Benchmarking SMAC and MIES on Mixed-Integer Problems

Name: Qinchen Shi
Date: 06/03/2018
1st supervisor: Prof. Thomas Bäck
2nd supervisor: Dr. Kaifeng Yang

1 Introduction

The mixed integer programming (MIP) stems from various real-world problems and has deeply rooted in the research of mathematical optimization / operational research. In order to effectively tackle such problems in practice, many well-designed mathematical / heuristical algorithms have been proposed in the literature. Depending on the form of the problem, it can be further categorized into Mixed integer non-linear programming (MINLP) and mixed integer linear programming. In order to solve such problems, many optimization techniques are developed. For instance, two optimization algorithms, Bayesian Optimization (BO) and Mixed-integer Evolution Strategy (MIES) have been developed independently in the field of machine learning and evolutionary computation respectively. In the machine learning community, Bayesian optimization approaches such SMAC is devised to handle the Algorithm Configuration (AC) task, while the evolutionary algorithm community has developed methods such as the MIES (mixed-integer evolution strategy) for such applications. Although these methods serve the same purpose, there is no systematic empirical comparison on them. This thesis aims at bridging this gap and identifying pros/cons of the considered algorithms, through empirical comparisons of the performance of the algorithms on some selected test functions.

2 Optimization Problems

The optimization problems are divided into continuous optimization problems and combinatorial optimization problems. In continuous optimization problems, the search variable usually takes values in a subset of $\mathbb{R}^n$, where n is the dimensionality. The combinatorial optimization problem deals with discrete search spaces.

2.1 Continuous Optimization Problems

Continuous optimization problems can often be described as: Let S be a subset of $\mathbb{R}^n$, that is, a domain of real-valued variables and $f : S \to R$ be a real-valued function. The global minimum of the function f in the S domain is defined as follows: $\forall X \in S : f(X_{min}) \leq f(X)$. The performance comparison of the algorithm is usually based on some typical problems called benchmark. The commonly used benchmark problem problems are::(1) Sphere Model (2) Schwefel's Problem (3) Rosenbrock's Function. Given the constraints of many engineering problems, the optimization of constrained functions has also been a major concern in the optimization field. For the constrained problem, in addition to the existence of a local minimum, the factors that affect the optimal performance mainly include: (1) The properties the function landscape, the linear or convex function is easier to solve than the irregular function. (2) The degree of density in the feasible region is usually measured by the ratio of the feasible region to the entire search space.

2.2 Combinatorial Optimization Problem

The combinatorial optimization problem can usually be described as follows: Let $\Omega = s_1, s_2, ..., s_n$ be the solution space formed by all states, $C(s_i)$ is the value of the objective function corresponding to the state s_i. It is required to find the optimal solution s^*, so that $\forall s_i \in \Omega, C(s^*) = minC(s_i)$. Combinatorial optimization often involves issues such as sorting, classification, and screening. It is a branch of operations research. Typical combinatorial optimization problems include Traveling salesman problem (TSP), Scheduling problem (eg, Flow-shop, Job-shop), Knapsack problem (0-1), Packing problem (Bin Packing Problem), Graph coloring problem, Clustering problem, etc. Obviously, the above problems are not complicated to describe and they are strong engineering representations, but the optimal solution is quite difficult to reach, the main reason is the so-called "combination explosion".

For example, clustering problems, there are n patters on m-dimendional space $\{X_i|i = 1, 2, \ldots, n\}$ which is required to cluster into k, so that the points within each cluster are most closed, for instance: $\chi^2 = \sum_{i=1}^{(p)} \parallel X_i^{(p)} - R_p \parallel$ where R_p is the amount of points in p cluster. And the traveling salesman problem (TSP), given the distance between n cities and a pair of cities, it is required to determine the shortest path that passes through each city once and only once. The Job-shop problem is a more complex typical process scheduling problem than TSP, and it is a simplified model of many practical problems. A Job-shop can be described as n workpieces machined on m machines, O_{ij} represents the operation of the i-th workpiece on the j-th machine, the corresponding operation time T_{ij} is known. Predetermining the processing sequence of each workpiece on each machine (called technical constraints), requiring the determination of the processing sequence of all workpieces on each machine that are compatible with the technical constraints, so as to optimize the processing performance index, usually it means the minimum value of makespan. In the Job-shop problem, in addition to technical constraints, it is generally assumed that each machine can only machine one workpiece at a time, and each workpiece can only be processed by one machine, and the processing process is continuous. If the technical constraints of the parts are the same, a Job-shop problem changes into a simple Flow-shop problem. Furthermore, if the processing order of the workpieces on each machine is also the same, the problem is further transformed into a replacement Flow-shop problem.

So, there are $kn/k!$ possible division methods for clustering problems, $(n!)m$ possible job-shop arrangements, and n-city TSP problems based on permutation arrangement descriptions, even for non-directional and cyclical plane problems, there are still $(n - 1)!/2$ different permutations. Obviously, the number of states increases exponentially with the scale of the problem. Therefore, the key to solving these problems lies in designing efficient heuristic solvers that approaches the optimum asymptotically.

2.3 Black-Box Optimization

For many real-world mixed-integer optimization problems, no analytical expression or assumption (e.g., convexity) can be given on them. Consequently, the most of mathematical optimization techniques are rendered inapplicable. In this situation, the so-called black-box optimization technique is quite suitable.

The black-box optimization problem is a special category of optimization problems based on three components: decision variables, constraints, and objective functions. Some practical problems are commonly treated as black-boxes: numerical code involving partial differential equations (PDE), integrals and crash tests and chemical reactions in experiments. The cost of this evaluation process is often expensive and time-consuming. When optimizing the black box problem, it is to construct a surrogate model over the problem to participate in the optimization process in order to reduce the evaluation cost. Assume that the decision variables is $x = (x_1, x_2, \ldots, x_n)^T \in R^n$, the constraints is

$$S = \{x | g_i(x) \leq 0, i = 1, 2, \ldots, m; h_j(x) = o, i = 1, 2, \ldots, l\}$$

Then the black-box optimization problem can be presented as follows:

$$\text{minimize} \quad f(x)$$

$$\text{s.t.} \quad x \in S$$

where $f(x)$ is the objective function; $S \in R^n$ is the feasible region.

2.4 Taxonomy on Optimization Techniques

The so-called optimization algorithm, in fact, is a search process or rule, it is based on a certain idea and mechanism, through a certain path or rule to get the solution to meet the user's requirements. Divided by the optimization mechanism and behavior, the current optimization algorithms commonly used in the project can be divided into classical algorithms, structural algorithms, improved algorithms, algorithms based on system dynamic evolution and hybrid algorithms.

The classic algorithm includes linear programming, dynamic programming, integer programming and branch-and-bound method and the traditional algorithms in operations research. The computational complexity of the classic algorithm is generally very large, and it is only suitable for solving small-scale problems. It is often not applicable in engineering issues.

The constructive algorithm uses the method of construction to quickly establish the solution to the problem. Usually, the quality of the constructive algorithm is poor and difficult to meet the engineering requirements. For example, the typical construction methods in the scheduling problem include Johnson method, Palmer method, Gupta method, CDS method, Daunenbring's rapid approach method, NEH method, etc.

The improved algorithm, or domain search algorithm. From any solution, optimization is performed by the continuous search of its domain and replacement of the current solution. According to the search behavior, it can be divided into local search method and guided search method.

- Local search method: Greedy search in the field of current solution with local optimization strategy, such as only accepting the state superior to the current solution as the climbing method of the next current solution etc. that accepts the best solution in the current neighborhood as the next current solution
- Guided search method: Use rules to guide the exploration of good solutions in the entire solution space, such as SA, GA, EP, ES, and TS.

The method in the basis of the dynamic evolution of the system is to change the optimizing process into a dynamic evolutional procedure of system to accomplish the implementation of optimization, such as neural network and chaotic search.
The hybrid algorithm is the algorithm which refers to the various algorithms generated by the above algorithms from the mix of structure or operation. We took this classified approach

here, however, the optimization algorithm can certainly be classified from other perspectives, such as deterministic and uncertainty algorithms, local optimization algorithms, and global optimization algorithms.

2.5 Neighborhood Function and Local Search

The so-called optimization algorithm, in fact, is a search process or rule, it is based on a certain idea and mechanism, through a certain path or rule to get the solution to meet the user's requirements. Divided by the optimization mechanism and behavior, the current optimization algorithms commonly used in the project can be divided into classical algorithms, structural algorithms, improved algorithms, algorithms based on system dynamic evolution and hybrid algorithms.

Neighborhood function is an important concept in optimization. Its function is to guide how to generate a (group) new solution from a (group) solution. The design of neighborhood functions often depends on the feature of the problem and the way the solution is expressed (encoding). Due to the different ways of characterization of optimization states, there are obvious differences between the specific methods of function optimization and the neighborhood function in combinatorial optimization. The concept of the neighborhood function in function optimization is more intuitive, and using the concept of distance to construct neighborhood functions by adding perturbations is the most common way. For example, $x^{'} = x + \eta\xi$, where x' is a new solution, x is an old solution, η is a scale parameter, and ξ is a random number or white noise or chaotic series or gradient information that satisfies a certain probability distribution. Obviously, the use of different probability distributions (such as Gaussian distribution, Cauchy distribution, uniform distribution, etc.) or descent strategy will achieve different states of state transfer.

