On constraint handling in surrogate-assisted evolutionary many-objective optimization

Tinkle Chugh, Karthik Sindhya, Kaisa Miettinen, Jussi Hakanen, Yaochu Jin*

University of Jyvaskyla, Department of Mathematical Information Technology
PO Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland
[tinkle.chugh, karthik.sindhya, kaisa.miettinen, jussi.hakanen, yaochu.jin]@jyu.fi

Abstract. Surrogate-assisted evolutionary multiobjective optimization algorithms are often used to solve computationally expensive problems. But their efficacy on handling constrained optimization problems having more than three objectives has not been widely studied. Particularly the issue of how feasible and infeasible solutions are handled in generating a data set for training a surrogate has not received much attention. In this paper, we use a recently proposed Kriging-assisted evolutionary algorithm for many-objective optimization and investigate the effect of infeasible solutions on the performance of the surrogates. We assume that constraint functions are computationally inexpensive and consider different ways of handling feasible and infeasible solutions for training the surrogate and examine them on different benchmark problems. Results on the comparison with a reference vector guided evolutionary algorithm show that it is vital for the success of the surrogate to properly deal with infeasible solutions.

1 Introduction

Problems involving several conflicting objective functions are called multiobjective optimization problems. Such problems are typical e.g. in industrial applications. Because of the conflict, there typically does not exist a single solution but multiple so-called Pareto optimal solutions. The set of all Pareto optimal solutions in the objective space is called a Pareto front. Problems involving more than three objectives are sometimes referred to as many-objective optimization problems. In industrial optimization problems, computationally expensive functions are common, where function evaluations are time-consuming because of employing e.g. finite element methods. Such problems are usually handled using surrogates, which are approximate functions that replace the computationally expensive ones. For overviews of surrogate-assisted evolutionary algorithms (SAEAs) for single and multiobjective optimization, see [1, 2]. Surrogate-assisted evolutionary algorithms for many-objective optimization have not received much attention but recently a novel Kriging-assisted evolutionary algorithm for many-objective optimization called K-RVEA [3] has been proposed.

* Yaochu Jin is also with the Department of Computer Science, University of Surrey, Guildford, United Kingdom

Although industrial problems involve constraints, they have received little attention in the literature. The constraints pose a challenge for evolutionary algorithms to generate feasible solutions. More importantly, the presence of both feasible and infeasible solutions within a population especially during early generations may cause problems in surrogate training. Usually, a feasible set of solutions is required to train the surrogate. In unconstrained problems, the solutions generated always lie in the feasible region while in constrained problems, it may not be the case. In many cases, an initial set of feasible solutions is not available or it may take a substantial number of function evaluations. In addition, infeasible solutions can also play a major role in updating the surrogates as it will affect the performance of the surrogates in subsequent generations.

Next, we present a summary of approaches used in the literature for constrained SAEAs. In [4–6], initial training of surrogates was performed without considering any information from infeasible solutions, while in [7] a prefixed number of feasible solutions was used to train Kriging models. For updating the surrogate, in [4], all feasible nondominated solutions from the latest generation were re-evaluated and added to the training data set. In [5], all nondominated solutions after using surrogates were reevaluated without considering the feasibility of the solutions. In [6, 7], the probability of feasibility was used for selecting individuals to update the surrogates. However, all these algorithms were tested on biobjective optimization problems. Therefore, a detailed investigation has not been done for handling infeasible solutions in constrained many-objective optimization.

In this study, we focus on constrained SAEAs for many-objective optimization problems and investigate three different approaches for creating a training data set for surrogates. In the first approach, we neglect all infeasible solutions and the surrogate is trained only with feasible solutions. In the second approach, we consider some infeasible solutions close to the feasible region in addition to the feasible ones and in the third approach, we add a penalty to infeasible solutions to train the surrogates. In all of these cases, we also consider infeasible solutions for selecting individuals to update the surrogates and to limit the size of the training data set. To update the surrogates, we select individuals so that a maximum number of feasible solutions is used without a compromise in convergence and diversity. A similar strategy is used to limit the size of the training data set. As this can affect the training time, we eliminate individuals in such a way that the performance of the surrogate is not compromised.

