Chapter 7
Constraint Meta-Modeling

7.1 Introduction

In practical optimization problems, constraints play an important role. Constraints decrease the allowed solution space to a feasible subset of the original one. Similar to the fitness function, we assume that fitness and constraint functions are blackboxes, i.e., nothing is known about the functions but evaluations via function calls with candidate solutions $\mathbf{x}$. The constraint function is usually denoted as $g(\mathbf{x})$ in literature and also in this chapter.

We define the constrained optimization problem as the problem to minimize $f(\mathbf{x})$ subject to the inequality constraints $g_i(\mathbf{x})$ with $i = 1, \ldots, n_c$. Inequality constraints can use the operators $\geq$ or $\leq$. If we define the constraints as $g_i(\mathbf{x}) \leq 0$, a feasible solution achieves values smaller than or equal to zero. Then, the overall constraint violation is measured by summing up all single constraint violations $G(\mathbf{x}) = \sum_{i=1}^{n_c} \max(0, g_i(\mathbf{x}))$, and for constraints of the form $g_i(\mathbf{x}) \geq 0$, the corresponding definition is $G(\mathbf{x}) = \sum_{i=1}^{n_c} \max(0, -g_i(\mathbf{x}))$.

Hence, a positive G indicates a constraint violation. To treat the constraint violation as binary decision problem, we employ the signum function $\text{sgn}(\cdot)$[1] with

$$g(\mathbf{x}) = \text{sgn}(G(\mathbf{x})). \tag{7.1}$$

It holds $g(\mathbf{x}) = 1$, if any constraint g_i is violated and $g(\mathbf{x}) = 0$, if solution $\mathbf{x}$ is feasible. Discretizing the constraint violation is useful for handling the meta-model as binary classification problem. In this chapter, we learn a meta-model $\hat{g}$ of the constraint function with SVMs. We treat the constraint meta-model learning problem as two-class classification problem, i.e., $\hat{g}(\mathbf{x}) = 0$ for a feasible solution $\mathbf{x}$ and $\hat{g}(\mathbf{x}) = 1$ for an infeasible one.

The chapter is structured as follows. First, SVMs are introduced in Sect. 7.2. The meta-model and management mechanism is introduced in Sect. 7.3 for the (1+1)-ES.

[1] $\text{sgn}(x) = 1$ for $x > 0$, 0 for $x = 0$, and -1 for $x < 0$.

Related work is presented in Sect. 7.4. An experimental study is shown in Sect. 7.5. The chapter closes with conclusions in Sect. 7.6. The benchmark problems are introduced in the appendix.

7.2 Support Vector Machines

Since decades, SVMs belong to the state-of-the art classification algorithms [1]. They have found their way into numerous applications. The variant SVR can be used for regression problems [2]. The idea of SVMs is to learn a separating hyperplane between patterns of different classes. The separating hyperplane should maintain a maximal distance to the patterns of the training set. With the hyperplane, novel patterns $\mathbf{x}'$ can be classified. A hyperplane $\mathbf{H}$ can be described by normal vector $\mathbf{w} = (w_1, \dots, w_d)^T \in \mathbb{R}^d$ and point $\mathbf{x}_0$ on the hyperplane. For each point $\mathbf{x}$ on $\mathbf{H}$ it holds $\mathbf{w}^T(\mathbf{x} - \mathbf{x}_0) = 0$, as the weight vector is orthogonal to the hyperplane. While defining shift $w_0 = -\mathbf{w}^T\mathbf{x}_0$, the hyperplane definition becomes

$$\mathbf{H} = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^T\mathbf{x} + w_0 = 0\}. \tag{7.2}$$

The objective is to find the optimal hyperplane. This is done by maximizing $1/\|\mathbf{w}\|_2$ corresponding to minimizing $\|\mathbf{w}\|_2$, and the definition of the optimization problem becomes

$$\min \frac{1}{2}\|\mathbf{w}\|_2^2 \tag{7.3}$$

subject to the constraint

$$y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq +1, \text{ for all i = 1, \dots, N} \tag{7.4}$$

Figure 7.1 shows the decision boundary based on a maximizing the margin $\|\mathbf{w}\|_2$. The patterns on the border of the margin are called support vectors. They define the hyperplane that is used as decision boundary for the classification process. Finding $\|\mathbf{w}\|_2$ is an optimization problem, which is the SVM training phase.