The local search algorithm is based on greedy thoughts and uses the neighborhood function to search. It can usually be described as starting from an initial solution, using the neighborhood function to continuously search for a solution better than it is in the neighborhood of the current solution, if such a solution can be found, it is referred to as a new current solution, and then the above process is repeated, otherwise the search process is ended and the current solution is used as the final solution. It can be seen that although the local search algorithm has the characteristics of general and easy implementation, the search performance completely depends on the neighborhood function and the initial solution. If the domain function is not properly designed or the initial value is not suitable, the final performance of the algorithm will be poor. At the same time, greed thinking will undoubtedly make the algorithm lose the ability of global optimization, that is, the algorithm can not avoid falling into a local minimum during the search process. Therefore, if people do not improve on the search strategy, then to achieve global optimization, the local function used by the local search algorithm must be "complete", the neighborhood function will lead to a complete enumeration of the solution. However, it cannot be achieved in most cases, and the exhaustive method is not allowed for large-scale search time. In view of the above shortcomings of local search algorithms, intelligent optimization algorithms, such as SMAC and MIES, use different search mechanisms and strategies from different perspectives to improve the local search algorithm to achieve better global optimization performance.

3 Mixed-integer Black-box Optimization Application

We can take a look at a representative real-world optimization task: optimization of multilayer optical coating which is of remarkable importance in a number of application fields such as optical and scientific instrumentation manufacturing, spectroscopy, medicine, and astronomy. This multilayer system is formed by a set of layers to separate air and a substrate and the permittivity and conductivity of the layers depend on one spatial coordinate perpendicular to the layer-media boundaries. Generally, one plane electromagnetic wave enters the multilayer system from the air, it will be partially selected or transmitted at the borders between layers of different refractive indices. So, the intention of the multilayer optical coating is to find a certain sequence of layers of certain materials and certain thickness that all undesirable frequencies are abandoned while the wanted frequencies go through without any reflection. The involved parameters mainly are: the number n of layers, their refractive indices $\vec{\eta} = (\eta_1, \ldots, \eta_n)$ of n layers and the thickness $\vec{d} = (d_1, \ldots, d_n)$ of the n layers. People took black-box optimization for MOCs design in result of its complexity, first, it contains real-valued thickness and integer-valued refractive indices variables, second, the dimension of decision variables are quite high, third, the equation which is used to compute objective values are complicated, forth, the number of dimensions is variable in the most general formulation of this problem. The detailed formulations are described in [1].

4 SMAC

In this section, we give a brief introduction to the SMAC (Sequential Model-based Algorithm Configuration), it is an instantiation of the general Sequential Model-Based Optimization (SMBO) [2] framework. The SMBO [3] uses a regression model, also called a response surface model, that predicts the objective value at unknown points and can be used for optimization. In addition, SMBO iteratively proposes candidate points for evaluation and update (re-train) the regression model after incorporating the newly evaluated point into the data set. In this thesis, the underlying data set is denoted as: $H = (x_1, f(x_1), \ldots, (x_n, f(x_n)))$. In order to propose candidate point in each iteration, some utility functions are defined over the regression model. Such functions typically balances the exploration and exploitation in the optimization. Some commonly used functions are: probability of improvement, expected improvement (EI) and upper confidence bound. In SMAC, EI is taken as the default option and a new candidate point is generated through the following maximization task: EI is used to recommend a new x^*.

$$x^* = \arg\max_{x \in S} \mathrm{EI}(x).$$

For this new recommendation x^*,SMAC first evaluates it and then add $(x^*, f(x^*))$ to domain H, then update the underlying regression model, and repeat the above procedure to acquire new x^*. The precised process are demonstrated as below: f denotes the objective function,Θ is a list of configurations. SMBO has its roots in the statistics literature on experimental design for the global (black-box) function optimization. The most notable algorithm in this category is the so-called efficient global optimization [4] (EGO) algorithm by Jones et al. [12], however, limited to optimizing continuous parameters for noise-free functions.

Commonly, Gaussian processes are used as the regression model, which requires the specification of the kernel function $k : \Theta \times \Theta \longmapsto R^+$ with specifying the similarity between two

Algorithm 1 *SMBO(f,T,S)*

```
1: $H \leftarrow \Theta$
2: for $t \leftarrow 1$ to $T$ do
3:     $x^* \rightarrow \arg\max_{x \in S} \mathrm{EI}(x)$
4:     Evaluate $f(x^*)$
5:     $H \leftarrow \cup(x^*, f(x^*))$
6:     fit a new model $\mathrm{EI}(x)$ to $H$
7: end for
```

candidate solutions. This approach was utilized in most recent sequential model-based work, including classic SMBO when it deals with numerical parameters:

$$k(\theta_i, \theta_j) = \exp[\sum_{l=1}^{d}(-\lambda_l \cdot (\theta_{i,l} - \theta_{j,l})^2)], \tag{1}$$

where $\lambda_1, \ldots, \lambda_d$ are the so-called kernel parameters.
For the categorical parameters, which measures the weighted Hamming distance instead of the weighted euclidean distance:

$$k_{cat}(\theta_i, \theta_j) = \exp[\sum_{l=1}^{d}(-\lambda_l \cdot [1 - \delta(\theta_{i,l}, \theta_{j,l})])], \tag{2}$$

where δ is the Kronecker delta function (which equals one if two arguments are identical, otherwise zero).
For a combination of continuous parameters P_{cont} and categorical parameters P_{cat}, SMAC also defines a combined kernel

$$K_{mixed}(\theta_i, \theta_j) = \exp[\sum_{l \in P_{cont}}(-\lambda_l \cdot (\theta_{i,l} - \theta_{j,l})^2) + \sum_{l \in P_{cat}}(-\lambda_l \cdot [1 - \delta(\theta_{i,l}, \theta_{j,l})])].$$

In view that Gaussian stochastic processes are with a similar pattern to kernel-based learning methods and the validation of K_{mixed} function, it can be exchanged for Gaussian kernel without altering any other element of GP construction.

Random forests [5] is an ensemble learning algorithm. The most significant premise of the random forests is that building a small, weak decision tree with few features is a computationally cheap process. so if we can build many small decision trees in parallel, then we can acquire a singer, strong through averaging and majority vote. Random forests are found to be the most accurate learning algorithm to date. Algorithm 2 is the detailed process of random forest.

The algorithm works as follows: for each tree in the forest, we select a bootstrap sample from S where $S^{(i)}$ denotes the ith bootstrap. We then learn a decision-tree using a modified decision-tree learning algorithm. The algorithm is modified as follows: at each node of the tree, instead of examining all possible features-splits, we randomly select some subset of the features $f \subseteq F$, where F is the set of features. The node then splits on the best feature in f rather than F. In practice, f is much much smaller than F. Deciding on which features to split is oftentimes the most computationally expensive aspect of decision tree learning. By

Algorithm 2 Random Forest

```
Precondition: A training set $S := (x_1, y_1), \dots, (x_n, y_n)$, features $F$, and number of trees
    in forest $B$.
 1: function RANDOM FOREST($S, F$)
 2:     $H \leftarrow \varnothing$
 3:     for $i \in 1, \dots, B$ do
 4:         $S^{(i)} \leftarrow$ A bootstrap sample from $S$
 5:         $h_i \leftarrow$ RANDOMIZEDTREELEAN ($S^{(i)}, F$)
 6:         $H \leftarrow H \cup \{h_i\}$
 7:     end for
 8:     return $H$
 9: end function
10:
11: function RANDOMIZEDTREELEAN($S, F$)
12:     At each node:
13:     $f \leftarrow$ very small subset of $F$
14:     Split on best feature in $f$
15:     return The learned tree
16: end function
```

narrowing the set of features, we drastically speed up the learning of the tree. The random forest algorithm uses the bagging technique for building an ensemble of decision trees. Bagging is known to reduce the variance of the algorithm.

Because SMAC originally aims at minimizing the runtime of computer programs and it has been discovered in [3] that the logarithmic transformation has an advantage in substantial improving model quality, thus SMAC took this method. In this study, we only consider some commonly used mixed-integer objective function other that the actual running time of some computer codes. Therefore, the logarithmic transformation is not applied in our experiments.

5 MIES

Evolution Strategies (ES) [6] are a branch of Evolutionary Algorithms (EA) [7], and have been successfully applied to various real-world applications. Although ES has been successfully applied in many applications, it has encountered challenges, one of which is the heterogeneity of the decision variables. There are real-world optimization problems, whose decision variables are of different types. This kind of optimization problems is called mixed-integer optimization problems [8]. It usually contains continuous variables, integer variables and discrete variables simultaneously. Mixed -integer evolution strategies [9] (MIES) is proposed by Emmerich et. al. to optimize the Hydrodealkylation (HDA) process simulators which is a mixed-integer optimization problem in the chemical plant design. MIES can deal with different variable types simultaneously. In [10] [11] [12] MIES have been employed to successfully solve mixed-integer optimization problems occurring in optical filter design, HDA process simulators for chemical plant design and image analysis agent for intravascular ultrasound image analysis.

A mixed-integer global optimization problem can be defined as follows [13]:

$$f(r_1,\dots,r_{n_r},z_1,\dots,z_{n_z},d_1\dots,d_{n_d})\rightarrow min$$

$$subject \quad to:$$

$$r_i\in[r_i^{min},r_i^{max}]\subset R, i=1,\dots,n_r$$

$$z_i\in[z_i^{min},z_i^{min}]\subset Z, i=1,\dots,n_z$$

$$d_i\in D_i=d_{i,1},\dots,d_{i,|D_I|}, i=1,\dots,n_d$$

Here, r denotes the continuous variables, z are integer variables, and d are the nominal discrete variables, where the subscript indexes each type of the variable. n_r, n_z and n_d is the number of continuous, integer, and nominal discrete variables, respectively. D_i denotes a set of discrete values. The fitness function is denoted by f. An individual in Evolution Strategies is denoted as [10]:

$$\alpha=(r_1,\dots,r_{n_r},z_1,\dots,z_{n_z},d_1\dots,d_{n_d},\sigma_1,\dots,\sigma_{n_\sigma},\varsigma_1,\dots,\varsigma_{n_\varsigma},\rho_1,\dots,\rho_{n_\rho})$$

The parameters $r_1,\dots,r_{n_r}$, $z_1,\dots,z_{n_z}$, $d_1,\dots,d_{n_d}$ are called simply parameters, correspond to the variables of mixed-integer optimization, while $\sigma_1,\dots,\sigma_{n_\sigma}$, $\varsigma_1,\dots,\varsigma_{n_\varsigma}$, $\rho_1,\dots,\rho_{n_\rho}$ are called strategy parameters for Evolution Strategies. $\sigma_1,\dots,\sigma_{n_\sigma}$ are the step size for continuous variables, $\varsigma_1,\dots,\varsigma_{n_\varsigma}$ are step size for integer values and $\rho_1,\dots,\rho_{n_\rho}$ are the mutation probabilities for discrete values.

