

56. How to Create Generalizable Results

Thomas Bartz-Beielstein

Basically, this chapter tries to find answers for the following fundamental questions in experimental research.

- (Q-1) How can problem instances be generated?
- (Q-2) How can experimental results be generalized?

The chapter is structured as follows. Section 56.2 introduces real-world and artificial optimization problems. Algorithms are described in Sect. 56.3. Objective functions and statistical models are introduced in Sect. 56.4; these models take problem and algorithm features into consideration. Section 56.5 presents case studies that illustrate our methodology. The chapter closes with a summary and an outlook.

56.1	Test Problems in Computational Intelligence	1127
56.2	Features of Optimization Problems	1128
56.2.1	Problem Classes and Instances	1128
56.2.2	Feature Extraction and Instance Generation	1128
56.3	Algorithm Features	1130
56.3.1	Factors and Levels	1130
56.3.2	Example: Evolution Strategy	1130
56.4	Objective Functions	1131
56.5	Case Studies	1133
56.5.1	Single Problem Designs: SASP and MASP	1133
56.5.2	SAMP: Single Algorithm, Multiple Problems	1133
56.5.3	MAMP: Multiple Algorithms, Multiple Problems	1137
56.6	Summary and Outlook	1141
	References	1142

56.1 Test Problems in Computational Intelligence

Computational intelligence (CI) methods have gained importance in several real-world domains such as process optimization, system identification, data mining, or statistical quality control. Tools to determine the applicability of CI methods in these application domains in an objective manner are missing. Statistics provide methods for comparing algorithms on certain data sets. In the past, several test suites were presented and considered as state of the art. However, these test suites have several drawbacks, namely:

- Problem instances are mostly artificial and have no direct link to real-world settings.
- Since there is a fixed number of test instances, algorithms can be fitted or tuned to this specific and very limited set of test functions. As a consequence, studies (benchmarks) provide insight how these algorithms perform on this specific set of test

instances, but no insight on how they perform in general.

- Statistical tools for comparisons of several algorithms on several test problem instances are relatively complex and not easy to analyze.

We propose a methodology to overcome these difficulties. This methodology, which generates problem classes rather than uses one instance, is constructed as follows:

1. First, we pre-process the underlying real-world data.
2. In a second step, features from these data are extracted. This extraction relies on the assumption that mathematical variables can be used to represent real-world features. For example, decomposition techniques can be applied to model the underlying data structures, if we are using time-series data. The

original time series is deconstructed into a number of component series, where each of these reflects a certain type of behavior, e.g., a trend or seasonality [56.1]. We obtain an analytical model of the data.

3. Then, we parameterize this model. Based on this parametrization and randomization, we can generate infinitely many new problem instances.
4. If no real-world data are available, problem instances can be generated using test-problem generators. The generation of test problems, which are well-founded and have practical relevance, has been an on-going field of research for several decades.
5. From this infinite set, we can draw a limited number of problem instances which will be used for the comparison.
6. Since problem instances are selected randomly, we apply random and mixed models for the analysis [56.2]. Mixed models include fixed and random effects. A fixed effect is an unknown constant. Its estimation from the data is a common practice in analysis of variance (ANOVA) or regression. A random effect is a random variable. We estimate the parameters that describe its distribution, because – in contrast to fixed effects –

it makes no sense to estimate the random effect itself.

This chapter combines ideas from two approaches: problem generation and statistical analysis of computer experiments. The work presented by *Chiarandini* and *Goegebeur* [56.3] provides the basis of our statistical analysis. They present a systematic and well-developed framework for mixed models. Related modeling approaches were suggested by *McGeoch* [56.4] and *Birattari* [56.5]. *Gallagher* and *Yuan* [56.6] present a problem instance (landscape) generator that is parameterized by a small number of parameters, and the values of these parameters have a direct and intuitive interpretation in terms of the geometric features of the landscapes that they produce. *Castiñeiras* et al. [56.7] present a parameterizable benchmark generator for bin packing instances based on the well-known Weibull distribution. Using the shape and scale parameters of the Weibull distribution, the authors generate benchmarks that contain a variety of item size distributions. They report that for all bin capacities, the number of bins required in an optimal solution increases as the Weibull shape parameter increases. Using this feature, scalability is enabled.

56.2 Features of Optimization Problems

56.2.1 Problem Classes and Instances

Nowadays, it is common practice in optimization to choose a *fixed* set of problem instances in advance and to apply classical ANOVA or regression analysis. In many experimental studies a few problem instances π_i ($i = 1, 2, \dots, q$) are used and the results of some runs of the algorithms α_j ($j = 1, 2, \dots, h$) on these instances are collected. The instances can be treated as *blocks* and all algorithms are run on each single instance. Results are grouped per instance π_i . Analyses of these experiments shed some light on the performance of the algorithms on those specific instances. However, the interest of the researcher should not be just the performance of the algorithms on those specific instances chosen, but rather on the generalization of the results to the entire class Π . Generalizations about the algorithm's performance on new problem instances are difficult or impossible in this setting.

Based on ideas from *Chiarandini* and *Goegebeur* [56.3], to overcome this difficulty, we propose the following approach: a small set of problem in-

stances $\{\pi_i \in \Pi | i = 1, 2, \dots, q\}$ is chosen at random from a large set, or class Π , of possible instances of the problem. Problem instances are considered as factor levels. However, this factor is of a different nature from the fixed algorithmic factors in the classical ANOVA setting. Indeed, the levels are chosen at random and the interest is not in these specific levels but in the problem class Π from which they are sampled. Therefore, the levels and the factor are *random*. Consequently, our results are not based on a limited, fixed number of problem instances. They are randomly drawn from an infinite set, which enables generalization.

56.2.2 Feature Extraction and Instance Generation

A problem class Π can be generated in different manners. We will consider *artificial* and *natural* problem class generators. Artificially generated problems allow feature generation based on some predefined characteristics. They are basically theory driven, i.e., the researcher defines certain features such as linearity or

multi modality. Based on these features, a model (formula) is constructed. By integrating parameters into this formula, many problem instances can be generated by parameter variation. We will exemplify this approach in the following paragraph. The second way, which will generate natural problem classes, uses a three-stage approach. First, the real-world system and its components are described. Then, features are extracted from a real-world system. Based on this feature set, a model is defined. Adding parameters to this model, new problem instances can be generated. There is also a third way to *generate* test instances: if we are lucky, many data are available. In this case, we can sample a limited number of problem instances from the larger set of real-world data. The statistical analysis is similar for these three cases.

Artificial Test Functions

Several problem instance generators have been proposed over the last years. For example, *Gallagher* and *Yuan* present a landscape test generator, which can be used to set up problem instances for continuous, bound-constrained optimization problems [56.6]. The *Max-set of Gaussian landscape generator* (MSG) uses the maximum of m weighted Gaussian functions

$$G(x) = \max_{i \in \{1, 2, \dots, m\}} (w_i g_i(x)),$$

where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ denotes an n -dimensional Gaussian function

$$g(x) = \left(\frac{\exp\left(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T\right)}{(2\pi)^{n/2}|\Sigma|^{1/2}} \right)^{1/n},$$

μ is an n -dimensional vector of means, and Σ is an $(n \times n)$ covariance matrix. The mean of each Gaussian corresponds to an optimum on the landscape and the location of all optima is known. The global optimum is the one with the largest value. We will use the MSG problem instance generator in Sect. 56.5 to demonstrate our approach.

Natural Problem Classes

This section exemplifies the three fundamental steps for generating real-world problem instances, namely:

1. Describing the real-world system and its data
2. Feature extraction and model construction
3. Instance generation.

We will illustrate this procedure by using the classic Box and Jenkins airline data [56.8]. These data contain

the monthly totals of international airline passengers from 1949 to 1961. The feature extraction is based on methods from time-series analysis. Because of its simplicity the Holt–Winters method is popular in many application domains. It is able to adapt to changes in trends and seasonal patterns. The Holt–Winters prediction function requires the estimation of three parameters, i. e., α , β and γ , which can be estimated from original time-series data. Their optimal values are determined by minimizing the squared one-step prediction error. To generate new problem instances, these parameters can be slightly modified. Based on these modified values, the model is re-fitted. Finally, we can extract the new time series. One typical result from this instance generation is shown in Fig. 56.1. *Bartz-Beielstein* [56.9] describes this procedure in detail.