The optimization problem is a convex one and can be solved with quadratic programming resulting in the following equation:

$$L_d = \frac{1}{2}(\mathbf{w}^T\mathbf{w}) - \mathbf{w}^T\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i - w_0\sum_{i=1}^{N}\alpha_i y_i + \sum_{i=1}^{N}\alpha_i \tag{7.5}$$

$$= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\alpha_i \tag{7.6}$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}\alpha_i \tag{7.7}$$

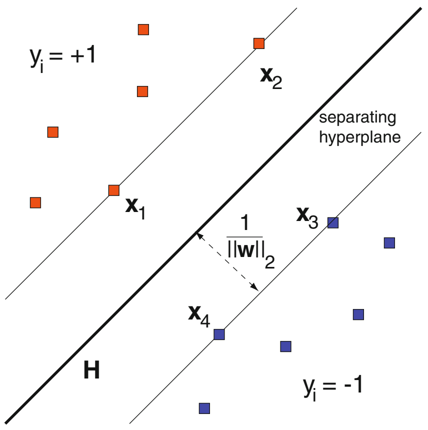

Fig. 7.1 Decision boundary of an SVM achieved by maximizing margin $1/\|\mathbf{w}\|_2$ and support vector lying on the border of the margin

This equation has to be maximized w.r.t. α_i subject to constraints $\sum_{i=1}^N \alpha_i y_i = 0$ and $\alpha_i \geq 0$ for $i = 1, \ldots, N$. Maximizing L_d, which stands for the dual optimization problem, can be solved with quadratic optimization methods. The dimensionality of the dual optimization problem depends on the number N of patterns, not on their dimensionality d. The upper bound for the runtime is $\mathcal{O}(N^3)$, while the upper bound for space is $\mathcal{O}(N^2)$.

We get rid of the constraint with a Lagrange formulation [3]. The result of the optimization process is a set of patterns that defines the hyperplane. These patterns are called support vectors and satisfy

$$y_i(\mathbf{w}^T\mathbf{x}_i + w_0) = 1, \tag{7.8}$$

while lying on the border of the margin, see Fig. 7.1. With any support vector $\mathbf{x}_i$, the SVM is defined as

$$\hat{g}(\mathbf{x}') = \text{sign}(\mathbf{w}^T\mathbf{x}_i + w_0) \tag{7.9}$$

with $w_0 = y_i - \mathbf{w}^T\mathbf{x}_i$. An SVM that is trained with the support vectors computes the same discriminant function as the SVM trained on the original training set.

For the case that patterns are not separable, slack variables $\xi_i \geq 0$ are introduced that store the deviation from the margin. The optimization problem is relaxed to

$$y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \tag{7.10}$$

while the slack variables $\xi_i \leq 0$. The number of misclassifications is $|\{\xi_i > 0\}|$. With the soft error $\sum_{i=1}^N \xi_i$, the soft margin optimization problem can be defined as

$$\frac{1}{2}\|\mathbf{w}\|_2^2 + C \cdot \sum_{i=1}^{N} \xi_i \tag{7.11}$$

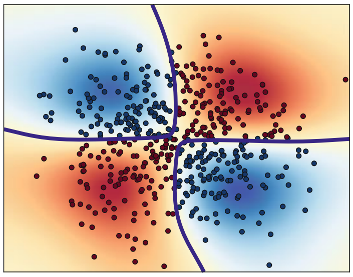

Fig. 7.2 Illustration of SVM learning on XOR data set consisting of $N = 1000$ patterns. The decision boundary shows that the RBF-kernel allows separating both classes

subject to constraints of Eq. 7.10. Penalty factor C serves as regularization parameter trading off complexity and data misfit, i.e., the number of non-separable patterns. For the soft margin optimization problem, the Lagrangian formulation is used with the primal and dual problem transformation.

