Estimation Distribution Algorithms Based on Extreme Elitism and Their Application in Engineering Optimization Problems

by

Shujun Gao

B.Sc., North University of China, 2008

M.Sc., Tianjin University, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2017

Abstract

This dissertation modifies several estimation distribution algorithms (EDAs) and implements them in engineering optimization problems. The EDAs are population-based evolutionary algorithms, which employ extreme elitism selection. The main work of the present study is outlined below.

First, an approach of extreme elitism selection is developed for EDAs. This selection highlights the effect of a few top solutions and advances EDAs to form a primary evolutionary direction. Simultaneously, this selection can also maintain population diversity to make EDAs avoid premature convergence. EDAs with the new selection approach are tested using a set of benchmark low-dimensional and high-dimensional optimization problems. The experimental results show that the EDA based on univariate marginal Gaussian distribution (UMGD) with extreme elitism selection can outperform some other classical evolution algorithms for most problems.

Second, the EDA based on UMGD with extreme elitism is implemented for solving the inverse displacement problem (IDP) of a robotic manipulator. This EDA is compared with the EDAs with other selection methods in solving the IDP of a 4-degree-of-freedom (DOF) robotic arm. Next the algorithm is integrated with differential mutation to solve the IDP of a 7-DOF robotic arm. After that, the proposed algorithm is used to search for satisfactory solutions as a continuous curve. The simulation results show this algorithm can reach real time speeds, in practical applications.

Third, EDAs based on five different Gaussian distributions are proposed to solve optimization problems with various types of constraints like equality, inequality, linear, nonlinear, continuous or discontinuous. It is found the EDA based on a single multivariate Gaussian distribution with extreme elitism selection can outperform other EDAs. Besides, this EDA has good performance for four engineering design problems.

Fourth, EDA is combined with differential mutation to solve multi-objective optimization problems (MOPs). The hybrid algorithm seeks to find the Pareto optimal front for MOPs. EDAs guide the search direction in the evolution while differential mutation keeps a diversified population. A new sampling method that uses more Gaussian models to generate offspring is specially designed for the EDAs for MOPs. In light of no-free-lunch theorem, different probabilistic models and programing codes are adopted for different MOPs.

Lay Summary

This thesis aims to improve the performance of estimation distribution algorithm for solving various optimization problems. The key contributions are summarized as follows:

1. An extreme elitism selection method is developed for EDA. The EDA with this selection can outperform the EDAs with truncation, tournament and proportional selection and some other evolution algorithms for a set of benchmark problems.

2. EDA is first extended to solve manipulator inverse displacement problem. It does not depend on the configuration of manipulators or require an initial guess as in some numerical methods.

3. EDA is fist systematically used to handle the optimization problems with various types of constraints, which can further extend the practical implementation of EDA.

4. EDA combines differential mutations to solve multi-objective optimization problems (MOPs). This new algorithm does not require prior knowledge or to transform MOPs into single objective problems and can provide more alternative choices for decision makers.

Preface

The entire work presented in this dissertation was conducted at the Industrial Automation Laboratory of University of British Columbia (Vancouver campus) under the supervision and guidance of Prof. Clarence W. de Silva. I was responsible for such work of this research as literature survey, algorithm development, implementation and experimentation. Dr. de Silva proposed and supervised the overall research project, acquired funding and resources for the project, suggested the topic of the thesis, suggested concepts and methodologies in addressing problems in the topic, provided research facilities in his Industrial Automation Laboratory, and revised the thesis presentation.

Parts of Chapter 2 have been published as:

Shujun Gao, Clarence W. de Silva. A modified estimation distribution algorithm based on extreme elitism. *Biosystems*, vol.150, pp.149-166, 2016.

Parts of Chapter 3 have been published as:

Shujun Gao, Clarence W. de Silva. A univariate marginal distribution algorithm based on extreme elitism and its application to the robotic inverse displacement problem. *Genetic Programming and Evolvable Machines*, vol.18, pp.283-312, 2017.

The following manuscript based on Chapter 4 has been submitted to a journal and is under review:

Shujun Gao, Clarence W. de Silva. Estimation Distribution Algorithms with Modified Extreme Elitism Selection for Constrained Optimization Problems and Its Application in Mechanical Design.

The following manuscript, which is based on Chapter 5 has been accepted by 2017 *IEEE International Conference on Systems, Man, and Cybernetics*.

Shujun Gao, Clarence W. de Silva. Estimation Distribution Algorithms Combining Differential Mutation for Multi-objective Optimization Problems.

Table of Contents

Abstract .... **ii**

Lay Summary .... **iv**

Preface .... **v**

Table of Contents .... **vii**

List of Tables .... **xi**

List of Figures .... **xiii**

List of Abbreviations .... **xviii**

Acknowledgements .... **xix**

Chapter 1: Introduction **1**

1.1 Motivation 1

1.2 Problem formulation 3

1.3 Related work 6

1.3.1 Estimation distribution algorithms 6

1.3.2 Manipulator inverse displacement problem 9

1.3.3 EDAs for constrained optimization problems 10

1.3.4 Multi-objective optimization problems (MOPs) 11

1.4 Contributions and organization of the dissertation 13

Chapter 2: Estimation Distribution Algorithms with Extreme Elitism Selection **16**

2.1 Introduction 16

2.2 EDAs using an extreme elitism selection 17

2.2.1 The selection methods in EDAs 17

2.2.2 The role of top solutions in evolution algorithms 20

2.2.3 EDAs scheme with extreme elitism 21

2.2.4 Percentage of top solutions in the parent population 27

2.2.5 The combination of top solutions in the elite corps 30

2.3 Experimental evaluation of EE-EDA 32

2.3.1 Studies of 30-*D* and low-dimensional problems 36

2.3.2 Special cases studies 41

2.3.3 The test results of 100-*D* and 200-*D* problems 42

2.4 The experimental results of other algorithms for high-dimensional problems 47

2.5 Summary 55

Chapter 3: Application of EDAs in the Robotic Inverse Displacement Problem 57

3.1 Introduction 57

3.2 IDP of a 4-DOF Barrett WAM Arm 58

3.2.1 Optimization IDP model of a 4-DOF Barrett WAM Arm 59

3.2.2 EE-EDA program for the problem 62

3.2.3 Comparison of EDA with different selection for IDP of 4 DOF arm 64

3.3 EE-EDA Combined with DM to solve IDP of the 7-DOF Barrett WAM arm 69

3.3.1 EE-EDA for IDP of 7-DOF arm 70

3.3.2 EE-EDA Combined with DM to solve IDP of 7-DOF arm 71

3.3.2.1 Differential Mutation strategies 71

3.3.2.2 Program of EE-EDA/DM for the IDP of 7-DOF 72

3.3.3 Comparison of EE-EDA/DE, ES, ELPSO and IDEA for IDP of 7-DOF arm 78

3.4 EE-EDA/DM for solving IDP of continuous trajectory 81

3.5 Summary 86

Chapter 4: Estimation Distribution Algorithms for Constrained Optimization Problems 87

4.1 Introduction 87

4.2 EDAs based on Gaussian probabilistic models 89

4.3 Solution of COPs using EDAs with extreme elitism selection 91

4.4 Experimental results and analysis 93

4.4.1 The performance evaluation of EDAs for 13 benchmark COPs 93

4.4.1.1 Comparison of EDAs based on MVGD with different selection methods ..101

4.4.1.2 MVGD with a smaller parent population and a constant tolerance 102

4.4.1.3 Comparison of EDAs based on different Gaussian distributions 103

4.4.1.4 Comparison of EDAs and some state-of-the-art algorithms for COPs 108

4.4.2 The application of EDAs in 4 mechanical engineering design problems 110

4.4.2.1 Welded beam design 111

4.4.2.2 Spring design 113

4.4.2.3 Speed reducer design 113

4.4.2.4 Three-bar truss design 115

4.5 Summary 115

Chapter 5: Estimation Distribution Algorithms for Multi-objective Optimization Problems 116

5.1 Introduction 116

5.2 The algorithm of combining EDA with DM for MOPs 120

5.2.1 The flowchart of EDA-DM and domination sorting 120

5.2.2 A modified extreme elitism selection for MOPs ... 123

5.2.3 A new sampling offspring method for MOPs ... 124

5.2.4 The performance metric of the algorithms for MOPs ... 125

5.3 Experiments and analysis ... 127

5.3.1 MOPs with a few variables ... 127

5.3.2 MOPs with more variables ... 130

5.3.3 MOPs with inequality constraints ... 135

5.4 Summary ... 136

Chapter 6: Conclusions and Future Work ... 138

6.1 Conclusions ... 138

6.2 Possible future work ... 140

Bibliography ... 142

Appendix ... 155

List of Tables

Table 2.1 ppTest results of EE-EDA with different percentage of leading best solutions in parent population. 29

Table 2.2 Test results of EE-EDA with different combinations in the elite corps 31

Table 2.3 High-dimensional test problems 33

Table 2.4 Low-dimensional test problems 34

Table 2.5 Results of EDA with different selection methods for 30-*D* problems 37

Table 2.6 Success rate of low-dimensional problems (TW indicates totally wining) 40

Table 2.7 Results of EDA with different selection methods for 100-*D* problems 43

Table 2.8 Results of EDA with different selection models for 200-*D* problems 44

Table 2.9 Results of other optimization strategies for 30-*D* problems 47

Table 2.10 Results of other optimization strategies for 100-*D* problemss 50

Table 2.11 Results of other optimization strategies for 200-*D* problems 51

Table 3.1 D-H parameters and joints bound of the 4-DOF Barrett WAM Arm 60

Table 3.2 Distribution of the stop iteration generations of 100 tests of each algorithm 67

Table 3.3 D-H parameters and joints bound of the 7-DOF Barrett WAM Arm 69

Table 3.4 Differential mutation strategies 71

Table 3.5 Distribution of stop generations of EE-EDA with *PS*=2500 in Section **3.3.1** 72

Table 3.6 Orientation and position matrix of 10 random desired points 79

Table 3.7 Distribution of the stop iteration generation of 100 operations of each algorithm 79

Table 3.8 Distribution of stop iteration generation of the operations for the 61 desired points 83

Table 4.1 Thirteen benchmark COPs 94

Table 4.2 The experimental results of thirteen benchmark problems... 98

Table 4.3 The test results of some state-of-the-art algorithms for these benchmark problems.. 108

Table 4.4 The test results of 4 mechanical engineering design problems 110

Table 5.1 Some benchmark MOPs with a few variables.. 127

Table 5.2 The mean and variance of the distance metric Υ 128

Table 5.3 The mean and variance of the diversity metric Δ 128

Table 5.4 Some benchmark MOPs with more variables... 130

Table 5.5 Some benchmark MOPs with inequality constraints.. 136

List of Figures

Figure 1.1 The flowchart of GAs and EDAs 4

Figure 1.2 Various types of probabilistic models 7

Figure 2.1 The procedure of extreme elitism selection........ 24

Figure 2.2 The initial population 25

Figure 2.3 Population in the 5th generation........ 25

Figure 2.4 Population in the 10th generation........ 25

Figure 2.5 Population in the 20th generation........ 26

Figure 2.6 Convergence of TR-EDA and EE-EDA for function (2.8) 27

Figure 2.7 The distribution of minima of Griewanks function........ 28

Figure 2.8 The best fitness convergence curves of some 30-*D* functions (1)........ 38

Figure 2.9 The best fitness convergence curves of some 100-*D* functions (1)........ 46

Figure 2.10 The best fitness convergence curves of some 200-*D* functions (1)........ 46

Figure 2.11 The best fitness convergence curves of some 30-*D* functions (2)........ 49

Figure 2.12 The best fitness convergence curves of some 100-*D* function (2) 53

Figure 2.13 The best fitness convergence curves of some 200-*D* functions (2)........ 54

Figure 3.1 The kinematic structure of the 4-DOF Barrett WAM Arm........ 61

Figure 3.2 The flowchart of EE-EDA for solving IDP of the 4-DOF arm 62

Figure 3.3 Convergence trend of EDA with different selection 65

Figure 3.4 Distribution of the final error of 100 tests of the EDA with different selection 66

Figure 3.5 Distribution of computing time of 100 tests of the EDA with different selection 68

Figure 3.6 The kinematic structure of the 7-DOF Barrett WAM Arm........ 69

Figure 3.7 Distribution of the final error of 100 tests of two different EE-EDAs 71

Figure 3.8 The flowchart of EE-EDA/DM for IDP of 7-DOF arm 73

Figure 3.9 Differential mutation and Gaussian search 75

Figure 3.10 Convergence trend of the error of EE-EDA/DM in three different situations 77

Figure 3.11 Distribution of final error of 100 tests of EE-EDA and EE-EDA/DM 78

Figure 3.12 Distribution of final error of 100 tests of EE-EDA/DM, ES, ELPSO and IDEA 80

Figure 3.13 Distribution of the computing time of EE-EDA/DM, ES, ELPSO and IDEA 81

Figure 3.14 Desired continuous trajectory 82

Figure 3.15 The IDP solution of each joint for the desired continuous trajectory 84

Figure 4.1 An example of Gaussian network 91

Figure 4.2 Flowchart of the EDAs that solve COPs with extreme elitism selection 92

Figure 4.3 Gaussian search and feasible region 93

Figure 4.4 The total win times of each algorithm in solving 13 COPs 100

Figure 4.5 The distribution of three types of Gaussian probabilistic models 104

Figure 4.6 The best fitness convergence curves of each algorithm when solving P1 106

Figure 4.7 The best fitness convergence curves of each algorithm when solving P5 107

Figure 4.8 The best fitness convergence curves of each algorithm when solving P12 108

Figure 4.9 (a) A welded beam and (b) A compression spring 112

Figure 4.10 The best fitness convergence curves of each algorithm when solving EG3 114

Figure 5.1 The definition of non-domination 117

Figure 5.2 Pareto front 118

Figure 5.3 Flowchart of EDA-DM for MOPs 121

Figure 5.4 Domination rank 122

Figure 5.5 Sample offspring around leading best non-dominated solutions 124

Figure 5.6 Distance metric 125

Figure 5.7 Diversity metric 126

Figure 5.8 Obtained Pareto front by EDA-DM for SCH and FON 129

Figure 5.9 Obtained Pareto front by EDA-DM for POL and KUR 129

Figure 5.10 Binary probability vector 131

Figure 5.11 Obtained Pareto front by EDA-DM for ZDT1 and ZDT2 132

Figure 5.12 Obtained Pareto front by EDA-DM for ZDT3 and ZDT4 133

Figure 5.13 Obtained Pareto front by EDA-DM for ZDT6 and CONSTR 134

Figure 5.14 Obtained Pareto front by EDA-DM for SRN and TNK 136

List of Symbols

D	Dimension of Variable Space
F	Scaling Factor
G	Maximum Iteration Generation
$g_j(\boldsymbol{X})$	Inequality Constraint
$h_j(\boldsymbol{X})$	Equality Constraint
M	Parent Population Size
m	Number of Objectives
n	Number of Constraints
$\boldsymbol{o}^{\mathbf{0}}_{\mathbf{Tool}}$	Orientation Matrix
$\boldsymbol{o_c}$	Current Orientation
$\boldsymbol{o_d}$	Desired Orientation
$\boldsymbol{p}^{\mathbf{0}}_{\mathbf{Tool}}$	Position Matrix
$\boldsymbol{p_d}$	Desired Position
$\boldsymbol{p_c}$	Current Position
PS	Population Size
$\boldsymbol{T}^{\mathbf{0}}_{\mathbf{Tool}}$	Homogeneous Matrix
$\boldsymbol{X}$	Variable Vector
x_{iL}	Lower Boundary of Variable
x_{iu}	Upper Boundary of Variable
α	Selection Proportion
Δo	Orientation Error
Δp	Position Error

γ	Learning Rate
μ_i	Mean
$\boldsymbol{\mu}$	Mean Vector
Ω	Feasible Region
σ_i^2	Variance
Σ	Covariance Matrix
θ	Joint Movement

List of Abbreviations

ABCA	Artificial Bee Colony Algorithm
ACO	Ant Colony Optimization
COPs	Constrained Optimization Problems
DOF	Degree-of-Freedom
DE	Differential Evolution
DM	Differential Mutation
EDAs	Estimation Distribution Algorithms
EE-EDA	Extreme Elitism Estimation Distribution Algorithm
ES	Evolution Strategies
GAs	Genetic Algorithms
IDP	Inverse Displacement Problem
JPDF	Joint Probability Density Function
MIXMVGD	Mixture of Multivariate Gaussian Distributions
MIXUMGD	Mixture of Univariate Marginal Gaussian Distribution
MOPs	Multi-objective Optimization Problems
MVGD	Multivariate Gaussian Distribution
PR-EDA	Proportional Estimation Distribution Algorithm
PSO	Particle Swarm Optimization
TR-EDA	Truncation Estimation Distribution Algorithm
TO-EDA	Tournament Estimation Distribution Algorithm
UMGD	Univariate Marginal Gaussian Distribution

Acknowledgements

I wish to express my sincere gratitude to my supervisor, Dr. Clarence W. de Silva for his help and supervision. He provided me a precious opportunity to study in the beautiful University of British Columbia and in his laboratory (Industrial Automation Lab). This experience will be always remembered as a great treasure in my life. He patiently taught me how to prepare the general knowledge and proposal exam. He carefully revised my publications and gave important suggestions. I greatly appreciate his valuable support, advice and guidance in the five years' PhD research and editing this thesis. Besides, I am grateful to receive funds through research grants like those from Natural Sciences and Engineering Research Council (NSERC) of Canada, the Canada Foundation for Innovation (CFI), the British Columbia Knowledge Development Fund (BCKDF) and the Senior Canada Research Chair in Mechatronics and Industrial Automation held by Dr. Clarence W. de Silva.

Next I wish to thank my colleagues in the Industrial Automation Laboratory (IAL), Dr. Roland Haoxiang Lang, Dr. Edward Yanjun Wang, Dr. Yunfei Zhang, Dr. Muhammad Tufail Khan, Ms. Pegah Maghsoud, Ms. Yu Du, Ms. Lili Meng, Mr. Hani Balkhair, Mr. Shan Xiao, Mr. Min Xia, Mr. Teng Li, Mr. Zhu Cheng, Mr. Fan Yang, Mr. Tongxin Shu, Mr. Jiahong Chen, Mr. Sheikh Tanvir, Mr. Bilal Riaz, Mr. Lucas Falch and Ms. Swapna Pemasiri. Thank you and I had a wonderful time with you in this lab.

Finally, I greatly appreciate my parents and parents-in-law, especially my mother and mother-in-law. You greatly facilitated my education by taking care of my two little children. Without your valuable help, I could not have focused on my studies. Thank you Mother. The

Special thanks go to my wife, Jiachao Yu. She persuaded me to study abroad in the beginning and always encouraged me to continue on the studies when I met some difficulties.

Chapter 1: Introduction

1.1 Motivation

Optimization is the process of finding the best solution with respect to some performance measures. Optimization problems arise naturally in many disciplines, such as engineering, economics, mathematics, commerce, administration, social science, health sciences and so on. In the field of engineering, there exist numerous optimization problems in the disciplines of mechanical, electrical, computer, chemical, biological, civil, and so on. The typical areas of implementation are design of devices, tools, circuits and control systems; modeling; design of structures and buildings; production scheduling; inventory control, accounting, budgeting, and so on. [1]. It is significant to research how to obtain relevant optimal solutions efficiently, for optimization problems. Many analytical optimization problems can be described by the following mathematical model:

$$\begin{aligned}
\text{minimize} \quad & f(\boldsymbol{X}), \quad \boldsymbol{X} = (x_1, \cdots, x_D) \in R^D \\
\text{subject to} \quad & g_j(\boldsymbol{X}) \le 0, \qquad j = 1, \cdots, q \\
& h_j(\boldsymbol{X}) = 0, \qquad j = q+1, \cdots, n \\
& x_{iL} \le x_i \le x_{iU} \quad i = 1, \cdots, D
\end{aligned} \tag{1.1}$$

where, $f(\boldsymbol{X})$ is the objective function. If there are many different performance functions $f(\boldsymbol{X})$, the problem is a multi-objective optimization problem. Also, $\boldsymbol{X}$ is a vector containing D variables; $g_j(\boldsymbol{X})$ and $h_j(\boldsymbol{X})$ are inequality and equality constraints, respectively; n is the total number of constraints, q is the number of inequality constraints, $(n\text{-}q)$ is the number of equality constraints, and x_{iL} and x_{iU} are the lower and upper bound of the variable x_i, respectively. Then $\boldsymbol{X} \in \Omega \in S$, Ω

is the feasible region and S is a D-dimensional rectangular space defined by x_{iL} and x_{iU} in the R^D space. Some optimization problems do not have constraints and there is only an objective function. Conventional approaches to optimization problems are analytical methods, numerical methods like Newton's method, gradient descent method, conjugate gradient method, Quasi-Newton methods, Interior point methods for the constrained optimization problems, and so on. These techniques generally evaluate the gradients or Hessians. For some problems, these approaches can search for the optimal solutions efficiently and converge rapidly. However, some optimization problems are high-dimensional and have many local minima; some problems could be discontinuous or non-differentiable. For these problems, the traditional approaches cannot provide good performance, or even handle them. Some other traditional methods like the simplex algorithm only can solve special problems in which the objective or constrained functions should be linear. In the past decades, motivated by the natural world, researchers developed evolutionary techniques to handle optimization problems. The evolution algorithms generate an initial population (a group of initial solutions or parent solutions) according to some criteria (often randomly), and then select some excellent solutions as the "offspring" population according to some fitness functions. After that, the operation of reproduction, mutation or recombination is done on these individuals (who become new parents) to produce the offspring for the next generation. Then the selection and reproduction processes repeat for some generations until termination when a fitness function (performance function) attains a suitable value (or when a termination condition is reached).

These evolution techniques like genetic algorithms (GAs), genetic programming (GP), ant colony optimization (ACO), particle swarm optimization (PSO), artificial bee colony algorithm (ABCA), differential evolution (DE), evolution strategies (ES), and so on, are population-based and can search the optimal solution by several tracks in parallel. Different tracks can interact and share useful information with each other during the evolutionary process. In addition, they do not require the analytical information of gradient or continuity. They can optimize various kinds of problems, even qualitative ones. In this dissertation, the evolution algorithms called estimation distribution algorithms (EDAs) based on Gaussian distribution are proposed to solve optimization problems. The present study focuses on improving the performance of the EDAs for solving various types of optimization problems. The application of the EDAs to handle some engineering optimization problems, particularly in the mechanical domain, is presented as illustrative examples and case studies.

1.2 Problem formulation

EDAs originated from GAs. Fig.1.1 shows the procedural flowchart of GAs and EDAs. The difference between GAs and EDAs is that GAs use the operations crossover and mutation to generate the offspring, while EDAs adopt a probabilistic model to sample the offspring. It is known that some evolutionary programs like GAs, ACO, PSO, DE, ES, and so on generally suffer from the "curse of dimensionality," which implies that their performance rapidly deteriorates as the problem dimension increases [2]. Also, the algorithms may be trapped in some local minima when the optimization problems have many variables and many optima. Similar to the evolution algorithms, EDAs also face this difficulty. Many existing EDAs have only used the benchmark optimization problems with the dimension less than 100 to test their performance [3-9]. In the present research, the modified EDA based on Gaussian distribution will be used to handle optimization problems with 100 and 200 variables.

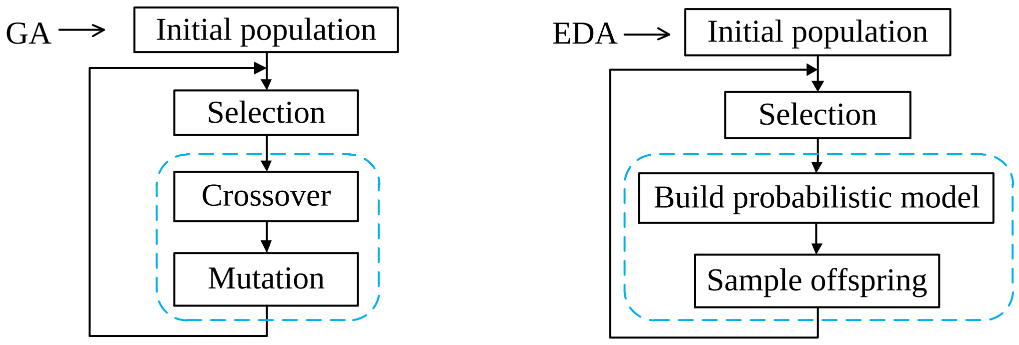


Figure 1.1 The flowchart of GAs and EDAs.

Another problem concerns the computational cost. Some EDAs have adopted complicated probabilistic models and procedures, which can improve the performance to some extent. However, these complicated probabilistic models and procedures usually lead to a high memory complexity and expensive computational cost, especially when solving high-dimensional problems. Gao and Culberson proved that the space complexity of factorized distribution algorithm (FDA) [10] and Bayesian optimization algorithm (BOA) [11]—two typical estimation distribution algorithms—was exponential in the problem size even if the optimization problem had a very sparse interaction structure [12]. Hence, how to find the suitable probabilistic models for the optimization problems is a crucial issue.

Besides, its practical application is also important for EDAs. If an algorithm cannot handle real-world optimization problems, it becomes meaningless. The present research extends the implementation of EDAs to solve the inverse displacement problem (IDP) of robotic manipulators. The current two main solution methods for this problem are algebraic and numerical. The algebraic solutions usually depend on the configuration of the robot and the existence of a closed-form solution [13-15]. Numerical methods are specific and usually need initial guesses, which may lead to extensive computational effort or instability [16-18]. The EDA that is proposed in the present work can be applied to robotic arms with different configurations, especially the redundant robots, which have more than 6 degrees of freedom operating in a 3-dimensional space. In addition, the method does not need to set an initial guess or step size. However, the problem of EDAs is the high computational cost since they need to deal with a group of solutions simultaneously. So the present study seeks to decrease the computational cost and make the proposed EDA achieve fast convergence, facilitating real time application.

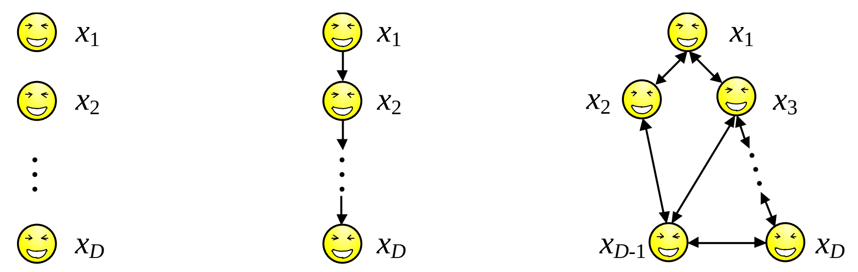
Some optimization problems only have the objective function, and some have constraints in the form of variable boundary. However, in disciplines of science and engineering, many optimization problems contain not only the constraints on variable boundary, but also some other types of constraints like equality, inequality, linear, nonlinear, continuous, or discontinuous. Various constraints increase the complexity of searching for the optimal solutions. A constrained optimization problem (COP) requires that the optimal solution falls within a feasible region, which may consist of a single bounded region or a set of disjoint regions. To date, most developed EDAs are applied on optimization problems that only have constraints of the variable boundary [3-11]. A few published papers have focused on the EDAs that can handle COPs with various types of constraints [19-20]. These EDAs have experimented only with three optimization problems and the test problems just have inequality constraints. Hence, the associated experimental results and conclusion are not representative of a general class of problems. Some researchers have adopted the penalty function method to handle constraints for EDAs [21]. They have used a penalty parameter of the constant value 10000 for all problems. However, it is known that the appropriate penalty parameter values are problem-dependent. So it is necessary to study how to reasonably handle various types of constraints for EDAs. Besides, the technique should be generalized.

In addition, it is known that some real-world optimization problems have multiple objectives. Generally, it is difficult to solve multi-objective optimization problems (MOPs), since the sub-objectives may have conflicts with each other and each sub-objective may have largely different types and magnitudes. The traditional methodologies, such as the ϵ-constraint method, weighted-sum approach, goal programming, and so on, generally transform the multi-objective problem into a single objective problem. These methods have low computational cost and complexity, but usually they require prior knowledge and experience. In the present research, EDAs are proposed to solve MOPs and they seek to obtain a set of uniformly distributed Pareto optimal solutions. EDAs do not require transformation or prior knowledge, and they can provide more choices for the decision makers. However, EDAs may trap in local optima as the population diversity decreases in the evolution. So, how to make EDAs generate a group of non-dominated solutions that can evenly spread in the Pareto front should be investigated.

1.3 Related work

1.3.1 Estimation distribution algorithms

Estimation distribution algorithms (EDAs) generally select some promising solutions from the current generation to constitute a parent population. Based on the parent population, a probabilistic model is built and then the offspring are sampled from this probabilistic model [9, 22-23]. In light of the probabilistic model, three main types of EDAs can be identified (seen in Fig. 1.2).



(a) univariate (b) bivariate (c) multivariate

Figure 1.2 Various types of probabilistic models.

(a) Univariate models, which assume that the problem variables are independent, such as the population-based incremental learning algorithm (PBIL) [24], univariate marginal distribution algorithm (UMDA) [22], and the compact genetic algorithm (cGA) [25]; (b) bivariate models like mutual information maximization and input clustering (MIMIC) [26] and bivariate marginal distribution algorithm [27]. These algorithms assume that there is some dependence between variable pairs; (c) multivariate models, which assume interactions between more problem variables, such as factorized distribution algorithm (FDA) [10], and Bayesian optimization algorithm (BOA) [11]. The early EDAs were used in the discrete domain. Since 2000, EDAs in continuous domain have been developed, such as UMDA$_C$, MIMIC$_C$, EGNA*ee*, EGNA$_{BGe}$ [28], EGNA$_{BIC}$, EMNA$_{global}$, EMNA$_a$, EMNA$_i$ [29], Iterated Density Estimation Evolutionary Algorithm (IDEA) [30], real-coded Bayesian optimization algorithm (rBOA) [31], and so on.

EDAs have the weakness of premature convergence as the population diversity decreases in the evolution. Some techniques have been introduced to solve this problem. Handa [32] presented a bitwise mutation operator that took account into the probabilistic model to enrich the population diversity. The experimental results indicated that the mutation improved the search ability of EDAs. Dong and Yao [7] adopted a niching method and a recombination operator into the population of EDAs. The resulting NichingEDA could make use of the diversity to solve some difficult problems with wide basins, flat plateaus, and deep valleys. Chen et al. [33] hybridized EDAs with other meta-heuristics and adjusted the procedures of sampling the offspring in accordance with the iteration generation to gradually increase the population diversity. Some other methods were also introduced to improve EDAs, like tuning the eigenvalues of the covariance matrix of the multivariate Gaussian models [8] and regularization [34]. Besides, EDAs have been integrated with other optimization techniques like differential evolution (DE) algorithm [6], particle swarm optimization (PSO) [35], and artificial immune system (AIS) [36] to obtain a better optimization strategy than the original one.

Theoretical research of EDAs also has been carried out. Lima et al. [37] investigated the relationship between the probabilistic models learned by BOA and the underlying problem structure. They demonstrated that Bayesian–Dirichlet scoring metric can generate more accurate models than the Bayesian information criterion metric. Besides, most of spurious dependencies are learned at the end of the network construction. Zhang and Mühlenbein [38] proved that EDAs with truncation, proportional and tournament selection can converge if the distribution of the offspring exactly matches the distribution of the parent set. Chen et al. analyzed the computation time of EDAs in relation to the problem size [39]. Grahl et al. [40] investigated the convergence behavior of the EDA based on univariate marginal Gaussian distribution in continuous domain with truncation selection for monotonous fitness functions. They found that the distance this algorithm travels across the search space is bounded and solely relies on α (the selection pressure). If an inappropriate α was adopted, the algorithm could not even explore the entire search space. This is a limitation of this EDA with truncation selection. Rastegar analyzed the probability of convergence to the optimal solution of cGA and PBIL, and obtained a sufficient condition for the convergence of these two algorithms [41]. Muelas et al. studied the effect of the information exchanged between different island-based models of EDAs in the evolution [42]. Besides, EDAs have been introduced to some practical applications like production scheduling [43], semiconductor testing [44], establishing a sequence detector [45], solving the dynamic economic dispatch problem in the power systems [46], image segmentation [47], and so on.

1.3.2 Manipulator inverse displacement problem

The inverse displacement problem (IDP, also known as the inverse kinematics problem) of a robotic manipulator involves determining the values of the movements of the robot joints corresponding to a specific orientation and position of the end effector. This is particularly important in trajectory planning and control of a robot arm. This problem is highly nonlinear and has multiplicity of solutions in general, especially for redundant robots, which have more than 6-degree of freedom (DOF) in a three-dimensional workspace. In the past decades, researchers have attempted to implement different methods to find the solution for the IDP accurately and efficiently. These methods can be classified into two main types: algebraic and numerical. The algebraic methods try to obtain the polynomial equations about the joint variables, and then they substitute the quantitative values, which come from the desired orientation and position matrix of the end effector into these equations to compute the joint values [13-15]. If the closed-form solutions exist, algebraic methods are preferred. However, algebraic methods generally depend on the specific configuration of the manipulators, and to overcome that obstacle numerical methods have been proposed for solving the IDP [16-18]. Numerical methods, such as Newton method, Gradient method and so on, usually adopt an iterative form. When a closed-form solution does not exist or is hard to obtain, numerical methods are used to find a solution.

However, they need initial guesses, which may greatly affect the convergence rate and the results. So, recently some researchers have tried to use evolutionary search algorithms to solve the engineering problem of robot inverse kinematics. Shital and Babu utilized a radial basis function neural network to solve the IDP for a 6R (6 revolute joints) serial robot. The proposed neural network can acquire the inverse displacement solution for any paths determined by the path planner [48]. Ayyıldız and Çetinkaya compared genetic algorithm (GA), particle swarm optimization (PSO), quantum particle swarm optimization (QPSO), and gravitational search algorithm (GSA) to solve the IDP of a real 4-DOF serial robot manipulator. They indicated that QPSO was most effective for solving this problem [49].

1.3.3 EDAs for constrained optimization problems

In the practical implementation, many constrained optimization problems (COPs) have constraints such as equality, inequality, linear, nonlinear, continuous, and discontinuous. The solution of COPs has commonly utilized deterministic methods like feasible direction methods, projection gradient methods, interior point and exterior point penalty function methods, and so on. But it is rather difficult for these methods to handle COPs with non-differentiable constraints, disjoint feasible regions, and objective functions that do not have analytical expressions. In the past two decades, many evolution algorithms such as genetic algorithms (GAs), evolution strategies (ES), artificial bee colony algorithm (ABCA), and particle swarm optimization (PSO) have been proposed to solve COPs. Typically, these algorithms are able to search the solution in a large space and do not require such mathematical properties as continuity and differentiability [50-53].

Some researchers have used EDAs to handle COPs. Grahl and Rothlauf [19] proposed a PolyEDA that used the method of Gibbs sampling to handle linear inequality constraints. But in their experiments, only the problem of Rosenbrock's function was tested by the algorithm. Moreover, the optimization problem only had the constraints of the variables’ upper and lower bounds. Simionescu et al. [20] presented EDAs with penalty and repair techniques to solve some COPs. However, only three problems were used to test the performance of EDAs. Besides, the constraints of these COPs did not contain equality constraints. Wang et al. [21] proposed EDAs to solve a class of nonlinear bi-level programming problems. They first transformed these problems into COPs with the Karush–Kuhn–Tucker (KKT) conditions and then adopted the penalty function method to handle the constraints. A disadvantage of their algorithms is that these problems must satisfy the KKT conditions. Besides, they set the penalty parameters uniformly at the constant value 10000 for all problems. However, the penalty parameter values usually should be adjusted depending on the problem. Some recently modified EDAs such as a new real-coded stochastic Bayesian optimization algorithm [3], the EDA with the capability of detecting promising areas [54], the regularized continuous EDAs [55], and so on, when testing the performance, they used optimization problems in which the variables only have the lower and upper bounds, but without equality, inequality, linear or nonlinear constraints.

1.3.4 Multi-objective optimization problems (MOPs)

Multi-objective optimization problems (MOPs) are very common in the engineering discipline. Conventional methods like ϵ-constraint method, weighted-sum approach, goal programming, and so on transformed the multi-objective problem into a single objective problem, for solution. They are convenient and simple to operate, but usually they ask for some prior knowledge and experience. Since the mid-1980s, some researchers have suggested to use evolutionary approaches to solve MOPs. These methods processed a group of solutions in parallel, possibly exploiting similarities of solutions by recombination. Finally they sought to obtain a set of uniformly distributed Pareto optimal solutions [56].

