On the Performance of Metamodel Assisted MOEA/D

Wudong Liu[1], Qingfu Zhang[1], Edward Tsang[1], Botond Virginas[2]

[1] Department of Computer Science, University of Essex, UK
[2] BT Research Laboratories, UK

Abstract. MOEA/D is a novel and successful Multi-Objective Evolutionary Algorithms(MOEA) which utilizes the idea of problem decomposition to tackle the complexity from multiple objectives. It shows better performance than most nowadays mainstream MOEA methods in various test problems, especially on the quality of solution's distribution in the Pareto set. This paper aims to bring the strength of metamodel into MOEA/D to help the solving of expensive black-box multi-objective problems. Gaussian Random Field Metamodel(GRFM) is chosen as the approximation method. The performance is analyzed and compared on several test problems, which shows a promising perspective on this method.

1 Introduction

Multi-Objective Evolutionary Algorithms(MOEA) is an important stochastic search method for Multi-Objective Optimization(MOO), gaining more and more attention from both research and real application fields. However, when applying to real problem, one main defect of most MOEA methods is that they need great amount of function evaluations before find a good Pareto set, while it is not uncommon to see a black-box and expensive function(or functions) in real-life application solving. Here, we say a function to be expensive and black-box means the function has no analytic form; function value can only be obtained by evaluation; and it needs lots of resource(time, computation power, etc.) for that evaluation. With every evaluation costs a lot, MOEA, which normally need thousands of, cannot be directly applied in those applications.

The hardness of expensive block-box function had been confronted and well handled in many fields. The essential idea behind is to use an approximated cheap function to estimate the real function value, thus reduce the pricy evaluations. The concept was widely adpoted in fields such as global optimization[9], experimental design[16] and the large varieties of Response Surface Method[16,3], when facing expensive black-box functions. Many researches[4,8,23,20,18] had also been done in Evolutionary Computation(EC) for single objective problem(seeing [7] for a good survey on this topic) on this matter. However, because of the difference between single and multiple objectives problem solving, those methods cannot be readily applied in MOEA. Recent researches on MOEA with

metamodel assistance also appeared, notably in [11] and [5], which we think represent two kinds of methods to transform metamodel assistance from single objective to multiple objectives.

The first kind is to adapt the metamodel itself. In [5], the underlying statistics model of Gaussian Random Field Metamodel(GRFM) was extended from single objective to multi-objectives. Various screening methods from global optimization were naturally augmented to multi-objective case. Then the individual passing through the screening was evaluated. Another kind of approach decomposes the Multi-Objective Problem(MOP) into Single Objective Problems(SOPs) using weighted-sum[14] or Tchebycheff[14] aggregation function. As those aggregation functions' optimal are also Pareto solutions(and vice verse), this approach can make use of the existed large amount of algorithms and techniques from single objective optimization theory. In [11], a method called ParEGO was introduced. In every iteration of ParEGO, it uniformly randomly picks a aggregation weight and then using an augmented Tchebycheff to form a singular function from the multiple objective functions. Then EGO[10], a global optimization method based on GRFM, was used to approximate the singular function based on the existed evaluation point set and find the next evaluation point. Currently no comparison has been done on the performance of the above two approaches. As we also take the second approach in this paper, we will put our attention on the comparison between our method and ParEGO.

MOEA/D was first proposed and developed by Zhang and Li in [24]. It explicitly made use the concept of decomposition to simplify a MOP into a serial of subproblems in forms of SOP, using weighted-sum, Tchebycheff aggregation or other more advanced decomposing techniques such as Penalty-based boundary intersection(PBI)[2]. It exploited the benefit of that decomposition further by a neighborhood relationship defined on the subproblems. Subproblems in a neighborhood can exchange information to accelerate the convergence. It exhibited better performance on most standard test problems. However, no research have been done on the performance of MOEA/D with metamodel assistance on the black-box expensive functions. This paper will show that MOEA/D is so flexible that it still works very well in that situation. In our experiments, we also choose to use GRFM for the purpose of function approximation.

The paper is organized as follow: section 2.1 and section 2.2 give some description of the background on MOEA/D and the metamodel GRFM we used; section 3 details the algorithm, then some experiment results are given in section 4. We conclude the paper in section 5 and give some the future research directions on this matter.

