Voronoi-based Estimation of Distribution Algorithm for Multi-objective Optimization

Tatsuya Okabe, Yaochu Jin, Bernhard Sendhoff, Markus Olhofer

2004

Preprint:

This is an accepted article published in Congress on Evolutionary Computation (CEC). The final authenticated version is available online at: https://doi.org/[DOI not available]

Published in: Congress on Evolutionary Computation. pp. 1594-1602, Portland, OR, 2004

Voronoi-based Estimation of Distribution Algorithm for Multi-objective Optimization

Tatsuya Okabe, Yaochu Jin, Bernhard Sendhoff and Markus Olhofer
Honda Research Institute Europe GmbH
Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany
Email: {tatsuya.okabe, yaochu.jin, bernhard.sendhoff, markus.olhofer}@honda-ri.de

***Abstract*— The distribution of the Pareto-optimal solutions often has a clear structure. To adapt evolutionary algorithms to the structure of a multi-objective optimization problem, either an adaptive representation or adaptive genetic operators should be employed. In this paper, we suggest an estimation of distribution algorithm for solving multi-objective optimization, which is able to adjust its reproduction process to the problem structure. For this purpose, a new algorithm called *Voronoi-based Estimation of Distribution Algorithm (VEDA)* is proposed. In VEDA, a Voronoi diagrams are used to construct stochastic models, based on which new offspring will be generated. Empirical comparisons of the VEDA with other estimation of distribution algorithms (EDAs) and the popular NSGA-II algorithm are carried out. In addition, representation of Pareto-optimal solutions using a mathematical model rather than a solution set is also discussed.**

I. Introduction

The search behavior of evolutionary algorithms (EAs) strongly depends on the representation and the genetic operators. The adaptation of the representation and/or the operators have been very successful for adjusting the search behavior to the local structure of the search space, examples are the self-adaptation principle in evolution strategies [21], operator adaptation in the structure optimization [8] or adaptive representations in design optimization [18]. For the area of multi-objective optimization (MOO), a hybrid representation (HR) has been proposed to utilize different search behavior in one algorithm in [15]. The HR showed better performance than the state-of-the-art multi-objective optimizers. As a natural extension of the HR algorithm, we consider in this paper another class of evolutionary algorithms, often known as *Estimation of Distribution Algorithms (EDAs)* [12], which are believed to have the ability to adapt their search distributions to the problem structure efficiently. The basic idea of EDAs is to build a stochastic model from the parental distribution in the parameter space and to generate offspring individuals by sampling from the model. Thus, crossover and mutation in standard genetic algorithms are replaced by estimation of the distribution and offspring generation from the estimated distribution.

A new algorithm, termed as Voronoi-based estimation of distribution algorithm (VEDA), will be proposed in this paper. It is motivated from the observations on the intrinsic interplay between the distributions in the parameter space and in the fitness space [14], [15]. The main purpose is to take advantage of the regularities (problem structure) in MOO by directly estimating the most appropriate search distribution. In estimating the search distribution, not only the selected individuals, but also those that are not selected are taken into account. The reason is that inferior solutions are also able to provide useful information for efficient search. This is of particular importance when fitness evaluations are computationally expensive. In contrast, other EDAs as well as EAs often neglect the information contained in the inferior individuals because only information of selected individual are used to generate offspring. An illustration is drawn in Figure 1. In the upper-right figure, only information of selected individuals are used to generate offspring. Since non-selected (inferior) individuals are not taken into account, the probability to generate offspring near inferior individuals is rather high. Our basic idea is to exploit all available information to overcome this drawback (see the bottom figures), so that the probability to generate offspring near inferior individuals is accordingly low.

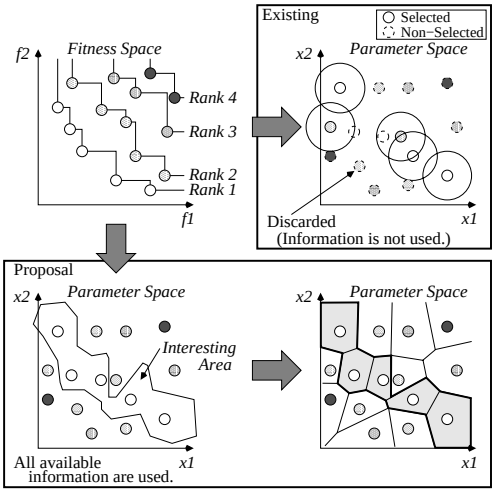


Fig. 1. Basic Idea of *Voronoi-based Estimation of Distribution Algorithm (VEDA)*. All available information are used to generate promising offspring efficiently. In other evolutionary algorithms (EAs) including other EDAs, only information from selected individuals are used.

The VEDA employs Voronoi diagrams to generate stochastic models, which cover the parameter space. A Voronoi diagram is a method for partitioning a space related to the nearest-neighbor partition, which can be defined as follows [7]:

Definition: Voronoi mesh
Given a set S of m data points in $\Re^n$, Voronoi mesh is the partition of $\Re^n$ into m polyhedral cells, $v(p)$ $(p \in S)$. Each cell $v(p)$, namely *Voronoi mesh* of p, is defined as the set of points in $\Re^n$ which are closer to p than to any other points in S, or more precisely, $v(p) = \{x \in \Re^n \ ; \ \text{dist}(x,p) \leq \text{dist}(x,q) \ \forall q \in S \setminus p\}$, where *dist* is the Euclidean distance.