There are two recombination operators in ES: discrete recombination, sometimes also referred to as dominant recombination and the intermediate recombination. In this paper, we adopted dominant recombination for object parameters and intermediate recombination [6] for the strategy parameters. For each object parameter of offspring individual, dominant recombination chooses the object parameter from parents with an equal probability. By contrast, for each strategy parameter of the offspring individual, intermediate recombination calculates the mean of the strategy parameter from the parents.

Different variable types need different mutation operators. To make mutation operator suited for mixed-integer optimization problems, Emmerich et. al. proposed a new mutation operator in [10]. This mutation operator is combined with the standard mutations for continuous, integer and nominal discrete, as described in [14] [15] [16]. Algorithm 3 presents the detail of the mutation, where τ_g denotes the global learning rate and τ_l the local learning rate. The recommended settings [10] are $\tau_l = 1/\sqrt{2\sqrt{n_r}}$ and $\tau_g = 1/\sqrt{2n_r}$. $U(0,1)$ denotes uniform distribution and $N(0,1)$ denotes the standardized normal distribution. In case of continuous variables, the new individual is acquired by adding a normally distributed perturbation to old values of vectors, the related standard deviations are resisted by the evolution process and are thus multiplied in each step with a logarithmically distributed random number, this process can be considered as self-adaptive. while in integer case, the normally distributed variables are altered by difference between two geometrically distributed variables and the self-adaption is utilized to control the width parameters with a global learning rate and a local learning rate. The discrete mutation is carried out with a mutation probability $p^{'}$, it can be regarded as a strategy parameter for each discrete variable, each new value is chosen randomly (uniformly distributed) from a finite domain D_i. The discrete self-adaption of the mutation probability

Algorithm 3 Mutation operators in MIES

```
1: $N_g \leftarrow N(0,1)$ {generate and store a normally distributed random number}
2: Mutation of continuous values
3: for $i = 1, \ldots, n_r$ do
4:     $\sigma'_i \leftarrow \sigma_i \exp(\tau_g N_g + \tau_l N(0,1))$ {a normally distributed random number}
5:     $r'_i = T^r_{r_i^{min}, r_i^{max}}(r_i + N(0, \sigma'_i))$
6: end for
7: Mutation for integer values
8: for $i = 1, \ldots, n_z$ do
9:     $\varsigma'_i \leftarrow max(1, \varsigma_i \exp(\tau_g N_g + \tau_l N(0,1)))$
10:    $u_1 = U(0,1); u_2 = U(0,1)$ {generate a uniformly distributed random number}
11:    $p = 1 - \frac{\varsigma_i / n_z}{1 + \sqrt{1 + (\frac{\varsigma_i}{n_z})^2}}$
12:    $G_1 = \lfloor \frac{ln(1-u_1)}{ln(1-p)} \rfloor; G_2 = \lfloor \frac{ln(1-u_2)}{ln(1-p)} \rfloor$; {two geometrically distributed varibales was generated}
13:    $z'_i = T^z_{z_i^{min}, z_i^{max}}(z_i + G_i - G_2)$
14: end for
15: Mutation for discrete values
16: $p' = 1/[1 + \frac{1-p}{p} * \exp(-\tau_l * N(0,1))]$
17: $p' = T_{1/(3n_d), 0.5}(p')$
18: for $i \in \{1, \ldots, n_d\}$ do
19:     if $U(0,1) < p'_i$ then
20:         choose a new element uniformly distributed out of $D_i \backslash \{d_i\}$
21:     end if
22: end for
```

is achieved by a logistic mutation of discrete parameters, generating new probabilities in the feasible domain. We recommend employing a second transformation function that keeps the value of ρ in the interval $[1/(3n_d), 0.5]$. The upper bound of 0.5 for the mutation probability is motivated by the observation that the mutation loses its causality once the probability exceeds the value of about 0.5. The lower bound is used to prevent the mutation probability from being too close to 0, in which case the MIES becomes insensitive to changes of that parameter.

To keep the parameters in their feasible interval, $T_{[a,b]}$ is a transformation function for integer parameters [10]. Given a step size of the mutation, we may consider this to be the length of a particle to travel in the interval. Starting in the direction of the unbounded mutation, when it meets the boundary, the direction is inverted until the entire length of unbounded mutation has been covered. Transformation makes sure that the values are within the boundaries. The details of the transformation are shown in [10].

6 Benchmarking

6.1 MILP and MINLP test functions

The aim of this section is to briefly describe the MILP and MINLP we applied and the reason for choosing mixed integer programming as target problems. Thus, we basically state several fields that the mixed integer linear or non-linear optimization are noticed: production planning, sequencing problems, scheduling problems, allocation problems, distribution and logistics problems, etc. However, above instances are not consummate with only mentioning the typical real-world problems in process industries, while in facts, many applications occur in other business and industries as well.

In consideration of above possibility, in our mixed integer linear and nonlinear testing suits, six classic mixed integer algorithms were chosen from literature as basis of the research. The number of parameters ranges from two to eleven, involving binary categorical parameters and real parameters.

MILP and MINLP formulations have been used quite successful in modeling a variety of optimization problems that occur in process design and synthesis (see Kocsis and Grossmann, 1988, and Floudas et al., 1989).This test case, taken from Kocis and Grossmann(1988), involves one binary variable and one continuous variable, and it is a linear problem. The formulation is given by:

$$\begin{array}{l} f_1 = 2r_1 + d_1 \rightarrow min \\ s.t. \quad r_1 \in [0, 1.6], d_1 \in \{0, 1\} \end{array} \tag{3}$$

This test case, taken from Kocis and Grossmann(1988), involves three binary variables and two continuous variables. The formulation is given by:

$$\begin{array}{l} f_2 = 2r_1 + 3r_2 + 1.5d_1 + 2d_2 - 0.5d_3 \rightarrow min \\ s.t \quad r_1^2 + d_1 = 1.25, r_2^{1.5} + 1.5d_2 = 3, \\ r_1 + d_1 \leq 1.6, 1.333r_2 + d_2 \leq 3, -d_1 - d_2 + d_3 \leq 0, \\ with \quad r_{1,2} \in [1, 10], d_{1,2,3} \in \{0, 1\} \end{array} \tag{4}$$

This test case is from Floudas (1995). The formulation involves two continuous variables and only one binary variable. r_2 is not the component of objective function but resisted in the constraints.

$$\begin{array}{l} f_3 = 0.8 + 5(r_1 - 0.5)^2 - 0.7d_1 \rightarrow min \\ s.t. \quad - \exp(r_1 - 0.2) - r_2 \leq 0, r_2 + 1.1d_1 \leq -1, r_1 - 1.2d_1 \leq 0, \\ with \quad r_1 \in [0.2, 1], r2 \in [-2.22554, -1], d_1 \in \{0, 1\} \end{array} \tag{5}$$

This test case was proposed by Yuan el. al. (1988) and involves three continuous variables and four binary variables. From this case, the number of categorical variables is increasing and the constraints become more and complicated. The formulation is

$$\begin{array}{l} f_4 = 6 + (r_1 - 1)^2 - (r_2 - 2)^2 - (r_3 - 3)^2 \\ -d_1 - 3d_2 - d_3 - 0.693147180559945d_4 \rightarrow min \\ s.t. \quad r_1 + r_2 + r_3 + d_1 + d_2 + d_3 \leq 5, r_1^2 + r_2^2 + r_3^2 = d_3 \leq 5.5, \\ r_1 + d_1 \leq 1.2, r_2 + d_2 \leq 1.8, r_3 + d_3 \leq 2.5, r_1 + d_4 \leq 1.2, \\ r_2^2 + d_2 \leq 1.64, r_3^2 + d_3 \leq 4.25, r_3^2 + d_2 \leq 4.64, \\ with \quad r_{1,2,3} \in [0, 10], d_{1,2,3,4} \in \{0, 1\} \end{array} \tag{6}$$

This test case in from Porn et. al. 1999 and involves two continuous variables and three binary variables, but entire categorical variables state in the constraints:

$$\begin{array}{l} f_5 = -5r_1 + 3r_2 \rightarrow min \\ s.t \quad 8r_1 - 2r_1^{1.5}r_2^2 + 11r_2 = 2r_2^2 - 2r_2^{0.5} \leq 39, r_1 - r_2 \leq 3, \\ 3r_1 + 2r_2 \leq 24, r_2 - d_1 - 2d_2 - 4d_3 = 1, d_2 + d_3 \leq 1, \\ with \quad r_1 \in [1, 10], r_2 \in [1, 6], d_{1,2,3} \in \{0, 1\} \end{array} \tag{7}$$

This test case was proposed by Yuan $el\ al.$ (1988) and involves seven continuous variables and four binary variables and a logarithmic transformation in the objective function. The formulation is

$$\begin{aligned}
&f_6 = (r_4-1)^2-(r5-2)^2-(r_6-1)^2-log(1+r_7)-(r_1-1)^2-(r_2-2)^2\\
&-(r_3-3)^2 \to min\\
&s.t. \quad r_1+r_2+r_3+d_1+d_2+d_3 \leq 5, r_6^2+r_1^2, r_2^2+r_3^2 \leq 5.5,\\
&r_1+d_1 \leq 1.2, r_2+d_2 \leq 1.8, r_3+d_3 \leq 2.5, r_1+d_4 \leq 1.2,\\
&r_5^2+r_2^2 \leq 1.64, r_6^2+r_3^2 \leq 4.25, r_5^2+r_3^2 \leq 4.64,\\
&r_4-d_1=0, r_5-d_2=0, r_6-d_3=0, r_7-d_4=0,\\
&with \quad r_{1,2,3} \in [0,10], r_{4,5,6,7} \in [0,1], d_{1,2,3,4} \in \{0,1\}
\end{aligned} \tag{8}$$

r and d describe the two types of variables: real and categorical. The experiment is conducted on MIES and SMAC with the following setting: 200 generations were set in both SMAC and MIES, in MIES setting, the λ is 10 to guarantee the 200 evaluation generations, and in SMAC, the iteration was marked 200, other algorithmic parameters are as default. 15 times of implementations have been applied in six functions separately and the presented values are the average of iterative best fitnesses and the average best fitness recorded since the beginning of of 15 times experiments. Figure 1 shows the line charts of entire six objective functions with the best fitness for each iterative generation. To enhance the comparison when the difference is small, the fitness values are plotted in the log scale. Across all six functions, MIES performed better but with much fluctuation. Figure 2 demonstrates the the best fitness found since the beginning of each run. Generally, MIES outperforms SMAC except that a tie is observed on the third test case.