For handling non-linear data, kernel functions are applied. For kernels, we replace the inner product computations, which are necessary for solve the SVM optimization problem by the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ that compares a similarity between instances in original input space. Instead of mapping $\mathbf{x}_i$ and $\mathbf{x}_j$ to the abstract feature space and computing the dot product there, the kernel function can directly be applied in the original space. An explicit mapping to the new space is not necessary, a concept known as kernel trick. Instead, we use kernel $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ to replace dot product in the new space. Besides a linear kernel, an RBF kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\gamma^2}\right) \tag{7.12}$$

with kernel bandwidth γ is often used. Parameter $\gamma > 0$ is usually tuned with grid search. The matrix of kernel values $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^N$ is called kernel or Gram matrix. In many mathematical models, the kernel matrix is a convenient mathematical expression. Figure 7.2 shows an example of an SVM with RBF-kernel learning the XOR data set, see Fig. 7.2.

SVMs are part of the SCIKIT-LEARN package. The following examples illustrate their use.

- `from sklearn import svm` imports the SCIKIT-LEARN SVM implementation.
- `clf = svm.SVC()` creates a support vector classification (SVC) implementation.
- `clf.fit(X,y)` trains the SVM with patterns `X` and corresponding labels `y`.

7.3 Algorithm

In this section, we introduce the constraint meta-model approach, which is based on the (1+1)-ES [4]. We denote the approach as CON-ES. The approach works as follows, see Algorithm 5. In the generational loop, a new offspring solution $\mathbf{x}'$ is generated with recombination and Gaussian mutation. The CON-ES works like the usual (1+1)-ES until the training set size N of patterns and constraint function calls is reached, i.e., a training set $\{(\mathbf{x}_i, g(\mathbf{x}_i))\}_{i=1}^N$ is available. In order not to overload the pseudocode, this condition is left out in Algorithm 5. The constraint meta-model $\hat{g}$ is trained with the training set employing cross-validation. As of now, each new solution $\mathbf{x}'$ is first examined on $\hat{g}$ concerning its constraint violation. If feasible, it is evaluated on the real constraint function g. New candidate solutions are generated until meta-model $\hat{g}$ and constraint function g indicate feasibility of $\mathbf{x}'$. The training set $\{(\mathbf{x}_i, g(\mathbf{x}_i))\}_{i=1}^N$ is updated in each step and consists of the last N pairs of solutions and their constraint evaluations corresponding to pattern-label pairs. At the end of the loop, step size σ is adapted according to Rechenberg's success rule. A certain balance of labels is required, i.e., $\{(\mathbf{x}_i, g(\mathbf{x}_i))\}_{i=1}^N$ is only an appropriate training set, if each class is represented by a similar ratio of labels. The practitioner has to take care for this balance. If large parts of the solution space are constrained, enough feasible solutions have to be generated to make the problem balanced.

Algorithm 5 CON-ES

```
1: initialize $\mathbf{x}$
2: repeat
3:   repeat
4:     $\mathbf{z} \sim \sigma \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:     $\mathbf{x}' = \mathbf{x} + \mathbf{z}$
6:     if $\hat{g}(\mathbf{x}') = 0$ then
7:       compute $g(\mathbf{x}')$
8:     end if
9:   until $g(\mathbf{x}') = 0$
10:  last $N$ solutions $\rightarrow \{(\mathbf{x}_i, g(\mathbf{x}_i))\}_{i=1}^N$
11:  train $\hat{g}$
12:  evaluate $\mathbf{x}' \rightarrow f(\mathbf{x}')$
13:  if $f(\mathbf{x}') \leq f(\mathbf{x})$ then
14:    replace $\mathbf{x}$ with $\mathbf{x}'$
15:  end if
16:  adapt $\sigma$ with Rechenberg
17: until termination condition
```

In our CON-ES variant, we use SVMs as constraint meta-models. An SVM is trained on training set $\{(\mathbf{x}_i, g(\mathbf{x}_i))\}_{i=1}^N$ with cross-validation and grid search for optimal parameters. A new training might not be necessary in each generation. In the experimental part, we will train the model every 20 generations.