Clustering techniques [11] have been used to divide the selected individuals into a number of groups and then one Voronoi diagram (one model) will be generated for one group. Meanwhile, principal component analysis (PCA) [10] has been employed to reduce the dimensionality. Both measures are very important in higher dimensional cases, where it becomes more time-consuming to generate stochastic models. Finally, new offspring will be generated by sampling from the generated models.

The rest of this paper is organized as follows. Related work is explained in Section II. Section III describes the main steps of VEDA in detail. The proposed method is tested on several MOO test functions and the results are shown in Section IV. Finally, a summary of the paper will be given in Section V.

II. Related Work

Genetic algorithms (GAs) are well known to be powerful tools to obtain optimal solutions for complex optimization problems. As a variance of GAs, estimation of distribution algorithms (EDAs) have recently received considerable attention [23].

According to recent surveys [12], [20], they can be classified into three classes according to the interactions in the models, i.e., no interaction, pairwise interaction and multivariable interaction. The first class of models does not consider any epistasis. Thus, each locus can be treated independently. The second class considers only pairwise dependency. Finally, models with multivariable interaction are able to take any type of dependency between variables into account. Since the model proposed in this paper belongs to the class of multivariable interaction, some of the popular methods in this category will be discussed next.

Bayesian Optimization Algorithm
To learn the linkage between parameters and therefore the structure of the problem, Bayesian networks are used, e.g. in [13]. With the Bayesian networks, the conditional probability is approximated. Each node and connection in the Bayesian networks correspond to the parameters and the conditional probability, respectively. Finally, the factorized probability is used to generate offspring. Recently, this method has been applied to MOO problems [13].

Iterated Density Estimation Evolutionary Algorithm (IDEA)
Bosman and Thierens have proposed four types of EDAs that all belong to the class of IDEA [1]. The first one is for the discrete domain where the conditional probability is used to build up the stochastic model. The others are for the continuous domain. A normalized Gaussian, a histogram method and a kernel method are used to generate stochastic models. They have mentioned that the use of a mixture distribution gives a powerful representation of complicated dependencies. The kernel-based method has been also applied to MOO, which is termed mixture-based IDEA [22].

Parzen-Based Estimation of Distribution Algorithm
To generate stochastic models, a Parzen estimator is used to approximate the probability density of solutions [3]. The Parzen method pursues a non-parametric approach to kernel density estimation. This method has been used for MOO [3].

Marginal Histogram Model
For each parameter, the search space is divided into small bins [23]. The ratio of the number of individuals in each bin to the whole number is assigned as the selection probability. With this probability, a bin is selected randomly. In the selected bin, an offspring is generated uniformly. Tsutsui et al. [23] have also pointed out the problem of an exponentially growing number of bins in higher dimensional cases.

III. Voronoi-based Estimation of Distribution Algorithm

In this section, details of the VEDA will be presented. The basic flow of the VEDA is shown in Figure 2. As pointed out in [20], construction of the stochastic model and generation of new offspring from the stochastic model are the two main issues in EDAs. Therefore, we show more details of these steps in Figures 3 (a) and (b), respectively.

In Figure 2, a stochastic model will be generated from a database. If the database does not exist, initial data will be generated randomly and evaluated. Based on the stochastic model, new promising individuals will be generated and evaluated. With the fast ranking method [5], the rank of the individuals will be calculated. Using the crowded tournament selection [5], individuals will be selected and stored in the database. The non-selected individuals will be stored in a different database as ”bad examples”. Note that when new data are added to the database, the stored rank information can become incorrect and thus should be updated. The maintenance of the databases will be explained in Section III-F in more detail. If a given termination condition is met, the VEDA will stop, otherwise the same procedure will be repeated. In the database, design parameters, fitness values and ranks are stored.

Figure 3 explains the details of the approach for building up the model and for generating offspring. To construct stochastic models, a clustering method [11] is used. In each cluster, principal component analysis (PCA) [10] is carried out to reduce the dimensionality and to build up a stochastic model efficiently. The data points will be projected to a new coordinate system of a lower dimensionality determined by PCA. The minimum and maximum value for each axis will be calculated. Since PCA was carried out, linear dependency among the design parameters should be minimal. In the new coordinate system, a Voronoi diagram will be generated as the stochastic model. Based on the rank of each mesh in

the diagram, the probability for generating offspring will be calculated for each mesh. To generate a new individual, a mesh will be selected based on the assigned probability and a new individual will be generated within the selected mesh with a uniform probability. Finally, the new individual will be projected back to the real coordinate system.

Although the Voronoi diagram seems to be similar to the histogram method, the method is different in the following aspects: (1) The shape of a mesh in the Voronoi diagram changes with the distribution of the data. (2) The number of data points in a mesh will not be counted. Whereas the histogram method needs a large amount of data in high-dimensional cases, the Voronoi approach in VEDA requires only a small amount of data. The reason is that VEDA uses the rank instead of the number of data points in a mesh to estimate the probability; (3) A bin without data has a probability of zero in the histogram method. In contrast, the probability in the region where no data points exist is approximated using the nearest data point in a Voronoi diagram. Thus, the probability of the unexplored region is not always zero.

The model construction and the offspring generation will be explained in more detail in the following sections. Additionally, maintenance of the databases will be explained in Section III-F.