To illustrate the wide applicability of this approach, we will list further real-work problem domains, which are subject of our current research:

- *Smart metering*: The development of accurate forecasting methods for electrical energy consumption profiles is an important task. We consider time series collected from a manufacturing process. Each time series contains quarter-hourly samples of the energy consumption of a bakery. A detailed data description can be found in [56.10].
- *Water industry*: Canary is a software developed by the *United States Environmental Protection Agency* (US EPA) and Sandia National Laboratories. Its purpose is to detect events in the context of wa-

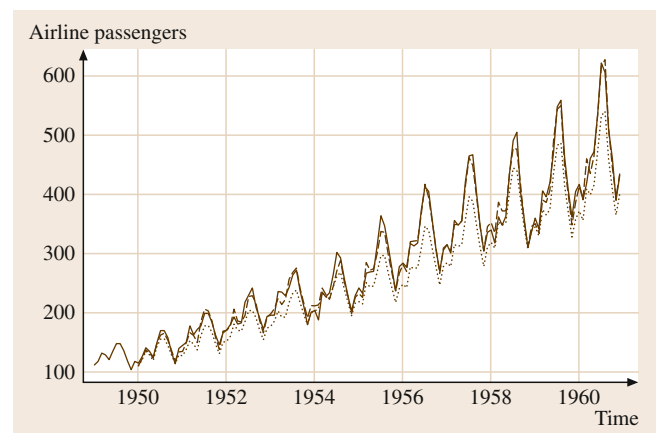


Fig. 56.1 Holt–Winters problem instance generator. The *solid line* represents the real data, the *dotted line* predictions from the Holt–Winters model and the *fine dotted line* modified predictions, respectively

ter contamination. An event is in this context defined as a certain time period where a contaminant significantly deteriorates the water quality. Distinguishing events from (i) background changes, (ii) maintenance and modification due to operation, and (iii) outliers is an essential task, which was implemented in the Canary software. Therefore, deviations are compared to regular patterns and short term changes. The corresponding data contains multi-variate time-series data. It is a selection from a larger dataset shipped with the open source event-detection software CANARY developed by US EPA and Sandia National Laboratories [56.11].

- **Finance:** The data are real-world data from intraday *foreign exchange* (FX) trading. The FX market is a financial market for trading currencies to enable international trade and investment. It is the largest and most liquid financial market in the world. Currencies can be traded via a wide variety of different financial instruments, ranging from simple spot trades over to highly complex derivatives.

56.3 Algorithm Features

56.3.1 Factors and Levels

Evolutionary algorithms (EA) belong to the large class of bio-inspired search heuristics. They combine specific components, which may be *qualitative*, like the recombination operator or *quantitative*, like the population size. Our interest is in understanding the contribution of these components. In statistical terms, these components are called *factors*. The interest is in the effects of the specific *levels* chosen for these factors. Hence, we say that the levels and, consequently, the factors are *fixed*. Although modern search techniques like sequential parameter optimization or Pareto genetic programming [56.13] allow multi-objective performance measures (solution quality versus variability or description length), we restrict ourselves to analyzing the effect of these factors on a univariate measure of performance. We will use the quality of the solutions returned by the algorithm at termination as the performance measure.

56.3.2 Example: Evolution Strategy

Evolution strategies (ES) are prominent representatives of evolutionary algorithms, which includes genetic algorithms and genetic programming as well [56.15].

We use three foreign exchange (currency rate) time series collected from Bloomberg. Each time series contains hourly samples of the change in currency exchange rate [56.12].

One typical goal in forecasting is the minimization of the forecast errors or the differences between real (observed) values, say y_i , and predicted values, say \hat{y}_i . This goal can be considered as an optimization problem.

As stated in Sect. 56.2.2, the statistical analysis is similar for artificial and natural problem classes. Our goal can be stated as follows: For a given problem class Π , which can be artificial or natural, we try to determine if an optimization algorithm α or several algorithm instances α_i show similar behavior on randomly selected problem instances $\pi_i \in \Pi$. This question will be formulated as a statistical hypothesis. Based on the related statistical framework, we can determine confidence intervals for the performance of the algorithm on unseen problem instances.

They can be classified as generic population-based metaheuristic optimization algorithms for global optimization that in some sense mimics the natural evolution. Evolution strategies are applied to hard real-valued optimization problems. Mutation is performed by adding a normally distributed random value to each vector component. The standard deviation of these random values is modified by self-adaptation. Evolution

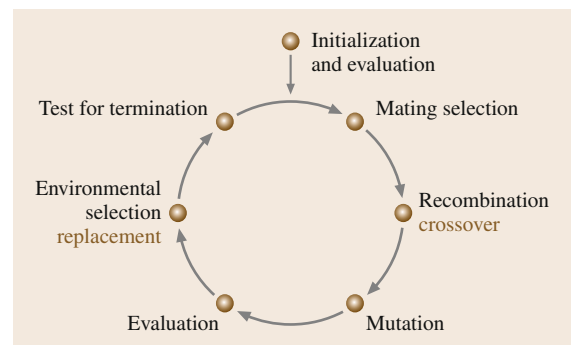


Fig. 56.2 The evolutionary cycle, basic working scheme of all ES and EA. Terms common for describing evolution strategies are used, alternative terms are added below in *brown*

Table 56.1 Settings of exogenous parameters of an ES. Recombination operators are labeled as follows: 1 = no, 2 = dominant, 3 = intermediate, 4 = intermediate as in [56.14]. Mutation uses the following encoding: 1 = no mutation, 2 = self adaptive mutation

Parameter	Symbol	Name	Range	Value
mue	μ	Number of parent individuals	\mathbb{N}	5
nu	$\nu = \lambda/\mu$	Offspring-parent ratio	\mathbb{R}_+	2.0
sigmaInit	$\sigma_i^{(0)}$	Initial standard deviations	\mathbb{R}_+	1.0
nSigma	n_σ	Number of standard deviations. d denotes the problem dimension	$\{1, d\}$	1
	c_τ	Multiplier for individual and global mutation parameters	\mathbb{R}_+	1.0
tau0			\mathbb{R}_+	0.0
tau			\mathbb{R}_+	1.0
rho	ρ	Mixing number	$\{1, \mu\}$	2
sel	κ	Maximum age	\mathbb{R}_+	1.0
mutation		Mutation	$\{1, 2\}$	2
sreco	r_σ	Recombination operator for strategy variables	$\{1, 2, 3, 4\}$	3
oreco	r_x	Recombination operator for object variables	$\{1, 2, 3, 4\}$	2

strategies can use a population of several solutions. Each solution is considered as individual and consists of object and strategy variables. Object variables represent the position in the search space, whereas strategy variables store the step sizes, i. e., the standard deviations for the mutation. We analyze the ES basic variant, which was proposed in [56.14].

Mutation means neighborhood-based movement in search space, which includes the exploration of the *outer space* currently not covered by a population, whereas recombination rearranges existing information and so focuses on the *inner space*. Selection is meant to introduce a bias towards better fitness values. A concrete ES may contain specific mutation, recombination, or selection operators, or call them only with a certain probability, but the control flow is usually left

unchanged. Each of the consecutive cycles is termed a *generation*. The control flow is shown in Fig. 56.2.

Concerning the representation, it should be noted that most empiric studies are based on canonical forms as binary strings or real-valued vectors, whereas many real-world applications require specialized, problem-dependent ones. Table 56.1 summarizes important ES parameters. This chapter presents two case studies. The first case study is based on a fixed ES parameter setting, whereas the second case study modifies the recombination operator for object variables. We are convinced that the applicability of the methods presented in this chapter goes far beyond the simplified case studies. Our main contribution is a framework, which allows conclusions that are not limited to a small number of problem instances but to problem classes.

56.4 Objective Functions

We will use the following optimization framework: an ES is applied as a minimizer on the test function $f(x)$. Formally speaking, let S denote some set, e.g., $S \subseteq \mathbb{R}^n$. We are seeking for values f^* and x^* , such that $\min_{x \in S} f(x)$ with $f^* = \min_{x \in S} f(x)$ and $x^* = \arg \min f(x)$. This approach can be extended in many ways. For example, if S denotes times-series data, then an optimization algorithm can be applied to minimize the empirical mean squared prediction error.