In the past decades, a number of evolutionary algorithms to search the Pareto optimal solutions were developed, such as vector evaluated genetic algorithm (VEGA) [57], Pareto archived evolution strategy (PAES) [58], strength Pareto evolution algorithm (SPEA) [56], NSGA-II [59], MOEA/D: A multi-objective evolutionary algorithm based on decomposition [60], NSGA-III [61], and so on. All above strategies used the genetic operations of crossover and mutation to produce the offspring. Some other evolution techniques, such as artificial immune system (AIS) [62], particle swarm optimization (PSO) [63], evolution strategies (ES) [64], differential evolution (DE) [65], and so on were proposed to handle MOPs, too. The EDAs were also applied to solving MOPs. For instance, Shim et al suggested the EDAs based on decomposition to handle MOPs [66]. In their research, EDAs were hybridized by three local search metaheuristic techniques: hill climbing, simulated annealing and evolutionary gradient search to improve the performance in solving the multi-objective traveling salesman problem. Karshenas et al. developed a multi-objective EDA that used the multi-dimensional Bayesian network as its probabilistic model [67]. In building the model, this EDA not only considered the linkage between the problem variables, but also made the objective values join the Bayesian network. The experimental results indicated that this multi-objective EDA could perform better than some algorithms with conventional genetic operators for some benchmark optimization problems.

1.4 Contributions and organization of the dissertation

This dissertation aims to improve the performance of EDAs in solving various types of optimization problems and extend the application of EDAs to engineering problems, particularly in the mechanical domain. The main contributions of this research can be summarized as follows:

1. An extreme elitism selection method is designed for the EDA based on univariate marginal Gaussian distribution. This selection method can make the EDA form a primary evolutionary direction and have a fast convergence rate. Meanwhile, it can also keep the population diversity to some extent and make the algorithm avoid premature convergence. The EDA with this selection can outperform the EDAs with other selection methods (truncation, tournament and proportional) and some other classical evolution algorithms for a set of low-dimensional and high-dimensional benchmark optimization problems. Specially, this modified EDA can obtain the optimal solutions in some problems with many variables (100-*D* and 200-*D*).

2. The application of EDAs is first extended to the manipulator inverse displacement problem. The method of EDA based on univariate marginal Gaussian distribution in continuous domain with extreme elitism selection does not depend on the configuration of the robotic arm or the existence of closed-form solutions. In addition, it does not need an initial guess or to set the step size as in some numerical methods. It provides an effective method for the inverse displacement problem of high degree-of-freedom (more than 6 degrees) robotic arms.

3. EDAs are fist systematically used to solve constrained optimization problems (COPs), which have various types of constraints like inequality, equality, linear, nonlinear, continuous or discontinuous. A group of benchmark COPs are used to test the performance of the modified EDAs. It is found that the EDA based on a single multivariate Gaussian distribution with extreme elitism selection can provide better performance than other EDAs and some well-known

evolution algorithms for most benchmark COPs. In addition, this EDA can show good performance for four mechanical design problems. The developed EDA does not need to transform the COPs to unconstrained problems or set extra parameters (like a penalty factor). Many types of COPs can be handled by this EDA, which can further extend the practical implementation of EDAs.

4. EDAs are proposed to combine differential mutations to solve multi-objective optimization problems (MOPs). This hybrid algorithm attempts to search a group of uniformly distributed Pareto optimal solutions for MOPs. It does not require prior knowledge or to transform MOPs into single objective problems. It can provide more alternative choices for the decision makers. Also, it needs fewer function evaluations than some state-of-the-art multi-objective evolutionary algorithms to provide good performance for some benchmark MOPs.

The thesis is organized into 6 Chapters.

Chapter 1 presents the research motivation, problem formulation, related work, the contributions of this study and the outline of the thesis.

Chapter 2 presents the design of an extreme elitism selection method for EDAs. Thirty different types of benchmark optimization problems are used to test the performance of the EDAs with the new selection in the experiment. The parameters of the algorithms for these problems are discussed and the no-free-Lunch theorem is implemented during the analysis. Finally, the performance of different EDAs and some other well-known evolution algorithms is compared.

Chapter 3 develops the application of the EDAs for the robotic arm inverse displacement problem (IDP). First, the advantages and disadvantages of some existing methods for IDP are discussed. Then the EDA based on univariate marginal Gaussian distribution with extreme elitism selection is used to solve the IDP of a 4-degree-of-freedom (DOF) Barrett WAM robotic arm. After that, the algorithm is combined with differential mutation to solve the IDP of the 7-DOF Barrett WAM robotic arm. Besides, the simulation of the developed EDA for searching the satisfactory solutions on a continuous curve is carried out. The results show that the proposed algorithm can achieve real-time application performance.

Chapter 4 presents the solution of constrained optimization problems (COPs) using EDAs. The EDAs with five different Gaussian distribution probabilistic models are evaluated using a set of benchmark COPs. It is found that for solving these benchmark problems, the EDA based on a single multivariate Gaussian distribution model with extreme elitism selection is more suitable than the EDAs with other distributions and some other typical evolution algorithms. Finally, this EDA is used to find optimal solutions of four mechanical design optimization problems.

Chapter 5 proposes EDAs that combine differential mutation (DM) to handle multi-objective optimization problems (MOPs). Some typical algorithms for MOPs are introduced. Then the hybrid algorithm EDA-DM is presented to search a group of uniformly distributed solutions for MOPs. EDAs determine the search direction while DM can maintain a diversified population. Some benchmark MOPs are introduced to carry out experiments. The test results of several different algorithms are compared.

Chapter 6 presents the summary of this dissertation and gives some suggestions for possible future work.

Chapter 2: Estimation Distribution Algorithms with Extreme Elitism Selection

2.1 Introduction

Estimation distribution algorithms (EDAs) are evolutionary population-based search techniques. The main advantage of these approaches is that they do not need the information of differentiability or continuity of the analytical optimization problem and they can search the optimal solutions from a group of points in parallel. A disadvantage of these techniques is that they may be trapped in some local minima as the population diversity fades during the search. Some improvements have been incorporated into the method in the past decades, such as introducing a niching method and a recombination operator into the population [7], tuning the eigenvalues of the covariance matrix of the multivariate Gaussian models [8], adjusting the procedures of sampling the offspring in accordance with the iteration generation to gradually increase the population diversity [33], detecting promising areas [54], and so on. Moreover, some other evolution algorithms like differential evolution (DE) algorithm [6], particle swarm optimization (PSO) [35], and artificial immune system (AIS) [36] have been integrated with EDAs to overcome the weaknesses of the original algorithms. However, most of these modified EDAs have been tested on optimization problems of dimension less than 100. It is known that the complexity and the search space of an optimization problem increase rapidly as the dimensionality rises. In addition, many evolution algorithms such as genetic algorithms (GAs), ant colony optimization (ACO), particle swarm optimization (PSO), differential evolution (DE), and so on suffer from the "curse of dimensionality," which means their performance deteriorates rapidly as the problem dimension increases [2]. So it is important to improve the performance of the EDAs in solving high-dimensional problems. In this chapter, the modified EDA not only seeks to achieve good performance for some low-dimensional problems, but also for some high-dimensional problems.

Some EDAs have adopted complicated probabilistic models and procedures, which can improve the performance to some extent. However, the complicated probabilistic models and procedures may lead to high complexity and computational cost, especially in high-dimensional problems. Larrañaga and Lozano indicated that the EDAs of EGNA and EMNA based on multiple interdependencies took a much longer computing time than the EDAs of UMDA$_C$ and MIMIC$_C$, which adopted the probabilistic models without interaction between variables or only with relationship between pair of variables. EMNA$_a$ was not suitable for the problems with a dimension of 50 as it spent too much computing time [9]. Gao and Culberson proved that the space complexity of FDA and BOA was exponential in the problem size even if the optimization problem had a very sparse interaction structure [12]. Therefore, in the present work the EDA based on the univariate marginal Gaussian distribution model is adopted to handle a group of optimization problems. This probabilistic model has a low computational cost. It assumes that the variables are independent and its joint probability equals the product of the marginal probability of each variable (i.e., the variables are assumed to be statistically independent).

2.2 EDAs using an extreme elitism selection

2.2.1 The selection methods in EDAs

EDAs usually adopt three widely used selection methods: truncation, proportionality, and tournament. The truncation selection ranks all solutions according to their fitness and then M

(Note: $M = \alpha * PS$, $\alpha \in (0, 1)$, α is the selection proportion and PS is the population size) best solutions are truncated as the parent population [69]. This method is rather convenient and fast, but it has the weakness that it treats every solution in the parent population equally. It cannot sufficiently incorporate the effect of a few leading best solutions, because from the 1st to Mth best solution, each of them is uniformly selected once. The proportional selection assigns selection probabilities according to the relative fitness: $p_i = f_i / (f_1 + f_2 + \cdots + f_{PS})$ [70]. This selection has the disadvantage that it is inconvenient to handle a negative fitness. When using the objective function as the fitness function, sometimes the fitness may be a negative value. Then, a new fitness function has to be designed. The tournament selection elects only the 1st best solutions from a random subpopulation and then repeats this selection PS times to choose PS solutions as the parent population [71]. The problem here is that the computing time can be excessive if the size of the population and subpopulation is high.

Although these three selection methods originated from GAs, some researchers still seek to analyze and improve the EDAs from the perspective of selection strategies. Yu et al. indicated that a binary tournament selection can produce a similar effect as truncation with a selection pressure of 1.25. Besides, they proposed a model based on the entropy measurement. This mode demonstrated that when constructing Bayesian networks, a rather small selection pressure cannot detect the correct linkages, while a too large selection pressure can make the sampling noise become large and cloud the signal of the correct linkages. An optimal selection pressure should be placed somewhere in the middle [72]. Lima et al. indicated that the use of truncation selection can result in more accurate structural linkage information than tournament selection for BOA. Moreover, if it is required to acquire near-optimal solutions with high reliability using minimal number of function evaluations, tournament selection with restricted replacement is the best strategy with BOA [73]. Lima et al. further demonstrated that tournament is an efficient selection method for Bayesian EDAs when the scoring metric is adjusted in light of its natural distribution in the mating pool. Specifically, greater tournament size needs more demanding metric to accept edge additions in a Bayesian network. Also, truncation selection is more appropriate when using standard scoring metrics, but its corresponding model quality is still inferior to the one obtained by tournament selection with the new s-penalty [37]. Hong et al. proposed an Over-Selection strategy to improve EDAs for small population size. This strategy selects much more solutions as the parent population than the original population size. Some experimental results showed that EDA with this Over-Selection method is often able to achieve a better result without significantly increasing its number of fitness evaluations when compared with the classical EDAs [74]. Brownlee et al. investigated an EDA with Markov networks. For this EDA, they compared the impact of selecting only highly fit solutions, only poor solutions, and a mixture of highly fit and poor solutions. They found that the selection of the fittest only is suboptimal in some circumstances. Some poor solutions also have important information [75]. Valdez et al. developed an empirical selection distribution for EDAs. This method can employ all the information from the population to accurately approximate the selection distribution that can be directly used to sample the offspring for the next generation [76]. Santana et al. proposed a customized selection based on the assumption that non-structural learning and structural learning may have different requirements of the population diversity for accurately modeling. EDAs with this selection learn the structure and the parameters of the model from different selection probabilities or population. This new selection improved the performance of EDAs in the optimization problem of the functional model protein [77].

2.2.2 The role of top solutions in evolution algorithms

EDAs are evolutionary algorithms based on a population of individuals from which some promising solutions are selected to generate new offspring. In some evolution algorithms, these few leading best solutions are sorted by the fitness; especially, the 1st best solution of every generation, which can play an important role in the evolution. For instance, the evolutionary strategy of particle swarm optimization (PSO) [78] is given by:

$$V_{id}^{(g+1)} = V_{id}^{(g)} + C_1 r_{1d} (\boldsymbol{Pbes_{id}} - X_{id}) + C_2 r_{2d} (\boldsymbol{Gbest_{id}} - X_{id}) \tag{2.1}$$

$$X_{id}^{(g+1)} = X_{id}^{(g)} + V_{id}^{(g+1)} \tag{2.2}$$

The personal best solution ***Pbest***

Generally, every parent ($\boldsymbol{X_i}, \boldsymbol{\sigma_i}$), $\forall i \in \{1, 2, \cdots, \mu\}$, produces λ/μ offspring on average. Here, λ is the population size, μ is the parent population size, and $\lambda/\mu \approx 7$ usually is optimal [81].

In view of these considerations, the present study develops a new selection model for EDAs that will highlight the role of the 1st best solution of every generation. Moreover, the effect of few other leading solutions such as the 2nd to the 5th will be included as well. Hence, the present approach is different from PSO and DE, as indicated above. PSO and DE emphasize only the 1st best solution. Sometimes the personal best and the global best solutions are different in PSO, but in fact they are both the 1st best solution. A minor distinction exists because they are chosen from different population ranges. The aim of highlighting the role of the few other leading solutions is to decentralize the effect of the 1st best solution and improve the population diversity. In addition, in order to further enrich the population, many other solutions that are defined as ordinary promising solutions (not the leading) will be retained by the new selection model that is proposed in the present work.

2.2.3 EDAs scheme with extreme elitism

An optimization problem is presented now as the basis to develop the EDA with a new selection method and to highlight the role of a few top solutions in the associated evolutionary procedure. Suppose that *f*(*x*) is a multimodal function and the reference analytical optimization problem is as follows:

$$\text{Minimize } f(x) = 7\sin^4(3x) + 8\cos^3(2x) + 2\cos^2(x) + 5\sin(x) - x\cos(3x) + \exp(-x^2) \quad (2.8)$$

$$\text{Subject to } x \in [-5\pi, 5\pi]$$

This problem has the global minimum -19.706 at x = -13.750. Presented now is the EDA scheme for this problem.

Step 1: Set the population size *PS* = 200 and the maximum iteration generation *G* = 25. Generate the initial population pop(x_i)^0 for the problem variable with the code given by:

$$\text{pop}(x_i)^0 = a_i + (b_i - a_i) * \text{rand}(1, PS) \quad (i = 1, 2, \cdots, D) \tag{2.9}$$

Here $x_i \in [a_i, b_i]$, *D* is the problem dimension, and "rand(1, *PS*)" produces a 1×PS matrix with the elements uniformly distributed in the range [0, 1]; hence, pop(x_i)^0 has its initial population uniformly distributed in the range [a_i, b_i]. For the present problem, [a_i, b_i] = [-5π, 5π] and *D* = 1.

Step 2: Set g=g+1. If $g <= G$, go to **Step 3**; else, stop;

Step 3: Compute the fitness value of each solution in pop(x_i)$^{g-1}$. If the fitness of some solutions meets the stopping criteria, stop the loop; else go to **Step 4**;

Step 4: With the fitness equals the objective function value, select *M* = 0.5 × 200 = 100 best solutions as the promising solutions to constitute the parent population;

Step 5: Estimate the mean and the variance of the parent population according to:

$$\mu_i = \frac{\sum_{j=1}^{M} x_{ij}}{M} \qquad \sigma_i^2 = \frac{\sum_{j=1}^{M} (x_{ij} - \mu_i)^2}{M - 1} \qquad (i = 1, 2, \cdots, D;\ j = 1, 2, \cdots, M) \tag{2.10}$$

Here, x_{ij} is the jth solution of x_i among the *M* best solutions.

Step 6: Use the mean and the variance to build a univariate marginal Gaussian distribution model. The probability density function of this distribution model is given by:

$$f(x_i, \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}) \qquad (i = 1, 2, \cdots, D) \tag{2.11}$$

Here *D*=1. If there are many variables, the density function of this model is expressed by:

$$f_{\boldsymbol{X}}(x_1, \cdots, x_D) = \prod_{i=1}^{D} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}) \qquad \boldsymbol{X} = (x_1, \cdots, x_D)^T \tag{2.12}$$

Step 7: Sample *PS* offspring solutions for each variable with the code given by:

$$\text{offspring\_individual}_{ij}=\text{normrnd}(\mu_i,\sigma_i,D,PS)\quad (i=1,2,\cdots,D;\ j=1,2,\cdots,PS) \tag{2.13}$$

Check every offspring solution: if some solution exceeds the bound, make it equal a new random number in the bound ("rand" produces a random number in [0, 1]), according to:

$$\text{if offspring\_individual}_{ij}\notin[a_i,b_i],\ \text{offspring\_individual}_{ij}=a_i+(b_i-a_i)*\text{rand} \tag{2.14}$$

Then return to **Step 2**.

In **Step 4**, the program uses truncation selection to select the 100 best solutions. For the new selection method, the **Step 4** is changed as follows:

```
Step 4: $M = 100$
    for $i = 1:D$
        for $j = 1:25$     parent population_$\theta_{ij} = \theta_i(\text{II}(1))$      end
        for $j = 26:45$    parent population_$\theta_{ij} = \theta_i(\text{II}(2))$      end
        for $j = 46:60$    parent population_$\theta_{ij} = \theta_i(\text{II}(3))$      end
        for $j = 61:70$    parent population_$\theta_{ij} = \theta_i(\text{II}(4))$      end
        for $j = 71:75$    parent population_$\theta_{ij} = \theta_i(\text{II}(5))$      end
        for $j = 76:M$     parent population_$\theta_{ij} = \theta_i(\text{II}(j-70))$ end
    end
```

Here θ_i(II(1)) to θ_i(II(5)) are the 1st to the 5th solutions (denoted as the top or the leading solutions), sorted according to the fitness. Then the 1st to the 25th solutions in the parent population all are made equal to the 1st best solution. The 26th to the 45th solutions in the parent population are all made equal to the 2nd solution, and so on. But each of the solutions from the 76th to 100th in the parent population are made equal to some ordinary solutions. For instance, the 76th solution equals the 6th solution and the 77th solution equals the 7th solution, and so on and finally the 100th solution equals the 30th solution. As a result, these leading solutions account for 25, 20, 15, 10, and 5 items in the parent population, while each of the 6th to the 30th solutions (thought as the ordinary solutions) equally accounts for only 1 item.

Fig.2.1 shows this selection procedure. It is seen that this selection effectively highlights the role of a few leading solutions. These leading solutions are thought of as the elites in the population and they build a small elite corps, which plays the role as ***Pbest*** and ***Gbest*** in PSO, as $\boldsymbol{X}^t_{best}$ in DE, or as ($\boldsymbol{X_i}, \boldsymbol{\sigma_i}$) in ES. EDA with this selection method is termed the extreme elitism estimation distribution algorithm (EE-EDA). This is because EE-EDA makes these elites account for more copies than the ordinary solutions in the parent population and strongly highlights the role of these elites in the evolutionary process. The proposed concept of elitism is different from the conventional elitism, which copies the best solution(s) from the current population to the next population, unaltered [82]. Now, EE-EDA and EDA with truncation selection (TR-EDA) are run separately to solve the problem 2.8.

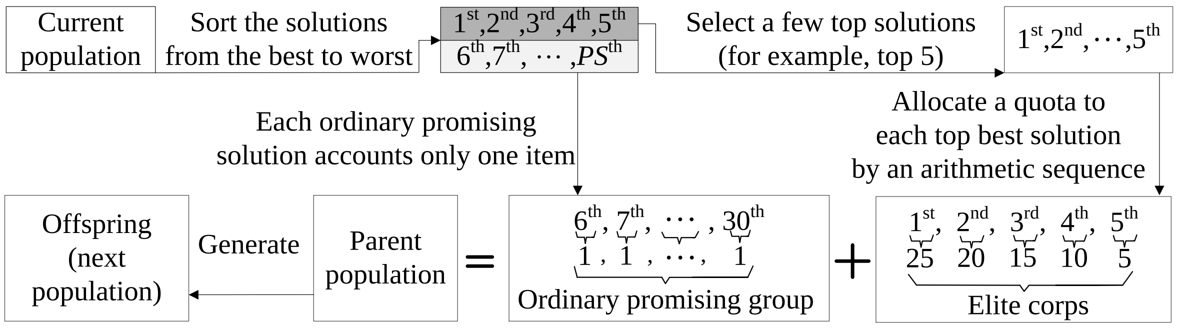
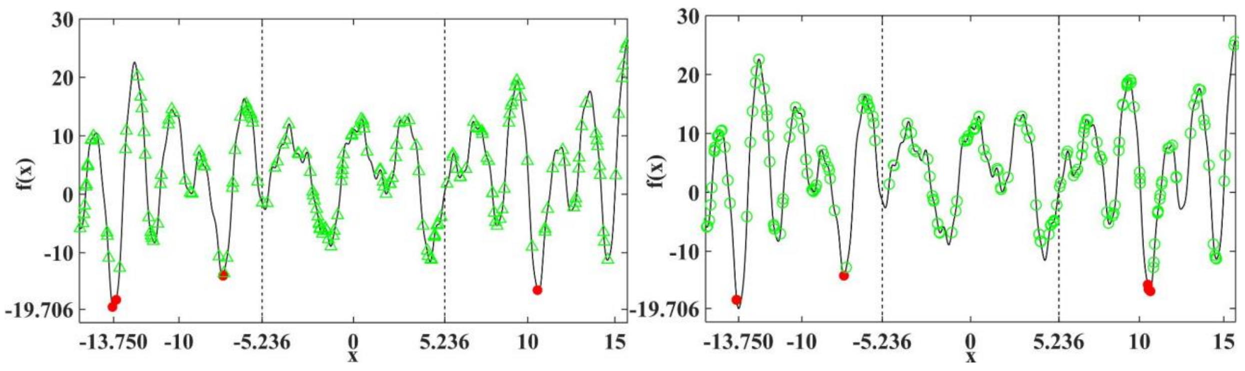

Current population
Sort the solutions from the best to worst
1st,2nd,3rd,4th,5th
6th,7th, ⋯ ,PSth
Select a few top solutions (for example, top 5)
1st,2nd,⋯,5th
Each ordinary promising solution accounts only one item
Allocate a quota to each top best solution by an arithmetic sequence
Offspring (next population)
Generate
Parent population
=
6th, 7th, ⋯, 30th
1, 1, ⋯, 1
Ordinary promising group
+
1st, 2nd, 3rd, 4th, 5th
25 20 15 10 5
Elite corps

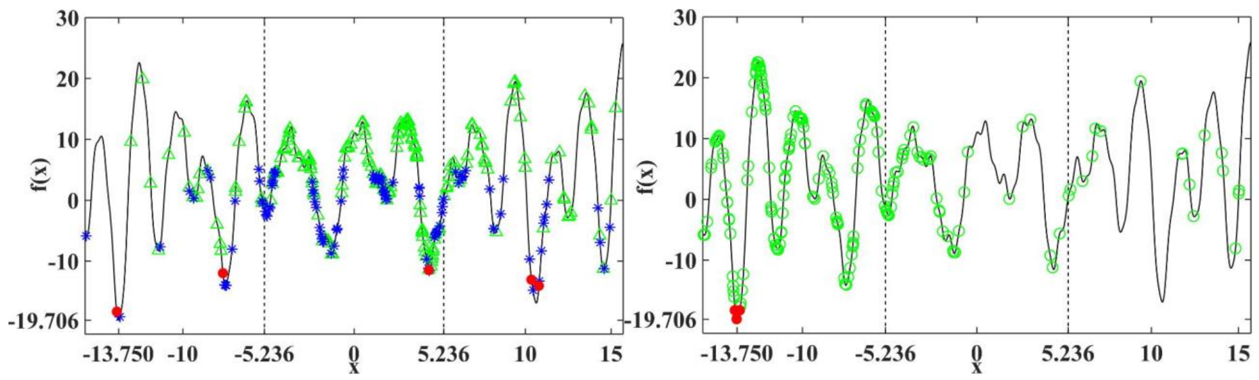
Figure 2.1 The procedure of extreme elitism selection

Fig. 2.2(a) and 2.2(b) show the initial population of TR-EDA and EE-EDA, respectively, and they both uniformly distribute in the range [-5π, 5π]. In fact, some leading solutions marked with solid circle (red colour) already have almost reached the global optimum in the initial population themselves. However, Fig. 2.3(a) shows that there are fewer solutions in the 5th generation of TR-EDA close to the global optimum than the initial population. Fig. 2.4(a) and 2.5(a) show that virtually no solution marked with triangle and solid circle is near the global optimum and more solutions have moved to the center region in the 10th and 20th generations.



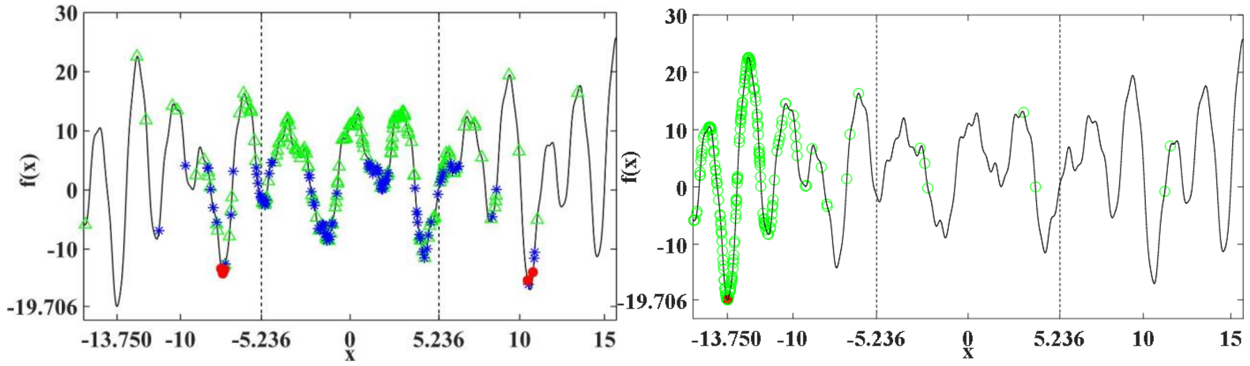
(a) TR-EDA (b) EE-EDA

Figure 2.2 The initial population.



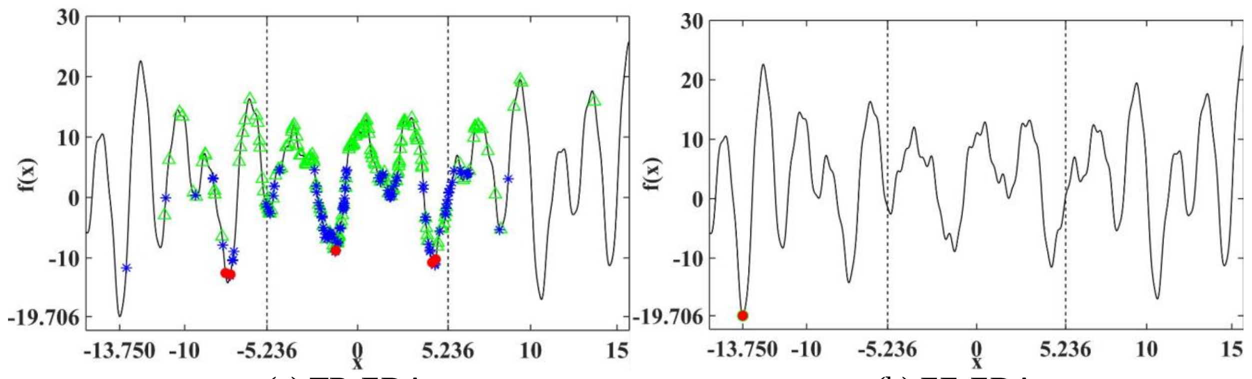
(a) TR-EDA (b) EE-EDA

Figure 2.3 Population in the 5th generation.



(a) TR-EDA (b) EE-EDA

Figure 2.4 Population in the 10th generation.



(a) TR-EDA (b) EE-EDA

Figure 2.5 Population in the 20th generation.

This is because TR-EDA adopts equalitarianism and each solution has the same frequency in the parent population. In Fig. 2.3(a), 2.4(a) and 2.5(a), the solutions marked with star (blue colour) are those selected to the parent population in the prior generation (4th, 9th and 19th). Based on them, a Gaussian model is built to sample the population of the 5th, 10th and 15th generations, shown with triangles and solid circles. It is seen that the star solutions are located nearly symmetrical manner about the center point, and hence the mean of the Gaussian model is close to the center. Besides, as the generation increases, the variance gradually decreases, and the population sampled from these Gaussian models tend to gather in the center segment of the range.

However, if these solutions do not have the same frequency and these top solutions have a higher weight in the parent population, the mean of Gaussian model will be pulled towards some leading best solutions, even though the selected solutions are symmetrical about the center point. In the Gaussian distribution, the mean vector guides the exploration direction of the algorithm and the offspring are sampled from the search space surrounding the mean vector. When the mean approaches the global optimum, the offspring usually also approach the global optimum. Hence, in Fig. 2.3(b), many solutions in the 5th generation of EE-EDA have moved to the left segment of the range; in Fig. 2.4(b), the population of the 10th generation has surrounded the global optimum; and in Fig. 2.5(b) the population of the 20th generation has already converged to the global optimum. Furthermore, since some of the same best solutions are in the parent population, the variance of the Gaussian model decreases faster than that produced by the truncation selection. Therefore, generally EE-EDA has a faster convergence rate than TR-EDA. In Fig. 2.6, EE-EDA has nearly converged to the minimum before the 15th generation while TR-EDA still has not converged in the 25th generation.

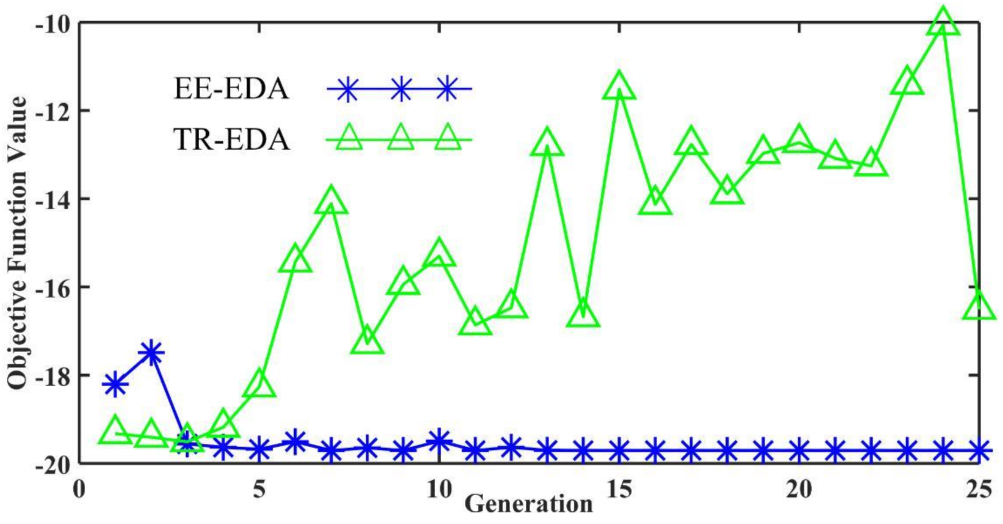


Figure 2.6 Convergence of TR-EDA and EE-EDA for function (2.8).

2.2.4 Percentage of top solutions in the parent population

From the given example, it is clear that a few top solutions should play an import role in the evolution. But their percentage cannot be too high, especially in complex high-dimensional problems. Consider Griewanks function given by:

$$f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad x_i \in [-600, 600] \tag{2.15}$$

Now this function is used to study what percentage of these top solutions is appropriate in the parent population. This function is a multimodal function. Fig. 2.7 shows that it already has several local minima even there are only 2 variables in the interval [-20, 20].

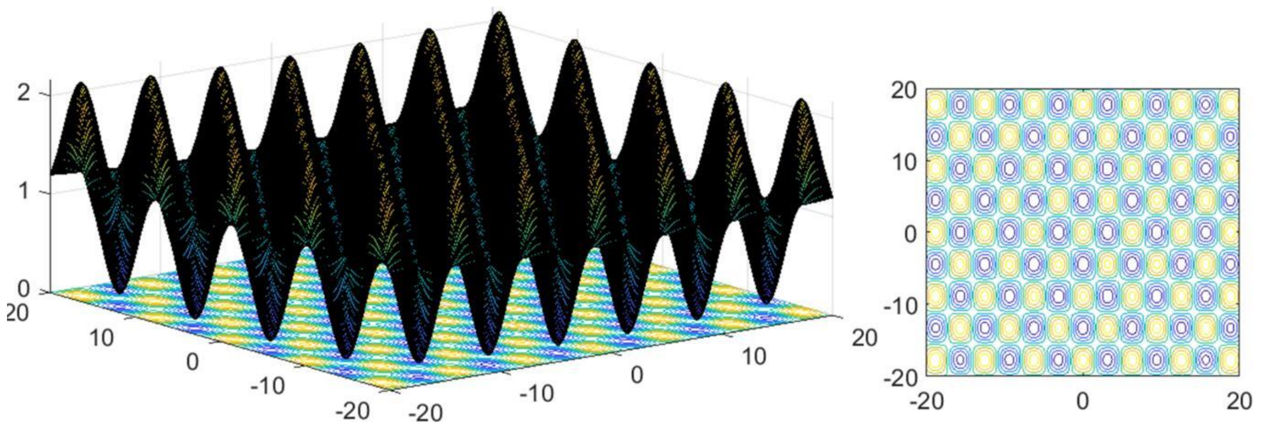


Figure 2.7 The distribution of minima of Griewanks function.

In the results presented in Table 2.1, the combination of portions of the top solutions is kept the same, as follows: the 1st to 5th solutions account for 25, 20, 15, 10 and 5 portions, respectively, in the parent population. Then from each of the 6th to the (M-70)th ordinary solution, only 1 portion is taken, where $M = 0.5\times PS$, and "D" is the dimension of the problem and is equal to 30, 100 and 200. The population size PS is separately set at 200, 300, ⋯, 1200 for D = 30; It ends at 1200 since the failure tests appear when PS = 1200; for D = 100, PS equals 400, ⋯, 900 since these values of PS for D = 30 are successful. "Per" indicates the percentage of the top solutions in the parent population (e.g., when PS = 400, $M = 0.5\times400 = 200$, so Per = (25+20+15+10+5) / 200×100% = 37.5%). "Mean" is the mean value of the final best fitness values of 30 independent tests. "Std" denotes the standard deviation of the 30 independent tests. "Best" is the best test result among 30 tests while "Worst" is the worst one. "ST" denotes the successful test time of obtaining the global optimum. "AG" indicates the average generation (excluding the generation of failed tests) in which the algorithm reaches the global optimum. Besides, for *D* = 30, the maximum iteration generation *G* = 150; for *D* = 100 it is *G* = 250; and for *D* = 200 it is *G* = 350. The test result shown in bold indicates that the algorithm successfully found the global minimum in all 30 tests.

Table 2.1 Test results of EE-EDA with different percentage of leading best solutions in parent population.

$f(\boldsymbol{x}_{\text{global optimum}}) = 0$ Combination: 25, 20, 15, 10, 5, 1, 1,⋯, 1								
D	***PS***	**Per**	**Mean**	**Std**	**Best**	**Worst**	**ST**	**AG**
30	200	75%	8.05e-01	5.24e-01	6.02e-10	2.56e+00	0	150
	300	50%	9.57e-04	2.77e-03	0	9.86e-03	24	92
	400	**37.5%**	**0**	**0**	**0**	**0**	**30**	**103**
	500	**30%**	**0**	**0**	**0**	**0**	**30**	**113**
	600	**25%**	**0**	**0**	**0**	**0**	**30**	**121**
	700	**21.4%**	**0**	**0**	**0**	**0**	**30**	**128**
	800	**18.8%**	**0**	**0**	**0**	**0**	**30**	**134**
	900	**16.7%**	**0**	**0**	**0**	**0**	**30**	**139**
	1000	**15%**	**0**	**0**	**0**	**0**	**30**	**143**
	1100	**13.6%**	**0**	**0**	**0**	**0**	**30**	**147**
	1200	12.5%	1.85e-17	4.21e-17	0	1.11e-16	26	149
100	400	37.5%	9.08e-03	3.95e-02	0	2.10e-01	3	209
	500	30%	6.06e-13	3.32e-12	0	1.82e-11	29	209
	600	**25%**	**0**	**0**	**0**	**0**	**30**	**222**
	700	**21.4%**	**0**	**0**	**0**	**0**	**30**	**235**
	800	**18.8%**	**0**	**0**	**0**	**0**	**30**	**245**
	900	16.7%	7.85e-16	2.47e-16	1.33e-15	4.44e-16	0	250
200	600	25%	1.70e-11	9.33e-11	0	5.11e-10	29	329
	700	**21.4%**	**0**	**0**	**0**	**0**	**30**	**339**
	800	18.8%	1.11e-16	0	1.11e-16	1.11e-16	0	350

From Table 2.1 it is seen that when the percentage is 75%, all the tests fail to find the global optimum and stop the iteration in the final generation 150 for *D* = 30. But after the percentage drops to 50%, 24 tests reached the global optimum. When the percentage is further lowered to 37.5%, all the tests achieved the global optimum at AG = 103. Therefore, the upper bound of the percentage is about 37.5% for *D* = 30. However, as the percentage is decreased to 12.5%, some failed tests begin to appear. So if the maximum iteration generation *G* is fixed (usually this has to be the case because the algorithm cannot run for ever if it does not reach the global optimum), there is a corresponding lower bound for the percentage. Besides, as the percentage decreases, AG increases. Therefore, the algorithm with a higher percentage of the leading solutions has a faster convergence rate. However, if the percentage is too high, the population diversity will be affected and the algorithm tends to converge prematurely. In contrast, if the percentage is too low, population diversity will increase but the convergence rate will decrease. For high-dimensional situations of D = 100 and 200, it is seen that the interval of the percentage that can make the algorithm find the global optimum becomes narrow. This is because the complexity of the function increases when the dimension rises. Generally, when the optimization problems are more complicated, the parameters of the algorithms for these problems become more sensitive.