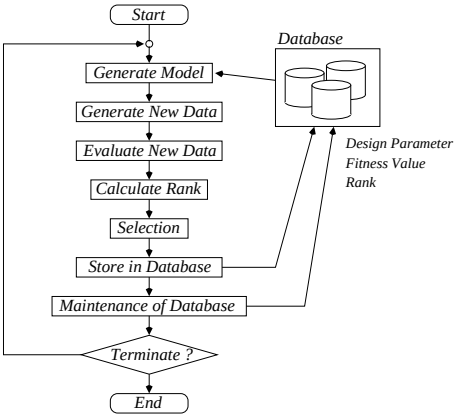


Fig. 2. The basic flow of VEDA.

A. Clustering

In the VEDA, the selected individuals are grouped into a number of clusters at first. Local models instead of a global model will be constructed for each cluster. For this purpose, a clustering method is needed to group the data. Refer to Figure 4. In this paper, the k-means clustering proposed by MacQueen [11] has been used. To use the k-means clustering, one has to determine the value of k, i.e., the number of clusters. In this paper, the value of k will be determined at random within a range of $[1, 10]$. Better performance can be reached when k is set using some a priori knowledge about the Pareto front in the parameter space. For example, the number of clusters can be determined by the number of disconnected pieces of the Pareto front. This implies that k could equal 1. However, this kind of *a priori* knowledge is often not available in real-world applications. In simulations, we do not use such knowledge to allow for a fair comparison with other algorithms.

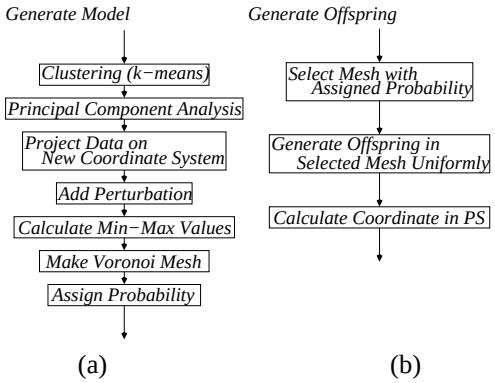


Fig. 3. The detailed flow in VEDA. (a) Flow for generating a stochastic model. (b) Flow for generating offspring.

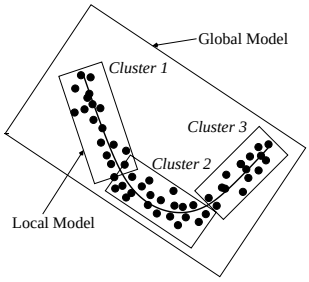


Fig. 4. Three clusters determined by *k-means clustering*. It is obvious that it will be easier to identify stochastic models for each of the clusters separately instead of for the whole data set.

B. Principal Component Analysis

To reduce the dimensionality and to construct models more efficiently, PCA is used in this paper. Figure 5 shows two different data sets. If there is epistasis between the variables, refer to Figure 5 (b), it is reasonable to map them onto a coordinate system that can minimize the linear correlations, see Figure 5 (a). Thereby, we can reduce the dimensionality and build models more efficiently.

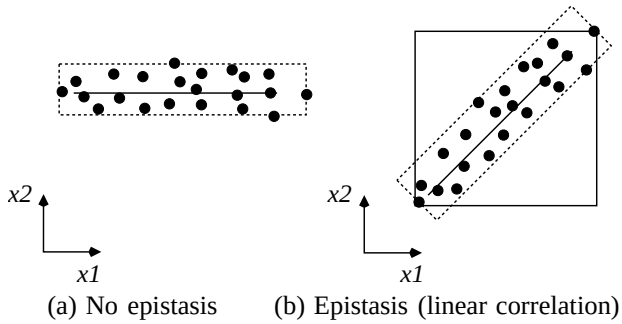


Fig. 5. Sample data sets. (a) Data without epistasis. (b) Data with epistasis.

C. Perturbation of Offspring Distribution

In real-coded genetic algorithms (RCGAs), e.g. [6], [19], the PCA or Gram-Schmidt orthogonalization have also been used. Unimodal normal distribution crossover (UNDX) [19] and parent-centric recombination (PCX) [6] are examples in which orthogonalization is used. However, the use of orthogonalization in MOO may give rise to problems. In Figure 6, the results of UNDX on SCH1 with 50 dimension are shown[1]. Clearly, the solution sets are not on the Pareto front. However, they seem to be on similar curves. Taking a closer look, we find out the reason for this problem. If all solutions are on one line in the parameter space, the algorithm loses its search ability in the orthogonal direction. A rough illustration is drawn in Figure 7. Since all solutions lie on one line which is not the Pareto front, all offspring will be generated on this line. Thus, there is no way to find the Pareto front. This happens in the VEDA too, which can be regarded as a kind of premature convergence.

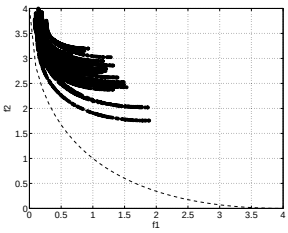


Fig. 6. The results of UNDX on SCH1 ($n = 50$) with 30 runs. The crowded tournament selection is used. Dotted curve is the Pareto front.

