Universiteit Leiden

Master Computer Science

Subpopulations with adaptive population size and search space partition in Covariance Matrix Adaptation Evolution Strategy

Name:	Ernest Vanmosuinck
Student ID:	s3210359
Date:	[28/08/2023]
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. A.V. Kononova
2nd supervisor:	Prof.dr. T.H.W. Bäck

Master's Thesis in Computer Science

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

Abstract

Over the past couple decades, the Covariance Matrix Adaptation Evolution Strategy algorithm has been expanded and improved to solve more and more complex problems. But there isn't much account for researching its performance with the concept of subpopulations. In this research, we investigate the effects that dividing a population into subpopulations of varying sizes has on the performance of CMA-ES. We extend the research to look at the potential benefits of different initialization techniques for the generation of subpopulations' first center of mass. We borrow from niching concepts to separate subpopulations on the search space and explore population size adaptation strategies to further improve subpopulations. We find that, when benchmarked on the BBOB test suite, subpopulations improve the performance of CMA-ES for solving constrained problems in low dimensions, especially when combined with population size adaptation.

Contents

1 Introduction

The search for optimality is at the core of many principles of nature. Fauna and flora evolve to their environment to maximize their survival, like bird beaks or tree leaves [45, 64]. Mankind naturally also searches for optimality when solving real-world problems. Before computation was invented, simulation was a practical way of solving problems. But as problems grow more and more complex, simulation of said problems can become unrealistic.
The development of optimization algorithms for solving complex problems, or optimization heuristics, has been an ever-evolving field since its beginnings [23, 56, 24]. This constant improvement and development led to the invention of many new techniques and algorithms over the years, many still utilized today. One such concept is Evolutionary Algorithms (*EAs*) [89], modelled after the theory of evolution, in which solutions are made to represent individuals of a population, mimicking natural selection, genetic variation, and survival of the fittest to find optimal solutions. Such algorithms usually consist of 4 phases; selection, crossover, mutation and evaluation. Evolutionary Strategies [67, 71, 7] are a category of EAs that does not employ crossover to generate new individuals, instead relying solely on mutation to generate new individuals.
A problem often attributed to population-based algorithms is their struggle to learn from the search space, leading potential solutions to converge too fast and get stuck in local optima. An issue made more obvious in non-convex problems. Due to the pressure of selection, solutions with higher fitness are favoured, which hinders the possibility of leaving a local optimum. There exist a few concepts to improve this problem. For example, adaptive parameters like mutation rate can adjust the exploration-exploitation balance as the evolution proceeds. Another concept is diversity maintenance, such as niching, or what we employ in this research: subpopulations.
In this research, we investigate the following research questions:

- **RQ1**: What are the effects of using subpopulations with the algorithm Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [38]?
- **RQ2**: How do different initialization techniques impact the performance of subpopulation CMA-ES?

Furthermore, we explore the application of promising concepts to CMA-ES when combined with subpopulations:

- **RQ3**: How does enforcing separation of subpopulations across the search space impact the performance of subpopulation CMA-ES?
- **RQ4**: How do adaptive population size techniques affect the performance of subpopulations in CMA-ES?

The run configurations of each experiment are benchmarked on two multi-modal BBOB problems [37]. We pick those two functions as they are among the most interesting functions in the BBOB test suite, being highly multimodal, non-convex problems with a lot of local optima. Finally, we benchmark the best/most promising configurations of previous experiments on an extended test suite of BBOB problems.

1.1 Related Works

The idea of dividing a population into smaller groups of individuals was proposed as early as 1997 by Izumi et al. [41], who introduced competing subpopulations for Evolutionary Strategies as a solution for population diversity. Finding the right population diversity is crucial; if the population becomes too diverse, it will lead to inefficiency and slow convergence by exploring irrelevant regions of the search space. On the other hand, if the algorithm is too exploitative, it will become homogeneous and could lead to sub-optimal solutions [55]. The concept of dividing a population into smaller groups stands at the basis of many other algorithms, such as parallel genetic algorithm (*PGA*) [61] and island models [79]. PGA divides a genetic algorithms population into separate parts that will then be evolved in parallel, using multi-processing to explore the search space faster. The idea of subpopulations was also applied to other algorithms to increase their performance, such as Multi-Swarm Particle Optimization [39] (or the more efficient Niching Migratory Multi-Swarm Optimizer (*NMMSO*) [28]), which uses a form of subpopulation with Particle Swarm Optimization (*PSO*) algorithm. PSO models the behaviour of a flock of birds, and Multi-Swarm PSO simply divides or adds multiple swarms evolving independently. Niching methods are techniques employed to address the loss of solution diversity in multimodal problems [88], finding a larger number of optimal solutions. Evolutionary Algorithms tend to converge toward a single optimal solution, which can be problematic when looking at multimodal problems [53, 78].

A lot of research has been conducted on the performance of Covariance Matrix Adaptation [4, 5, 29] due to its capacity at solving black-box problems. It has been employed for hyperparameter tuning of neural networks, whether that is for surrogate models [49, 48] or for speech recognition [87]. Due to the nature of CMA-ES, the algorithm was quickly applied to solve multi-objective multimodal problems [69, 46, 40]. The use of subpopulations with Evolutionary Algorithms has grown in the past three decades [90, 50, 68, 18]. In the work by Ahrari et al. [1], the authors proposed a new algorithm in the form of covariance matrix self-adaptation evolution strategy with repelling subpopulations (*RS-CMSA*), a niching method using repelling subpopulations and taboo regions. Their algorithm, which expands the pre-existing Covariance Matrix Self-Adaptation Evolution Strategy (*CMSA-ES*) [9], proved to perform well in the CEC2013 competition. The algorithm was further improved in *RS-CMSA-ESII* [2]. In their research, Chen et al. [15] proposed a new algorithm S^3*-CMA-ES*, using subpopulations to improve solving multi-objectives large-scale problems.

We note a lot of research analyzing the effect of different population initialization methods on Evolutionary Algorithms [81, 43, 27], but little research focused on evaluating the effect of those methods on the CMA-ES algorithm as it is designed to function almost parameter-free. Furthermore, unlike a majority of Evolutionary Algorithms, CMA-ES does not initialize the first population individually, but rather samples the individuals from a centroid and sigma [38]. In recent years, de Nobel et al. [19] took inspiration from modular Evolutionary Algorithms framework `ModEA` [82] to create a modular CMA-ES framework. In their works, the authors assess the performance of individual algorithmic ideas by performing hyperparameter tuning on the novel framework ModularCMAES. We will extend this framework in our research to incorporate subpopulations. We note the research of Vermetten et al. [84] in which they investigate the effect of strict box-constraints along with two separate initialization techniques; *center* (center of the search space) and *random* (uniformly random in the domain). We aim to investigate the effect of using different initialization methods to generate the subpopulations' initialization points.

Other techniques and concepts have been invented and researched to improve the performance of CMA-ES. As earlier mentioned, niching methods have shown to improve the performance of CMA-ES, as shown in the work of Shir et al. [76], in which they explore a novel "*adaptive niche radius*" technique for solving the *niche radius problem* [75] and achieve great results. RS-CMSA [1] similarly employ niching techniques for the core of their algorithm to solve multimodal multi-objective problems. In our research, we look into the concept of search space partition as a form of niching to maintain diversity of solutions. The partitioning of the search space has been applied to a variety of problems. While its main uses have been to decrease the complexity of high-dimensional problems [70, 86]. El dor et al. [25] applied space partitioning to the multi-swarm PSO algorithm to solve stagnating convergence to local optima, with the results showing great efficiency.

Another great improvement that CMA-ES has benefited from is population size adaptation. Being a population-based algorithm, the population size is an important yet hard-to-tune parameter. One solution is to employ population size adaptation. Algorithms such as *IPOP-CMA-ES* [6], *BIPOP-CMA-ES* [33] or *CMAES-APOP* [58] are among the most popular algorithms using a form of population size adaptation to improve on the performance of CMA-ES. The many benchmarks on BBOB test suites often put CMA-ES variants with adaptive population size among the best performing algorithm [36, 58, 33, 3]. In their research, authors Nishida and Akimoto [60] propose a CMA-ES variant with adaptive population size, showing competitive results against the BIPOP-CMA-ES algorithm [59].

1.2 Structure

This thesis is divided into the following sections; firstly, in Section 2, we provide definitions and background on concepts used, which are then explained in detail in Section 3. We then cover the experimentation and results in Section 4. Finally, we discuss and summarise this research in Section 5, and present directions for future work in Section 6.

2 Definitions / Background

In this chapter, we provide extended definitions for concepts employed in this study. We begin with an extended definition of the core algorithm used, Covariance Matrix Adaptation Evolution Strategy. We then break down concepts relating to subpopulations, search space partitioning and adaptive population size. Finally, we provide a background on the tools and frameworks employed.

2.1 CMA-ES

Covariance Matrix Adaptation Evolutionary Strategy, more commonly known by its acronym CMA-ES, is an algorithm developed by Hansen, Müller and Koumoutsakos in 2003 [38]. CMA-ES is a stochastic algorithm for optimization of non-linear, non-convex functions. Being an evolution strategy [67, 71, 7], CMA-ES uses populations of individuals to represent solutions, which are updated by the means of mutation and selection. The Covariance Matrix is used by the algorithm to learn the landscape of the objective function, guiding the sampled individuals toward promising areas of the domain. CMA-ES is considered by many to be a state-of-the-art algorithm [47, 42, 57, 74] particularly efficient for black-box continuous optimization problems, which, unlike other powerful algorithms, does not require extensive parameter tuning. Figure 1 shows effectively how CMA-ES samples individuals toward the global optimum. The algorithm can be broken down into the following steps [34].

2.1.1 Offspring Sampling

CMA-ES generates new points of a population (i.e. the offspring) by using multivariate normal distribution. It uses the following equation to sample new search points at a generation g:

$$x_k^{g+1} \sim m^g + \sigma^g \mathcal{N}\Big(0, C^g\Big) \qquad \text{for } k = 1, \ldots, \lambda \tag{1}$$

where:

$x_k^{g+1} \in \mathbb{R}^n$ denotes the k-th offspring from generation $g+1$.

$m^g \in \mathbb{R}^n$ denotes the mean value of the search distribution at generation g.

$\sigma^g \in \mathbb{R}_{>0}$ denotes the standard deviation (or step-size) at generation g.

$\mathcal{N}\Big(0, C^g\Big)$ denotes the multivariate normal distribution (see 3.2.2) with zero mean and covariance matrix C^g.

C^g denotes the covariance matrix at generation g.

From the above equation, we can gather that three parts have to be computed to generate the next population of offspring; the mean m^{g+1}, the covariance matrix C^{g+1} and the step-size σ^{g+1}. Note that this sampling also applies to the first population of individuals, although for this first generation, the mean m^{g_0} cannot be determined from a parent population, and is instead specified (for example the center of the search space) or is randomly placed on the search space.

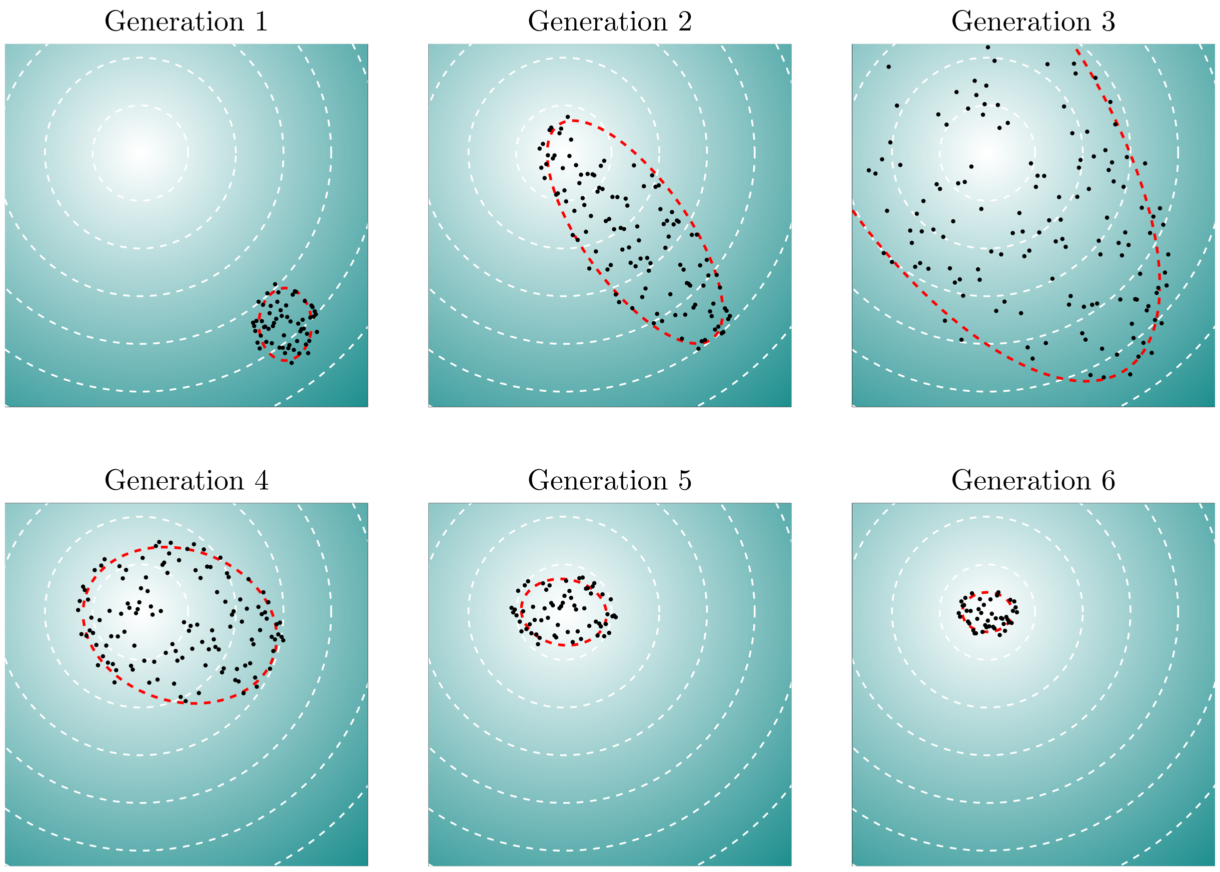


Figure 1: CMA-ES algorithm. The population is sampled from a multivariate distribution. The distribution leads the population to the global optimum.

2.1.2 Selection (i.e. moving the mean)

The mean of the next generation is computed as a weighted average of μ selected points from sample $x_1^{g+1}, ..., x_\lambda^{g+1}$, using the following equation:

$$m^{g+1} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{g+1} \tag{2}$$

where:

$\mu \leq \lambda$ denotes the parent population size.

$w_{i...\mu} \in \mathbb{R}$ denotes the positive weight coefficients for recombination, such that $w_1 \geq ... \geq w_\mu > 0$ and $\sum\limits_{i=1}^{\mu} w_i = 1$.

$x_{i:\lambda}^{g+1}$ denotes the i-th ranked individual in $f(x_{i:\lambda}^{g+1})$ where f is the objective function to be minimized. f is defined by the BBOB problem, and can be bounded or unbounded. CMA-ES can handle both but will require a form of bound correction to relocate individuals sampled outside bounds.

2.1.3 Adapting the Covariance Matrix

CMA-ES's covariance matrix adaptation amounts to learning a second-order model of the underlying objective function, meaning the curvature of the gradient. The algorithm learns the objection function's optimization landscape, helping it reach faster and more accurate results. The update method is a combination of multiple methods, mixing rank-μ and rank-one updates.

Rank-μ update computes the weighted average of elite solutions as a reliable estimator in the form of the following method:

$$C^{g+1} = (1 - c_\mu \sum_{i=1}^{\lambda} w_i) C^g + c_\mu \sum_{i=1}^{\lambda} w_i y_{i:\lambda}^{g+1} \left(y_{i:\lambda}^{g+1}\right)^{\mathsf{T}} \tag{3}$$

where:

$c_\mu \leq 1$ denotes the learning rate for updating the covariance matrix.

$w_{1\ldots\lambda} \in \mathbb{R}$ denotes positive weight coefficient, such that $w_1 \geq \ldots \geq w_\mu > 0 \geq w_{\mu+1} \geq w_\lambda$ and usually $\sum\limits_{i=1}^{\mu} w_i = 1$ and $\sum\limits_{i=1}^{\lambda} w_i \approx 0$.

$y_{i:\lambda}^{g+1} = (x_{i:\lambda}^{g+1} - m^g)/\sigma^g$.

Rank-one update on the other hand modifies the covariance matrix using the evolution path across generations, exploiting the correlation between consecutive steps:

$$C^{g+1} = (1 - c_1) C^g + c_1 p_c^{g+1} {p_c^{g+1}}^{\mathsf{T}} \tag{4}$$

Those two methods (Equations 3 and 4) are combined to form the following update rule:

$$C^{g+1} = (1 - c_1 - c_\mu \sum w_j) C^g + c_1 \underbrace{p_c^{g+1} {p_c^{g+1}}^{\mathsf{T}}}_{\text{rank-one update}} + \underbrace{c_\mu \sum_{i=1}^{\lambda} w_i y_{i:\lambda}^{g+1} \left(y_{i:\lambda}^{g+1}\right)^{\mathsf{T}}}_{\text{rank-}\mu\text{ update}} \tag{5}$$

2.1.4 Step-size control

To control step-size, CMA-ES uses Cumulative Step Length Adaptation (*CSA*), which will adapt the step-size by exploiting the sum of successive steps that have been taken beforehand. CSA tracks the progress of the evolution path and increases the step-size when making progress, and decreases it when it stagnates. CMA-ES uses CSA in the form of the following equation:

$$\sigma^{g+1} = \sigma^g \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{g+1}\|}{\mathbb{E}\|\mathcal{N}(0,\mathbb{I})\|} - 1\right)\right) \tag{6}$$

where:

σ^g denotes the step-size at generation g.

$c_g < 1$ denotes a user-defined learning rate.

$c_g \approx 1$ denotes a user-defined damping parameter.

$p_\sigma^{g+1} \in \mathbb{R}^n$ denotes the evolution path at generation $g+1$.

$\mathbb{E}\|\mathcal{N}(0,\mathbb{I})\|$ denotes an expectation of the Euclidean norm of a $\mathcal{N}(0,\mathbb{I})$, where $\mathbb{I}$ represents an identity matrix.

2.2 Niching

In many cases involving a multitude of optima, Evolutionary Algorithms may struggle to efficiently find that set of optima without resorting to multiple independent runs, which is not computationally efficient [61, 78, 77]. One can consider running those processes simultaneously on separate CPUs (as is the idea behind Parallel Genetic Algorithms [61]), but this process still does not ensure finding all of the optima. A more efficient technique comes in the form of niching methods.
Niching methods are techniques/concepts used by Evolutionary Algorithms to address the loss of solution diversity when dealing with multimodal problems [78, 76]. A traditional EA will only be able to find a single optimal solution from a problem with multiple solutions. Niching addresses this issue by finding a set of optimal solutions. Individuals of a population are separated in some way into different subgroups (i.e. *niches*) that will focus on finding the optima of a specific area of the search space, discouraging convergence toward a single region of the objective function. One such technique is fitness sharing, where the fitness of individuals are scaled based on their relative proximity to other individuals. This ensures that densely populated regions have a lower fitness value than sparsely populated regions [53]. Another popular technique is crowding, which modifies the selection scheme to decide which individual is going to survive to the next generation. Crowding pairs up an individual from the offspring population with an individual from the parent population using a crowding distance d. The strategy then chooses which individual to choose based on its fitness. Deterministic crowding for example will keep the individual with the best fitness [53, 13]. Niching has been applied to a multitude of algorithms (GAs, PSO, ES, ...), proving to improve diversity and discovering a greater set of optima for multimodal problems [53, 30, 78, 75]. While they borrow from similar principles, it is important to note the difference between niching and subpopulation, and their intended purposes. Niching methods aim to divide populations of individuals to focus on specific sections of the search space, identifying global optima in that regions. Subpopulations on the other hand enable groups of individuals to evolve independently, resulting in more exploration of the search space and greater diversity. We provide a summary of similarities and differences between methods that separate populations of individuals to palliate specific issues in Table 1.

2.3 Island Models

As stated previously, three issues commonly attributed to Evolutionary Algorithms are that they can be computationally expensive and take a long time to reach the optimal solution, and they often converge and stall at a local optimum [31, 72]. Island models (*IMs*) have been found to resolve those issues quite well, especially the former [79]. Island models are models of Evolutionary Algorithms inspired by the concept of biogeography. In island models, a population of candidate solutions is divided into multiple subpopulations called

Table 1: Similarities and differences between cited algorithms using subpopulations

Name	Similarities	Differences
Parallel Genetic Algorithm	population-based, evolves populations using selection, mutation and crossover	subpopulations are distributed over separate processes to evolve populations simultaneously and are kept strictly separate
Island Models	population-based, maintains diversity withing subpopulations similarly to niching	individuals often migrate between island (i.e. subpopulations)
CMA-ES	population-based, uses evolutionary principles (without crossover)	uses covariance matrix adaptation to sample individuals from mutation, using CSA (equation 6) to control the mutation step-size
Niching strategies	typically population-based, like island models, maintains population diversity	primary focus is to address multimodal problems, discouraging convergence toward a single region of the objective function

islands. Each island operates independently and runs its own evolutionary algorithm. Islands can exchange knowledge periodically through a variety of different mechanisms, be that through copying individuals, or performing mutation and crossover operations between islands.
The underlying idea behind island models is to introduce forced diversity and prevent premature convergence to a local optimum, similarly to the concept of niching. Through the use of parallelism borrowed from parallel GAs, island models can run each subpopulation's algorithm concurrently, reducing computation. By allowing islands to evolve independently and migrate individuals, the algorithm explores multiple areas of the search space simultaneously.

2.4 Promising Concepts

Additionally, it is important to provide a background of the techniques that were researched to further improve subpopulations and CMA-ES: search space separation and adaptive population size control.

2.4.1 Support Vector Machine for Separation

Support Vector Machines (SVMs) are supervised machine learning models most commonly used for classification and regression analysis, proven to be effective for solving linear and non-linear problems. The concept behind Support Vector Machines is to find an optimal hyperplane that will separate different classes of data points.
Support Vector Machines can handle linear and non-linear machines through the use of different types of kernels. The kernel transforms input data into a higher-dimensional feature space that is easier to linearly separate. The open-source library `scikit-learn` offers a variety of kernels, such as RBF, sigmoid, polynomial, and linear [65].

2.4.2 CMA-ES with adapting population size

Population size is one of the core parameters of any evolutionary algorithm and can be hard to properly tune. A population size larger than its default $4 + \lfloor 3\ln(n) \rfloor$ can lead to better results [6, 4, 5] of noisy multimodal problems, but it becomes on the same occasion more computationally expensive.

IPOP-CMA-ES, which stands for Increased POPulation size CMA-ES, builds upon the CMA-ES algorithm by incorporating the concept of increasing the population size at restarts [6]. If CMA-ES stops before reaching the optimum target or/and before exceeding the budget, a restart occurs. At each restart, the parent λ and offspring μ sizes are increased by a set factor, typically between 1.5 and 5. Authors of the IPOP-CMA-ES paper [6] found doubling the population up to 512 to be a good balance.
By increasing the population size, IPOP-CMA-ES aims to hop out of local optimum by covering more of the search space, providing a good balance between exploration and exploitation. IPOP-CMA-ES has proven to be quite effective on multimodal functions [3].

BI-Population CMA-ES, or BIPOP-CMA-ES, is another algorithm that uses increasing population size at restarts [33]. Compared to IPOP-CMA-ES however, BIPOP-CMA-ES switches between two different interlaced regime schemes using the regime with the lowest budget of function evaluations. If the first regime is used, the population size is doubled as $\lambda_{large} = 2^{i_{restart}} \lambda_{default}$ up to a limit of 9 restarts (or $\lambda_{large} = 512$). Under the second regime, CMA-ES is restarted with a smaller population size:

$$\lambda_{small} = \lfloor \lambda_{default} (\frac{1}{2} \frac{\lambda_{large}}{\lambda_{default}})^{\mathcal{U}[0,1]^2} \rfloor$$

where $\mathcal{U}[0,1]$ is a uniform distribution in [0,1] and $\lambda_{small} \in [\lambda_{default}, \lambda/2]$.
When benchmarked on BBOB noiseless and noisy functions, IPOP-CMA-ES and BIPOP-CMA-ES have shown to perform quite well [4, 5].

CMA-ES-APOP is a variant of CMA-ES which uses adaptive population sizes to solve noiseless multimodal functions [58]. The algorithm collects information on the parameter n_{up}, which is the number of times the median fitness of the current iteration is better than the previous iteration for $S := 5$ consecutive iterations. The size is then adapted for the next S iterations. CMA-ES-APOP was benchmarked on the BBOB test suite, and showed good performance on well-structured multimodal functions, but performed poorly on weakly-structured multimodal functions [58].

2.5 Frameworks

Additionally, we present a background to the resources and frameworks that are used to conduct the research. This regards a summary of available CMA-ES frameworks available, and the one selected. We also detail the platform used for experiments and the test suite on which those experiments are benchmarked.

2.5.1 ModularCMAES

There are a few frameworks available offering implementations of the CMA-ES algorithm, such as `pycma` [35], a python implementation by N. Hansen, author of the CMA-ES paper. We also note the python packages `cmaes` [73], which offers CMA-ES and some of its variants (Warm-Starting, IPOP-CMA-ES, ...) and `pymoo` [10], which offers greater support for multi-objective problems.
In this project, we used the python library `ModularCMAES` [83, 19]. This package offers a modular implementation of CMA-ES algorithms, which allows for easy tuning and testing. Furthermore, this package is built with the `ioh` package, which makes benchmarking on the BBOB test suite more accessible.