We assume that constraint functions themselves are not computationally expensive. In other words, the computation time of evaluating constraints is significantly lower than evaluating objective functions and therefore, surrogates are not trained for constraint functions. Such a scenario where constraint functions are not computationally expensive can exist in different cases. For instance, if objective and constraint functions are independently evaluated or constraints are available as analytical functions of the decision variables e.g. thickness to height ratio while considering the design of some structural part of an aircraft.

Regardless of this assumption, our major contribution is towards showing the effect of infeasible solutions in training surrogates.

To test different approaches for handling infeasible solutions, we use K-RVEA [3]. One of the main reasons and also an advantage to use K-RVEA is its ability to solve problems with more than three objectives. K-RVEA is based on the reference vector guided evolutionary algorithm RVEA [8], where the management of surrogates involves reference vectors. In RVEA, reference vectors are used to decompose the original problem into a number of subproblems. These subproblems are simultaneously solved and a set of solutions that approximate the entire Pareto front is finally obtained. Additionally, the balance of convergence and diversity of the solutions in the high-dimensional objective space is achieved by using a novel scalarization approach called angle penalized distance (APD) [8]. K-RVEA is an extension of this algorithm and it uses Kriging models as surrogates to approximate computationally expensive functions. A flowchart of K-RVEA is presented in Figure 1.

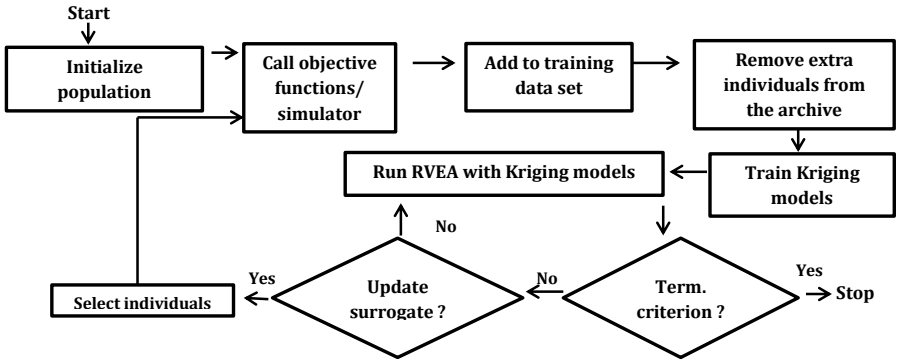

Fig. 1. Flowchart of K-RVEA

Initially, a population is initialized randomly or e.g. using Latin hypercube sampling [9]. Individuals of this population are then evaluated with the original objective functions and added to a training data set. If the size of this set exceeds a predefined limit, we eliminate individuals from it. Kriging models for each objective function are then trained and used to approximate objective function values. In any generation, if a termination criterion e.g. maximum number of function evaluations is not met, we update the surrogates after a prefixed number of generations. To update the surrogates, an efficient selection of individuals is performed with the help of reference vectors. Individuals are selected so that both convergence and diversity are managed while updating the surrogates. These individuals are then re-evaluated with the original functions and added to the training data set. If the termination criterion is met, nondominated solutions among all individuals evaluated with the original functions are obtained as the final solutions. For more details about K-RVEA, see [3].

In the next section, we provide the details of different approaches to handle infeasible solutions. In Section 3, we test and compare three approaches with the

constrained RVEA [8]. Finally, in Section 4, we conclude the paper and discuss future research directions.

2 Approaches to handle infeasible solutions

In this section, an extension of K-RVEA for constrained problems is presented, to be called cK-RVEA is given in Algorithm 1. cK-RVEA has three phases; initialization, using the surrogates and updating the surrogates.