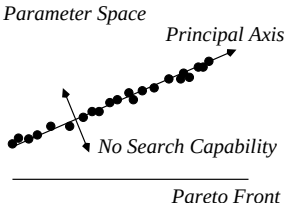


Fig. 7. The search power for the orthogonal direction will be lost, if all solutions are on one line.

To avoid this premature convergence, perturbation in the orthogonal direction[2] is introduced, see Figure 8 for two dimensional cases. In Figure 8, the "obtained area" is given by the maximum and the minimum value in each direction. The widths of the obtained area are β and γ in the principal direction and the orthogonal direction, respectively. Basically, offspring will be generated in this obtained area. However, if offspring are generated in this area only, the above problem can occur. Thus, perturbation is introduced. The "obtained area" is shifted in the orthogonal direction by an offset δ given by $\delta = \pm 0.25\gamma$. The direction, i.e., "+" or "-", is chosen randomly. Furthermore, the width β is enlarged: $\alpha = 1.25\beta$. The new "generative area" is given by α times γ shifted by δ. Although several parameters were introduced and empirically determined, the performance of VEDA seems to be robust against these parameters. Thus, we fix these parameters and leave them unchanged throughout this paper.

[1]The used algorithm is NSGA-II proposed in [5]. But, the UNDX is used instead of simulated binary crossover (SBX).

[2]In the reduced space, we term the axis with the maximum spread the *principal axis* (direction) and the others *orthogonal axes* (directions).

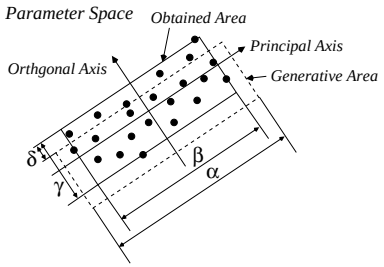


Fig. 8. Perturbation added in the orthogonal direction and extension in the principal direction. β and γ are the differences between the maximum value and the minimum value in the principal and the orthogonal direction. δ is the perturbation in the orthogonal direction. α is the extension for the principal direction.

D. Voronoi-based Stochastic Model

To simplify the generation of the Voronoi diagram, a discrete Voronoi diagram is used, see Figure 9.

The procedure for generating a discrete Voronoi diagram for a cluster is illustrated in Figure 10. In generating a discrete diagram, the concerned space is at first divided into a number of small grids. The number of discretization level in each axis (in the reduced space), D_s, is determined as follows: $D_s = \varepsilon \times |N_C|$, here, ε and $|N_C|$ are a predefined parameter and the number of data points in a cluster.

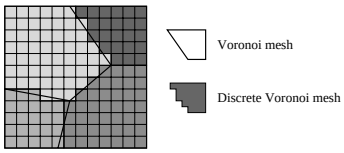


Fig. 9. A discrete Voronoi diagram.

One grid contains at most one individual (data point), which is denoted by a filled circle in the figure. If there is an individual in a grid, its rank is assigned to the grid. Then, this rank is also assigned to the neighboring grids in the same row or in the same column. This step is indicated by *1* in Figure 10. Next, the neighboring grids of the grids indicated by *1* are also assigned the same rank, which are now indicated by *2*. This procedure continues until all grids which do not contain a data point are assigned with the rank. If a grid is in the neighborhood of more than one individual (data point), the lowest rank will be assigned to this grid. All grids with the same rank is grouped into a mesh, in which only one individual exists.

E. Generate Offspring

In the previous subsection, all meshes in the Voronoi diagram are assigned a rank. With the assigned rank, the selection probability can be calculated. To calculate the selection probability P, the geometry distribution is used. Mathematically, the geometry distribution is calculated as follows:
$P = P_G(1 - P_G)^{r-1}$, here, P_G and r are the given value in the range $[0.0, 1.0]$ and the rank, respectively.

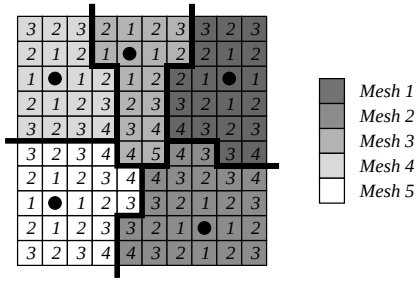


Fig. 10. The sample flow of generation of the Voronoi diagram. The thicker lines show the boundary of the Voronoi meshes. Each square is a grid.

For $P_G = 0.2, 0.5, 0.8$, the geometry distributions are shown in Figure 11. In this paper, $P_G = 0.8$ is used. To generate offspring, a mesh is randomly selected according to the above selection probability first. In the selected mesh, an offspring is generated with a uniform probability. This is repeated until all offspring have been generated.

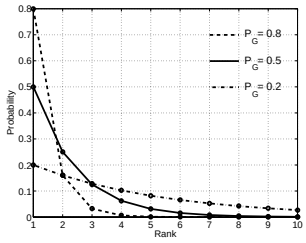


Fig. 11. Geometry Distribution with $P_G = 0.2, 0.5, 0.8$.