2 Background

2.1 MOEA/D

Most MOEAs do not directly involve in decomposition of the problem; they normally treat a MOP as a whole. In scalar optimization, all solutions can be

Algorithm 1 MOEA/D pseudo code

```
weights[]                                          ▷ The subproblem weights
neighbours[][]                                     ▷ The neighbourhood table
pop[]                                              ▷ The main population
P, N, D   ▷ The population size, neighbourhood size and dimension size respectively

procedure MoeaD(objs)                              ▷ The main routine
    Initialize
    while Not Terminated do
        for i = 1 to m do
            ind ← GeneticOp(i)                     ▷ Generate new individual
            Evaluate(ind)                          ▷ Evaluate the new indivisual
            Update(i, ind)                         ▷ Update neighbourhood
        end for
    end while
end procedure
```

compared based on the single objective function. However, that is not the case in MOP, as domination does not define a complete order among the solutions in objective space. To use the readily made techniques for scalar objective optimization, many MOEAs focus on how to sort the solutions in objective space in a sensible order, and assign fitness according to that order to solutions. They normally aims to generate the whole Pareto front(PF).

MOEA/D took a different approach. On the consideration that in normal case, only a limited number of solutions are needed for decision making, MOEA/D aimed to solve only a subset of all the possible subproblems, which deemed to be more computational efficient and practical. A set of weights were fixed ahead, with each weight corresponding to one subproblem defined by weight-sum, Tchebycheff or any other decomposition approach. Then the neighborhood of every subproblem was constructed based on the distance between weights. Later, all the new solution generation and updating were performed on the neighborhood. The rationale behind the neighborhood is: for closer weights with a less distance, they have a larger chances to share similar good solutions. The using of neighbor greatly sped up the convergence to subproblems' optimal.

Those functions not defined in the pseudo code are implemented straightly forward and omitted from it to save the page. More detail on this method can be found in [24]. This algorithms worked so well that it outperformed the mainstream algorithm NSGA2 on a majority of test problems, as reported in [24].

2.2 Gaussian Random Field Metamodeling

GRFM has been long recognized as a powerful framework to build metamodel for arbitrary unknown functions, especially those are simulated by computer experiment[21]. It's essentially a statistics method based on Bayesian reasoning. It have been widely used[19,12,23,1] in Evolutionary Computation for metamodel assistance. Both the papers we mentioned in the introduction used this method.

Algorithm 2 MOEA/D pseudo code continue

```
procedure INITIALIZE
    weights[] ← WEIGHTGEN(P, D)
    neighbours[][] ← NEIGHBOURGEN(weights)
    pop[] ← INITPOP(m)
    for i = 1 to P do
        EVALUATE(pop[i]); UPDATE(i, pop[i])
    end for
end procedure
procedure UPDATE(index, ind)
    for i = 1 to N do                          ▷ Update the neighbour's solution
        nindex ← neighbour[index][i]; nweight ← weights[nindex]
        newval ← SCALARPROBLEM(nweight, ind)
        oldval ← SCALARPROBLEM(nweight, pop[nindex])
        if newval ≤ oldval then
            pop[nindex] ← ind
        end if
    end for
end procedure
procedure GENETICOP(index)
    neighbour[] ← neighbours[index]
    for i = 1 to N do
        j ← RANDOM(neighbour); k ← RANDOM(neighbour)
        ind ← CROSSOVER(pop[j], pop[k])
        ind ← MUTATE(ind)
        return ind
    end for
end procedure
```

In our implementation, a simplified version is used. We are not intended to gives the full derivation of the formula, but just gives the result here. To see more detail on this, please refer to[10,22].