6.2 Generalized Sphere Function

The generalized sphere model is an extension of a standard problem [10], this problem is relatively simple, as it is additively separable and unimodal. We can use it to gain some insights of how an algorithm behaves on rather simple problems and thus to estimate the best case behavior of the algorithm.

$$f_1(r,z,d) = \sum_{i=1}^{n_r} r_i^2 + \sum_{i=1}^{n_z} z_i^2 + \sum_{i=1}^{n_d} d_i^2 \tag{9}$$

In the above equation, r, z, d stand for real variables, integer variables and categorical variables. i is the index and n_r, n_z and n_d are the number of three types of variables. The constraints fof the real and integer variable is [-19, 19], for categorical is [-19,19] as well. The target is to reach the minimum solution of f_1. Figure 3 demonstrates the best solution for 200 iterations, the objective function was benchmarked for 15 times, then we calculate the average values. Again, the function values are plotted in the log scale. During the initialization, SMAC apparently finds a better fitness value roughly before 50 generation. After that, MIES starts to outperform SMAC, with an increasingly difference. Figure 4 is the line chart for the best solution of each iteration, each experiment is set within 200 generations and run 15 times for every generation, the shown values are the arithmetic means of 15 times. After 45 generation, the mies' solution is obviously better than smac's, though smac's initialized parents are quite close to optimal, this situation also delivered in the ordered generations with strong fluctuation. Since we found the best fitnesses are discrepancy during the previous 15 times of 200 evolutions experiments,

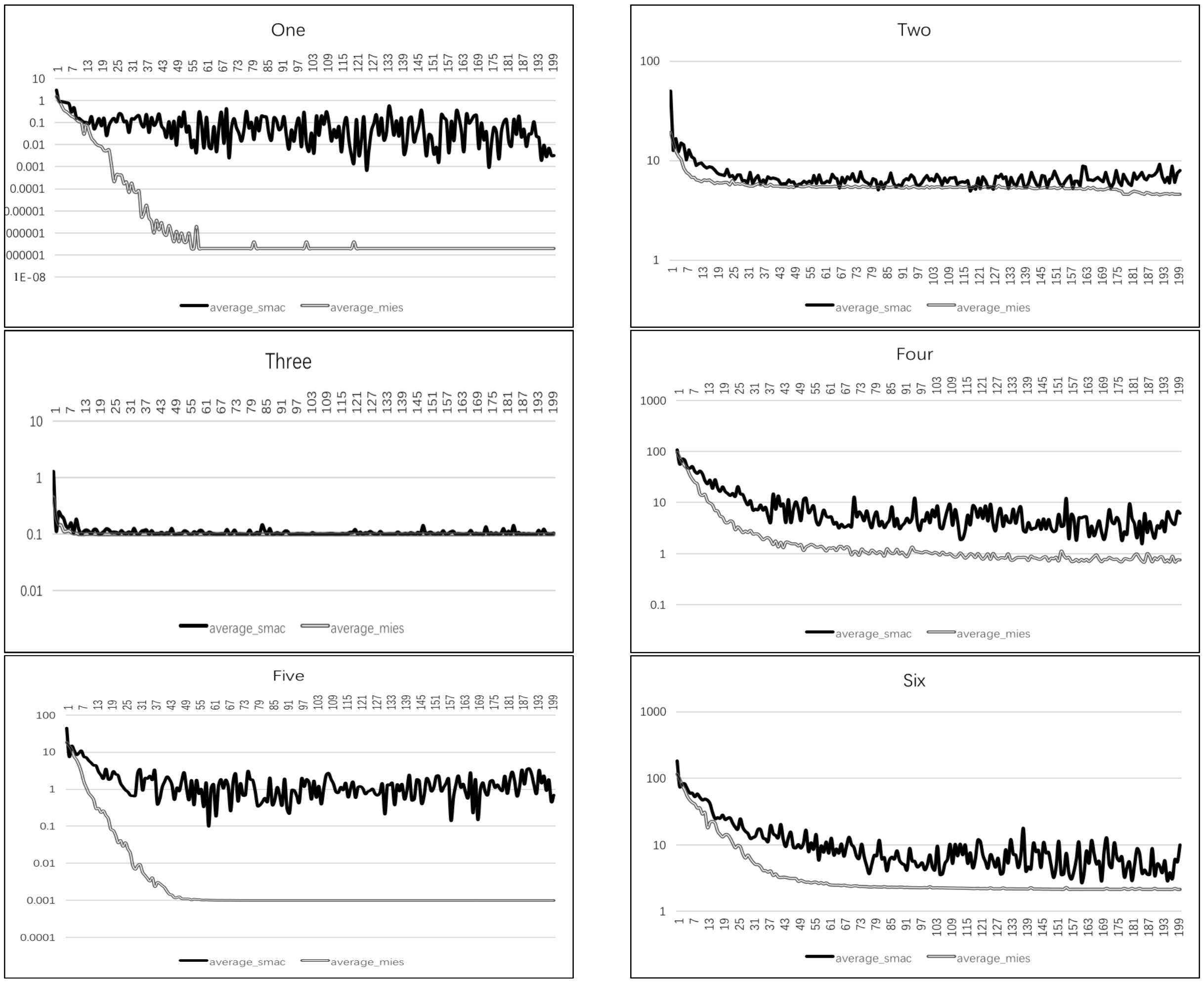

Figure 1: Average best fitness values per iteration, recorded from MIES (light gray curves) and SMAC (black curves) on 6 test functions. 15 runs are conducted.

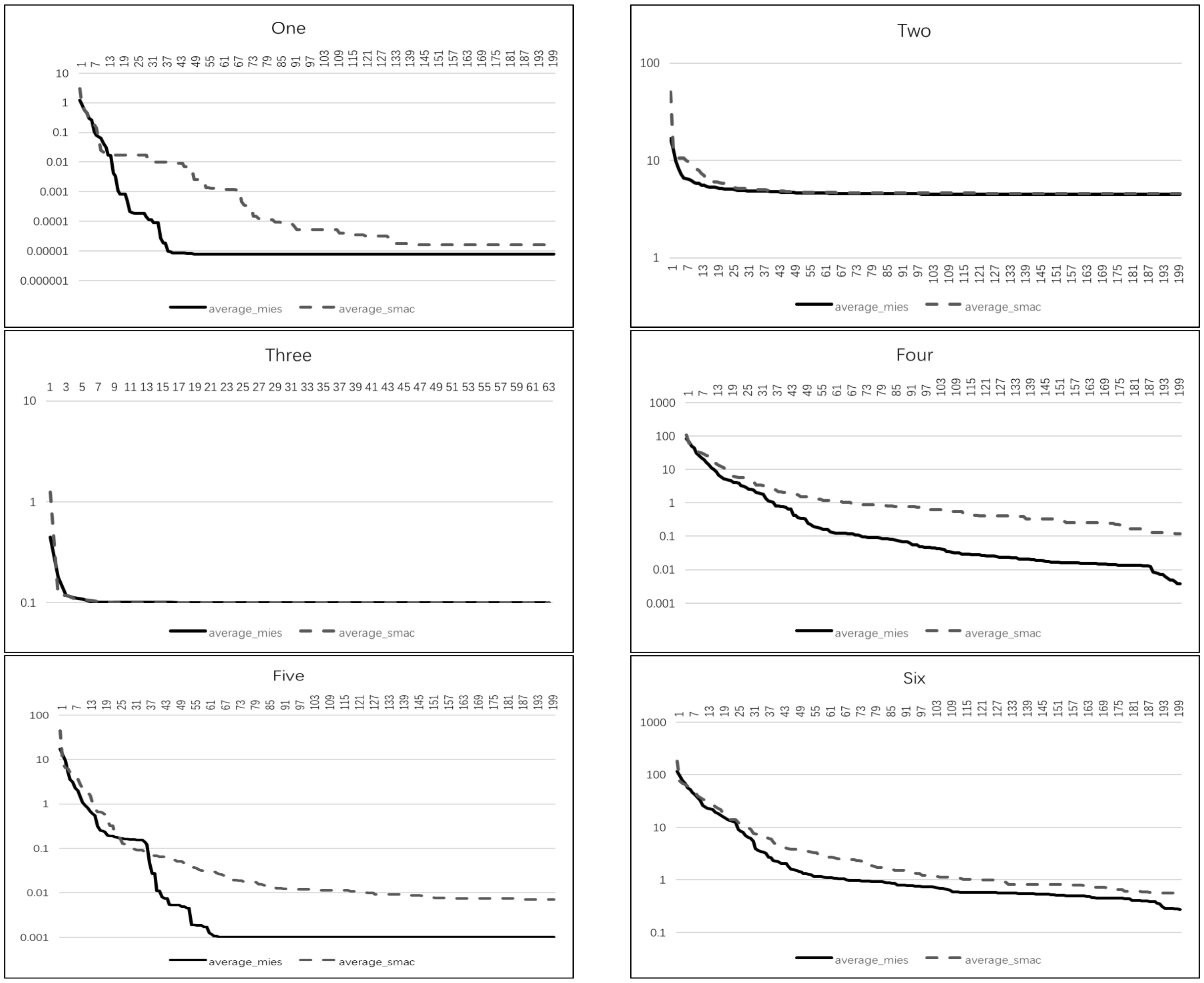

Figure 2: Average best fitnesses since the beginning recorded from MIES (real curves) and SMAC (dotted curves) on 6 test functions. 15 runs are conducted.