F. Database Maintenance

In VEDA, a fast ranking method, namely, non-dominated sorting proposed by Deb et al. [5] is used. Even with this method, an increase in the number of data results in expensive computational cost to evaluate the rank of all data in the databases. To avoid this increase, we take the following measure: Two databases are prepared in VEDA: one is for storing promising data namely *G-database*; the other is for non-promising data namely *B-database*. The number of data in the *G-database* is the number of individuals in a population, μ. This means that only the number of data in the *B-database* is increasing during an optimization. In the beginning, the *G-database* is empty. Thereafter, μ parents and μ offspring are stored in the *G-database*, i.e. the number of data is 2μ. The non-dominated sorting method is carried out for 2μ data. The best μ data can stay in the *G-database*. The worst μ data move to the *B-database*. In the *B-database*, all data are assigned the same rank which is the worst rank increased by one in the *G-database*, so that all data in the *B-database* have worse rank than the ones in the *G-database*.

IV. Simulation Results

A. Comparison of VEDA with NSGA-II

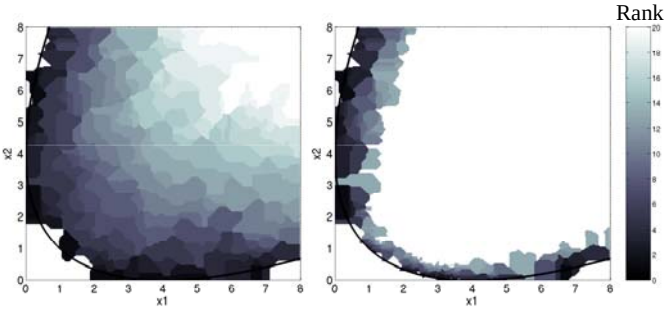
The proposed method is tested on SCH1 ($n = 2, 5$), FON2 ($n = 2, 5$) and OKA4 ($n = 2$). For a comparison, NSGA-II proposed in [5] is also tested. The used parameters are shown in Table I. In many real-world applications, only a limited number of fitness evaluations can be carried out. Thus, only 1000 fitness evaluations are allowed in the simulation. In Table I, the ε corresponds to the accuracy of a model. To reduce the computational cost for $n = 5$, the accuracy is reduced from $\varepsilon = 1.0$ to 0.5.

TABLE I

Parameters used here.

VEDA	
Parameter	Value
Number of Data	100
Population size	100
Maximum iterations	10
Parameter ε (See Section **??**)	1.0 for $n = 2$ 0.5 for $n = 5$
NSGA-II	
Population size	100
Maximum iterations	10
Coding	Gray coding
Crossover	One-point crossover
Crossover rate	0.9
Mutation (GA)	Bit flip
Mutation rate (GA)	0.01
Number of bits per a design parameter	20

In order to give an impression of the generated models, the models for OKA4 in the first generation and the fifth generation are shown in Figure 12. The color corresponds to the rank, i.e., the darker the color, the higher the rank. Clearly, the generated model has a higher probability near the Pareto front in the parameter space.



(a) Generation 1 (b) Generation 5

Fig. 12. Generated stochastic models for OKA4 in the parameter space. The Pareto front is $x_2 = x_1 \pm 4\sqrt{x_1} + 4$.

The results of SCH1, FON2 and OKA4 are shown in Figure 13. Since OKA4 is designed with a non-linear Pareto front in the parameter space [17], the solutions in the parameter space on OKA4 are also shown in Figure 14.

In the figures, all parents are plotted. Since many parent individuals in NSGA-II have not converged to the shown area within 10 generations, the number of individuals seems to be smaller than VEDA. On test functions SCH1 and FON2 with a dimension of 2, the difference in performance between NSGA-II and VEDA is minor, although VEDA seems to perform a bit better than NSGA-II. On test functions SCH1 and FON2 with a dimension of 5, the performance of both is not sufficient. However, VEDA shows a better performance than NSGA-II. On the OKA4, the difference is more obvious. VEDA achieves almost a complete Pareto front but NSGA-II does not. From these results, VEDA shows better performance than NSGA-II when only a limited number of fitness evaluations is allowed. This indicates that VEDA is very promising for solving real-world optimization problems where only a limited number of fitness evaluations can be afforded, see e.g. [16].

B. Comparison of VEDA with Other EDAs for MOO

We compare VEDA with two EDAs for MOO, the mixture-based iterated density estimation evolutionary algorithm (MIDEA) [22] and the Parzen-based EDA (PEDA) [3]. Since both of them have used the same test functions, we use these test functions for the comparison. Since MIDEA [22] did not show a sufficiently good quality on ZDT4 ($n = 10$), ZDT4 is not used. The maximum number of iterations is set as shown in Table II. In VEDA, the numbers of initial data and offspring are 100 and the value of ε is 1.0 for FON2 and KUR1 and 0.1 for DEB4.