Test problem instances will be drawn from Gallagher's and Yuan's MSG test function generator. The following parameters can be used to specify the MSG generator:

- The number of Gaussian components m .
- The mean vector μ of each component.
- The covariance matrix Σ of each component.
- The weight of each component w_i .
- A maximum threshold $t \in [0; 1]$ can be specified for local optima and the fitness value of the global optimum G^* . Local optima are randomly generated within $[0; t \times G^*]$.

The following tuple can be used to specify an MSG generator

$$\Pi := ([-c, c]^n, n, m, D_\mu, \{D_\Sigma\}, \{t, G^*\}) \quad (56.1)$$

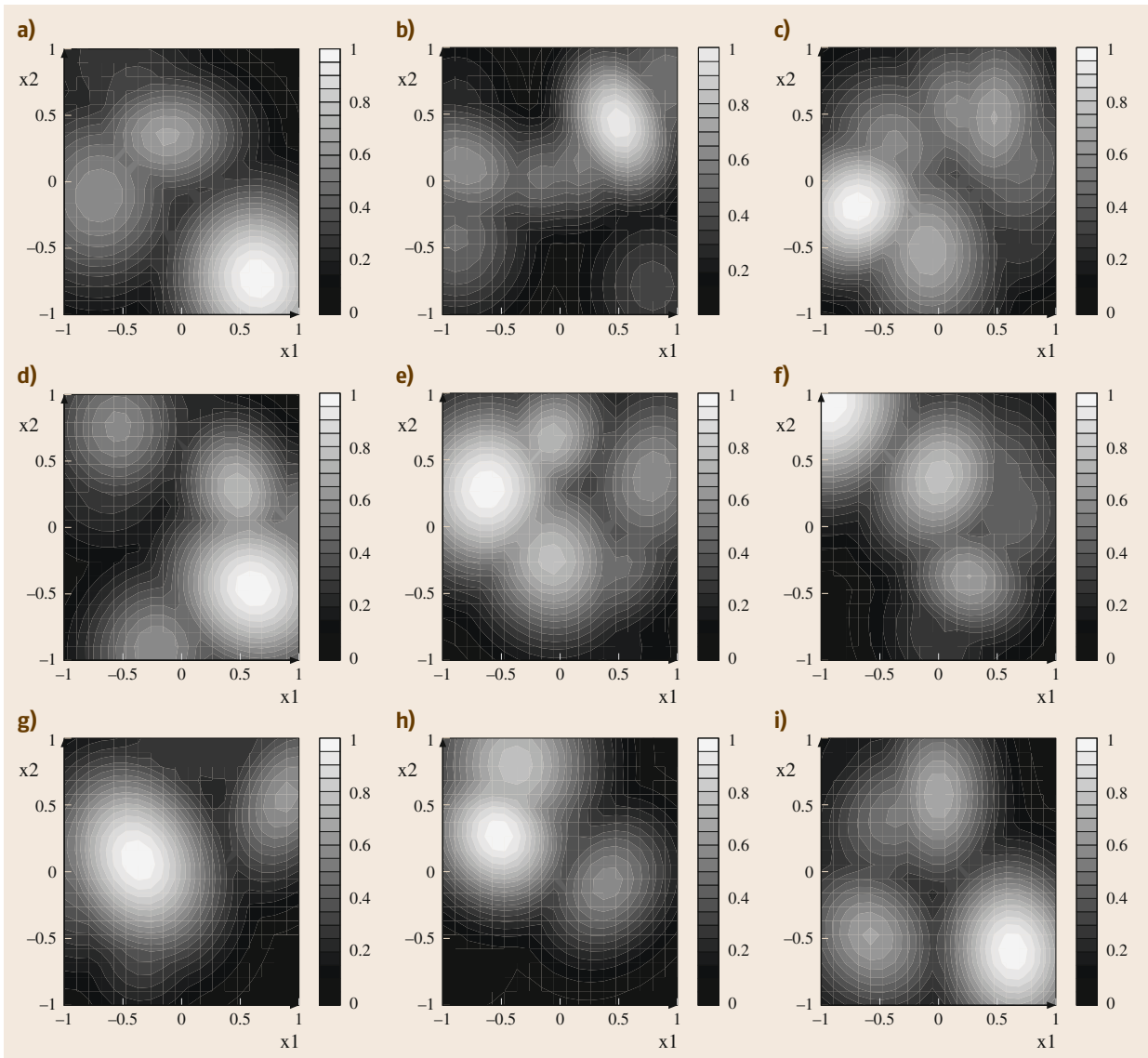


Fig. 56.3a-i Nine test problem instances from Π_{MSG} , generated with the MSG landscape generator as specified in (56.2). These figures exemplify how numbers and locations of the randomly generated optima can vary. Usually, the optima are evenly distributed in the search space. In some settings, there are a few dominating optima as can be seen in part (g)

where $c \in \mathbb{R}$ defines the boundary constraints of the search space, n the search space dimensionality, m the number of Gaussian components, D_μ the distribution used to generate the mean vectors of components, D_Σ the distribution or procedures used to generate covariances of components, $t \in [0; 1]$ the threshold for local optima, and G^* the function value of the global optimum.

Based on (56.1), we have specified the following MSG landscape generator for our experiments

$$\Pi_{\text{MSG}} := ([-1; 1]^2, 2, 10, \mathcal{U}[-1; 1], \{\mathcal{U}[0.05; 0.15], \mathcal{U}[-\pi/4, \pi/4]\}, \{0.8, 1\}). \quad (56.2)$$

With this setting, the mean vector of each component is generated randomly within $[-1, 1]^2$. The covariance

matrix of each component is generated with the procedure D_{Σ} in three steps:

1. A diagonal matrix \mathbf{S} with eigenvalues is generated.
2. An orthogonal matrix \mathbf{T} is generated through $n(n-1)/2$ rotations with random angles between $[-\pi/4, \pi/4]$.
3. The covariance matrix is generated as $\mathbf{T}^{\top}\mathbf{S}\mathbf{T}$.

The weight w_i of the component corresponding to the global optimum is set to 1 while other weights are randomly generated within $[0; 0.8]$. The nine problem instances, $\pi_i \in \Pi_{\text{MSG}}$, ($i = 1, \dots, 9$)

56.5 Case Studies

Bartz-Beielstein [56.9] introduced the acronyms:

- **SASP**: one single algorithm and one single problem instance
- **SAMP**: one single algorithm and multiple problems instances
- **MASP**: multiple algorithms and one single problem instance
- **MAMS**: multiple algorithms and multiple problem instances

for classifying optimization designs [56.17].

56.5.1 Single Problem Designs: SASP and MASP

In SASP we analyze the performance of an *optimization algorithm* α on a single problem instance π . An optimization problem has a set of input data which instantiates the problem. This might be a function in continuous optimization or the location and distances between cities in a traveling salesman problem. In the following, we will use Y to denote the random performance measure obtained by r runs of algorithm α on problem instance π . Because many optimization algorithms such as evolutionary algorithms are randomized, their performance Y on one instance is a random variable. It might be described by a probability density/mass function $p(y|\pi)$. Running the algorithm with different random seeds on one problem instance, we collect sample data y_1, \dots, y_r , which are *independent and identically distributed* (i.i.d.).

There are situations, in which SASP is the method of first choice. Real-world problems, which have to be solved only once in a very limited time, are good ex-

amples for using SASP optimizations. MASP shares several characteristics with SASP. Because of their limited capacities for generalization, SASP and MASP will not be investigated further in this study.

Note that we are using the distance to the optimum as an objective function in our experiments. Our objective function reads $G^* - f(x)$, because we are considering minimization problems. Other measures of interest might be the *gap percent of optimality*

$$\frac{(G^* - f(x))}{G^*} \times 100,$$

or computation time, etc., see, e.g., [56.16].

amples for using SASP optimizations. MASP shares several characteristics with SASP. Because of their limited capacities for generalization, SASP and MASP will not be investigated further in this study.