2.5.2 IOHprofiler / IOH

IOHprofiler [21] is a benchmarking platform that offers a variety of tools for evaluating iterative optimization heuristics. It can be broken down into multiple components handling different purposes. Three of those components will be broken down in detail as they will be used in this research.
IOHproblem offers the implementation of multiple benchmark problems, such as Pseudo-Boolean problems (PBO) and Black-Box Optimization Benchmarking (BBOB), which is used to benchmark this research. The BBOB test suite is an easy-to-use framework designed by COCO for benchmarking black-box optimization algorithms. The framework has been used for multiple workshops of different conferences, like GECCO (2009, 2010, 2012, 2013, 2015-2022) and CEC (2015) [4, 5, 37].
IOHanalyzer accepts benchmarked data from the user and provides detailed graphical analysis back [85]. IOHanalyzer allows for the analysis of results in two ways: as fixed-target results and as fixed-budget results. Fixed-target results encapsulate the running time required to obtain a desired target value. Fixed-budget results encapsulate the objective function values obtained given a certain budget of function evaluations.
IOHdata contains an archive of selected sets of benchmarks of different sources, such as COCO and Nevergrad. Using those recorded results, we are able to compare the performance of our algorithms.

2.6 BBOB

As mentioned in the section above, the BBOB test suite is a powerful framework often used to benchmark optimization algorithms [37]. The test suite covers a variety of functions, separated into multiple categories. For this research, we used the standard test suite of noiseless, single-objective and scalable test functions. This suite contains 24 functions, available in variable dimensions (2, 3, 5, 10, 20 and 40), and 15 instances. Table 3 provides a description of the functions available and their properties. While CMA-ES is able to handle both constrained and unconstrained problems, we will only be using constrained domains in our research. We bind the domain between -5 and 5.

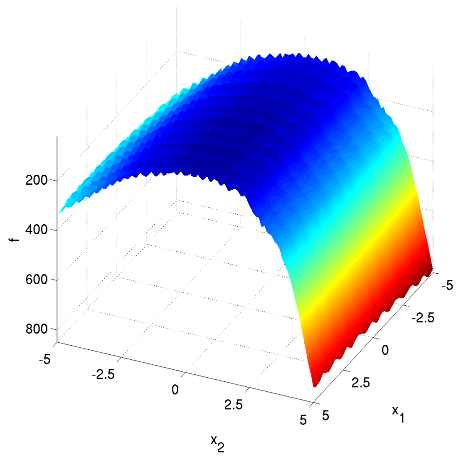
In our experiments (Sections 4.2, 4.3, 4.4 and 4.5), BBOB functions 3 (Equation 7) and 4 (Equation 8) were used to evaluate the performances of configurations. Those functions are highly multimodal, with roughly 10^D local optima (D is the dimension). The BBOB Rastrigin function introduces a factor to alleviate the symmetry and regularity of the original Rastrigin function.

$$f_3(\mathbf{x}) = 10\left(D - \sum_{i=1}^{D} cos(2\pi z_i)\right) + \|\text{z}\|^2 + f_{opt} \tag{7}$$

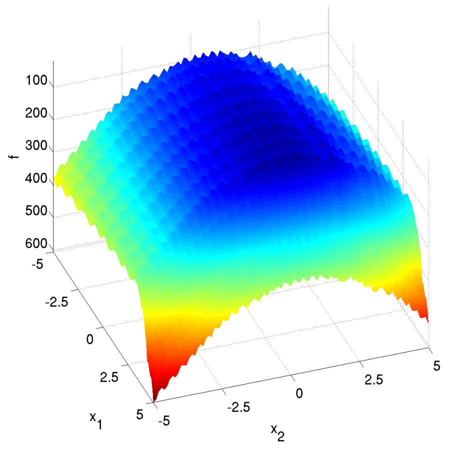
with $\mathbf{z} = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(x - x^{opt}))$ as disturbance factor

$$f_4(\mathbf{x}) = 10\left(D - \sum_{i=1}^{D} cos(2\pi z_i)\right) + \sum_{i=1}^{D} z_i^2 + 100 f_{pen}(\mathbf{x}) + f_{opt} \tag{8}$$

- $z_i = s_i T_{osz}(x_i - x_i^{opt}) \quad$ for i=1...D
- $s_i = \begin{cases} 10 \times 10^{\frac{1}{2}\frac{i-1}{D-1}} & \text{if } z_i > 0 \text{ and } i = 1, 3, 5, ... \\ 10^{\frac{1}{2}\frac{i-1}{D-1}} & \text{otherwise} \end{cases}$ for i=1...D



Rastrigin function (equation 7)



Büche Rastrigin function (equation 8)

Figure 2: BBOB objection functions 3 and 4 taken from the BBOB function techreport [29]. Both functions show high multimodularity, which complicates optimization.

3 Methodology

In this section, we detail the methods and concepts that were investigated and experimented on. We provide definitions of each technique or algorithm benchmarked in Section 4. The source code of our implementation is available in the project repository[1].

3.1 Sub-population in CMA-ES

As mentioned earlier, we use the pre-existing python package `ModularCMAES` [19]. Subpopulations are represented as identically initialized algorithms evolving together in a same search space. The modularity of ModularCMAES enables us to create separate instances of CMA-ES with independent variables, but share an IOH logger to record the results of all subpopulations as one algorithm. This means that we are able to create instances with exactly the same parameters (e.g. `SP-CMA[10]` which has 10 identical subpopulations, or instances with different parameters (`SP-CMA[MIXED]` which has a big subpopulation along with 5 smaller ones). We use the abbreviation *SP-CMA*$[n]$ for subpopulation CMA-ES configurations with subpopulations of n individuals. By making subpopulations identical, we posit that subpopulations are given equal chances of exploring and exploiting the search space. The algorithm employs pseudo-parallelism to evolve populations one at a time. Instead of using PGA's parallelism on separate CPUs [61], each subpopulation takes turn evolving on one shared CPU. The initialization point of a subpopulation, i.e. the first population's center of mass, is determined using uniform sampling as this is the default setting for ModularCMAES and is the conventional initialization technique used by many algorithms. While the subpopulations share the search space and may explore the same areas of the domain, they do not interact with each other in any capacity. This makes our algorithm a mix of island models and PGA, separating a population into subgroups (i.e. subpopulations) and evolving them in pseudo-parallel, prohibiting any interactions between the subpopulations.

3.2 Centroid initialization techniques

The performance of an Evolutionary Algorithm is greatly impacted by the initialization of the first population, and while making *good* guesses can facilitate finding a local optimum [66], one cannot make guesses about the search space in black-box problems [16]. For this reason, many researchers employ pseudo-random number generators to initialize the initial population (or population centroid for CMA-ES), and using a strong initialization method can improve the performance of the algorithm [16, 51, 52]. When dealing with constrained problems (as is the case in this study), points may be sampled outside of the bounds and will need to be resampled or corrected to prevent the algorithm from exploring infeasible solutions [84]. We use *saturate* correction, which corrects out-of-bounds points to the nearest corresponding bound [84]. With limited population sizes, the chances of exploring promising regions of the search space decrease as the dimensions of the search space increase. The importance of generating an initial population covering the search space effectively can thus result in higher solution quality. We break down a few techniques that we used to initialize the initial centroid of subpopulations. Figure 3 displays how each method described below affects the sampling of 100 points in 2 and 3 dimensions.

[1]`https://github.com/ernestvmo/ModularCMAES`

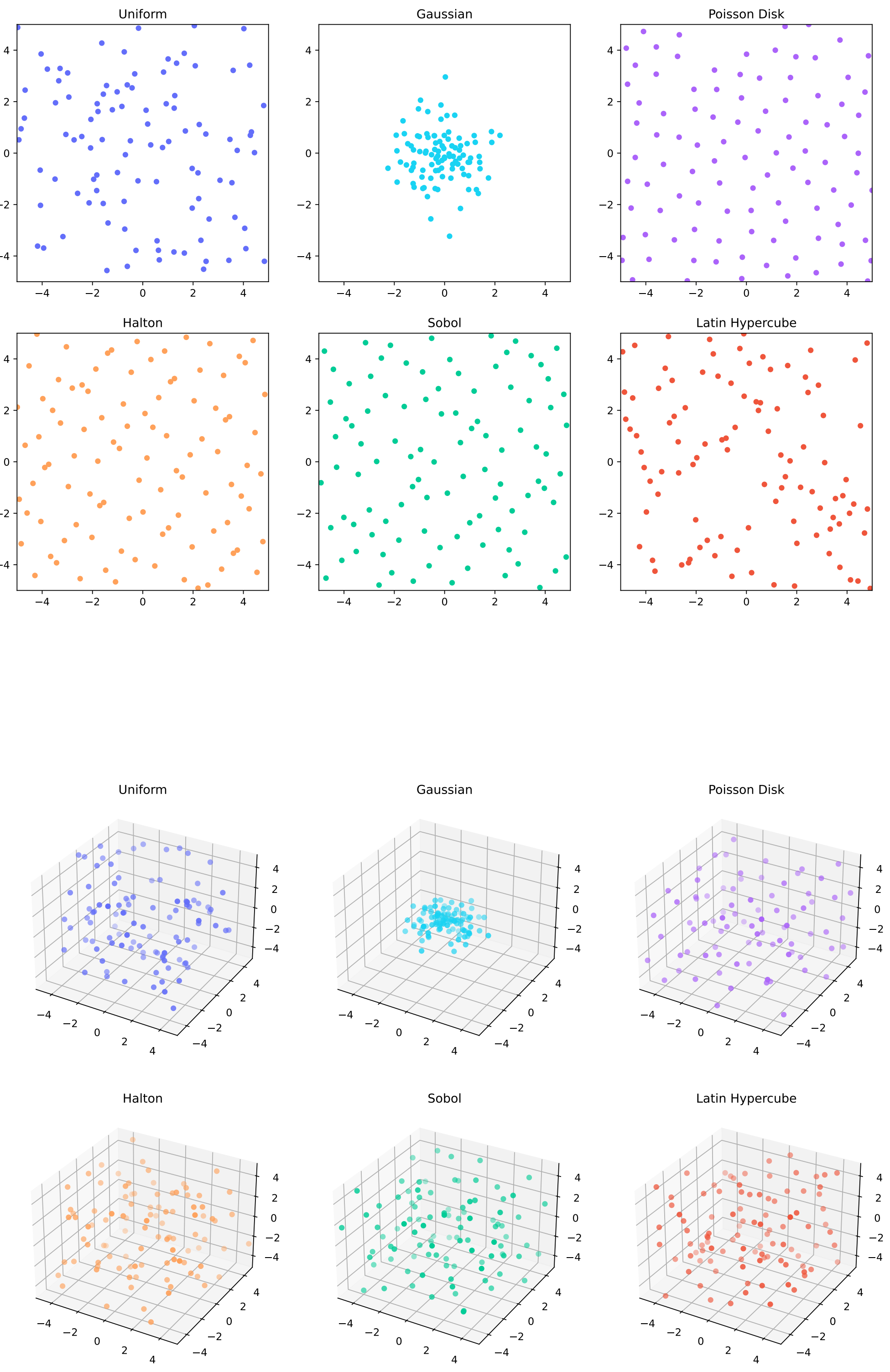


Figure 3: Sampling 100 points using different initialization methods.

3.2.1 Uniform Sampling

Uniform sampling involves generating random points or samples within a given space, such that each point has an equal probability of being selected. The distribution of points is uniform, meaning they are evenly spread across the space. In uniform sampling, all values within the range of the distribution have an equal chance of being sampled, resulting in a flat or rectangular probability distribution. With CMA-ES, we uniformly sample n number of points, with n representing the number of subpopulations. The points are used for the first centroid of each subpopulation.

3.2.2 Gaussian Sampling

Gaussian sampling generates samples taken from the normal distribution. The distribution is symmetric around its mean, showing that data occurs more frequently around the mean. It is the core principle behind the probability bell curve. The distribution uses the formula $f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$, where μ represents the mean and σ represents the standard deviation. This formula transforms a uniform distribution into a Gaussian distribution. In a Gaussian distribution, 68.2% of the samples will be within $\pm$ one σ away from the mean, 95.4% of the samples within $\pm$ two σ from the mean, and 99.7% of the samples within $\pm$ three σ from the mean.

3.2.3 Poisson Disk Sampling

Poisson Disk sampling [12] generates sets of points that are uniformly distributed across a given space while maintaining a minimum distance, known as radius, between two points. Poisson Disk sampling ensures that newly generated points are placed at a safe distance from existing points, resulting in well-distributed points while avoiding clustering. Taking an n-dimensional space, the domain is divided into cells of size $r/\sqrt{n}$, where r is the minimum distance between samples (radius), and each cell can only contain one sample. The disadvantage of Poisson Disk is that it is quite dependent on finding the right radius according to the number of points to generate. A smaller radius might lead to regions of the search space not covered, while a bigger radius might lead to points outside the search space. We found that using a radius of 0.2 provided a good distribution for the different subpopulation configurations in 2 dimensions and 3 dimensions (see Poisson Disk in Figure 3).

3.2.4 Latin Hypercube Sampling

Latin Hypercube sampling is a stratified-random procedure proposed in 1979 [54]. To generate n points, Latin Hypercube sampling divides each dimension of the parameter space into n equally spaced non-overlapping intervals and then samples points randomly from each interval. This process ensures great coverage of the search space. While Latin Hypercube is typically used to cut down computer processing time [62], its design is compelling for population initialization. Furthermore, Latin Hypercube scales well to higher-dimensional spaces by comparison to uniform sampling as dimensions are sampled separately [63].

3.2.5 Halton Sampling

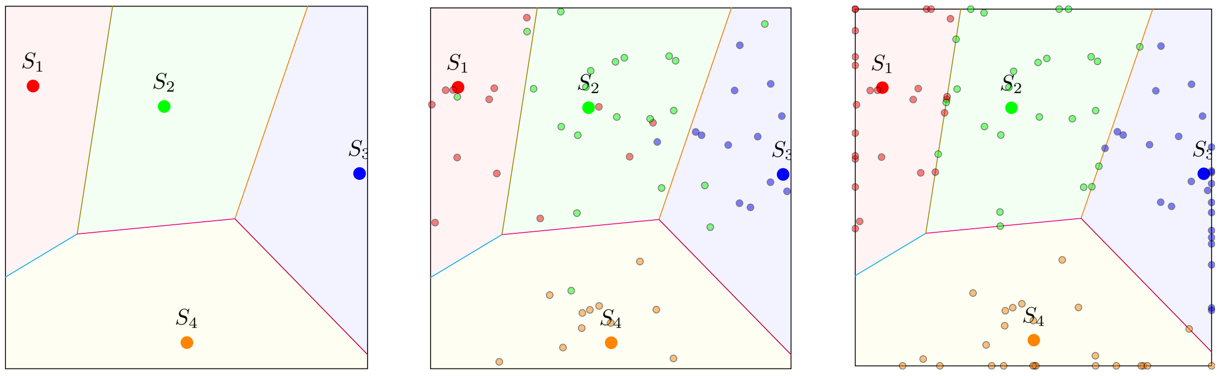
Halton sampling is a low-discrepancy, deterministic quasi-Monte Carlo method that generates points based on the Halton sequence [32]. The Halton sequence uses specific sets of coprime numbers, integers whose only common positive integer divisor is 1. The n-th sample from the sequence is generated by using the coprime of each dimension to convert n to base of the coprime. For example, consider the 10th point of the Halton sequence in three dimensions. We use coprimes 2, 3 and 5 for each dimension respectively. We transform 10 to base 2, '1010', invert it and add it behind the decimal point, 0.0101_2, which results in $1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} + 0 * 1^{-4} = 0.3125$. Apply the same process on the other dimensions (but with bases 3 and 5), we find that the 10th point of the sequence is (0.3125, 0.37035, 0.08). While the Halton sequence performs well in low dimensions, it is often modified to alleviate linear correlation in higher dimensions by using scrambled Halton [11] or leaped Halton [44].

3.2.6 Sobol Sampling

Sobol sampling is a low-discrepency, deterministic quasi-Monte Carlo method that generates points using the Sobol sequence [80]. The Sobol sequence, first introduced by Ilya M. Sobol in 1967, uses a combination of primitive polynomials, direction vectors and gray code [17, 22] to generate uniform points on the search space with better coverage than uniform sampling. Gray code is a form of binary that uses a different encoding method to increment from one number to another, making the encoding less prone to errors [22]. Similar to Halton, variants have been researched to render Sobol less linearly correlated [8].

3.3 Enforcing separability

As we mentioned earlier, evolutionary algorithms can be improved by applying a process called niching. Niching aims to maintain diversity by leading individuals to explore different regions of the search space [78]. While we do not employ niching to extract a multitude of solutions from a multimodal problem, we explore the concept of niching through the use of Support Vector Machines (SVM) to enforce separability of the subpopulations. We



(a) Subpopulation centroids and their SVM region (b) Points sampled before correction (c) Points sampled after correction

Figure 4: Forced separation of subpopulations through the use of SVM regions

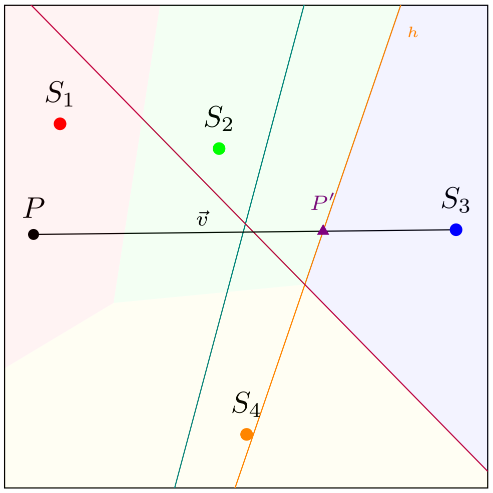
coin the term *centroid* to represent a subpopulation's initialization point, or its mean m^g if the algorithm is running. Using the subpopulations' centroid, we can linearly separate each subpopulation and restrict future sampled individuals to their respective regions. The Support Vector Machine is initially trained on the initialization points. The SVM is then retrained from the new centroid/mean of the sampled individuals. We define *SVM regions* as the decision regions split from the decision boundaries of the SVM.
Due to the randomness of CMA-ES, individuals can be sampled outside of the dedicated SVM region. Sampling those individuals until they fit the SVM bounds can lead the algorithm to be stuck in a loop, which hinders the algorithm's performance. An overview of this process is shown in Algorithm 1. Instead, the individuals can be corrected back to their dedicated SVM region. Figure 4 shows the process of assigning SVM regions to each subpopulation and correcting out-of-bounds samples. Correction is only applied to out-of-bounds points, and solutions are only evaluated after they have been corrected by SVM correction and *saturate* correction. We recognize a drawback of this method being that the algorithm can sample good points outside of their allocated SVM region, resulting in losing these points and potentially missing out on great solutions. We focus on the potential benefits of restricting sampling to SVM regions. We discuss two methods that were investigated to correct points.

Algorithm 1: SVM correction

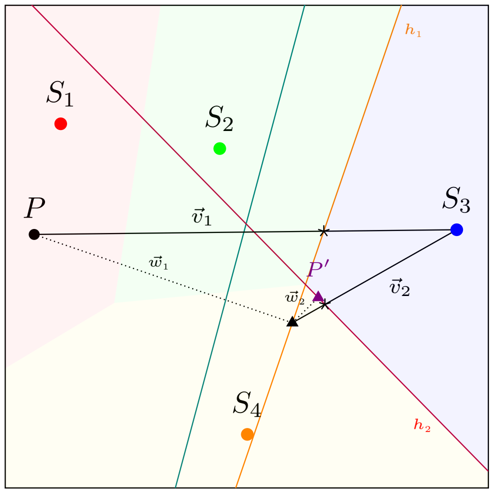
```
Data: $X$ = sampled individuals, $coefs$ = decision boundaries, $t_{subpop}$ =
      subpop_target, $C$ = centroid
Result: $X$ corrected individuals
while any $x \in X$ not in $t_{subpop}$ do
    for $h \in coefs$ do
        intersect $\leftarrow$ solve linearly for $h$ and $x$
    end
    $I \leftarrow$ closest intersect to $C$ on vector $\vec{v}$ using Euclidean distance
    if $mode = orthogonal$ then
        $\vec{w} \leftarrow$ vector from $I$ to $x$
        $x_{proj} \leftarrow h * \frac{-h \cdot \vec{w}}{h \cdot h} + x$
    end
    $x \leftarrow x_{proj}$
end
```

3.3.1 Correction at the Intersection with a decision boundary

With this method, points are corrected back to the SVM region by finding the intercept between the hyperplane and the vector of the point and the centroid. We take h the hyperplane that separates two points; point A the centroid of the supposed SVM region, and point B the centroid of the SVM region in which sampled point P is incorrectly located. The hyperplane is the decision boundary that separates the two regions. We define a vector $\vec{v}$ as the vector leaving from point A and point P. Finally, we find the intercept between the vector $\vec{v}$ and hyperplane h. This intercept is now the corrected individual.



(a) Intersection Correction



(b) Orthogonal Projection Correction

Figure 5: Support Vector Machine correction. Point P is supposed to be in the **blue** region, but was sampled out of bounds in the **red** region. Both methods correct the point back to P_{corr}. For orthogonal projection, the process must be applied twice to bring the point to the right region.

Consider the extreme scenario of Figure 5a. In this scenario, we have 4 different subpopulations. Let's assume that point P should be in the **blue** region, but was sampled out of bounds in the **red** region, close to centroid S_1.
We begin by identifying vector $\vec{v}$ starting from centroid S_3 and ending at new point P. Next, we need to extract the hyperplanes bounding centroid S_3. n number of subpopulations results in $\sum_{i=1}^{n-1} 1$ total of decision boundaries. Extracting the decision boundary only bounding S_3, we are left with $n-1$ decision boundaries.
Using linear algebra, we can find the intersection of a hyperplane and a vector. We transform vector $\vec{v}$ to equation form, solving for each variable. We can then solve for t with the hyperplane, and use the t value to find x, y and z. This operation is broken down below:

$$\begin{gathered}
h \rightarrow a\mathbf{x} + b\mathbf{y} + c\mathbf{z} = d \qquad \vec{v} = \langle i, j, k \rangle \qquad P = (e, f, g) \\
\mathbf{x} = i * \mathbf{t} + e \\
\mathbf{y} = j * \mathbf{t} + f \\
\mathbf{z} = k * \mathbf{t} + g \\
\mathbf{t} = a * (i * t + e) + b * (j * \mathbf{t} + f) + c * (k * \mathbf{t} + g) - d
\end{gathered} \tag{9}$$

Since we can have multiple decision boundaries/hyperplanes intersecting with the vector, we need to check all intersections and extract the one closest to the SVM region's centroid. This point is considered the corrected new point P'.

3.3.2 Correction using Orthogonal Projection onto a decision boundary

This method extends the intersection correction with a few more additional steps. Instead of correcting the point to the intersection of the vector and the hyperplane, the point is corrected to the orthogonal projection of the point onto the hyperplane. Let the intersection between the vector $\vec{v}$ and the hyperplane be point I, we establish vector $\vec{w}$ starting from intersection I and ending at new point P. We then compute the projection of point P through vector $\vec{w}$ onto hyperplane h using the following operation:

$$P_{proj} = h * \frac{-h \cdot \vec{w}}{h \cdot h} + P \tag{10}$$

The projection P_{proj} may get corrected outside the dedicated region, in which case the process is repeated until the point is accurately corrected (see Figure 5b).

3.4 Population Size Adaptation

Additionally, we consider the method at the core of the IPOP-CMA-ES algorithm [6]. IPOP-CMA-ES increases population size at local restarts. We apply this process of increasing the population size to the concept of subpopulations. After a user-defined t iterations, we order the subpopulations in order of overall fitness progress, meaning the value by which the fitness of the subpopulation has improved. That best subpopulation's population size is then increased by a user-defined r rate. After trial and error, we found that increasing the population size by 10% and 20% every 500 or 1000 iterations provided the algorithm with a good balance.
We also decided to investigate the effect of maintaining the overall individuals count the same at restarts, opting for resource re-allocation rather than increased population size. For this method, the subpopulation with the lowest increase in fitness of a t period is given a population decrease identical to the increase provided to the best-performing subpopulation. A subpopulation's size can be decreased up to 2, to ensure that this subpopulation can still evaluate. Furthermore, as the subpopulation's population size is modified by a percentage, we ensure that the total added and removed is even and thus not lower than 2. This is most prominent when dealing with subpopulations of smaller population sizes. This process is broken down in Algorithm 2.

```
Algorithm 2: Population Size Adaptation
  Data: $t$ = iteration period, $r$ = increment rate, $S$ = subpopulations
  Result: $X$ corrected individuals
  if $t$ time for restart then
      $S_{sorted} \leftarrow$ sort subpopulations by fitness improvement over $t$ iterations
      $S_{best} \leftarrow$ best performing subpopulation
      $S_{worst} \leftarrow$ worst performing subpopulation
      increment = $\texttt{max} \max(2 * \lfloor(\lambda_{S_{worst}} * r) \div 2\rfloor, 2)$
      $S_{best} \leftarrow$ update $(\lambda_{S_{best}} +$ increment)
      if $mode = ipop$ then
          $S_{worst} \leftarrow$ (update $\lambda_{S_{worst}} -$ increment)
      end
  end
```

4 Experiments

In the following section, we provide a detailed explanation of the experiments and the research questions they aim to solve (Table 2 provides a summary of the experiments run and the research question they address). The results are plotted and analyzed for further comprehension.

Unless stated otherwise, the configurations use default parameters described in Table 4. In our experiments, the search space of each problem was bounded between -5 and 5, so as to help the optimizer avoid infeasible solutions [84]. The initialization centroid of subpopulations are sampled in the same range of -5 and 5. We used *saturate* correction [14], which set out-of-bounds coordinates to the boundary.

Experiments 1 (see Section 4.2), 2 (see Section 4.3), 3 (see Section 4.4) and 4 (see Section 4.5) were run on on a AMD Ryzen 5 3600 CPU with 16GB of RAM. Experiment 5 (see Section 4.6) was run on the LIACS mithril server, which has 64 Intel Xeon E5-4667v3 CPUs and 1TB of RAM.