2.2.5 The combination of top solutions in the elite corps

Now the best parameters for Griewanks function are chosen: for D=30, PS=400, Per=37.5% and G=150; for D=100, PS=600, Per=25% and G=250; and for D=200, PS=700, Per=21.4% and G=350. These parameters can make the algorithms maintain a good balance between the convergence rate and the population diversity, as they can make the algorithms find the global optimum very fast and have many successful tests simultaneously. In this type of algorithms, the small elite corps typically consists of 5 leading best solutions. Table 2.2 lists the test results when the elite corps has different numbers of leading best solutions while the total percentage is kept the same. For instance, in the combination: 75, 1,⋯, 1, the corps only has the 1st best solution, which takes 75 items in the parent population, resulting in the same percentage 37.5%. For other combinations, the portions from the 1st to the last top solution in the corps decreases according to an arithmetic progression seen in the following equation:

$$S_k = ka_1 + \frac{k(k-1)}{2}d \qquad a_k = a_1 + (k-1)d \tag{2.16}$$

For example, suppose that the corps consists of the 1st to 3rd leading solutions, S_k=75, k=3, and a_1=15; then giving d=10, a_2=25 and a_3=35. If the total number of top solutions in the corps cannot be an integer, the last top solution gets the additional portions.

Table 2.2 Test results of EE-EDA with different combinations in the elite corps.

D	*PS*	Combination	Mean	Std	Best	Worst	ST	AG
30	400	75	1.81e-02	3.18e-02	3.47e-08	1.34e-01	0	150
		50, 25	7.39e-04	2.83e-03	0	1.23e-02	21	100
		35, 25, 15	2.47e-04	1.35e-03	0	7.39e-03	29	100
		28, 22, 16, 9	**0**	**0**	**0**	**0**	**30**	**101**
		25, 20, 15, 10, 5	**0**	**0**	**0**	**0**	**30**	**103**
		15, 14,13, 12, 11, 10	**0**	**0**	**0**	**0**	**30**	**106**
100	600	75	1.36e+00	2.52e-01	1.12e+00	2.39e+00	0	250
		50, 25	2.73e-02	4.00e-02	3.58e-05	1.42e-01	0	250
		35, 25, 15	3.61e-07	1.75e-06	0	9.53e-06	14	219
		28, 22, 16, 9	**0**	**0**	**0**	**0**	**30**	**220**
		25, 20, 15, 10, 5	**0**	**0**	**0**	**0**	**30**	**222**
		15, 14,13, 12, 11, 10	**0**	**0**	**0**	**0**	**30**	**228**
200	700	75	1.22e+00	2.00e-01	7.97e-01	1.84e+00	0	350
		50, 25	7.20e-04	1.98e-03	5.99e-09	9.87e-03	0	350
		35, 25, 15	2.49e-4	1.35e-03	0	7.40e-03	19	340
		28, 22, 16, 9	**0**	**0**	**0**	**0**	**30**	**338**
		25, 20, 15, 10, 5	**0**	**0**	**0**	**0**	**30**	**339**
		15, 14,13, 12, 11, 10	**0**	**0**	**0**	**0**	**30**	**346**

It is seen from Table 2.2 that when the elite corps only has the 1st best solution, the algorithms all fail to reach the global optimum. When the corps consists of the 1st and 2nd solutions, some tests succeed for *D*=30. No test succeeds for *D*=100 or 200. But when the team has four, five or six solutions, the algorithms can all achieve the global optimum in 30 tests. This is because extruding only one or two leading solutions can make the parent population possess too many of the same solutions. As a result, the variance of the Gaussian model becomes small and the population diversity decreases, which leads to premature convergence. Hence, the robustness of the algorithm can be improved by selecting more top solutions into the elite team. In order to make the program of EE-EDA explicit and uniform, the elite corps of the subsequent test problems in the present work is made to have 5 top solutions and the used combination is 25, 20, 15, 10 and 5. Any exception will be noted.

2.3 Experimental evaluation of EE-EDA

As introduce in section 2.2.1, there are three main selection methods for EDAs: truncation, proportional, and tournament. Truncation has been analyzed in the previous section. Both proportional and tournament selection methods can exploit the effect of some top solutions. In the proportional selection, the probability of the solution that is being elected is proportional to its fitness. Hence, if one solution is much better than the other solutions, it can take more portions than the other solutions in the parent population. The tournament selection runs several "tournaments" among a few random solutions chosen from the current population and selects the 1st best solution. Hence, some solutions may be selected many times. Therefore, the proportional and the tournament methods are similar to the extreme elitism selection and can also develop the effect of some leading solutions in the evolution. Next, EE-EDA is compared with EDA in the presence of proportional, tournament, and truncation selection methods by testing 30 well-known benchmark functions.

Functions $f_1(x)$ through $f_{11}(x)$ represent high-dimensional problems as seen in Table 2.3. Among these, $f_1(x)$ has one minimum; $f_2(x)$ is a noisy quartic function; $f_3(x)$ through $f_5(x)$ are unimodal; $f_6(x)$ through $f_{11}(x)$ are multimodal; functions $f_{12}(x)$ through $f_{30}(x)$ are low dimensional as seen in Table 2.4, which have only a few local minima. It is necessary to test a variety of functions for making general observations about them. Besides, these functions have been widely used in the experiments of testing other famous optimization strategies such as ES, DE, PSO and so on [6, 8, 83-87] in the past decades, so they represent some class of typical optimization problems.

Table 2.3 High-dimensional test problems.

	Function	GO	Min
Step	$f_1(\boldsymbol{x})=\sum_{i=1}^{D}\lvert x_i+0.5\rvert^2 \quad x\in\left[-100,100\right]^{\mathrm{D}}$	[-0.5,⋯, -0.5]	0
Quartic with noise	$f_2(\boldsymbol{x})=\sum_{i=1}^{D} ix_i^4-random[0,1) \quad x\in\left[-1.28,1.28\right]^{\mathrm{D}}$	[0, ⋯,0]	0
Sphere	$f_3(\boldsymbol{x})=\sum_{i=1}^{D} x_i^2 \quad x\in\left[-100,100\right]^{\mathrm{D}}$	[0, ⋯,0]	0
Schwefel2.22	$f_4(\boldsymbol{x})=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert \quad x\in\left[-10,10\right]^{\mathrm{D}}$	[0, ⋯,0]	0
Rosenbrock	$f_5(\boldsymbol{x})=\sum_{i=1}^{D}[100(x_i^2-x_{i+1})^2+(x_i-1)^2] \quad x\in[-2.048,2.048]^{\mathrm{D}}$	[1, ⋯,1]	0
Schwefel2.26	$f_6(\boldsymbol{x})=\sum_{i=1}^{D}-x_i\sin(\sqrt{\lvert x_i\rvert}) \quad x\in\left[-500,500\right]^{\mathrm{D}}$	[418.9829, ⋯,418.9829]	-418.9829**D*
Ackley	$f_7(\boldsymbol{x})=20+e-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i))$ $x\in[-32.768,32.768]^{\mathrm{D}}$	[0, ⋯,0]	0
Rastrigin	$f_8(\boldsymbol{x})=10D+\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)) \quad x\in\left[-5.12,5.12\right]^{\mathrm{D}}$	[0, ⋯,0]	0
Non-continuous Rastrigin	$f_9(\boldsymbol{x})=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10) \quad x\in\left[-5.12,5.12\right]^{\mathrm{D}}$ $y_i=\begin{cases} x_i & \lvert x_i\rvert<\frac{1}{2} \\ \frac{round(2x_i)}{x} & \lvert x_i\rvert>=\frac{1}{2}\end{cases} \quad for \;\; i=1,2,\cdots,D.$	[0, ⋯,0]	0
Weierstrass	$f_{10}(\boldsymbol{x})=\sum_{i=1}^{D}(\sum_{i=1}^{k_{\max}}[a^k\cos(2\pi b^k(x_i+0.5))])$ $-D\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k\cdot 0.5)]$ where $a=0.5, b=0.3, k_{\max}=20, x\in\left[-0.5,0.5\right]^{\mathrm{D}}$	[0, ⋯,0]	0
Griewanks	$f_{11}(\boldsymbol{x})=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1 \quad x\in\left[-600,600\right]^{\mathrm{D}}$	[0, ⋯,0]	0

During the comparison, on the same function with the same number of dimensions, the algorithms use a uniform population size *PS* and maximum iteration generation *G*, and only the selection model is distinct. But for different functions, *PS* and *G* will change, because each function has its own particular biases and properties.

Table 2.4 Low-dimensional test problems.

	Function	GO	Min
Beale	$f_{12}(\boldsymbol{x})=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.265-x_1+x_1x_2^3)^2$ $x_i\in[-4.5,4.5]^2 \quad i=1,2$	[3,0.5]	0
Bohachevsky1	$f_{13}(\boldsymbol{x})=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$ $x_i\in[-100,100]^2 \quad i=1,2$	[0,0]	0
Bohachevsky2	$f_{14}(\boldsymbol{x})=x_1^2+2x_2^2-0.3\cos(3\pi x_1)\cdot\cos(4\pi x_2)+0.3$ $x_i\in[-100,100]^2 \quad i=1,2$	[0,0]	0
Bohachevsky3	$f_{15}(\boldsymbol{x})=x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$ $x_i\in[-100,100]^2 \quad i=1,2$	[0,0]	0
Booth	$f_{16}(\boldsymbol{x})=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2 \quad x_i\in[-10,10]^2 \quad i=1,2$	[1,3]	0
Branin	$f_{17}(\boldsymbol{x})=(x_2-\frac{5}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos(x_1)+10$ $x\in[-5,10]^2 \quad i=1,2$	[π,2.275]	0.397887
Eason	$f_{18}(\boldsymbol{x})=-\cos(x_1)\cdot\cos(x_1)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$ $x\in[-100,100]^2 \quad i=1,2$	[π, π]	-1
Goldstein-Price	$f_{19}(\boldsymbol{x})=[1+(x_1+x_2+1)^2\cdot(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\cdot[30+(2x_1-3x_2)^2(18-32x_1+12_1^2+48x_2-36x_1x_2+27x_2^2)]$ $x\in[-2,2]^2 \quad i=1,2$	[0, -1]	3
Hartmann 1	$f_{20}(\boldsymbol{x})=-\sum_{i=1}^{4}\boldsymbol{a}_i\exp(-\sum_{j=1}^{3}\boldsymbol{A}_{ij}(x_j-P_{ij})^2),$ where $\boldsymbol{a}=[1.0\ 1.2\ 3.0\ 3.2]^T, \quad x\in[0,1]^3 \quad i=1,2,3$ $\boldsymbol{A}=\begin{bmatrix}3.0 & 10 & 30\\ 0.1 & 10 & 35\\ 3.0 & 10 & 30\\ 0.1 & 10 & 35\end{bmatrix} \quad \boldsymbol{P}=10^{-4}\begin{bmatrix}6890 & 1170 & 2673\\ 4699 & 4387 & 7470\\ 1091 & 8732 & 5547\\ 381 & 5743 & 8828\end{bmatrix}$	[0.114614, 0.555649, 0.852547]	-3.86278
Hartmann 2	$f_{21}(\boldsymbol{x})=-\sum_{i=4}^{4}\boldsymbol{a}_i\exp(-\sum_{j=1}^{6}\boldsymbol{A}_{ij}(x_j-\boldsymbol{P}_{ij})^2), \quad x\in[0,1]^3 \quad i=1,\cdots,6$ where $\boldsymbol{a}=[1.0\ 1.2\ 3.0\ 3.2]^T, \boldsymbol{A}=\begin{bmatrix}10 & 3 & 17 & 3.5 & 1.7 & 8\\ 0.05 & 10 & 17 & 0.1 & 8 & 14\\ 3 & 3.5 & 1.7 & 10 & 17 & 8\\ 17 & 8 & 0.05 & 10 & 0.1 & 14\end{bmatrix}$ $\boldsymbol{P}=10^{-4}\begin{bmatrix}1312 & 1696 & 5569 & 124 & 8283 & 5886\\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991\\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650\\ 4047 & 8828 & 8732 & 5743 & 1091 & 381\end{bmatrix}$	[0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.653]	-3.22237
Hump	$f_{22}(\boldsymbol{x})=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4 \quad x\in[-5,5]^2 \quad i=1,2$	[0.0898, -0.7126]	-1.0316

	Function	GO	Min
Matyas	$f_{23}(\boldsymbol{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2 \quad x \in [-10,10]^2 \quad i = 1,2$	[0, 0]	0
Michalewics 1	$f_{24}(\boldsymbol{x}) = -\sum_{i=1}^{2} \sin(x_i)(\sin(\frac{ix_i^2}{\pi}))^{2m}$ where $m = 10,\ x \in [0,\ \pi]^2 \ i = 1,2$	[2.203, 1.571]	-1.8013
Michalewics 2	$f_{25}(\boldsymbol{x}) = -\sum_{i=1}^{5} \sin(x_i)(\sin(\frac{ix_i^2}{\pi}))^{2m}$ where $m = 10, x \in [0,\ \pi]^5 \ i = 1,\cdots,5$	[2.20,1.57, 1.29,1.92, 1.72]	-4.6877
Michalewics 3	$f_{26}(\boldsymbol{x}) = -\sum_{i=1}^{10} \sin(x_i)(\sin(\frac{ix_i^2}{\pi}))^{2m}$ where $m = 10, x \in [0,\ \pi]^{10} \ i = 1,\cdots,10$	[2.20,1.57, 1.29,1.92, 1.72,1.57, 1.45,1.76, 1.66,1.27]	-9.6601
Needle-in-a-haystack	$f_{27}(\boldsymbol{x}) = -((\frac{a}{b + (x^2 + y^2)})^2 + (x^2 + y^2)^2)$ where $a = 3.0,\ \ b = 0.05,\ x \in [-5.12, 5.12]^2 \quad i = 1,2$	[0,0]	-3600
Shekel 1	$f_{28}(\boldsymbol{x}) = -\sum_{i=1}^{5}(\sum_{j=1}^{4}(x_j - C_{ij})^2 + \beta_i)^{-1} \quad i = 1,\cdots,4$	[4,4,4,4]	-10.1532
Shekel 2	$f_{29}(\boldsymbol{x}) = -\sum_{i=1}^{7}(\sum_{j=1}^{4}(x_j - C_{ij})^2 + \beta_i)^{-1} \quad i = 1,\cdots,4$	[4,4,4,4]	-10.4029
Shekel 3	$f_{30}(\boldsymbol{x}) = -\sum_{i=1}^{10}(\sum_{j=1}^{4}(x_j - C_{ij})^2 + \beta_i)^{-1}$ for Shekel function 1, 2, 3, where $\boldsymbol{\beta} = \frac{1}{10}(1,2,2,4,4,6,3,7,5,5)^T$ $\boldsymbol{C} = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix}$ $x \in [0,\ 10]^4 \quad i = 1,\cdots,4$	[4,4,4,4]	-10.5364

For instance, the Sphere function is convex, differentiable and has only one minimum. It is shaped like a deep valley, but the bottom is gentle. Hence the top and the bottom have a wide separation and the algorithm requires a large *G*. The Rastrigin function has a large number of local optima and the algorithm may be easily trapped at a local optimum. As a result, a large *PS* is beneficial. But since this function does not have a valley as deep as in a Sphere, it requires a smaller *G*. According to the no-free-lunch theorem, nothing comes free in effective optimization. So the properties of these functions should be taken into account when EE-EDA sets the parameters of the algorithm in these functions. Otherwise, the expected performance of all algorithms on some functions is exactly the same [88].

In Table 2.5, for the 30-*D* Sphere, *PS*=600 and *G*=2000, while for Rastrigin, *PS*=1500 and *G*=250. Besides, in the proportional selection, the fitness function may be modified as Equation (2.17). *X* is a proper positive number which keeps the $fitness_{\text{proportional}}$ greater than 0, and *f*(*x*) is the objective function value. This is because all functions correspond to minimization problems. In the tournament method, the 1st best solution is selected from 0.1∗*PS* random solutions. In addition, the size of the parent population for all selection models equals 0.5∗*PS*.

$$fitness_{\text{proportional}} = \frac{1}{X + f(x)} \tag{2.17}$$

2.3.1 Studies of 30-*D* and low-dimensional problems

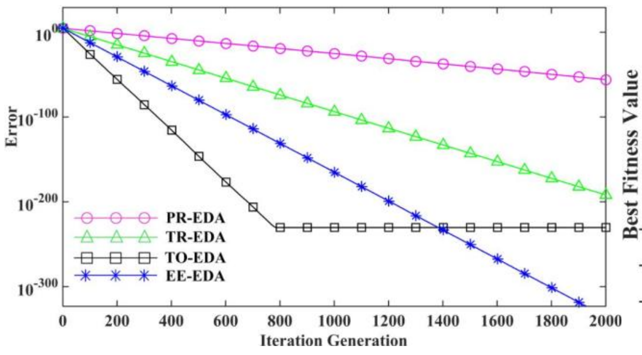
Table 2.5 presents the test results of EDA with different selection methods for 30-*D* problems (EE denotes extreme elitism selection; TR denotes truncation; TO denotes tournament; PR denotes proportional selection). It is seen that EE-EDA achieves good performance for most problems. Although the tournament and proportional selection methods can exploit the role of some leading best solutions in the evolution, this behaviour is random and unstable. Hence, the experimental results vary considerably. For instance, TO-EDA can reach the best solution of 4.94e-324, but the worst solution is 1.28e-11 for the Sphere. Also, the best values for Weierstrass and Griewanks both are 0 while the worst values are 2.31e-01 and 3.07e-02, respectively. Besides, if TO-EDA can achieve the global optimum, it can reach the optimum in a very early generation, even earlier than EE-EDA; for example, for Step (EE: AG=165, TO: AG=144), Weierstrass (EE: AG=136, TO: AG=121), and Griewanks (EE: AG=103, TO: AG=75). p

Table 2.5 Results of EDA with different selection methods for 30-*D* problems.

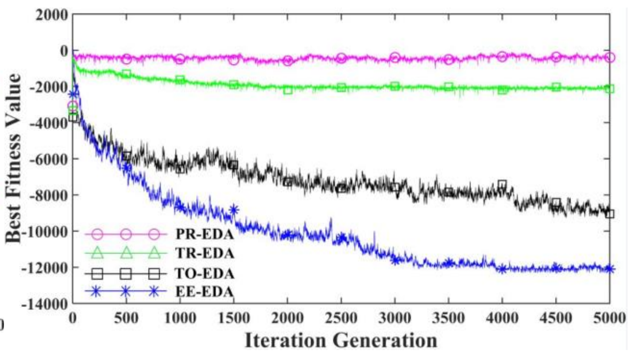
	Mean±Std	**Best**	**Worst**	**AG**	**Mean±Std**	**Best**	**Worst**	**AG**
$f_1(\boldsymbol{x})$ Step PS=400 $f_1(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_2(\boldsymbol{x})$ Quartic function with noise PS=1500 $f_2(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**185**	**9.35e-05±4.09e-05**	**3.10e-05**	**1.73e-04**	**5000**
TR	5.44e-07±7.48e-08	4.33e-07	7.35e-07	200	4.84e-04±1.92e-04	1.83e-04	9.19e-04	5000
TO	4.87e-13±2.67e-12	0	1.46e-11	144	8.53e-04±4.29e-04	2.70e-04	2.30e-03	5000
PR	2.97e+01±1.09e+01	1.49e+01	5.52e+01	200	1.98e-04±7.52e-05	4.21e-05	3.17e-04	5000
$f_3(\boldsymbol{x})$ Sphere PS=600 $f_3(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_4(\boldsymbol{x})$ Schwefel 2.22 PS=500 $f_4(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**1927**	**7.06e-162±0**	**5.30e-162**	**9.05e-162**	**2000**
TR	1.28e-192±0	2.01e-193	3.42e-192	2000	1.30e-90±4.68e-91	3.56e-91	2.85e-90	2000
TO	6.28e-13±2.42e-12	4.94e-324	1.28e-11	2000	1.30e-161±2.22e+162	9.01e-162	1.73e-161	2000
PR	2.92e-55±8.18e-55	1.07e-56	4.49e-54	2000	2.43e+03±1.13e+04	4.87e-14	6.15e+04	2000
$f_5(\boldsymbol{x})$ Rosenbrock (MIX) PS=500 $f_5(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_6(\boldsymbol{x})$ Schwefel 2.26 (MIX) PS=300 $f_6(\boldsymbol{x}_{\text{global optimum}}) = -12569.487$			
EE	**4.45e-04±9.30e-04**	6.51e-08	**3.77e-04**	**5000**	**-12565.53±2.82**	**-12568.15**	**-12554.83**	**5000**
TR	8.33e-04±1.89e-03	3.51e-08	9.66e-03	5000	-3182.08±443	-4842.91	-2589.55	5000
TO	5.60e-04±7.61e-04	7.48e-07	1.89e-03	5000	-7817.45±363	-8467.48	-7069.33	5000
PR	1.10e-03±2.13e-03	1.06e-006	8.59e-03	5000	-3161.52±321	-3750.77	-2623.97	5000
$f_7(\boldsymbol{x})$ Ackley PS=400 $f_7(\boldsymbol{x}_{\text{global optimum}}) = 0$								
EE	**2.66e-15±0**	**2.66e-15**	**2.66e-15**	**162**		**EE+MIX**		
TR	9.90e-09±1.18e-09	8.49e-09	1.06e-08	200	**0±0**	**0**	**0**	**175**
TO	2.68e-04±1.05e-03	2.66e-15	5.06e-03	200		MIX		
PR	7.56e+00±1.45e+00	5.63e+00	1.17e+01	200	3.74e-01±3.32e-02	4.32e-01	2.97e-01	200
$f_8(\boldsymbol{x})$ Rastrigin PS=1500 $f_8(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_9(\boldsymbol{x})$ Non-continuous Rastrigin PS=3000 $f_9(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**199**	**0±0**	**0**	**0**	**306**
TR	4.09e-12±5.21e-12	5.68e-14	1.61e-11	250	**0±0**	**0**	**0**	**320**
TO	5.16e+00±2.67e+00	1.99e+00	1.09e+01	250	1.70e+01±2.26e+00	1.20e+01	2.20e+01	350
PR	1.45e+0±7.81e+00	1.24e+02	1.56e+02	250	1.08e+02±7.63e+00	8.77e+01	1.19e+02	350
$f_{10}(\boldsymbol{x})$ Weierstrass PS=300 $f_{10}(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_{11}(\boldsymbol{x})$ Griewanks PS=400 $f_{11}(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**136**	**0±0**	**0**	**0**	**103**
TR	7.49e-03±7.50e-04	5.62e-03	8.64e-03	150	1.71e-10±3.98e-11	9.11e-11	2.77e-10	150
TO	7.76e-03±4.22e-02	0	2.31e-01	121	1.03e-03±5.61e-02	0	3.07e-02	75
PR	1.35e+01±1.59e+00	1.09e+00	1.68e+01	150	1.13e+00±2.88e-02	1.09e+00	1.20e+00	150

Fig. 2.8 shows the best fitness value convergence of some functions. In Fig. 2.8(d), TO-EDA has the fastest convergence rate for Griewanks and stops the iteration before the maximum allowed iteration generation. The tournament method selects the 1st best solution from some

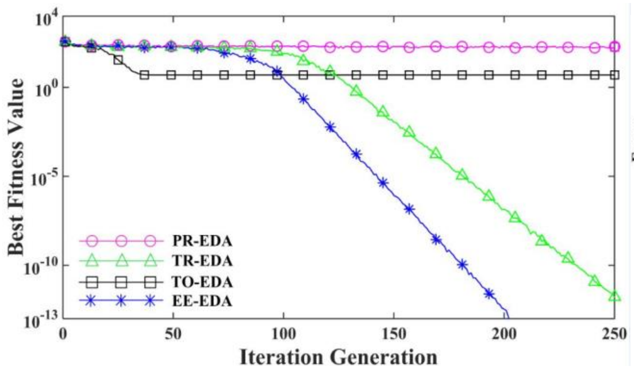
random solutions and repeats this 0.5∗*PS* times in every generation. Hence, it is likely to select more top solutions to the parent population than by the other selection methods. A dense presence of these solutions can make the mean of the Gaussian model approach the global minimum quickly. As a result, it can iterate just a few generations to reach the global minimum. But if these leading solutions are close to some local optimum, the algorithm will more likely get trapped there. It follows that TO-EDA sometimes can find a very good solution in an early generation, but sometimes it can quickly get trapped in a local optimum, as shown in Fig. 2.8(a) and (c).



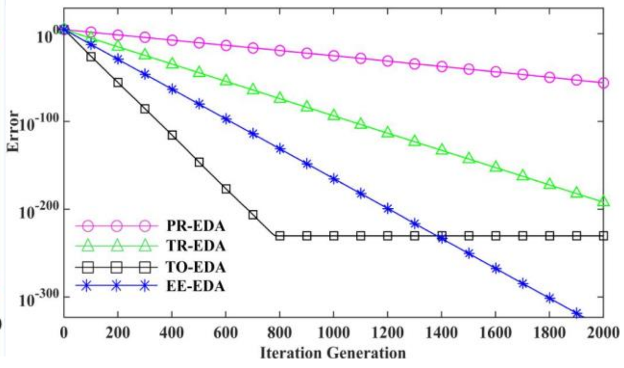
(a) $f_3(x)$ Sphere



(b) $f_6(x)$ Schwefel 2.26



(c) $f_8(x)$ Rastrigin



(d) $f_{11}(x)$ Griewanks

Figure 2.8 The best fitness convergence curves of some 30-*D* functions (1).

For EE-EDA, if these top are close to some local optimum, the algorithm can still adjust its search direction because some ordinary promising solutions are selected in a stable manner into the parent population and can balance the influence of these solutions. Overall, EE-EDA outperforms EDA with other selection methods for these functions. However, in view of the no-free-lunch theorem, there is no search algorithm that can outperform all others for all problems. It means EDA with truncation, tournament or proportional selection must outperform EE-EDA in some other optimization problems. Actually, here appropriate parameters of *PS* and *G* are set for EE-EDA in each function because we focus on studying EE-EDA in this work, while these parameters may not be appropriate for EDA with other selection for these functions. For instance, if *G* is set large enough for function Rastrigin, TR-EDA can also find its global optimum. Besides, all these functions correspond to minimization problems; so, the new fitness functions have to be designed for proportional selection, which somehow affects the performance of PR-EDA. In solving maximization problems, PR-EDA may achieve better results.

Table 2.6 indicates a success rate of these algorithms for low-dimensional functions. The percentage 100 indicates that all 30 tests successfully obtained the global optimum and AG indicates the average generation at which EE-EDA found the global optimum. The *PS* of functions $f_{12}(x)$ through $f_{20}(x)$, $f_{22}(x)$, $f_{24}(x)$ through $f_{26}(x)$ is 200; for $f_{23}(x)$, $f_{29}(x)$ and $f_{30}(x)$ it is 500; for $f_{27}(x)$ it is 1200; and for $f_{22}(x)$ it is 2000. Besides, the test results for some modified EDA, DE and PSO are also shown here. These algorithms include EDA-DE [6], FEP [83], SaDE [84], CPSO-outer [85], MGBDE [86] and ELPSO [87]. Some of these algorithms have not been tested for low-dimensional functions, and the same *PS* and *G* as EE-EDA have been used. It is seen that EE-EDA provides good test results for most functions. However, according to no-free-lunch theorem, it cannot be concluded that EE-EDA is better than other algorithms, and one can only indicate that EE-EDA is suitable for optimizing some functions. For other functions, such as $f_{26}(x)$ (Michalewics 3), EE-EDA cannot find the global optimum, although various parameters have been used with it. But, SaDE is able to obtain the optimum of $f_{26}(x)$. In contrast, for function $f_{23}(x)$ (Matyas) or $f_{27}(x)$ (Needle-in-a-haystack) SaDE cannot provide good performance regardless of the parameters while EE-EDA is very effective in solving these two problems. It is found that EE-EDA usually can provide good performance for the problems whose global optima are located in a smooth bowl. Even if these problems have many bowls (local minima), EE-EDA can still find the global optima of them. This situation is reasonable in light of the no-free-lunch theorem, since a general-purpose, universal optimization strategy is impossible theoretically. If an optimization algorithm has good performance on average for one class of problems then it may have poor performance on average over the remaining problems [89].

Table 2.6 Success rate of low-dimensional problems (TW indicates totally wining).

	EE (AG)	TR	TO	PR	FEP	MI MIC$_C$	EGN A$_{BIC}$	MIXM VGD	EDA-DE	Sa DE	MGB DE	CPSO -outer	EL PSO
$f_{12}(x)$	100(40)	0	93.3	96.7	0	0	0	100	0	0	0	100	0
$f_{13}(x)$	100(12)	100	100	96.7	0	0	0	100	30	86.7	100	100	100
$f_{14}(x)$	100(13)	100	100	100	0	0	0	100	0	0	100	100	100
$f_{15}(x)$	100(16)	0	100	96.7	0	0	0	100	0	0	100	100	100
$f_{16}(x)$	100(25)	0	100	86.7	0	0	0	100	0	0	0	100	0
$f_{17}(x)$	100(11)	100	93.3	3.3	96.7	100	100	100	100	100	100	100	100
$f_{18}(x)$	100(14)	0	100	0	13.3	100	100	100	40	100	100	100	100
$f_{19}(x)$	100(12)	93.3	100	100	83.3	60	100	100	100	100	100	100	100
$f_{20}(x)$	100(6)	26.7	100	0	3.3	0	93.3	100	0	100	100	100	0
$f_{21}(x)$	100(27)	100	43.3	0	0	0	0	0	0	60	40	46.7	0
$f_{22}(x)$	100(7)	93.3	100	10	43.3	66.7	100	100	0	100	100	100	100
$f_{23}(x)$	100(447)	0	96.7	0	0	0	0	100	0	0	0	100	100
$f_{24}(x)$	100(6)	100	100	100	30	0	80	100	0	100	100	100	0
$f_{25}(x)$	16.7	0	0	13.3	0	0	0	16.7	0	100	60	83.3	0
$f_{26}(x)$	0	0	0	0	0	0	0	0	0	100	3.3	0	0
$f_{27}(x)$	100(139)	0	0	0	0	0	13.3	0	0	0	0	100	96.7
$f_{28}(x)$	100(45)	100	90	0	0	0	0	0	0	100	0	100	0
$f_{29}(x)$	100(15)	100	100	0	0	0	0	100	0	100	70	100	0
$f_{30}(x)$	100(16)	100	100	3.3	0	0	0	100	56.7	100	100	100	0
TW	17	8	10	3	0	2	4	14	2	11	10	16	8

2.3.2 Special cases studies

Define

$$f_5(x)\text{: offspring\_individual}_{ij} = 1.2 * \mu_i + (5\times 10^{-10}) * (b_i - a_i) * \text{Cauchy}(1, PS) \quad (2.18)$$

$$f_6(x)\text{: offspring\_individual}_{ij} = \mu_i + 10^{-3} * (b_i - a_i) * \text{Cauchy}(1, PS) \ (i = 1, \cdots, D;\ j = 1, \cdots, PS) \quad (2.19)$$

For $f_5(x)$, $f_6(x)$ and $f_7(x)$, the mixed probabilistic model of Gaussian and Cauchy distribution is adopted (in Table 2.5 MIX is noted along with the function name). Generally, Cauchy distribution has a longer search step size than the Gaussian distribution [83]. For some optimization problems, a mixed probabilistic model may provide better performance than the single model. However, if a single Gaussian or Cauchy model can produce better results, it is not necessary to use a mixed model, because a mixed model makes the program complex and leads to increased computing cost. Besides, Cauchy mutation usually generates considerable oscillations during convergence as shown in Fig. 2.8(b) and sometimes may lead to poor convergence. In Table 2.6，neither FEP with Cauchy model nor EDA-DE with mixed model generated good test results for most of these functions. But Cauchy model can help $f_5(x)$, $f_6(x)$ and $f_7(x)$ achieve better solutions.

In Equations (2.18) and (2.19), μ_i is the mean of one problem variable, which is obtained through extreme elitism selection and kept the same for every offspring for one variable; $(b_i$-$a_i)*$Cauchy(1, PS) generates an $1\times PS$ matrix with individuals in the range $[a_i, b_i]$ (the scale parameter = 1) and these individuals follow the Cauchy distribution. The global minimum of $f_5(x)$ (Rosenbrock) lies in a narrow and parabolic valley. It is easy to find this valley, but difficult to converge to the global minimum. After the algorithm finds the valley, it needs a Cauchy mutation to extend the search space to exploit better solutions in the valley. But the coefficient of

Cauchy mutation must be very small (say, 5×10^{-10}) because if it is large, the search may easily escape from the narrow valley. Function $f_6(x)$ has many deep local optima far away from the global optimum, so it needs a rather wide Cauchy search (the coefficient is 10^{-3}) to help the algorithm find better solutions if the search is trapped in some local optima.

Furthermore, EE-EDA obtains the solution 2.66e-15±0 for the function $f_7(x)$, but cannot reach 0. This function has many local optima and a very salient valley. Hence the search is separated into two steps. The first step adopts EE-EDA to find the valley (for 30-D, before 170th generation; 100-D: 360th; 200-D: 610th) and the second step uses a mixed model ($0.8*\mu_i+0.001*$Cauchy) to find the global optimum. If the algorithm only consists of the second step, it cannot reach the global optimum. The algorithm MIX faces difficulty in finding the valley in the beginning and finally gets trapped in some local optima. From these special case studies, it is seen that the no-free-lunch situation is ubiquitous in the field of optimization. In other words, we have to modify the algorithm for some problems. The only way that one algorithm can outperform another is by specializing it to the structure of the specific problem [90].

2.3.3 The test results of 100-*D* and 200-*D* problems

From Tables 2.7 and 2.8 it is seen that the performance of EE-EDA does not deteriorate as the problem dimension increases. It can find the global optimum for the functions $f_1(x)$, $f_3(x)$, $f_7(x)$, $f_8(x)$, $f_9(x)$, $f_{10}(x)$ and $f_{11}(x)$ with 100 and 200 dimensions (the algorithm for $f_7(x)$ is EE+MIX). It is known that the volume of the search space grows exponentially with the dimensionality; so, more function evaluations are beneficial. Here *G* and *PS* are added for high-dimensional problems. Increasing *PS* can improve the frequency of finding some high quality solutions while EE-EDA can adequately take advantage of this feature. The high quality solutions are those solutions that are distributed around the global optimum.

Table 2.7 Results of EDA with different selection methods for 100-*D* problems.