56.5.2 SAMP: Single Algorithm, Multiple Problems

Fixed-Effects Models

This setup is commonly used for testing an algorithm on a given (fixed) set of problem instances. Standard assumptions from *analysis of variance* (ANOVA) lead us to propose the following *fixed-effects model* [56.2]

$$Y_{ij} = \mu + \tau_i + \varepsilon_{ij}, \quad (56.3)$$

where μ is an overall mean, τ_i is a parameter unique to the i -th treatment (problem instance factor), and ε_{ij} is a random error term for replication j on problem instance i . Usually, the model errors ε_{ij} are assumed to be normally and independently distributed with mean zero and variance σ^2 . If problem instance factors are considered fixed, i. e., non random, the stochastic behavior of the response variable originates from the algorithm. This implies the experimental results

$$Y_{ij} \approx N(\mu + \tau_i, \sigma^2), \quad i = 1, \dots, q, j = 1, \dots, r, \quad (56.4)$$

and that the Y_{ij} are mutually independent. Results from statistical analyses remain valid only on the specific instances. Furthermore, SAMP with a fixed set of problem instances is subject to criticism, e.g., that algorithms are trained for this specific set up test instances (over fitting).

In order to make the results of the analysis independent of the specific instances and dependent instead

on the class of instances from which the specific instances are drawn, *Chiarandini* and *Goegebeur* propose randomized and mixed models for the experimental analysis of optimization algorithms as an extension of (56.3) [56.3]. In contrast to model (56.3), these models allow generalizations of results to the whole class of instances.

Randomized Models

In the following, we consider a population or *class* of instances Π . The class Π consists of a large, possibly an infinite, number of problem instances $\pi_i, i = 1, 2, 3, \dots$. Let $p(\pi)$ denote the probability of sampling instance π . The performance Y of the algorithm α on the class Π is described by the probability function

$$p(y) = \sum_{\pi \in \Pi} p(y|\pi)p(\pi). \quad (56.5)$$

If we run an algorithm α r times on instance π , then we receive r replicates of α 's performance, denoted by Y_1, \dots, Y_r . These r observations are i.i.d., i.e.,

$$p(y_1, \dots, y_r|\pi) = \prod_{j=1}^r p(y_j|\pi). \quad (56.6)$$

So far, we have considered r replicates of the performance measure Y on *one* problem instance π . Now we consider *several*, randomly sampled problem instances. Over all the instances the joint probability distribution of the observed performance measures is obtained by marginalizing over all instances

$$p(y_1, \dots, y_r) = \sum_{\pi \in \Pi} p(y_1, \dots, y_r|\pi)p(\pi). \quad (56.7)$$

Extending the model (56.7) to the case where one algorithm with several parameter settings or several algorithms are analyzed leads to mixed models, which will be discussed in Sect. 56.5.3.

Example SAMP: ES on Π_1 (Random-Effects Design)

The simplest random-effects experiment is performed as follows. For $i = 1, \dots, q$ a problem instance π_i is drawn randomly from the class of problem instances Π . On each of the sampled π_i , the algorithm α is run r times using different seeds for α . Due to α 's stochastic nature, we obtain, *conditionally on the sampled instance*, r replications of the performance measure that are i.i.d.

Let Y_{ij} ($i = 1, \dots, q; j = 1, \dots, r$) denote the random performance measure obtained in the j -th replication of α on π_i . We are interested in drawing conclusions about α 's performance on a larger set of problem instances from Π and not just on those q problem instances included in the experiment. A systematic approach to accomplish this task comprehends the following steps:

- SAMP-1 algorithm and problem instances
- SAMP-2 ANOVA and restricted maximum likelihood estimator (REML) model building
- SAMP-3 validation of the model assumptions
- SAMP-4 hypothesis testing
- SAMP-5 Confidence intervals and prediction.

SAMP-1 Algorithm and Problem Instances. The goal of this case study is to analyze if one algorithm shows a similar performance on a class of problem instances, say Π_{MSG} . A random-effects design will be used to model the results. We illustrate the decomposition of the variance of the response values in (i) the variance due to problem instance and (ii) the variance due to the algorithm and derive results, which are based on hypotheses testing as introduced in (56.12).

We consider one algorithm, an ES, which is run $r = 10$ times on a set of randomly generated problem instances. The ES is parameterized with the default setting from Table 56.1. These parameters are kept constant during the experiment. Nine instances are drawn from the set of problem instances Π_{MSG} . Problem instances were generated with the MSG landscape generator as specified in (56.2). The corresponding problem instances are shown in Fig. 56.3.

The null hypothesis reads *There is no instance effect*. Since we are considering the SAMP case, our experiment is based on one ES instance only. There are 90 observations, because 10 repeats were performed on 9 problem instances. Figure 56.4 shows the performance of the ES on these nine instances. The variable fSeed is used to denote the problem instance number π_i .

SAMP-2 ANOVA and REML Model Building.

ANOVA Model Building. The following analysis is based on the linear statistical model

$$Y_{ij} = \mu + \tau_i + \varepsilon_{ij} \begin{cases} i = 1, \dots, q, \\ j = 1, \dots, r, \end{cases} \quad (56.8)$$

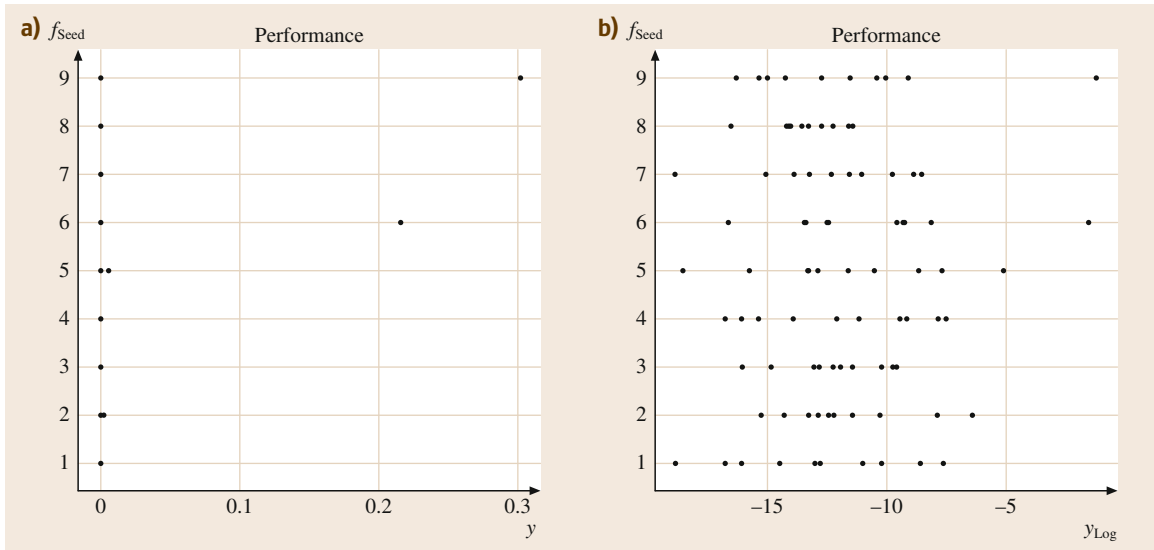


Fig. 56.4a,b Performance of the ES on nine test problem instances. (a) Problem instances plotted versus algorithm performance. (b) Problem instances plotted against logarithmic performance. Smaller values are better

where μ is an overall mean and ε_{ij} is a random error term for replication j on instance i . Note that in contrast to the fixed-effects model from (56.3), τ_i is a random variable representing the effect of instance i . The stochastic behavior of the response variable originates from both the instance and the algorithm. This is reflected in (56.8), where both τ_i and ε_{ij} are random variables. The model (56.8) is the so-called *random-effects model*, cf. [56.2] or [56.3].