4.1 Metrics

Experiments are evaluated using two performance indicators: Fixed-Target results and Fixed-Budget results. The Fixed-Target results provide important information on the number of function evaluations required to achieve a certain target value, which is taken from the real values of the objective function. The Fixed-Budget results provide a quantified metric on the quality of an algorithm at a given budget. Additionally, we look at the Empirical Cumulative Distribution function [20] (Equation 11) for analyzing the performance of configurations over all functions. The cumulative distribution estimates the proportion of runs that satisfy the algorithms finding the best solution within the given budget:

$$\widehat{F}_T(t; A, f, d, v) = \sum_{i=1}^{r} (T(A, f, d, v, i) \leq t) \tag{11}$$

where $T(A, f, d, v, i)$ is the fixed target result. It denotes the number of function evaluation needed for algorithm A to reach, in its i-th run, for the f problem variant in d dimensions, a solution that satisfies $f(x) \geq v$, within a budget B.

Table 2: Summary of Experiments and Parameter Settings

Experiment	**Research Question**	**Configuration parameters**
Default Parameters		Table 4
1 (see Section 4.2)	RQ1 (see Section 1)	Table 5
2 (see Section 4.3)	RQ2 (see Section 1)	Table 6
3 (see Section 4.4)	RQ3 (see Section 1)	Table 7
4 (see Section 4.5)	RQ4 (see Section 1)	Table 8
5 (see Section 4.6)	Extended benchmark of promising configurations	Table 9

IOH also offers the aggregation of ECDF over a set of target values $\mathcal{V}$ (Equation 12), and/or over a set of functions $\mathcal{F}$ (Equation 13):

$$\widehat{F}_T(t; A, f, d, \mathcal{V}) = \frac{1}{r|\mathcal{V}|} \sum_{v\in\mathcal{V}} \sum_{i=1}^{r} (T(A, f, d, v, i) \leq t) \tag{12}$$

$$\widehat{F}_T(t; A, \mathcal{F}, d, \mathcal{V}) = \frac{1}{r|\mathcal{V}||\mathcal{F}|} \sum_{f\in\mathcal{F}} \sum_{v\in\mathcal{V}} \sum_{i=1}^{r} (T(A, f, d, v, i) \leq t) \tag{13}$$

4.2 Experiment 1 - Baseline Comparison

To establish a baseline result, we compare the performance of ModularCMA against a few different configurations of subpopulation CMA-ES. We establish two baseline performances of CMA-ES. The first baseline initializes the first population at the center of the search space, as Vermetten et al. [84] found that initializing at the center of the search space resulted in fewer infeasible solutions. The other baseline uniformly places the initialization centroid of the first population in the search space. In all the other configurations, the subpopulations' initialization points are uniformly distributed. We investigate the impact of using different initialization strategies for the first population in experiment 2.

We used four different subpopulation setups; `SP-CMA[50]`, `SP-CMA[20]`, `SP-CMA[10]` and `SP-CMA[MIXED]`, along with two configurations of `ModularCMAES` that do not employ subpopulations and are setup using the parameters of Table 4. `ModCMA-[0]` is initialized at the center of the domain, where `ModCMA` is initialized uniformly. The configurations are broken down in Table 5. Those configurations also serve as baseline configurations for experiments 2, 3 and 4. Each configuration is benchmarked on the BBOB test suite, using functions 3 and 4, ran for 15 different instances of each problem, for dimensions 2, 5, 20 and 40.

In the Fixed-Target results (Figures 6 and 7), configurations have relatively similar performances, other than `ModCMA[0]` which is able to find better results faster, but is eventually caught up by other configurations.
In the Fixed-Budget results (Figures 8 and 9), the impact of dimensionality on the performance of subpopulations is greater. In $D = 5$, we see that `SP-CMA-[10]` and `SP-CMA-[20]` are able to reach better results around 1000 function evaluations, with `SP-CMA-[10]` clearly reaching the best results. While `SP-CMA-[MIXED]` is not able to reach the same results as `SP-CMA-[10]` and `SP-CMA-[20]`, it outperforms the other configurations. The opposite is seen in $D = 20$, where `SP-CMA-[10]` is not able to reach the same results as configurations with bigger population sizes. `SP-CMA-[20]`, `SP-CMA-[50]`, `ModCMA` and `ModCMA-[0]` arrive to similar performances, with `SP-CMA-[50]` slightly outperforming the others.
The impact of dimensions is further demonstrated in the ECDF Figures 10a and 10b, where configurations using subpopulation, more specifically `SP-CMA-[10]` and `SP-CMA-[20]`, outperform the default CMA-ES configurations in lower dimensions. In $D = 20$ however, `SP-CMA-[10]` and `SP-CMA-[20]` are unable to reach the target value as often as the other configurations, leaving `SP-CMA-[50]` with a slightly better ratio.

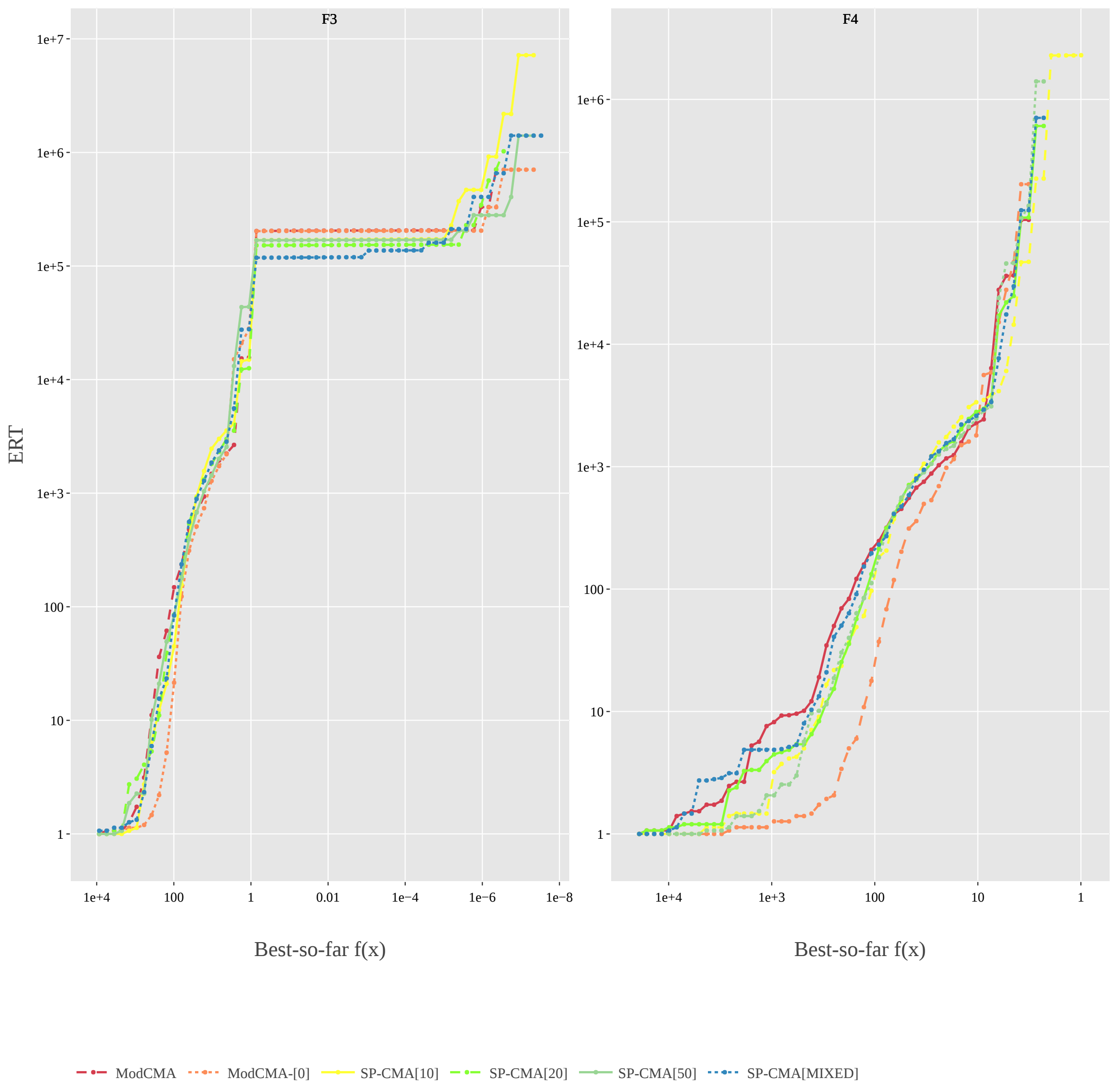


Figure 6: Experiment 1 Fixed-Target Results on F3 and F4 in 5 dimensions

From the experiment results, we understand that subpopulations help to improve CMA-ES. We posit that this is due to the ability of subpopulations to cover a wider area of the domain. As seen with low dimensions results, the higher the number of subpopulations, the better the results. But although subpopulations improve CMA-ES in lower dimensions, they hinder the performance in higher dimensions, suffering from the curse of dimensionality.

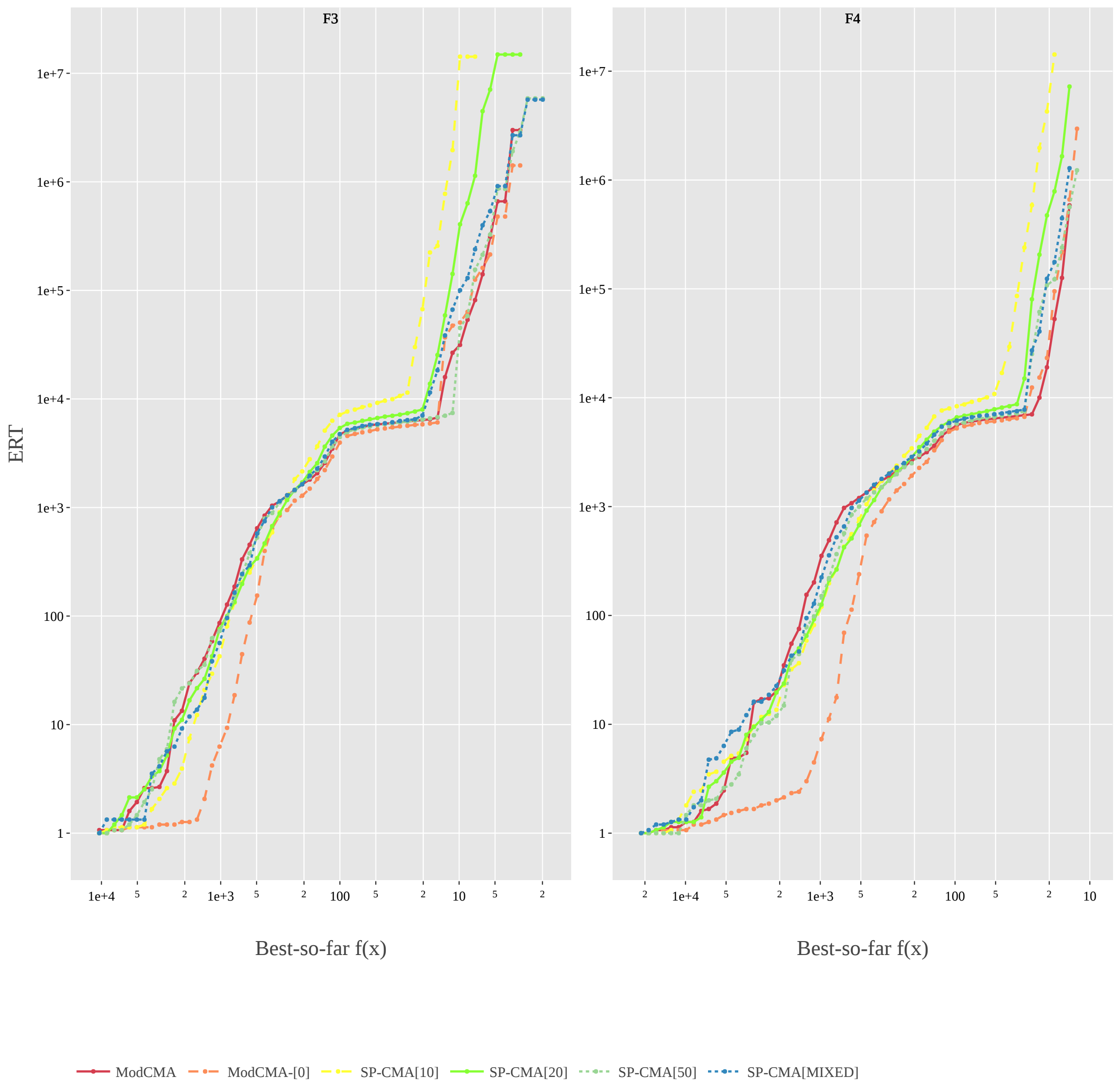


Figure 7: Experiment 1 Fixed-Target Results on F3 and F4 in 20 dimensions

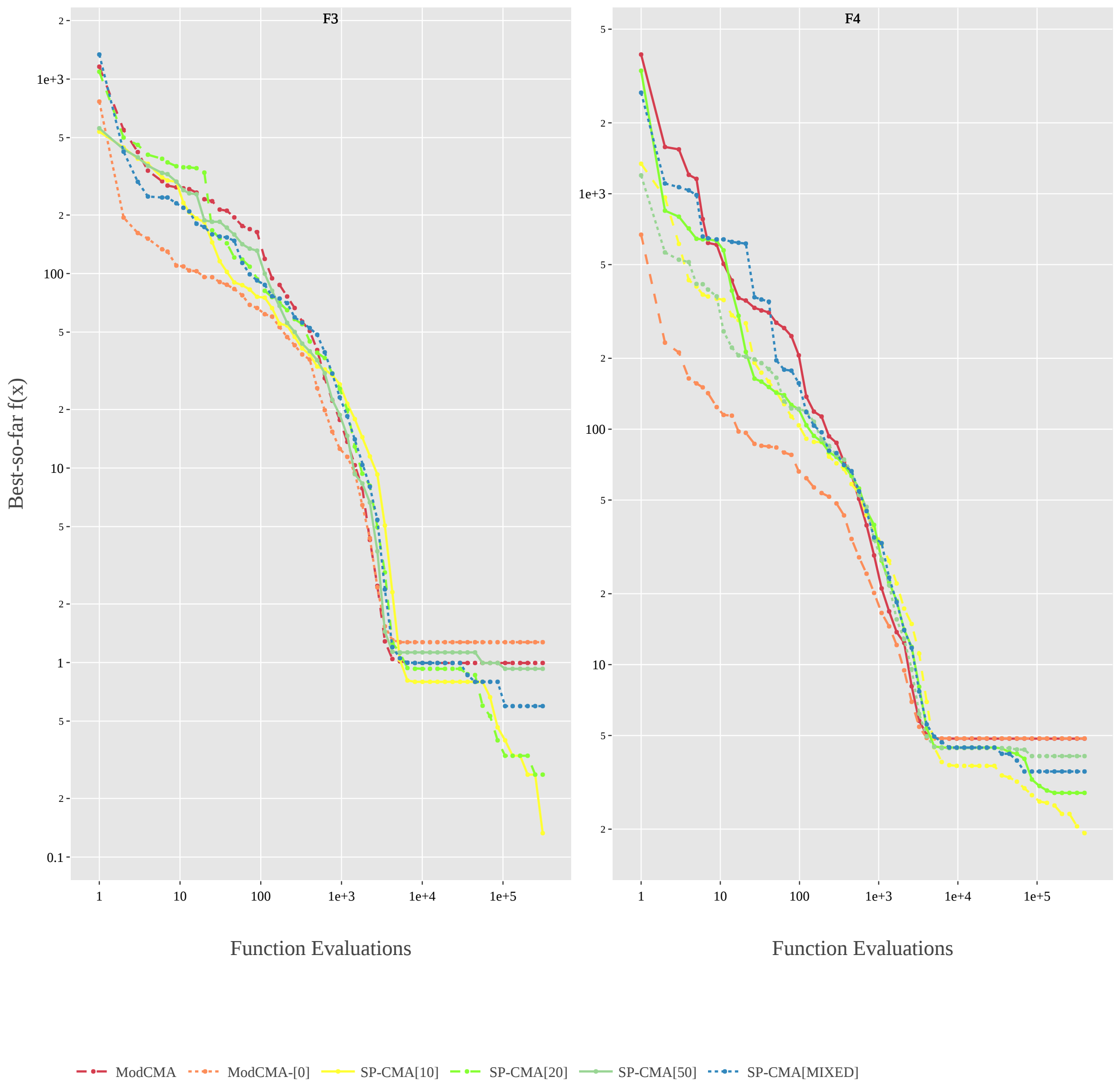


Figure 8: Experiment 1 Fixed-Budget Results on F3 and F4 in 5 dimensions

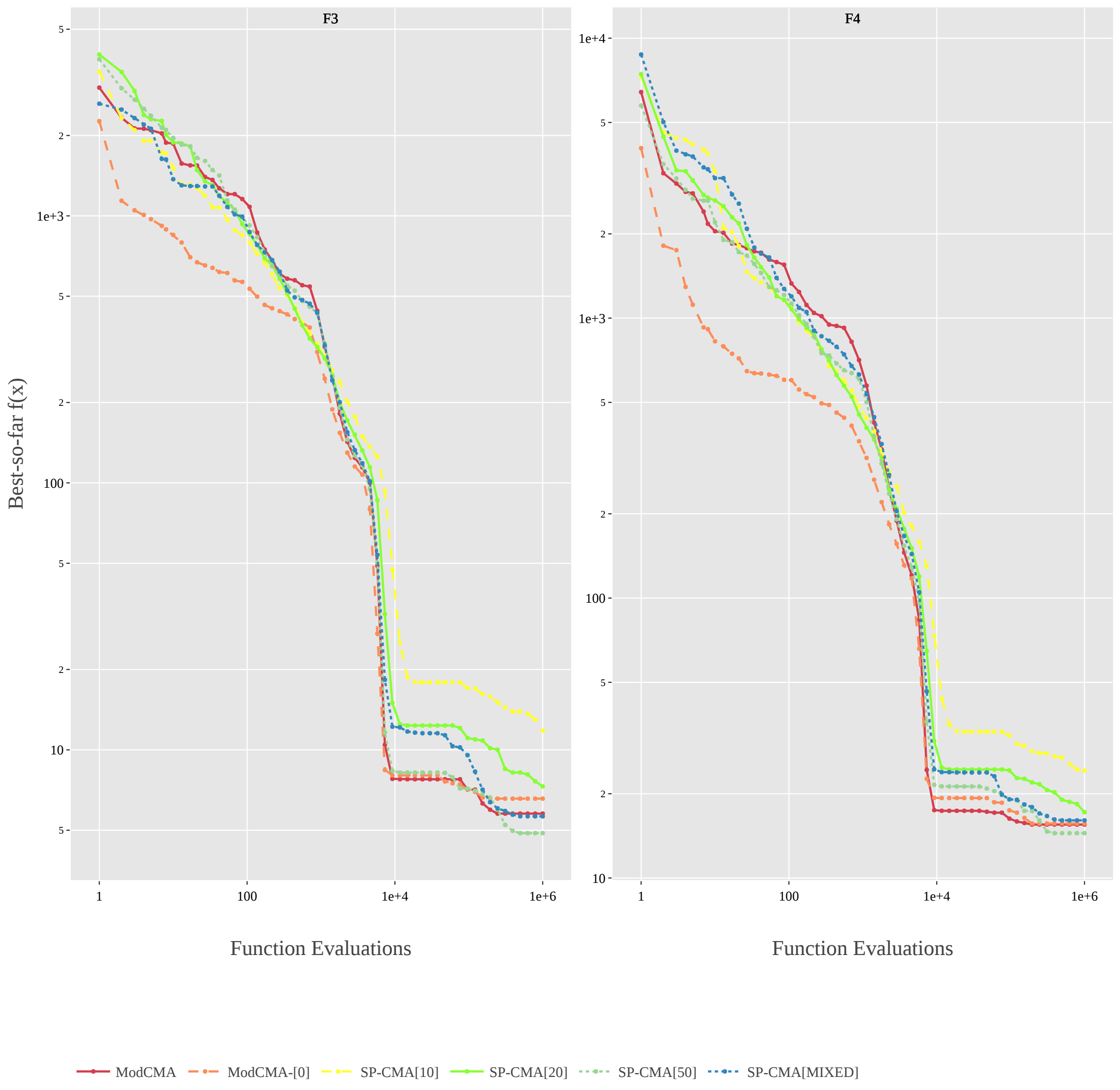
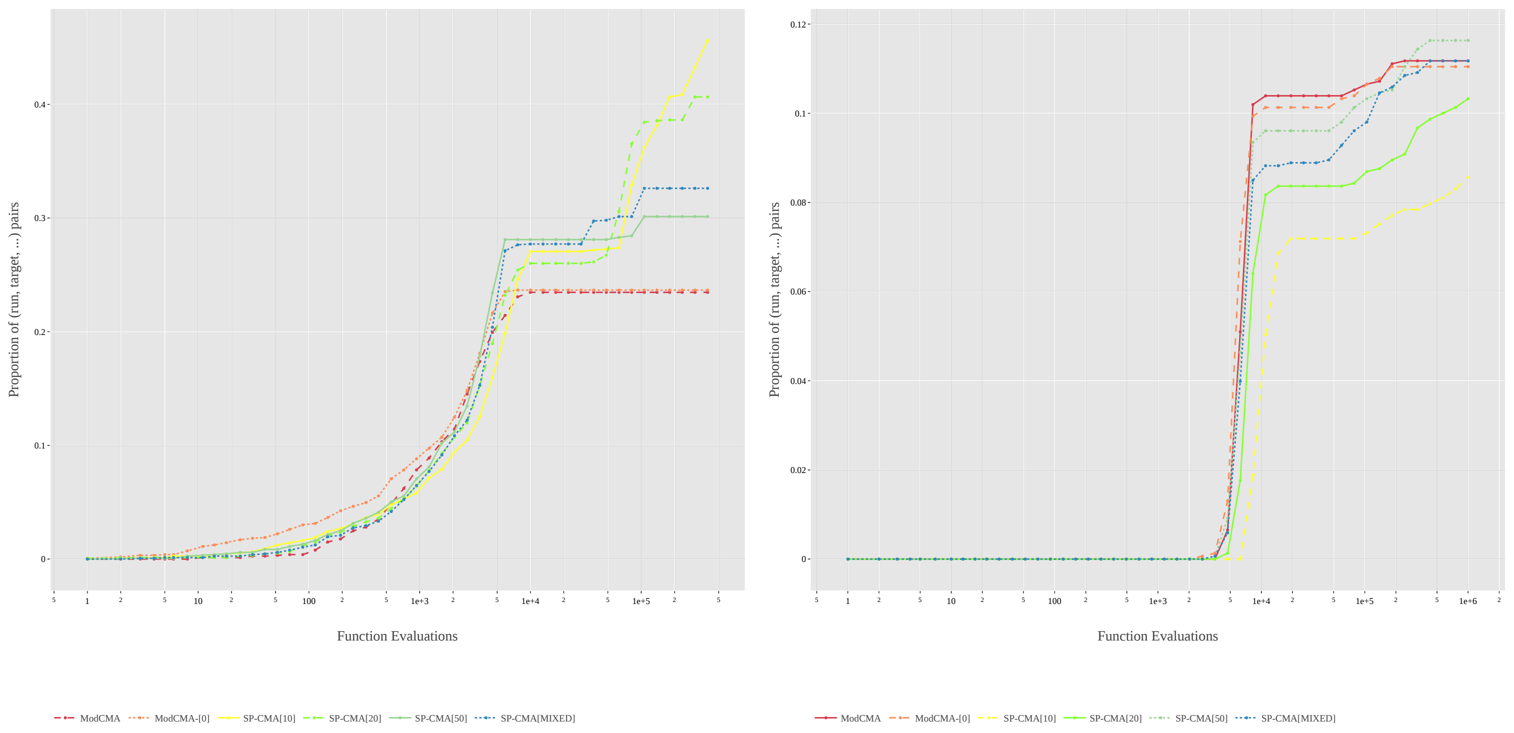


Figure 9: Experiment 1 Fixed-Budget Results on F3 and F4 in 20 dimensions



(a) $D = 5$ (b) $D = 20$

Figure 10: Aggregated Empirical Cumulative Distribution of Experiment 1 over all functions

4.3 Experiment 2 - Initialization Method

Following the results of experiment 1, we investigate the effects of different initialization methods on the performance of subpopulations CMA-ES (see Table 6 for run configurations). Borrowing from experiment 1, we use 3 different subpopulation setups; `SP-CMA[50]`, `SP-CMA[20]` and `SP-CMA[10]`, as those configurations showed the most potential in experiment 1. Each configuration is tested on six different initialization method; uniform, Gaussian, Poisson disk, Halton, Sobol and Latin Hypercube sampling. The configurations are tested on 15 instances of problems 3 and 4, for dimensions 5 and 20, and the results are reported in Figures 11, 12, 13, 14. Note that Poisson Disk sampling was not tested for dimension 20 as the sampling algorithm cannot scale to this dimension. Poisson disk sampling falls victim to the curse of dimensionality and cannot satisfy the minimum distance between points as those distances become extremely large in high dimensions [12].

Looking at the results of experiment 2 (Figures 11, 12, 13, 14, 15a, 15b), we observe a similar impact of dimension on the performance of configurations, although not as great as experiment 1.
Looking at the Fixed-Target results, there does not seem to be a clear best initialization method outperforming the others, other than for $D = 20$, where configurations using Gaussian sampling were able to improve the fitness faster than the others.
In the Fixed-Budget results, we see the same trend as experiment 1, with configurations using lower population sizes outperforming others. We also find that some initialization methods help configurations improve their performance. For example, in function 3, Halton and Latin Hypercube improves `SP-CMA[10]`, or Halton and Gaussian for `SP-CMA[20]`.
We can observe that the initialization methods did not have a great impact on the performance of a configuration as is prominent in Figure 14. Figure 14 shows 3 different groupings of results. We see that the initialization methods did not have an impact on the performance of a subpopulation over the others, as they are all grouped by subpopulations count. We see from the ECDF Figure15a that in lower dimensions, the initialization method has a greater impact on smaller subpopulations. We posit that initialization methods are typically used for sampling a greater number of points than were used here (2, 5 or 10), which could render initialization techniques less effective.