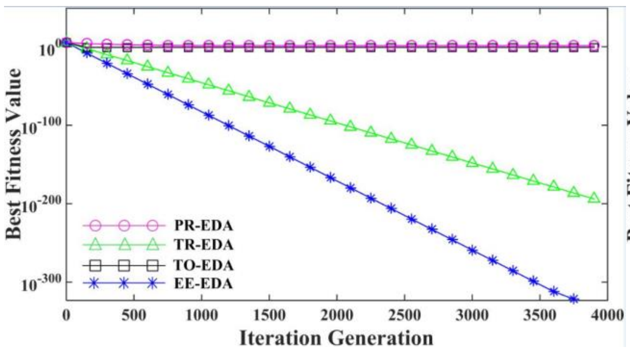
	Mean±Std	**Best**	**Worst**	**AG**	**Mean±Std**	**Best**	**Worst**	**AG**
$f_1(x)$ Step *PS*=650 $f_1(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_2(x)$ Quartic function with noise *PS*=2000 $f_2(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**436**	**3.17e-04±6.48e-05**	**1.99e-04**	**4.33e-04**	**5000**
TR	4.67e-08±4.70e-09	3.87e-08	5.54e-08	450	2.17e-03±4.74e-04	1.44e-03	3.52e-03	5000
TO	3.55e-01±7.48e-01	1.42e-11	3.60e+00	450	1.09e-02±6.58e-03	2.89e-03	3.59e-02	5000
PR	4.23e+02±6.23e+01	3.03e+02	5.42e+02	450	7.17e-04±2.10e-04	3.59e-04	1.16e-03	5000
$f_2(x)$ Sphere *PS*=650 $f_3(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_4(x)$ Schwefel 2.22 *PS*=600 $f_4(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**3930**	**2.34e-161±2.22e-162**	**1.99e-162**	**2.75e-161**	**5000**
TR	9.25e-200±0	3.76e-200	2.22e-199	4000	3.22e-118±8.16e-119	1.93e-118	5.04e-118	5000
TO	2.72e+00±5.26e-00	1.22e-02	2.23e+01	4000	2.72e-02±3.65e-02	5.04e-09	1.40e-01	5000
PR	7.81e+00±8.67e-01	2.66e+00	3.96e+02	4000	1.18e+39 ±6.33e+39	1.36e+02	3.47e+40	5000
$f_5(x)$ Rosenbrock *PS*=500 $f_5(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_6(x)$ Schwefel 2.26 *PS*=2500 $f_6(\boldsymbol{x}_{\text{global optimum}}) = -41898.29$			
EE	**4.17e-04±9.20e-04**	**1.01e-07**	4.79e-03	**10000**	**-41832.95±3.59**	-41840.29	**-41827.05**	5000
TR	5.53e-04±1.01e-03	2.17e-06	4.53e-03	10000	-7299.67±341.53	-8203.60	-7043.26	5000
TO	1.50e-03±2.07e-04	1.18e-03	1.63e-03	10000	-41610.55±138	-41857.76	-41540.58	5000
PR	1.56e-03±2.74e-03	1.66e-05	9.02e-03	10000	-6845.17±733.25	-9279.66	-5677.47	5000
$f_7(x)$ Ackley *PS*=500 $f_7(\boldsymbol{x}_{\text{global optimum}}) = 0$								
EE	**6.22e-15±0**	**6.22e-15**	**6.22e-15**	335	**EE+MIX**			
TR	7.43e-08±5.93e-09	6.11e-08	9.05e-08	350	**0±0**	**0**	**0**	**372**
TO	4.61e-01±3.41e-01	2.63e-03	1.28e+00	350	MIX			
PR	1.31e+01±6.75e-01	1.19e+01	1.43e+01	350	9.33e-01±4.49e-02	1.01e-01	8.66e-01	450
$f_8(x)$ Rastrigin *PS*=2500 $f_8(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_9(x)$ Non-continuous Rastrigin *PS*=4500 $f_9(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**407**	**0±0**	**0**	**0**	**551**
TR	8.53e-11±4.25e-11	2.92e-11	2.12e-10	450	4.91e-15±1.11e-14	0	4.80e-14	595
TO	3.46e+01±5.97e+00	2.41e+01	5.07e+01	450	6.01e+01±6.33e+00	4.90e+01	7.30e+01	600
PR	7.44e+02±2.03e+01	6.90e+02	7.72e+02	450	6.78e+02 ±1.96e+01	6.25e+02	7.22e+02	600
$f_{10}(x)$ Weierstrass *PS*=500 $f_{10}(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_{11}(x)$ Griewanks *PS*=600 $f_{11}(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**321**	**0±0**	**0**	**0**	**222**
TR	3.61e-03±2.69e-04	2.98e-03	4.13e-03	350	1.43e-08±1.41e-09	1.14e-08	1.76e-08	250
TO	4.19e-01±4.18e-01	3.69e-03	1.82e+00	350	2.52e-01±2.35e-01	3.62e-03	8.92e-01	250
PR	6.73e+01±4.38e+00	5.71e+01	7.39e+01	350	8.29e+00 ±1.52e-01	5.74e+00	1.23e+01	250

Table 2.8 Results of EDA with different selection models for 200-*D* problems.

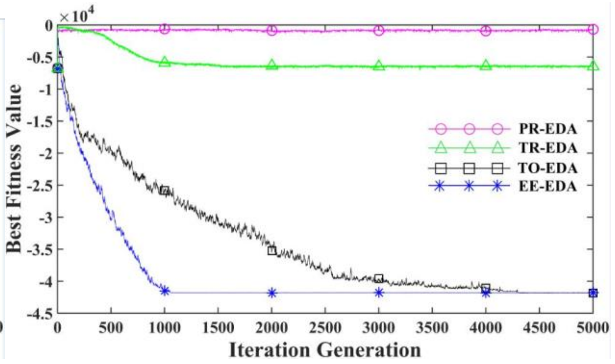
	Mean±Std	**Best**	**Worst**	**AG**	**Mean±Std**	**Best**	**Worst**	**AG**
$f_1(x)$ Step *PS*=600 $f_1(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_2(x)$ Quartic function with noise *PS*=2500 $f_2(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**695**	**5.64e-04±1.46e-04**	**3.66e-04**	**6.42e-04**	**5000**
TR	1.85e-09±1.11e-10	1.68e-09	2.05e-09	750	4.08e-03±6.58e-04	3.03e-04	5.37e-03	5000
TO	1.25e+01±6.95e+00	4.45e+00	2.80e+01	750	6.94e-02±3.11e-02	3.11e-02	1.16e-01	5000
PR	1.50e+03±1.08e+02	1.32e+03	1.82e+03	750	1.39e-03±3.62e-04	1.03e-03	1.99e-03	5000
$f_3(x)$ Sphere *PS*=700 $f_3(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_4(x)$ Schwefel 2.22 *PS*=700 $f_4(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**8902**	**2.90e-160±6.29-e162**	2.79e-162	3.00e-160	**6000**
TR	5.34e-322±0	4.74e-322	5.68e-322	10000	6.59e-98 ±1.40e-98	3.67e-98	9.18e-98	6000
TO	8.68e+01±7.95e+01	1.26e+01	3.45e+02	10000	1.23e+00 ±9.37e-01	1.61e-01	4.17e+00	6000
PR	1.12e+04±3.01e+03	5.64e+03	1.70e+04	10000	3.74e+91±2.04e+92	5.45e+02	1.12e+93	6000
$f_5(x)$ Rosenbrock *PS*=600 $f_5(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_6(x)$ Schwefel 2.26 *PS*=3000 $f_6(\boldsymbol{x}_{\text{global optimum}}) = -86796.58$			
EE	9.18e-04±1.09e-03	9.57e-06	6.00e-03	10000	**-83332.42±3.22**	**-83280.91**	-83154.82	20000
TR	**5.76e-04±9.55e-04**	**1.93e-06**	**2.96e-03**	**10000**	-12641.36±88.39	-12781.17	-12461.48	20000
TO	1.38e-03±1.63e-03	7.07e-05	4.04e-03	10000	-67122.90±373	-67763.27	-67311.32	20000
PR	2.16e-03±3.78e-03	6.43e-08	1.68e-02	10000	-9122.90±373	-9763.27	-8811.32	20000
$f_7(x)$ Ackley *PS*=700 $f_7(\boldsymbol{x}_{\text{global optimum}}) = 0$								
EE	**6.22e-15±0**	**6.22e-15**	**6.22e-15**	567	**EE+MIX**			
TR	1.45e-09±7.67e-11	1.22e-09	1.60e-09	600	**0±0**	**0**	**0**	**626**
TO	1.47e+00 ±3.02e-01	1.07e+00	2.39e+00	600	MIX			
PR	1.40e+01 ±4.95e-01	1.29e+01	1.50e+01	600	1.25e+01±5.56e-02	1.33e+00	1.13e+00	700
$f_8(x)$ Rastrigin *PS*=3500 $f_8(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_9(x)$ Non-continuous Rastrigin *PS*=6000 $f_9(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**610**	**0±0**	**0**	**0**	**800**
TR	8.45e-11±3.43e-11	3.55e-11	2.19e-10	650	3.47e-14±3.59e-14	0	1.69e-13	850
TO	1.02e+02±9.67e+00	8.13e+01	1.19e+02	650	1.67e+00 ±4.11e-01	1.10e+00	2.56e+00	850
PR	1.68e+03±2.28e+01	1.63e+03	1.72e+03	650	1.66e+03±3.40e+03	1.58e+03	1.72e+03	850
$f_{10}(x)$ Weierstrass *PS*=700 $f_{10}(\boldsymbol{x}_{\text{global optimum}}) = 0$					$f_{11}(x)$ Griewanks *PS*=700 $f_{11}(\boldsymbol{x}_{\text{global optimum}}) = 0$			
EE	**0±0**	**0**	**0**	**519**	**0±0**	**0**	**0**	**339**
TR	2.46e-03±1.11e-04	2.22e-03	2.75e-03	550	3.50e-08±3.30e-09	2.79e-08	4.13e-08	350
TO	3.85e+00±1.15e+00	1.51e+00	6.04e+00	550	1.52e+00 ±3.99e-01	1.09e+00	3.27e+00	350
PR	1.59e+02±5.16e+00	1.48e+03	1.67e+02	550	7.20e+01±1.15e+00	5.44e+01	9.95e+01	350

EE-EDA makes a few top solutions have a higher proportion in the parent population to push the mean vector of the Gaussian model towards the regions where these few top solutions locate.

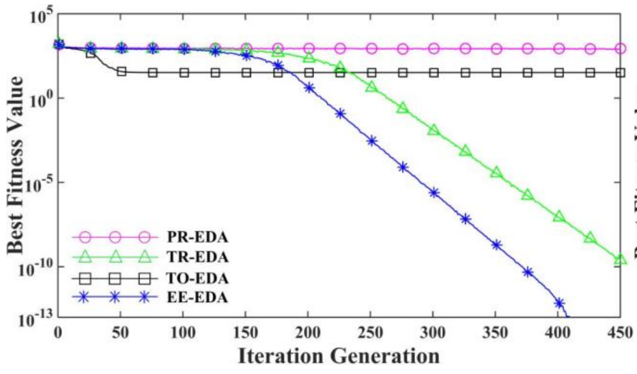
These top solutions are more likely to be the high quality solutions than other ordinary solutions. If there are more high quality solutions in the parent population, the algorithm can find the global optimum at a greater probability. Even if these leading solutions are far away from the global optimum, their effect can be balanced by other leading solutions as EE-EDA does not use just one or two top solutions, but it exploits five top solutions. For other functions, although EE-EDA cannot reach the global optimum, the test results do not have a large difference for 100 and 200 dimensions. Here, for $f_6(x)$, the combination of top in the elite corps is changed to 250, 200, 150, 100 and 50 when D=100; and to 450, 350, 250, 150 and 50 when D=200, because high-dimensional $f_6(x)$ has more local minima and is more complex. If the algorithm can find some high quality solutions, but their percentage in the parent population is low, the mean vector of the Gaussian model cannot approach these high quality solutions and the algorithm may miss the opportunity of obtaining better solutions. If the percentage is high, however, the algorithm may utilize the opportunity. Moreover, even if these leading solutions are close to some local optima, a large Cauchy mutation can make the search escape from these local optima. The oscillations shown in Fig. 2.9(b) and 2.10(b) indicate repeated escapes from some local minima.



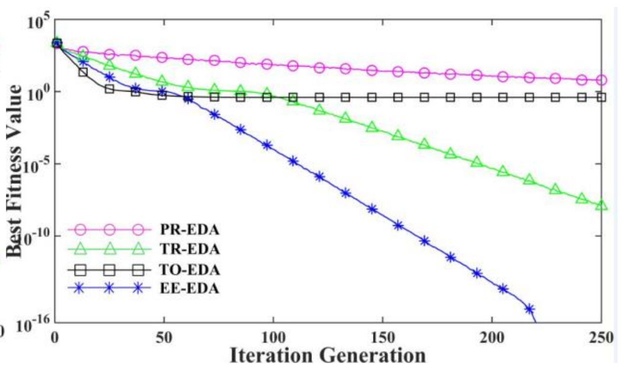
(a) $f_3(x)$ Sphere



(b) $f_6(x)$ Schwefel 2.26

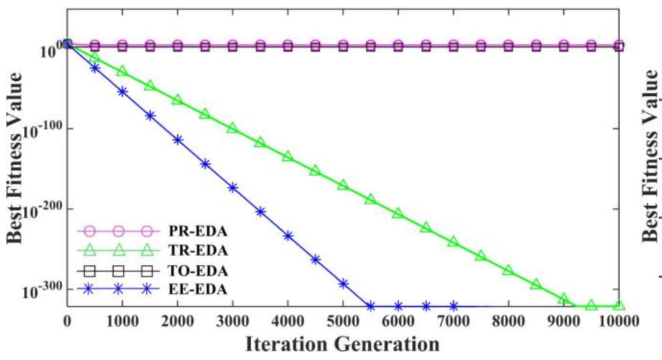


(c) $f_8(x)$ Rastrigin

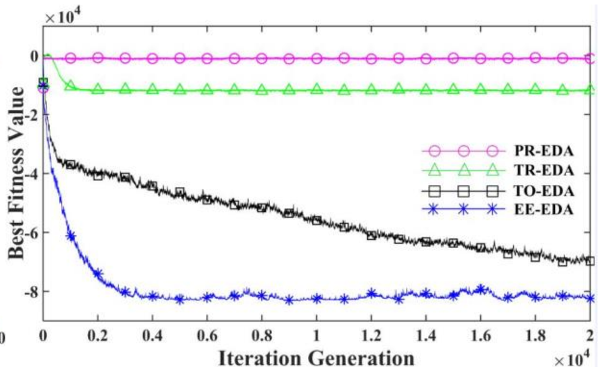


(d) $f_{11}(x)$ Griewanks

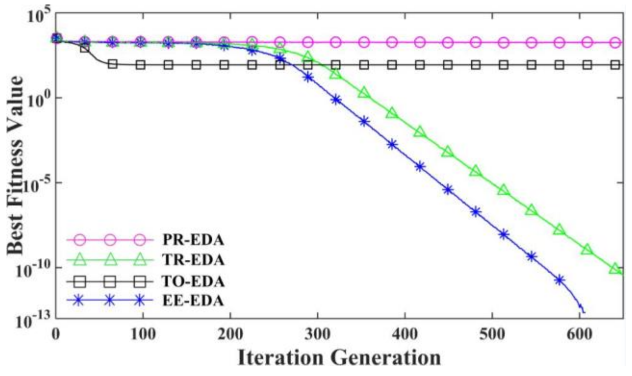
Figure 2.9 The best fitness convergence curves of some 100-*D* functions (1).



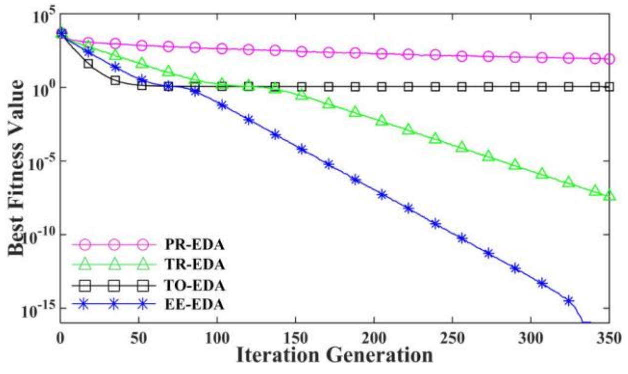
(a) $f_3(x)$ Sphere



(b) $f_6(x)$ Schwefel 2.26



(c) $f_8(x)$ Rastrigin



(d) $f_{11}(x)$ Griewanks

Figure 2.10 The best fitness convergence curves of some 200-*D* functions (1).

2.4 The experimental results of other algorithms for high-dimensional problems

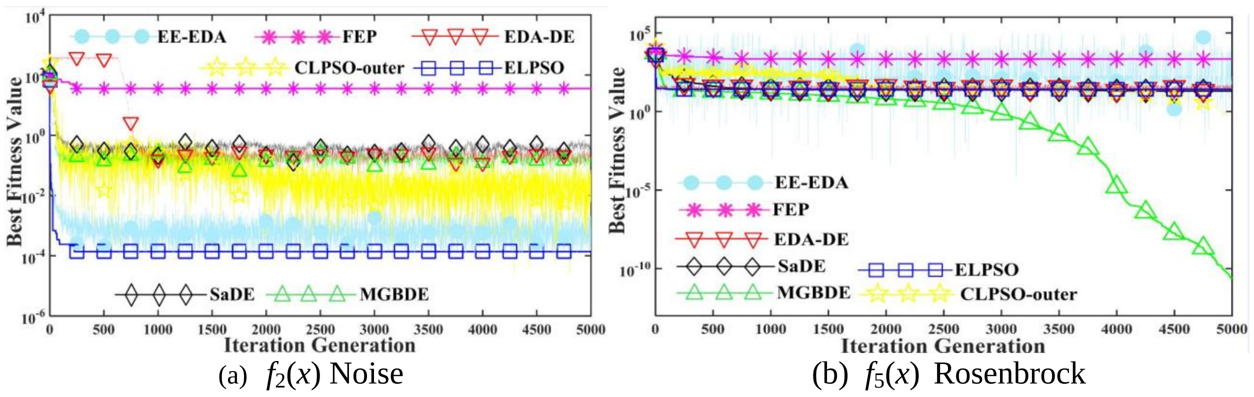
Table 2.9 shows the test results of other optimization strategies for 30-*D* problems. It is seen that some algorithms are particularly suitable for some functions. For instance, only ELPSO can find the global optimum of $f_4(x)$ and EDA-DE obtains the global optimum of $f_6(x)$. No algorithm can find the global optimum of $f_2(x)$ or $f_5(x)$. Also, $f_2(x)$ is difficult to handle since it has a noise term *random* [0, 1).

Table 2.9 Results of other optimization strategies for 30-*D* problems.

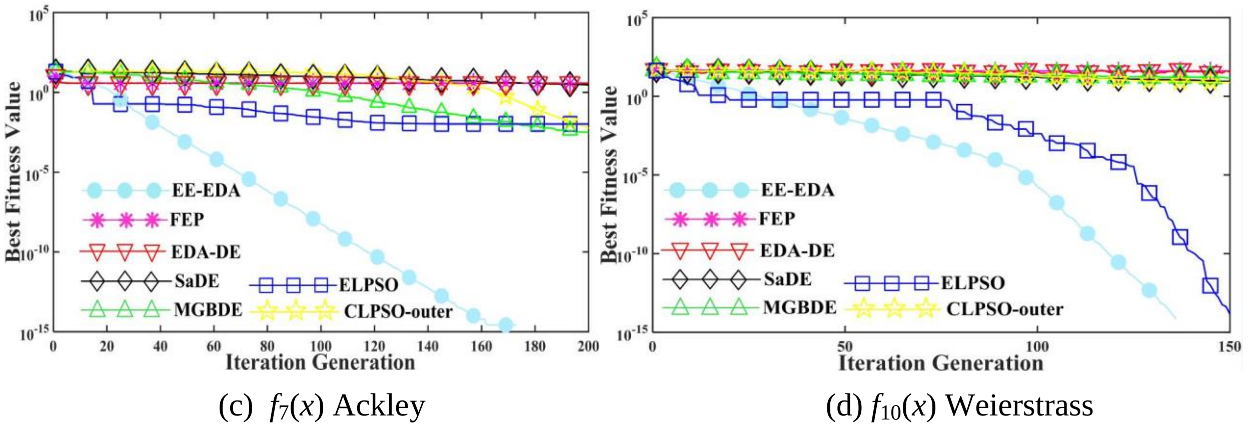
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
	$f_1(x)$ Step		$f_2(x)$ Noise		$f_3(x)$ Sphere		$f_4(x)$ Schwefel 2.22	
Algorithm	$f_1(\boldsymbol{x}_{\text{global optimum}})=0$		$f_2(\boldsymbol{x}_{\text{global optimum}})=0$		$f_3(\boldsymbol{x}_{\text{global optimum}})=0$		$f_4(\boldsymbol{x}_{\text{global optimum}})=0$	
EE-EDA	**0**	**0**	**1.01e-04**	**3.16e-05**	**0**	**0**	7.06e-162	0
MIMIC$_C$	4.73e+01	2.40e+00	4.92e-01	9.57e-02	2.96e-49	1.71e-49	5.26e+00	2.07e-01
ENGA$_{BIC}$	1.87e+00	6.06e-02	1.82e-01	2.66e-02	2.10e-01	1.94e-02	1.89e+00	1.43e-01
MIXMVGD	4.59e+01	9.52e+00	2.82e+00	3.48e-01	5.49e-50	4.26e-50	4.89e-44	1.24e-44
FEP	**0**	**0**	7.62e-03	2.60e-03	5.70e-04	1.30e-04	8.10e-03	7.70e-04
EDA-DE	**0**	**0**	5.42e-03	1.79e-03	5.22e-109	1.28e-108	8.26e-23	9.52e-23
SaDE	**0**	**0**	3.15e-03	7.50e-04	1.18e-28	1.06e-28	1.00e-23	9.70e-24
MGBDE	**0**	**0**	2.14e-03	1.08e-03	8.79e-68	5.21e-69	8.50e-41	3.38e-41
CPSO-outer	2.50e+02	7.11e+01	1.54e-04	1.10e-04	9.48e-71	5.13e-70	1.50e+01	7.07e+00
ELPSO	1.36e+00	5.57e-02	9.87e-04	8.05e-04	5.24e-08	1.64e-08	**0**	**0**
	$f_5(x)$ Rosenbrock		$f_6(x)$ Schwefel 2.26		$f_7(x)$ Ackley		$f_8(x)$ Rastrigin	
Algorithm	$f_5(\boldsymbol{x}_{\text{global optimum}})=0$		$f_2(\boldsymbol{x}_{\text{global optimum}})=-12569.487$		$f_3(\boldsymbol{x}_{\text{global optimum}})=0$		$f_4(\boldsymbol{x}_{\text{global optimum}})=0$	
EE-EDA	4.45e-04	9.30e-04	-12565.5	2.82e+00	**0**	**0**	**0**	**0**
MIMIC$_C$	2.66e+01	6.37e-01	-4194.26	1.56e+02	2.41e-05	1.05e-04	1.14e+02	5.58e+00
ENGA$_{BIC}$	4.84e+01	1.18e+00	-6123.63	1.26e+02	5.76e-01	5.85e-03	8.39e+01	2.65e+01
MIXMVGD	2.36e+01	1.53e+00	-4500.41	2.42e+01	2.75e-01	5.57e-02	2.59e+02	1.88e+01
FEP	5.06e+00	5.87e+00	-12554.5	5.26e+01	1.80e-02	2.10e-03	4.60e-02	1.20e-02
EDA-DE	3.85e-10	1.19e-09	**-12569.5**	**0**	6.43e-15	1.78e-15	2.14e-81	5.72e-81
SaDE	2.17e+01	3.00e-01	-12569.5	7.00e-12	7.70e-15	1.40e-15	**0**	**0**
MGBDE	**2.31e-12**	**7.04e-09**	-12569.5	1.09e-12	7.69e-15	0	3.98e+00	2.98e+00
CPSO-outer	1.10e+00	6.65e-01	-12418.5	1.42e+03	5.03e-15	2.90e-15	5.00e+01	2.49e+01
ELPSO	5.82e+00	1.38e+00	-5443.84	1.17e+02	2.78e-01	8.16e-02	8.64e+00	4.19e+00

	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
	$f_9(x)$Non-con Rastrigin		$f_{10}(x)$ Weierstrass		$f_{11}(x)$ Griewanks	
Algorithm	$f_9(\boldsymbol{x}_{\text{global optimum}})=0$		$f_{10}(\boldsymbol{x}_{\text{global optimum}})=0$		$f_{11}(\boldsymbol{x}_{\text{global optimum}})=0$	
EE-EDA	**0**	**0**	**0**	**0**	**0**	**0**
MIMIC$_C$	7.68e+01	5.24e+00	3.64e+01	1.24e+00	1.34e-05	2.35e-06
ENGA$_{BIC}$	1.13e+02	6.77e+00	2.76e+01	9.06e-01	8.64e-03	3.77e-04
MIXMVGD	1.93e+02	1.31e+01	4.74e+00	1.04e+00	2.04e-01	2.12e-01
FEP	2.27e+02	1.78e+01	4.00e+01	1.79e-01	1.6e-02	2.2e-02
EDA-DE	**0**	**0**	2.49e-14	5.02e-15	**0**	**0**
SaDE	**0**	**0**	**0**	**0**	**0**	**0**
MGBDE	6.50e+00	7.07e-01	**0**	**0**	**0**	**0**
CPSO-outer	3.52e+01	7.52e+01	1.91e+00	1.38e+00	1.52e-02	2.22e-02
ELPSO	1.13e+00	1.47e+00	2.52e-03	3.56e-03	2.75e-04	1.23e-04

Fig. 2.11(a) displays the best fitness convergence curve of these algorithms for this function, and there are many oscillations due to the noise term. Function $f_5(x)$ is also challenging as its global optimum lies in a narrow, parabolic valley. Fig. 2.11(b) shows that EE-EDA has high oscillations in the evolution. This is because the probabilistic model for $f_5(x)$ contains Cauchy mutation. Fig. 2.11(c) and 11(d) demonstrated that EE-EDA can find the global optimum for $f_7(x)$ and $f_{10}(x)$ before the 200th and 150th generations, so G equals 200 and 150, respectively. However, other algorithms, like EDA-DE, SaDE, MGBDE, and so on needed more iteration generations (EDA-DE and MGDE: G=5000×D). As a result, in Fig. 2.11, these algorithms converged to some fitness values that were not as good as the values in Table 2.9. The values in Table 2.9 come from the literature [6, 8, 83-87] where EDA-DE, SaDE and MGBDE have been experimented with a larger G.



(a) $f_2(x)$ Noise (b) $f_5(x)$ Rosenbrock



(c) $f_7(x)$ Ackley (d) $f_{10}(x)$ Weierstrass

Figure 2.11 The best fitness convergence curves of some 30-*D* functions (2).

Tables 2.10 and 2.11 indicate the test results of 100-*D* and 200-*D* functions. Fig. 2.12 and 2.13 show the convergence curves of the best fitness value of some 100-*D* and 200-*D* functions. It is seen that EDA with extreme selection, based on bivariate model (MIMIC$_C$) [28], multivariate model (EGNA$_{BIC}$) [29], and mixtures of Multivariate Gaussian distributions (MIXMVGD) [68], can generate small solutions for some functions, but they cannot find the global optimum for these minimization problems.

Table 2.10 Results of other optimization strategies for 100-*D* problems.

	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
	$f_1(x)$ Step		$f_2(x)$ Noise		$f_3(x)$ Sphere		$f_4(x)$ Schwefel 2.22	
Algorithm	$f_1(\boldsymbol{x}_{\text{global optimum}})=0$		$f_2(\boldsymbol{x}_{\text{global optimum}})=0$		$f_3(\boldsymbol{x}_{\text{global optimum}})=0$		$f_4(\boldsymbol{x}_{\text{global optimum}})=0$	
EE-EDA	**0**	**0**	3.17e-04	6.48e-05	**0**	**0**	**2.90e-160**	**6.29e-162**
MIMIC$_C$	4.23e+01	2.81e+00	1.37e+01	1.94e+00	1.72e+01	2.16e+00	3.81e+01	1.48e+00
ENGA$_{BIC}$	1.26e+01	2.90e-01	1.63e+01	3.59e+00	2.71e+00	7.40e-02	5.66e+01	8.66e+00
MIXMVGD	2.77e+02	3.99e+01	9.65e+01	8.76e+00	5.62e+04	8.69e+03	7.78e+01	5.92e+00
FEP	1.91e+03	3.49e +02	8.90e+02	9.66e+01	7.03e+04	2.41e+04	7.62e+01	1.27e+00
EDA-DE	1.43e+02	2.62e-01	1.30e-01	7.24e-02	3.42e-08	2.73e-08	9.25e+01	1.95e+00
SaDE	3.00e+02	1.20e+02	6.12e-01	3.14e-02	3.98e-13	2.93e-13	7.08e-11	3.58e-11
MGBDE	1.10e+00	7.98e-02	2.61e-01	6.15e-02	1.48e-55	5.14e-56	8.83e-37	6.42e-37
CPSO-outer	4.03e+02	2.83e+02	**7.82e-07**	**14.34e-07**	9.17e-44	1.29e-43	1.00e+01	1.41e+01
ELPSO	2.53e+01	5.85e+00	3.41e-03	4.50e-03	6.04e-05	1.86e-05	2.42e-01	2.91e-01
	$f_5(x)$Rosenbrock(MIX)		$f_6(x)$Schwefel2.26(MIX)		$f_7(x)$ Ackley9(+MIX)		$f_8(x)$ Rastrigin	
Algorithm	$f_5(\boldsymbol{x}_{\text{global optimum}})=0$		$f_6(\boldsymbol{x}_{\text{global optimum}})$=- 41898.29		$f_7(\boldsymbol{x}_{\text{global optimum}})=0$		$f_8(\boldsymbol{x}_{\text{global optimum}})=0$	
EE-EDA	**4.17e-04**	**9.20e-04**	-41832.95	3.59e+00	**0**	**0**	**0**	**0**
MIMIC$_C$	1.48e+03	1.02e+02	-19572.28	1.41e+03	3.60e+00	9.92e-02	6.00e+02	1.32e+01
ENGA$_{BIC}$	3.69e+02	1.24e+01	-31891.35	2.53e+03	5.76e-01	3.82e-03	2.53e+02	3.82e+01
MIXMVGD	2.06e+02	9.33e+00	-26981.65	5.23e+03	5.07e+00	2.69e-01	1.13e+03	8.00e+00
FEP	1.73e+04	1.06e+03	-22551.03	3.77e+03	1.92e+01	2.18e-01	1.28e+03	5.66e+00
EDA-DE	1.92e+01	7.88e-01	-17286.80	2.60e+02	3.67e+00	1.50e-01	1.43e+02	1.19e+00
SaDE	9.21e+01	2.13e-01	**-41898.12**	**2.13e-01**	6.44e+00	9.70e-01	4.70e+02	4.04e+00
MGBDE	4.32e+01	5.11e+00	-37264.40	9.08e+02	1.03e+00	2.54e-01	6.14e+02	3.86e+01
CPSO-outer	6.94e+01	1.22e+00	-31186.81	1.31e+09	4.25e+00	5.20e-03	3.15e+02	1.21e+02
ELPSO	9.80e+01	2.57e+00	-12592.32	1.09e+03	2.26e-01	8.67e-02	5.54e+00	1.16e+00
	$f_9(x)$ Non-con Rastrigin		$f_{10}(x)$ Weierstrass		$f_{11}(x)$ Griewanks			
Algorithm	$f_9(\boldsymbol{x}_{\text{global optimum}})=0$		$f_{10}(\boldsymbol{x}_{\text{global optimum}})=0$		$f_{11}(\boldsymbol{x}_{\text{global optimum}})=0$			
EE-EDA	**0**	**0**	**0**	**0**	**0**	**0**		
MIMIC$_C$	6.03e+02	3.19e+01	1.50e+02	1.34e+00	4.16e-01	5.47e-02		
ENGA$_{BIC}$	2.05e+02	1.03e+01	1.25e+02	1.40e+00	1.46e+00	2.36e-01		
MIXMVGD	1.03e+03	3.75e+01	2.67e+01	2.10e+00	5.74e+02	2.98e+01		
FEP	7.70e+04	1.40e+02	1.66e+02	7.19e+00	8.11e+02	5.03e+02		
EDA-DE	9.72e+01	1.03e+00	7.06e+01	8.60e+00	8.92e-01	2.75e-03		
SaDE	2.43e+02	5.32e+00	4.09e+01	5.59e+00	1.22e+02	3.04e+01		
MGBDE	4.43e+02	1.15e+02	9.82e+01	6.84e+00	1.43e+00	6.81e-02		
CPSO-outer	2.70e+02	6.64e+01	4.16e+01	7.66e+00	5.15e-01	7.28e-01		
ELPSO	9.37e+00	1.74e+01	**0**	**0**	2.61e-01	1.09e-01		

First, for these EDAs, when the offspring of one variable are sampled, it usually is affected by other variables. For instance, some variables have one or more parents in the Gaussian network in EGNA$_{BIC}$. The influence from other variables might reduce the effect of these few leading best solutions. However, EE-EDA does not need to consider this influence and it sampled the offspring, which can more likely approach these leading best solutions. This behaviour might be beneficial for some functions. However, for some other optimization functions like Summation cancellation, these probabilistic models with multiple interdependencies have led to better test results than the univariate and bivariate models (Chapter 8, Experimental results in function optimization with EDAs in continuous domain [9]). For the problem trap-5, the linkage learning is also important. The EDA by learning the linkage information between different variables has achieved better performance than the univariate models [23]. Therefore, the question of which probabilistic model is suitable is another situation of no-free-lunch theorem. Second, EGNA$_{BIC}$ did not run all the iteration generations for some 100-*D* or 200-*D* functions in the test. If the best fitness in the current generation was not considerably better than those in the previous 20 generations, the algorithm stopped running. The stop condition was set on since EGNA$_{BIC}$ would take much computing time if it ran all generations. The early stopping might affect its performance.

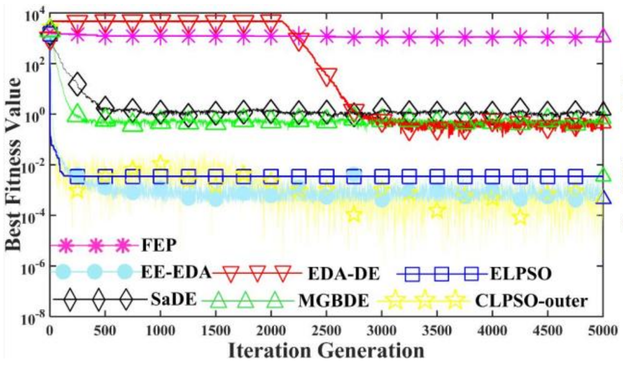
Table 2.11 Results of other optimization strategies for 200-*D* problems.