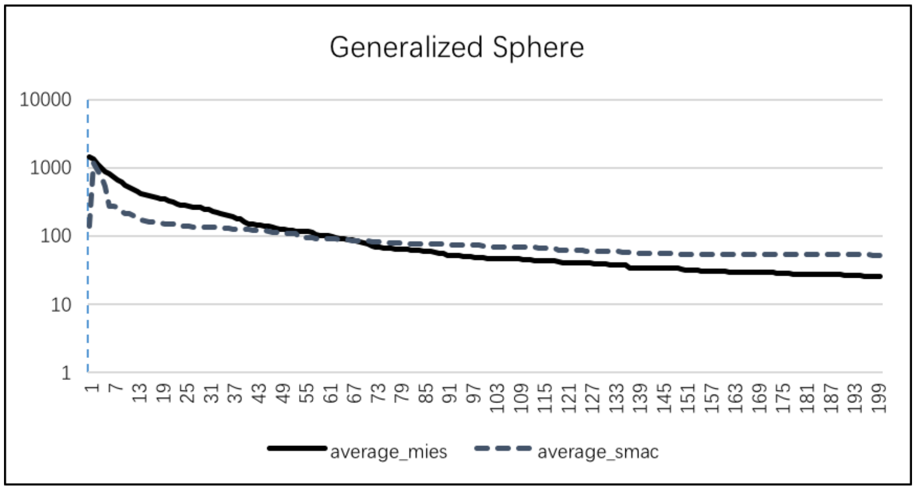

Figure 3: Average best fitnesses since the beginning recorded from MIES (real curves) and SMAC (dotted curves) on 6 test functions. 15 runs are conducted.

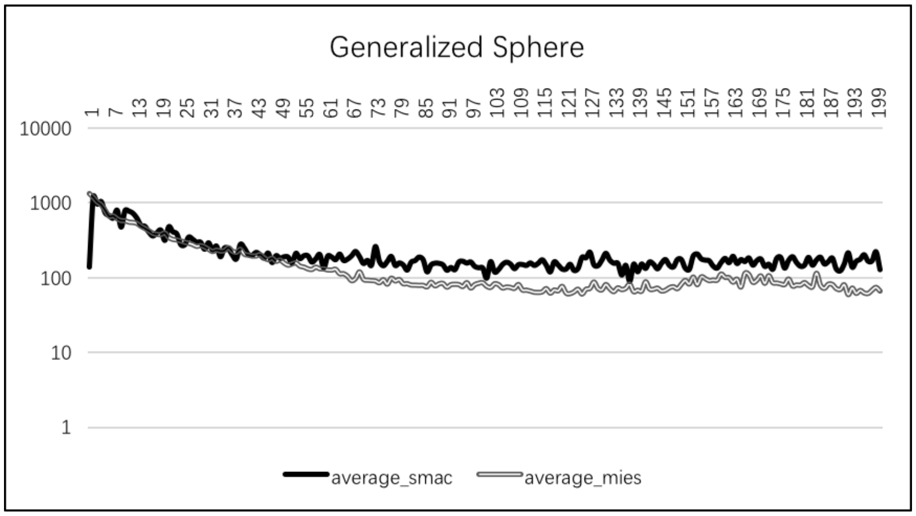

Figure 4: Average best fitness values per iteration, recorded from MIES (light gray curves) and SMAC (black curves) on 6 test functions. 15 runs are conducted.

to validate our arguments against the dispersion of the data, a boxplot has been drawn for 15 times solution of 200 generations (Figure 5-8), after analyzing we found the population of fitness values approaches the normal distribution and disperses well.

6.3 Weighted Sphere Function

The weighted sphere model represents a function with an elliptical geometry. Experiments on this function can detect if a speed up can be achieved by the learning of individual strategy parameters for each parameter. Furthermore it is an example for a function with a simple quadratic and convex geometry.

$$f_2(r, z, d) = \sum_{i=1}^{n_r} i r_i^2 + \sum_{i=1}^{n_z} i z_i^2 + \sum_{i=1}^{n_d} i d_i^2 \tag{10}$$

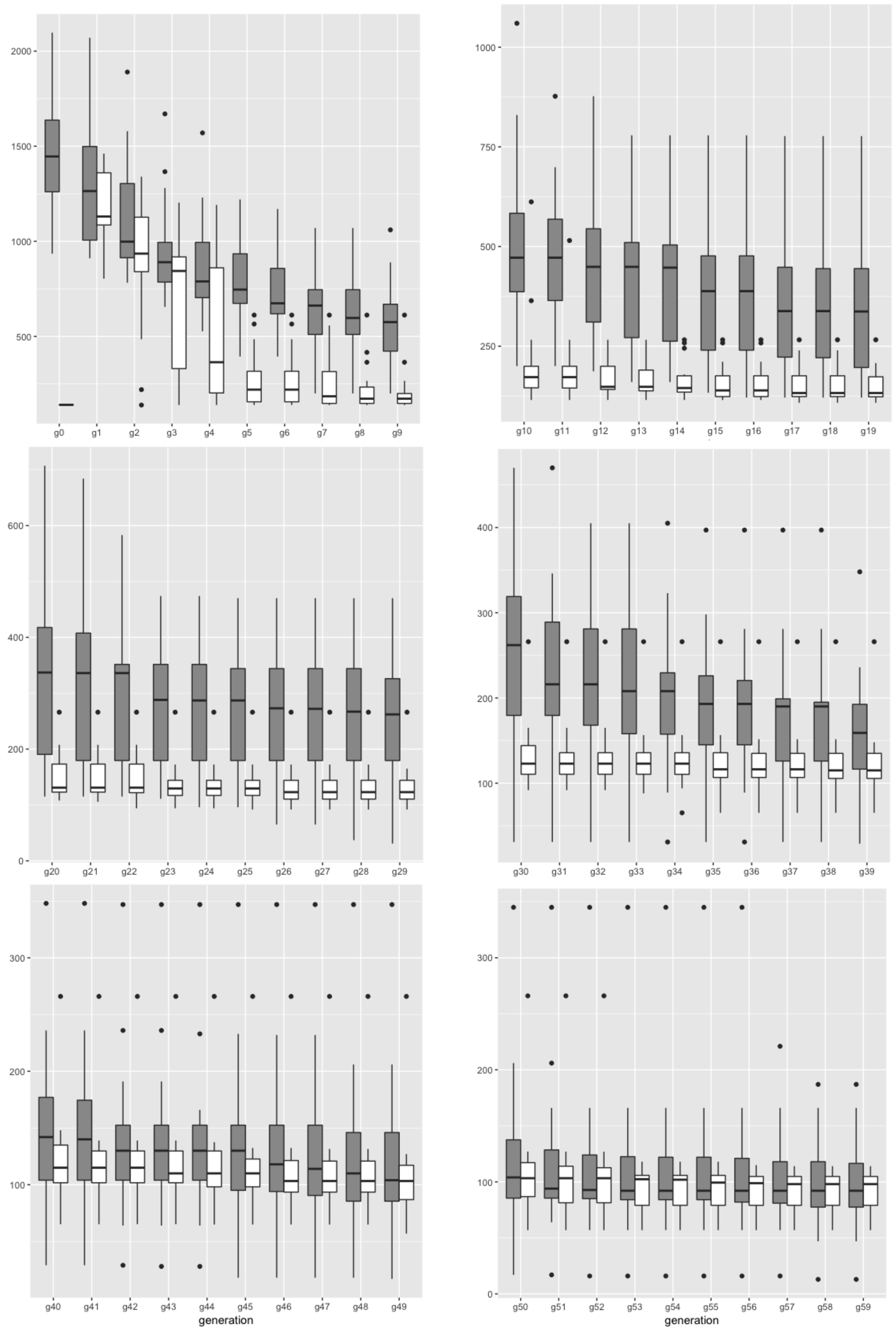

Figure 5: Boxplot for dispersion of 15 solutions of 200 generations (0-59 generation) from MIES (grey box) and SMAC (white box) for generalized sphere function