Formally speaking, for an unknown function f, given the training set $T = \{\langle \mathbf{x}_i, f(\mathbf{x}_i)\rangle \mid i = 1..n, \mathbf{x}_i \in D \subseteq R^m\}$of size n, to predict the function value at a new point $y^* = f(\mathbf{x}^*), \mathbf{x}^* \in D$, the following formulas are used to estimate the value of y^*:

$$\hat{y} = \hat{\mu} + \mathbf{r}'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{1}$$

In the above equation, $\mathbf{y}$ is the vector of the training point's value: $y = \{f(\mathbf{x}_1), .., f(\mathbf{x}_n)\}$; $\hat{\mu}$ is the expected value of the estimated value, which will be given below shortly, $\mathbf{r}$ and $\mathbf{R}$ are the so called correlation vector and matrix respectively, their elements are computed as:

$$R(i,j) = \exp(-d(\mathbf{x}_i, \mathbf{x}_j)); \; r(i) = \exp(-d(\mathbf{x}_i, \mathbf{x}^*))$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$is the distance function:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{h=1}^{m} \theta_h |x_{ih} - x_{jh}|^{p_h}; \theta_h \geq 0 \wedge p_h \in [1, 2] \tag{2}$$

where x_{ih} is the hth element of vector $\mathbf{x}_i$; θ_h and ρ_h are the hyperparameters that need to be computed first; and that is actually where the learning procedure of Gaussian Process occurs. They are computed by maximize the following likelihood function:

$$\frac{1}{(2\pi)^{\frac{n}{2}} (\hat{\sigma})^n |\mathbf{R}|^{\frac{1}{2}}} \exp\left(-\frac{(\mathbf{y} - \mathbf{1}\hat{\mu})' \mathbf{R}^{-1} (y - 1\hat{\mu})}{2\hat{\sigma}^2}\right) \tag{3}$$

The $\hat{\mu}$ and $\hat{\sigma}$ are the expectation and standard deviation separately, which are obtained also by maximize the above likelihood when treat θ and ρ as constant:

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}'\mathbf{R}^{-\mathbf{1}}\mathbf{1}}; \hat{\sigma} = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})' \mathbf{R}^{-1} (y - 1\hat{\mu})}{n}$$

When substituting the above equations into the equation 3, we can get the so-called "concentrated likelihood function", which depends only upon the parameter θ_h and ρ_h. After maximize this likelihood function, we can get the value of θ_h and ρ_h, then put them back in the equation 1, we can get an approximation of y^*, namely $\hat{y}$.

One key feature of Gaussian prediction is that, not only a prediction is given, the estimation of the accuracy of this prediction is also given by:

$$s^2 = \sigma^2 \left(1 - \mathbf{r}'\mathbf{R}^{-1}\mathbf{r} + \frac{\left(1 - \mathbf{1}'\mathbf{R}^{-\mathbf{1}}\mathbf{r}\right)^2}{\mathbf{1}'\mathbf{R}^{-\mathbf{1}}\mathbf{r}}\right) \tag{4}$$

In this way, the prediction of $f(\mathbf{x}^*)$ is estimated to follow a normal distribution: $y^* \sim N\left(\hat{y}, s^2\right)$. The additional deviation information is of great value as it provides guidance for the search of the solution space, which will be seen in the next section.

3 MOEA/D with GP model assistant

If we want to solve a black-box expensive MOP, the target is to limit the number of evaluations as much as possible while can still get a reasonable closeness and distribution of the Pareto front. Here, we present an analysis on the performance of MOEA/D with GP model assistance.

We take a traditional way to do the job, i.e., using model as a prescreening mechanism[5]. Similar to ParEGO, at every iteration, MOEA/D runs on a learned model, then the best result is selected for evaluation. However, our approach is rather different than ParEGO's. In ParEGO, at every iteration, a GP model is constructed for one selected scalar subproblem. However, as in MOEA/D, we maintain a set of subproblems, it may not be appropriate or

practical to construct GP models for every subproblem when the normal size of subproblems ranges from 20 to 100.

Taken the above consideration, we model every objective function instead, and construct the model for the subproblem model based on the objective models. Given a MOP with objective function $\mathbf{f} = \langle f_1, ..f_n \rangle$, we construct the GP model at every generation as $\hat{\mathbf{f}} = \left\langle \hat{f}_1, ..\hat{f}_n \right\rangle$, with every $\hat{f}_i$ is estimated to follow normal distribution $\hat{f}_i \sim \mathbf{N}\left(\hat{y}_i, s_i^2\right)$. Then in weight-sum decomposition, the scalar subproblem corresponding to weight $\mathbf{w} = \langle w_1, .., w_n \rangle$ of form $sf = \sum_{i=1}^{n} w_i f_i$ can be estimated as:

$$\hat{sf} \sim \mathbf{N}\left(\sum_{i=1}^{n} w_i \hat{y}_i, \sum_{i=1}^{n} \left(w_i s_i\right)^2\right) \tag{5}$$