	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
	$f_1(x)$ Step		$f_2(x)$ Noise		$f_3(x)$ Sphere		$f_4(x)$ Schwefel 2.22	
Algorithm	$f_1(x_{\text{global optimum}})=0$		$f_2(x_{\text{global optimum}})=0$		$f_3(x_{\text{global optimum}})=0$		$f_4(x_{\text{global optimum}})=0$	
EE-EDA	**0**	**0**	5.64e-04	1.46e-04	**0**	**0**	**2.90e-160**	**6.29-e162**
MIMIC$_C$	1.19e+02	5.02e+00	8.41e+01	1.50e+01	7.36e+01	4.26e+00	3.49e+01	4.07e+00
ENGA$_{BIC}$	5.86e+02	1.75e+01	7.48e+01	5.68e+00	1.75e+01	2.93e+00	8.56e+02	1.98e+01
MIXMVGD	4.59e+03	1.07e+02	2.07e+02	3.99e+01	7.92e+04	3.549+03	5.13e+03	1.97e+01
FEP	4.86e+03	7.10e+02	5.21e+03	1.20e+02	1.72e+05	4.33e+04	2.48e+05	3.47e+04
EDA-DE	2.91e+02	1.15e+00	2.07e+01	3.99e+00	2.44e-17	1.56e-18	1.94e+02	2.42e+00
SaDE	5.75e+02	4.05e+02	1.35e+00	1.42e-01	2.08e-19	4.78e-20	3.56e-06	3.20e-06
MGBDE	5.60e+00	3.89e-01	3.23e-01	5.93e-03	4.49e-154	3.18e-15	1.96e-23	1.47e-23
CPSO-outer	1.14e+03	4.23e+02	**4.74e-07**	**6.41e-07**	1.50e+04	2.12e+03	4.00e+01	8.85e-01
ELPSO	1.32e+02	8.83e+01	1.20e-03	1.70e-03	6.82e+00	3.64e+00	8.57e+00	5.01e+00

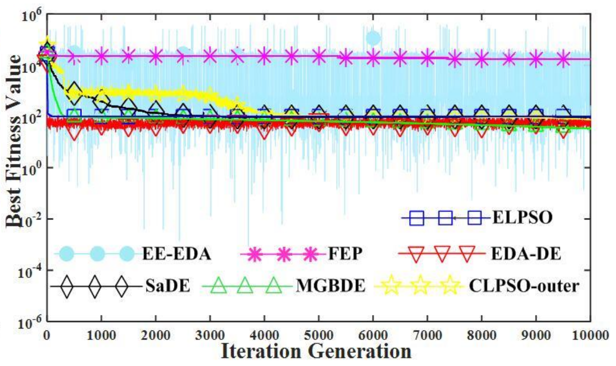
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
	$f_5(x)$ Rosenbrock(MIX)		$f_6(x)$ Schwefel 2.26(MIX)		$f_7(x)$ Ackley9(+MIX)		$f_8(x)$ Rastrigin	
Algorithm	$f_5(x_{\text{global optimum}})=0$		$f_6(x_{\text{global optimum}})=-\ 86796.58$		$f_7(x_{\text{global optimum}})=0$		$f_8(x_{\text{global optimum}})=0$	
EE-EDA	**9.18e-04**	**1.09e-03**	-83332.42	3.22e+00	**0**	**0**	**0**	**0**
MIMIC$_C$	3.21e+03	4.17e+02	-36703.53	1.21e+03	3.94e+00	1.84e-01	1.36e+03	5.87e+00
ENGA$_{BIC}$	2.76e+03	5.63e+02	-48932.23	2.56e+02	2.56e+00	6.32e-01	6.58e+02	6.89e+01
MIXMVGD	1.28e+04	8.02e+02	-16684.59	1.21e+03	1.93e+01	2.24e-02	8.98e+03	7.45e+02
FEP	5.02e+04	1.37e+03	-32237.78	5.64e+03	2.01e+01	1.11e-01	2.86e+03	6.48e+00
EDA-DE	3.63e+01	3.63e+01	-11310.86	7.02e+01	3.66e+00	1.90e-01	2.56e+02	2.31e+00
SaDE	2.34e+02	3.78e+00	**-83797.21**	**1.08e+00**	9.19e+00	9.00e-01	1.25e+03	8.29e+00
MGBDE	1.92e+02	4.85e+01	-60078.96	3.03e+03	6.43e-01	1.38e-01	1.56e+03	1.60e+02
CPSO-outer	1.68e+02	6.34e-01	-59158.73	1.25e+03	1.49e+01	4.79e+00	6.61e+02	7.33e+01
ELPSO	1.99e+02	3.26e+00	-16768.02	2.21e+03	5.95e-01	3.99e-01	1.74e+00	8.67e-01
	$f_9(x)$ Non-con Rastrigin		$f_{10}(x)$ Weierstra		$f_{11}(x)$ Griewanks			
Algorithm	$f_9(x_{\text{global optimum}})=0$		$f_{10}(x_{\text{global optimum}})=0$		$f_{11}(x_{\text{global optimum}})=0$			
EE-EDA	**0**	**0**	**0**	**0**	**0**	**0**		
MIMIC$_C$	1.39e+03	3.40e+01	3.14e+02	3.50e+00	6.38e-01	2.23e-03		
ENGA$_{BIC}$	1.56e+03	2.28e+01	2.49e+02	5.37e+00	7.56e+00	5.69e-01		
MIXMVGD	7.87e+03	3.59e+01	2.76e+02	1.02e+00	1.86e+03	7.67e+01		
FEP	2.89e+03	2.11e+01	3.47e+02	9.17e+00	1.66e+03	1.22e+03		
EDA-DE	1.97e+02	1.61e+00	1.09e+02	1.77e+00	9.50e-01	3.80e-02		
SaDE	9.84e+02	3.30e+00	1.19e+02	7.27e+00	5.53e+02	2.00e+01		
MGBDE	1.43e+03	1.73e+02	2.53e+02	1.07e+00	8.91e+00	2.92e+00		
CPSO-outer	9.69e+02	1.11e+02	1.19e+02	2.64e+01	5.00e+00	1.75e+00		
ELPSO	2.08e+01	2.50e+01	5.68e-13	3.04e-14	1.22e+00	8.90e-02		

For MIXMVGD, it is suitable for some low-dimensional functions (as seen in Table 2.6), but it cannot acquire good results for some high-dimensional functions. Moreover, the algorithm of MIXMVGD set more parameters like the number of clusters, the metrics of the distance used to cluster the solutions, whether normalized, the scaling factor for the covariance values, and so on, which increased its sensitivity and sometimes made it hard to find suitable parameters for some functions.

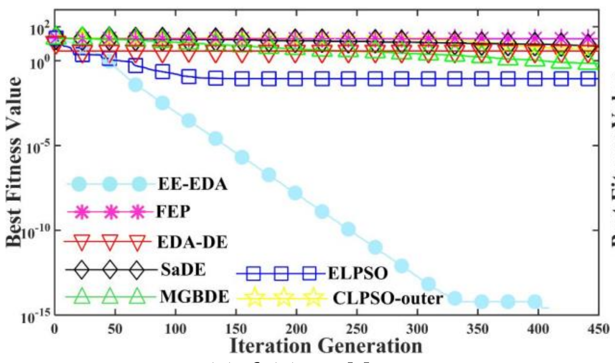
It should be mentioned that, except EE-EDA, most of the algorithms were not tested using 100-*D* or 200-*D* functions by their original researchers. The code of these algorithms for 100-*D* or 200-*D* problems was programmed in MATLAB, according to the flowcharts provided in the relevant literature [6, 83, 84, 85, 86, 87]. Besides, they used the same *PS* and *G* as EE-EDA. Some other parameters, such as initial *F* (scaling factor) and *CR* (crossover probability) in two modified DE algorithms and *w* (inertia weight), C_1 (cognitive acceleration coefficients) and C_2 (social acceleration coefficients) in two modified PSO algorithms, we kept the same as in 30-*D* problems. These operations might degrade the performance of these algorithms. FEP, EDA-DE, SaDE, MGBDE, and so on generally need a larger *G*, but here *G* for EE-EDA is not very large for some functions.



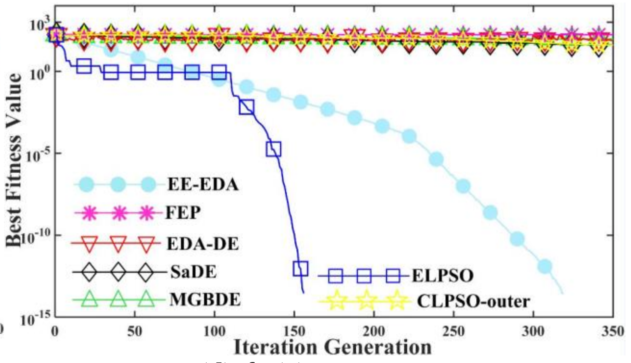
(a) $f_2(x)$ Noise



(b) $f_5(x)$ Rosenbrock

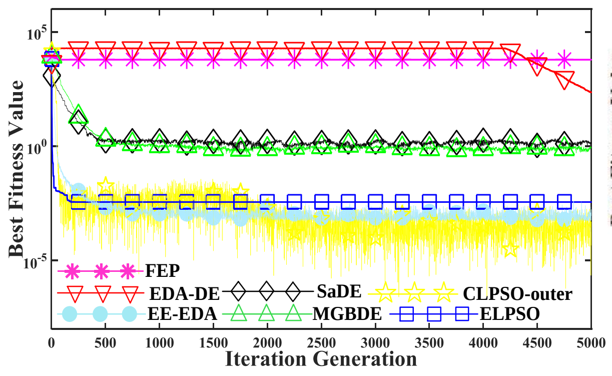


(c) $f_7(x)$ Ackley

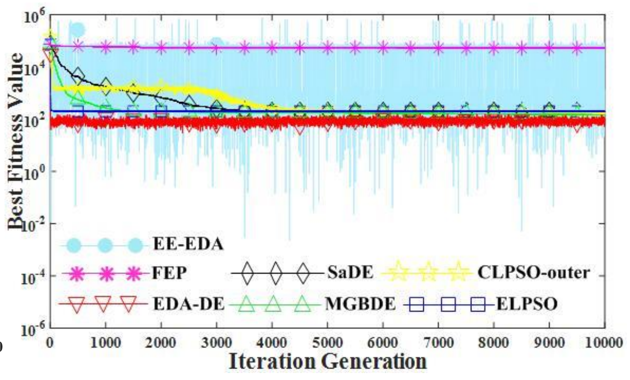


(d) $f_{10}(x)$ Weierstrass

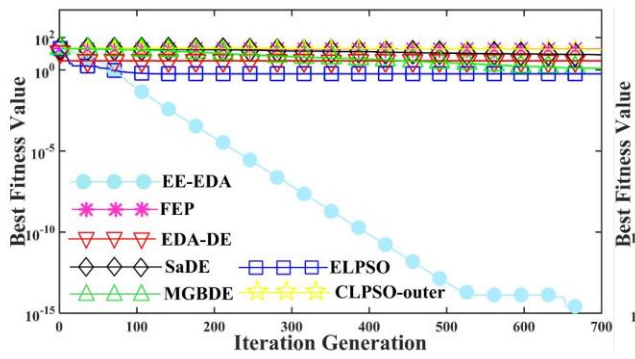
Figure 2.12 The best fitness convergence curves of some 100-*D* function (2).



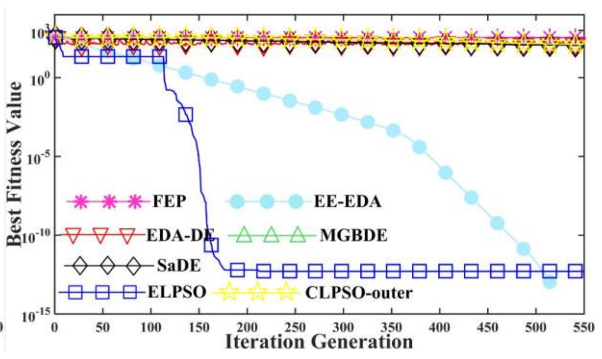
(a) $f_2(x)$ Noise



(b) $f_5(x)$ Rosenbrock



(c) $f_7(x)$ Ackley



(d) $f_{10}(x)$ Weierstrass

Figure 2.13 The best fitness convergence curves of some 200-*D* functions (2).

Here we evaluate the complexity of EE-EDA from the perspective of function evaluation, which is equal to the product of the population size (*PS*) and the maximum iteration generation (*G*). EE-EDA usually adopts a large *PS* while other techniques use a small *PS*. However, the function evaluations of EE-EDA are not excessive, because the generation (AG) in which the algorithm can find the optimal solution is not too large. For example, when EE-EDA solves low-dimensional problems, AG is 40, 12, 13, and so on. When D=100, AG=103 for $f_1(x)$, AG=372 for $f_7(x)$, AG=408 for $f_8(x)$, and so on; when D=200, AG=519 for $f_9(x)$, AG=519 for $f_{10}(x)$ and AG=339 for $f_{11}(x)$ (These data were presented in Tables 2.4, 2.5 and 2.6).

Besides, the algorithms have two stopping criteria in the present research: 1. the algorithm is stopped once it reaches the global optimum of the problem; 2. the algorithm is terminated at the maximum iteration generation, regardless of whether the global optimum is reached. The global optima of these benchmark functions are known prior, so the two stopping criteria can be adopted in the present study. However, in practical applications, the actual global optima may not be known, except in linear programming or convex optimization problems. In such situations, we may use some other stopping criterion like: if the best fitness in the current generation was not considerably better than those in the previous 20 (threshold value) generations, the algorithm is stopped. The maximum variance of all problem variables also can be used as the stopping condition: if the maximum variance is less than some threshold value, the algorithmis stopped. It follows that there is no guarantee that the algorithms will converge to the global optima, for all problems.

2.5 Summary

In this chapter, EDA with univariate marginal Gaussian distribution was improved from the perspective of designing an extreme elitism selection method. This selection made a few leading best solutions account for a higher percentage than other ordinary promising solutions in the parent population to extrude the effect of these few leading best solutions in the evolution. The experimental results demonstrated that extreme elitism selection significantly improved the performance of EDA in the optimization of a set of benchmark problems, both low-dimensional and high-dimensional. In view of the no-free-lunch theorem, there is no optimization algorithm that can generate good performance for all problems. If one algorithm outperforms another, it is likely exploiting the specific structure of the problem. Therefore, EE-EDA set different parameters for different functions. For some special problems, EE-EDA adopted the mixed probabilistic model.

Chapter 3: Application of EDAs in the Robotic Inverse Displacement Problem

3.1 Introduction

Estimation distribution algorithms (EDAs) have been implemented in some practical engineering optimization problems. Ceberio et al. combined an EDA based on the generalized Mallows model with a variable neighbourhood search to solve the permutation flow shop scheduling problem [43]. Hao et al. [44] introduced an algorithm of cooperative estimation of distribution to optimize the scheduling problems in semiconductor testing. Bashir et al. utilized the EDA based on univariate marginal Gaussian distribution to establish a sequence detector, which can jointly estimate the symbols transmitted in a multi-input-multi-output communication system [45]. Gu et al. made use of a similar EDA to solve a dynamic economic dispatch problem in the power systems [46]. There are some other applications, such as the image segmentation [47], the design of passive analog electronic circuits [91], network intrusion detection [92], bi-criteria stochastic job-shop scheduling problem [93], and so on. In this chapter, the EDA with extreme elitism selection (EE-EDA) is first proposed to handle the robotic inverse displacement problem (IDP). The inverse displacement problem (IDP) of a robot, also known as the inverse kinematics problem, involves achieving the robotic arm joint motion values corresponding to a specific orientation and position (motion) of the end effector. It plays an important role in trajectory planning and control of manipulators. This problem is highly nonlinear and has multiplicity of solutions in general, especially for redundant robots, which have more than 6 degree of freedom (DOF) operating in a 3-dimensional space.

Currently, there are two main types of methods for IDP: algebraic and numerical. The algebraic methods are efficient, but they usually depend on the configuration of the robot and the existence of a closed-form solution [13-15]. Hence they are not general approaches. Numerical methods can be universal, but they usually need initial guesses, which may lead to extensive computational effort or instability [16-18]. Numerical methods like the gradient method require an appropriate scalar step size. If the step size is too large, the algorithm may miss a satisfactory solution. But, if the step size is too small, the convergence can be extremely slow. Also, numerical instability is possible in these methods. Some evolution methods have been proposed to handle IDP, such as neural networks (NNs), genetic algorithms (GAs), particle swarm optimization (PSO), and so on [48-49]. They usually do not need initial guesses or a scalar step size, but sometimes they converge slowly or prematurely. Besides, descriptions of some numerical and evolution algorithms only indicate how fast and accurately they can acquire the solution, but not how stable the methods are [16, 48]. A method may be very precise and fast, but it also has to maintain stability, which is particularly important in the operation of a robot manipulator. In this chapter, a manipulator Barrett WAM Arm is used as the simulation platform to show that the proposed EE-EDA can be effectively applied to robotic arms with different configurations, especially those with high and redundant degrees of freedom. In addition, the present chapter addresses the accuracy and efficiency of the algorithm in obtaining the inverse solution, and also the underlying stability of the solution.

3.2 IDP of a 4-DOF Barrett WAM Arm

The kinematic structure of a 4-DOF Barrett WAM Arm is shown in Fig.3.1. It is a robotic arm with the direct-drive capability supported by transparent dynamics between motors and joints. The IDP solution of this robotic arm using EE-EDA is presented now.

3.2.1 Optimization IDP model of a 4-DOF Barrett WAM Arm

Generally, the orientation and position of the end tool (end effector) of a robot arm, expressed in the base coordinate frame, can be represented by the homogeneous matrix $\boldsymbol{T}^{\mathbf{0}}_{\mathbf{Tool}}$ with:

$$\boldsymbol{T}^{\mathbf{0}}_{\mathbf{Tool}} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{o}^{\mathbf{0}}_{\mathbf{Tool}} = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \qquad \boldsymbol{p}^{\mathbf{0}}_{\mathbf{Tool}} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{3.1}$$

Here "**0**" denotes the base coordinate frame and "**Tool**" denotes the end effector. Also, $\boldsymbol{o}^{\mathbf{0}}_{\mathbf{Tool}}$ is the orientation matrix and $\boldsymbol{p}^{\mathbf{0}}_{\mathbf{Tool}}$ is the position matrix (vector). Note that $\boldsymbol{o}^{\mathbf{0}}_{\mathbf{Tool}}$ and $\boldsymbol{p}^{\mathbf{0}}_{\mathbf{Tool}}$ both are functions of the joint movements θ_i (i=1, 2,⋯, n). Specifically, for a revolute robot, θ is the angle of rotation of the joint. If the robot has a prismatic joint, θ is the distance of movement of the joint. Here n is the number of joints (i.e., degrees of freedom). Define $\boldsymbol{o_d}$ and $\boldsymbol{p_d}$ as the desired orientation and position matrices, respectively. Also, $\boldsymbol{o_c}$ and $\boldsymbol{p_c}$ are the current orientation and position matrices. The following equations can be written:

$$\Delta p(\theta_1, \theta_2, \cdots, \theta_n) = \left| \boldsymbol{p_d} - \boldsymbol{p_c} \right|^2 \tag{3.2}$$

$$\Delta o(\theta_1, \theta_2, \cdots, \theta_n) = \sum_{j=1}^{3} (\boldsymbol{o_{d\,j}} \cdot \boldsymbol{o_{c\,j}} - 1)^2 \tag{3.3}$$

$$e(\theta_1, \theta_2, \cdots, \theta_n) = \Delta p(\theta_1, \theta_2, \cdots, \theta_n) + \Delta o(\theta_1, \theta_2, \cdots, \theta_n) \tag{3.4}$$

Then the optimization form may be expressed as:

$$\begin{aligned} &\min \quad e(\theta_1, \theta_2, \cdots, \theta_n) \\ &\text{s.t} \quad \theta_i^L \le \theta_i \le \theta_i^U, \quad i = 1, 2, \cdots, D \end{aligned}$$

Here if $e(\theta_1^*, \theta_2^*, \cdots, \theta_n^*) \le \varepsilon$ ($\varepsilon \to 0$) is satisfied, then $\theta_1^*, \theta_2^*, \cdots, \theta_n^*$ is an acceptable IDP solution, with ε as the threshold value. This optimization form is explicit, but it has the problem that it makes the orientation error $\Delta\boldsymbol{o}$ and position error $\Delta\boldsymbol{p}$ have the same weight [94]. However, in practical applications, $\Delta\boldsymbol{p}$ sometimes is much larger than $\Delta\boldsymbol{o}$**,** in relative (non-dimensional) terms. As a result, the objective function would be biased towards meeting the position objective as opposed to meeting the orientation objective. Hence $\Delta\boldsymbol{p}$ generally should have an appropriate coefficient to make it have the same magnitude as $\Delta\boldsymbol{o}$**.** In the present study, the unit of the position is meter and the distance between the end tool and base origin usually is less than 1*m*, so $\Delta\boldsymbol{p}$ usually would not be much larger than $\Delta\boldsymbol{o}$ (in radians) and the coefficient is set as 1. Besides, IDP generally has a finite multiple of solutions for non-redundant robotic arms and infinite number of solutions for redundant arms. In the present study, it is sought to find one solution. This solution should have an error less than the threshold value for a single discrete point. Moreover, this solution should also vary somewhat, compared to the solution of the previous point in a continuous trajectory, which will be introduced in Section 3.4.

Now this optimization form is applied to solve the IDP for the 4-DOF arm shown in Figure 3.1. First, the Denavit-Hartenberg (D-H) parameters of the arm given in Table 3.1 are substituted into $\boldsymbol{T}_i^{i\text{-}1}$ and $\boldsymbol{T}_1^0$, $\boldsymbol{T}_2^1$, $\boldsymbol{T}_3^2$, $\boldsymbol{T}_4^3$, $\boldsymbol{T}_{\mathrm{Tool}}^4$ are computed. Here $\boldsymbol{T}_i^{i\text{-}1}$ is the generalized transformation matrix, which gives the orientation and position of a coordinate frame *i* with respect to its previous coordinate frame *i*-1 [95].

Table 3.1 D-H parameters and joint bounds of the 4-DOF Barrett WAM Arm.

i	a_i (*m*)	α_i	d_i (*m*)	θ_i	Upper bound *rad*(*deg*)	Lower bound *rad*(*deg*)
1	0	$-\pi/2$	0	θ_1	2.6(150)	-2.6(150)
2	0	$\pi/2$	0	θ_2	2.0(113)	-2.0(113)
3	0.045	$-\pi/2$	0.55	θ_3	2.8(157)	-2.8(157)
4	-0.045	$\pi/2$	0	θ_3	3.1(180)	-0.9(-50)
T	0	0	0.35			

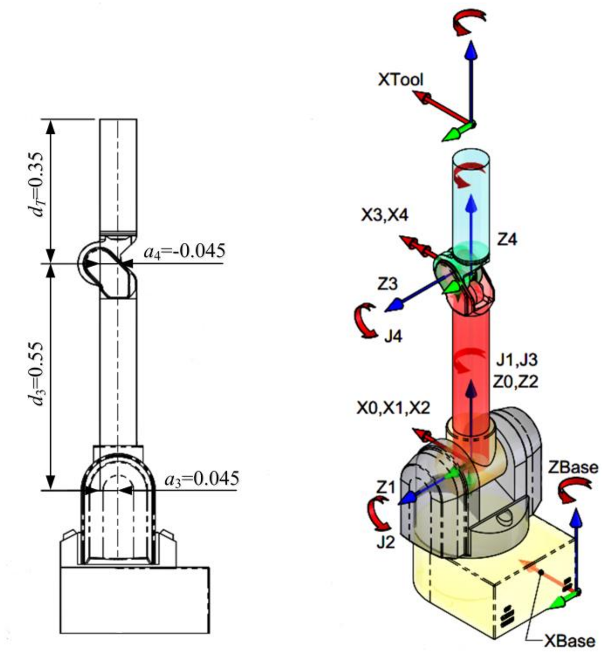


Figure 3.1 The kinematic structure of the 4-DOF Barrett WAM Arm.

$$\boldsymbol{T}_{i}^{i\text{-}1}=\begin{bmatrix}\cos(\theta_i) & \text{-}\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\cos(\alpha_i) & a_i\cos(\theta_i)\\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & \text{-}\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i)\\ 0 & \sin(\alpha_\text{i}) & \cos(\alpha_\text{i}) & d_i\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.5}$$

$$\boldsymbol{T}_{\textbf{Tool}}^{\mathbf{0}}=\boldsymbol{T}_{\mathbf{1}}^{\mathbf{0}}\boldsymbol{T}_{\mathbf{2}}^{\mathbf{1}}\boldsymbol{T}_{\mathbf{3}}^{\mathbf{2}}\boldsymbol{T}_{\mathbf{4}}^{\mathbf{3}}\boldsymbol{T}_{\textbf{Tool}}^{\mathbf{4}} \tag{3.6}$$

Then homogeneous transformation matrix $\boldsymbol{T}_{\textbf{Tool}}^{\mathbf{0}}$ is computed according to Equation (3.6). The following desired orientation and position matrices are used as an instance:

$$\boldsymbol{o_d}=\begin{bmatrix}-0.9143 & -0.3188 & -0.2499\\ -0.1909 & -0.2049 & 0.9600\\ -0.3572 & -0.9254 & 0.1265\end{bmatrix}; \qquad \boldsymbol{p_d}=\begin{bmatrix}0.1611\\ 0.8399\\ 0.2415\end{bmatrix}$$

$$\Delta p(\theta_1,\theta_2,\theta_3,\theta_4)=(0.55c_2+0.35c_2c_4+0.045c_2s_3-0.045c_3s_2+0.045c_3^2s_2-0.35c_3s_3s_4-0.2415)^2+(0.045c_1s_3+0.55s_1s_2-0.045c_3(c_1s_3+c_2c_3s_1)+0.35s_4(c_1s_3+c_2c_3s_1)+0.045c_2c_3s_1+0.35c_4s_1s_2+0.045s_1s_2s_3-0.8399)^2+(0.55c_1s_2-0.045s_1s_3+0.045c_3(s_1s_3-c_1c_2c_3)-0.35s_4(s_1s_3-c_1c_2c_3)+0.35c_1c_4s_2+0.045c_1s_2s_3+0.045c_1c_2c_3-0.1612)^2$$

$$\Delta o(\theta_1,\theta_2,\theta_3,\theta_4)=(0.1265c_2c_4+0.2499s_4(s_1s_3-c_1c_2c_3)+0.96c_4s_1s_2-0.1265c_3s_2s_4-1.0)^2+(0.3188c_3s_1-0.2049c_1c_3+0.9254s_2s_3+0.3188c_1c_2s_3+0.2049c_2s_1s_3-1.0)^2+(0.3572c_2s_4+0.9143c_4(s_4s_3-c_1c_2c_3)-0.1909c_4(c_1s_3+c_2c_3s_1)+0.3572c_3c_4s_2+0.9143c_1s_2s_4+0.1909s_1s_2s_4-1.0)^2$$

Here, $c_i=\cos(\theta_i)$, $s_i=\sin(\theta_i)$, i=1, 2, 3, 4. The final optimization model is given by:

$$\begin{aligned} \min \quad & e=\Delta p(\theta_1,\theta_2,\theta_3,\theta_4)+\Delta o(\theta_1,\theta_2,\theta_3,\theta_4) \\ \text{s.t.} \quad & -2.6\le\theta_1\le 2.6 \\ & -2.0\le\theta_2\le 2.0 \\ & -2.8\le\theta_3\le 2.8 \\ & -0.9\le\theta_4\le 3.1 \end{aligned}$$

3.2.2 EE-EDA program for the problem

Fig. 3.2 demonstrates the flowchart of EE-EDA for IDP of 4 DOF.

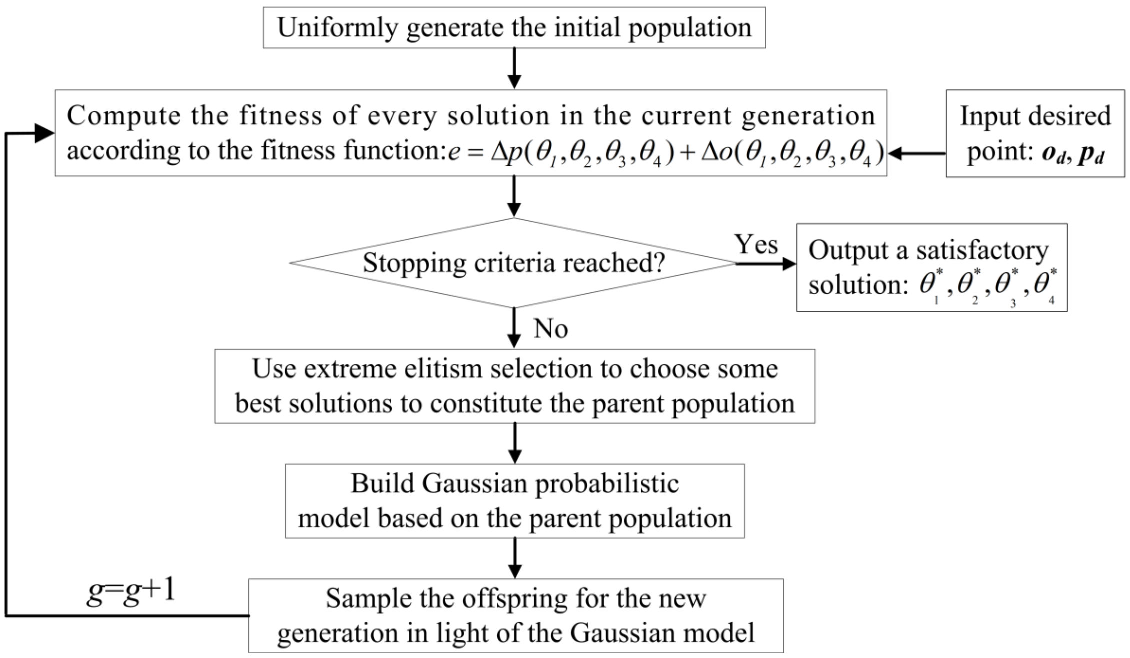

Uniformly generate the initial population
Compute the fitness of every solution in the current generation according to the fitness function: $e=\Delta p(\theta_1,\theta_2,\theta_3,\theta_4)+\Delta o(\theta_1,\theta_2,\theta_3,\theta_4)$
Input desired point: $\boldsymbol{o_d}$, $\boldsymbol{p_d}$
Stopping criteria reached?
Yes
Output a satisfactory solution: $\theta_1^*,\theta_2^*,\theta_3^*,\theta_4^*$
No
Use extreme elitism selection to choose some best solutions to constitute the parent population
Build Gaussian probabilistic model based on the parent population
Sample the offspring for the new generation in light of the Gaussian model
$g=g+1$

Figure 3.2 The flowchart of EE-EDA for solving IDP of the 4-DOF arm.

Presented below are the programs of EE-EDA for this optimization problem.

Step 1: Initialization: Set the population size as PS=2000 and the maximum iteration generation as G=50, and generate the initial population. Here pop(θ_1)$^0, \cdots$, pop(θ_4)^0 are the initial populations of the 1st to the 4th joint angles, in order.

$$\text{pop}(\theta_1)^0 = \text{-2.6} + (2.6 \text{ - } (\text{-2.6})) * \text{rand}(1, PS);$$
$$\vdots$$
$$\text{pop}(\theta_4)^0 = \text{-0.9} + (3.1 \text{ - } (\text{-0.9})) * \text{rand}(1, PS);$$

Step 2: Set g=g+1; if g <= G, go to **Step 3**, else stop;

Step 3: Use Equation (3.7) as the fitness function to compute the fitness of each solution in pop(θ_i)$^{g\text{-}1}$;

$$e = \Delta p(\theta_1, \theta_2, \theta_3, \theta_4) + \Delta o(\theta_1, \theta_2, \theta_3, \theta_4) \tag{3.7}$$

If the fitness of some solution is less than the threshold value ε=1×10^{-5}, stop loop and output θ_1^*, θ_2^*, θ_3^*, θ_4^* as a satisfied solution; else go to **Step 4**;

```
Step 4: $M = 1000$
    for $i = 1:4$
        for $j = 1:50$       parent population_$\theta_{ij} = \theta_i(II(1))$         end
        for $j = 51:90$      parent population_$\theta_{ij} = \theta_i(II(2))$         end
        for $j = 91:120$     parent population_$\theta_{ij} = \theta_i(II(3))$         end
        for $j = 121:140$    parent population_$\theta_{ij} = \theta_i(II(4))$         end
        for $j = 141:150$    parent population_$\theta_{ij} = \theta_i(II(5))$         end
        for $j = 151:M$      parent population_$\theta_{ij} = \theta_i(II(j\text{-}145))$  end
    end
```

In **Step 4**, the parent population size M is set at 1000. According to the fitness, which equals the objective function value, the solutions are sorted from the best to the worst. $\theta_i(II(1))$ ~$\theta_i(II(5))$ are the 1st through the 5th solutions. Then the 1st through the 50th individuals in the parent population are all made equal to the 1st best solution; the 51st through the 90th individuals are all made equal to the 2nd solution, and so on. But the 151st through the 1000th individuals, each only equals one ordinary solution. For instance, the 151st individual equals the 6th ordinary solution and the 152nd equals the 7th solution, and so on until the 1000th individual equals the 855th solution. As a result, these top solutions take the 50, 40, 30, 20, 10 portions in the parent population, while the 6th to the 855th solutions, each equally accounts for only 1 item.

Step 5: Use Equation (3.8) to estimate the mean and variance of the parent population. θ_{ij} is the j^{th} solution of the i^{th} joint angle among the M promising solutions;

$$\mu_i = \frac{\sum_{j=1}^{M}\theta_{ij}}{M} \qquad \sigma_i^2 = \frac{\sum_{j=1}^{M}(\theta_{ij}-\mu_i)^2}{M-1} \qquad (i=1,2,3,4;\ j=1,2,\cdots,M) \tag{3.8}$$

Step 6: Use the mean and variance to build the univariate marginal Gaussian distribution model. The density function of this distribution is given by Equation (3.9). Sample 2000 offspring solutions for each joint angle with the code in Equation (3.10) and check every offspring solution. If some solution exceeds the bound, Equation (3.11) makes it equal to a new random number in the bound (“rand” produces a random number in [0, 1]). Then return to **Step 2**.

$$f_{\boldsymbol{X}}(x_1,\cdots,x_4) = \prod_{i=1}^{4}\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left(-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right) \qquad \boldsymbol{X} = (x_1,\cdots,x_4)^T \tag{3.9}$$

$$\text{offspring\_individual}_{ij} = \text{normrnd}(\mu_i,\sigma_i,1,2000) \quad (i=1,2,3,4;\ \ j=1,\cdots,2000) \tag{3.10}$$

$$\text{if offspring\_individual}_{ij} \notin [\theta_i^L,\theta_i^U],\ \text{offspring\_individual}_{ij} = \theta_i^L + (\theta_i^U-\theta_i^L)*\text{rand} \tag{3.11}$$

3.2.3 Comparison of EDA with different selection for IDP of 4 DOF arm

Fig. 3.3 shows the convergence process of one operation of the EDAs with four different selection models (here a threshold value is not set). The dash line indicates Error = 1×10^{-5}. The

EDA with truncation model (TR-EDA) could not converge to a satisfactory error (SE) before the 50th generation while the tournament (TO-EDA) reached an SE in the earliest generation (6th), then the proportional (PR-EDA) (12th) followed by EE-EDA (15th). The tournament selection chooses the 1st best solution from a random subpopulation and repeats this selection *PS* times; so it is possibly to make more leading solutions enter the parent population than by the other selection models. A dense presence of these top solutions can make the mean vector of the Gaussian model approach an SE quickly. As a result, the algorithm can iterate a few generations to find this SE. But if these leading best solutions are close to some local optimum, the algorithm will more likely get trapped there, as there may be many of the same solutions in the parent population. These same solutions make the variance of the Gaussian model become very small and the algorithm cannot search for diversified solutions.

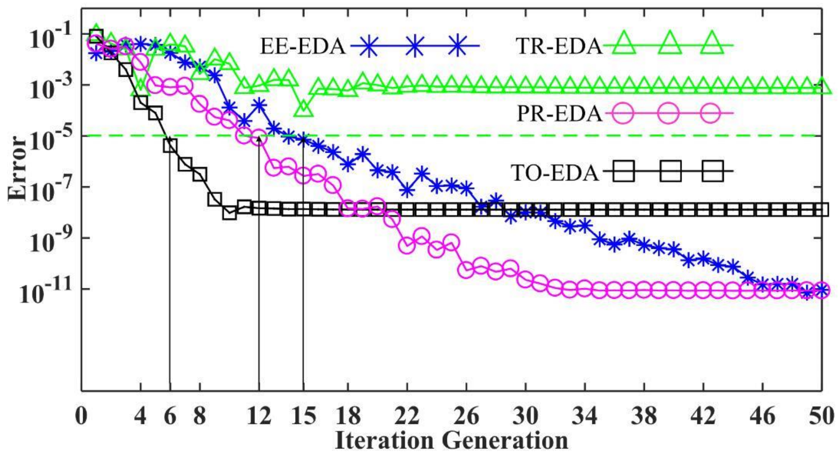


Figure 3.3 Convergence trend of EDA with different selection.

In contrast, the truncation treats every solution in the parent population equally. So it does not improve the ability of some top solutions, and TR-EDA hardly acquires an SE in an early generation. In proportional selection, the probability of one solution that is being elected is proportional to its fitness. Hence, if one solution is much better than the other solutions, it can account for much more portions than other solutions in the parent population, which makes it likely that PR-EDA reaches an SE in an early generation. However, similar to the tournament method, if these top solutions are near a local optimum, it will likely converge prematurely. EE-EDA can achieve an SE as it enhances the effect of some top solutions. But it extrudes only a few top solutions and these top solutions do not take too many portions in the parent population. So it cannot get an SE in a very early generation. To test the stability of finding an SE of EDA with each selection model, the algorithm runs 100 times each.

In Fig. 3.4, there is no test result for the EDA with the truncation selection below the solid line 10^{-5}**.** Hence, they all fail to reach an SE. Also, 26 tests of the algorithm with tournament selection and 4 tests of the algorithm with proportional selection fail while EE-EDA has no failure. But it is seen that TO-EDA and PR-EDA can achieve much lower error than EE-EDA; for example, in the 43th and 88th operations (the error reached the accuracy of 10^{-17}).

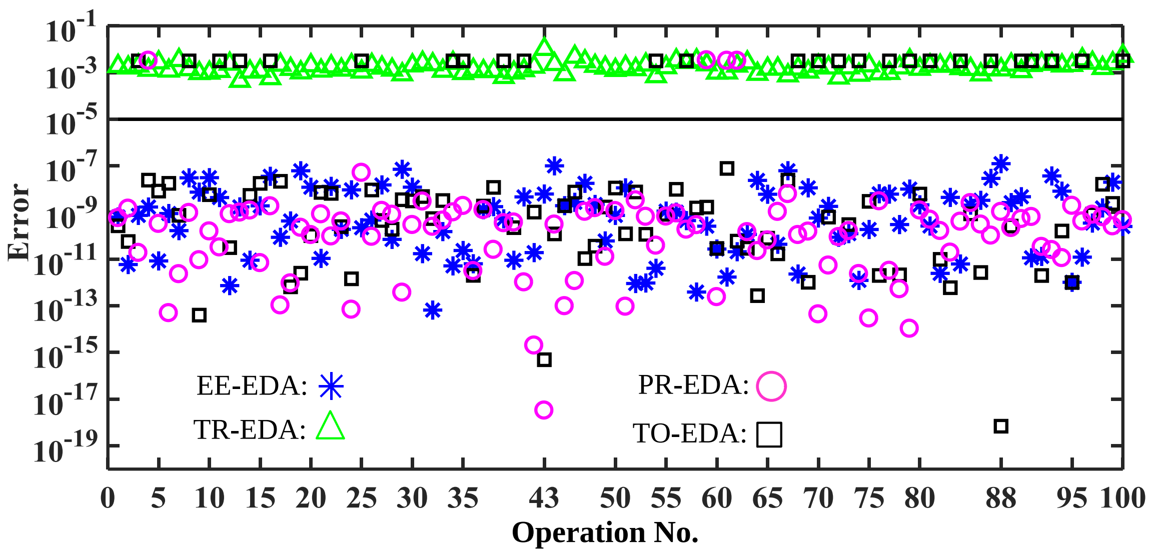


Figure 3.4 Distribution of the final error of 100 tests of the EDA with different selection.