We assume that τ_1, \dots, τ_q are i.i.d. $\mathcal{N}(0, \sigma_\tau^2)$ and $\varepsilon_{ij}, i = 1, \dots, q, j = 1, \dots, r$, are i.i.d. $\mathcal{N}(0, \sigma^2)$. If τ_i is independent of ε_{ij} and has variance $V(\tau_i) = \sigma_\tau^2$, the variance of any observation is $V(Y_{ij}) = \sigma^2 + \sigma_\tau^2$. Similar to the partition in classical ANOVA, the variability in the observations can be partitioned into a component that measures the variation between treatments and a component that measures the variation within treatments. Based on the fundamental ANOVA identity $SS_{total} = SS_{treat} + SS_{err}$, we define

$$MS_{treat} = \frac{SS_{treat}}{q-1} = \frac{r \sum_{i=1}^q (\bar{Y}_i - \bar{Y}_{..})^2}{q-1},$$

and

$$MS_{err} = \frac{SS_{err}}{q(r-1)} = \frac{\sum_{i=1}^q \sum_{j=1}^r (Y_{ij} - \bar{Y}_i)^2}{q(r-1)}.$$

It can be shown that

$$E(MS_{treat}) = \sigma^2 + r\sigma_\tau^2 \quad \text{and} \quad E(MS_{err}) = \sigma^2, \tag{56.9}$$

cf. [56.2]. Therefore, the estimators of the variance components are

$$\hat{\sigma}^2 = MS_{err}, \tag{56.10}$$

$$\hat{\sigma}_\tau^2 = \frac{MS_{treat} - MS_{err}}{r}. \tag{56.11}$$

The corresponding ANOVA table is shown in Table 56.2. Based on ANOVA calculations, with (56.10) we obtain an estimator of the first variance component $\hat{\sigma}^2 = -0.4848257$, and from (56.11), we obtain

Table 56.2 ANOVA table for a one-factor fixed and random effects models

Source of variation	Sum of squares	Degrees of freedom	Mean square	EMS fixed	EMS random
Treatment	SS_{treat}	$q-1$	MS_{treat}	$\sigma^2 + r \frac{\sum_{i=1}^q \tau_i^2}{q-1}$	$\sigma^2 + r\sigma_\tau^2$
Error	SS_{err}	$q(r-1)$	MS_{err}	σ^2	σ^2
Total	SS_{total}	$qr-1$			

the second component $\hat{\sigma}_\tau^2 = 11.32854$. The model variance can be determined as $\hat{\sigma}^2 + \hat{\sigma}_\tau^2 = 10.84372$. The mean $\mu = -12.05554$ from (56.8) can be extracted. Finally, the p value in the ANOVA table is calculated as 0.7979083.

Note that we have obtained a negative variance. Since negative variances are not feasible, we can proceed by setting their values to zero and proceed with these modified values. A more elegant way is presented in the following.

Restricted Maximum Likelihood. In some cases, the standard ANOVA, which was used in our example, produces a negative estimate of a variance component. This can be seen in (56.11): if $MS_{\text{err}} > MS_{\text{treat}}$, negative values occur. By definition, variance components are positive. Methods that always yield positive variance components have been developed. Here, we will use *restricted maximum likelihood estimators* (REML). The ANOVA method of variance component estimation, which is a method of moments procedure, and REML estimation may lead to different results. Output from an R-based analysis with the function `lme` from the package `lme4` reads as follows (`fSeed` denotes the problem instance) [56.18]:

```
Linear mixed model fit by REML
Formula: yLog ~ 1 + (1 | fSeed)
Data: samp.df
   AIC   BIC logLik deviance REMLdev
475.6 483.1 -234.8   469.3   469.6
Random effects:
Groups   Name              Variance Std.Dev.
fSeed    (Intercept)    0.000    0.0000
```

```
Residual          10.893   3.3004
Number of obs: 90, groups: fSeed, 9

Fixed effects:
              Estimate Std. Error t value
(Intercept) -12.0555    0.3479  -34.65
```

Compared to the ANOVA setting, different values for $\hat{\sigma}^2$, $\hat{\sigma}_\tau^2$, and μ were obtained. However, the REML-based analysis also shows that the variability in the response observations can be attributed to the variability of the algorithm.

SAMP-3 Validation of the Model Assumptions. Before performing hypothesis testing based on the models introduced in SAMP-2, the validity of the model assumptions has to be investigated. If the model is adequate, the residuals should exhibit no structure. Residuals are plotted against fitted values to check the assumption of homoscedasticity and quantile–quantile (Q–Q) plots are used to check if residuals meet the normality assumption. Quantile–quantile plots of the residuals are shown in Fig. 56.5 for the raw and the log-transformed responses. These plots provide a good way to compare the distribution of a sample with a distribution. Large deviations from the line indicate non-normality of the sample data. These Q–Q plots indicate that a log transformation of the response might be useful in our setting.

SAMP-4 Hypothesis Testing. Testing hypotheses about individual treatments (instances) is useless be-

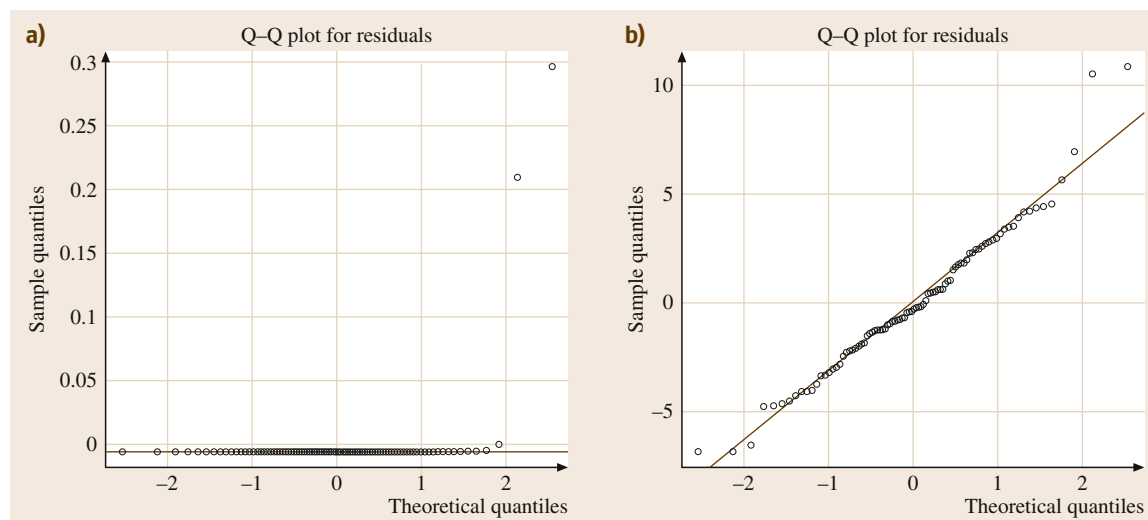


Fig. 56.5a,b (a) Q–Q plot of the residuals for raw data. (b) Q–Q plot for the log-transformed responses

cause the problem instances π_i are here considered as samples from some larger population of instances Π . We test hypotheses about the variance component σ_τ^2 , i. e., the null hypothesis

$$H_0 : \sigma_\tau^2 = 0 \text{ versus } H_1 : \sigma_\tau^2 > 0. \quad (56.12)$$

Under H_0 , the algorithm performance is identical on all problem instances (*all treatments are identical*), i. e., $r\sigma_\tau^2$ is very small. Based on (56.9), we conclude that $E(\text{MS}_{\text{treat}}) = \sigma^2 + r\sigma_\tau^2$ and $E(\text{MS}_{\text{err}}) = \sigma^2$ are similar. Under the alternative, variability exists between treatments. Standard analysis shows that $\text{SS}_{\text{err}}/\sigma^2$ is distributed as chi-square with $q(r-1)$ degrees of freedom. Let $F_{u,v}$ denote the F distribution with u numerator and v denominator degrees of freedom. Under H_0 , the ratio

$$F_0 = \frac{\text{SS}_{\text{treat}}/q-1}{\text{SS}_{\text{err}}/q(r-1)} = \frac{\text{MS}_{\text{treat}}}{\text{MS}_{\text{err}}}$$

is distributed as $F_{q-1, q(r-1)}$. To test hypotheses in (56.8), we require that τ_1, \dots, τ_q are i.i.d. $\mathcal{N}(0, \sigma_\tau^2)$, ε_{ij} , $i = 1, \dots, q$, $j = 1, \dots, r$, are i.i.d. $\mathcal{N}(0, \sigma^2)$, and all τ_i and ε_{ij} are independent of each other. These considerations lead to the decision rule to reject H_0 at the significance level α if

$$f_0 > F(1 - \alpha; q - 1, q(r - 1)), \quad (56.13)$$

where f_0 is the realization of F_0 from the data observed. An intuitive motivation for the form of statistic F_0 can be obtained from the expected mean squares. Under H_0 both MS_{treat} and MS_{err} estimate σ^2 in an unbiased way, and F_0 can be expected to be close to one. On the other hand, large values of F_0 give evidence against H_0 .