Also, as suggested in [9], the optimal of the model cannot be directly evaluated because of false convergence or local optimal. There must be some balance between exploit and explore. Thus, as in ParEGO, the concept of Expected Improvement[15] for the normal distribution is also considered here. Formally speaking, in the training set T, if the minimal value for f is f_{min}, then for every other point $\mathbf{x}$, which $f(\mathbf{x})$ is estimated to have a mean $\hat{y}$ and standard deviation s(just as given by the GP model), the *Expected Improvement* over $f_{min}$is defined as[9]:

$$EI(\mathbf{x}) = E\left(\max\left(f(x) - f_{min}, 0\right)\right) = s\left(u\Phi(u) + \phi(u)\right) \tag{6}$$

where $u = \frac{f_{min} - \hat{y}}{s}$, Φ and ϕ are the normal cumulative distribution function and density function respectively. Thus for every maintained scalar function, we need to find a point that maximize the formula 6; The objective is changed from finding its minimal to finding the maximal expected improvement. This should be reflected in MOEA/D when updating the main population. If we maintain n subproblems, at every iteration we could get n points that could act as candidates for real evaluation. To choose one from them, we simply choose the point that has a maximal sum of all the EI over all the subproblems.

As a good solution from an iteration of MOEA/D could possible still be a good solution in the next iteration, a good portion of main population of MOEA/D should be kept between iterations. In our implementation, a k-mean cluster[13] algorithm is used. The number of k is chosen to be $S/5$(the size of main population divided by 5). Then, from each cluster, the best individual that is closet to the cluster's center is reserved. All the other individuals in the main population are randomized. The cluster technique proved to work well in our settings.

After taken the above decisions, all other things are rather straight forward. Thus we could have algorithm 3.

Algorithm 3 MOEA/D with GP model

evalpop[] ▷ The already evaluaed points
gpmodel[] ▷ The GP models learned from the evaluated points

procedure GPMoeaD
  InitGPModel
  while Not Terminated **do**
    MoeaD(*gpmodel*)
    $ind \leftarrow \max_{ind \in mainpop}(\sum_{sf} EI(ind, sf))$
    Evaluate(*ind*)
    $evalpop[] \leftarrow evalpop[] \cup ind$
    UpdateGP
    ClusterPop($mainpop, S/5$)
  end while
end procedure
procedure UpdateGP
  for $i = 1$ to d **do**
    $gpmodel[i] \leftarrow \mathrm{GP}(evalpop, obj[i])$
  end for
end procedure

4 Experiment and Comparison

This section deals with experiments on several standard test problems. Even thought this test problems are not black-box expensive, we limit the number of evaluation to 100, 150, or 250, which makes them "expensive" essentially. On every problem, we compared three settings of test: MOEA/D without evaluation limit, MOEA/D with GP assistant, and ParEGO. The test problem chosen are ZDT1(150), KNO1(100), OKA1(100); the number in parenthesis indicates the evaluation limit.