Initialization In the initialization phase, an initial set of feasible and/or infeasible solutions is used to train the surrogates. It may be difficult to obtain enough feasible solutions in the first generation therefore, in some cases, we first find feasible solutions by optimizing the constraint violation as an objective function. These individuals are stored in a training data set A_1. In addition, another set A_2 is used as the storage of nondominated feasible solutions.

Using the surrogates In the phase of using the surrogates, Kriging models are used to approximate objective function values. We use Kriging up to a predefined fixed number of generations (w_{max}) before updating the surrogates. We use the same parameter for the prefixed number of generations that was proposed for K-RVEA based on a sensitivity analysis. For the selection criterion in this phase, an individual from each subpopulation with minimum APD is selected if it is feasible. Otherwise, an individual with a minimum constraint violation is selected. Individuals thus selected are used as the population for the next generation.

Algorithm 1 cK-RVEA

Input: FE_{max}= max number of function evaluations; w_{max}= prefixed number of generations before updating Kriging models; N_I= max number of individuals in set A_1
Output: nondominated feasible solutions of all evaluated ones from A_2
/*Initialization*/
1: Initialize the number of function evaluations FE=0, the generation counter for using Kriging models w=1 and a counter for the number of updates, $t_u = 0$. Initialize set $A_2 = \phi$
2: Obtain solutions (all feasible OR feasible and infeasible) in the training data set A_1 and update $A_2 = A_1$
3: Train a Kriging model for each objective function by using individuals in A_1
4: **while** $FE \leq FE_{max}$ **do**
/*Using the surrogates*/
5: **while** $w \leq w_{max}$ **do**
6: Run RVEA with Kriging models and update $w = w + 1$
7: **end while**
/*Updating the surrogates*/
8: Select a set of individuals U using a selection strategy to update the surrogates and re-evaluate them with the original functions and update $FE = FE + |U|$
9: Add individuals from step 8 to A_1 and A_2 and update $|A_1| = |A_1|+|U|$ and $|A_2| = |A_2|+|U|$
10: Remove $|A_1| - N_I$ individuals from A_1 using management of the training data, update $w = 1$ and $t_u = t_u + 1$ and go to step 3
11: **end while**

Updating the surrogates The Kriging models are updated after using them for a fixed number of generations. The selection of individuals to be re-evaluated is very important for the performance of the surrogates especially when constraints are involved. For example, it may be possible that after re-evaluations, the number of infeasible solutions increases. Therefore, a maximum number of

feasible solutions should be selected. In K-RVEA, a set of individuals U is selected based on the need of convergence or diversity. To this end, a fixed set of reference vectors (V_f) is generated in addition to the adaptive reference vectors (V_a). These reference vectors are used in the selection strategy to be described below.

Selection strategy to update the surrogates In K-RVEA, after using Kriging models for a fixed number of generations, individuals are assigned to the fixed reference vectors. Then the change in the number of inactive (or empty) fixed reference vectors from the previous update is calculated. If this change is smaller than a threshold, we select an individual with the minimum APD, otherwise with a maximum uncertainty (from the Kriging models). In cK-RVEA, we use APD or uncertainty if there is at least one feasible solution. Otherwise, we select an individual with a minimum constraint violation. Next, we provide a strategy to manage the training data set.

Managing the training data In order to reduce the computation time to train the Kriging models, we limit the size (maximum size is N_I) of the training data set. For this purpose, we eliminate some individuals from the set after every time we update the surrogates. We first assign individuals other than the recently evaluated ones to the adaptive reference vectors. These reference vectors are then clustered into a prefixed number of clusters and an individual either randomly (if feasible) or with a minimum constraint violation (if infeasible) is selected from each cluster. In this way, a fixed number of individuals is maintained in the training data set in order to improve the quality of Kriging models as much as possible while limiting the computation time.

In the following, we present three different approaches to handle infeasible solutions and variants of cK-RVEA using them are denoted by cK-RVEA1, cK-RVEA2 and cK-RVEA3.