We can gather from the experiment results that the initialization method of subpopulations' initialization point has little to no effect on the performance of CMA-ES. We posit that this is due to the number of points that need to be generated. Since the highest number of subpopulations present only goes as high as 10, the algorithm cannot benefit from the advantages (or suffer from the disadvantages) of a specific initialization technique. Generated points seem closer to random distribution than their respective distribution pattern, which for certain techniques (namely Poisson, Halton, Sobol or LHS) renders the method pointless.

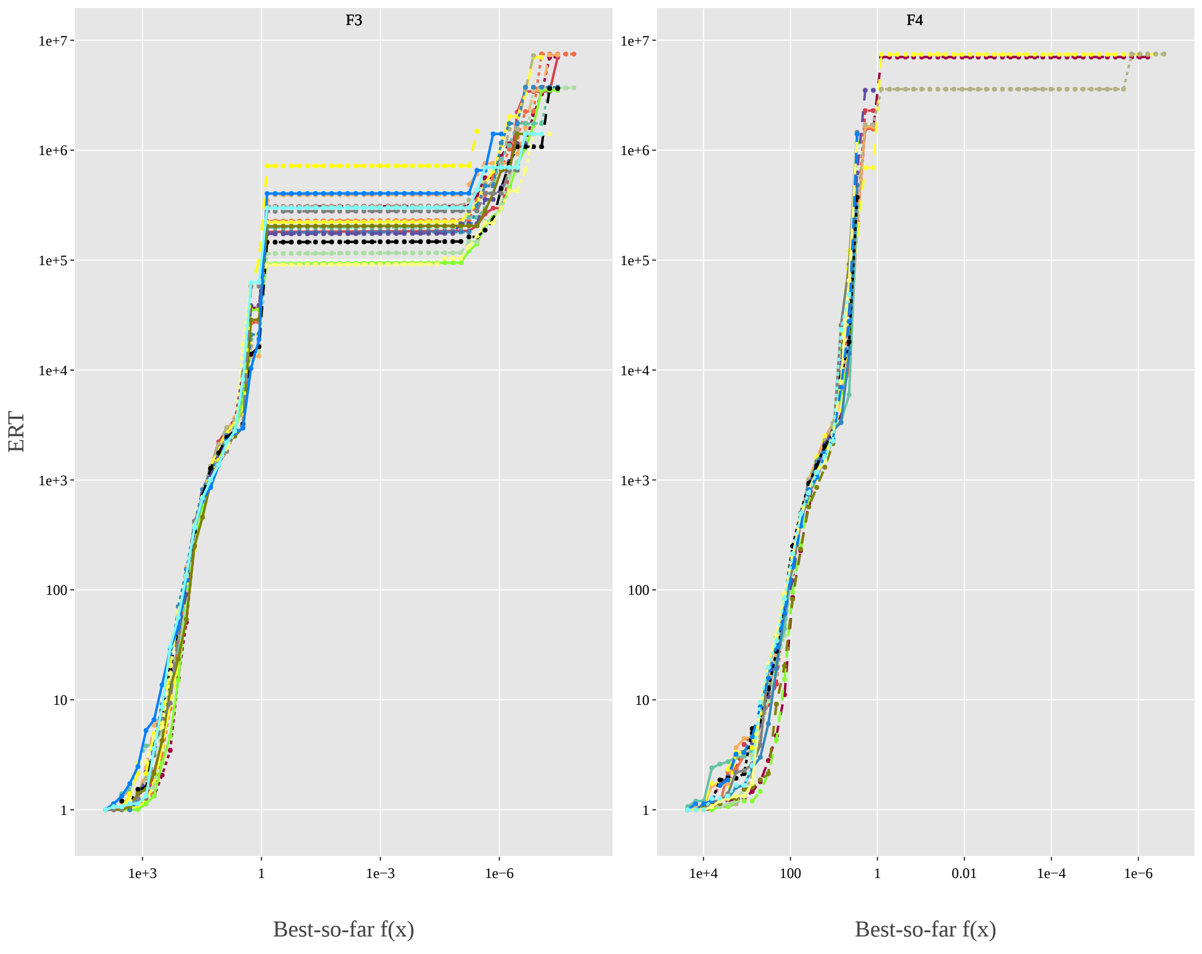


Figure 11: Experiment 2 Fixed-Target Results on F3 and F4 in 5 dimensions

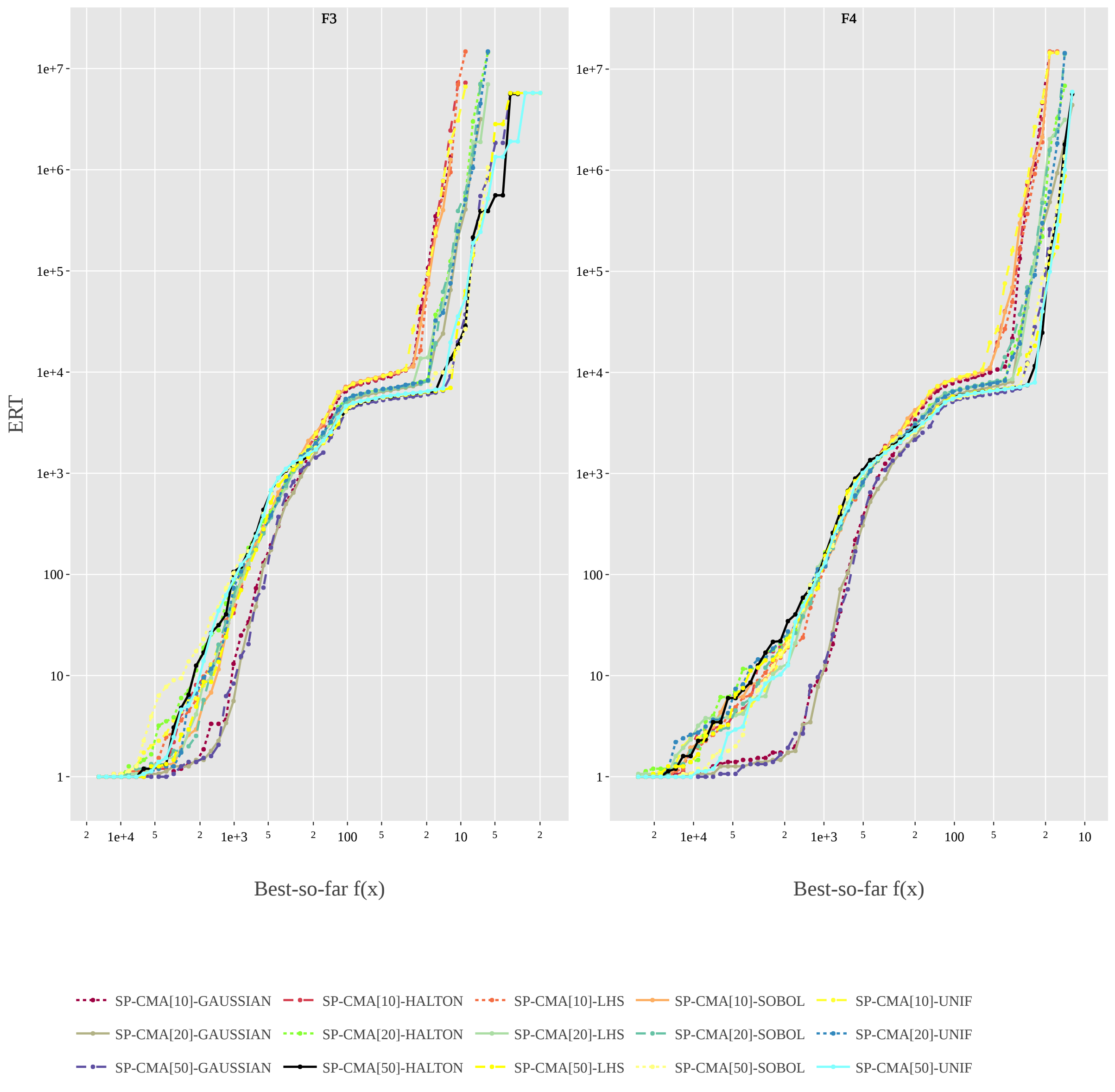


Figure 12: Experiment 2 Fixed-Target Results on F3 and F4 in 20 dimensions

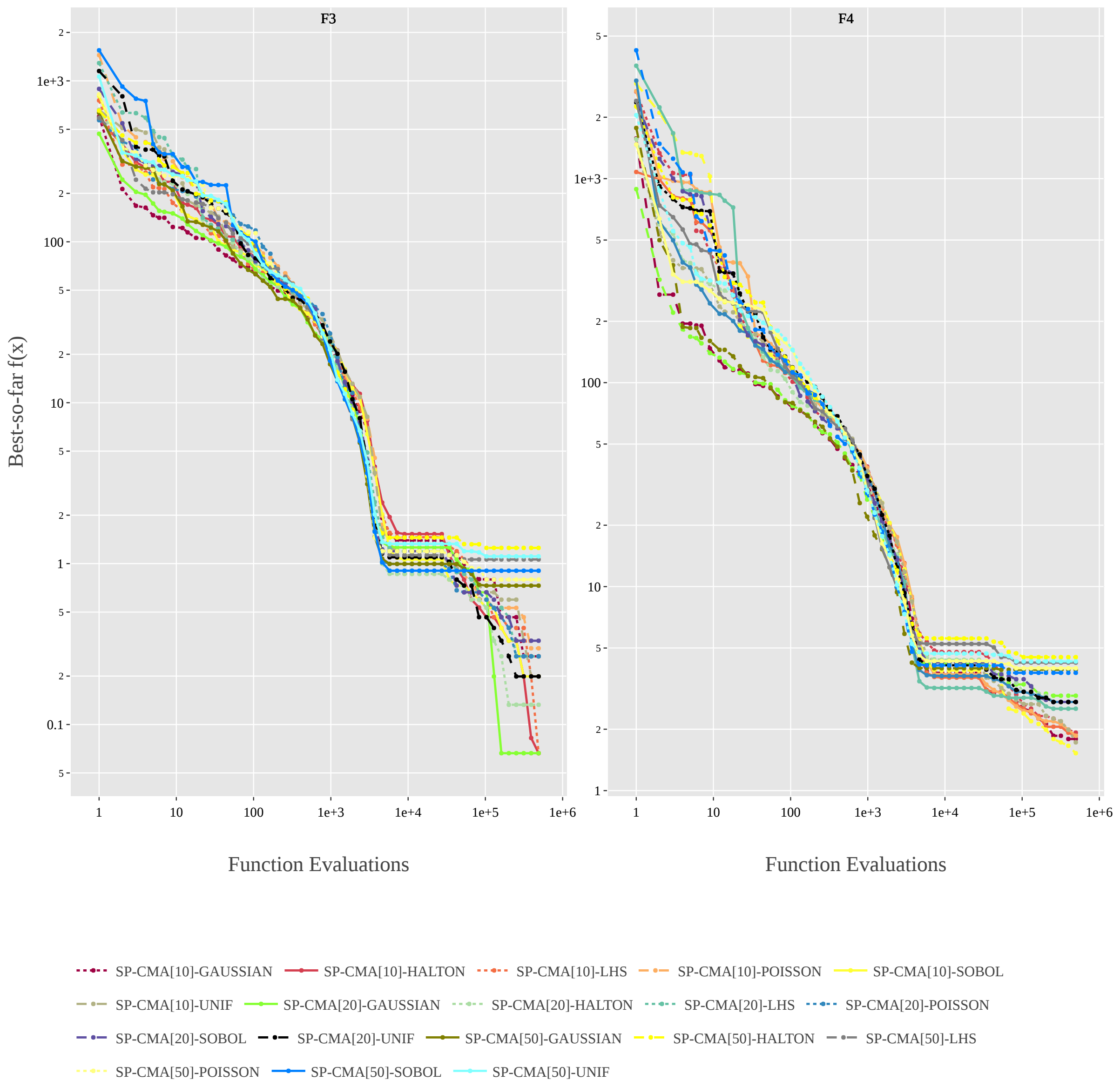


Figure 13: Experiment 2 Fixed-Budget Results on F3 and F4 in 5 dimensions

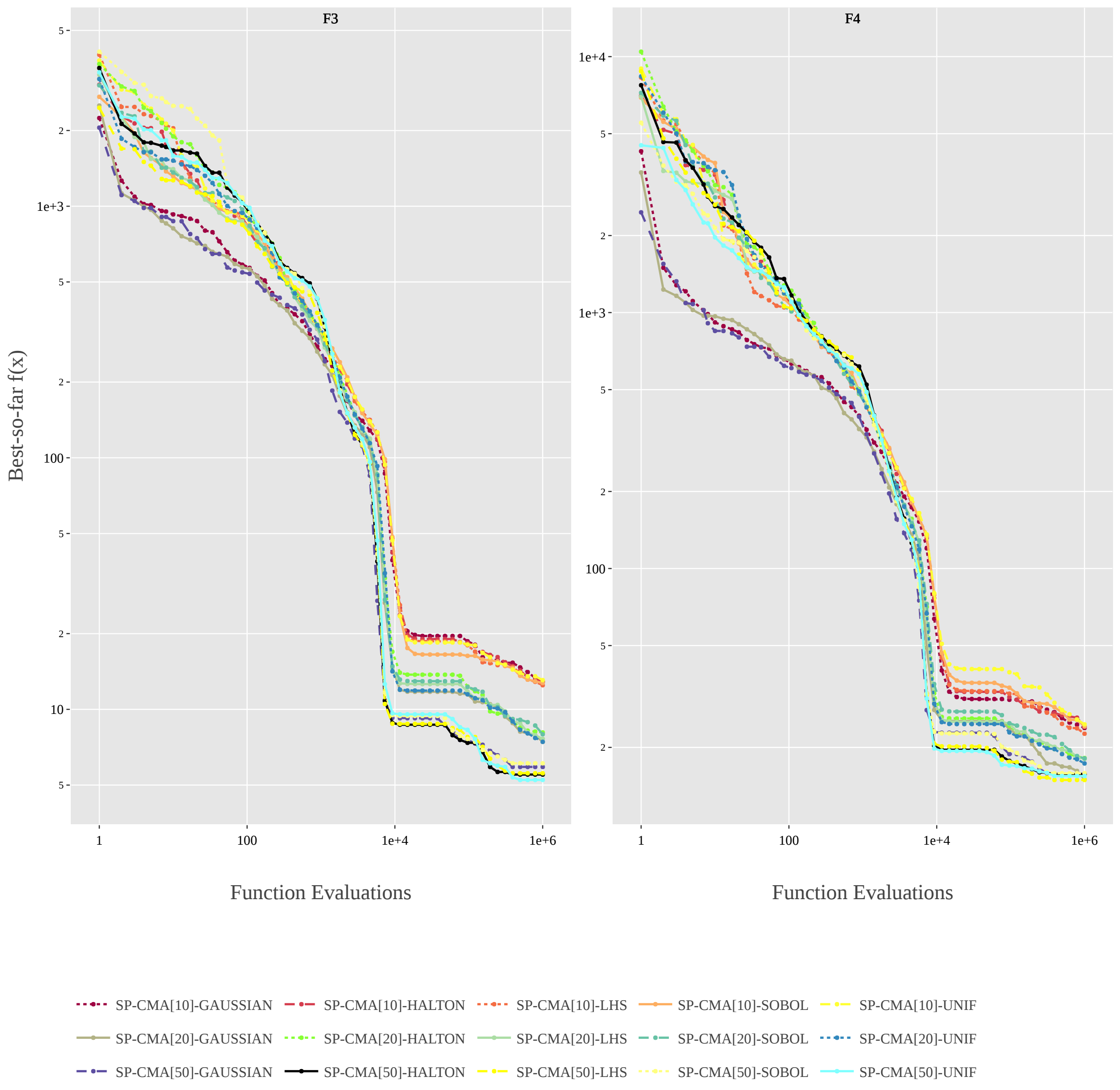
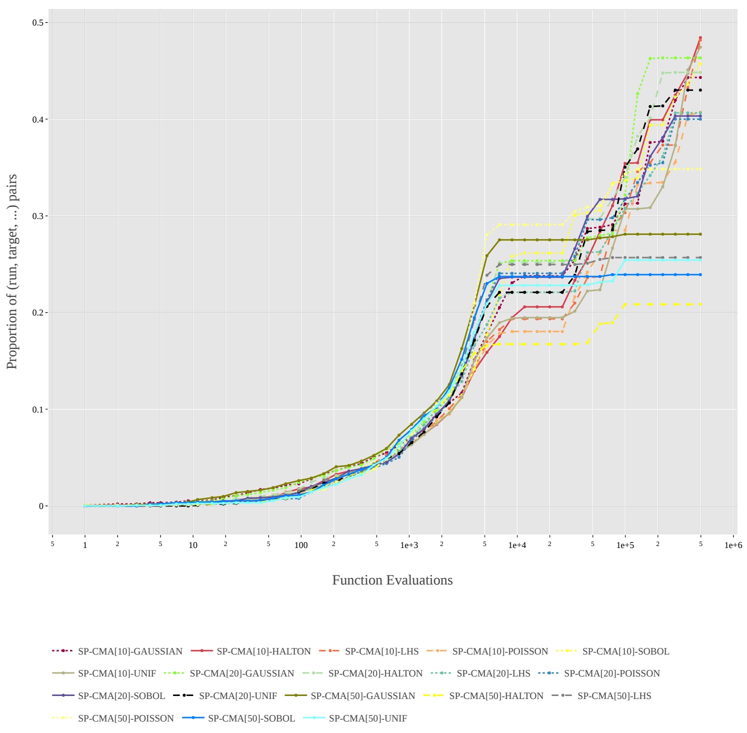


Figure 14: Experiment 2 Fixed-Budget Results on F3 and F4 in 20 dimensions


(a) $D = 5$

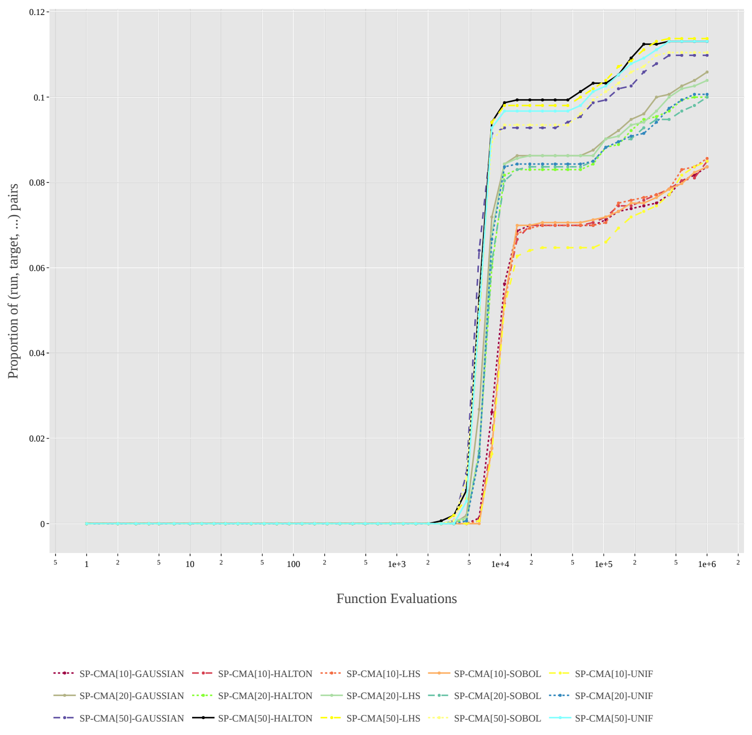

(b) $D = 20$

Figure 15: Aggregated Empirical Cumulative Distribution of Experiment 2 over all functions

4.4 Experiment 3 - Enforced Separation

We follow up experiment 2 by testing the enforced separation of subpopulations through Support Vector Machines (see Table 7 for run configurations). `SP-CMA[50]`, `SP-CMA[20]` and `SP-CMA[10]` were compared against configurations using Support Vector Machine corrections and against configurations not using any correction. We compare the performance of using intersection correction, orthogonal projection correction and no correction in Figures 16, 17, 18, 19 and 20. Furthermore, we combine the initialization techniques Latin Hypercube with Support Vector Machine corrections to evaluate the performance of combining both techniques. Configurations are tested on 15 instances of problems 3 and 4, for dimensions 5 and 20.

Looking at the results of Figures 16, 17, 18, 19 and 20, we can see that notice that forcing separation does not improve or lead to any improvement, but instead hinders the performance of subpopulations. Fixed-Target results show that while high subpopulation count configurations are faster at reaching good fitness, they are quickly caught up by lower subpopulations configurations. Combining `SP-CMA[50]` with orthogonal correction and Latin Hypercube sampling did result in better performance, especially for lower dimensions, and while its ECDF score is not far from the correction-free configuration, it does not improve the overall performance. Looking closely at function 3 of Fixed-Budget results, we note that SVM correction ends up hurting low dimensionality results of `SP-CMA[10]`.

We can gather from the experiment results that using forced separation through Support Vector Machine failed to improve CMA-ES, and overall hinders the capacity of CMA-ES to solve complex optimization problems. We posit that this is most likely due to a separation too strict, which prevents the algorithm from being able to explore areas of the domain with good solutions. The algorithm may also be losing out on good guesses made by the algorithm by correcting the points back to a worse location.