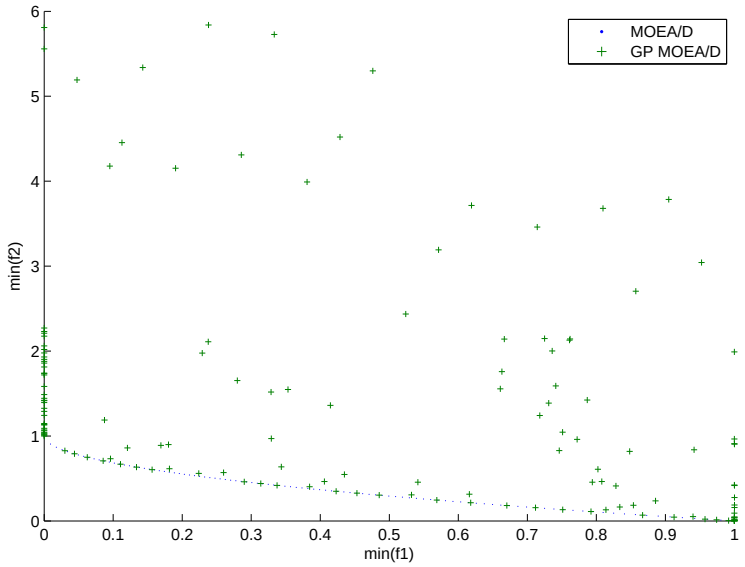
Two metrics are chosen for measurement of the quality of result: D-metric and I-metric. D-metric measure the distance from the Pareto Front(PF) and is calculated as: $D(A, P^*) = \frac{\sum_{v \in P^*} d(v,A)}{|P^*|}$, where A is the approximation to the PF, P^* is the set of uniformly distributed points along the PF, and $d(v, A)$ is the minimal Euclidean distance between v and A. While $|P^*|$ is large enough, D-metric measure both diversity and convergence in a sense. I-metric measure the hypervolume difference between the PF and the approximation set, it's calculated as: $I(A, P^*, R) = H(P^*, R) - H(A, R)$, where H is the hypervolume indicator, or S-metric defined in [25], measures how much volume dominated between the set and R(a reference point). Both D and I are the smaller, the better. All the problems are tested for 10 times and the average and deviation are computed on D and I. The experiment result are listed in table 1.

Experiment shows MOEA/D GP works extremely well on ZDT1(4 parameters), not only in the metric D and I, but in the figure. On KNO1, MOEA/D GP works also very well, comparing to ParEGO and NSGA2. On OKA1, both

Table 1. D and I Metric

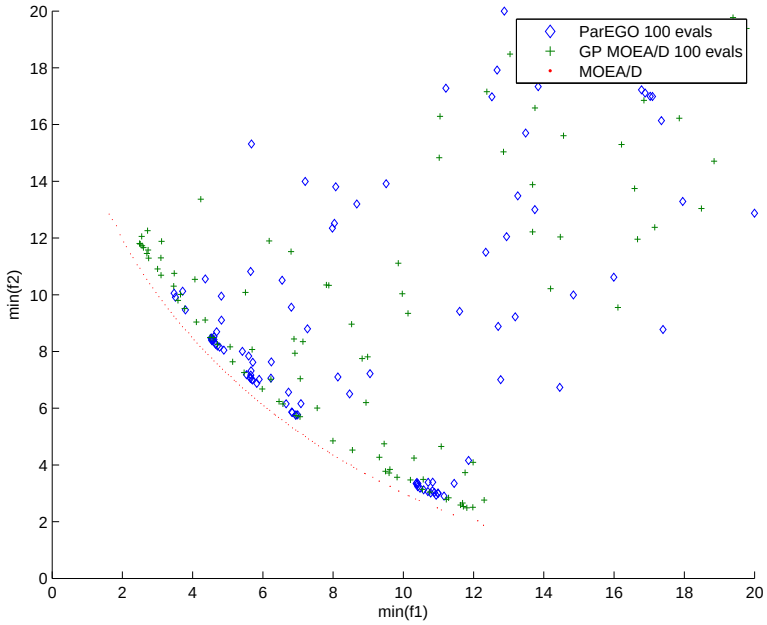
	MOEA/D GP		ParEGO		NSGA2	
instance	D	I	D	I	D	I
ZDT1	0.01(6.61e-4)	0.02(0.03)	-	-	-	-
KNO1	0.54(0.12)	21.25(5.70)	1.66(0.31)	47.51(8.32)	1.70(0.54)	53.80(16.62)
OKA1	1.43(1.06)	5.24(4.75)	0.65(0.05)	6.10(0.55)	0.95(0.15)	13.64(2.77)

Fig. 1. ZDT1 with 4 parameters, 150 evaluations



the metric cannot show exactly which is better(MOEA/D GP is better in I but worse in D). The reason may lie behind the complexity of OKA1 itself[17], which make it very hard for any algorithm to approximate with mere 100 evaluations.

Fig. 2. KNO1 with 100 evaluation



The experiments demonstrate that on the selected test problems, MOEA/D GP exhibits good and consistent performances that at least no worse than ParEGO and NSGA2 on the black-box and expensive condition. To be noted that, as most settings used in this initial experiments are rather primitive and reserved, MOEA/D GP should gain better result after refinement such as normalization, advanced decomposition techniques(both was taken by ParEGO), etc.