Rejecting all infeasible solutions In cK-RVEA1, surrogates are trained only with feasible solutions. Using feasible solutions to train the surrogates can help in increasing their performance, especially when the feasible region is very small. This is the case for example in problem C1-DTLZ1 [10], which also contains many locally Pareto optimal solutions. If the surrogate is trained with infeasible solutions, the approximated values from it may be far from the feasible region. Therefore, it is appropriate to find feasible solutions first and then use the surrogate for approximating objective functions.

Using some infeasible solutions In cK-RVEA2, we use some infeasible solutions close to the feasible region in addition to the feasible ones to train the surrogates. The main advantage of this is that when infeasible solutions are also used for training, the surrogates may be able to approximate a more diverse area without too much reduction in their performance. However, how close and how many infeasible solutions should be used are two important challenges.

For both cases mentioned above i.e. cK-RVEA1 and cK-RVEA2, a single objective genetic algorithm with niche based selection [11] is used for considering constraint violation as the objective function to obtain a fixed number of solutions in the feasible region. The termination criterion in this algorithm is to

obtain adequate number of feasible solutions. A niche based selection ensures that a diverse set of feasible individuals in the decision space is obtained to train the surrogates. However, the diversity in the decision space does not guarantee diversity in the objective space and needs further attention.

Adding penalty to infeasible solutions In cK-RVEA3, we train surrogates with individuals generated randomly or e.g. with a Latin hypercube sampling and add a penalty to infeasible solutions. The main challenge in penalty based methods is to use an appropriate penalty parameter and we adopt here three methods from the study in [12]. In the first method, denoted by cK-RVEA3-I, a static penalty is added to each objective function value of f_i i.e.

$$f_i(x) = f_i(x) + R\sum_{j=1}^{m} |g_j(x)|, \tag{1}$$

where R is the penalty parameter and $|\ |$ denotes the absolute value of the constraint g_j. Note, however, that, we use $|\ |$ to represent the number of individuals in a set hereafter (except in (3)).

In the second method of using a penalty parameter denoted by cK-RVEA3-II, we adapt it with the number of feasible solutions obtained. After a certain number of function evaluations e.g. $FE \geq FE_{th}$, the penalty parameter R is decreased if the number of feasible solutions has increased from the previous generation and vice versa i.e.

$$R = \begin{cases} \frac{R}{c_1} & \text{if } FE \geq FE_{th}\ \Delta|P_f| > 0 \\ Rc_2 & \text{if } FE \geq FE_{th}\ \Delta|P_f| < 0 \end{cases} \tag{2}$$

where c_1 and c_2 are predefined parameters and $\Delta|P_f|$ denotes the change in the number of feasible solutions.

In the third method, cK-RVEA3-III, we use the method of parameter free penalty, where

$$f_i(x) = \begin{cases} f_i(x) & \text{if} \quad g_j(x) \geq 0,\ j = 1, \ldots, m \\ f_i^{max} + \sum_{j=1}^{m} |g_j(x)| & \text{otherwise} \end{cases} \tag{3}$$

where f_i^{max} is the maximum value of f_i at the current generation. The main advantage of using this method is that no parameter is included and infeasible solutions are always penalized.

3 Numerical experiments

In this section, results of experiments on the constrained versions of DTLZ problems [10] are presented. As mentioned, we consider three different approaches and compare them with each other and also with the constrained variant of RVEA [8]. Parameter values for niching are the same as used in [11] and values of parameters involved in K-RVEA are as follows: a) number of individuals to train the surrogate in initialization phase = number of reference vectors, N_I=50,

b) number of independent runs = 10, c) maximum number of function evaluations = 300 and d) number of generations before updating Kriging models, w_{max}=20. In addition, we introduced the following parameters in cK-RVEA: a) number of feasible solutions in cK-RVEA2 = 40, b) static penalty used in cK-RVEA3-I, R=10000, c) parameters used in cK-RVEA3-II (from [12]), c_1=3, c_2=4 and initial value of penalty parameter, R=1,

The number of decision variables was set to 10 for all problems and the number of constraints varied from one to ten. Inverted generational distance (IGD) was used as the performance measure and a Wilcoxon rank sum test analysis with a significance level of 5% was adopted to compare the results. Results for cK-RVEA1, cK-RVEA2, cK-RVEA3-I and cRVEA for different numbers of objectives (denoted by k) are reported in Table 1, where ↑ represents that cK-RVEA1 performed better than the other, ↓ means that it performed worse, while ≈ means that statistically there is no significant difference between the two algorithms.