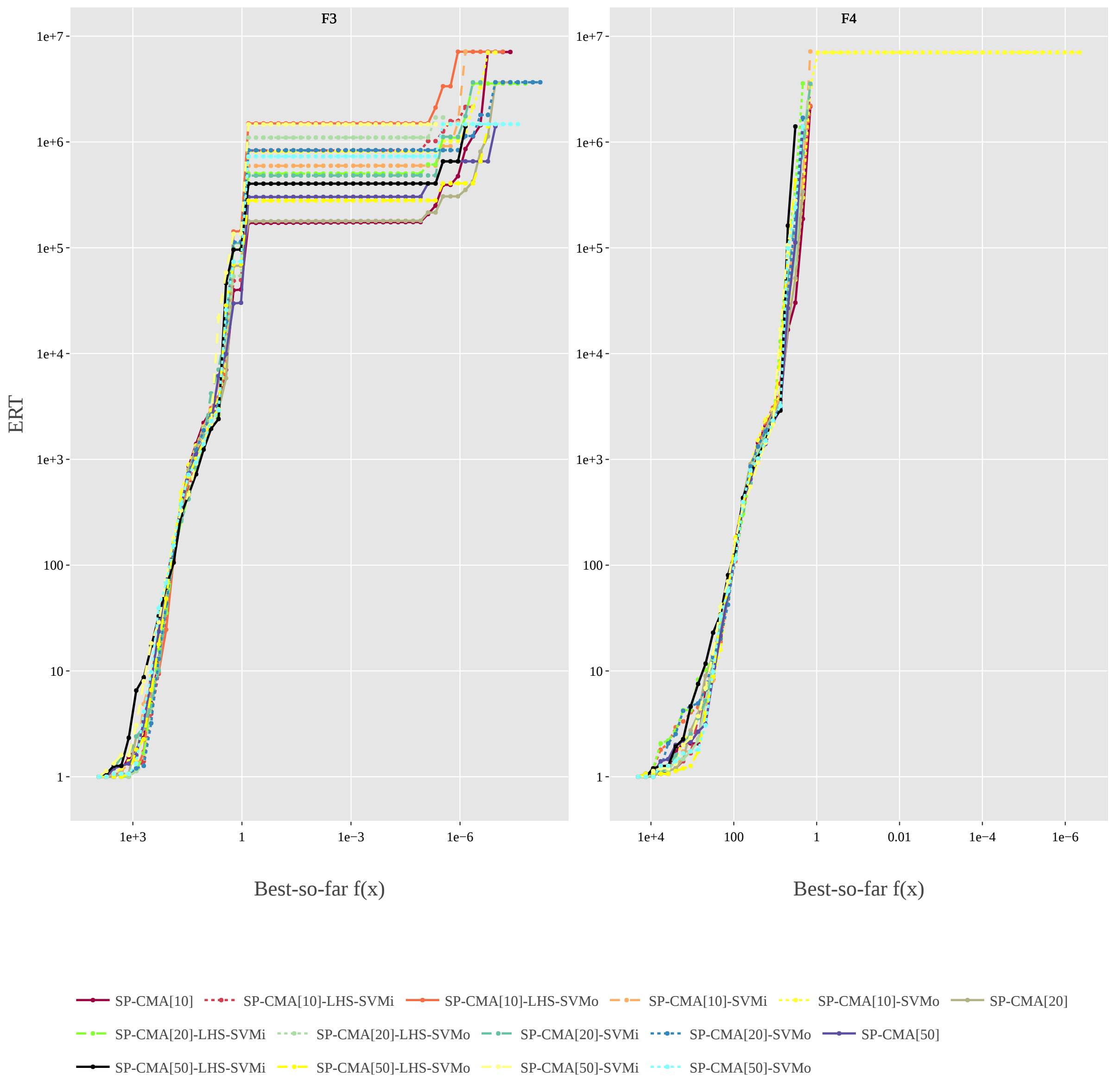


Figure 16: Experiment 3 Fixed-Target Results on F3 and F4 in 5 dimensions

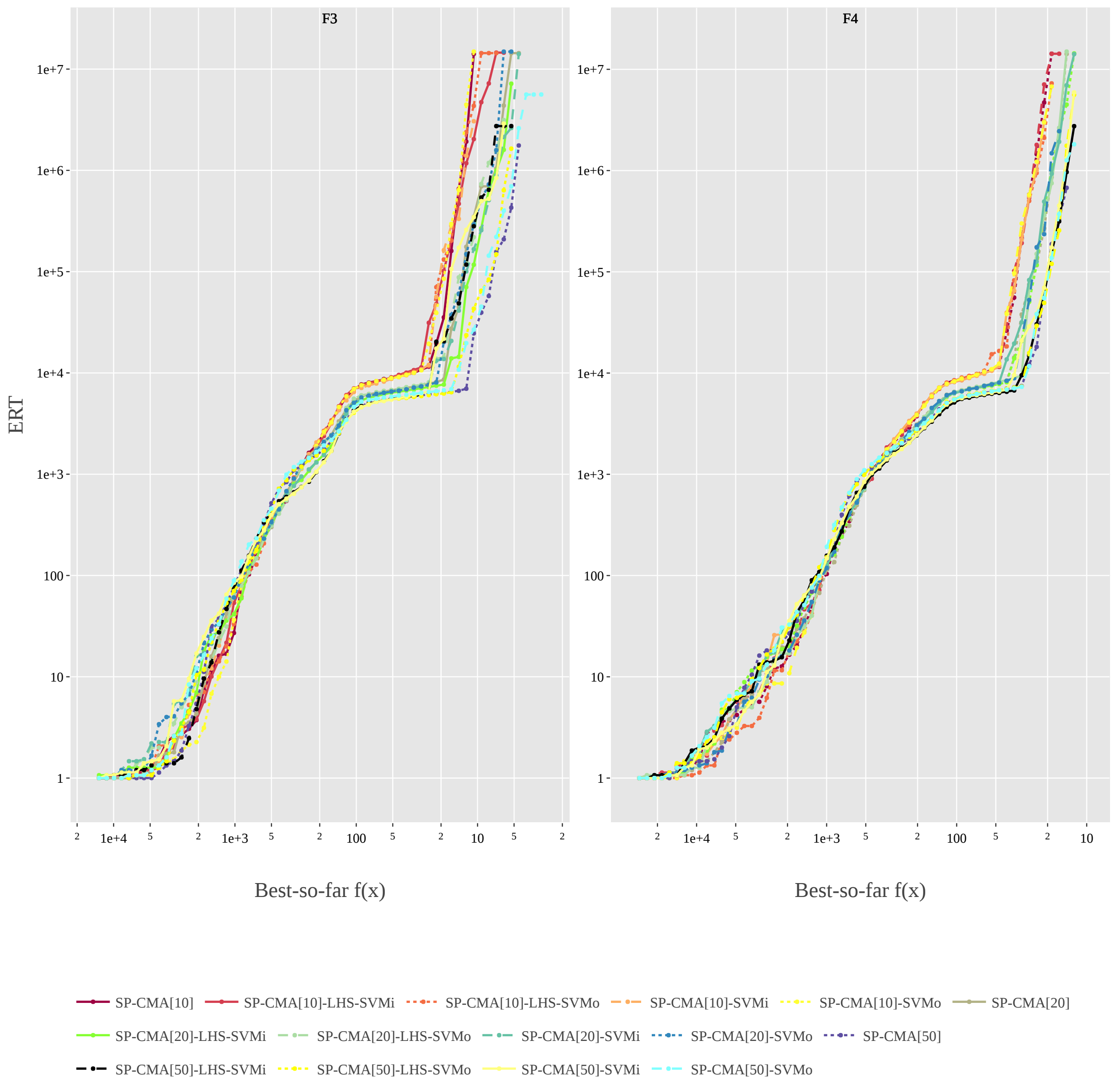


Figure 17: Experiment 3 Fixed-Target Results on F3 and F4 in 20 dimensions

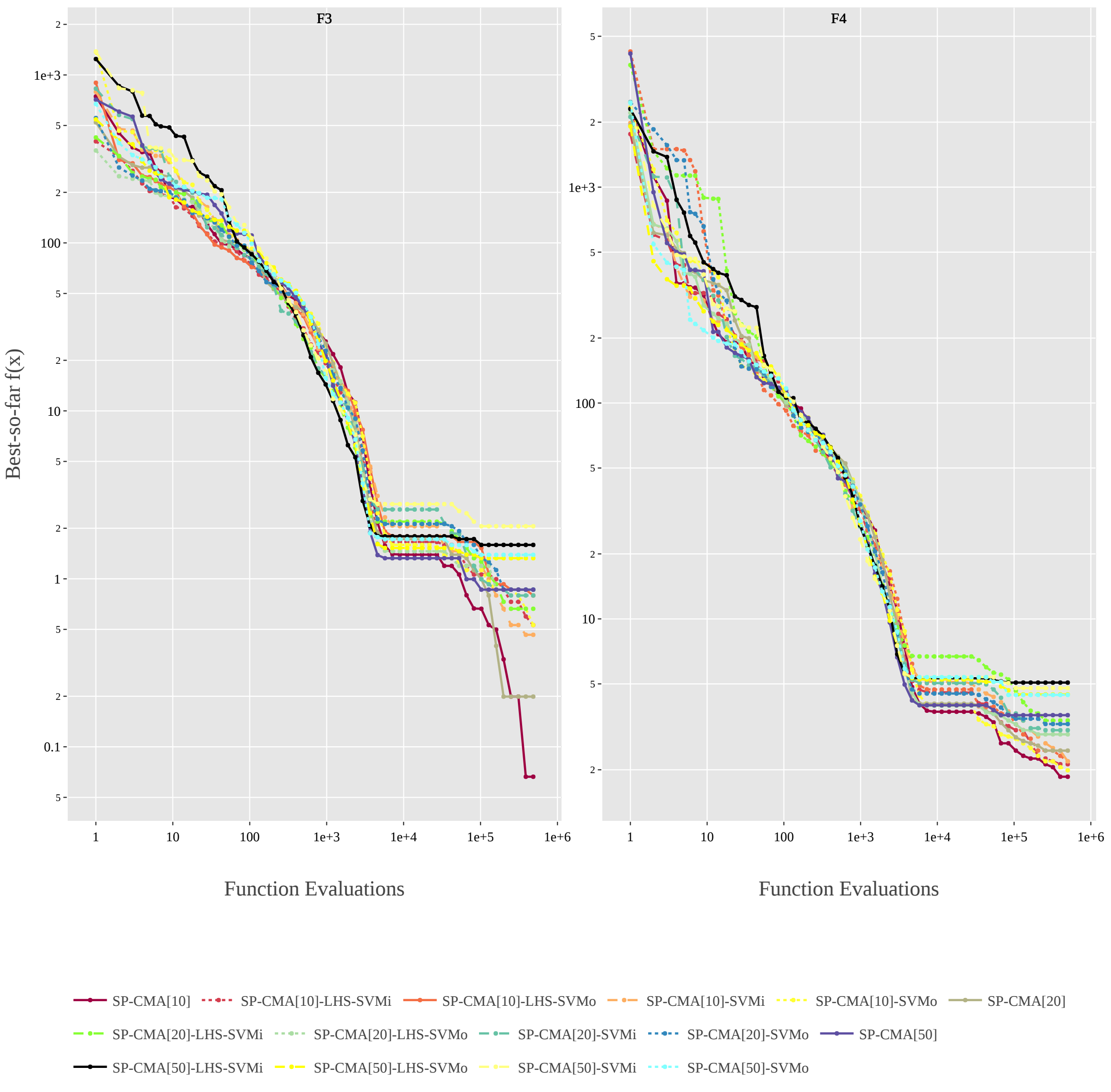


Figure 18: Experiment 3 Fixed-Budget Results on F3 and F4 in 5 dimensions

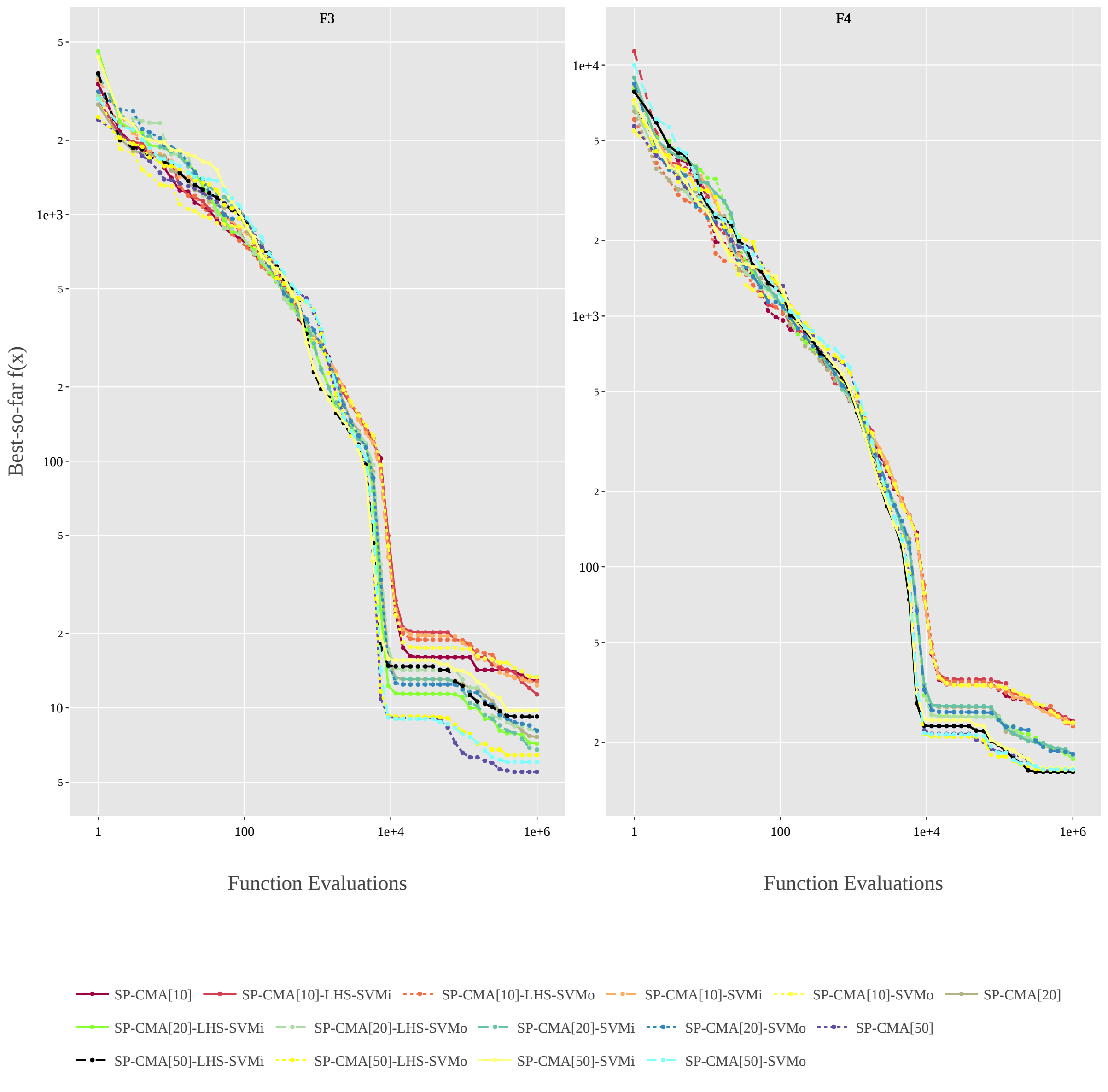
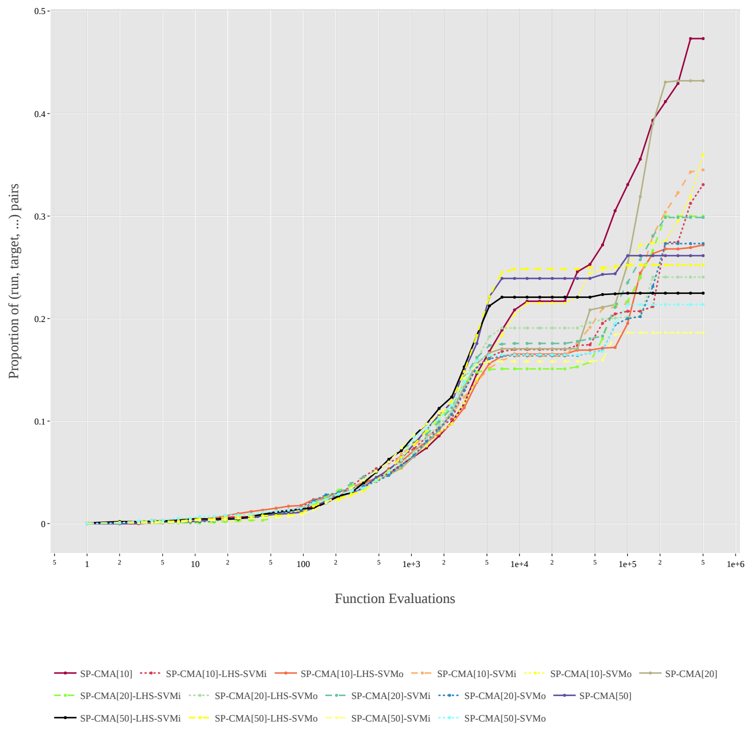
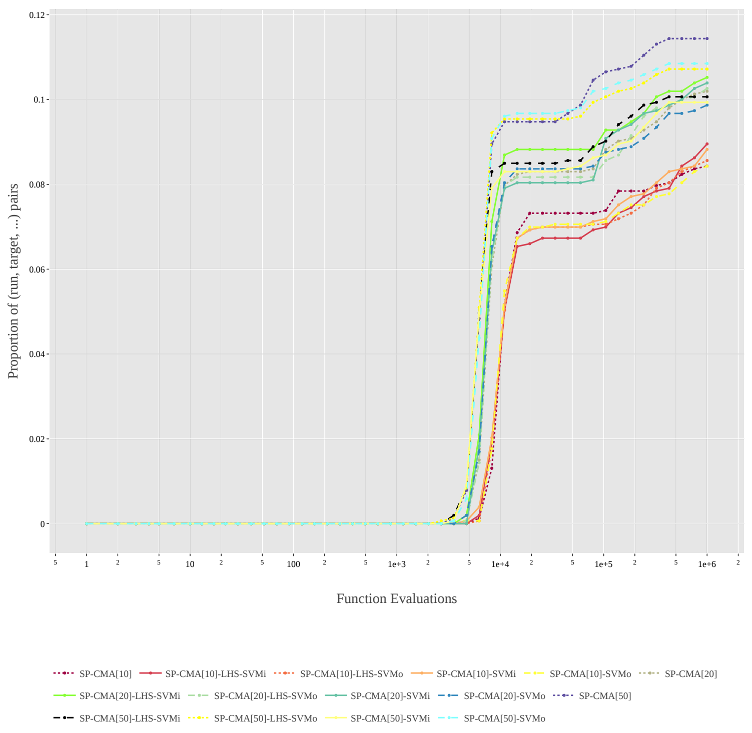


Figure 19: Experiment 3 Fixed-Budget Results on F3 and F4 in 20 dimensions



(a) $D = 5$



(b) $D = 20$

Figure 20: Aggregated Empirical Cumulative Distribution of Experiment 3 over all functions

4.5 Experiment 4 - Population Size Adaptation

In this experiment, we look at the performance of adapting the population sizes of well-performing subpopulations of the CMA-ES configurations. We test the 3 subpopulation setups `SP-CMA[50]`, `SP-CMA[20]` and `SP-CMA[10]`, and test t iteration period (i.e. the number of iterations between two restarts) [`500`, `1000`] and population change rate r [`0.1`, `0.2`]. Additionally, we compare the performance of using resource re-allocation against only increasing the population size (see Table 8 for run configurations). We denote configurations using resource re-allocation, meaning individuals added to a subpopulation will need to be removed from another, as `RA`, with configurations only increasing the best subpopulation's size denoted with `IPOP`. Configurations are tested on 15 instances of problems 3 and 4, for dimensions 5 and 20.

From Figures 21 and 22, we can see that using adaptive population size is not as greatly impacted by dimensionality as previous experiments. Looking at function 3, we can see that using size adaptation improved the ERT `of SP-CMA[20]` and `SP-CMA[50]` to beat their adaption-free counterparts. The Fixed-Budget (see Figures 23 and 24) and ECDF (see Figure 25) results further demonstrate that using size adaptation improves the performance of subpopulations, with the impact being greater in smaller dimensions. Fixed-Budget results (Figure 23) of function 3 show that, in lower dimensions, using a lower rate $r = 0.1$ with a high t delay outperforms `SP-CMA[10]`. In higher dimensions (Figure 24), adapting the population size also leads to some configurations with adaptive size control to improve from configurations without it (`SP-CMA[10]` or `SP-CMA[50]`). From the results, we could not find that population re-allocation nor population size increase was better than the other.

We can gather from the experiment results that adding adaptive population control to CMA-ES with subpopulation succeeded in improving the overall performance of the algorithm. Although it is difficult to find whether population size re-allocation (`RA`) or population size increase (`IPOP`) is better than the other, we prove that a population increase of well-performing subpopulations (whether that is from re-allocating from poorly performing subpopulations or simple population size increase), helps CMA-ES improve its performance for configurations that use subpopulations of lower initial population size. This enables the algorithm to start off evenly distributed and focus attention on promising areas.