To summarize, based on the constraint model evaluation (in case of infeasibility) or the real constraint function evaluation (in case of feasibility), the feasibility of a new solution $\mathbf{x}'$ is determined. Each time the meta-model $\hat{g}$ predicts infeasibility, a constraint function evaluation on g can be saved.

7.4 Related Work

Since decades of research, many constraint handling methods for evolutionary algorithms have been developed. Methods range from penalty functions that decrease the fitness of infeasible solutions [5] and decoder functions that let the search take place in another unconstrained or less constrained solution space [6] to feasibility preserving approaches that adapt representations or operators [7] to enforce feasibility. Multi-objective approaches treat each constraint as objective that has to be considered separately [8]. For this sake, evolutionary multi-objective optimization methods like non-dominated sorting (NSGA-ii) [9] can be adapted. Penalty functions are powerful methods to handle constraints. A convenient variant is death penalty that rejects infeasible solutions and generates new ones until a sufficient number of feasible candidates are available. A survey on constraint handing for ES gives [10].

Theoretical result on constraint handling and also constraint handling techniques for the CMA-ES [11] are surprisingly rare. For the (1+1)-CMA-ES variant, Arnold and Hansen [12] propose to approximate the directions of the local normal vectors of the constraint boundaries and to use these approximations to reduce variances of the Gaussian distribution in these directions.

We show premature convergence for the Tangent problem [13]. It is caused by dramatically decreasing success probabilities when approximating the constraint boundary. Arnold and Brauer [14] start the theoretical investigation of the (1+1)-ES on the Sphere function with one constraint with a Markov chain analysis deriving progress rates and success probabilities. Similarly, Chotard et al. [15] perform a Markov chain analysis of a $(1, \lambda)$-ES and demonstrate divergence for constant mutation rates and geometric divergence for step sizes controlled with path length control.

Unlike meta-modeling of fitness functions, results on meta-modeling of the constraint boundary are also comparatively rare in literature. Poloczek and Kramer [16] propose an active learning scheme that is based on a multistage model, but employ a linear model with binary search. A pre-selection scheme allows the reduction of constraint function calls. In [17] we combine an adaptive penalty function, which is related to Rechenberg's success rule, with a nearest neighbor fitness and constraint meta-model. Often, the optimal solution of a constrained problem lies in the neighborhood of the feasible solution space. To let the search take place in this region, the adaptive penalty function balances the penalty factors as follows. If less than $1/5$th of the population is feasible, the penalty factor γ is increased to move the population into the feasible region

$$\gamma = \gamma/\tau \tag{7.13}$$

with $0 < \tau < 1$. Otherwise, i.e., if more than 1/5th of the population of candidate solutions is feasible, the penalty is weakened

$$\gamma = \gamma \cdot \tau \tag{7.14}$$

to allow the search moving into the infeasible part of the solution space. The success rate of 1/5th allows the fastest progress towards the optimal solution.

In [18], Kramer et al. propose a linear meta-model that adapts the covariance matrix of the CMA-ES. The model is based on binary search between the feasible and the infeasible solution space to detect points on the constraint boundary and to define a linear separating hyper-plane. Gieseke and Kramer [19] propose a constraint meta-model scheme for the CMA-ES that employs active learning to select reasonable training points that improve the classifier.

7.5 Experimental Analysis

In this section, we experimentally analyze the CON-ES on two constrained benchmark problems based on the Sphere function. The first is the Sphere function $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$ with a linear constraint $g_1(\mathbf{x}) = \sum_{i=1}^{N} x_i \geq 0$ through the optimum $\mathbf{0}$. The optimum lies at the constraint boundary making approximately one half of the population infeasible for isotropic Gaussian mutation. The second constraint is the Tangent problem [12, 13], which is parallel to the previous function but shifted, i.e., $g_1(\mathbf{x}) = \sum_{i=1}^{N} x_i - N \geq 0$. The Tangent results in a constraint function that lies tangential to the contour lines of the fitness function and makes the optimization process a tedious task as the success rates decrease while approximating the optimum.