Table 1. Results for IGD values obtained by cK-RVEA1, cK-RVEA2, cK-RVEA3-I and cRVEA. The best results are highlighted

Prob	k	cK-RVEA1				cK-RVEA2				cK-RVEA3-I				cRVEA		
		Min	Mean	Max		Min	Mean	Max		Min	Mean	Max		Min	Mean	Max
C1-DTLZ1	3	**0.098**	**0.154**	**0.166**	≈	0.147	0.159	0.168	↑	No feasible solution			↑	No feasible solution		
	6	0.148	0.176	**0.199**	≈	**0.107**	**0.174**	0.219	↑	No feasible solution			↑	No feasible solution		
	8	0.258	0.269	0.281	↓	**0.217**	**0.248**	**0.270**	↑	No feasible solution			↑	No feasible solution		
	10	0.197	**0.205**	**0.236**	≈	**0.166**	0.212	0.252	↑	0.309	0.359	0.420	↑	0.194	0.228	0.311
C2-DTLZ2	3	0.155	**0.213**	**0.271**	≈	0.189	0.215	0.283	↑	0.433	0.592	0.752	↑	0.205	0.260	0.291
	6	**0.373**	**0.388**	**0.407**	≈	**0.349**	0.406	0.443	↑	0.599	0.737	0.965	↑	0.389	0.435	0.530
	8	**0.387**	**0.479**	**0.598**	≈	0.424	0.542	0.755	↑	0.533	0.782	0.974	↑	0.522	0.601	0.703
	10	0.527	0.623	0.729	↑	0.571	0.727	0.878	↑	0.624	0.783	0.956	↑	0.571	0.615	0.673
C3-DTLZ4	3	**0.163**	**0.198**	0.256	≈	**0.160**	**0.187**	**0.216**	↑	0.183	0.249	0.386	≈	0.199	0.220	0.236
	6	**0.467**	**0.500**	**0.534**	≈	0.489	0.527	0.602	↑	0.537	0.587	0.646	↑	0.574	0.595	0.649
	8	**0.629**	**0.674**	**0.713**	≈	**0.602**	0.682	0.808	↑	0.713	0.801	0.856	↑	0.739	0.798	1.008
	10	0.779	0.860	0.903	↓	0.781	**0.824**	**0.897**	↑	0.891	0.991	1.299	≈	0.799	**0.836**	0.916

As can be seen, in C1-DTLZ1, cK-RVEA3-I and cRVEA were not able to find any feasible solutions in 300 function evaluations. The feasible region in C1-DTLZ1 is very small, therefore, using directly a surrogate or adding a penalty without finding feasible solutions was not useful for the surrogates as solutions are far from the feasible region. Therefore, it is important to find sufficiently many feasible solutions and then use surrogates. Both cK-RVEA1 and cK-RVEA2 found feasible solutions using the single objective genetic algorithm with constraint violation as the objective function.

We also performed a sensitivity analysis for the parameters of cK-RVEA2 i.e. the number of infeasible solutions and how close to the feasible region they should be. As mentioned in the parameter settings, the number of solutions was 50 to train the surrogates. In this sensitivity analysis, we used 10, 20 and 30 infeasible solutions out of 50 and rest of them were feasible. For each case, we changed the distance of solutions from the feasible region. To do that, we used the normalized constraint violation of 0.5, 0.25, 0.1 and 0.001. Therefore, all

together 12 studies were performed to analyze the number of infeasible solutions and their distance to the feasible region. Out of all these limited studies, the case with 10 infeasible solutions and the normalized constraint violation of 0.1 performed best and results from this case are shown in Table 1. However, self-adapting both the parameters is a future research topic.