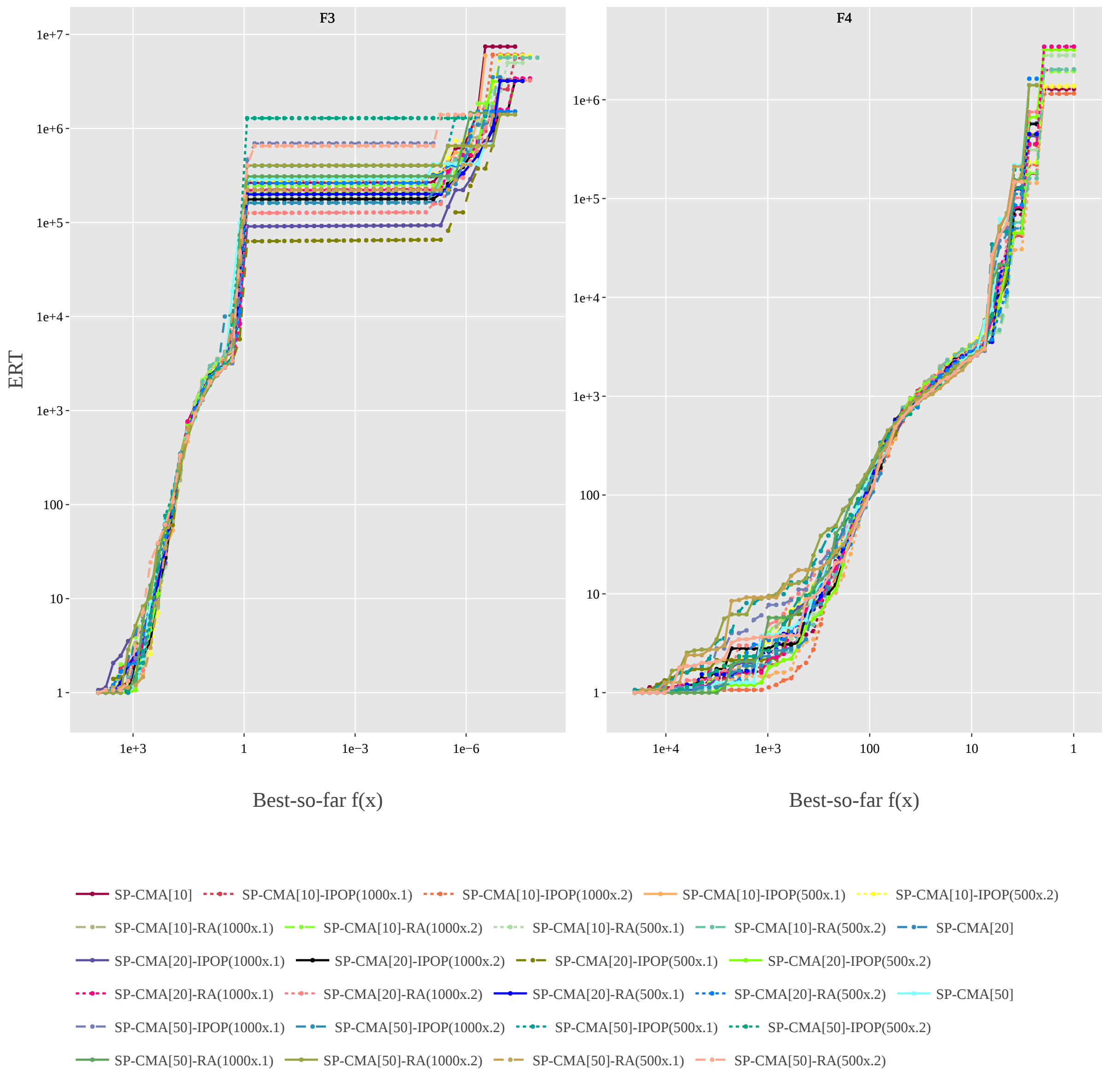


Figure 21: Experiment 4 Fixed-Target Results on F3 and F4 in 5 dimensions

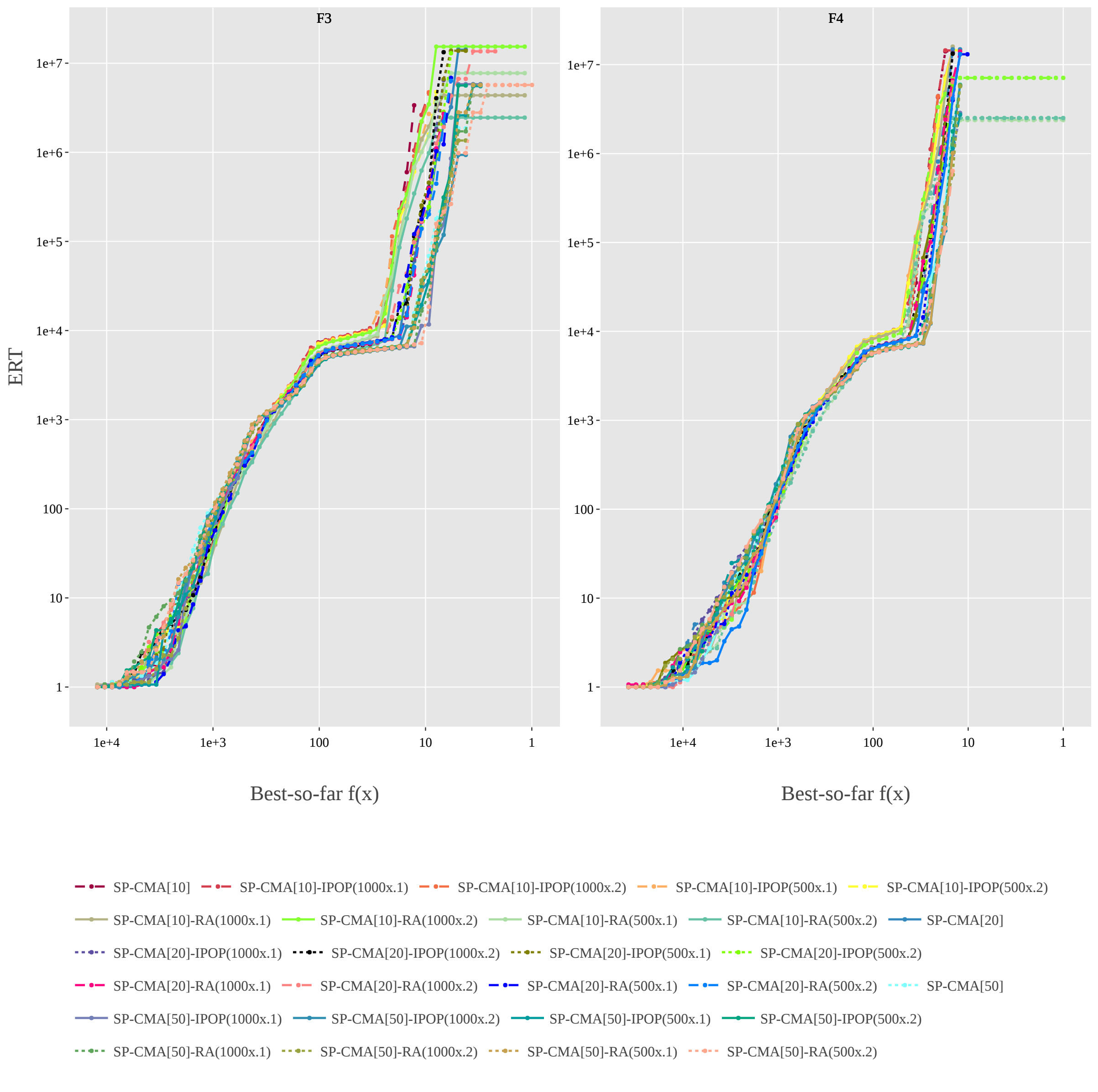


Figure 22: Experiment 4 Fixed-Target Results on F3 and F4 in 20 dimensions

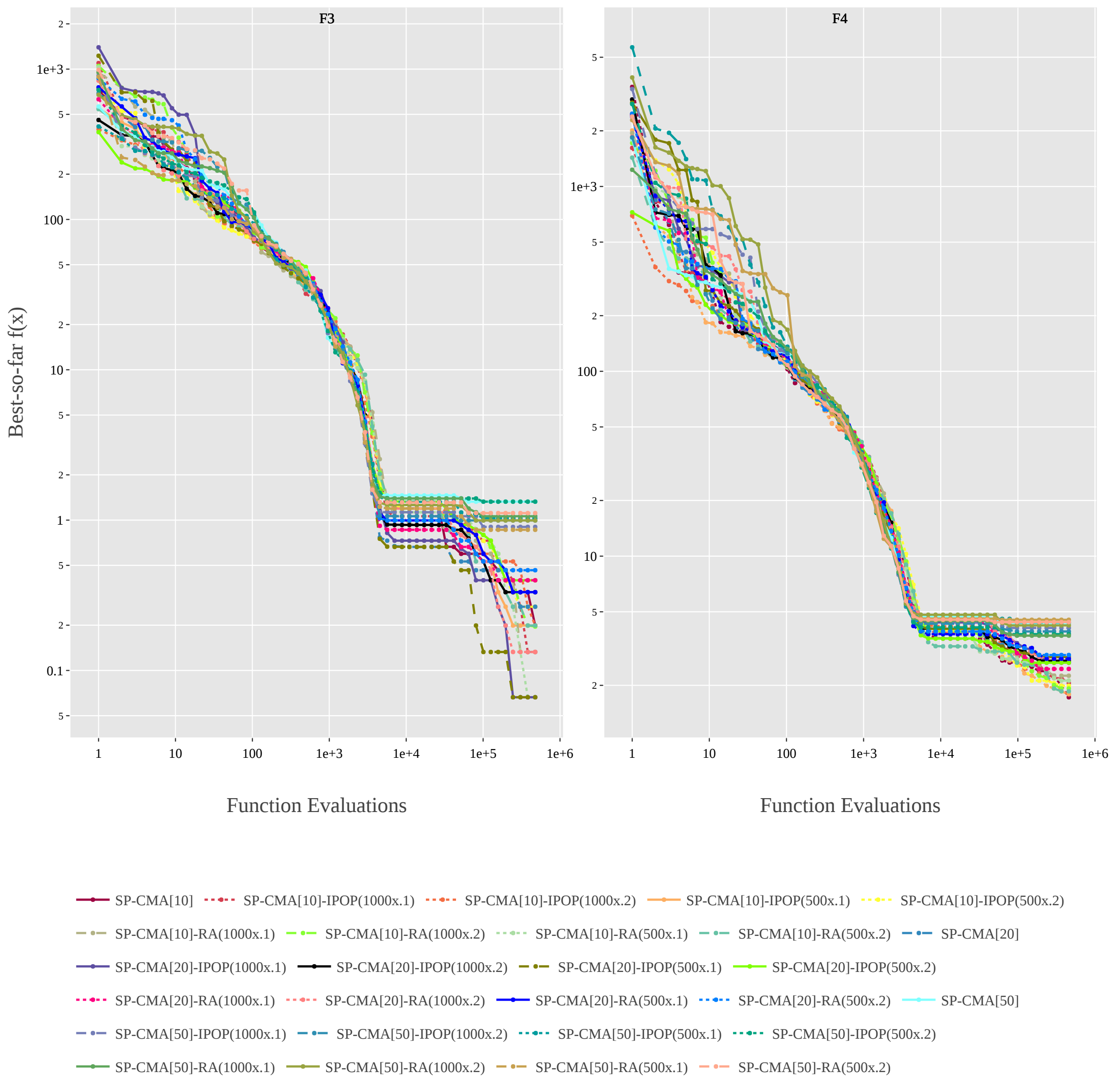


Figure 23: Experiment 4 Fixed-Budget Results on F3 and F4 in 5 dimensions

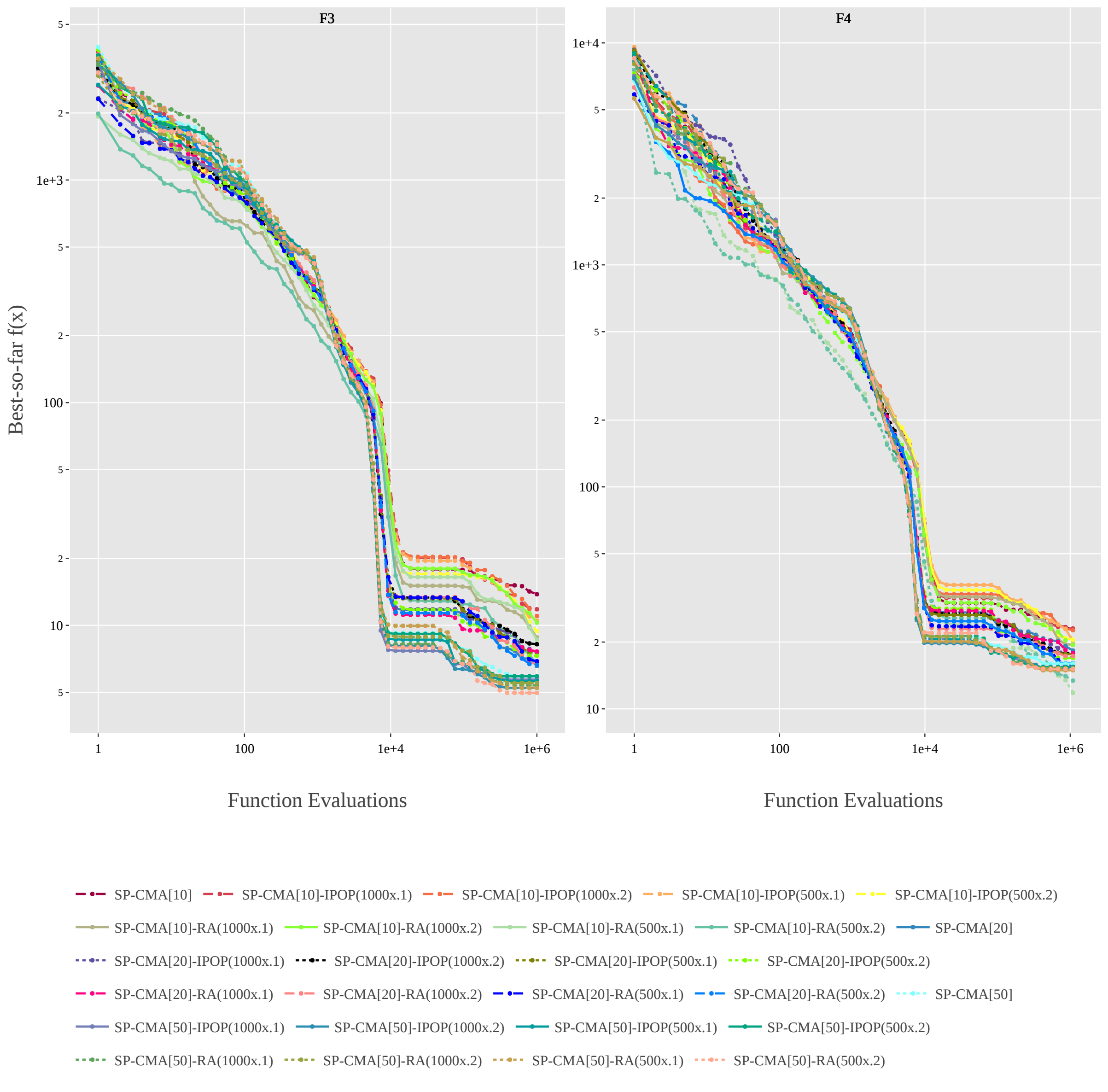
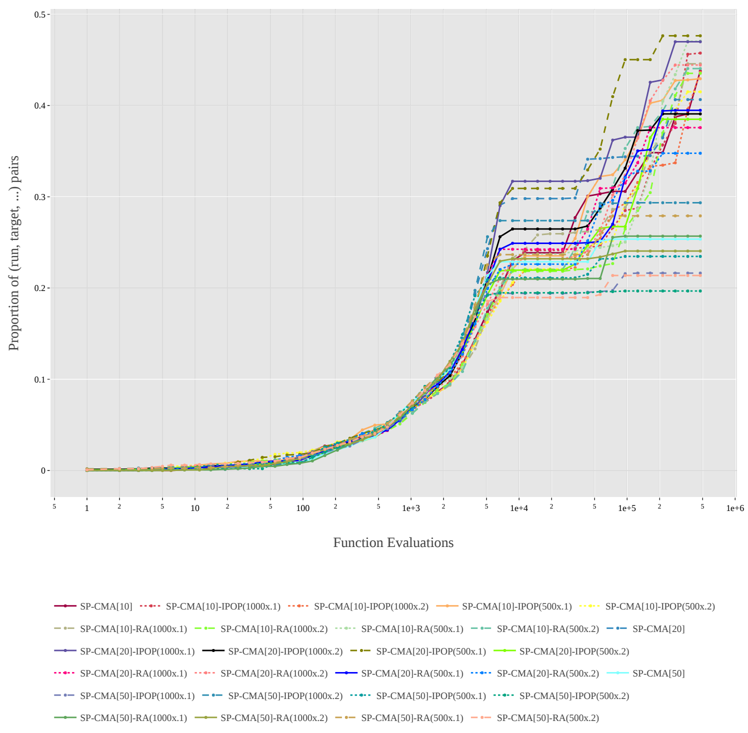
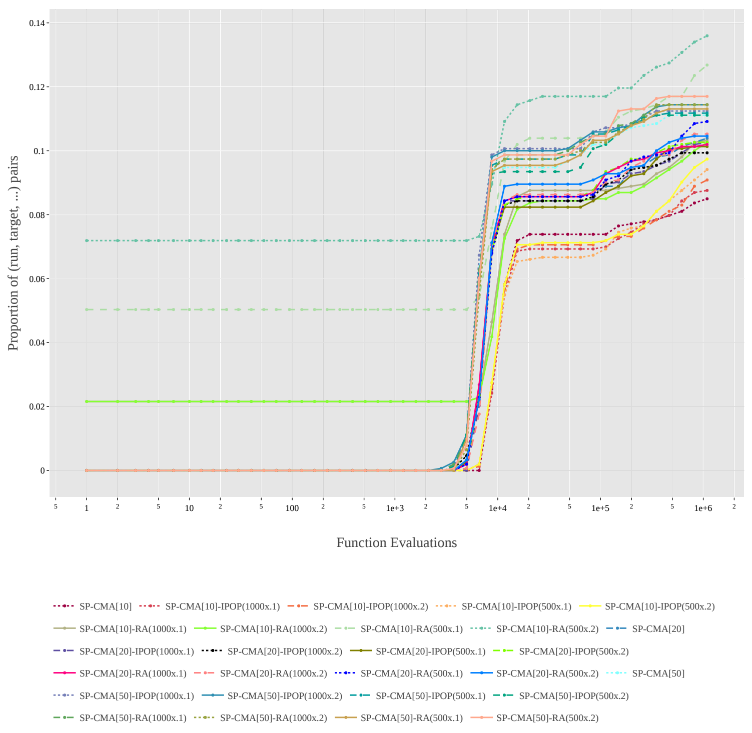


Figure 24: Experiment 4 Fixed-Budget Results on F3 and F4 in 20 dimensions



(a) $D = 5$



(b) $D = 20$

Figure 25: Aggregated Empirical Cumulative Distribution of Experiment 4 over all functions

4.6 Experiment 5 - Extended Benchmark

Finally, we benchmark some of the configurations from previous experiments on an extended test suite. More specifically, we extend the experiments on BBOB noiseless functions 3, 4, 15, 16, 17, 19, 20 and 24. The additional functions tested are all highly multimodal functions, with functions 15, 16, 17, and 19 possessing adequate global structure, while functions 20 and 24 possess weak global structure. Those functions are tested with the most promising or interesting configurations from previous experiments for 15 instances, in dimensions 5 and 20. Table 9 displays the configurations run for this experiment. The results are plotted in Figures 26, 27, 28, 29 for Fixed-Target results, Figures 30, 31, 32, 33 for Fixed-Budget results. The ECDF results are plotted in Figure 34.

After analyzing the results of other functions, we see a similar trend observed from function 3 and function 4 on the other functions. The ERT Figures 26, 27, 28 and 29 show that while the default `ModCMA` configuration is faster at reaching a certain target, it gets caught up by other configurations in bigger budgets (see functions 17, 20 and 24 in Figures 26 and 27).
Looking closely at functions 19 and 24's Fixed-Budget results (Figure 30), we see that `ModCMA`, although it outperforms other configurations initially, it gets outperformed by all configurations after the 500 function evaluation mark. It must be said however that the configurations are compared against a default `ModCMA` baseline, which could be optimized through hyperparameter optimization.

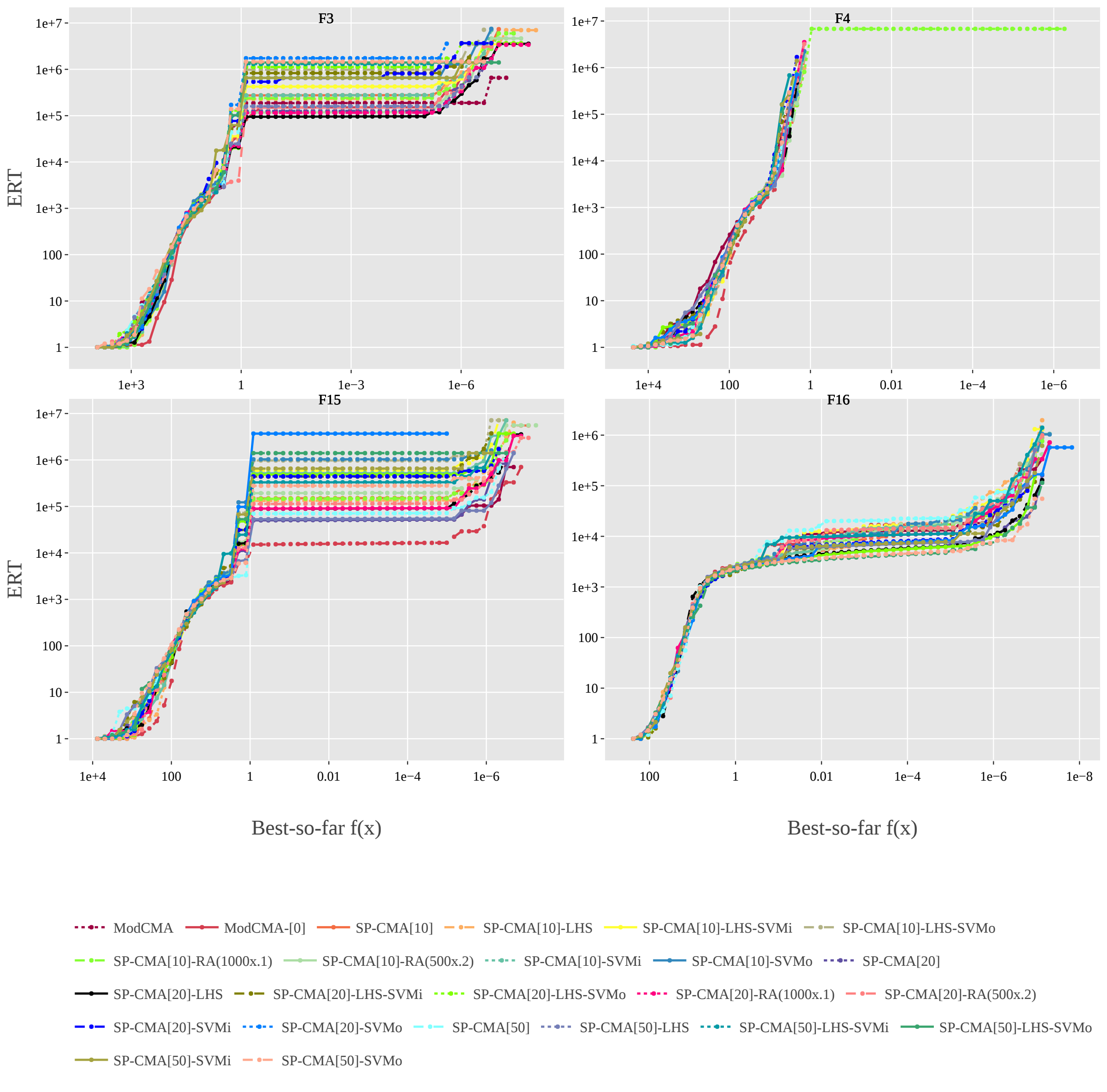
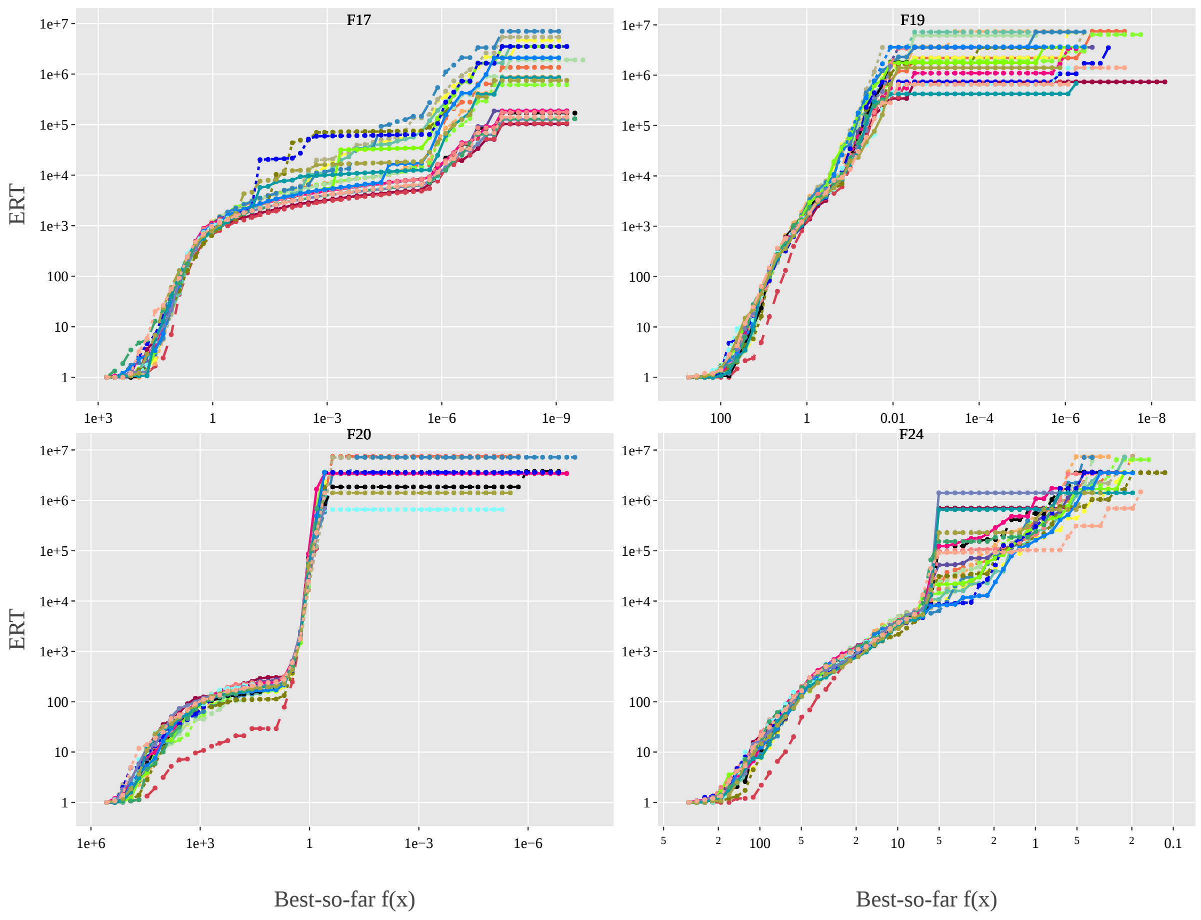


Figure 26: Experiment 5 Fixed-Target Results on F3, F4, F15 and F16 in 5 dimensions



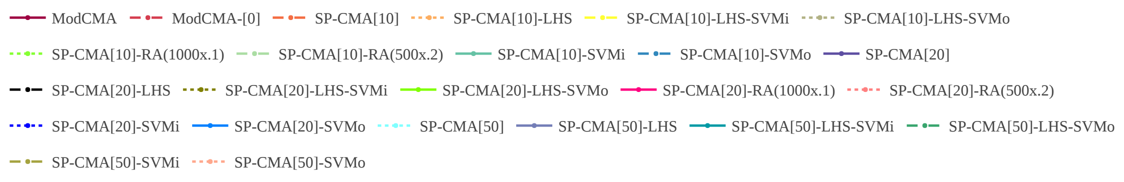


Figure 27: Experiment 5 Fixed-Target Results on F17, F19, F20 and F24 in 5 dimensions

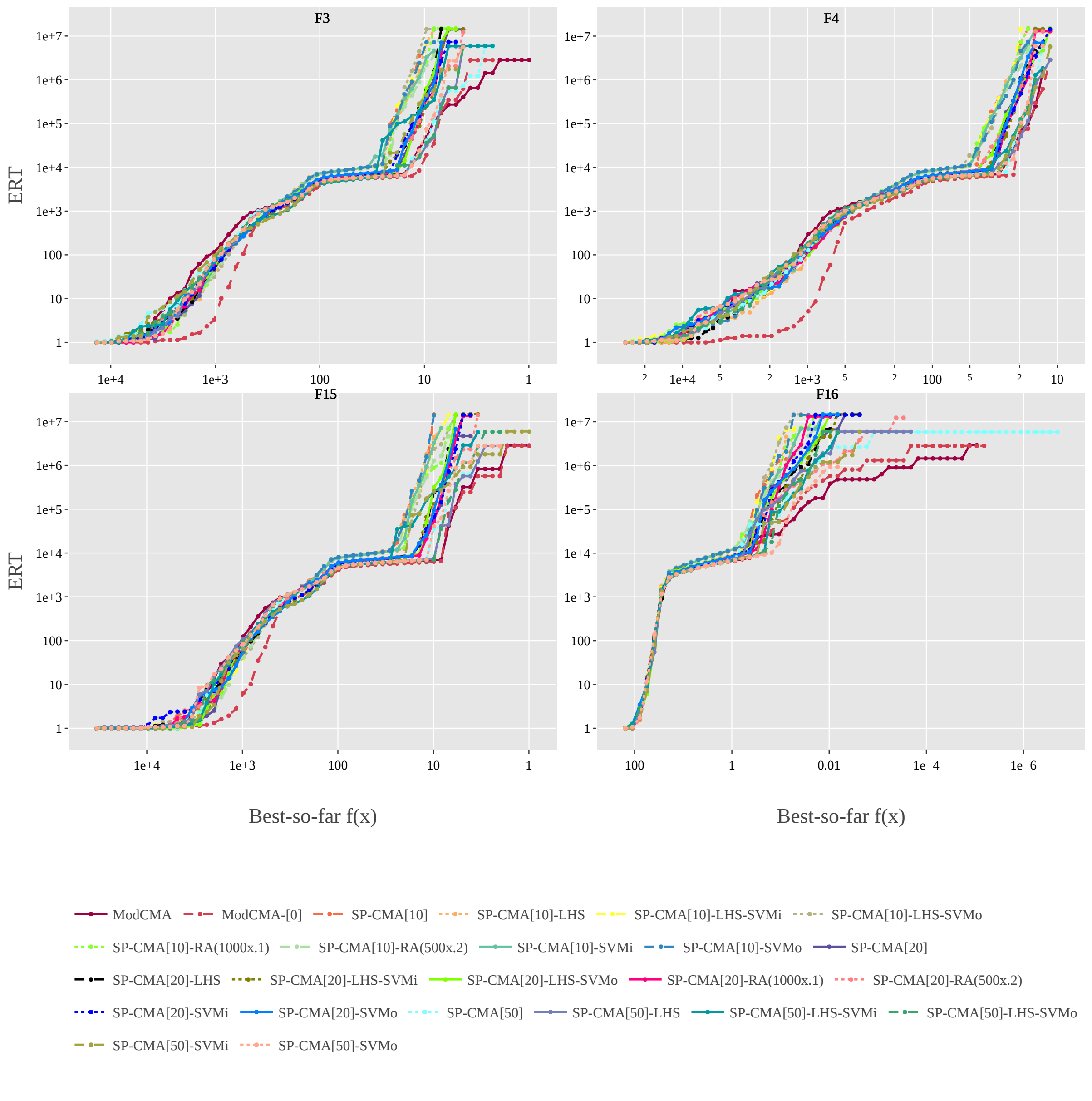


Figure 28: Experiment 5 Fixed-Target Results on F3, F4, F15 and F16 in 20 dimensions

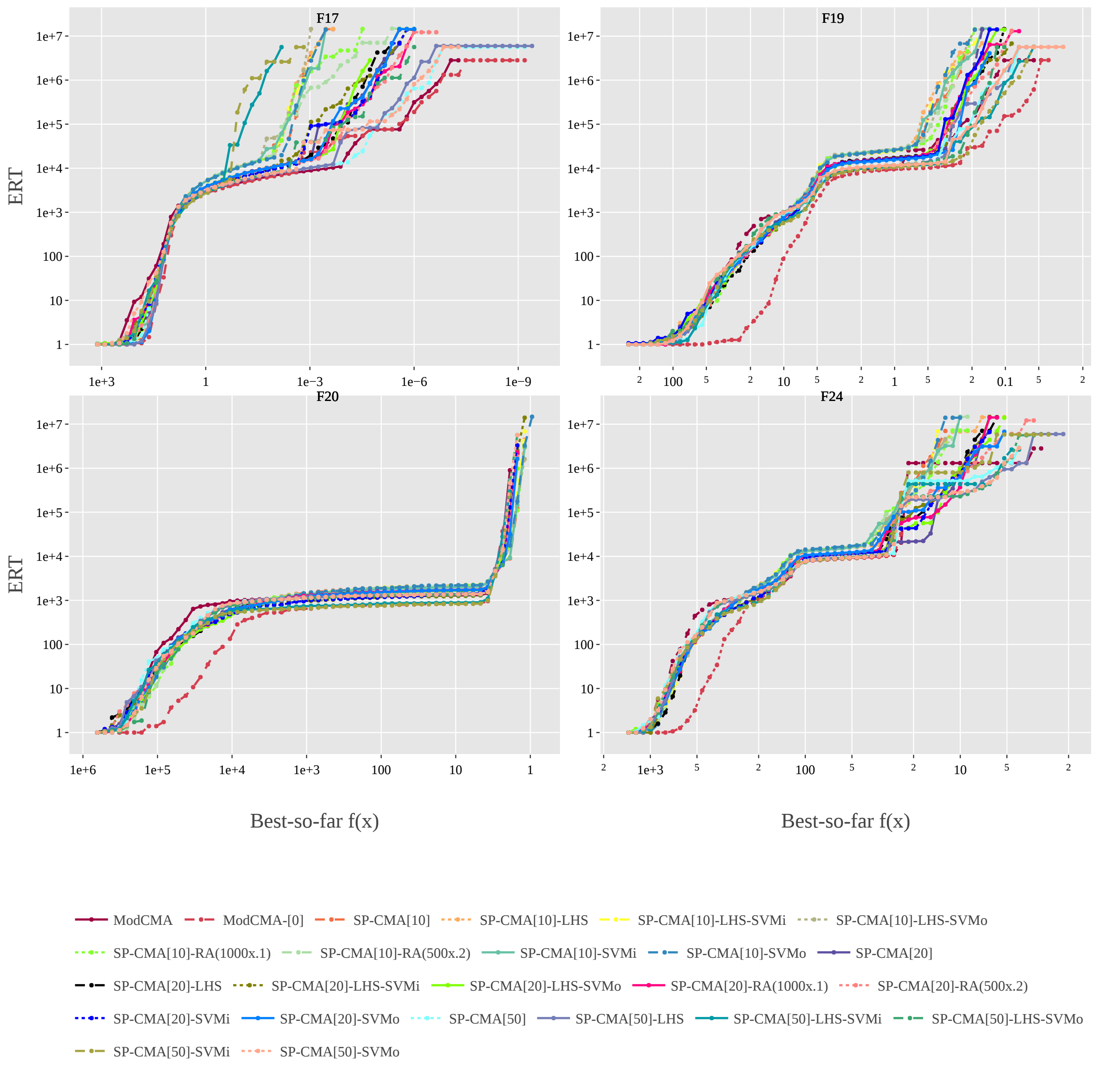


Figure 29: Experiment 5 Fixed-Target Results on F17, F19, F20 and F24 in 20 dimensions