TABLE II

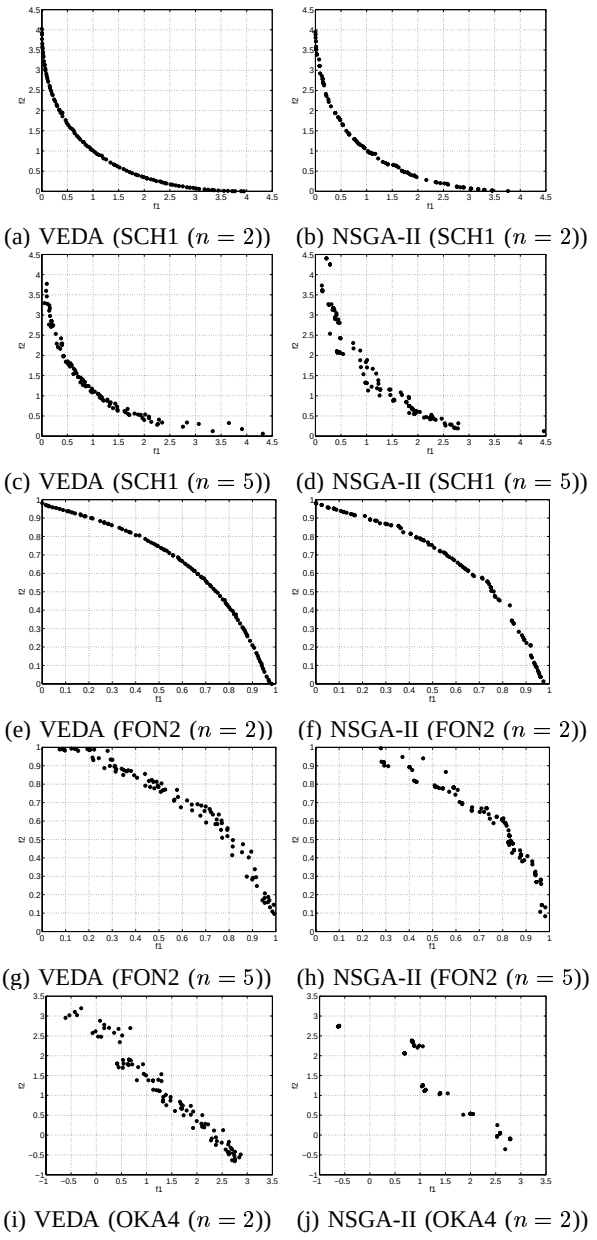
MAXIMUM NUMBER OF ITERATIONS. SEE [22] FOR MIDEA AND [3] FOR PEDA.

Method	FON2 ($n = 3$)	KUR1 ($n = 3$)	DEB4 ($n = 10$)
MIDEA [22]	3754	10762	8426
PEDA [3]	3100	11000	8300
VEDA	1000	5000	8300

The results of VEDA are shown in Figure 15. By comparison with the results reported in [3], [22], one can conclude that the performance of VEDA with fewer fitness evaluations is similar to MIDEA and PEDA on FON2 and KUR1. But VEDA is inferior to them on DEB4. Taking a closer look, it is found that VEDA generated many infeasible solutions in the DEB4 function. Since the current VEDA has no restrictions in perturbing the orthogonal direction, VEDA is very likely to generate offspring in the infeasible region where the variables are smaller than zero. By adding additional constraints in perturbation, the performance could be improved.

C. Modeling Pareto Front with VEDA

In the literature, most MOO methods output a set of solutions to represent the Pareto front. It is argued that this is not the best representation because the structure of the Pareto-optimal solutions is completely lost [9]. To address this problem, a new way to represent Pareto optimal solutions is suggested by modeling Pareto-optimal solutions using piecewise linear functions in the parameter space in [9]. Since the



(a) VEDA (SCH1 ($n = 2$)) (b) NSGA-II (SCH1 ($n = 2$))

(c) VEDA (SCH1 ($n = 5$)) (d) NSGA-II (SCH1 ($n = 5$))

(e) VEDA (FON2 ($n = 2$)) (f) NSGA-II (FON2 ($n = 2$))

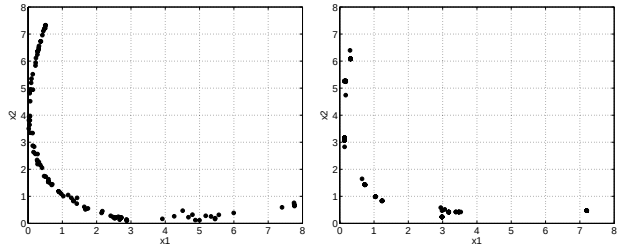
(g) VEDA (FON2 ($n = 5$)) (h) NSGA-II (FON2 ($n = 5$))

(i) VEDA (OKA4 ($n = 2$)) (j) NSGA-II (OKA4 ($n = 2$))

Fig. 13. The results in the fitness space by VEDA and NSGA-II. Only 1000 fitness evaluations are allowed.

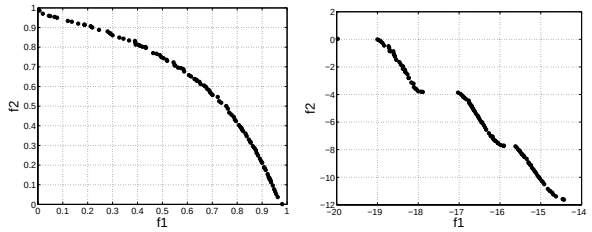
Pareto front in the parameter space of most MOO problems has a simple structure, it is possible to improve the accuracy by modeling the Pareto-optimal solutions.

One of the implicit characteristics of VEDA is the availability of a mathematical description instead of just a solution set. Since VEDA uses the PCA and the maximum and minimum values in each axis, VEDA can output a model for representing the Pareto-optimal solutions. As an example, the parameters of the mathematical model obtained by VEDA on SCH1 ($n = 2$) are given in Table III and its graphical description is shown in Figure 16. In the figure, the dotted line is the Pareto front in the parameter space. The principal axis shows the gradient of the solution set and the minimum and maximum values indicate

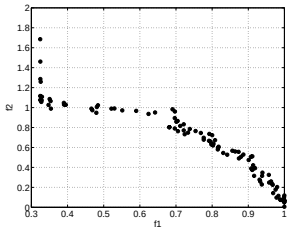


(i) VEDA (OKA4 ($n = 2$)) (j) NSGA-II (OKA4 ($n = 2$))

Fig. 14. The results in the parameter space on OKA4 by VEDA and NSGA-II. The Pareto front is $x_2 = x_1 \pm 4\sqrt{x_2} + 4$.