Nondominated solutions of C2-DTLZ2 with three objectives of the run with the best IGD value from cK-RVEA1, cK-RVEA2, cK-RVEA3-I and cRVEA are shown in Figure 2. As can be seen, cK-RVEA1 and cK-RVEA2 performed comparably and solutions from both variants got close to the Pareto front. In contrast, solutions of cK-RVEA3-I, where a penalty is added to infeasible solutions did not converge to the Pareto front. However, when infeasible solutions were used in addition to feasible ones in cK-RVEA2, they got closer to the Pareto front. Parallel coordinate plots of C3-DTLZ4 with 10 objectives of the run with the best IGD values are shown in Figure 3. As can be seen, solutions from both cK-RVEA1 and cK-RVEA2 had large ranges in some of the objective values compared to other algorithms. Furthermore, as can be seen from the table, in most of the cases, the constrained variant of RVEA (i.e. without surrogates) performed worse than the others.

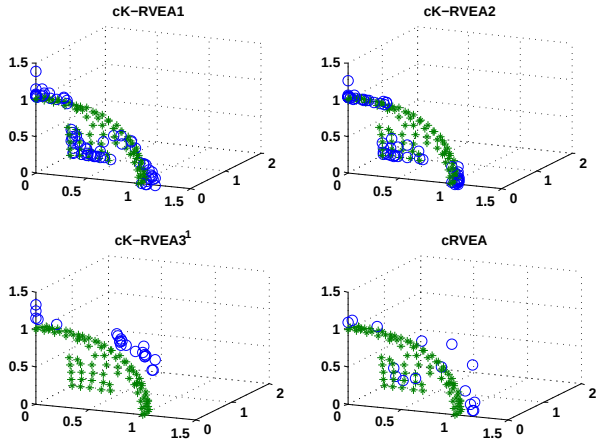

Fig. 2. Nondominated solutions obtained by cK-RVEA1, cK-RVEA2, cK-RVEA3-I and cRVEA denoted by circles of the run with the best IGD value for three-objective C2-DTLZ2 test problem. Here $*$'s represent the Pareto front.

When comparing penalty based methods as detailed in Section 2, results are given in Table 2. As can be seen, the adaptive penalty method in most of the cases performed equivalently or better than the static penalty method. In any case, the method of the parameter free penalty was not able to outperform other methods. All these results show the influence of infeasible solutions on the performance of the surrogates. These results indicate that an adaptive way of

handling infeasible solutions seems to be needed although more testing needs to be done on other benchmark problems.

Table 2. Results for IGD values obtained by cK-RVEA3-I, cK-RVEA3-II and cK-RVEA3-III. The best results are highlighted

Problem	k	cK-RVEA3-I				cK-RVEA3-II				cK-RVEA3-III		
		Min	Mean	Max		Min	Mean	Max		Min	Mean	Max
C1-DTLZ1	10	0.309	0.359	0.420	≈	**0.257**	**0.313**	**0.336**	≈	0.284	0.362	0.497
C2-DTLZ2	3	0.433	0.592	0.752	↓	**0.206**	**0.332**	**0.461**	≈	0.318	0.637	0.961
	6	0.599	0.737	0.965	↓	**0.528**	**0.636**	**0.913**	≈	0.570	0.850	1.036
	8	**0.533**	0.782	0.974	≈	0.579	**0.682**	**0.787**	≈	0.620	0.888	1.038
	10	0.624	0.783	0.956	≈	**0.575**	**0.704**	**0.867**	↑	0.632	0.903	0.998
C3-DTLZ4	3	0.183	0.249	0.386	↓	**0.180**	**0.200**	**0.222**	≈	0.195	0.251	0.293
	6	**0.537**	**0.587**	**0.646**	≈	0.554	0.599	0.723	↑	0.586	0.673	0.730
	8	**0.713**	**0.801**	**0.856**	≈	0.731	0.837	1.006	↑	0.772	0.938	1.052
	10	0.891	0.991	1.299	≈	**0.836**	**0.967**	**1.126**	↑	1.013	1.169	1.363