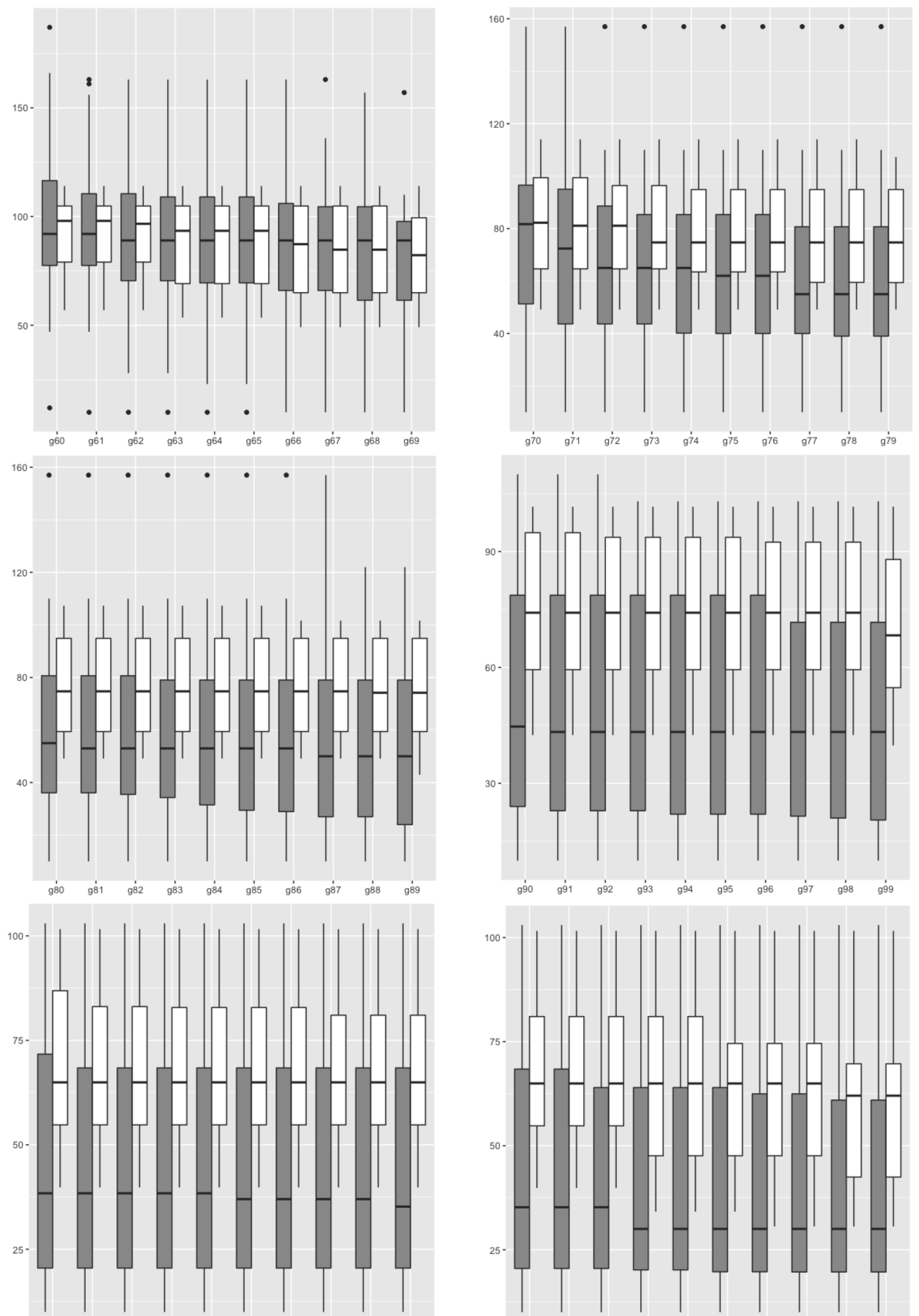

Figure 6: Boxplot for dispersion of 15 solutions of 200 generations(60-119 generation) from MIES (grey box) and SMAC (white box) for generalized sphere function

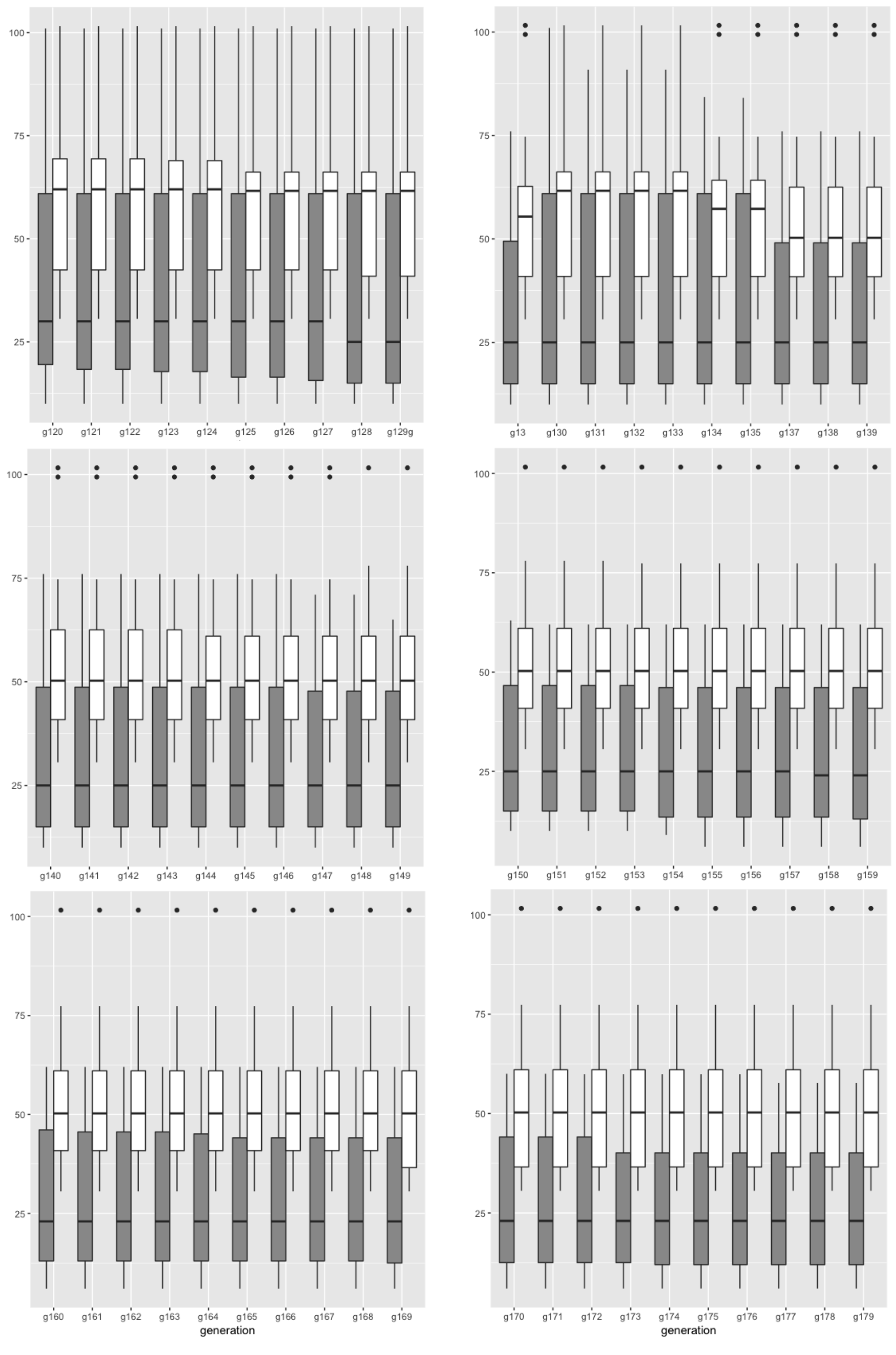

Figure 7: Boxplot for dispersion of 15 solutions of 200 generations(120-179 generation) from MIES (grey box) and SMAC (white box) for generalized sphere function

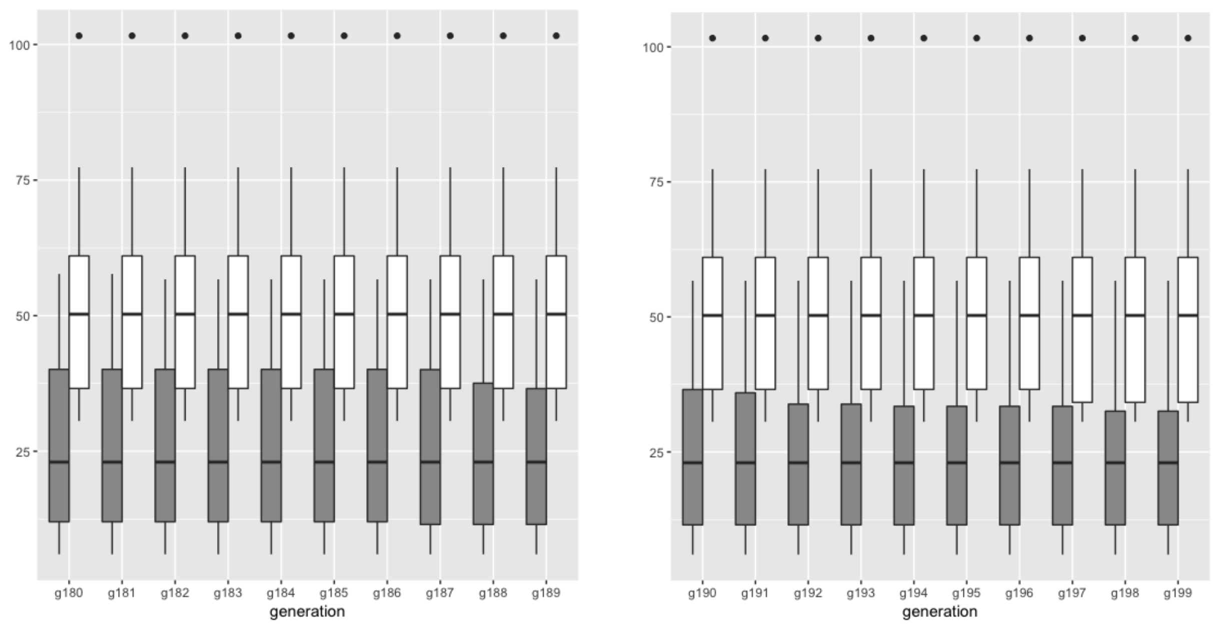

Figure 8: Boxplot for dispersion of 15 solutions of 200 generations(180-199 generation) from MIES (grey box) and SMAC (white box) for generalized sphere function