(a) FON2 ($n = 3$) (b) KUR1 ($n = 3$)



(c) DEB4 ($n = 10$)

Fig. 15. Comparison of VEDA with the literature [3], [22].

the range of the solution distribution. Since the true Pareto front of SCH1 in the parameter space is $x_2 = x_1,\ x_1 \in [0, 2]$, the VEDA seems to be successful in obtaining a mathematical description of the Pareto-optimal solutions.

TABLE III
MATHEMATICAL OUTPUT OF VEDA.

Principal Axis	$\{0.701764, 0.712410\}$
Orthogonal Axis	$\{0.712410, -0.701764\}$
Minimum Value in Principal Axis	-0.042017
Maximum Value in Principal Axis	2.841821
Minimum Value in Orthogonal Axis	-0.166802
Maximum Value in Orthogonal Axis	0.208333

V. SUMMARY

In this paper, a Voronoi-based estimation of distribution algorithm (VEDA) has been proposed for solving MOO problems. In VEDA, the offspring distribution in terms of ranking is directly used to generate new individuals. Information from not only the selected individuals but also from those that are not selected is included in the model. Using the concept of Voronoi mesh, a stochastic model is constructed and the promising solutions are generated according to the model. The

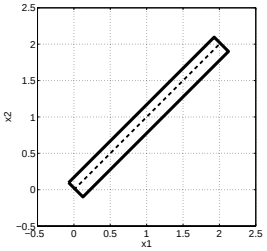


Fig. 16. Graphical output of VEDA. The solid rectangle is the output of VEDA. The dotted line is the true Pareto front in the parameter space.

performance of VEDA has been compared with the NSGA-II on several test functions. It is shown that the performance of VEDA is better than NSGA-II when a limited number of fitness evaluations is allowed. This implies that VEDA is very promising for solving real-world problems where fitness evaluations are very time-consuming.

Although VEDA shows good performance on several test functions, a drawback of VEDA as well as other EDAs is that the computational complexity increases rapidly with the increase of the dimensionality. In addition, proper constraints should be included in perturbing the orthogonal search direction. In the next step, we would also like to verify the performance of VEGA for real-world applications.

ACKNOWLEDGMENT

The authors would like to thank Prof. Dr. E. Körner and Mr. A. Richter for their supports and reviewers for their fruitful comments.

REFERENCES

[1] Bosman, P. A. N. and Thierens, D., An Algorithmic Framework for Density Estimation Based Evolutionary Algorithms, Technical Report UU-CS-1999-46, Utrecht University, 1999.
[2] Coello, C. A. C., Van Veldhuizen, D. A. and Lamont, G. B., *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, 2001.
[3] Costa, M. and Minisci, E., MOPED: A Multi-objective Parzen-based Estimation of Distribution Algorithm for Continuous Problems, In Fonseca C. M. et al., editors, *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization*, pages 282–294, 2003.
[4] Deb, K., *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, LTD., 2001.
[5] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6(2), pages 182–197, 2002.
[6] Deb, K., A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization, *Evolutionary Computation*, 10(4), pages 371–395, 2002.
[7] Fukuda, K., http://www.ifor.math.ethz.ch/staff/fukuda/polyfaq/node29.html.
[8] Igel, C. and Kreutz, M., Operator Adaptation in Evolutionary Computation and Its Application to Structure Optimization of Neural Networks, Neurocomputing, 55(1-2), pages 347–361, 2003.
[9] Jin, Y. and Sendhoff, B., Connectedness, Regularity and the Success of Local Search in Evolutionary Multi-Objective Optimization, In *Proceedings of Congress on Evolutionary Computation*, pages 1910–1917, 2003.
[10] Jolliffe, I. T., *Principal Component Analysis*, Springer Verlag, 2002.
[11] Kaufman, L. and Rousseeuw, P. J., *Finding Groups in Data - An Introduction to Cluster Analysis*, Wiley-Interscience, 1990.