However, this situation is occasional and cannot maintain stability. EE-EDA highlights the role of a few leading solutions and simultaneously retains most of other common promising solutions as the parent to enrich the population diversity. As a result, it usually can avoid premature convergence. Now set the threshold value at 10^{-5} in the programs and run them 100 times each, to compare how fast these algorithms can reach an SE. Table 3.2 shows the distribution of the stop generations in which the algorithms can achieve an SE. Here if the algorithm stops in the last generation (50th), it means the algorithm fails to get an SE. It is noted that 100 operations of TR-EDA, 26 of TO-EDA and 4 of PR-EDA stopped in the 50th generation while all operations of EE-EDA stopped before the 50th generation. Then it is seen that if the TO-EDA does not fail prematurely, it usually can reach an SE in a very early generation (70 operations reach an SE before the 11th generation while no EE-EDA can do this). Also, 85 operations of PR-EDA find an SE before the 14th generation. These test results verify the previous analysis that the tournament and proportional selection can choose more of the same solutions than EE-EDA to make the EDA reach an SE in an early generation, if these solutions are close to a potential SE. But it does not necessarily mean that their running speed is high.

Table 3.2 Distribution of the stop iteration generations of 100 tests of each algorithm.

Interval of Iteration Generation	[0, 10]	[11, 13]	[14, 22]	[23, 49]	G=50
TR-EDA	0	0	0	0	100
TO-EDA	70	3	1	0	26
PR-EDA	16	69	11	0	4
EE-EDA	0	43	57	0	0

Here the complexity of EDAs is evaluated from the perspective of computing time. Fig. 3.5 shows the computing time of these algorithms (vertical axis uses log scale) reaching an SE. These tests are done using MATLAB with a processor Intel(R) Corei5-2320 CPU@3.00GHz. Note that EE-EDA has the lowest average computing time of 0.0595s. Although some operations of the EDAs with the tournament and proportional selection methods could achieve an SE in an earlier generation than EE-EDA, the computing time of every generation of them was much larger than that of EE-EDA. Tournament selects only the 1st solution from a random subpopulation with a size of $\alpha * PS$ for PS times (α=0.05, PS=2000). So in every generation, it performs the sort operation 2000 times to acquire 2000 solutions as the parent. The proportional model computes the cumulative probability for every solution and generates 2000 random numbers within [0, 1]. Then it compares these random numbers with the cumulative probability to select 2000 solutions. If TO-EDA and PR-EDA both select 0.5×2000=1000 solutions as the parent in every generation like EE-EDA and TR-EDA, they may save half the computing time, but still the computing time is greater than that of EE-EDA and TR-EDA.

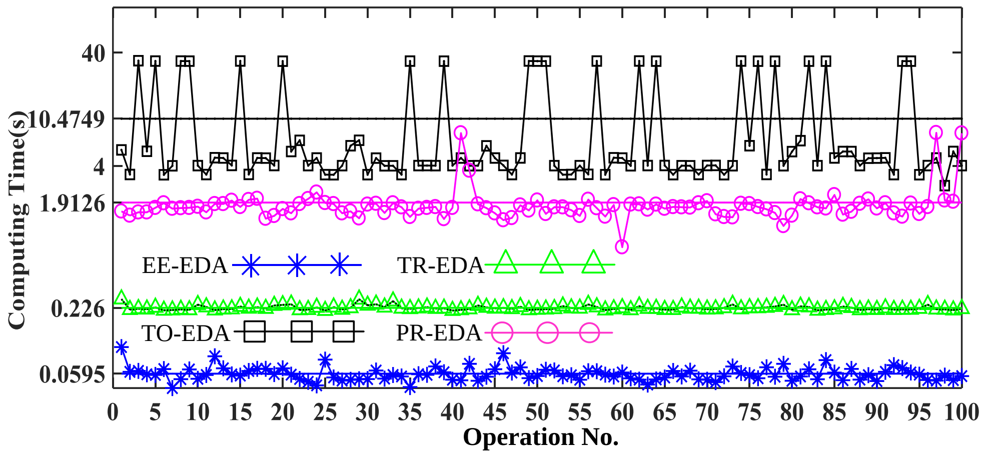


Figure 3.5 Distribution of the computing times of 100 tests of the EDA with different selection.

The truncation selection does not have complex operations, unlike the tournament and proportional methods. It just uses once sort operation and directly truncates ahead half of the solutions. As a result it takes less computing time, even though it iterates through all 50 generations. Essentially, EE-EDA uses the same computing time as TR-EDA. But since it exploits the role of a few top solutions and can reach an SE before the 50th generation, it requires less computational time. So, by comparing the test results and the computing time, it is seen that EE-EDA is better suited than the EDAs with truncation, tournament and proportional selection for this optimization problem.

3.3 EE-EDA Combined with DM to solve IDP of the 7-DOF Barrett WAM arm

Fig. 3.6 shows the kinematic structure of the 7-DOF Barrett WAM Arm. The D-H parameters of the 7-DOF arm are given in Table 3.3. Then the generalized transformation matrix $\boldsymbol{T}_i^{i-1}$ for each link and the homogeneous matrix $\boldsymbol{T}_{\mathrm{Tool}}^{0}$ are computed. The following desired orientation and position matrices are used as a numerical example:

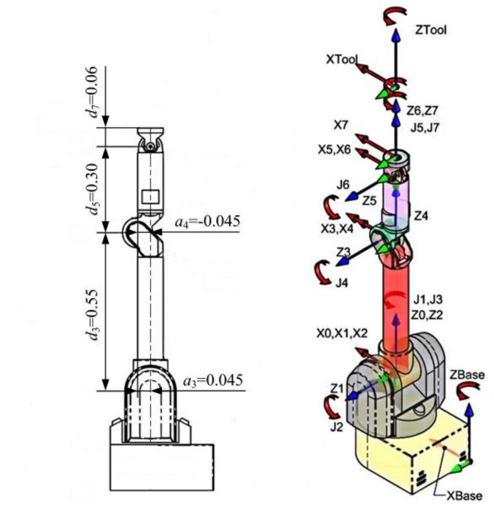


Figure 3.6 The kinematic structure of the 7-DOF Barrett WAM Arm.

Table 3.3 D-H parameters and joint bounds of the 7-DOF Barrett WAM Arm.

i	a_i (m)	α_i	d_i (m)	θ_i	Upper bound *rad*(*deg*)	Lower bound *rad*(*deg*)
1	0	$-\pi/2$	0	θ_1	2.6(150)	-2.6(-150)
2	0	$\pi/2$	0	θ_2	2.0(113)	-2.0(-113)
3	0.045	$-\pi/2$	0.55	θ_3	2.8(157)	-2.8(-157)
4	-0.045	$\pi/2$	0	θ_4	3.1(180)	-0.9(-50)
5	0	$-\pi/2$	0.3	θ_5	1.24(71)	-4.76(-273)
6	0	$\pi/2$	0	θ_6	1.6(90)	-1.6(-90)
7	0	0	0.06	θ_7	3.0(172)	-3.0(-172)
T	0	0	0			

$$\boldsymbol{o_d} = \begin{bmatrix} 0.8822 & -0.4040 & -0.2419 \\ 0.4704 & 0.7788 & 0.4150 \\ 0.0274 & -0.4799 & 0.8771 \end{bmatrix} \qquad \boldsymbol{p_d} = \begin{bmatrix} 0.2401 \\ 0.2486 \\ 0.8382 \end{bmatrix}$$

The Δ*o* and Δ*p* are shown in the Appendix.

$$\begin{array}{lll} \min & e = \Delta p(\theta_1,\theta_2,\theta_3,\theta_4) + \Delta o(\theta_1,\theta_2,\theta_3,\theta_4) & \\ \text{s.t.} & -2.6 \le \theta_1 \le 2.6 & -2.0 \le \theta_2 \le 2.0 \\ & -2.8 \le \theta_3 \le 2.8 & -0.9 \le \theta_4 \le 3.1 \\ & -4.76 \le \theta_5 \le 1.24 & -1.6 \le \theta_6 \le 1.6 \\ & -3.0 \le \theta_7 \le 3.0 & \end{array}$$

3.3.1 EE-EDA for IDP of 7-DOF arm

The same algorithm of EE-EDA for the 4-DOF arm is used to solve the present problem. In order to compare the test results of EE-EDA with different parameters, EE-EDA with *PS*=2500 also used to solve this problem. In addition, when *PS*=2500, the portion of each top solution in the elite corps increases to 100, 80, ⋯, 20. These two programs are run 100 times, each.

In Fig. 3.7, there are 15 test results above the solid line 10^{-5} for EE-EDA with *PS*=2500 while 45 results for EE-EDA with *PS*=2000. So the percentage of top solutions in the parent population usually should increase as the dimension increases. Besides, a large *PS* is beneficial. However, constantly increasing the percentage of the top solutions may reduce the population diversity. Also, if the *PS* is too large, the computational cost will increase accordingly.

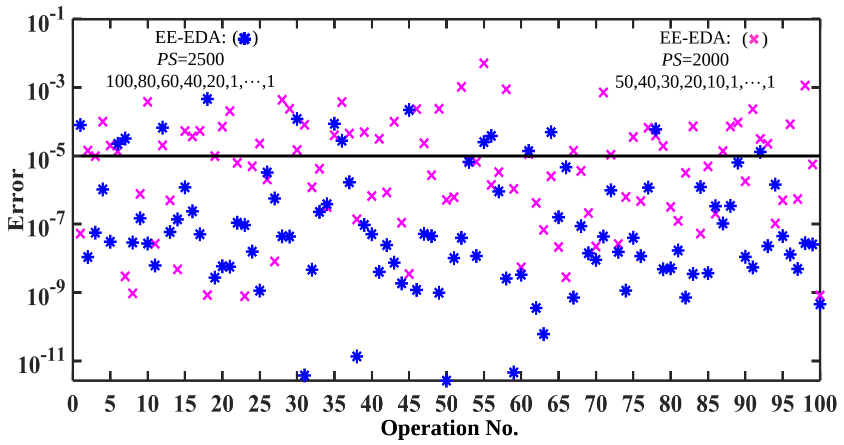


Figure 3.7 Distribution of the final error of 100 tests of two different EE-EDAs.

3.3.2 EE-EDA Combined with DM to solve IDP of 7-DOF arm

3.3.2.1 Differential Mutation strategies

In order to make EE-EDA more stable and efficient, it can be modified in other aspects, not just the selection method and the population size. Here, differential mutation (DM) is incorporated to improve EE-EDA. The DM comes from the evolution algorithm of differential evolution (DE). There are five popular strategies of mutation operation, as given in Table 3.4 [79]. Here V_j^g is the new solution after mutation, X_j^g is the original solution, X_{best}^g is the 1st best solution of the g^{th} generation, $F \in [0, 2]$ is the scaling factor, $X_{r_1}^g \sim X_{r_5}^g$ is the selected random solution, $j \neq r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5$, and r_1, r_2, r_3, r_4, r_5 is randomly generated integer within the range [1, *PS*].

Table 3.4 Differential mutation strategies.

DE/rand/1	$V_j^g = X_{r_1}^g + F\cdot(X_{r_2}^g - X_{r_3}^g)$
DE/best/1	$V_j^g = X_{\text{best}}^g + F\cdot(X_{r_1}^g - X_{r_2}^g)$
DE/current-to-best/1	$V_j^g = X_j^g + F\cdot(X_{\text{best}}^g - X_{r_1}^g) + F\cdot(X_{r_2}^g - X_{r_3}^g)$
DE/best/2	$V_j^g = X_{\text{best}}^g + F\cdot(X_{r_2}^g - X_{r_3}^g) + F\cdot(X_{r_4}^g - X_{r_5}^g)$
DE/rand/2	$V_j^g = X_{r_1}^g + F\cdot(X_{r_2}^g - X_{r_3}^g) + F\cdot(X_{r_4}^g - X_{r_5}^g)$

$$V_j^g = X_j^g + F \cdot (X_{\text{best}}^g - X_{r_1}^g) + F \cdot (X_{\text{best}}^g - X_{r_2}^g) \tag{3.12}$$

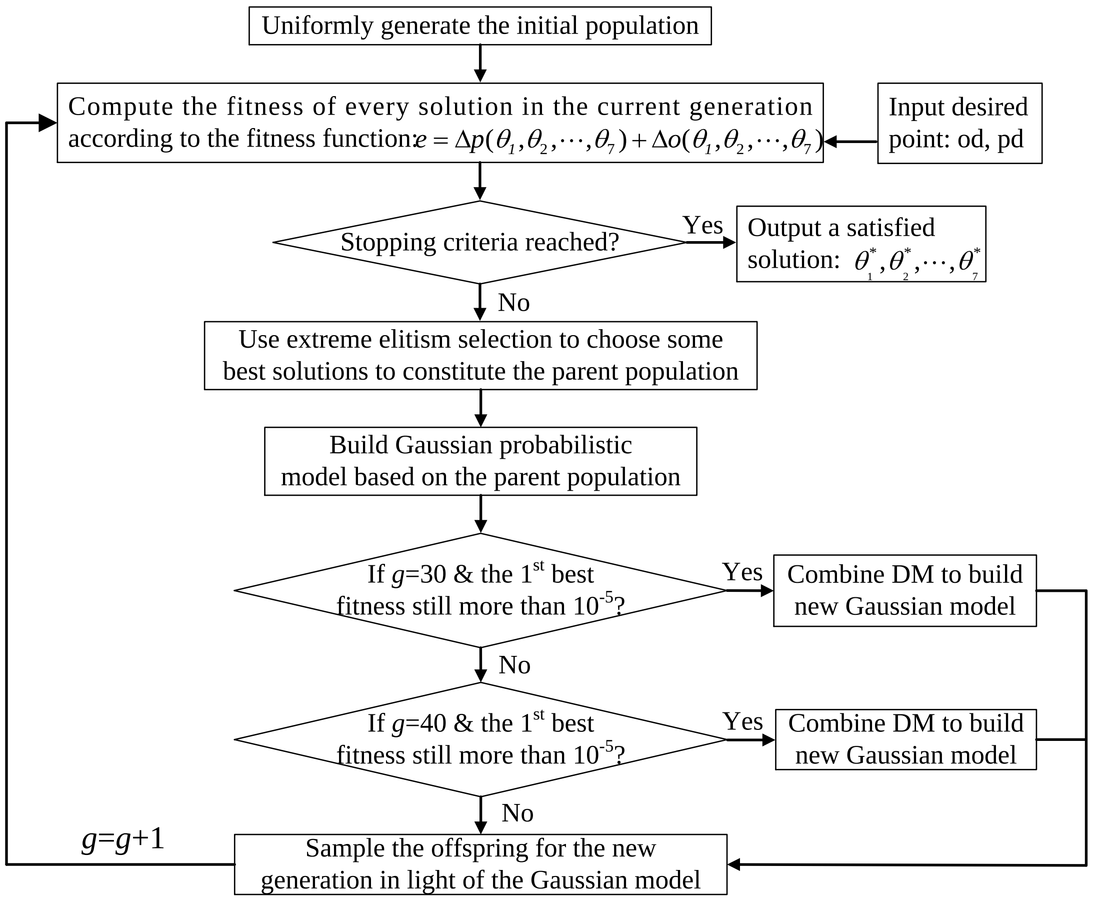
Here, a new strategy called "DE/target-to-2best/1" is designed for EE-EDA given by Equation (3.12). The new strategy can enrich the population diversity while keeping the effect of the 1st best solution as it has two X_{best}^g. In order to combine EE-EDA with DM, a supervisory mechanism is incorporated into the algorithm. Usually, the distance between different solutions can be calculated to determine whether the algorithm has stuck in a premature state. But this method consumes much time when the *PS* is large. In order to improve the efficiency, a supervisory method is used for checking the fitness of the 1st best solution, which is faster. Table 3.5 presents the distribution of stop generations in which EE-EDA can reach an SE. Note that 56 tests reach an SE within the (20th, 30th] generation and 16 tests within the (30th, 40th] generation. They account for 65.9% and 18.8%, respectively, among the total successful tests. So, the fitness of the 1st best solution in the 30th generation is checked, and if it is still more than 10^{-5}, DM is incorporated to mutate the population. If not, no mutation is carried out. The same supervisory method is also done in the 40th generation. The new algorithm is called EE-EDA/DM.

Table 3.5 Distribution of stop generations of EE-EDA with *PS*=2500 in Section **3.3.1.**

Interval of iteration generation	[0, 10]	(10, 20]	(20, 30]	(30, 40]	(40, 50]	(50, 60)	*G*=60
Times of operation	0	7	56	16	6	0	15

3.3.2.2 Program of EE-EDA/DM for the IDP of 7-DOF

Fig. 3.8 shows the flowchart of EE-EDA/DM that solves the IDP of 7-DOF. Presented below is the program of EE-EDA/DM handling this problem.



Uniformly generate the initial population
Compute the fitness of every solution in the current generation according to the fitness function: $e = \Delta p(\theta_1,\theta_2,\cdots,\theta_7) + \Delta o(\theta_1,\theta_2,\cdots,\theta_7)$
Input desired point: od, pd
Stopping criteria reached?
Yes
Output a satisfied solution: $\theta_1^*,\theta_2^*,\cdots,\theta_7^*$
No
Use extreme elitism selection to choose some best solutions to constitute the parent population
Build Gaussian probabilistic model based on the parent population
If g=30 & the 1st best fitness still more than 10^{-5}?
Yes
Combine DM to build new Gaussian model
No
If g=40 & the 1st best fitness still more than 10^{-5}?
Yes
Combine DM to build new Gaussian model
No
g=g+1
Sample the offspring for the new generation in light of the Gaussian model

Figure 3.8 The flowchart of EE-EDA/DM for the IDP of 7-DOF arm.

Step 1: Initialization: set *PS*=2500, *G*=60, and generate an initial population;

Step 2: Set $g=g+1$, if $g<=G$, go to **Step 3**; else stop;

Step 3: If the fitness of some solution is less than the threshold value $\varepsilon=1\times10^{-5}$, stop loop and output θ_1^*, θ_2^*,$\cdots$, θ_7^* as an accepted solution; else go to **Step 4**;

Step 4: Sort the solutions from the best to the worst and choose the parent solutions with the flowing code. The parent population size *M*=1250.

```
for $i = 1:7$
    for $j = 1:100$      parent population_$\theta_{ij} = \theta_i(\mathrm{II}(1))$      end
    for $j = 101:180$    parent population_$\theta_{ij} = \theta_i(\mathrm{II}(2))$      end
    for $j = 181:240$    parent population_$\theta_{ij} = \theta_i(\mathrm{II}(3))$      end
    for $j = 241:280$    parent population_$\theta_{ij} = \theta_i(\mathrm{II}(4))$      end
    for $j = 281:300$    parent population_$\theta_{ij} = \theta_i(\mathrm{II}(5))$      end
    for $j = 301:M$      parent population_$\theta_{ij} = \theta_i(\mathrm{II}(j-295))$  end
end
```

Step 5: Estimate the mean and variance of the parent population to establish the univariate marginal Gaussian model;

```
Step 6: if $g == 30 \,\&\&\, e(\mathrm{gbest}_1^{30}, \mathrm{gbest}_2^{30}, \cdots, \mathrm{gbest}_7^{30}) > 10^{-5}$
            $F = 1;$
            for $i = 1:7$
                for $j = 1:PS$
                    $R = \mathrm{randi}([1, PS], 2, 1);$
                    $\theta(j,:)=\theta(\mathrm{II}(j),:)+F\cdot(\theta(\mathrm{II}(1),:)-\theta(R(1,1),:))+F\cdot(\theta(\mathrm{II}(1),:)-\theta(R(2,1),:));$
                end
                $\mu_i^{30} = \mathrm{gbest}_i^{30};$
                $\sigma_i^{30} = 0.98\sigma_i^{29};$
            end
            turn to Step 8
        else    turn to Step 8
        end
```

Comment: If the fitness of 1st best solution in the 30th generation is more than 10^{-5}, run **Step 6** Else, turn to **Step 8**. In **Step 6**, do differential mutation for every solution. Every new solution θ (*j*,:) is generated by its original value θ (II(*j*),:) plus scaling factor F times the differential vectors between two random original solutions $\theta(R(1,1), :)$, $\theta(R(2,1), :)$ and the old 1st best solution $\theta(\mathrm{II}(1),:)$. Then, a novel scheme of obtaining the mean of Gaussian model is adopted here: the mean of the 30th generation μ_{i}^{30} is directly equal to gbest_I^{30}, which is the new 1st best solution of the 30th generation after differential mutation. The mean is not determined by the method in **Step 4** and **5**. After just one differential mutation, the quality of the total population may decrease, but the small perturbation around some original excellent solutions may find a few more excellent solutions.

Now the following example is used for illustration:

$$y = 3(\cos(x))^2 + 4\sin(x) - x\cos(x) + e^{-x^2} \tag{3.13}$$

Fig. 3.9 shows the curve of function *y*. Its minimum is -7.2486 where, *x* = 4.31. For instance, there are three original solutions: -1.86, 2.59 and 3.52. Compared to the other two original solutions, 3.52 is an excellent original solution because it is closest to the optimum 4.31. After differential mutation, -1.86 is changed to -1.23, 2.59 to 2.11, and 3.52 to 4.00. Note that the mean value is (-1.23+2.11+4.00)/3≈1.63, which is far away from the optimum point 4.31. But if the new 1st best solution 4.00 is directly used as the mean, it is more likely to find the optimum.

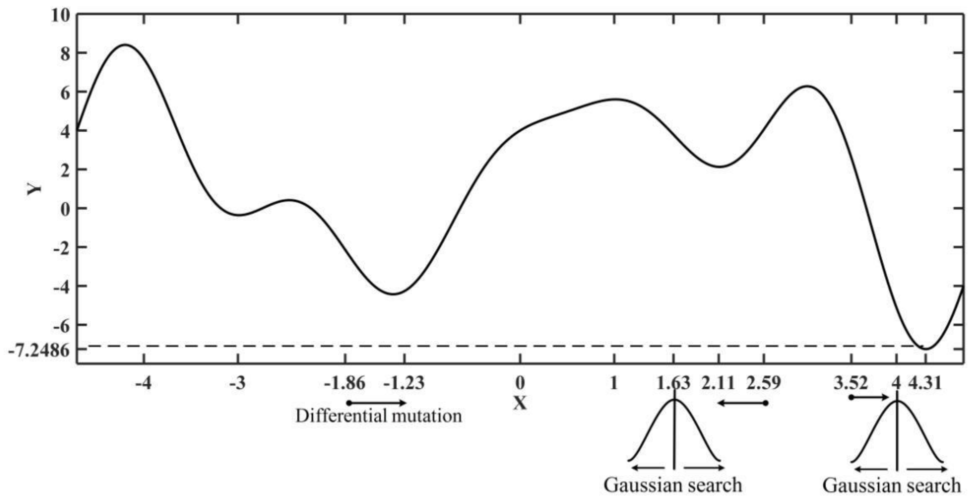


Figure 3.9 Differential mutation and Gaussian search.

Here mean determines the search direction in the Gaussian model. Besides, the new variance is made equal to 0.98 times the variance of the last generation. This is because the variance determines the search step size. A small search step size implies searching of new solutions in a small neighbourhood of the mean. So it increases the probability of finding high-quality solutions in the next population.

```
Step 7:  if $g == 40 \&\& e(\text{gbest}_1^{40}, \text{gbest}_2^{40}, \cdots, \text{gbest}_7^{40}) > 10^{-5}$
             $F = 1$;
             for $i = 1:7$
                  for $j = 1:PS$
                       $R = \text{randi}([1, PS], 2, 1)$;
                       $\theta(j,:) = \theta(\text{II}(j),:) + F \cdot (\theta(\text{II}(1),:) - \theta(R(1,1),:)) + F \cdot (\theta(\text{II}(1),:) - \theta(R(2,1),:))$;
             end
                  $\mu_i = \text{gbest}_i^{40}$;
                  $\sigma_i = 1.5\sigma_i^{39}$;
             end
             turn to Step 8
         else    turn to Step 8
         end
```

Comment: If the algorithm still cannot reach an SE in the 40th generation, run **Step 7**. Here the new variance is equal to 1.5 times that of the last generation. This is because the population diversity likely becomes very low when the algorithm has run 40 generations. The larger variance can facilitate the algorithm to search increasingly various solutions.

Step 8: Sample *PS* new individuals based on the Gaussian model. Then return to **Step 2**.

Fig. 3.10 demonstrates three distinct situations of the convergence trend of EE-EDA/DE when solving this optimization problem. The circle line shows the convergence trend of one operation of EE-EDA/DE using the method in **Steps 4** and **5** to get the mean and variance of the Gaussian model after differential mutation. In the 30th generation, the error of the 1st best solution was still greater than 10^{-5}. Hence it went through the differential mutation. But after the mutation, the error of the 1st best solution in the 31th generation became greater than that of the 30th generation while the algorithm that directly adopted the 1st best solution as the mean vector of Gaussian model reached an SE after the differential mutation, which is shown by the star line.

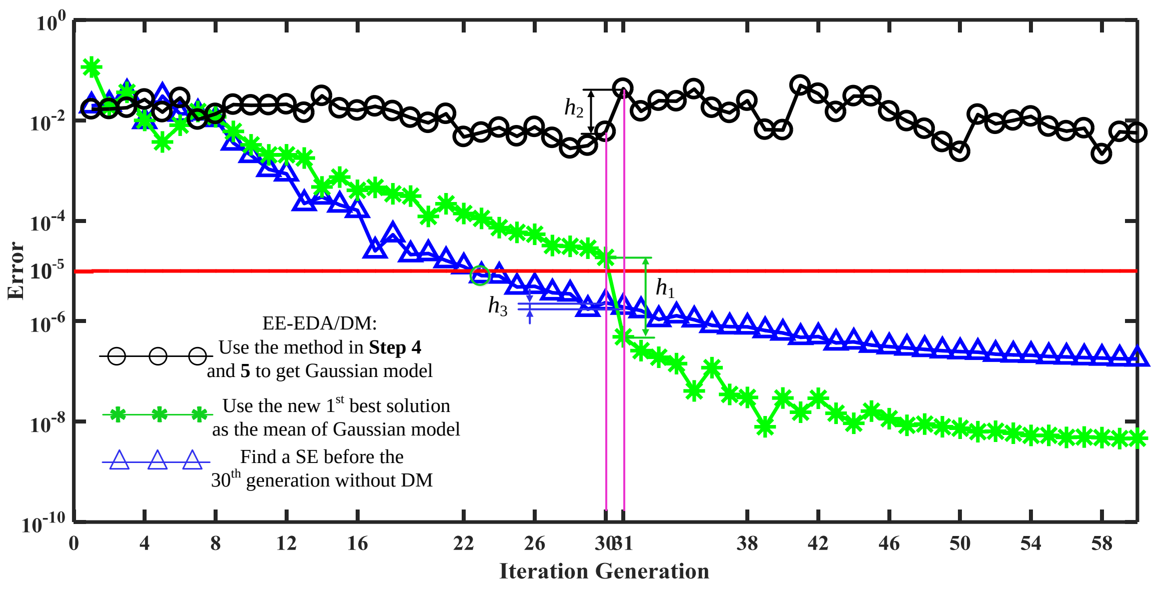


Figure 3.10 Convergence trend of the error of EE-EDA/DM in three different situations.

The triangle line indicates that the algorithm did not experience the mutation since it reached an error less than 10^{-5} in the 23rd generation (here no threshold value is set in the program). Hence the change $\boldsymbol{h_3}$ between the 30th and the 31th generation is small as usual while the other two algorithms have large changes $\boldsymbol{h_1}$, $\boldsymbol{h_2}$. Here it should be noted that the differential mutation cannot guarantee that the next generation will reach an SE. However, it can enrich the population diversity even if it does not make the algorithm achieve a better error. Then the algorithm is run 100 times to compare its stability with EE-EDA. In Fig. 3.11, just one operation result of EE-EDA/DM is located above the solid line of 10^{-5}; so the combination of EE-EDA and DM has resulted in an obvious improvement in this optimization problem. Besides, DM is not integrated in the solution of the IDP of the 4-DOF arm, because EE-EDA can handle this problem of 4-DOF arm on its own. Moreover, from Table 3.2 it is seen that the algorithm can achieve a satisfactory solution before the 30th generation, in this problem. Hence, it is not necessary to make the algorithm go through DM.

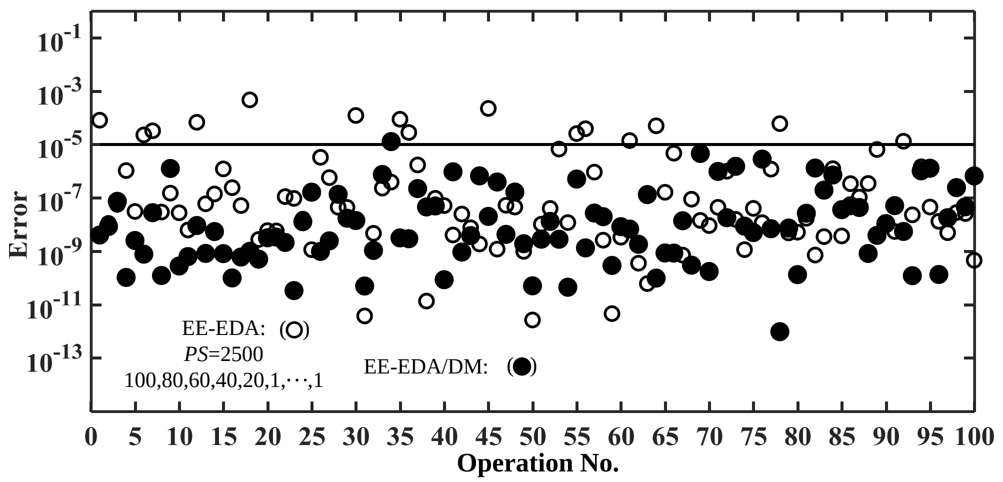


Figure 3.11 Distribution of final error of 100 tests of EE-EDA and EE-EDA/DM.

3.3.3 Comparison of EE-EDA/DE, ES, ELPSO and IDEA for IDP of 7-DOF arm

All the simulations as reported above have used one desired orientation and position. Now 10 random desired points are used to test EE-EDA/DE. This new algorithm is similar in concept to the heuristic optimization algorithms: evolution strategies (ES) and PSO. It also takes advantage of the mutation operation of DE. Hence, PSO, ES and DE are used to compare the performance. Here, a recently improved PSO called enhanced leader PSO (ELPSO) [87] and a classical ES [96] are adopted. ELPSO uses Gaussian, Cauchy, opposition learning, separate DE and overall DE to mutate the global best solution in every generation. If the mutated global optimum is better than the old one, it becomes the new best solution. It has produced better results for some benchmark functions than with other modified PSO. Advantage can be taken as well of an improved DE named intersect mutation differential evolution (IDEA), which divides the population into a better part and a worse part according to the fitness and then uses different mutation and crossover strategies for the better and worse populations. It also has generated better results than with other similar heuristic optimization algorithms for some popular benchmark functions. Here, IDEA second process is adopted [97]. In Table 3.6, $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, $\boldsymbol{d}_3$ represent the $\boldsymbol{n}$, $\boldsymbol{o}$, $\boldsymbol{a}$ direction vectors of the orientation matrix, respectively, and $\boldsymbol{p}$ denotes the position vector. Every point is tested 10 times by ES, IDEA, ELPSO and EE-EDA/DM with 60 iterations and the threshold value ε=1×10^{-5}.

Table 3.6 Orientation and position matrix of 10 random desired points.

No.	**Orientation and position matrices**
1	$\boldsymbol{d}_1=[0.0060\ -0.9287\ 0.3708]^T;\boldsymbol{d}_2=[-0.9675\ -0.0992\ -0.2327]^T;\boldsymbol{d}_3=[0.2529\ -0.3573\ -0.8991]^T;\boldsymbol{p}=[0.3244\ -0.1097\ -0.1097]^T$
2	$\boldsymbol{d}_1=[0.8948\ -0.4332\ -0.1079]^T;\boldsymbol{d}_2=[0.4009\ -0.6733\ 0.6212]^T;\boldsymbol{d}_3=[0.1965\ -0.5991\ -0.7762]^T;\boldsymbol{p}=[-0.1199\ -0.8849\ 0.0436]^T$
3	$\boldsymbol{d}_1=[-0.7281\ 0.6316\ 0.2666]^T;\boldsymbol{d}_2=[0.3819\ 0.6966\ -0.6073]^T;\boldsymbol{d}_3=[-0.5693\ -0.3404\ -0.7484]^T;\boldsymbol{p}=[-0.1590\ 0.2943\ -0.1117]^T$
4	$\boldsymbol{d}_1=[0.5570\ 0.5018\ -0.6618]^T;\boldsymbol{d}_2=[-0.2573\ 0.8618\ 0.4370]^T;\boldsymbol{d}_3=[0.7896\ -0.7311\ 0.6092]^T;\boldsymbol{p}=[0.5622\ 0.3402\ 0.0772]^T$
5	$\boldsymbol{d}_1=[0.1780\ -0.8978\ -0.4028]^T;\boldsymbol{d}_2=[0.3211\ 0.4399\ -0.8387]^T;\boldsymbol{d}_3=[0.9302\ 0.0200\ 0.3666]^T;\boldsymbol{p}=[0.3222\ 0.1357\ 0.1952]^T$
6	$\boldsymbol{d}_1=[0.3728\ -0.6852\ -0.6257]^T;\boldsymbol{d}_2=[0.4104\ 0.7266\ -0.5511]^T;\boldsymbol{d}_3=[0.8322\ -0.5136\ 0.5520]^T;\boldsymbol{p}=[0.4377\ 0.2757\ 0.2947]^T$
7	$\boldsymbol{d}_1=[-0.7707\ -0.4884\ -0.4091]^T;\boldsymbol{d}_2=[-0.6371\ 0.5890\ 0.4972]^T;\boldsymbol{d}_3=[-0.1882\ 0.6439\ -0.7651]^T;\boldsymbol{p}=[-0.1598\ -0.2486\ 0.2123]^T$
8	$\boldsymbol{d}_1=[-0.4524\ -0.8917\ 0.1465]^T;\boldsymbol{d}_2=[0.8912\ -0.4526\ -0.2994]^T;\boldsymbol{d}_3=[0.0333\ -0.0005\ 0.9994]^T;\boldsymbol{p}=[-0.4785\ 0.2568\ 0.5178]^T$
9	$\boldsymbol{d}_1=[0.8043\ -0.5200\ -0.2875]^T;\boldsymbol{d}_2=[0.5939\ 0.7178\ 0.3634]^T;\boldsymbol{d}_3=[0.0174\ -0.4630\ 0.8818]^T;\boldsymbol{p}=[-0.3369\ -0.7568\ 0.1558]^T$
10	$\boldsymbol{d}_1=[-0.3562\ -0.7886\ 0.5012]^T;\boldsymbol{d}_2=[-0.7050\ 0.5788\ 0.4098]^T;\boldsymbol{d}_3=[-0.6132\ -0.2074\ -0.7621]^T;\boldsymbol{p}=[-0.1981\ -0.3276\ 0.3224]^T$

Table 3.7 Distribution of the stop iteration generation of 100 operations of each algorithm.

Interval of iteration generation	[0, 10]	(10, 20]	(20, 30]	(30, 40]	(40, 50]	(50, 60)	*G*=60
EE-EDA/DM	0	23	36	28	10	1	2
ES	0	0	0	0	0	0	100
ELPSO	0	0	2	43	19	1	35
IDEA	0	0	0	0	0	0	100

Fig. 3.12 presents the experimental results of these four heuristic optimization algorithms. It is seen that only two operation results of EE-EDA/DM exceed 10^{-5}. Also, 35 operation results of ELPSO are above 10^{-5} and every point violated the failure test. No result of ES or IDEA is below 10^{-5}. Table 3.7 shows the stop generations in which the algorithms reach an SE. All operations of ES and IDEA have stopped in the 60th generation and failed to obtain an SE. If the generations of IDEA are increased to about 200, it usually can achieve an error less than 10^{-5}. But even if ES

increases its iteration generations, it still cannot achieve better performance. Hence, ES is not suitable for the present optimization problems. EE-EDA/DM usually reaches an acceptable error in an early generation. It has 59 operations reaching an SE before the 31st generation while ELPSO just has 2 operations. This is because EE-EDA/DM uses a larger population size (*PS*=2500) than by ELPSO (*PS*=350), so it has a higher probability of acquiring a better result in the early phase of the evolution. However, if the population size for ELPSO is increased, ELPSO still cannot find a better solution in an early generation or have a high successful rate of finding an SE, but can improve the accuracy of some results. ES and IDEA (*PS*=100) have shown similar performance.