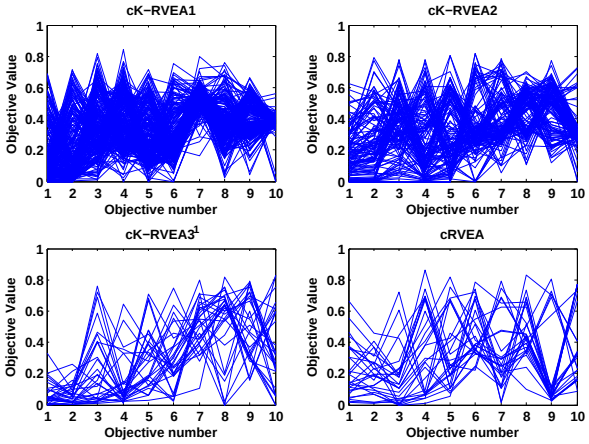

Fig. 3. Parallel coordinate plot of nondominated solutions obtained by cK-RVEA1, cK-RVEA2, cK-RVEA3-I and cRVEA of the run with the best IGD value on 10-objective C3-DTLZ4 test problem.

4 Conclusions and future research

In this paper, we investigated the influence of different constraint handling strategies for collecting training data on the performance of surrogates. These strategies were investigated on the constrained DTLZ problems using K-RVEA. Results from the study show that handling infeasible solutions in selecting training data is very important. Using only feasible solutions i.e. cK-RVEA1 in most cases performed better than others because individuals approximated by the surrogates lie in the feasible region. However, it depends on the problem used as

infeasible solutions may be helpful to increase the performance of the surrogates. Moreover, a hybrid approach to combine the different approaches e.g. how many and how close should be infeasible solutions to the feasible region, how to adapt the penalty parameter etc. can be beneficial. In addition, as few constrained many-objective optimization problems exist in the literature, developing and testing on new problems will also be our future work.

Acknowledgement

This work was supported by the FiDiPro project DeCoMo funded by TEKES, The Finnish Funding Agency for Innovation.

References

1. Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9:3–12, 2005.
2. T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen. Handling computationally expensive multiobjective optimization problems with evolutionary algorithms - A survey. Reports of the Department of Mathematical Information Technology, Series B, Scientific Computing no. B 4/2015, University of Jyvaskyla, 2015.
3. T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya. K-RVEA: A Kriging-assisted evolutionary algorithm for many-objective optimization. Reports of the Department of Mathematical Information Technology, Series B, Scientific Computing no. B 2/2016, University of Jyvaskyla, 2016.
4. H.K. Singh, T. Ray, and W. Smith. Surrogate assisted simulated annealing (SASA) for constrained multi-objective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.
5. G. Chen, X. Han, G. Liu, C. Jiang, and Z. Zhao. An efficient multi-objective optimization method for black-box functions using sequential approximate technique. *Applied Soft Computing*, 12:14–27, 2012.
6. P. Singh, I. Couckuyt, F. Ferranti, and T. Dhaene. A constrained multi-objective surrogate-based optimization algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3080–3087. IEEE, 2014.
7. J. Martinez-Frutos and D. Herrero-Perez. Kriging-based infill sampling criterion for constraint handling in multi-objective optimization. *Journal of Global Optimization*, 64:97–115, 2016.
8. R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. A reference vector guided evolutionary algorithm for many objective optimization. *IEEE Transactions on Evolutionary Computation, DOI: 10.1109/TEVC.2016.2519378*, 2016 (accepted).
9. M.D. Mckay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42:55–61, 2000.
10. H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18:602–622, 2014.
11. K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186:311–338, 2000.
12. K. Miettinen, M. M. Makela, and J. Toivanen. Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. *Journal of Global Optimization*, 27:427–446, 2003.