In the this objective function, the r, z and d shows the same types of variables like previous generalized sphere function, but each one has added an weight integer number i, and i is also the index for each sum function. The lower bound and upper bound for this objective function is $[-19, 19]$ as well. The feasible region for categorical variables is [-19,19]. The target is to reach the minimum solution of f_2. We implemented the above function for 15 times

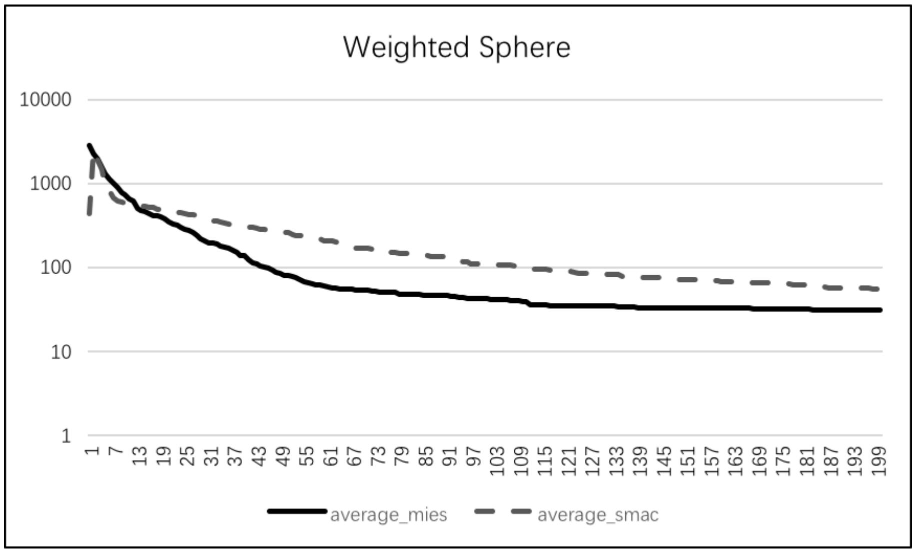

Figure 9: Average best fitnesses since the beginning recorded from MIES (real curves) and SMAC (dotted curves) on 6 test functions. 15 runs are conducted.

for each generation, according to the collected statistics, Figure 9 was drawn to illustrate the best solutions for entire 200 generations. Generally, MIES searched the better quality solution. Although the SMAC's initialized parents are better than MIES' and actually SMAC performs better in the period of former 13 evaluations. However, during first 50 iterations, MIES speeds up the searching process and exceeds SMAC at approximate 13th generation. For clear presentation, all the solution value has been logged. If people observe the procedure

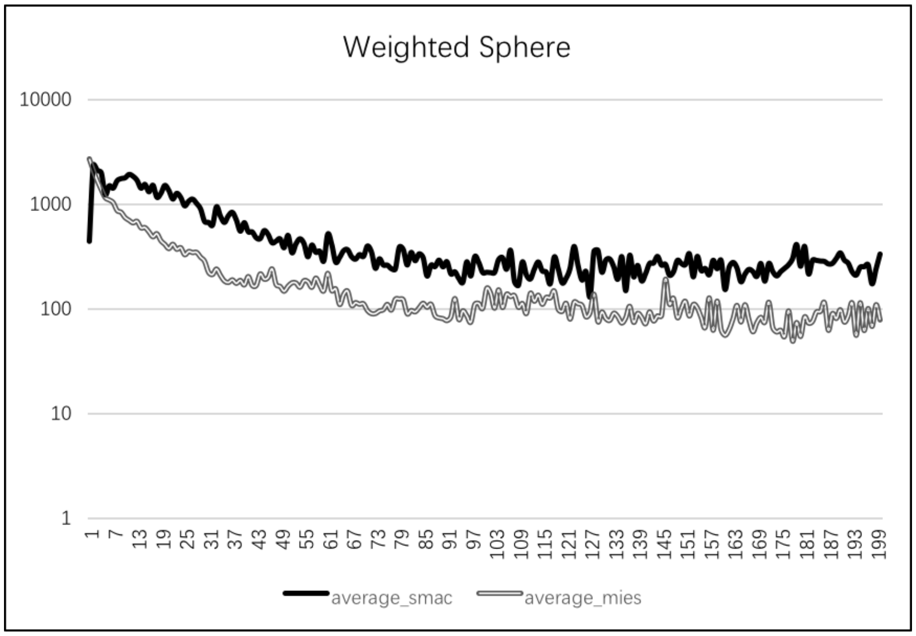

Figure 10: Average best fitness values per iteration, recorded from MIES (light gray curves) and SMAC (black curves) on 6 test functions. 15 runs are conducted.

carefully, they will study the contrast of each generation optimal which is demonstrated in Figure 10. In this graph, the iterative best fitness of MIES surpasses that of smack in almost every evaluation.

6.4 Modified Step Function

The step function has been chosen to show that MIES and SMAC are capable to tackle large plateaus in the fitness landscape. The plateau links in the search space, that lead to the same fitness value. Such plateaus occurs in practice for example when searching for feasible points, using penalty functions that are proportional to the number of violated constraints or simulation errors.

$$f_3(r, z, d) = \sum_{i=1}^{n_r} \lfloor r_i \rfloor^2 + \sum_{i=1}^{n_z} (z_i \text{ div } 10)^2 + \sum_{i=1}^{n_d} (d_i \text{ mod } 2)^2 \tag{11}$$

r, z, d are separately real variables, integer variables, and categorical variables, the purpose is to minimize the objective function and get optimal. In the first sum function, the floor of r was requested, then square it; in the second sum, div stands for division, z_i will be divided by 10 and the result is squared; the third sum is is remainder of the division by 2, and the result is squared again. i means the index, n_r, n_z and n_d describe the number of real values and amount of integer values and amount categorical values. To assure the equivalent number of generations, λ of mies has been set as 10, the evaluation quantity was set to 2000; the iteration of SMAC has been set as 200. The constraint of variables is $[-19, 19]$. In this test function, the general performance of MIES is better than that of SMAC, after 50th generation,

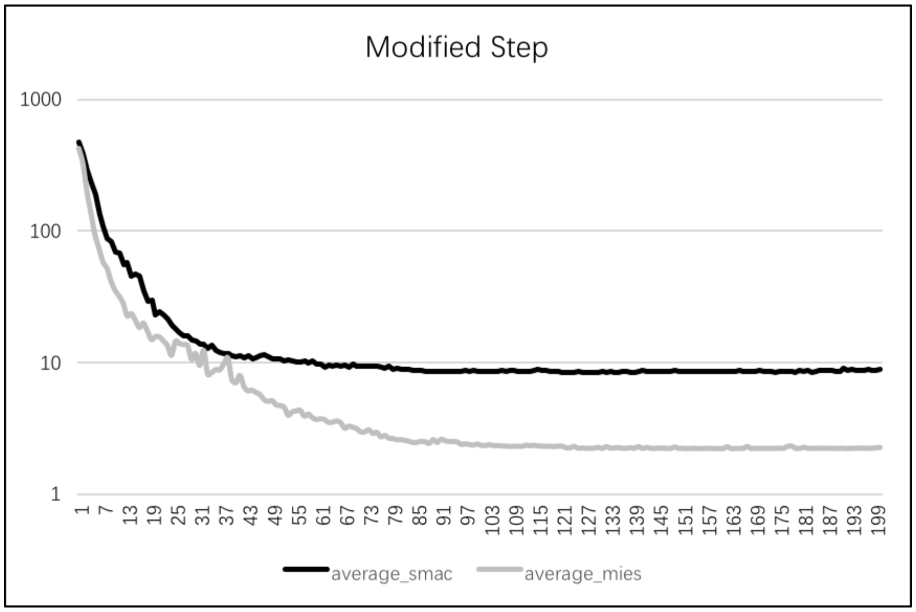

Figure 11: Average best fitnesses since the beginning recorded from MIES (light grey curves) and SMAC (black curves) on 6 test functions. 15 runs are conducted.

SMAC gets "lost" on the plateau while MIES keeps searching as it is a global optimization algorithm. See Figure 11 and Figure 12.

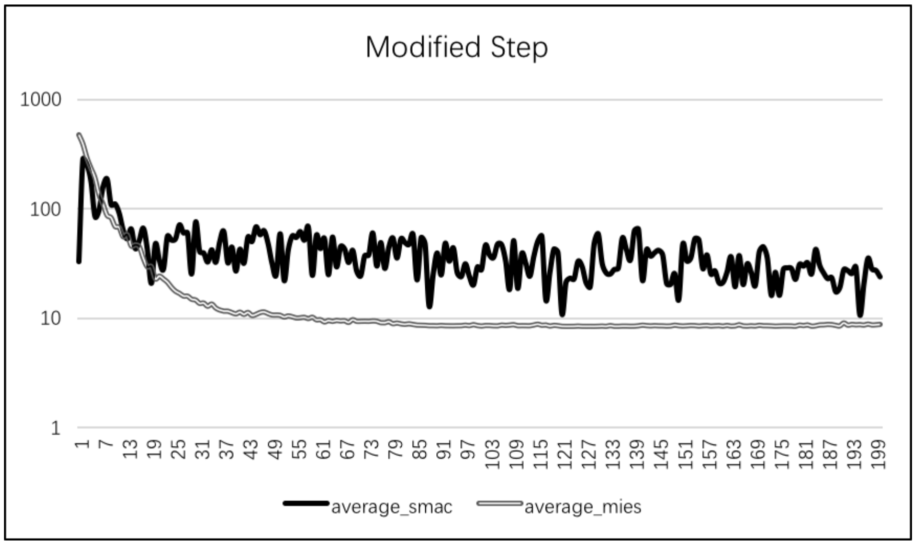

Figure 12: values recorded from MIES (light gray curves) and SMAC (black curves) on 6 test functions. 15 runs are conducted.