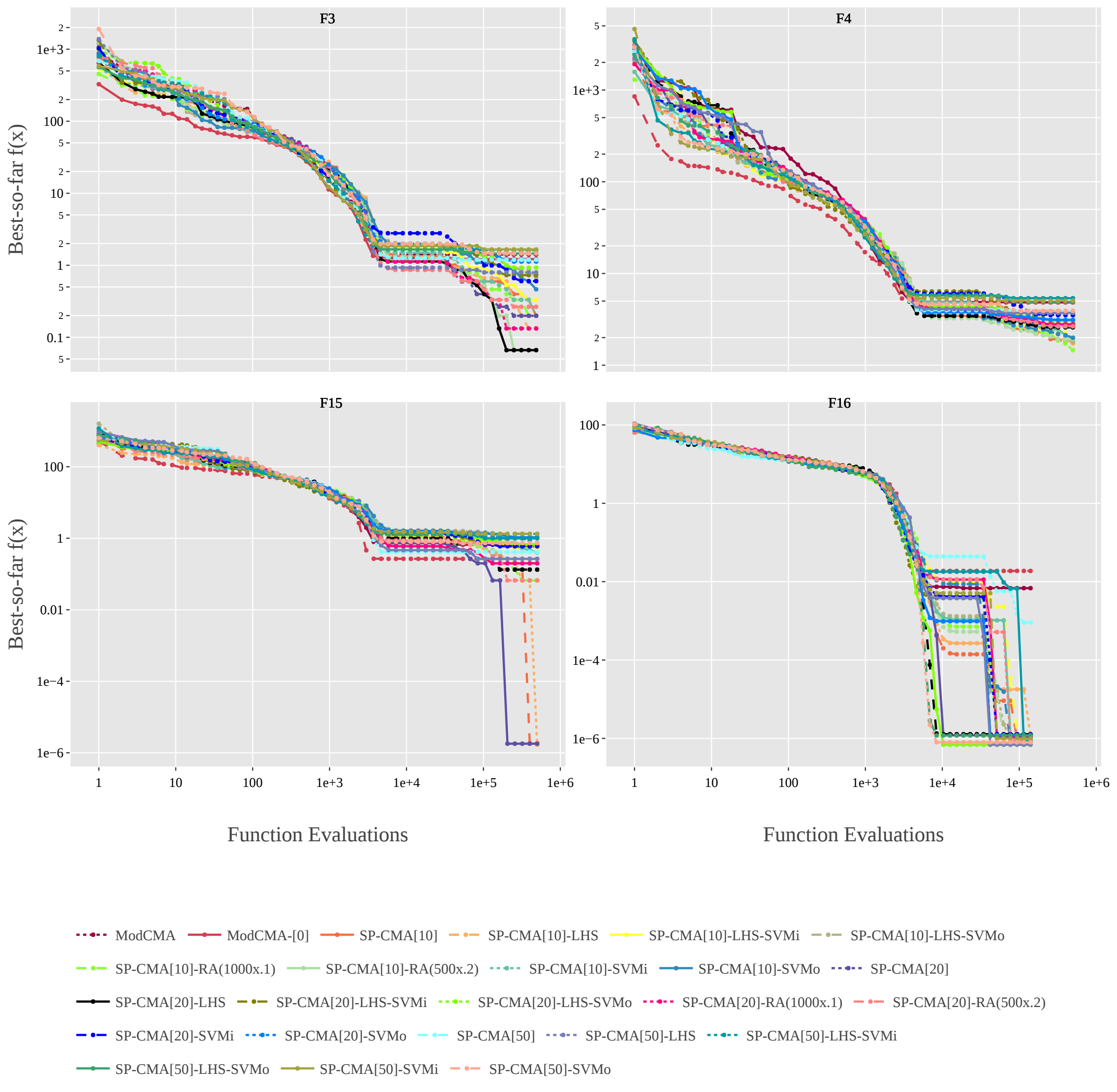


Figure 30: Experiment 5 Fixed-Budget Results on F3, F4, F15 and F16 in 20 dimensions

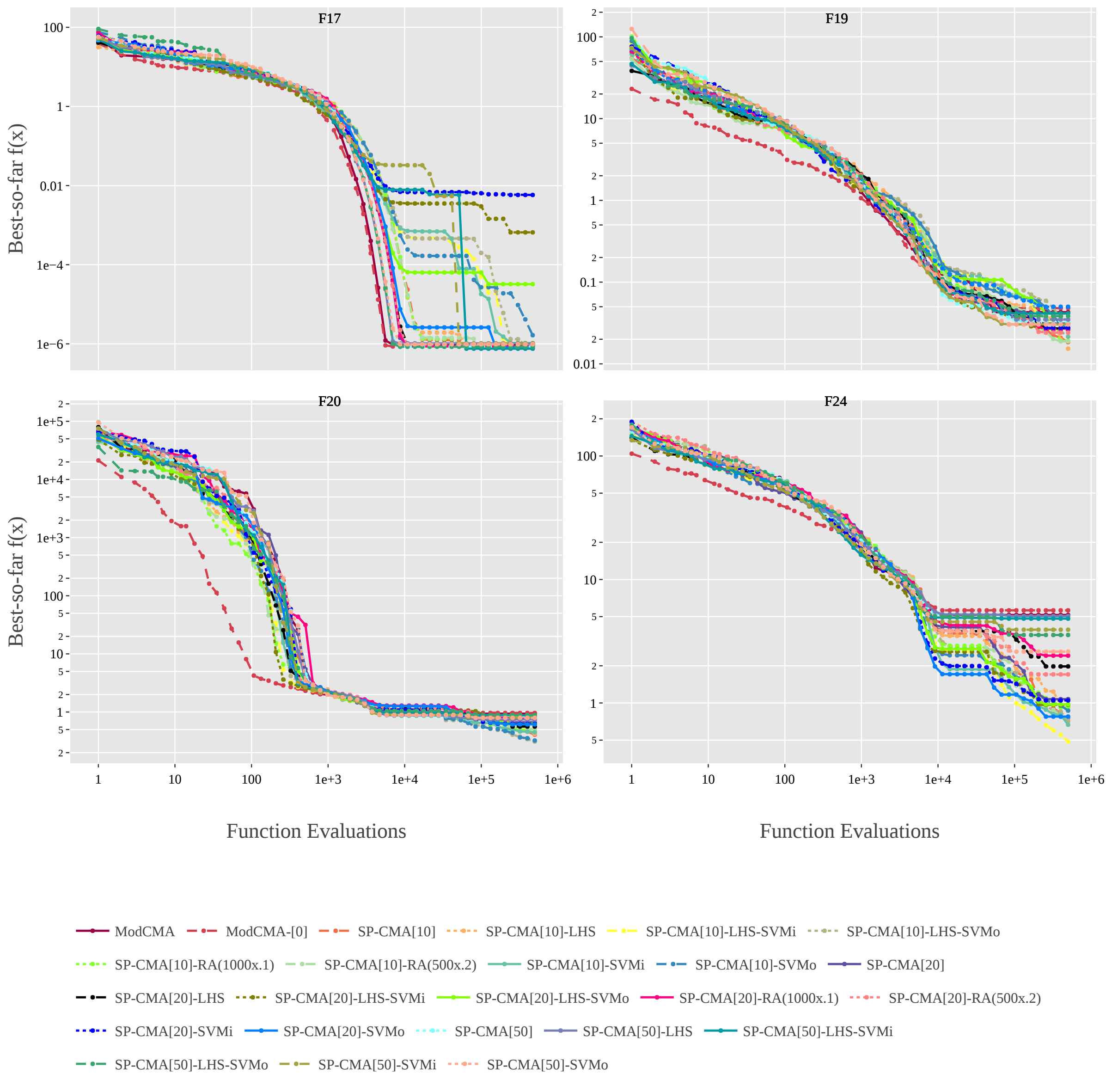


Figure 31: Experiment 5 Fixed-Budget Results on F17, F19, F20 and F24 in 20 dimensions

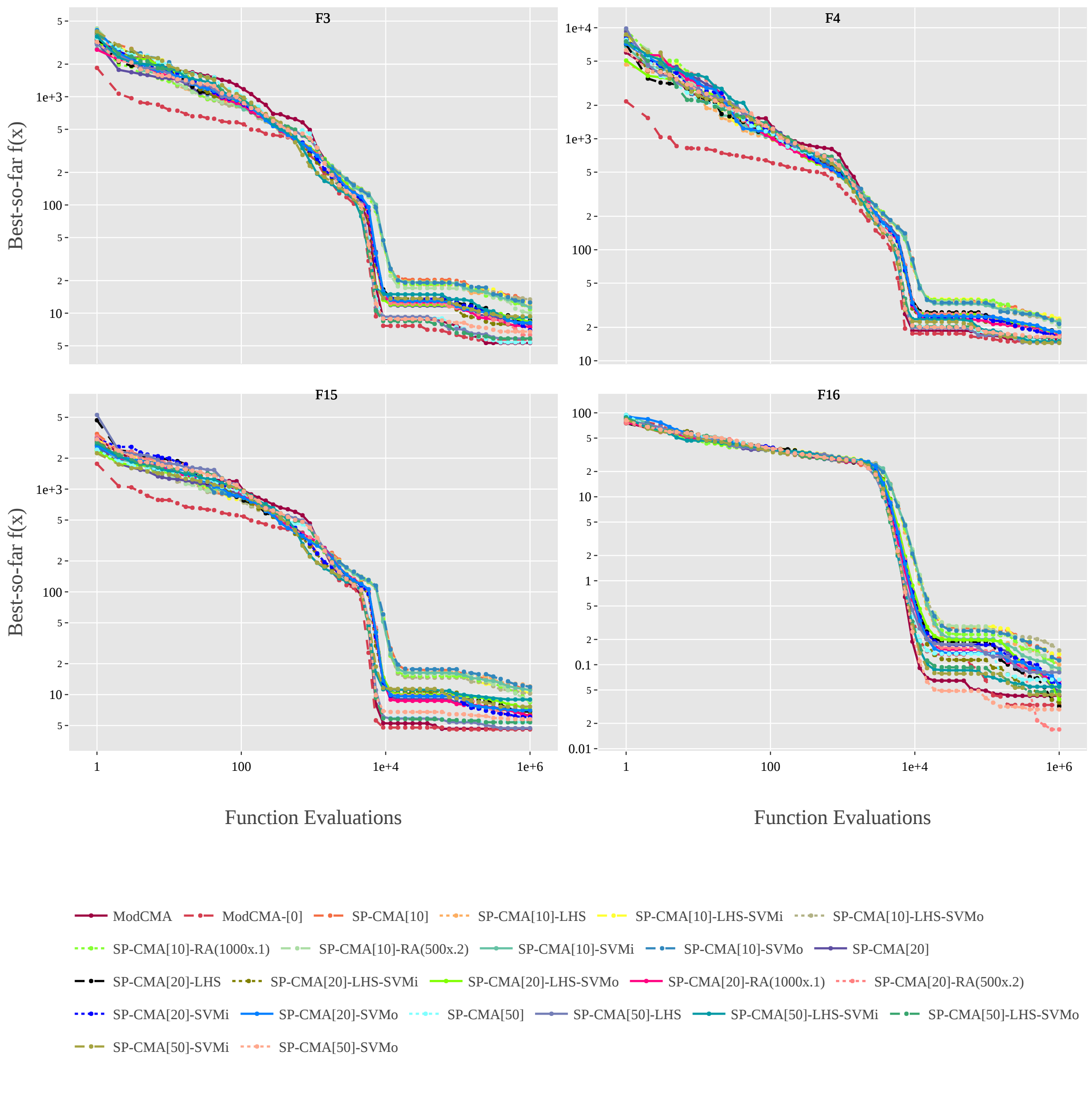


Figure 32: Experiment 5 Fixed-Budget Results on F3, F4, F15 and F16 in 20 dimensions

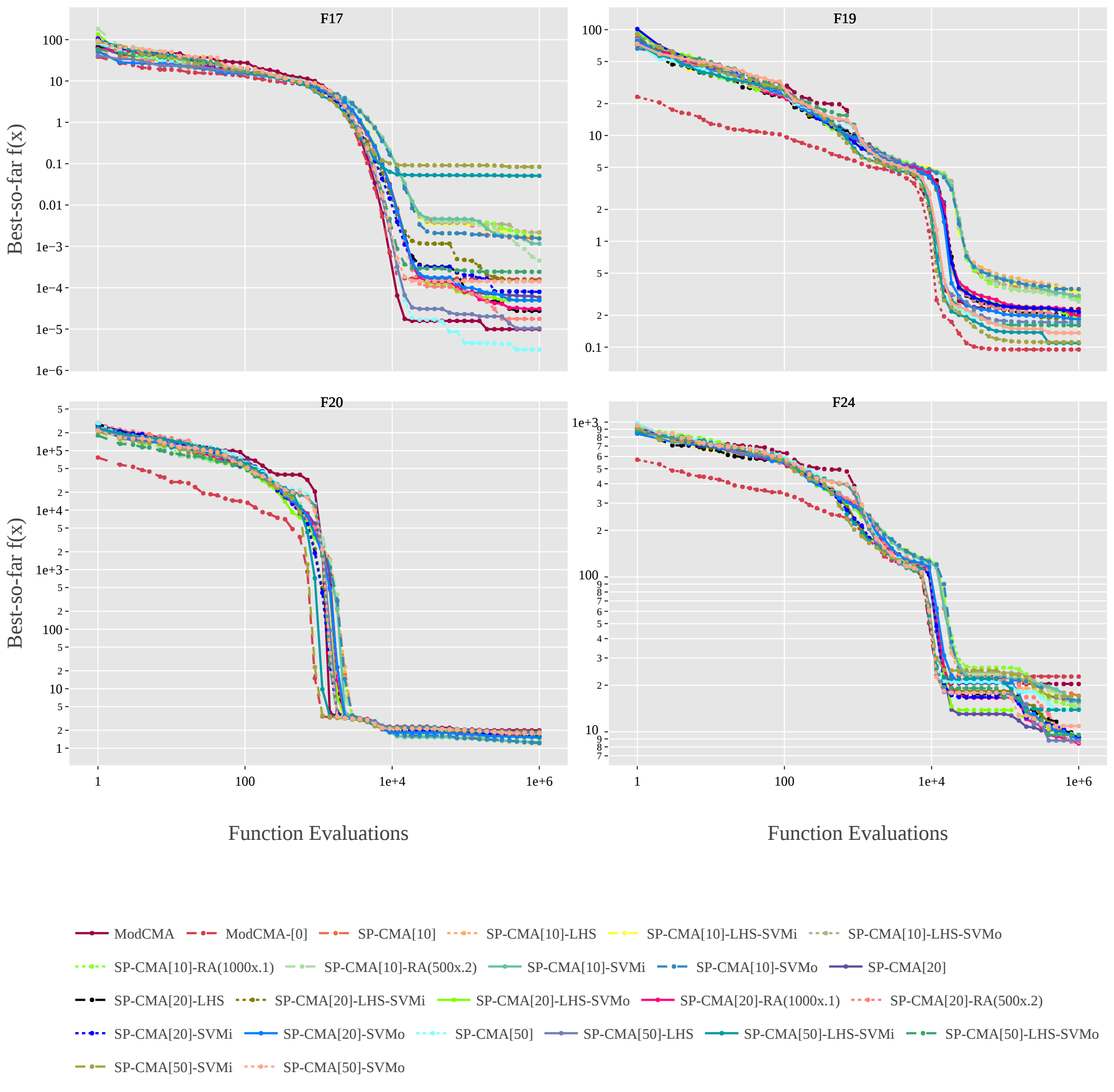
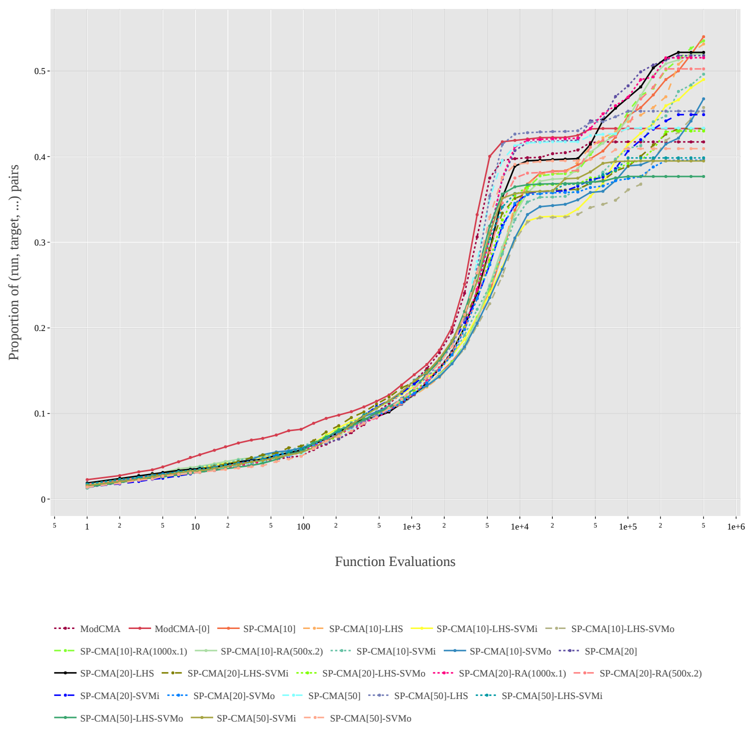
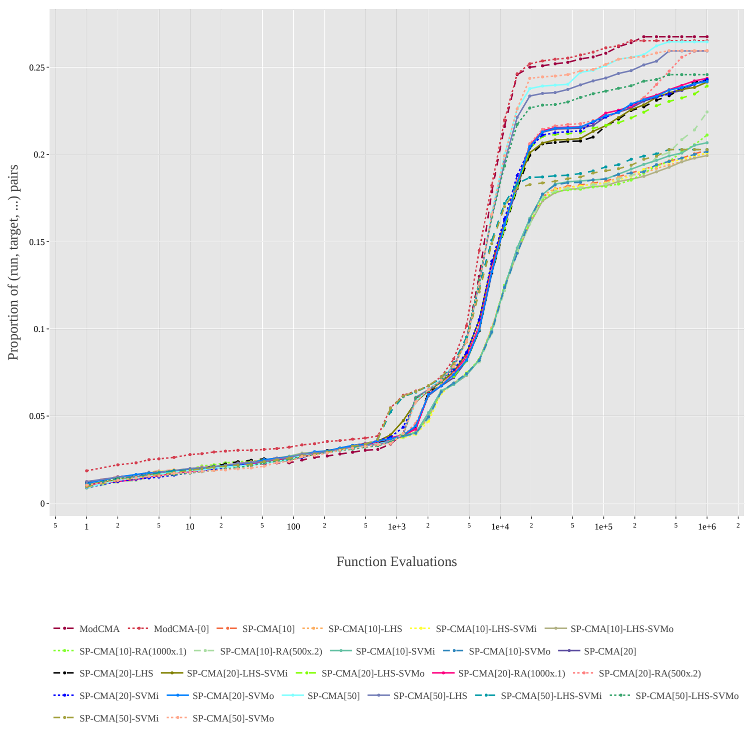


Figure 33: Experiment 5 Fixed-Budget Results on F17, F19, F20 and F24 in 20 dimensions



(a) $D = 5$



(b) $D = 20$

Figure 34: Aggregated Empirical Cumulative Distribution of Experiment 5 overall functions

5 Conclusion

In this project, we set out to investigate the effects of applying the principle of subpopulations, along with a few other methods, to the Covariance Matrix Adaptation Evolution Strategy algorithm. We looked into the effect of incorporating subpopulations of different sizes, analyzing the performance of a higher count of smaller subpopulations to a lower count of bigger subpopulations (RQ1). Configurations were benchmarked on the popular BBOB test suite [37] in multiple dimensions. We found that in lower dimensions, smaller subpopulations are more efficient and outperform bigger subpopulations, even outperforming the default CMA-ES.
The results imply a great potential for employing subpopulations to improve the performance of the CMA-ES algorithm, with some configurations outperforming the default CMA-ES benchmark. Through the use of subpopulations, CMA-ES is able to explore a wider range of solutions in the search space, ultimately finding better solutions. We also find that while this is true for a lower dimensionality, subpopulations did not improve CMA-ES in larger dimensions. Still, we did not extend this research to cover large-scale performance, which opens the door for future research.

We then looked at the effect different population initialization methods have on the performance of subpopulations (RQ2). We could not find any concrete evidence that a specific initialization method improves the performance of subpopulations in CMA-ES. Instead, we posited that due to the low number of points being generated for the subpopulations' initialization centroid, CMA-ES cannot benefit from the sampling method used. We did not observe any run clearly outperforming others, although we note that Gaussian is able to reach better results faster than other methods, most likely due to the nature of the objective functions benchmarked.

Additionally, we looked at the potential benefit of combining promising concepts with CMA-ES and the idea of subpopulations. We borrow from the concept of niching to propose a novel strategy of separating subpopulations on the search space using Support Vector Machines (RQ3). While we theorized that this could enable subpopulations to explore a more diverse set of solutions, we found that this strategy proved to hinder the performance of CMA-ES, enforcing boundaries too strict preventing the algorithm from exploring areas of the search space where optimal solutions might be. Future research exploring a relaxation of the boundaries might lead to better results. Furthermore, this technique was not explored to extract more than one optimal solution, which could be realized in future works.

We also analyzed the effects of adaptive population size with subpopulation (RQ4), as this concept has already been extended to the CMA-ES algorithm and has proved to improve its performance [3, 58]. We explored the concept of increasing the population size, borrowing from IPOP-CMA-ES [6], and a novel technique for re-allocating population size to better-performing subpopulations. We found this technique to improve the performance of small subpopulations. Increasing population at specified t intervals improved results in lower dimensions, while reallocating individuals between subpopulations showed better performances in higher dimensions.

We find that the use of subpopulations in CMA-ES is a field that shows a lot of potential for future research and improvements. We successfully found that employing subpopulations improves the performance of the CMA-ES algorithm in low dimensions, especially when coupled with population size adaptation. We also discovered that the initialization method of subpopulations was not an important factor of performance. While other methods tested in this research leave much to be desired, they also open the door for future research, which is further discussed in Section 6.

6 Future Works

The results obtained by this project have shown interesting results, and open the door for further research in a variety of aspects. We were successfully able to improve the performance of CMA-ES using subpopulations in low dimensions, but unsuccessful when looking at high-dimensional problems. We did not extend this research to investigate large-scale problems, suggesting a trial on the large-scale BBOB test suite [26] in future works. Furthermore, ModularCMAES [83, 19] possesses a lot of parameter settings that were not tuned or tested for this research. An extensive study could be performed to tune the performance of subpopulations and CMA-ES.

We also noted the unsuccessful attempt at using search space partitioning to improve the diversity of solutions in subpopulation CMA-ES. While we did not find any benefit of using SVM separation, we suggest the possibility of improving the technique to extract a multitude of solutions, better mimicking the concept it is modelled after: niching. In our works, we borrowed from niching to model this separation technique, but we did not use it as a niching method and instead only extracted a single optimal solution. We also propose that looser restrictions, or updating the SVM boundaries less regularly could help the algorithm.

Finally, we found that combining population size adaptation, an already successful concept for CMA-ES [6, 3], improved the performance of subpopulations. In our research however, we were unable to properly tune this configuration and found that no technique of population size adaptation (resource re-allocation `RA` or population size increase `IPOP`) was better than the other. We suggest repeating the experiment after a more thorough tuning could lead to an even greater improvement.

References

[1] A. Ahrari, K. Deb, and M. Preuss. Multimodal Optimization by Covariance Matrix Self-Adaptation Evolution Strategy with Repelling Subpopulations. *Evolutionary Computation*, 25, 04 2016.

[2] A. Ahrari, S. Elsayed, R. Sarker, D. Essam, and C. A. C. Coello. Static and Dynamic Multimodal Optimization by Improved Covariance Matrix Self-Adaptation Evolution Strategy With Repelling Subpopulations. *IEEE Transactions on Evolutionary Computation*, 26(3):527–541, 2022.

[3] A. Atamna. Benchmarking IPOP-CMA-ES-TPA and IPOP-CMA-ES-MSR on the BBOB Noiseless Testbed. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1135–1142, New York, NY, USA, 2015. Association for Computing Machinery.

[4] A. Auger, S. Finck, N. Hansen, and R. Ros. BBOB 2009: Comparison Tables of All Algorithms on All Noiseless Functions. 04 2010.