We employ the following algorithmic settings. The (1+1)-ES uses Gaussian mutation. The first solution is initialized with $\mathbf{x} = (N, \ldots, N)^T$ and the initial step size is $\sigma = 1.0$. Rechenberg's rule employs the setting $\tau = 0.5$ and examines the success rate every 10 generations. We train the SVM with the new training set every 20 iterations and use 5-fold cross-validation and grid search for C and γ with $C, \gamma \in \{10^{-10}, 10^{-9}, \ldots, 10^{10}\}$.

Table 7.1 shows the classification report for the constraint boundary meta-model after 300 generations on the Tangent problem with $d = 10$ dimensions on an arbitrary optimization run. The evaluation is based on one third of the training set. Precision and recall are high for the class of feasible patterns (class 0) and slightly lower for the class of infeasible patterns (class 1). In the following, we will show how the meta-model performs in the course of the constrained optimization process.

Table 7.1 Classification report for constraint meta-model on the tangent problem

Domain	Precision	Recall	F1	Support
0	0.98	0.97	0.97	98
1	0.73	0.80	0.76	10
Avg/total	0.96	0.95	0.95	108

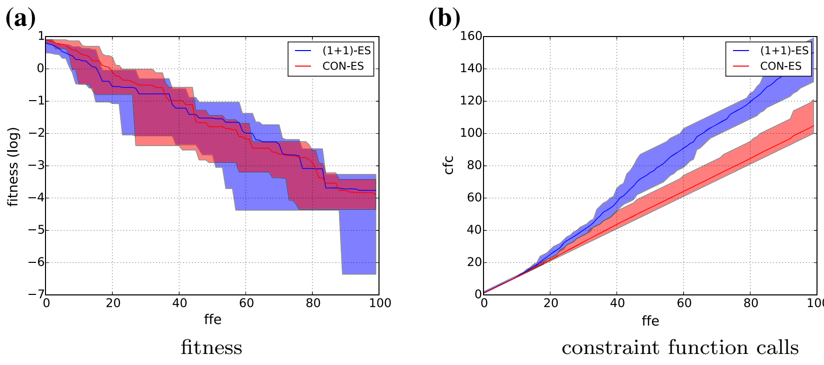

Fig. 7.3 Fitness development and constraint function calls of (1+1)-ES and CON-ES on Sphere with constraint, $N = 2$, and 100 generations

Table 7.2 Experimental comparison of (1+1)-ES without meta-model and CON-ES on the benchmark function set w.r.t. constraint function calls

Problem		(1+1)-ES		CON-ES	
	d	Mean	Dev	Mean	Dev
Sphere	2	74.08	46.46	52.35	30.88
	10	174.65	105.62	156.05	90.30
Tangent	2	87.3	52.76	55.30	33.47
	10	194.25	98.29	175.39	87.30

Figure 7.3 shows the average fitness development of each 30 (1+1)-ES and CON-ES runs and the corresponding number of constraint function savings on the Sphere problem with a constraint through the origin for $N = 2$. Figure 7.3a shows that both algorithms have the same capability to approximate the optimum. The main result is that the CON-ES is able to save a significant number of constraint function calls, see Fig. 7.3b. We observe similar results for the Tangent problem.

In the Table 7.2, we concentrate on the saving capabilities of the SVM constraint meta-model for the two dimensions $N = 2$ and 10 on the benchmark problems. The (1+1)-ES and the CON-ES terminate after 100 generations for $N = 2$ and after 300 generations for $N = 10$. In case of the Sphere with constraint, the optimum is approximated with arbitrary accuracy, while the search stagnates in the vicinity of the optimum in case of the Tangent problem, see [13]. The stagnation is caused by decreasing success rates due to contour lines that become parallel to the constraint boundary when the search approximates the optimum. The results show that the CON-ES saves constraint function evaluations in all cases, i.e., for both dimensions on both problems.

7.6 Conclusions

In practical optimization problems, constraints reduce the size of the feasible solution space and can increase the difficulty of the optimization problem. Many methods have been proposed for handling constraints. Penalty functions belong to the most popular ones. They penalize infeasible solutions by decreasing the fitness function value with a penalty term. The strengths of these penalties are usually controlled deterministically or adaptively.