6.5 General Quadratic Function

The general quadratic function represents a strong iteration between all parameters. The contour lines of this function have approximately the shape of ellipsoids.

$$f_4(r, z, d) = \sum_{i=1}^{n} (\sum_{j=1}^{i} r_j + z_j + d_j)^2 \tag{12}$$

i and j are two index number, n is the amount of real values and integer values and categorical values: r, z and d. Also, in this case, the number of three types of variables are supposed to be equivalent. During the experiment, one phenomenon was noticed that SMAC will automatically abandon the extreme large solution on purpose of finding the minimum, so to assure the accuracy of the testing, the feasible region of the variables will be narrowed down and set as $[-10, 10]$. The total iteration is 200 and the times of implementation is 15. The number mentioned in the following graphs are all arithmetic value. From Figure 13, MIES begins better

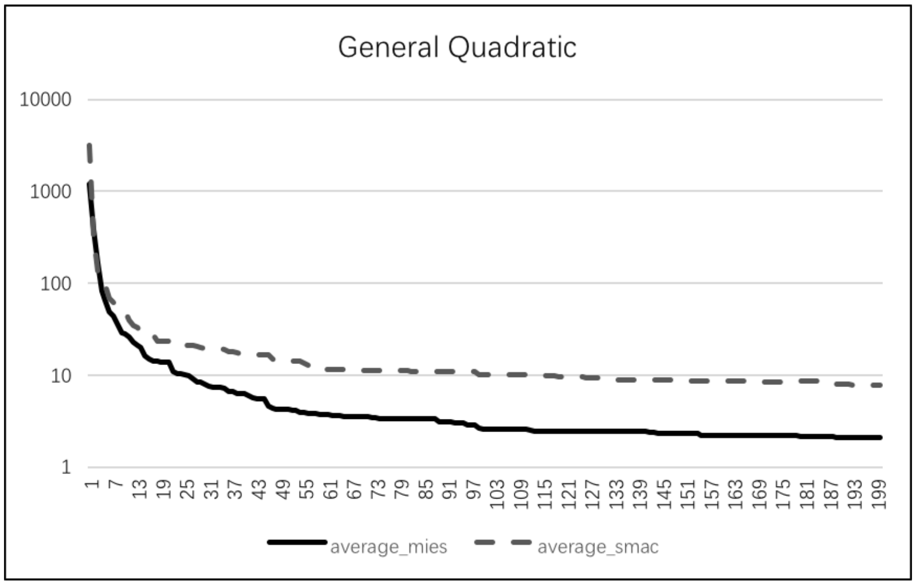

Figure 13: Average best fitnesses since the beginning recorded from MIES (real curves) and SMAC (dotted curves) on 6 test functions. 15 runs are conducted.

performance with the higher quality solution after the 7th generation. Over the entire 200 generations, MIES gives the better optimal solution than SMAC. Figure 14, it demonstrates the activeness of both optimization algorithms and the average value of MIES is always lower than SMAC.

7 Conclusion

SMAC shows an advantage when large plains are present in the fitness landscape, taking weighted step function test case as an example, when facing the real-world application where the violation-based penalty function is used, MIES is supposed to choose. Moreover, to tackle plenty of MINLPs with numerical, categorical and integer parameters, MIES shows its strength

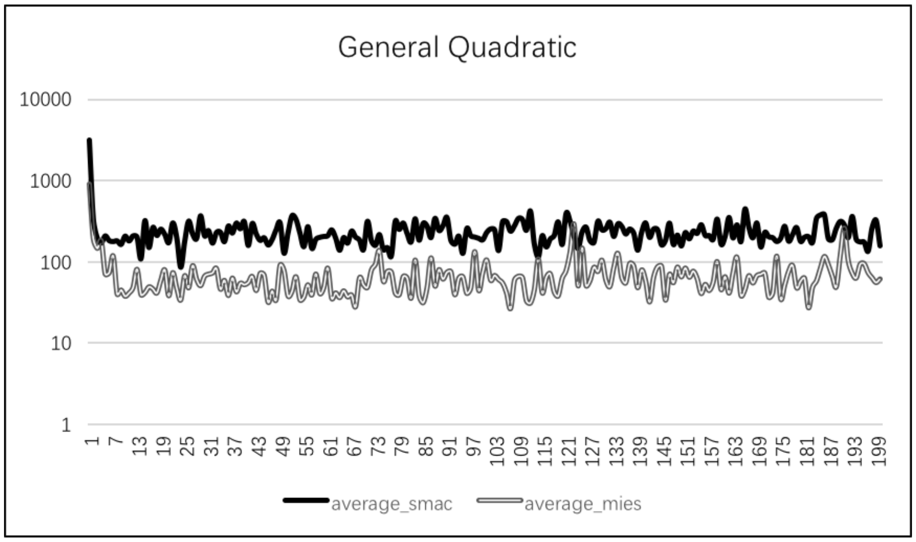

Figure 14: Average best fitness values per iteration, recorded from MIES (light gray curves) and SMAC (black curves) on 6 test functions. 15 runs are conducted.

Table 1: Average Running Times of Ten Different Benchmarking Functions for MIES and SMAC

Test Case	MIES Running Time/s	SMAC Running Time/s
One	8.62683	242.455
Two	8.49213	237.854
Three	8.00148	246.991
Four	8.40456	259.601
Five	7.95467	249.783
Six	9.13996	252.162
Generalized Sphere	8.99489	271.839
Weighted Sphere	8.31788	263.209
Modified Step	8.51589	270.725
General Quadratic	8.74782	264.195

in this field.Though in some cases,SMAC's solution is closed to MIES', but in consideration of the quality of the solution found, MIES would be a suitable algorithm. For most of decomposable and unimodal problems with iterations, MIES still optimizes them better. MIES did even better by comparing generalized sphere model and weighted sphere model, the weight i actually emphasizes the MIES's better performance. When some the problem is mentioned with approximate shape of ellipsoids, according to the benchmarking statistics, the best solution found by MIES is better.

Another results of this testing is the average running time of each algorithm for different functions. As indicated in the table, there is a huge gap between SMAC and mies in the running time, MIES' running time is only 1/30 of the SMAC's. In a real-world application, the running time should also be taken into account, which gives more preference to MIES.

This thesis only discussed the unimodal condition. In this type of situation, MIES performed better than SMAC, both in terms of time and the quality of the solution. However, we do not deal with non-deterministic problems where whether MIES still keeps its advantages requires further research. At the same time, there are many other methods dealing with mixed-integer problems like IRACE, but we did not verify this performance in the thesis, from a theoretical point of view, IRACE is possible to be more suitable to tackle non-deterministic problems. In fact, during the experiments, we did tried IRACE algorithm, but since it is not accessible to separate the evaluation process into 200 generations which costs such a long time, the IRACE method did not continue, but this issue is still worth consistently being studied.

8 Acknowledgment

The completion of this thesis cannot be separated from the concern and help of all the professors. I would like to thank Professor Thomas Bäck and Hao Wang for their guidance and help, their professional knowledge, serious scientific attitude and rigorous scholarship all have an impact on me. They have done a lot of work on this topic to help me overcome difficulties, thank them very much!

References

[1] H. Wang and T. Bäck. Evolutionary algorithms for industrial problems. `http://liacs.leidenuniv.nl/~csnaco/COST/`. Accessed Oct 24, 2017.

[2] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P.Adams, and Nando de Freitas. *Taking the Human Out of the Loop: A Review of Bayesian Optimization*, volume 104. IEEE, 2016.

[3] Frank Hutter, Holger H.Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, volume 6683, pages 507–523, 2011.

[4] Donald R.Jones, Matthias Schonlau, and William J.Welch. Efficient global optimization of expensive black-box function. *Journal of Global Optimization*, 13:455–492, 1998.

[5] Leo Breiman. Random forests. In *Machine Learning*, volume 45, pages 5–32, 2001.

[6] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[7] T. Bäck. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. *Oxford University Press on Demand*, 1996.

[8] Thomas Bäck and Martin Schütz. Evolution strategies for mixed–integer optimization of optical multilayer systems. In *Evolutionary Programming IV*, pages 33–51. The MIT Press, 1995.

[9] Li. R. *Mixed Integer Evolution Strategies for Parameter Optimization*. PhD thesis, Massachusetts Institute of Technology, 2013.

[10] Michael Emmerich, Monika Grötzner, and Martin Schütz. Mixed-integer evolution strategy for chemical plant optimization with simulators. In *Evolutionary Design and Manufacture*, pages 55–67, 2000.

[11] Michael Emmerich, Monika Grötzner, and Martin Schütz. Design of graph-based evolutionary algorithms: A case study for chemical process networks. *Evolutionary Computation*, 9, 2001.

[12] Rui Li, Michael Emmerich, Jeroen Eggermont, and Ernst G.P. Bovenkamp. Mixed-integer optimization of coronary vessel image analysis using evolution strategies. In *Conference: Genetic and Evolutionary Computation*, 2006.

[13] Rui Li, Michael T.M. Emmerich, Jeroen Eggermont, Ernst G.P. Bovenkamp, Thomas Bäck, Jouke Dijkstra, and Johan H.C. Reiber. Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis. In *IEEE World Congress on Computational Intelligence (IEEE World Congress on Computational Intelligence)*. IEEE, 2008.

[14] Hans-Paul Schwefel. *Evolution and optimum seeking*. Wiley, 1995. ISBN:978-0-471-57148-3.

[15] M. Schütz. Eine evolutionsstrategie für gemischt-ganzzahlige optimierungsprobleme mit variabler dimension. Diploma thesis, FB Informatik, University Dortmund, Germany, 1994.

[16] Günter Rudolph. An evolutionary algorithm for integer programming. In *International Conference on Parallel Problem Solving from Nature*, pages 139–148, 1994. PPSN 1994: Parallel Problem Solving from Nature — PPSN III.