Regarding the SAMP case, we obtain the following values: Based on (56.9) and (56.13), we can determine the F statistic and the p value. We get $\text{MS}_{\text{treat}} = \text{MS}_{\text{err}} = 10.89275$ and $f_0 = 1$, which results in a large p value: 0.4426363. The null hypothesis $H_0 : \sigma_\tau^2 = 0$ from (56.12) cannot be rejected, i. e., we conclude that there is no instance effect. A similar conclusion was obtained from the ANOVA method of variance component estimation as introduced in Table 56.2.

SAMP-5 Confidence Intervals and Prediction. An unbiased estimator of the overall mean μ is

$$\hat{\mu} = \bar{y}_{..} = \sum_{i=1}^q \sum_{j=1}^r \frac{y_{ij}}{(qr)}.$$

Its variance is given by

$$V(\bar{y}_{..}) = V\left(\sum_{i=1}^q \sum_{j=1}^r \frac{y_{ij}}{(qr)}\right) = \frac{r\sigma_\tau^2 + \sigma^2}{qr}.$$

With (56.9) and (56.10), we obtain an estimator of the variance of the overall mean μ as

$$\hat{V}(\bar{y}_{..}) = \frac{\text{MS}_{\text{treat}}}{qr}.$$

Since

$$\frac{\bar{Y}_{..} - \mu}{\sqrt{\frac{\text{MS}_{\text{treat}}}{qr}}} \approx t_{q(r-1)},$$

the confidence limits for μ can be derived as

$$\bar{y}_{..} \pm t_{1-\alpha/2; q(r-1)} \sqrt{\frac{\text{MS}_{\text{treat}}}{qr}}. \quad (56.14)$$

We conclude the SAMP case study with prediction of the algorithm's performance on a new instance from the same class. Based on (56.14), we obtain the following 95% confidence interval: $[2.6773 \times 10^{-6}; 1.262 \times 10^{-5}]$. Again, confidence intervals from the REML and ANOVA methods are very similar. Summarizing, we can conclude that the ES performs similarly on instances from Π_{MSG} , which were generated with (56.2).

56.5.3 MAMP: Multiple Algorithms, Multiple Problems

In the MAMP case study, fixed effects are included in the conditional structure of (56.6), which leads to a mixed model. Instead of one fixed algorithm as in the SAMP case, we consider either several algorithms or algorithms with several parameters. Both situations can be treated while considering algorithms as levels of a fixed factor, whereas problem instances are drawn randomly from the population of instances Π_{MSG} :

- MAMP-1 algorithm and problem instances
- MAMP-2 ANOVA and REML model building
- MAMP-3 validation of the model assumptions
- MAMP-4 hypothesis testing:
 1. Random effects
 2. Fixed effects
- MAMP-5 confidence intervals and prediction.

MAMP-1 Algorithm and Problem Instances

We aim at comparing the performance of the ES with different recombination operators over an instance class. More precisely, we have four ES instances using recombination operators $\{1, 2, 3, 4\}$ and nine instances randomly sampled from the class Π_{MSG} as illustrated in Fig. 56.3. Each run is repeated ten times. In this study $4 \times 9 \times 10 = 360$ data were used. We are interested in the following questions:

- Is there an instance effect?
- Do the mean performances of the ES with different recombination operators differ?
- Do the instance–algorithm interactions contribute to the variability of the response?

A first visual inspection, which plots the performance of the algorithm within each problem instance, is shown in Fig. 56.6. In eight of the nine instances the linear regression line has a negative slope and the intercepts do not differ very much. This indicates that there is no significant interaction between the fixed and the random factors.

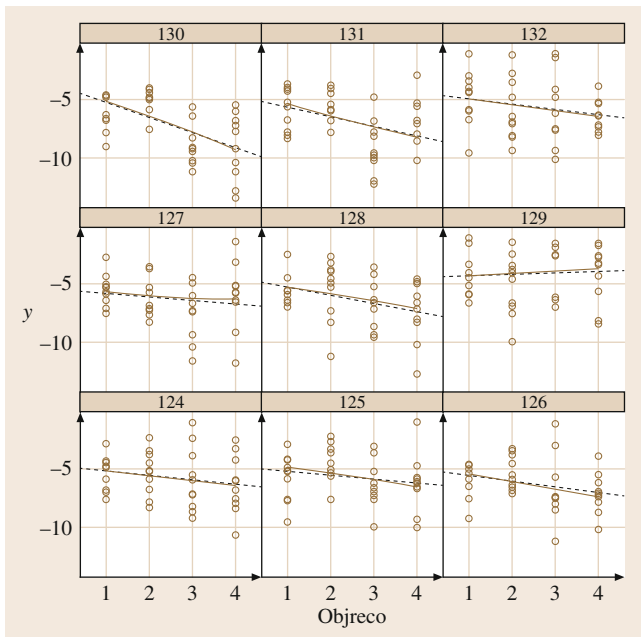


Fig. 56.6 Four algorithms (ES with modified recombination operators) on nine test problem instances. Each *panel* represents one problem instance and problem instances are labeled from 124 to 130. Performance is plotted against the level of the recombination operator

MAMP-2 ANOVA and REML Model Building

The variability in the performance measure can be decomposed according to the following *mixed-effects ANOVA model*

$$Y_{ijk} = \mu + \alpha_j + \tau_i + \gamma_{ij} + \varepsilon_{ijk} , \quad (56.15)$$

where μ is an overall performance level common to all observations, α_j is a fixed effect due to the algorithm j , τ_i is a random effect associated with instance i , γ_{ij} is a random interaction between instance i and algorithm j , and ε_{ijk} is a random error for replication k of algorithm j on instance i . We assume that the α_j 's are fixed effects such that $\sum_{j=1}^h \alpha_j = 0$ and that the random elements τ_i are i.i.d. $\mathcal{N}(0, \sigma_\tau^2)$, γ_{ij} are i.i.d. $\mathcal{N}(0, \sigma_\gamma^2)$, ε_{ijk} are i.i.d. $\mathcal{N}(0, \sigma^2)$ and τ_i , γ_{ij} and ε_{ijk} are mutually independent random variables. Similarly to (56.6) the conditional distribution of the performance measure given the instance and the instance–algorithm interaction is given by

$$Y_{ijk} | \tau_i, \gamma_{ij} \approx \mathcal{N}(\mu + \alpha_j + \tau_i + \gamma_{ij}, \sigma^2) , \quad (56.16)$$

with $i = 1, \dots, q$, $j = 1, \dots, h$, and $k = 1, \dots, r$. The marginal model reads (after integrating out the random effects τ_i and γ_{ij}):

$$Y_{ijk} \approx \mathcal{N}(\mu + \alpha_j, \sigma^2 + \sigma_\tau^2 + \sigma_\gamma^2) . \quad (56.17)$$

Based on these statistical assumptions, hypothesis tests can be performed about fixed and random factor effects. Using the mixed model (56.16), we are interested in testing whether there is a difference between the factor level means $\mu + \alpha_j$ ($j = 1, \dots, h$). The hypotheses for testing the fixed effects can be formulated as

$$H_0 : \alpha_i = 0 \forall i \quad \text{against} \quad H_1 : \exists \alpha_j \neq 0 . \quad (56.18)$$

Regarding random effects, tests about particular levels are useless. This is similar to the random-effects model (56.8). Again, we perform tests on the variance components σ_τ^2 and σ_γ^2 instead. These can be formulated as follows

$$\begin{aligned} H_0 : \sigma_\tau^2 = 0 , \quad \text{and} \quad H_0 : \sigma_\gamma^2 = 0 , \\ H_1 : \sigma_\tau^2 > 0 , \quad \text{and} \quad H_1 : \sigma_\gamma^2 > 0 , \end{aligned} \quad (56.19)$$

respectively. If all treatment (problem instances) combinations have the same number of observations, i. e., if the design is balanced, the test statistics for these hypotheses are ratios of mean squares that are chosen such