[12] Larrañaga, P. and Lozano, J. A., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Publishers, 2002.
[13] Laumanns, M. and Ocenasek, J. , Bayesian Optimization Algorithms for Multi-objective Optimization In Merelo Guervós, J. J. el al., editors, *Proceedings of Parallel Problem Solving from Nature VII*, pages 298–307, 2002.
[14] Okabe, T., Jin, Y. and Sendhoff, B., On the Dynamics of Evolutionary Multi-Objective Optimisation, In Langdon, W. B. et al., editors, *Proceedings of Genetic and Evolutionary Computation Conference*, pages 878–885, 2002.
[15] Okabe, T., Jin, Y. and Sendhoff, B., Evolutionary Multi-Objective Optimisation with a Hybrid Representation, In *Proceedings of Congress on Evolutionary Computation*, pages 2262–2269, 2003.
[16] Okabe, T., Foli, K., Olhofer, M., Jin, Y. and Sendhoff, B., Comparative Studies on Micro Heat Exchanger Optimisation, In *Proceedings of Congress on Evolutionary Computation*, pages 647–654, 2003.
[17] Okabe, T., Evolutionary Multi-Objective Optimization: On the Distribution of Offspring in Parameter and Fitness Space, Dissertation in Bielefeld University, 2004, (Submitted).
[18] Olhofer, M., Erweiterung Evolutionärer Algorithmen zur Designoptimierung, Dissertation in Ruhr-Universität Bochum, 2002.
[19] Ono, I., Kita, H. and Kobayashi, S., A Robust Real-Coded Genetic Algorithm using Unimodal Normal Distribution Crossover Augmented by Uniform Crossover : Effects of Self-Adaptation of Crossover Probabilities, In Banzhaf, W. et al., editors, *Proceedings of Genetic and Evolutionary Computation Conference*, pages 496–503, 1999.
[20] Pelikan, M., Goldberg, D. E. and Lobo, F., A Survey of Optimization by Building and Using Probabilistic Models, Technical Report 99018, University of Illinois, 1999.
[21] Rechenberg, I., Evolutionsstrategie '94, Frommann-Holzboog, 1994.
[22] Thierens, D. and Bosmann, P. A. N., Multi-Objective Mixture-based Iterated Density Estimation Evolutionary Algorithms, In Spector, L. et al., editors, *Proceedings of Genetic and Evolutionary Computation Conference*, pages 663–670, 2001.
[23] Tsutsui, S., Pelikan, M. and Goldberg, D. E., Probabilistic Model-building Genetic Algorithms Using Marginal Histograms in Continuous Domain, In *Proceedings of Knowledge-based Intelligent Information Engineering Systems and Allied Technologies*, 1, pages 112–121, 2001.

APPENDIX

Test functions used in this paper are as follows:

- SCH1 test function

$$f_1 = \frac{1}{n}\sum_{i=1}^{n} x_i^2, \qquad f_2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - 2)^2$$
$$x_i \in [-4, 4], \quad (i = 1, 2, ..., n). \tag{1}$$

Pareto front in the fitness space

$$f_2 = f_1 - 4\sqrt{f_1} + 4, \quad 0 \le f_1 \le 4. \tag{2}$$

Pareto front in the parameter space

$$x_1 = x_2 = ... = x_n, \quad 0 \le x_1 \le 2. \tag{3}$$

- FON2 test function

$$f_1 = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right)$$
$$f_2 = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right),$$
$$-4 \le x_i \le 4, \quad (i = 1, 2, ..., n). \tag{4}$$

Pareto front in the fitness space

$$f_2 = 1 - \frac{1 - f_1}{e^4}\exp\left\{4\sqrt{-\log(1 - f_1)}\right\}$$
$$0 \le f_1 \le 1 - e^{-4}. \tag{5}$$

Pareto front in the parameter space

$$x_1 = x_2 = ... = x_n, \quad -\frac{1}{\sqrt{n}} \le x_1 \le \frac{1}{\sqrt{n}}. \tag{6}$$

- OKA4 test function

$$\begin{aligned} f_1 &= 2 - \frac{1}{4}(x_1 - x_2 + 4) \\ &+ \frac{1}{4}\sqrt{-x_1^2 - x_2^2 - 16 + 2x_1x_2 + 8x_1 + 8x_2} \\ f_2 &= \frac{1}{4}(x_1 - x_2 + 4) \\ &+ \frac{1}{4}\sqrt{-x_1^2 - x_2^2 - 16 + 2x_1x_2 + 8x_1 + 8x_2} \end{aligned}$$

subject to

$$x_1 - 4\sqrt{x_1} + 4 \le x_2 \le x_1 + 4\sqrt{x_1} + 4$$
$$0 \le x_1, x_2 \le 8. \tag{7}$$

Pareto front in the fitness space

$$f_2 = 2 - f_1, \quad 2 - 2\sqrt{2} \le f_1, f_2 \le 2\sqrt{2}. \tag{8}$$

Pareto front in the parameter space

$$x_2 = x_1 \pm 4\sqrt{x_1} + 4, \quad 0 \le x_1, x_2 \le 8. \tag{9}$$

- KUR1 test function

$$f_1 = \sum_{i=1}^{n-1}\left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right)$$
$$f_2 = \sum_{i=1}^{n}\left(|x_i|^{0.8} + 5\sin(x_i)^3\right)$$
$$x_i \in [-5, 5], \quad (i = 1, 2, ..., n). \tag{10}$$

See [2], [4] for the Pareto front in the fitness and parameter space.

- DEB4 test function

$$f_1 = 1 - \exp(-4x_1)\sin^4(5\pi x_1)$$
$$f_2 = g(x)h(f_1(x), g(x))$$
$$h(f_1(x), g(x)) = \begin{cases} 1 - (f_1/(\beta g))^\alpha & \text{if } f_1 \le \beta g \\ 0 & \text{Otherwise} \end{cases}$$
$$g(x) = 1 + 10x_2, \quad \alpha = 4.0, \quad \beta = 1.0$$
$$0.0 \le x_i \le 1.0. \tag{11}$$

Pareto front in the fitness space

$$f_2 = 1 - f_1^4, \quad 0.3242^* \le f_1 \le 1. \tag{12}$$

$*$: Obtained empirically.

Pareto front in the parameter space

$$x_2 = 0, \quad 0 \le x_1 \le 1. \tag{13}$$