[5] A. Auger, S. Finck, N. Hansen, and R. Ros. BBOB 2010: Comparison Tables of All Algorithms on All Noisy Functions. 09 2010.

[6] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776 Vol. 2, 2005.

[7] T. Bäck, F. A. Hoffmeister, and H.-P. Schwefel. A Survey of Evolution Strategies. In *International Conference on Genetic Algorithms*, 1991.

[8] P. Beerli, D. Evans, and M. Mascagni. On the Scrambled Sobol Sequence. volume 3516, pages 775–782, 05 2005.

[9] H.-G. Beyer and B. Sendhoff. Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy –. In G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, editors, *Parallel Problem Solving from Nature – PPSN X*, pages 123–132, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[10] J. Blank and K. Deb. Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

[11] E. Braaten and G. Weller. An improved low-discrepancy sequence for multidimensional quasi-Monte Carlo integration. *Journal of Computational Physics*, 33(2):249–258, 1979.

[12] R. Bridson. Fast Poisson Disk Sampling in Arbitrary Dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, pages 22–es, New York, NY, USA, 2007. Association for Computing Machinery.

[13] J. Bustos, V. A. Jimenez, and A. Will. A comparison of different types of Niching Genetic Algorithms for variable selection in solar radiation estimation, 2020.

[14] F. Caraffini, A. V. Kononova, and D. Corne. Infeasibility and structural bias in differential evolution. *Information Sciences*, 496:161–179, 2019.

[15] H. Chen, R. Cheng, J. Wen, H. Li, and J. Weng. Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. *Information Sciences*, 509:457–469, 2020.

[16] C.-H. Chou and J.-N. Chen. Genetic algorithms: initialization schemes and genes extraction. In *Ninth IEEE International Conference on Fuzzy Systems. FUZZ- IEEE 2000 (Cat. No.00CH37063)*, volume 2, pages 965–968 vol.2, 2000.

[17] I. L. Dalal, D. Stefan, and J. Harwayne-Gidansky. Low discrepancy sequences for Monte Carlo simulations on reconfigurable platforms. In *2008 International Conference on Application-Specific Systems, Architectures and Processors*, pages 108–113, 2008.

[18] S. Das, S. Maity, B.-Y. Qu, and P. Suganthan. Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88, 2011.

[19] J. de Nobel, D. Vermetten, H. Wang, C. Doerr, and T. Bäck. Tuning as a Means of Assessing the Benefits of New Ideas in Interplay with Existing Algorithmic Modules. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, pages 1375–1384, New York, NY, USA, 2021. Association for Computing Machinery.

[20] J. Devore. A modern introduction to probability and statistics: Understanding why and how. *Journal of the American Statistical Association*, 101(473):393–394, mar 2006.

[21] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck. IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics. *CoRR*, abs/1810.05281, 2018.

[22] R. W. Doran. The Gray Code. *JUCS - Journal of Universal Computer Science*, 13(11):1573–1597, 2007.

[23] D.-Z. Du, P. M. Pardalos, and W. Wu. History of Optimization. In *Encyclopedia of Optimization*, pages 1538–1542. Springer US, 2008.

[24] A. E. Eiben and J. E. Smith. Evolutionary Computing: The Origins. In *Natural Computing Series*, pages 13–24. Springer Berlin Heidelberg, 2015.

[25] A. El Dor, M. Clerc, and P. Siarry. A multi-swarm PSO using charged particles in a partitioned search space for continuous optimization. *Computational Optimization and Applications*, 53(1):271–295, 2012.

[26] O. A. ElHara, K. Varelas, D. M. Nguyen, T. Tuar, D. Brockhoff, N. Hansen, and A. Auger. COCO: The Large Scale Black-Box Optimization Benchmarking (bbob-largescale) Test Suite. *ArXiv*, abs/1903.06396, 2019.

[27] S. Elsayed, R. Sarker, and C. A. Coello Coello. Sequence-Based Deterministic Initialization for Evolutionary Algorithms. *IEEE Transactions on Cybernetics*, 47(9):2911–2923, 2017.

[28] J. E. Fieldsend. Running Up Those Hills: Multi-modal search with the niching migratory multi-swarm optimiser. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2593–2600, 2014.

[29] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions. Technical report, Research Center PPE, 2010.

[30] N. N. Glibovets and N. M. Gulayeva. A Review of Niching Genetic Algorithms for Multimodal Function Optimization. *Cybernetics and Systems Analysis*, 49(6):815–820, 2013.

[31] S. Gobeyn, A. M. Mouton, A. F. Cord, A. Kaim, M. Volk, and P. L. Goethals. Evolutionary algorithms for species distribution modelling: A review in the context of machine learning. *Ecological Modelling*, 392:179–195, 2019.

[32] J. H. Halton. Algorithm 247: Radical-Inverse Quasi-Random Point Sequence. *Commun. ACM*, 7(12):701–702, dec 1964.

[33] N. Hansen. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2389–2396, New York, NY, USA, 2009. Association for Computing Machinery.

[34] N. Hansen. The CMA Evolution Strategy: A Tutorial. *CoRR*, abs/1604.00772, 2016.

[35] N. Hansen, Y. Akimoto, and P. Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, Feb. 2019.

[36] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, pages 1689–1696, New York, NY, USA, 2010. Association for Computing Machinery.

[37] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. COCO: a platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

[38] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[39] T. Hendtlass. WoSP: a multi-optima particle swarm algorithm. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 727–734 Vol.1, 2005.

[40] C. Igel, N. Hansen, and S. Roth. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1):1–28, 03 2007.

[41] K. Izumi, M. Hashem, and K. Watanabe. An evolution strategy with competing subpopulations. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 306–311. IEEE, 1997.

[42] B. Karmakar, A. Kumar, R. Mallipeddi, and D.-G. Lee. CMA-ES with exponential based multiplicative covariance matrix adaptation for global optimization. *Swarm and Evolutionary Computation*, 79:101296, 2023.

[43] B. Kazimipour, X. Li, and A. K. Qin. A review of population initialization techniques for evolutionary algorithms. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2585–2592, 2014.

[44] L. Kocis and W. J. Whiten. Computational Investigations of Low-Discrepancy Sequences. *ACM Trans. Math. Softw.*, 23(2):266–294, jun 1997.

[45] S. Lamichhaney, J. Berglund, M. S. Almén, K. Maqbool, M. Grabherr, A. Martinez-Barrio, M. Promerová, C.-J. Rubin, C. Wang, N. Zamani, B. R. Grant, P. R. Grant, M. T. Webster, and L. Andersson. Evolution of Darwin's finches and their beaks revealed by genome sequencing. *Nature*, 518(7539):371–375, 2015.

[46] W. Li. Matrix Adaptation Evolution Strategy with Multi-Objective Optimization for Multimodal Optimization. *Algorithms*, 12(3), 2019.

[47] I. Loshchilov and F. Hutter. CMA-ES for Hyperparameter Optimization of Deep Neural Networks, 2016.

[48] I. Loshchilov, M. Schoenauer, and M. Sebag. Self-Adaptive Surrogate-Assisted Covariance Matrix Adaptation Evolution Strategy, 2012.

[49] I. Loshchilov, M. Schoenauer, and M. Sèbag. Bi-Population CMA-ES Agorithms with Surrogate Models and Line Searches. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 1177–1184, New York, NY, USA, 2013. Association for Computing Machinery.

[50] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, and Z. Zhu. A Survey on Cooperative Co-Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2019.

[51] H. Maaranen, K. Miettinen, and M. Mäkelä. Quasi-random initial population for genetic algorithms. *Computers Mathematics with Applications*, 47(12):1885–1895, 2004.

[52] H. Maaranen, K. Miettinen, and A. Penttinen. On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, 37(3):405–436, 2007.

[53] S. W. Mahfoud. *Niching Methods for Genetic Algorithms.* PhD thesis, USA, 1996. UMI Order No. GAX95-43663.

[54] M. Mckay, R. Beckman, and W. Conover. A Comparison of Three Methods for Selecting Vales of Input Variables in the Analysis of Output From a Computer Code. *Technometrics*, 21:239–245, 05 1979.

[55] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs, Third Revised and Extended Edition*. Springer, 1996.

[56] M. Mitchell and C. E. Taylor. EVOLUTIONARY COMPUTATION: An Overview. *Annual review of ecology and systematics*, 30(1):593–616, 1999.

[57] J. Nagar and D. H. Werner. A Comparison of Three Uniquely Different State of the Art and Two Classical Multiobjective Optimization Algorithms as Applied to Electromagnetics. *IEEE Transactions on Antennas and Propagation*, 65(3):1267–1280, 2017.

[58] D. M. Nguyen and N. Hansen. Benchmarking CMAES-APOP on the BBOB Noiseless Testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 1756–1763, New York, NY, USA, 2017. Association for Computing Machinery.

[59] K. Nishida and Y. Akimoto. Benchmarking the PSA-CMA-ES on the BBOB Noiseless Testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '18, pages 1529–1536, New York, NY, USA, 2018. Association for Computing Machinery.

[60] K. Nishida and Y. Akimoto. PSA-CMA-ES: CMA-ES with Population Size Adaptation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 865–872, New York, NY, USA, 2018. Association for Computing Machinery.

[61] M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In *1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings (Cat. No.99TH8410)*, pages 88–92, 1999.

[62] A. Olsson, G. Sandberg, and O. Dahlblom. On Latin hypercube sampling for structural reliability analysis. *Structural Safety*, 25(1):47–68, 2003.

[63] A. B. Owen. Monte Carlo theory, methods and examples. `https://artowen.su.domains/mc/`, 2013.

[64] D. Parkhurst and O. Loucks. Optimal Leaf Size in Relation to Environment. *The Journal of Ecology*, 60, 07 1972.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[66] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama. Quasi-oppositional Differential Evolution. In *2007 IEEE Congress on Evolutionary Computation*, pages 2229–2236, 2007.

[67] I. Rechenberg. Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 1973.

[68] D. Reina, H. Tawfik, and S. Toral. Multi-subpopulation evolutionary algorithms for coverage deployment of UAV-networks. *Ad Hoc Networks*, 68:16–32, 2018. Advances in Wireless Communication and Networking for Cooperating Autonomous Systems.

[69] S. Rodrigues, P. Bauer, and P. A. Bosman. A Novel Population-Based Multi-Objective CMA-ES and the Impact of Different Constraint Handling Techniques. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 991–998, New York, NY, USA, 2014. Association for Computing Machinery.

[70] M. Sazanovich, A. Nikolskaya, Y. Belousov, and A. Shpilman. Solving Black-Box Optimization Challenge via Learning Search Space Partition for Local Bayesian Optimization. In H. J. Escalante and K. Hofmann, editors, *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 77–85. PMLR, 06–12 Dec 2021.

[71] H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung.* PhD thesis, 01 1975.

[72] H.-P. Schwefel. Advantages (and Disadvantages) of Evolutionary Computation Over Other Approaches. *Evolutionary Computation*, 1, 01 2000.

[73] M. Shibata and M. Nomura. Lightweight Covariance Matrix Adaptation Evolution Strategy (CMA-ES) implementation. `https://github.com/CyberAgentAILab/cmaes`, 2023.

[74] H. Shimizu and M. Toyoda. CMA-ES with Coordinate Selection for High-Dimensional and Ill-Conditioned Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, pages 209–210, New York, NY, USA, 2021. Association for Computing Machinery.

[75] O. Shir and T. Bäck. Niche Radius Adaptation in the CMA-ES Niching Algorithm. pages 142–151, 01 2006.

[76] O. Shir, M. Emmerich, and T. Bäck. Adaptive Niche Radii and Niche Shapes Approaches for Niching with the CMA-ES. *Evolutionary computation*, 18:97–126, 03 2010.

[77] O. M. Shir. Niching in Evolutionary Algorithms. In *Handbook of Natural Computing*, pages 1035–1069. Springer Berlin Heidelberg, 2012.

[78] O. M. Shir and T. Bäck. Niching in Evolution Strategies. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 915–916, New York, NY, USA, 2005. Association for Computing Machinery.

[79] Z. Skolicki. An Analysis of Island Models in Evolutionary Computation. In *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*, GECCO '05, pages 386–389, New York, NY, USA, 2005. Association for Computing Machinery.

[80] I. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[81] A. Tharwat and W. Schenck. Population initialization techniques for evolutionary algorithms for single-objective constrained optimization problems: Deterministic vs. stochastic techniques. *Swarm and Evolutionary Computation*, 67:100952, 2021.

[82] S. van Rijn, H. Wang, M. van Leeuwen, and T. Bäck. Evolving the structure of evolution strategies. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.

[83] D. Vermetten and J. de Nobel. ModularCMAES. `https://github.com/IOHprofiler/ModularCMAES`, 2023.

[84] D. Vermetten, M. López-Ibáñez, O. Mersmann, R. Allmendinger, and A. V. Kononova. Analysis of Modular CMA-ES on Strict Box-Constrained Problems in the SBOX-COST Benchmarking Suite. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, pages 2346–2353, New York, NY, USA, 2023. Association for Computing Machinery.

[85] H. Wang, D. Vermetten, F. Ye, C. Doerr, and T. Bäck. IOHanalyzer: Detailed Performance Analyses for Iterative Optimization Heuristics. *ACM Trans. Evol. Learn. Optim.*, 2(1), apr 2022.

[86] L. Wang, R. Fonseca, and Y. Tian. Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19511–19522. Curran Associates, Inc., 2020.

[87] S. Watanabe and J. Le Roux. Black box optimization for automatic speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3256–3260, 2014.

[88] X. Yu and M. Gen. Multimodal Optimization. In *Decision Engineering*, pages 165–191. Springer London, 2010.

[89] X. Yu and M. Gen. Simple Evolutionary Algorithms. In *Decision Engineering*, pages 11–38. Springer London, 2010.

[90] J. Zou, C. Ji, S. Yang, Y. Zhang, J. Zheng, and K. Li. A knee-point-based evolutionary algorithm using weighted subpopulation for many-objective optimization. *Swarm and Evolutionary Computation*, 47:33–43, 2019. Special Issue on Collaborative Learning and Optimization based on Swarm and Evolutionary Computation.

Appendix

BBOB Functions

Table 3: 24 noiseless BBOB functions and their properties

Function ID	Objective Function	Properties
1	Sphere	unimodal, highly symmetric
2	Ellipsoidal	unimodal, conditioning is about 10^6
3	**Rastrigin**	highly multimodal, regular placement of local optimas, roughly 10^D local optimas
4	**Büche-Rastrigin**	highly multimodal, asymmetric placement of local optimas, roughly 10^D local optimas
5	Linear	linear function, x^{opt} is on domain boundary
6	Attractive Sector	unimodal, highly asymmetric
7	Step Ellipsoidal	unimodal, highly asnon-separable
8	Rosenbrock	unimodal, highly tri-band dependency structure
9	Rosenbrock (rotated)	rotated version of function 8
10	Ellipsoidal	unimodal, quadratic ill-condition function with smooth local irregularities, non-separable version of function 2
11	Discus	unimodal, quadratic function with local irregularities
12	Bent Cigar	unimodal, rotated
13	Sharp Ridge	unimodal, high-conditioning
14	Different Powers	unimodal, high-conditioning
15	**Rastrigin**	highly multimodal, non-separable less regular counterpart of function 3, roughly 10^D local optimas
16	**Weierstrass**	highly rugged and moderately repetitive landscape, locally irregular, non-unique global optimum
17	**Schaffers F7**	highly multimodal, asymmetric, low conditioning
18	Schaffers F7 (ill-conditioned)	moderately ill-conditioned counterpart of function 17
19	**Composite Griewank-Rosenbrock**	resembles function 8 but highly multimodal
20	**Schwefel**	prominent 2^D minima are located close to corners of the unpenalized search area
21	Gallagher's Gaussian 101-me Peaks	consists of 101 optimas with unrelated position and height randomly chosen
22	Gallagher's Gaussian 21-hi Peaks	consists of 21 optimas with unrelated position and height randomly chosen, higher conditioning than function 21
23	Katsuura	highly rugged and repetitive, more than 10^D global optimas
24	**Lunacek bi-Rastrigin**	highly multimodal, constructed to be deceptive to EAs with larger populations

Default Experiments Parameters

Table 4: Default Experiment Parameters

Parameter	Value
ub	5
lb	-5
budget	$dimension * $ 10000
target	-inf
iterations	10000
x0	uniform distribution
sigma0	0.2
bound_correction	"saturate"
step_size_adaptation	"csa"
base_sampler	"gaussian"
active	False
elitist	False
sequential	False
threshold_convergence	False
orthogonal	False
local_restart	None
mirrored	"mirrored"
weights_option	"default"
ps_factor	1.0

Experiment 1

Table 5: Experiment 1 runs configurations. S_n = subpopulations count

Configurations				
Name	S_n	λ	μ	Initilization
ModCMA	n/a	100	50	uniform
ModCMA-[0]	n/a	100	50	center of search space
SP-CMA[50]	2	$[50] * 2$	$[25] * 2$	uniform
SP-CMA[20]	5	$[20] * 5$	$[10] * 2$	uniform
SP-CMA[10]	10	$[10] * 10$	$[5] * 2$	uniform
SP-CMA[MIXED]	6	[50,10,10,10,10,10]	[25,5,5,5,5,5]	uniform

Experiment 2

Table 6: Experiment 2 runs configurations. S_n = subpopulations count

Configurations				
Name	S_n	λ	μ	Initilization
`SP-CMA[50]-UNIF`	2	$[50] * 2$	$[25] * 2$	uniform
`SP-CMA[50]-GAUSSIAN`	2	$[50] * 2$	$[25] * 2$	gaussian
`SP-CMA[50]-POISSON`	2	$[50] * 2$	$[25] * 2$	poisson disk
`SP-CMA[50]-HALTON`	2	$[50] * 2$	$[25] * 2$	halton
`SP-CMA[50]-SOBOL`	2	$[50] * 2$	$[25] * 2$	sobol
`SP-CMA[50]-LHS`	2	$[50] * 2$	$[25] * 2$	latin hypercube
`SP-CMA[20]-UNIF`	5	$[20] * 5$	$[10] * 2$	uniform
`SP-CMA[20]-GAUSSIAN`	5	$[20] * 5$	$[10] * 2$	gaussian
`SP-CMA[20]-POISSON`	5	$[20] * 5$	$[10] * 2$	poisson disk
`SP-CMA[20]-HALTON`	5	$[20] * 5$	$[10] * 2$	halton
`SP-CMA[20]-SOBOL`	5	$[20] * 5$	$[10] * 2$	sobol
`SP-CMA[20]-LHS`	5	$[20] * 5$	$[10] * 2$	latin hypercube
`SP-CMA[10]-UNIF`	10	$[10] * 10$	$[5] * 2$	uniform
`SP-CMA[10]-GAUSSIAN`	10	$[10] * 10$	$[5] * 2$	gaussian
`SP-CMA[10]-POISSON`	10	$[10] * 10$	$[5] * 2$	poisson disk
`SP-CMA[10]-HALTON`	10	$[10] * 10$	$[5] * 2$	halton
`SP-CMA[10]-SOBOL`	10	$[10] * 10$	$[5] * 2$	sobol
`SP-CMA[10]-LHS`	10	$[10] * 10$	$[5] * 2$	latin hypercube

Experiment 3

Table 7: Experiment 3 runs configurations. S_n = subpopulations count

Configurations					
Name	S_n	λ	μ	Initilization	SVM correction
`SP-CMA[50]`	2	$[50] * 2$	$[25] * 2$	uniform	none
`SP-CMA[50]-SVMi`	2	$[50] * 2$	$[25] * 2$	uniform	intersection correction
`SP-CMA[50]-LHS-SVMi`	2	$[50] * 2$	$[25] * 2$	latin hypercube	intersection correction
`SP-CMA[50]-SVMo`	2	$[50] * 2$	$[25] * 2$	uniform	orthogonal projection
`SP-CMA[50]-LHS-SVMo`	2	$[50] * 2$	$[25] * 2$	latin hypercube	orthogonal projection
`SP-CMA[20]`	5	$[20] * 5$	$[10] * 2$	uniform	none
`SP-CMA[20]-SVMi`	5	$[20] * 5$	$[10] * 2$	uniform	intersection correction
`SP-CMA[20]-LHS-SVMi`	5	$[20] * 5$	$[10] * 2$	latin hypercube	intersection correction
`SP-CMA[20]-SVMo`	5	$[20] * 5$	$[10] * 2$	uniform	orthogonal projection
`SP-CMA[20]-LHS-SVMo`	5	$[20] * 5$	$[10] * 2$	latin hypercube	orthogonal projection
`SP-CMA[10]`	10	$[10] * 10$	$[5] * 10$	uniform	none
`SP-CMA[10]-SVMi`	10	$[10] * 10$	$[5] * 10$	uniform	intersection correction
`SP-CMA[10]-LHS-SVMi`	10	$[10] * 10$	$[5] * 10$	latin hypercube	intersection correction
`SP-CMA[10]-SVMo`	10	$[10] * 10$	$[5] * 10$	uniform	orthogonal projection
`SP-CMA[10]-LHS-SVMo`	10	$[10] * 10$	$[5] * 10$	latin hypercube	orthogonal projection

Experiment 4

Table 8: Experiment 4 runs configurations. S_n = subpopulations count

Configurations						
Name	S_n	λ	μ	buffer t	rate r	mode
`SP-CMA[50]`	2	$[50] * 2$	$[25] * 2$	n/a	n/a	n/a
`SP-CMA[50]-RA(500x.1)`	2	$[50] * 2$	$[25] * 2$	500	0.1	none
`SP-CMA[50]-IPOP(500x.1)`	2	$[50] * 2$	$[25] * 2$	500	0.1	ipop
`SP-CMA[50]-RA(500x.2)`	2	$[50] * 2$	$[25] * 2$	500	0.2	none
`SP-CMA[50]-IPOP(500x.2)`	2	$[50] * 2$	$[25] * 2$	500	0.2	ipop
`SP-CMA[50]-RA(1000x.1)`	2	$[50] * 2$	$[25] * 2$	1000	0.1	none
`SP-CMA[50]-IPOP(1000x.1)`	2	$[50] * 2$	$[25] * 2$	1000	0.1	ipop
`SP-CMA[50]-RA(1000x.2)`	2	$[50] * 2$	$[25] * 2$	1000	0.2	none
`SP-CMA[50]-IPOP(1000x.2)`	2	$[50] * 2$	$[25] * 2$	1000	0.2	ipop
`SP-CMA[20]`	5	$[20] * 5$	$[10] * 5$	n/a	n/a	n/a
`SP-CMA[20]-RA(500x.1)`	5	$[20] * 5$	$[10] * 5$	500	0.1	none
`SP-CMA[20]-IPOP(500x.1)`	5	$[20] * 5$	$[10] * 5$	500	0.1	ipop
`SP-CMA[20]-RA(500x.2)`	5	$[20] * 5$	$[10] * 5$	500	0.2	none
`SP-CMA[20]-IPOP(500x.2)`	5	$[20] * 5$	$[10] * 5$	500	0.2	ipop
`SP-CMA[20]-RA(1000x.1)`	5	$[20] * 5$	$[10] * 5$	1000	0.1	none
`SP-CMA[20]-IPOP(1000x.1)`	5	$[20] * 5$	$[10] * 5$	1000	0.1	ipop
`SP-CMA[20]-RA(1000x.2)`	5	$[20] * 5$	$[10] * 5$	1000	0.2	none
`SP-CMA[20]-IPOP(1000x.2)`	5	$[20] * 5$	$[10] * 5$	1000	0.2	ipop
`SP-CMA[10]`	10	$[10] * 10$	$[5] * 2$	n/a	n/a	n/a
`SP-CMA[10]-RA(500x.1)`	10	$[10] * 10$	$[5] * 2$	500	0.1	none
`SP-CMA[10]-IPOP(500x.1)`	10	$[10] * 10$	$[5] * 2$	500	0.1	ipop
`SP-CMA[10]-RA(500x.2)`	10	$[10] * 10$	$[5] * 2$	500	0.2	none
`SP-CMA[10]-IPOP(500x.2)`	10	$[10] * 10$	$[5] * 2$	500	0.2	ipop
`SP-CMA[10]-RA(1000x.1)`	10	$[10] * 10$	$[5] * 2$	1000	0.1	none
`SP-CMA[10]-IPOP(1000x.1)`	10	$[10] * 10$	$[5] * 2$	1000	0.1	ipop
`SP-CMA[10]-RA(1000x.2)`	10	$[10] * 10$	$[5] * 2$	1000	0.2	none
`SP-CMA[10]-IPOP(1000x.2)`	10	$[10] * 10$	$[5] * 2$	1000	0.2	ipop

Experiment 5

Table 9: Experiment 5 runs configurations

Name	Configuration Settings
ModCMA	see Table 5
ModCMA-[0]	see Table 5
SP-CMA[50]	see Table 5
SP-CMA[50]-LHS	see Table 6
SP-CMA[50]-SVMi	see Table 7
SP-CMA[50]-LHS-SVMi	see Table 7
SP-CMA[50]-SVMo	see Table 7
SP-CMA[50]-LHS-SVMo	see Table 7
SP-CMA[20]	see Table 5
SP-CMA[20]-LHS	see Table 6
SP-CMA[20]-SVMi	see Table 7
SP-CMA[20]-LHS-SVMi	see Table 7
SP-CMA[20]-SVMo	see Table 7
SP-CMA[20]-LHS-SVMo	see Table 7
SP-CMA[10]	see Table 5
SP-CMA[10]-LHS	see Table 6
SP-CMA[10]-SVMi	see Table 7
SP-CMA[10]-LHS-SVMi	see Table 7
SP-CMA[10]-SVMo	see Table 7
SP-CMA[10]-LHS-SVMo	see Table 7
SP-CMA[20]-RA(500x.2)	see Table 8
SP-CMA[20]-RA(1000x.1)	see Table 8
SP-CMA[10]-RA(500x.2)	see Table 8
SP-CMA[10]-RA(1000x.1)	see Table 8