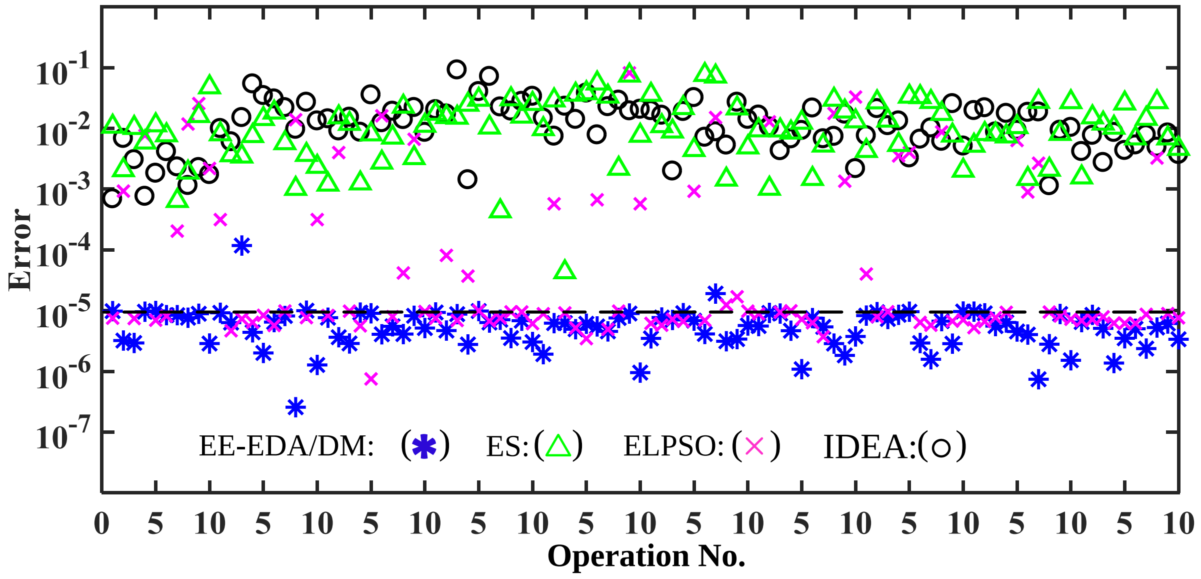


Figure 3.12 Distribution of final error of 100 tests of EE-EDA/DM, ES, ELPSO and IDEA.

Fig. 3.13 presents the computing time. If EE-EDA/DM runs most of the generations in one operation, it spends more time than that of IDEA and ELPSO. This is true since it has a larger population. However, the average computing time of EE-EDA/DM is less than that of ES, ELPSO and IDEA. This is because EE-EDA/DM is more stable than the other three algorithms in the present problem, and most operations stopped running before the 60th generation while reaching an acceptable error. In summary, EE-EDA/DM is more efficient and stable than ES, ELPSO and IDEA for the present optimization problem.

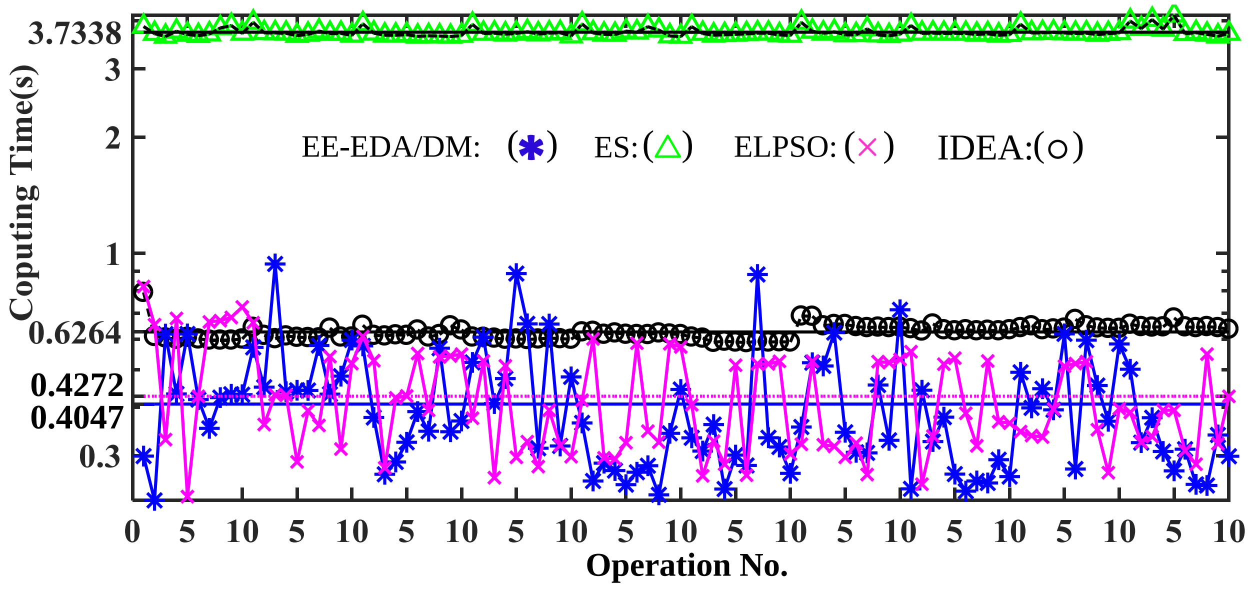


Figure 3.13 Distribution of the computing time of EE-EDA/DM, ES, ELPSO and IDEA.

3.4 EE-EDA/DM for solving IDP of continuous trajectory

All the foregoing tests have used discrete and random points, but finding the IDP solution for the random points is not enough. For the algorithm to be used in a physical robot, it must be able to achieve the IDP solution for a desired continuous trajectory, because in a practical application the end tool should not be jerky and jump rapidly from one point to the next. Here the curve shown in Fig. 3.14 is used as the test trajectory. The end tool of the 7-DOF Barrett WAM Arm is made to move along this test trajectory. The curve is discretized to 61 desired points. Every point has the same orientation, but the position changes according to this curve.

$$\text{Orientation:}\ \boldsymbol{d}_1=[1\quad 0\quad 0]^T;\ \boldsymbol{d}_2=[0\quad 1\quad 0]^T;\ \boldsymbol{d}_3=[0\quad 0\quad 1]^T.$$

Curve function: $x = 3t^2 - 2t;\ y = t^2 - t;\ z = 1.5t^2 - t,\ t \in [0.15\ \ 0.75]$.

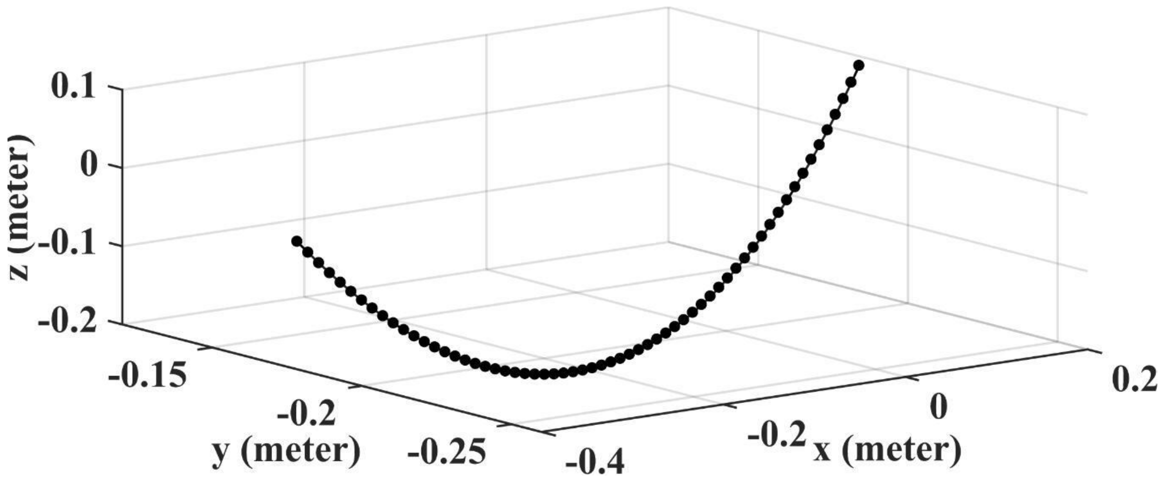


Figure 3.14 Desired continuous trajectory.

In order to avoid choppy angle values for the robot joints, when the IDP solution of the i^{th} (i=1, 2,⋯, 60) desired point is obtained, this solution is used as the mean of the Gaussian model to produce the initial population for the $(i+1)^{th}$ point, and a small variance (σ^2=0.2) is adopted. This makes the IDP solution of the $(i+1)^{th}$ point change slightly compared to the solution of the i^{th} point. If each desired point restarts search from a random initial population every time, the final IDP solutions of these points may change significantly from each other, which is not desirable in manipulator control. Besides, from the 2nd desired point, the population size is decreased to 500 since the initial population has already approached the solution adequately and it is not necessary to maintain a large population size as before. This operation reduces the computational time.

Table 3.8 shows the stop iteration generations in which the algorithm reached an SE. One operation obtained the SE in the 41st generation. In fact, this is the operation of the 1st desired point and it has searched the solution from a random initial population, so it needed more iteration generations. However, the other operations have started the search from a known direction, so they could find an SE in very early generations (all before the 7th generation).

Table 3.8 Distribution of stop iteration generation of the operations for the 61 desired points.

Interval of iteration generation	[0, 2]	G=3	G=4	G=5	G=6	[7, 40]	G=41	[42, 60]
Times of operation	0	3	45	11	1	0	1	0

The algorithm has been implemented in MATLAB and the total computing time of finding the IDP solutions for the 61 desired points is about 3.656186s. Except for the 1st desired point, the average real computational time of the other 60 points is about 0.053796s. If the algorithm was implemented in C++ and in a high-performance computer, the computational time could be reduced further, facilitating real time application. The computational times of some other algorithms in finding an SE for one desired pose are listed as well. Kucuk and Bingul proposed a new inverse kinematics algorithm (NIKA) for robot manipulators that did not have closed form solutions [17]. In their experimental results, the average computational time of obtaining an SE for a 6-DOF manipulator was about 0.140876s. This time was obtained on a 2.8 GHz Pentium i7 computer with 8 GB RAM memory. Ayyıldız and Çetinkaya solved the IDP for a real 4-DOF serial robot manipulator, where the quantum particle swarm optimization (QPSO) had the best performance, compared to other heuristic optimization algorithms [49]. Its average computing time of finding an SE with an error less than 10^{-5} was about 2s on a 3.1 GHz Pentium 4 computer with 4 GB RAM memory. Köker developed an algorithm that combined neural networks and genetic algorithms to solve the IDP of a six-joint Stanford robotic manipulator [98]. The execution time of this algorithm in finding an SE was approximately 0.28s using a six-core Intel Xeon 2.40-GHz computer workstation.

Fig. 3.15 shows the IDP solution of each joint for the desired continuous trajectory. The angle values of all joints had a gradual variation, as in a practical situation of robot control. Actually, each point of the trajectory had infinite IDP solution, because this robotic arm is 7-DOF and redundant. If the error of the solution is less than the threshold and does not change extensively (compared to the solution of the previous point in the trajectory), the solution is accepted to be one of the infinite solutions.

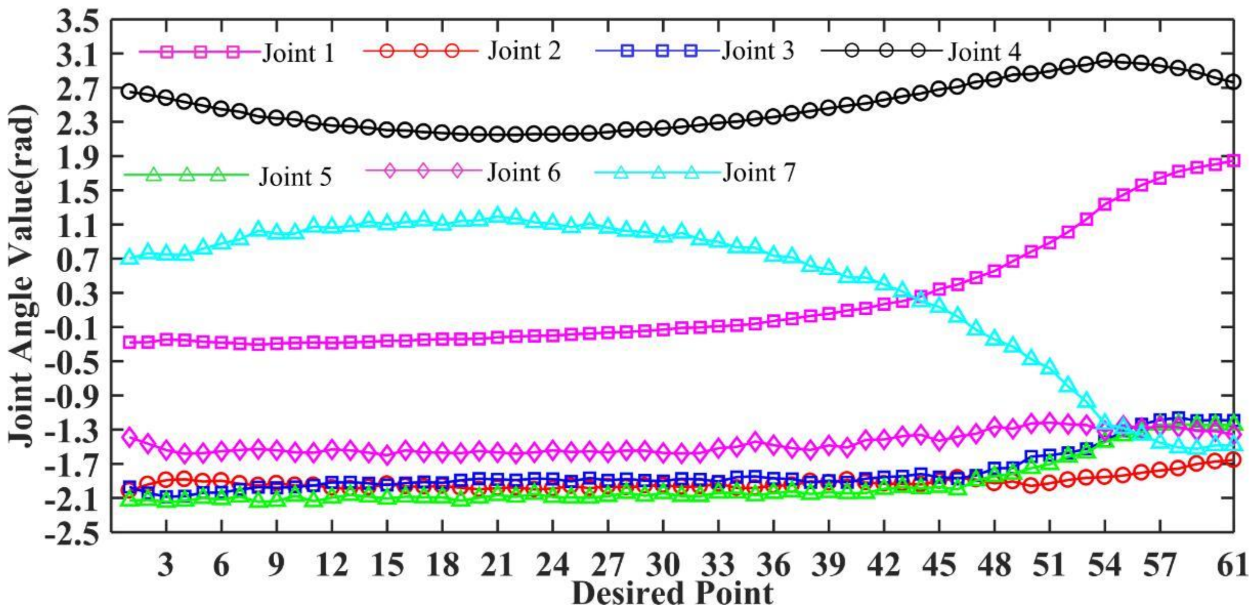


Figure 3.15 The IDP solution of each joint for the desired continuous trajectory.

In view of the no-free-lunch theorem, there is no general-purpose, universal optimization strategy that can always have good performance for all problems or for all aspects. The EE-EDA/DM, which has been developed in the present work, also has some limitations. When building the probabilistic model, the proposed approach adopts the univariate marginal Gaussian distribution. This model assumes that the problem variables are probabilistically independent and there is no interaction between the variables. The advantage of this model is that the computational cost of learning the model and sampling the offspring individuals is lower, compared to the models that incorporate relations between variables. This model enables the algorithm to obtain an SE at high speed, thereby facilitating real time application.

However, some other optimization problems, such as trap-5 and Summation cancellation, require a probabilistic model involving the interaction between variables. Besides, when EE-EDA/DM produces the individuals for the next generation with the univariate marginal Gaussian distribution model, the offspring of each variable (the joint angle) is sampled along the respective dimension of the variable, and every variable does not relate to any other variable. Essentially, during the sampling there is no constraint on a problem variable except for its lower and upper bounds. Other constraints, including inequality or equality constraints, may be incorporated into the robotic inverse displacement problem, by enhancing the current probabilistic model. The solution must be located in a feasible region, which is usually smaller than the search region when the variables have lower and upper bounds only as their constraints. For example, suppose that the problem contains equality constraints or the optimal solutions are located at the intersection of some constraints. Then when one variable varies, the other variables must vary correspondingly. Otherwise, the search may deviate from the feasible region. Then the interaction between variables should be included in the probabilistic model.

The inverse displacement problem seeks a satisfactory solution with sufficiently low error ($< 10^{-5}$). It is not necessarily the global optimal solution, which simplifies the solution to some extent. If further reduction in error is needed, the proposed algorithm has to use a more diversified population. The extreme elitism selection sacrifices the population diversity to some extent to obtain fast convergence. Selecting some of the same leading best individuals from the parent population makes the variance of the Gaussian model decrease rapidly. As a result, the algorithm can converge to an acceptable solution fast.

3.5 Summary

In this chapter, estimation distribution algorithm was first applied to the inverse displacement problem of robotic manipulators. The EDA based on univariate marginal Gaussian distribution with extreme elitism selection can have a fast convergence rate. EE-EDA was able to solve IDP for the 4-DOF Barrett WAM arm more stably and rapidly than the EDAs with truncation, proportional and tournament selection methods. Then EE-EDA was incorporated with differential mutation to solve the 7-DOF arm. After differential mutation operation, by directly making the 1st best solutions as the mean vector of Gaussian model and using a somewhat smaller variance than that of the last generation to produce the offspring, this novel scheme achieved the ability of reaching a satisfactory error limit in the evolution. Then, EE-EDA/DM was compared to the other three heuristic optimization algorithms: ES, ELPSO and IDEA by testing 10 random desired points. The test results showed that EE-EDA/DM was more efficient and stable in finding the IDP solution of the 7-DOF arm. EE-EDA/DM adopted a univariate marginal Gaussian distribution model, which did not consider the interaction between variables. If the IDP involves some inequality or equality constraints, this model should be enhanced appropriately.

Chapter 4: Estimation Distribution Algorithms for Constrained Optimization Problems

4.1 Introduction

Many real world optimization problems in the science and engineering disciplines involve different types of constraints, like equality, inequality, linear, nonlinear, continuous, or discontinuous. The conventional methods of handling constrained optimization problems (COPs) such as feasible direction methods, projection gradient methods, interior point and exterior point penalty function methods usually need the objective and constrained functions to be continuous and differentiable. In the past two decades, many evolutionary algorithms have been proposed to solve COPs. These approaches do not concern mathematical properties such as continuity or differentiability and can search the solutions in a large space and in parallel.

TFGA is a two-phase GA proposed by Venkatraman and Yen [50]. This GA first searches the feasible region and then the COPs are treated as a bi-objective optimization problem to be solved in the second phase. Wang et al. developed an evolution strategy based on an adaptive trade-off model (ATMES) [51]. This approach sought to strike an appropriate trade-off between the objective function and constraint violations during different search stages and achieve a diverse and robust search. Montes and Domínguez proposed a modified artificial bee colony (M-ABC) algorithm to handle COPs. Compared to the original ABC, this M-ABC replaced the proportional selection with a binary tournament selection. Also, it used a smart flight operator to facilitate the location of food sources as convenient attractors in a constrained search space [52]. Yang et al. integrated PSO with GA (PSGA) for COPs. The PSO phase improved the worst solutions by using the global-local best inertia weight and acceleration coefficients, and the GA adopted a rank-based multi-parent crossover to favour both local and global explorations simultaneously [53]. Several other techniques are available, as can be seen in the bibliography [99-104].

With regard to estimation distribution algorithms (EDAs), to date only a few published papers have studied this topic of handling COPs by EDAs. Grahl and Rothlauf [19] developed a PolyEDA which adopted the method of Gibbs sampling to deal with linear inequality constraints. The problem of this EDA is that it can only handle linear constraints. In the experiments, only the problem of the Rosenbrock function has been used to test the performance, and the constraints were just the upper and lower bounds of the variables. Simionescu et al. [20] used the penalty and repair techniques to assist EDA in solving the COPs. However, they only used three problems to test the performance of the proposed algorithm. In addition, the constraints of these problems do not contain equality constraints. So their study results cannot be representative. Wan et al. [21] introduced EDAs to solve a class of nonlinear bi-level programming problems. They first transformed these problems into COPs in light of the Karush–Kuhn–Tucker (KKT) conditions and then adopted the penalty function method to handle the constraints. These optimization problems must satisfy the KKT conditions. Moreover, they set the penalty parameters uniformly as the constant 10000 for all problems. However, it is known that the appropriate penalty parameter values are problem-dependent. Furthermore, too large penalty parameter values can produce a high selection pressure and make the algorithm converge to some unsatisfactory solutions while too small parameter values may lead to a broad search region and a slow convergence speed. So the developed EDAs can only deal with some classes of COPs and there are still some limitations for these EDAs. Therefore, it is necessary to improve the capability of EDAs in handling various types of constraints and use more and representative benchmark problems to test the performance of EDAs in solving COPs.

4.2 EDAs based on Gaussian probabilistic models

In the present study, the EDAs based on five different Gaussian distributions in the continuous domain are implemented to solve COPs.

(a) Univariate marginal Gaussian distribution (UMGD) where the problem variables are independent and there is no interaction between variables. The mean, variance and the joint probability density function (JPDF) can be expressed as

$$\mu_i = \frac{\sum_{j=1}^{M} x_{ij}}{M} \qquad \sigma_i^2 = \frac{\sum_{j=1}^{M} (x_{ij} - \mu_i)^2}{M-1} \qquad (i = 1, 2, \cdots, D;\ j = 1, 2, \cdots, M) \tag{4.1}$$

$$f_{\boldsymbol{X}}(x_1, \cdots, x_D) = \prod_{i=1}^{D} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}\right) \qquad \boldsymbol{X} = (x_1, \cdots, x_D)^T \tag{4.2}$$

(b) Multivariate Gaussian distribution (MVGD).The variables in this distribution are dependent of each other. The covariance matrix and the covariance formula are given by:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \mathrm{cov}(x_1, x_2) & \cdots & \mathrm{cov}(x_1, x_D) \\ \mathrm{cov}(x_2, x_1) & \sigma_2^2 & \cdots & \mathrm{cov}(x_2, x_D) \\ \vdots & \vdots & & \vdots \\ \mathrm{cov}(x_D, x_1) & \mathrm{cov}(x_D, x_2) & \cdots & \sigma_D^2 \end{bmatrix} \tag{4.3}$$

$$\mathrm{cov}(x_i, x_k) = \frac{\sum_{j=1}^{M} (x_{ij} - \mu_i)(x_{kj} - \mu_k)}{M-1} \qquad (i = 1, \cdots, D; k = 1, \cdots, D; i \neq k) \tag{4.4}$$

The JPDF of MVGD is expressed as

$$f_{\boldsymbol{X}}(x_1, \cdots, x_D) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{X} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{X} - \boldsymbol{\mu})\right] \qquad \boldsymbol{\mu} = (\mu_1, \cdots, \mu_D)^T \tag{4.5}$$

(c) A mixture of univariate marginal Gaussian distributions (MIXUMGD) [68]. The JPDF of the distribution can be expressed as

$$f_{\boldsymbol{X}}(x_1,\cdots,x_D)=\sum\nolimits_{w=1}^{W}\pi_w N(\boldsymbol{X},\boldsymbol{\mu}_w,\boldsymbol{\sigma}_w) \quad \sum\nolimits_{w=1}^{W}\pi_w=1 \quad \boldsymbol{\sigma}=(\sigma_1,\cdots,\sigma_D)^T \tag{4.6}$$

Here W is the number of marginal Gaussian distributions and π_w is the weight of each N ($\boldsymbol{X}$, $\boldsymbol{\mu}_w$, $\boldsymbol{\sigma}_w$) that denotes one UMGD.

(d) A mixture of multivariate Gaussian distributions algorithm (MIXMVGD). This is similar to MIXUMGD, but it consists of a combination of several MVGD. Its JPDF is given by

$$f_{\boldsymbol{X}}(x_1,\cdots,x_D)=\sum\nolimits_{w=1}^{W}\pi_w N(\boldsymbol{X},\boldsymbol{\mu}_w,\Sigma_w) \qquad \sum\nolimits_{w=1}^{W}\pi_w=1 \tag{4.7}$$

Here, W is the number of Gaussian distributions, and π_w is the weight of each N ($\boldsymbol{X}$, $\boldsymbol{\mu}_w$, $\sum_w$) that represents one MVGD.

(e) Gaussian network distribution [105]. The JPDF of distribution is given by

$$f_{\boldsymbol{X}}(x_1,\cdots,x_D)=\prod_{i=1}^{D} f(x_i \mid x_1,\cdots,x_{i-1})=\prod_{i=1}^{D} N(x_i;\mu_i+\sum_{k=1}^{i-1} b_{ki}(x_k-\mu_k),v_i) \tag{4.8}$$

where μ_i is the unconditional mean of x_i, v_i is the variance of x_i (v_i=σ^2), and b_{ki} is a linear coefficient reflecting the strength of the relationship between the variables x_k and x_i. For any $b_{ki} \neq 0$ with $k < i$, the network will contain an arc from x_k to x_i. For instance, Fig. 4.1 shows a Gaussian network, where the variable x_3 has two parents, x_1 and x_2, x_4 has one parent x_3, while x_1 or x_2 has no parent. The JPDF of this network can be described by

$$f_{\boldsymbol{X}}(x_1,x_2,x_3,x_4)=f(x_1)f(x_2)f(x_3|x_1,x_2)f(x_4|x_3) \quad \boldsymbol{X}=(x_1,x_2,x_3,x_4)^T \tag{4.9}$$

Here an EDA named estimation of Gaussian networks algorithm (EGNA$_{BIC}$) is compared with other EDAs. This EDA adopts the score of Bayesian Information Criterion (BIC) to build a Gaussian network [29].

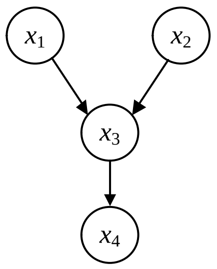
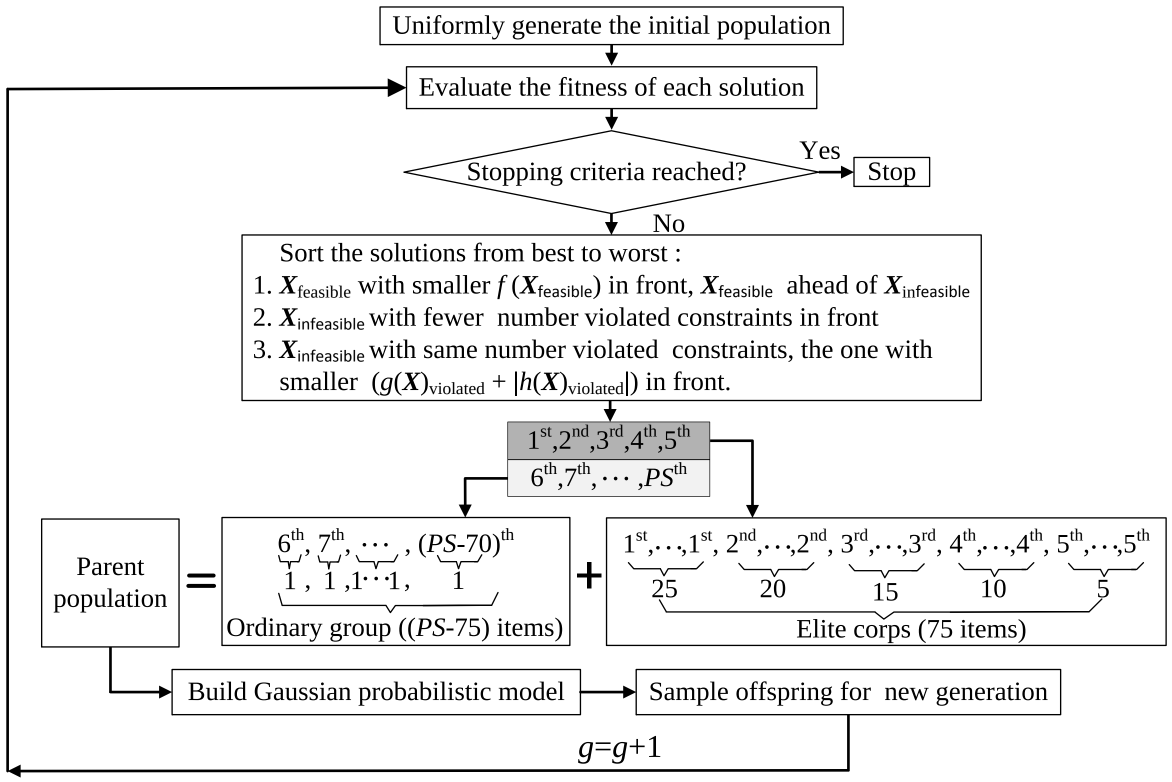


Figure 4.1 An example of Gaussian network.

4.3 Solution of COPs using EDAs with extreme elitism selection

Fig. 4.2 presents the flowchart of the EDAs that solves COPs with the method of extreme elitism selection. This selection process is divided into two steps for COPs. The first step sorts the solutions in the current generation and the second step chooses some solutions to form the parent population. The sorting criteria are as follows: The feasible solutions with smaller objective function values are in front; the feasible solutions are always ahead of infeasible solutions; the infeasible solutions with fewer number of violated constraints are in front; and if two infeasible solutions violate the same number of constraints, the one with the smaller value of $(g(\boldsymbol{X})_{\text{violated}} + |h(\boldsymbol{X})_{\text{violated}}|)$ is in front. In light of these criteria, the solutions are sorted from the best to the worst. Then a few leading best solutions are chosen as the elites, and they account for more portions than other ordinary promising solutions in the parent population. In Fig. 4.2, the 1st to the 5th top solutions are allocated a quota in the order according to an arithmetic sequence, such as 25, 20, 15, 10 and 5. Then from the 6th to the $(PS\text{-}70)^{\text{th}}$ ordinary solutions account for only 1 item each uniformly in the parent population. The EDAs with this extreme elitism selection process can handle various types of constraints. In addition, this method does not ask for the information of continuity and gradient or setting the penalty parameters.



Uniformly generate the initial population
Evaluate the fitness of each solution
Stopping criteria reached?
Yes
Stop
No
Sort the solutions from best to worst :
1. X_feasible with smaller f (X_feasible) in front, X_feasible ahead of X_infeasible
2. X_infeasible with fewer number violated constraints in front
3. X_infeasible with same number violated constraints, the one with smaller (g(X)_violated + |h(X)_violated|) in front.
1st,2nd,3rd,4th,5th
6th,7th,⋯ ,PSth
Parent population
6th, 7th, ⋯ , (PS-70)th
1 , 1 ,1⋯1, 1
Ordinary group ((PS-75) items)
1st,⋯,1st, 2nd,⋯,2nd, 3rd,⋯,3rd, 4th,⋯,4th, 5th,⋯,5th
25 20 15 10 5
Elite corps (75 items)
Build Gaussian probabilistic model
Sample offspring for new generation
g=g+1

Figure 4.2 Flowchart of the EDAs that solve COPs with extreme elitism selection.

Sometimes, the optimum solutions of the COPs may be located at the boundary of the feasible region, and some infeasible solutions also can provide useful information in the search. For instance, in Fig. 4.3, g_1 and g_2 are the inequality constraints, and h_3 is an equality constraint. The part of h_3 between g_1 and g_2 (green part) is the feasible region, (x_{11}, x_{21}) is an infeasible solution, and (x_{12}, x_{22}) and (x_{13}, x_{23}) are feasible solutions. Then the mean value of ((x_{11}+ x_{13})/2, (x_{21}+ x_{23})/2) is closer to the optimal solution (x_{1op}, x_{2op}) than the mean value of ((x_{12}+ x_{13})/2, (x_{22}+ x_{23})/2). If the Gaussian search starts from ((x_{11}+ x_{13})/2, (x_{21}+ x_{23})/2), the algorithm is more likely to reach the point (x_{1op}, x_{2op}) than if searched from ((x_{12}+ x_{13})/2, (x_{22}+ x_{23})/2). It is seen that the infeasible solution (x_{11}, x_{21}) also sometimes can play an important role in searching the optimal solution. So in this study, the parent population size is increased to the same size as the original population. Increasing the size of the parent population can improve the probability of choosing some useful infeasible solutions as the parent. This operation can also make more ordinary solutions join the parent population to increase the variance of each variable. The variance represents the step size of the exploitation in a Gaussian search. A large variance can sample more diversified offspring and make the algorithms avoid premature convergence.

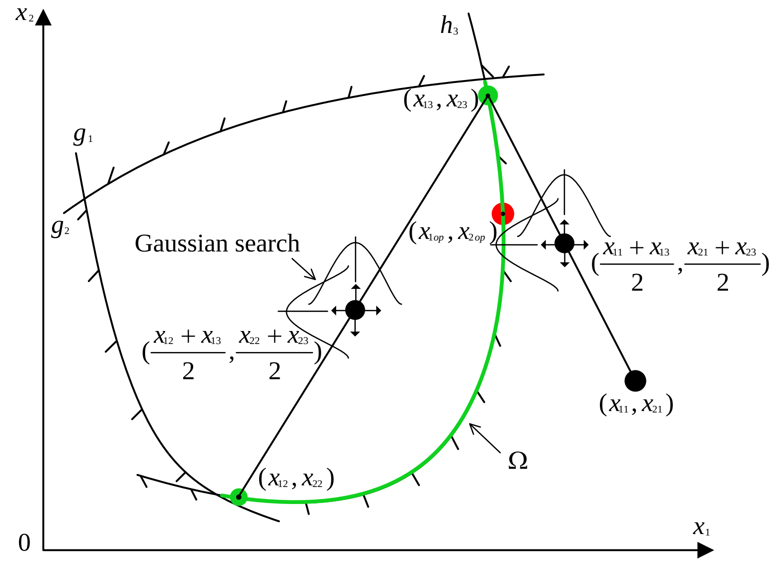


Figure 4.3 Gaussian search and feasible region.

4.4 Experimental results and analysis

4.4.1 The performance evaluation of EDAs for 13 benchmark COPs

In the present experiments, 13 benchmark problems are used to evaluate the performance of the EDAs. These problems are widely used to test the performance of other evolution algorithms [50-53, 99-101]. The expressions of these problems are presented in Table 4.1, where "Min" donates a minimization problem, "Sub" means subject to these constraints, and $\boldsymbol{X}_{optimal}$ is the optimal solution found thus far. The population size *PS* is set at 1000 for all the algorithms of all the problems, and the parent population size *M* also equals 1000. When the problems have equality constraints, they are transformed into inequality constraints as $|h_j(\boldsymbol{X})| - \varepsilon \leq 0$, where ε is the allowed tolerance. Commonly, ε is set at a constant value like 10^{-5}. However, this operation restricts the search to a small region from the beginning and reduces the population diversity; as a result, the algorithms sometimes find the feasible region with difficulty or converge to some local optima. So, ε is set equal to [$1 / (1.02^g)$], where 1.02 is an experience-based value and g is the generation number. Also, ε goes down as g goes up, and the search area gradually changes from large to small. Meanwhile, the algorithms can also take advantage of some helpful infeasible solutions. In the next section, the results of the algorithms with a constant tolerance and a dynamic tolerance are compared when solving the COPs involving equality constraints.

Table 4.1 Thirteen benchmark COPs.

P1	Min	$f(\mathbf{X}) = x_1^2+x_2^2+x_1x_2-14x_1-16x_2+(x_3-10)^2+4(x_4-5)^2+(x_5-3)^2+2(x_6-1)^2+5x_7^2+7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2+45$
		$g_1(\boldsymbol{X}) = -105+4x_1+5x_2-3x_7+9x_8\leq0$ $\quad g_2(\boldsymbol{X}) = 10x_1-8x_2-17x_7+2x_8\leq0$
	Sub	$g_3(\boldsymbol{X}) = -8x_1+2x_2+5x_9-2x_{10}-12\leq0$ $\quad g_4(\boldsymbol{X}) = 3(x_1-2)^2+4(x_2-3)^2+2x_3^2-7x_4-120\leq0$
		$g_5(\boldsymbol{X}) = 5x_1^2+8x_2+(x_3-6)^2-2x_4-40\leq0$ $\quad g_6(\boldsymbol{X}) = 0.5(x_1-8)^2+2(x_2-4)^2+3x_5^2-x_6-30\leq0$
		$g_7(\boldsymbol{X}) = x_1^2+2(x_2-2)^2-2x_1x_2+14x_5-6x_6\leq0$ $\quad g_8(\boldsymbol{X}) = -3x_1+6x_2+12(x_9-8)^2-7x_{10}\leq0$ $\quad -10\leq x_i \leq10$ $(i=1,\cdots, 10)$
		$\boldsymbol{X}_{optimal}$ = [2.17199634; 2.36368304; 8.77392574; 5.09598444; 0.99065476; 1.43057393; 1.32164415; 9.82872577; 8.28009159; 8.37592665] and $f(\boldsymbol{X}_{optimal}) = 24.30621$.
P2	Min	$f(\mathbf{X}) = 5.3578547x_3^2+0.8356891x_1x_5+37.293239x_1-40792.141$
		$g_1(\boldsymbol{X}) = 85.334407+0.0056858x_2x_5+0.0006262x_1x_4-0.0022053x_3x_5-92\leq0$
		$g_2(\boldsymbol{X}) = -85.334407-0.0056858x_2x_5-0.00062624x_1x_4+0.0022053x_3x_5\leq0$
		$g_3(\boldsymbol{X}) = 80.51249+0.0071317x_2x_5+0.0029955x_1x_2+0.0021813x_3^2-110\leq0$
	Sub	$g_4(\boldsymbol{X}) = -80.51249-0.0071317x_2x_5-0.0029955x_1x_2-0.0021813x_3^2+90\leq0$
		$g_5(\boldsymbol{X}) = 9.300961+0.0047026x_3x_5+0.0012547x_1x_3+0.0019085x_3x_4-25\leq0$
		$g_6(\boldsymbol{X}) = -9.300961-0.0047026x_3x_5-0.0012547x_1x_3-0.0019085x_3x_4+20\leq0$
		$78\leq x_1 \leq102$, $33\leq x_2 \leq45$, $27\leq x_i \leq45$ $(i=3, 4, 5)$
		$\boldsymbol{X}_{optimal}$ = [78; 33; 29.995256; 45; 36.775813] and $f(\boldsymbol{X}_{optimal}) = -30665.53867$.
P3	Min	$f(\mathbf{X}) = (x_1-10)^2+5(x_2-12)^2+ x_3^4+3(x_4-11)^2+ 10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$
	Sub	$g_1(\boldsymbol{X}) = -127+2x_1^2+3x_2^4+x_3+4x_4^2+5x_5\leq0$ $\quad g_2(\boldsymbol{X}) = -282+7x_1+3x_2+10x_3^2+x_4-x_5\leq0$
		$g_3(\boldsymbol{X}) = -196+23x_1+x_2^2+6x_6^2-8x_7\leq0$ $\quad g_4(\boldsymbol{X}) = 4x_1^2+x_2^2-3x_1x_2+2x_3^2+5x_6-11x_7\leq0$ $\quad -10\leq x_i \leq10$ $(i=1,\cdots, 7)$
		$\boldsymbol{X}_{optimal}$ = [2.33049935; 1.95137237; -0.47754140; 4.36572625; -0.62448696; 1.03813099; 1.59422668] and $f(\boldsymbol{X}_{optimal}) =680.6300574$.