Table 56.3 Expected mean squares and consequent appropriate test statistics for a mixed two-factor model with h fixed factors, q random factors, and r repeats (after [56.3])

Effects	Mean squares	Df	Expected mean squares	Test statistics
Fixed factor	MSA	$h - 1$	$\sigma^2 + r\sigma_y^2 + rq \frac{\sum_{j=1}^h \alpha_j^2}{h-1}$	MSA/MSAB
Random factor	MSB	$q - 1$	$\sigma^2 + r\sigma_y^2 + rh\sigma_\tau^2$	MSB/MSAB
Interaction	MSAB	$(h - 1)(q - 1)$	$\sigma^2 + r\sigma_y^2$	MSAB/MSE
Error	MSE	$hq(r - 1)$	σ^2	

Table 56.4 ANOVA for the MAMP case

Mean squares	Factors	Df	Sum Sq	Mean Sq	F value	Pr(>F)
MSA	objreco	3	154.59	51.53	11.05	0.0000
MSB	fSeed	8	251.79	31.47	6.75	0.0000
MSAB	objreco:fSeed	24	185.60	7.73	1.66	0.0288
MSE	Residuals	324	1511.27	4.66		

that the expected mean squares of the numerator differs from the expected mean squares of the denominator only by the variance components of the random factor under test. *Chiarandini* and *Goegebeur* [56.3] present the resulting analysis of variance, which is shown in Table 56.3.

ANOVA Model Building

The ANOVA table for the experiments from the MAMP case study is shown in Table 56.4. Equating the observed mean squares in the lines of the ANOVA table to their expected values and solving for the variance components leads to the following equations [56.2]

$$\hat{\sigma}_\tau^2 = \frac{MSB - MSAB}{hr} = 0.593502,$$

$$\hat{\sigma}_y^2 = \frac{MSAB - MSE}{r} = 0.306907,$$

$$\hat{\sigma}^2 = MSE = 4.664423.$$

Next, we will compare these results to the REML-based analysis of the mixed model.

REML Model Building

We have specified sum contrasts instead of the default treatment contrasts used in `lmer()`. Again, `fSeed` represents the problem instance, whereas the algorithm instance $\alpha_j, j = 1, \dots, 4$, is represented by `objreco`.

```
Linear mixed model fit by REML
Formula: yLog ~ objreco + (1 | fSeed)
          + (1 | fSeed:objreco)
```

```
Random effects:
Groups      Name      Variance Std.Dev.
fSeed:objreco (Intercept) 0.30691  0.55399
fSeed      (Intercept) 0.59351  0.77039
```

```
Residual                4.66442  2.15973
Number of obs: 360,
groups: fSeed:objreco, 36; fSeed, 9
```

```
Fixed effects:
      Estimate Std. Error t value
(Intercept)  -6.0222    0.2956 -20.370
objreco1      0.6176    0.2539   2.433
objreco2      0.6918    0.2539   2.725
objreco3     -0.6671    0.2539  -2.628
```

As can be seen from the `Random effects` section of the REML model output, the estimated variances for the problem instance and the instance-interaction random effects are $\hat{\sigma}_\tau^2 = 0.59351$ and $\hat{\sigma}_y^2 = 0.30691$, respectively. The `Random effects` section presents the estimates of the fixed effects model parameters, i. e., `objreco`.

MAMP-3 Validation of the Model Assumptions

Again, a check of the diagnostic plots (Fig. 56.7) reveals that a log transformation of the response improves the model adequacy.

MAMP-4a Hypothesis Testing: Random Effects

We will consider random effects first. Regarding problem instances, test about levels are meaningless. Hence, we perform tests about the variance components σ_τ^2 and σ_y^2 , which were presented in (56.19). First, we test the null hypothesis, which states that the components of the random effects are zero. Based on the ANOVA from Table 56.3, we obtain the values for the MAMP case that are shown in Table 56.4. The values reveal that there are main factor effects (fixed and random), but no significant interaction effects.

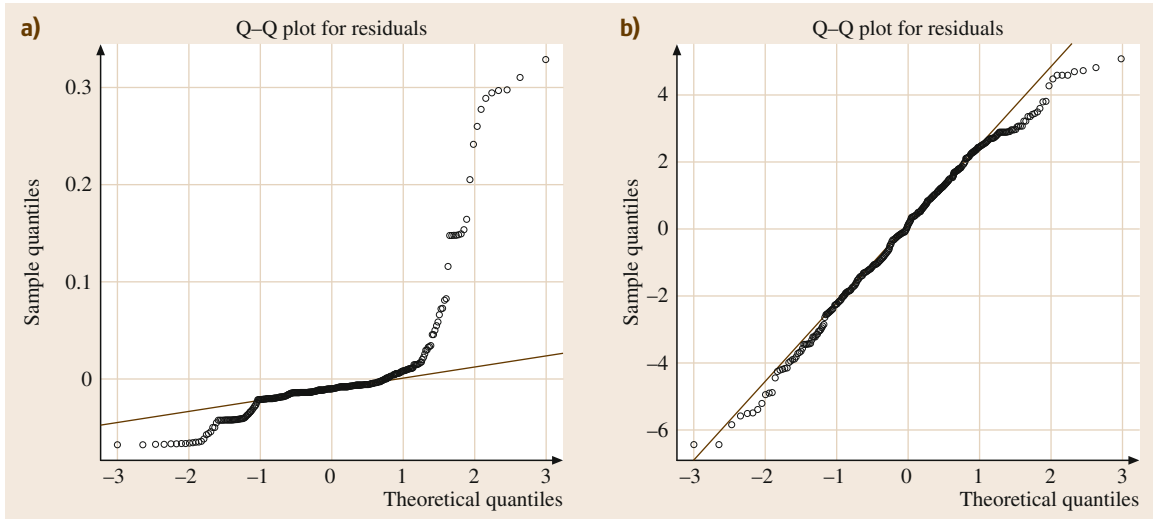


Fig. 56.7a,b (a) Q-Q plot of the residuals for raw data. (b) Q-Q plot for the log-transformed responses

Alternatively, we can compute the likelihood ratios of models with and without the factors under observation.

```
Data: mamp.df
Models:
mamp.lmer2: yLog ~ objreco + (1 | fSeed)
mamp.lmer3: yLog ~ objreco + (1 | fSeed)
              + (1 | fSeed:objreco)
              Df    AIC    BIC logLik
mamp.lmer2   6 1616.7 1640.0 -802.35
mamp.lmer3   7 1616.6 1643.8 -801.31
              Chisq Chi Df Pr(>Chisq)
              2.0929   1   0.148
```

These tests indicate that there are also no significant instance-algorithm interactions. Additional likelihood-ratio tests show that the fixed factor and random factor effects are significant.