5 Conclusion and Future Study

The paper shows some initial result from the experimenting of a new method to solve black-box expensive multi-objective problems using the newly introduced algorithm MOEA/D.

Although MOEA/D GP's performance was good on the selected test functions, the research is only in primitive and tentative status, which at least can be improved in the following way:

- Advanced decomposition techniques: MOEA/D GP does not work well on concave problem such as VLMOP2 due to the used weight-sum decomposition. The choosing of weight-sum here is just for simplicity because the distribution as solved in equation 5 is hard (but not impossible) to solve for Tchebycheff.
- Alternative exploration techniques: in recent research of global optimization as surveyed in [9], new techniques to replace Expected Improvement have been introduced and exhibited better performance. In [6], augmented EI was proposed . Those advance in single objective optimization can be readily borrowed in the framework of MOEA/D and better performance could be expected in MOEA/D GP.
- Better selection strategy: when choose the next point for evaluation, a simple strategy is applied to choose the best sum of EI over all subproblem. However, as every subproblem has its own improvement measurement, the strategy may not works well especially when no normalization is done beforehand. Some other smarter strategy can be imaged to speed up the convergence, such as to maximize the best improvement percentage.
- Dynamic weights control: the weights used to decompose the MOP in MOEA/D were fixed, which made it difficult to generate an evenly distributed Pareto set when the weights set size is limited for some known test problem(OKA1). Some dynamic control machenism of weights may help solve this problem after obversing the population's distribution from generation to generation.

The future work will continue on the directions suggested above, and try it on some realistic problems such as BT's multi-objective workforce scheduling problem.

References

1. D. Buche, N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Trans. Syst., Man, Cybern. C*, 35(2):184–194, 2005.
2. I. Das and J.E. Dennis. Normal-boundary intersection: A new method for generating pareto optimal points in multicriteria optimization problems. *SIAM J. Optim.*, 8(3):631–657, 1998.
3. T.A. Donnelly. Response-surface experimental design. *Potentials, IEEE*, 11(1):19–21, Feb 1992.
4. M.A. El-Beltagy and A.J. Keane. Evolutionary optimization for computationally expensive problems using gaussian processes. In *Proc. Int. Conf. on Artificial Intelligence ICAI 2001*, pages 708–714, Las Vegas, 2001.
5. M.T.M. Emmerich, K.C. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *Evolutionary Computation, IEEE Transactions on*, 10(4):421–439, Aug. 2006.
6. D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466, 2006.

7. Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1):3–12, 2005.
8. Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *Evolutionary Computation, IEEE Transactions on*, 6(5):481–494, Oct. 2002.
9. Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.
10. D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.
11. J. Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *Evolutionary Computation, IEEE Transactions on*, 10(1):50–66, Feb. 2006.
12. K.H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):172–183, 2000.
13. J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proc. of the 5th Berkeley Symp. on Mathematics Statistics and Probability*, 1967.
14. K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer, Norwekk, MA, 1999.
15. J. Mockus. *Bayesian approach to global optimization*. Kluwer, Dordrecht, 1989.
16. Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons, 2 edition, 1984.
17. Tatsuya Okabe, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. On test functions for evolutionary multi-objective optimization. In Xin Yao et al., editor, *Parallel Problem Solving from Nature - PPSN VIII*, pages 792–802, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.
18. I. Paenke, J. Branke, and Yaochu Jin. Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *Evolutionary Computation, IEEE Transactions on*, 10(4):405–420, Aug. 2006.
19. A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. *Parallel Problem Solving from Nature-PPSN V*, pages 87–96, 1998.
20. R.G. Regis and C.A. Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *Evolutionary Computation, IEEE Transactions on*, 8(5):490–505, Oct. 2004.
21. J. Sacks, W. Welch, T. Mitcheel, and H. Wynn. Design and analysis of computer experiments(with discussion). *Statist. Sci.*, 4:409–435, 1989.
22. Matthias Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, 1997.
23. H. Ulmer, F. Streichert, and A. Zell. Evolution strategies with controlled model assistance. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, 2004.
24. Qingfu Zhang and Hui Li. A multiobjective evolutionary algorithms based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 2007.
25. Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.