The constraint handling problem is still not conclusively solved. In particular the reduction of constraint function calls is an important aspect. A promising direction is meta-modeling of the constraint boundary, an approach that is well-known for fitness function surrogates. In this scenario, machine learning models can be applied to learn the feasibility of the solution space based on examples from the past. If constraints deliver a binary value (feasible and infeasible), a classifier can be used as meta-model. We use SVMs to learn the constraints with cross-validation and grid search, while updating the model every 20 generations. For other benchmark problems, these settings have to be adapted accordingly. In the experiments with a (1+1)-ES on a benchmark function set, we demonstrate that significant savings of constraint function evaluations can be achieved. The trade-off between the frequency of meta-model updates and their accuracy has to be balanced accordingly for practical optimization processes.

The combination with fitness meta-models can simultaneously decrease the number of fitness and constraint function evaluations. The interactions between both meta-models have to be taken into account carefully to prevent negative effects.

References

1. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
2. Smola, A., Vapnik, V.: Support vector regression machines. Adv. Neural Inf. Process. Syst. **9**, 155–161 (1997)
3. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2009)
4. Beyer, H., Schwefel, H.: Evolution strategies: a comprehensive introduction. Natural Comput. **1**(1), 3–52 (2002)
5. Joines, J., Houck, C.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In: Fogel, D.B. (ed.) Proceedings of the 1st IEEE Conference on Evolutionary Computation, pp. 579–584. IEEE Press, Orlando (1994)
6. Koziel, S., Michalewicz, Z.: Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evol. Comput. **7**(1), 19–44 (1999)

7. Schoenauer, M., Michalewicz, Z.: Evolutionary computation at the edge of feasibility. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) Proceedings of the 4th Conference on Parallel Problem Solving from Nature, PPSN IV 1996, pp. 245–254. Springer, Berlin (1996)
8. Coello, C.A.: Constraint-handling using an evolutionary multiobjective optimization technique. Civil Eng. Environ. Syst. **17**, 319–346 (2000)
9. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, PPSN IV 2000, pp. 849–858. Paris, France, 18–20 September 2000
10. Kramer, O.: A review of constraint-handling techniques for evolution strategies. Appl. Comput. Int. Soft Comput. **2010**, 185063:1–185063:11 (2010)
11. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: International Conference on Evolutionary Computation, pp. 312–317 (1996)
12. Arnold, D.V., Hansen, N.: A (1+1)-CMA-ES for constrained optimisation. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2012, pp. 297–304. Philadelphia, PA, USA, 7–11 July 2012
13. Kramer, O.: Premature convergence in constrained continuous search spaces. In: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature, PPSN X 2008, pp. 62–71. Dortmund, Germany, 13–17 September 2008
14. Arnold, D.V., Brauer, D.: On the behaviour of the (1+1)-ES for a simple constrained problem. In: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature, PPSN X 2008, pp. 1–10. Dortmund, Germany, 13–17 September 2008
15. Chotard, A.A., Auger, A., Hansen, N.: Markov chain analysis of evolution strategies on a linear constraint optimization problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, pp. 159–166. Beijing, China, 6–11 July 2014
16. Poloczek, J., Kramer, O.: Multi-stage constraint surrogate models for evolution strategies. In: Proceedings of the 37th Annual German Conference on AI, KI 2014: Advances in Artificial Intelligence, pp. 255–266. Stuttgart, Germany, 22–26 September 2014
17. Kramer, O., Schlachter, U., Spreckels, V.: An adaptive penalty function with meta-modeling for constrained problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, pp. 1350–1354 (2013)
18. Kramer, O., Barthelmes, A., Rudolph, G.: Surrogate constraint functions for CMA evolution strategies. In: Proceedings of the 32nd Annual German Conference on AI, KI 2009: Advances in Artificial Intelligence, pp. 169–176. Paderborn, Germany, 15–18 September 2009
19. Gieseke, F., Kramer, O.: Towards non-linear constraint estimation for expensive optimization. In: Proceedings of the Applications of Evolutionary Computation—16th European Conference, EvoApplications 2013, pp. 459–468. Vienna, Austria, 3–5 April 2013