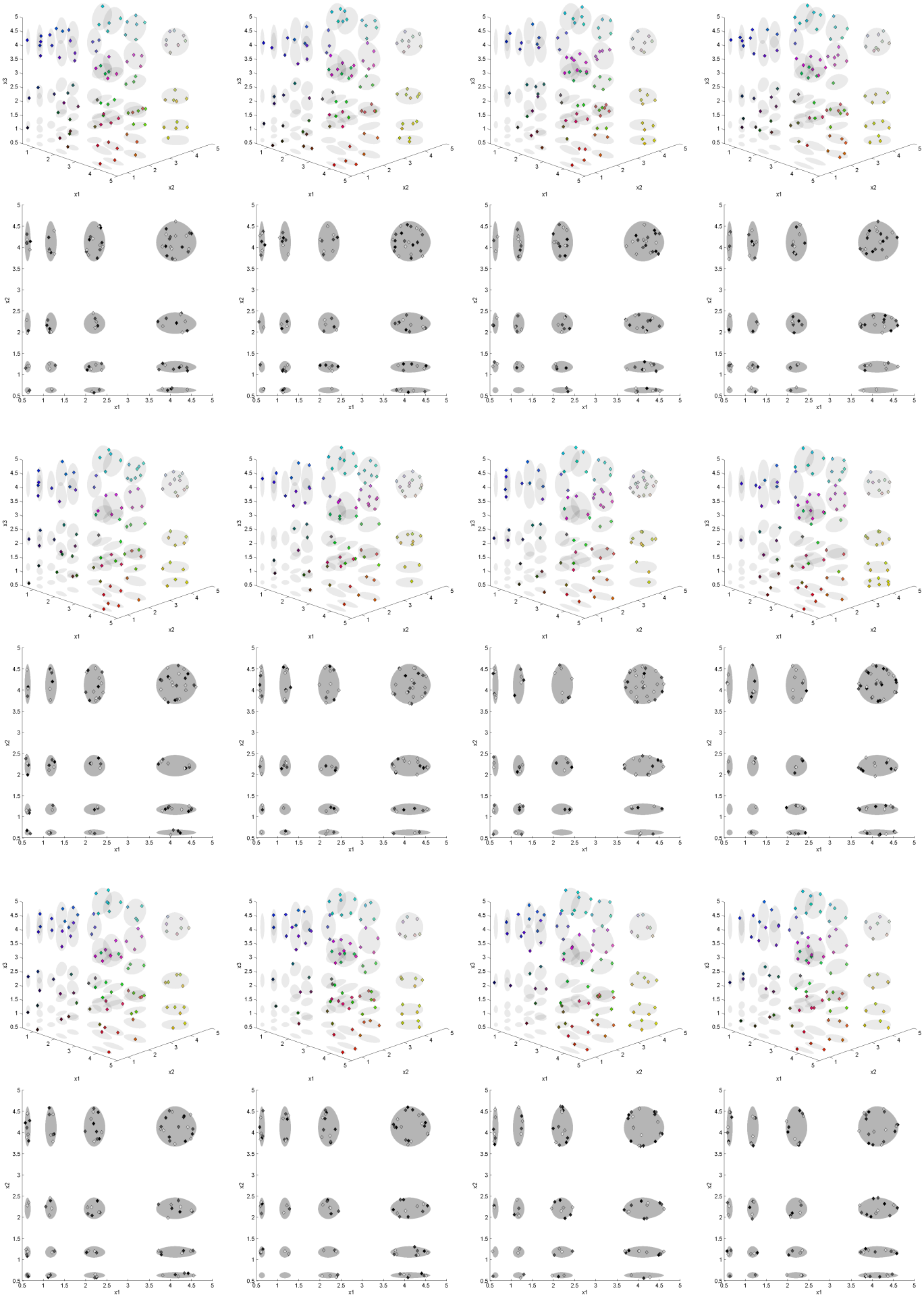

Figure 28: 3D Vincent level set with populations (visualised in pairwise fashion with 3D view on top and orthographic view on bottom) generated from the following quality indicators: first row: GI_N^+; second row: GI_Π^+; third row: SPI^+, and with the following algorithms: first column: MCS; second column: ELSA; third column: ELSA-SR$_{\omega=0.1}$; fourth column: ELSA-SR$_{\omega=1}$