MAMP-4b Hypothesis Testing: Fixed Factor Effects

Regarding fixed factors, we are interested in testing for differences in the factor level means $\mu + \alpha_i$. These tests were formulated in (56.18), i.e., we are testing H_0 : all α_i are equal to 0 versus H_1 : at least one $\alpha_j \neq 0$. Here, we use the test statistic from [56.2, p. 523] for testing that the means of the fixed factor effects are equal. The appropriate test statistic for testing that the means of the fixed factor effects are equal, i.e., H_0 is true, is

$$F_0 = \frac{MSA}{MSAB} = \frac{154.59/3}{185.6/24} = 6.663\ 362,$$

with values taken from Table 56.4. The reference distribution is $F_{n-1, (n-1)(q-1)}$. We calculate the p value for the test on the fixed-effect term. The p value obtained is 0.002, hence the results collected indicate that the factor recombination (`objreco`) has a statistically significant impact on the performance of the algorithm. Using sum of contrasts implies that $\sum \alpha_j = 0$. The point estimates for the mean algorithm performance with the j -th fixed factor setting can be obtained by $\mu_j = \mu + \alpha_j$. The fixed factor effects can be estimated

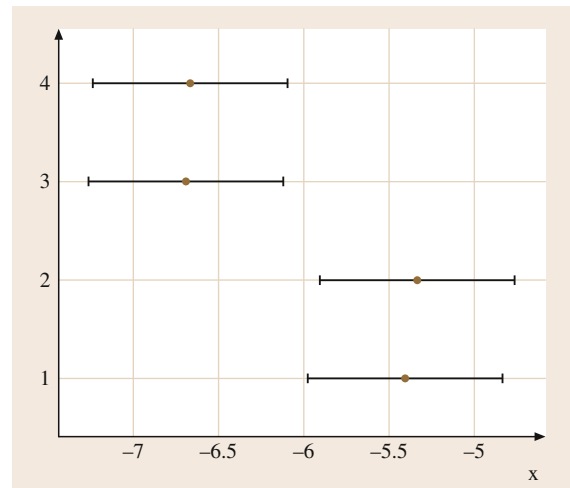


Fig. 56.8 Paired comparison plots. Results from four ES instances with different recombination operators are shown in this plot

in the mixed model as

$$\hat{\mu} = \bar{y}_{...}$$

$$\hat{\alpha}_j = \bar{y}_{j.} - \bar{y}_{...} ,$$

which results in the following estimates: $\hat{\alpha}_1 = 0.6175519$, $\hat{\alpha}_2 = 0.6918047$, $\hat{\alpha}_3 = -0.6671266$, and $\hat{\alpha}_4 = -0.6423659$.

The same estimates were obtained with the REML analysis as can be seen from the REML model output in Sect 56.5.3. The corresponding fixed effects are shown in the `Fixed effects` section of the REML output. For example, we obtain the following value: `objreco1 = $\hat{\alpha}_1 = 0.6176$` .

56.6 Summary and Outlook

In order to answer question (Q-1), we propose an approach to generate natural problem classes, which are based on real-world data. If no such data are available, artificial problem generators such as MSG can be used. Since our approach uses a model, say M , to generate new problem instances, one conceptual problem arises: this approach is not applicable, if the final goal is the determination of a model for the data, because M is per definition the best model in this case, and the search for good models will result in M . However, there is a simple solution to this problem. In this case, the feature extraction and model generation should be skipped and the original data should be modified by adding some noise or performing transformations on the data. Nevertheless, if applicable, the model-based approach is preferred, because it sheds some light on the underlying problem structure.

The model-based approach can be used to generate infinitely many test-problem instances. Instead of using a fixed number of problem instances, we propose:

1. Using randomly generated problem instances
2. Treating the problem instance as a random factor.

Algorithms with different parameterizations are tested on this set of randomly generated problem instances. This experimental setup requires modified statistics, so-called random-effects models or mixed models. This approach may lead to objective evaluations and comparisons. If normality assumptions are

MAMP-5 Confidence Intervals and Prediction

We generate paired comparisons plots, which are based on confidence intervals. The wrapper function `intervals()` from *Chiarandini and Goebel* [56.3] was used for visualizing these confidence intervals as shown in Fig. 56.8. When intervals overlap we conclude that there is no significant difference. Here, we can conclude that the recombination operators (1) and (2) show a similar performance, whereas performances between (3) and (2) are different.

Intermediate recombination of the object variables, i. e., (3) and (4), results in a significant improvement of the performance.

met, confidence intervals can be determined, which *forecast* the behavior of an algorithm on unseen problem instances. Furthermore, results can be generalized in real-world settings. This gives an answer to question (Q-2).

In order to demonstrate the applicability of our approach, the performance of an evolution strategy was analyzed. The first SAMP example illustrates that the selection of the problem instance from the problem class Π_{MSG} has no significant impact on the performance of the optimization algorithm. Furthermore, confidence intervals, which can be used to predict the performance of the algorithm on a problem class, were determined. The MAMP case exemplifies how to analyze the effect of different algorithm parameter settings on the performance. Four variants of the recombination operator and nine problem instances were used. The analysis reveals that the choice of the recombination operator has a significant effect on the algorithm's performance: the performance of the algorithm differs with different recombination operators. Intermediate recombination of the object variables results in a performance improvement. We demonstrated that the problem instances contribute significantly to the variability in the response and that there is no significant instance–algorithm interaction.

The software that was used in this study will be integrated into the R package SPOT (sequential parameter optimization toolbox) [56.19].

References

- 56.1 P.J. Brockwell, R.A. Davis: *Introduction to Time Series and Forecasting* (Springer, New York 2002)
- 56.2 D.C. Montgomery: *Design and Analysis of Experiments* (Wiley, New York 2001)
- 56.3 M. Chiarandini, Y. Goegebeur: Mixed models for the analysis of optimization algorithms. In: *Experimental Methods for the Analysis of Optimization Algorithms*, ed. by T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss (Springer, Berlin Heidelberg 2010) pp. 225–264, preliminary version available as Tech. Rep. DMF-2009-07-001 at the The Danish Mathematical Society
- 56.4 C.C. McGeoch: Toward an experimental method for algorithm simulation, *INFORMS J. Comput.* **8**(1), 1–15 (1996)
- 56.5 M. Birattari: *On the Estimation of the Expected Performance of a Metaheuristic on a Class of Instances* 2004), Tech. Rep. (IRIDIA, Bruxelles 2004)
- 56.6 M. Gallagher, B. Yuan: A general-purpose tunable landscape generator, *IEEE Trans. Evol. Comput.* **10**(5), 590–603 (2006)
- 56.7 I. Castiñeiras, M.D. Cauwer, B. O'Sullivan: Weibull-based benchmarks for bin packing, *Lect. Notes Comput. Sci.* **7514**, 207–222 (2012)
- 56.8 G.E.P. Box, G.M. Jenkins, G.C. Reinsel: *Time Series Analysis, Forecasting and Control* (Holden-Day, San Francisco 1976)
- 56.9 T. Bartz-Beielstein: *Beyond Particular Problem Instances: How to Create Meaningful and Generalizable Results*, Tech. Rep. TR 03/2012 (Cologne University of Applied Sciences, 2012)
- 56.10 T. Bartz-Beielstein, M. Friese, B. Naujoks, M. Zaefferer: SPOT applied to non-stochastic optimization problems – An experimental study, *Genet. Evol. Comput. Conf. (GECCO 2012)* (ACM, Philadelphia 2012)
- 56.11 M. Zaefferer: Optimization and Empirical Analysis of an Event Detection Software for Water Quality Monitoring, Master Thesis (University of Applied Sciences, Cologne 2012)
- 56.12 O. Flasch, T. Bartz-Beielstein, D. Bicker, W. Kantschik, C. von Strachwitz: *Results of the GECCO 2011 Industrial Challenge: Optimizing Foreign Exchange Trading Strategies*. CIOF Tech. Rep. 10/11, Res. Center CIOF (Cologne University of Applied Science, Cologne 2011)
- 56.13 E. Vladislavleva: Model-based Problem Solving through Symbolic Regression via Pareto Genetic Programming, Ph.D. Thesis (Tilburg University, Tilburg 2008)
- 56.14 H.-G. Beyer, H.-P. Schwefel: Evolution strategies – A comprehensive introduction, *Nat. Comput.* **1**, 3–52 (2002)
- 56.15 H.-P. Schwefel: *Numerical Optimization of Computer Models* (Wiley, Chichester 1981)
- 56.16 T. Bartz-Beielstein: *Experimental Research in Evolutionary Computation – The New Experimentalism*, Natural Computing Series (Springer, Berlin, Heidelberg, New York 2006)
- 56.17 T. Bartz-Beielstein, M. Preuss: Automatic and interactive tuning of algorithms, *Proc. 13th Annu. Conf. Companion Genet. Evol. Comput. (GECCO)* (ACM, New York 2011) pp. 1361–1380
- 56.18 J. Pinheiro, D. Bates: *Mixed-Effects Models in S and S-PLUS* (Springer, New York 2000)
- 56.19 T. Bartz-Beielstein, M. Zaefferer: A Gentle Introduction to Sequential Parameter Optimization, Tech. Rep. TR 01/2012 C1plus Bd. 1/2012 (Cologne University of Applied Sciences, Cologne 2012)