P4	Min	$f(\mathbf{X}) = -\sin^3(2\pi x_1)\sin(2\pi x_2)/[x_1^3(x_1+x_2)]$		
	Sub	$g_1(\boldsymbol{X}) = x_1^2-x_2+1\leq 0$	$g_2(\boldsymbol{X}) = 1-x_1+(x_2-4)^2\leq 0$	$0\leq x_1 \leq 10$, $0\leq x_2 \leq 10$
		$\boldsymbol{X}_{optimal}$ = [1.2279713526; 4.2453733661)] and $f(\boldsymbol{X}_{optimal})$ =-0.095825041.		
P5	Min	$f(\boldsymbol{X}) = 0.000117y_{14}+0.1365+0.00002358y_{13}+0.000001502y_{16}+0.0321y_{12}+0.004324y_5+0.0001c_{15}/c_{16}$ $+37.48y_2/c_{12}-0.0000005843y_{17}$		
	Sub	$g_1(\boldsymbol{X}) = (0.28/0.72)y_5-y_4\leq 0$	$g_2(\boldsymbol{X}) = x_3-1.5x_2\leq 0$	$g_3(\boldsymbol{X}) =3496y_2/c_{12}-21\leq 0$
		$g_4(\boldsymbol{X}) = 110.6+y_1-62212/c_{17}\leq 0$	$g_5(\boldsymbol{X}) = 213.1-y_1\leq 0$	$g_6(\boldsymbol{X}) = y_1-405.23\leq 0$
		$g_7(\boldsymbol{X}) = 17.505-y_2 \leq 0$	$g_8(\boldsymbol{X}) = y_2-1053.6667 \leq 0$	$g_9(\boldsymbol{X}) = 11.275-y_3 \leq 0$
		$g_{10}(\boldsymbol{X}) = y_3-35.03\leq 0$	$g_{11}(\boldsymbol{X}) = 214.228-y_4\leq 0$	$g_{12}(\boldsymbol{X}) = y_4-665.585\leq 0$
		$g_{13}(\boldsymbol{X}) = 7.458-y_5 \leq 0$	$g_{14}(\boldsymbol{X}) = y_5-584.463 \leq 0$	$g_{15}(\boldsymbol{X}) = 0.961-y_6 \leq 0$
		$g_{16}(\boldsymbol{X}) = y_6-265.916\leq 0$	$g_{17}(\boldsymbol{X}) = 1.612-y_7\leq 0$	$g_{18}(\boldsymbol{X}) = y_7-7.046\leq 0$
		$g_{19}(\boldsymbol{X}) = 0.146-y_8 \leq 0$	$g_{20}(\boldsymbol{X}) = y_8-0.222 \leq 0$	$g_{21}(\boldsymbol{X}) = 107.99-y_9 \leq 0$
		$g_{22}(\boldsymbol{X}) = y_9-273.366\leq 0$	$g_{23}(\boldsymbol{X}) = 922.693-y_{10}\leq 0$	$g_{24}(\boldsymbol{X}) = y_{10}-1286.105\leq 0$
		$g_{25}(\boldsymbol{X}) = 926.832-y_{11} \leq 0$	$g_{26}(\boldsymbol{X}) = y_{11}-1444.046 \leq 0$	$g_{27}(\boldsymbol{X}) = 18.766-y_{12} \leq 0$
		$g_{28}(\boldsymbol{X}) = y_{12}-537.141\leq 0$	$g_{29}(\boldsymbol{X}) = 1072.163-y_{13}\leq 0$	$g_{30}(\boldsymbol{X}) = y_{13}-3247.039\leq 0$
		$g_{31}(\boldsymbol{X}) = 8961.448-y_{14} \leq 0$	$g_{32}(\boldsymbol{X}) = y_{14}-26844.086 \leq 0$	$g_{33}(\boldsymbol{X}) = 0.063-y_{15} \leq 0$
		$g_{34}(\boldsymbol{X}) = y_{15}-0.386\leq 0$	$g_{35}(\boldsymbol{X}) = 71084.33-y_{16}\leq 0$	$g_{36}(\boldsymbol{X}) = -140000+y_{16}\leq 0$
		$g_{37}(\boldsymbol{X}) = 2802713-y_{17} \leq 0$	$g_{38}(\boldsymbol{X}) = y_{17}-12146108 \leq 0$	$y_1 = x_2+x_3+41.6$
		$c_1 = 0.024x_4-4.62$ $y_2 = 12.5/c_1+12$ $c_2 = 0.0003535x_2 +0.5311x_1+0.08705y_2x_1$ $c_5=100x_2$ $y_5=c_6c_7$		
		$c_3=0.052x_1+78+0.002377y_2x_1$ $y_6=x_1-y_3-y_4$ $c_4=0.04782(x_1-y_3)+0.1956(x_1-y_3)^2/x_2+0.6376y_4+1.594y_3$		
		$c_8=0.995(y_5+y_4)$ $y_7=c_8/y_1$ $y_8=c_8/3798$ $c_9=y_7-0.0663y_7/y_8-0.3153$ $y_9=96.82/c_9+0.321y_1$		
		$y_{10}=1.29y_5+1.258y_4+2.29y_3+1.71y_6$ $c_{10}=12.3/752.3$ $c_{11}=(1.75y_2)(0.995x_1)$ $c_{12}=0.995y_{10}+1998$		
		$y_{12}=c_{10}x_1+c_{11}/c_{12}$ $y_{13}=c_{12}-1.75y_2$ $y_{15}=y_{13}/c_{13}$ $y_{16}=148000-331000y_{15}+40y_{13}-61y_{15}y_{13}$ $c_{17}=y_9+x_5$		
		$y_{14}=3623+64.4x_2+58.4x_3+58.4x_3+146312/(y_9+x_5)$ $c_{16}=1.104-0.72y_{15}$ $c_{15}=y_{13}/y_{15}-y_{13}/0.52$		
		$c_{14}=2324y_{10}-28740000y_2$ $y_{17}=14130000-1328y_{10}-531y_{11}+c_{14}/c_{12}$		
		$704.4148\leq x_1 \leq 906.3855$, $68.6\leq x_2 \leq 288.88$, $0\leq x_3 \leq 134.75$, $193\leq x_4 \leq 287.0966$, $25\leq x_5 \leq 84.1988$		
		$\boldsymbol{X}_{optimal}$ = [705.174537070; 68.6; 102.9. 282.324931594; 37.584116426] and $f(\boldsymbol{X}_{optimal})$ =-1.90515526.		
P6	Min	$f(\boldsymbol{X}) = (x_1-10)^3+(x_2-20)^3$		
	Sub	$g_1(\boldsymbol{X}) = -(x_1-5)^2-(x_2-5)^2+100\leq 0$	$g_2(\boldsymbol{X}) = (x_1-6)^2+(x_2-5)^2-82.81\leq 0$	$13\leq x_1 \leq 100$, $0\leq x_2 \leq 100$
		$\boldsymbol{X}_{optimal}$ = [14.095; 0.84296] and $f(\boldsymbol{X}_{optimal})$ = -6961.8138756.		
P7	Min	$f(\mathbf{X}) = x_1+x_2+ x_3$		
	Sub	$g_1(\boldsymbol{X}) = -1+0.0025(x_4+ x_6)\leq 0$	$g_2(\boldsymbol{X}) = -1+0.0025(x_5+ x_7- x_4)\leq 0$	$g_3(\boldsymbol{X}) = -1+0.01(x_8-x_5)\leq 0$
		$g_4(\boldsymbol{X}) = -x_1x_6+833.33252x_4+100x_1-83333.333\leq 0$	$g_5(\boldsymbol{X}) = -x_2x_7+1250x_5+x_2x_4-1250x_4\leq 0$	
		$g_6(\boldsymbol{X}) = -x_3x_8+1250000+x_3x_5-2500x_5\leq 0$ $100\leq x_1 \leq 10000$, $1000\leq x_i \leq 10000$ (i=2, 3), $10\leq x_i \leq 1000$ (i=4, ⋯, 8). $\boldsymbol{X}_{optimal}$ = [579.306685; 1359.970678; 5109.970657; 182.0176996; 295.601174; 217.982300; 286.416526; 395.601174] and $f(\boldsymbol{X}_{optimal})$ =7049.24802.		

P8	Min	$f(\boldsymbol{X}) = -0.5(x_1x_4-x_2x_3+x_3x_9-x_5x_9+x_5x_8-x_6x_7)$
	Sub	$g_1(\boldsymbol{X}) = x_3^2+x_4^2-1\leq 0$ $g_2(\boldsymbol{X}) = x_9^2-1\leq 0$ $g_3(\boldsymbol{X}) = x_5^2+x_6^2-1\leq 0$ $g_4(\boldsymbol{X}) = x_1^2+(x_2-x_9)^2-1\leq 0$ $g_5(\boldsymbol{X}) = (x_1-x_5)^2+(x_2-x_6)^2-1\leq 0$ $g_6(\boldsymbol{X}) = (x_1-x_7)^2+(x_2-x_8)^2-1\leq 0$ $g_7(\boldsymbol{X}) = (x_3-x_5)^2+(x_4-x_6)^2-1\leq 0$ $g_8(\boldsymbol{X}) = (x_3-x_7)^2+(x_4-x_8)^2-1\leq 0$ $g_9(\boldsymbol{X}) = x_7^2+(x_8-x_9)^2-1\leq 0$ $g_{10}(\boldsymbol{X}) = x_2x_3-x_1x_4\leq 0$ $g_{11}(\boldsymbol{X}) = -x_3x_9\leq 0$ $g_{12}(\boldsymbol{X}) = x_5x_9\leq 0$ $g_{13}(\boldsymbol{X}) = x_6x_7-x_5x_8\leq 0$ $-10\leq x_i \leq 10$ (i=1,⋯, 8), $0\leq x_9 \leq 20$.
		$\boldsymbol{X}_{optimal}$ = [-0.65777619; -0.15341877; 0.32341387; -0.94625761; -0.65777619; -0.75321343; 0.32341387; -0.34646295; 0.59979466] and $f(\boldsymbol{X}_{optimal})$ =-0.8660254.
P9	Min	$f(\mathbf{X}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$
	Sub	$g_1(\boldsymbol{X}) = 2x_1+2x_2+x_{10}+x_{11}-10 \leq 0$ $g_2(\boldsymbol{X}) = 2x_1+2x_3+x_{10}+x_{12}-10 \leq 0$ $g_3(\boldsymbol{X}) = 2x_2+2x_3+x_{11}+x_{12}-10 \leq 0$ $g_4(\boldsymbol{X}) = -8x_1+x_{10}\leq 0$ $g_5(\boldsymbol{X}) = -8x_2+x_{11}\leq 0$ $g_6(\boldsymbol{X}) = -8x_3+x_{12}\leq 0$ $g_7(\boldsymbol{X}) = -2x_4-x_5+x_{10}\leq 0$ $g_8(\boldsymbol{X}) = -2x_6-x_7+x_{11}\leq 0$ $g_9(\boldsymbol{X}) = -2x_8-x_9+x_{12}\leq 0$ $0\leq x_i \leq 1$ (i=1,⋯, 9), $0\leq x_i \leq 100$ (i=10, 11, 12), $0\leq x_{13}\leq 1$
		$\boldsymbol{X}_{optimal}$ = [1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 3; 3; 1] and $f(\boldsymbol{X}_{optimal})$ = -15.00.
P10	Min	$f(\mathbf{X}) = \sum_{i=1}^{10} x_i\left(c_i + \ln\frac{x_i}{\sum_{j=1}^{10} x_j}\right)$
	Sub	$h_1(\boldsymbol{X}) = x_1+2x_2+2x_3+x_6+x_{10}-2=0$ $h_2(\boldsymbol{X}) = x_4+2x_5+x_6+x_7-1=0$ $h_3(\boldsymbol{X}) = x_3+x_7+x_8+2x_9+x_{10}-1=0$ $0\leq x_i \leq 10$ (i=1, ⋯, 10) and c_1=-6.089, c_2=-17.164, c_3=-34.054, c_4=-5.914, c_5=-24.721, c_6=-14.986, c_7=-24.1, c_8=-10.708, c_9=-26.662, c_{10}=-22.179.
		$\boldsymbol{X}_{optimal}$ = [0.0406684; 0.1477212; 0.7832057; 0.0014143; 0.48529364; 0.0006932; 0.02740520; 0.0179510; 0.03732682; 0.0968845] and $f(\boldsymbol{X}_{optimal})$ = -47.764888.
P11	Min	$f(\boldsymbol{X}) = f_1(x_1)+f_2(x_2)$ $f_1(x_1) = \begin{cases} 30x_1 & 0\leq x_1 < 300 \\ 31x_1 & 300\leq x_1 < 400 \end{cases}$ $f_2(x_2) = \begin{cases} 28x_2 & 0\leq x_1 < 100 \\ 29x_2 & 100\leq x_1 < 200 \\ 30x_2 & 200\leq x_1 < 1000 \end{cases}$
	Sub	$h_1(\boldsymbol{X}) = -x_1+300-x_3x_4\cos(1.48477-x_6)/131.078+0.90798x_3^2\cos(1.47588)/131.078=0$ $h_2(\boldsymbol{X}) = -x_2-x_3x_4\cos(1.48477+x_6)/131.078+0.90798x_4^2\cos(1.47588)/131.078=0$ $h_3(\boldsymbol{X}) = -x_5-x_3x_4\sin(1.48477+x_6)/131.078+0.90798x_4^2\sin(1.47588)/131.078=0$ $h_4(\boldsymbol{X}) = 200-x_3x_4\sin(1.48477-x_6)/131.078+0.90798x_3^2\sin(1.47588)/131.078=0$ $0\leq x_1 \leq 400$, $0\leq x_2 \leq 1000$, $340\leq x_3 \leq 420$, $340\leq x_4 \leq 420$, $-1000\leq x_5 \leq 1000$ and $0\leq x_6 \leq 0.5236$.
		$\boldsymbol{X}_{optimal}$ = [201.7844672; 99.9999999999999; 383.0710349; 420; -10.9076585; 0.0731482] and $f(\boldsymbol{X}_{optimal})$ =8853.539675.
P12	Min	$f(\boldsymbol{X}) = 3x_1+0.000001x_1^3+2x_2+(0.000002/3)x_2^3$
	Sub	$g_1(\boldsymbol{X}) = -x_4+x_3-0.55\leq 0$ $g_2(\boldsymbol{X}) = -x_3+x_4-0.55\leq 0$ $h_3(\boldsymbol{X}) = 1000\sin(-x_3-0.25)+1000\sin(-x_4-0.25)+894.8-x_1=0$ $h_4(\boldsymbol{X}) = 1000\sin(x_3-0.25)+1000\sin(x_3-x_4-0.25)+894.8-x_2=0$ $0\leq x_1 \leq 1200$, $0\leq x_2 \leq 1200$, $-0.55\leq x_3 \leq 0.55$, $h_5(\boldsymbol{X}) = 1000\sin(x_4-0.25)+1000\sin(x_4-x_3-0.25)+1294.8=0$ $-0.55\leq x_4 \leq 0.55$. $\boldsymbol{X}_{optimal}$ = [679.9451483; 1026.0669760; 0.1188764; -0.3962335] and $f(\boldsymbol{X}_{optimal})$ = 5126.496714.

P13	Max Sub	$f(\mathbf{X}) = (100-(x_1-5)^2-(x_2-5)^2-(x_3-5)^2)/100$ $g(\boldsymbol{X}) = (x_1-p)^2+(x_2-q)^2+(x_3-r)^2-0.0625\le 0$ $0\le x_i \le 10$ (i=1, 2, 3) and p, q, r = 1, 2,⋯, 9. The feasible region consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds. $\boldsymbol{X}_{optimal}$ = [5; 5; 5] and $f(\boldsymbol{X}_{optimal})$ =1.

The maximum iteration generation G is set at 500 for most of the problems; and for P7 to P9, G = 700. For the problems containing equality constraints, G=600 and only the solutions that are obtained after the 582nd generation are considered as the valid solutions, since the allowed tolerance ε is 10^{-5}. With g = 582, the tolerance $[1 / (1.02^g)] = 9.87 \times 10^{-6}$ is less than 10^{-5}. The best solution within the 582nd to 600th generation is adopted as the final result in one test. Besides, the number of clusters in the EDAs with a mixture of univariate marginal Gaussian distributions (MIXUMGD) and a mixture of multivariate Gaussian distributions (MIXMVGD) are both set at 3. The distance of clustering uses the Euclidean metric, and the scaling coefficient for the variance and the covariance values is 1. For the algorithm of EMGA_{BIC}, the maximum parent of each node (problem variable) in the Gaussian network is equal to D-1(where D is the number of variables). Also, the K2 algorithm [68] is adopted to learn the network.

Table 4.2 presents the experimental results of the 13 benchmark problems. The value in the first column is the objective function value of the optimal solution found so far. All the algorithms belong to EDAs. The second column shows them with various Gaussian distribution models. According to the mean and the standard deviation (STD) of 30 independent runs, the best test results are shown in boldface. N/A indicates that the algorithms cannot find a feasible solution for the corresponding problem.

Table 4.2 The experimental results of thirteen benchmark problems.

Problem	Algorithm	Best	Median	Worst	Mean	STD
P1 24.30621	MVGD-TR	146.59351	370.66850	1218.41029	430.40244	2.72E+02
	MVGD-TO	59.82129	128.4677	639.25987	172.75143	1.16E+02
	MVGD(0.5)	24.34703	25.08787	28.14291	25.26348	9.13E-01
	MVGD	**24.30621**	**24.30621**	**24.30622**	**24.30621**	1.43E-06
	MIXMVGD	24.30621	24.30621	24.30622	24.30621	1.53E-06
	UMGD	24.35913	24.51992	24.88805	24.53235	1.16E-01
	MIXUMGD	24.39742	24.52548	24.60527	24.52037	6.39E-02
	EGNA$_{BIC}$	28.08906	28.75890	29.98564	28.94368	6.09E-01
P2 -30665.53867	MVGD-TR	-30396.07926	-30006.28181	-29785.94252	-30024.51709	1.62E+02
	MVGD-TO	-30373.38871	-30242.21676	-29954.34867	-30227.26756	8.60E+01
	MVGD(0.5)	-30665.53867	-30665.53867	-30665.53717	-30665.53862	2.75E-07
	MVGD	**-30665.53867**	**-30665.53867**	**-30665.53867**	**-30665.53867**	4.05E-09
	MIXMVGD	-30665.53867	-30065.53867	-30665.53867	-30665.53867	6.18E-09
	UMGD	-30636.07385	-30607.30539	-30553.76709	-30603.47909	1.84E+01
	MIXUMGD	-30627.83423	-30600.34243	-30584.06405	-30603.19953	1.39E+01
	EGNA$_{BIC}$	-30664.98591	-30663.15135	-30662.74469	-30663.49991	8.43E-01
P3 680.6300574	MVGD-TR	710.2880854	839.9662143	913.6463811	835.5605324	4.75E+01
	MVGD-TO	709.2817151	734.7938259	780.4584603	736.9573149	1.75E+01
	MVGD(0.5)	680.6300574	680.6300574	680.6300574	680.6300574	5.64E-10
	MVGD	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	4.44E-10
	MIXMVGD	680.6300574	680.6300574	680.6300574	680.6300574	4.68E-10
	UMGD	680.6311313	680.6402961	680.7021735	680.6432836	1.28E-02
	MIXUMGD	680.6323501	680.6365227	680.6595789	680.6429420	1.13E-02
	EGNA$_{BIC}$	681.1644189	682.6673911	683.4017651	682.4454605	8.17E-01
P4 -0.095825041	MVGD-TR	**-0.095825041**	**-0.095825041**	**-0.095825041**	**-0.095825041**	2.82E-17
	MVGD-TO	**-0.095825041**	**-0.095825041**	**-0.095825041**	**-0.095825041**	2.82E-17
	MVGD(0.5)	**-0.095825041**	**-0.095825041**	**-0.095825041**	**-0.095825041**	2.82E-17
	MVGD	**-0.095825041**	**-0.095825041**	**-0.095825041**	**-0.095825041**	2.82E-17
	MIXMVGD	-0.095825041	-0.095825041	-0.095825041	-0.095825041	5.63E-17
	UMGD	**-0.095825041**	**-0.095825041**	**-0.095825041**	**-0.095825041**	2.82E-17
	MIXUMGD	-0.095825041	-0.095825041	-0.095825041	-0.095825041	3.69E-17
	EGNA$_{BIC}$	-0.105459504	-0.105458091	-0.105455895	-0.105458124	1.01E-06
P5 -1.90515526	MVGD-TR	-1.55756661	-1.34124099	-1.11081672	-1.33544951	1.08E-01
	MVGD-TO	-1.62701343	-1.54087261	-1.47732481	-1.55367841	4.12E-02
	MVGD(0.5)	-1.90515526	-1.90515507	-1.90201033	-1.90380749	8.25E-04
	MVGD	**-1.90515526**	**-1.90515525**	**-1.90515525**	**-1.90515525**	2.09E-09
	MIXMVGD	-1.90515526	-1.90515525	-1.90515525	-1.90515525	2.20E-09
	UMGD	-1.90512182	-1.90508326	-1.89526779	-1.90463995	1.82E-03
	MIXUMGD	-1.90511012	-1.90509281	-1.90418987	-1.90493184	3.37E-04
	EGNA$_{BIC}$	-1.90249086	-1.90105729	-1.90075592	-1.90125231	5.43E-04
P6 -6961.813876	MVGD-TR	N/A	N/A	N/A	N/A	N/A
	MVGD-TO	-6332.418774	-4899.615255	-3830.101329	-4952.333856	7.17E+02
	MVGD(0.5)	-6961.813876	-6961.813876	-6504.690057	-6930.911817	1.00E+02
	MVGD	**-6961.813876**	**-6961.813876**	**-6961.813876**	**-6961.813876**	1.32E-09
	MIXMVGD	-6961.813876	-6961.813876	-6961.813876	-6961.913876	1.56E-09
	UMGD	-6961.676935	-6961.518136	-6961.177406	-6961.487329	1.35E-01
	MIXUMGD	-6961.597533	-6961.448874	-6961.146459	-6961.433218	1.33E-01
	EGNA$_{BIC}$	-6960.229309	-6957.048514	-6952.197150	-6956.685778	2.78E+00

Problem	Algorithm	Best	Median	Worst	Mean	STD
P7 7049.24802	MVGD-TR	N/A	N/A	N/A	N/A	N/A
	MVGD-TO	11557.00693	13117.03269	19609.66721	14928.87413	3.18E+03
	MVGD(0.5)	7054.02998	7114.55094	7857.85152	7215.76273	2.03E+02
	MVGD	**7049.24802**	**7049.24802**	**7049.29318**	**7049.24975**	8.29E-03
	MIXMVGD	7049.24802	7049.24802	7057.03780	7050.44775	2.67E+00
	UMGD	7161.59440	7488.23249	8206.47211	7555.16036	2.72E+02
	MIXUMGD	7256.89947	7460.056197	7732.03963	7457.09101	1.50E+02
	EGNA$_{BIC}$	7060.10186	7063.621358	7077.20849	7065.74750	5.86E+00
P8 -0.8660254	MVGD-TR	N/A	N/A	N/A	N/A	N/A
	MVGD-TO	N/A	N/A	N/A	N/A	N/A
	MVGD(0.5)	-0.8660254	-0.8660071	-0.8602903	-0.8655967	1.10E-03
	MVGD	**-0.8660254**	**-0.8660254**	**-0.8660254**	**-0.8660254**	8.70E-09
	MIXMVGD	-0.8660252	-0.8660249	-0.8660242	-0.8660249	2.81E-07
	UMGD	-0.8659985	-0.8659438	-0.8637950	-0.8658228	4.07E-04
	MIXUMGD	-0.8659887	-0.8658484	-0.7419183	-0.8460793	4.34E-02
	EGNA$_{BIC}$	-0.7803491	-0.7474665	-0.7240270	-0.7530561	2.13E-02
P9 -15.00	MVGD-TR	N/A	N/A	N/A	N/A	N/A
	MVGD-TO	N/A	N/A	N/A	N/A	N/A
	MVGD(0.5)	-14.83	-12.74	-9.60	-12.59	1.45E+00
	MVGD	-15.00	-13.38	-9.01	-13.21	1.48E+00
	MIXMVGD	-15.00	12.76	-7.00	-12.25	2.67E+00
	UMGD	**-15.00**	**-15.00**	**-15.00**	**-15.00**	2.38E-05
	MIXUMGD	-15.00	-15.00	-14.99	-15.00	5.69E-04
	EGNA$_{BIC}$	-14.65	-14.27	-14.10	-14.32	2.15E-01
P10 -47.764888	MVGD-TR	N/A	N/A	N/A	N/A	N/A
	MVGD-TO	N/A	N/A	N/A	N/A	N/A
	MVGD(0.5)	N/A	N/A	N/A	N/A	N/A
	MVGD(CS ε)	-47.761471	-47.761459	-46.027199	-47.675452	3.23E-01
	MVGD	-47.761419	-47.761360	-47.760964	-47.761293	1.22E-04
	MIXMVGD	**-47.761417**	**-47.761374**	**-47.761138**	**-47.761335**	9.55E-05
	UMGD	-45.728271	-45.313567	-44.783326	-45.308322	2.06E-01
	MIXUMGD	-45.328327	-45.186067	-45.044207	-45.188322	9.87E-02
	EGNA$_{BIC}$	N/A	N/A	N/A	N/A	N/A
P11 8853.539675	MVGD-TR	N/A	N/A	N/A	N/A	N/A
	MVGD-TO	N/A	N/A	N/A	N/A	N/A
	MVGD(0.5)	N/A	N/A	N/A	N/A	N/A
	MVGD(CS ε)	8868.305729	8949.782536	8965.230232	8942.472842	2.83E+01
	MVGD	**8853.539354**	**8928.481296**	**8958.393142**	**8917.109591**	3.24E+01
	MIXMVGD	8877.725705	8951.024483	8964.353763	8945.730069	2.46E+01
	UMGD	8943.764691	9148.339833	9171.638285	9066.053225	1.06E+02
	MIXUMGD	8942.484182	8946.558929	9150.876581	9026.841925	1.05E+02
	EGNA$_{BIC}$	N/A	N/A	N/A	N/A	N/A
P12 5126.496714	MVGD-TR	N/A	N/A	N/A	N/A	N/A
	MVGD-TO	N/A	N/A	N/A	N/A	N/A
	MVGD(0.5)	N/A	N/A	N/A	N/A	N/A
	MVGD(CS ε)	5255.870417	5381.812239	5502.136379	5368.892511	6.41E+01
	MVGD	5126.497980	5126.497983	5126.497989	5126.497984	2.91E-06
	MIXMVGD	**5126.497980**	**5126.497982**	**5126.497988**	**5126.497983**	2.53E-06
	UMGD	N/A	N/A	N/A	N/A	N/A
	MIXUMGD	N/A	N/A	N/A	N/A	N/A
	EGNA$_{BIC}$	N/A	N/A	N/A	N/A	N/A

Problem	Algorithm	Best	Median	Worst	Mean	STD
P13 (Max) 1	MVGD-TR	1	1	0.991	0.996	3.80E-03
	MVGD-TO	N/A	N/A	N/A	N/A	N/A
	MVGD(0.5)	N/A	N/A	N/A	N/A	N/A
	MVGD	**1**	**1**	**1**	**1**	2.07E-06
	MIXMVGD	N/A	N/A	N/A	N/A	N/A
	UMGD	1	1	1	1	4.76E-06
	MIXUMGD	N/A	N/A	N/A	N/A	N/A
	EGNA$_{BIC}$	N/A	N/A	N/A	N/A	N/A

Fig. 4.4 shows the total win times of each EDA when solving the 13 benchmark COPs. For instance, when solving P01, the EDA with the probabilistic model of MVGD reached the best result. Hence, it won 7 times (there are 8 different EDAs in total). The EDA with MIXMVGD obtained the second best value; it won 6 times. The result of the EDA with MVGD-TR came last and it won 0 times in this problem. If the algorithms cannot find a feasible solution for one problem, it is also considered to win 0 times. According to these criteria, the win times of the EDAs for other problems are calculated and then the total win times are obtained. It is seen that the EDA with MVGD can win most of the time. Here since the algorithm MVGD(CS ε) solved only three problems, its win times was not investigated in Fig. 4.4.

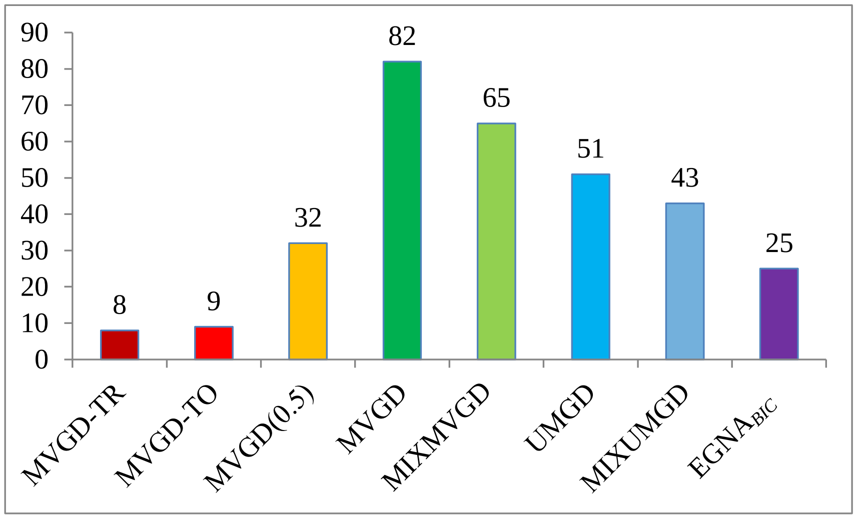


Figure 4.4 The total win times of each algorithm in solving 13 COPs.

4.4.1.1 Comparison of EDAs based on MVGD with different selection methods

All the EDAs used extreme elitism selection to choose the parent population, except MVGD-TR and MVGD-TO. Note that MVGD-TR is the EDA based on the model of MVGD with truncation selection. This selection method chooses the leading half of the best solutions as the parent after the sorting step, and each elected solution takes one item equally in the parent population. From Table 4.2 and Fig. 4.4, it is seen that MVGD-TR cannot have good performance in solving COPs. This algorithm can find the feasible solutions for P1 to P5 and sometimes can reach the optimal solution for P13. But it cannot find a feasible solution for the other problems. The truncation selection makes each solution have the same weight in the parent population. Hence, it cannot sufficiently develop the potential of these few top solutions. These solutions are more likely to be the high quality solutions than the other ordinary solutions. The high quality solutions are the feasible solutions with small objective function values or the solutions close to the optimal solution. If there are more high quality solutions in the parent population, it is more likely that the algorithm will reach the optimal solution.

It is easy to observe the difference between GAs and EDAs. GAs use crossover and mutation to generate the offspring. Hence, they demand a more diverse parent population, and the original truncation or tournament selection may be more appropriate for them. But EDAs employ a probabilistic model to sample the offspring and expect the mean of the probabilistic model to move toward the optimal solution in the search. If the mean can come close to the optimal solution, it is more likely to find the optimal solution among the sampled offspring. Hence, the moving direction of the mean is very important in EDAs. Making these few top solutions represent a somewhat higher percentage than other ordinary solutions in the parent population, can pull the mean toward these leading solutions and improve the likelihood of obtaining optimal solutions.

MVGD-TO is the EDA with MVGD, but using the tournament selection method. In this selection, every time two solutions are randomly elected, one solution will win in light of these criteria: if the two solutions are both feasible, the one with the smaller objective value wins; if one solution is feasible and the other is infeasible, the feasible one wins; if the two solutions are both infeasible, the one with a fewer number of violated constraints wins; and if they violate the same number of constraints, the one with the smaller value of ($g(\boldsymbol{X})_{\text{violated}} + |h(\boldsymbol{X})_{\text{violated}}|$) wins. It is seen from Table 4.1 and Fig. 4.4 that the performance of MVGD-TO is similar to that of MVGD-TR and cannot have stable performance in solving COPs. It can find feasible solutions for P1 to P7, but these solutions are far away from the optimal solutions except for P4. For other problems, this algorithm is unable to reach a feasible solution. The tournament selection method can choose high quality solutions, but this behaviour is random and intermittent. Hence, the algorithm cannot effectively take advantage of these high quality solutions. As a result, it might not find the feasible region or the optimal solution for most problems. It is necessary to steadily extrude the effect of a few leading best solutions for EDAs when solving COPs. However, if the percentage of these leading best solutions is too high, the population diversity will reduce and the result may be not satisfactory.

4.4.1.2 MVGD with a smaller parent population and a constant tolerance

MVGD(0.5) is the EDA using the MVGD model with the extreme elitism selection. But its parent population is half the size of the entire population. It can reach the optimal solution for P2 to P6 and P8 while MVGD-TR and MVGD-TO can just find some feasible solutions for these problems except P4. Besides, it can find better feasible solutions than these two algorithms for P1, P7 and P9. From Fig. 4.4 it is seen that overall, MVGD(0.5) outperforms MVGD-TR and MVGD-TO. According to these results, it is beneficial for EDAs to steadily highlight the role of a few top solutions. However, compared to MVGD that uses the parent population with the same size as the entire population, the stability of MVGD(0.5) in obtaining the optimal solutions is lower (see Table 4.2 and Fig. 4.4). MVGD(0.5) uses a smaller population size, which means the diversity of the parent population decreases and a number of lower-ranking solutions will not enter the parent population. But these solutions may include some useful infeasible solutions. So MVGD(0.5) sometimes cannot find the optimal solutions or the feasible region for some problems. MVGD(CSε) algorithm uses a constant allowed tolerance when solving the COPs with equality constraints. Although it can find some feasible solutions for P10 to P12, the mean of these solutions cannot outperform the results of MVGD with a varying tolerance.

4.4.1.3 Comparison of EDAs based on different Gaussian distributions

From the presented analysis it is seen that extruding the effect of a few leading best solutions and keeping a moderate diversity in the parent population are both very important for EDAs when solving COPs. However, the algorithms of EDAs with the models of UMGD, MIXUMGD and EMGA$_{BIC}$ all use the modified extreme elitism selection. Their parent population size is also the same as the entire population. But from Table 4.2 and Fig. 4.4 it is clear that their experimental results for most problems cannot surpass the performance of MVGD or MIXMVGD. So there must be some other important factor that impacts the performance of these algorithms.

Fig. 4.5(a) shows a two-dimensional univariate marginal Gaussian distribution where the correlation coefficient between two variables is 0. Both variables have zero mean, and the variances are taken as 1. In this distribution, when x_1 is increased from 0 to 0.5, the change trend of x_2 is not known. The point may be sampled at A or B, and x_2 may be positive or negative. In other words, the information of x_1 will not provide any information on x_2. Hence, when the EDA samples the offspring from this distribution, every variable is generated along its respective dimension and does not interact with each other. Essentially, there is no constraint for each variable except for their lower and upper bounds.

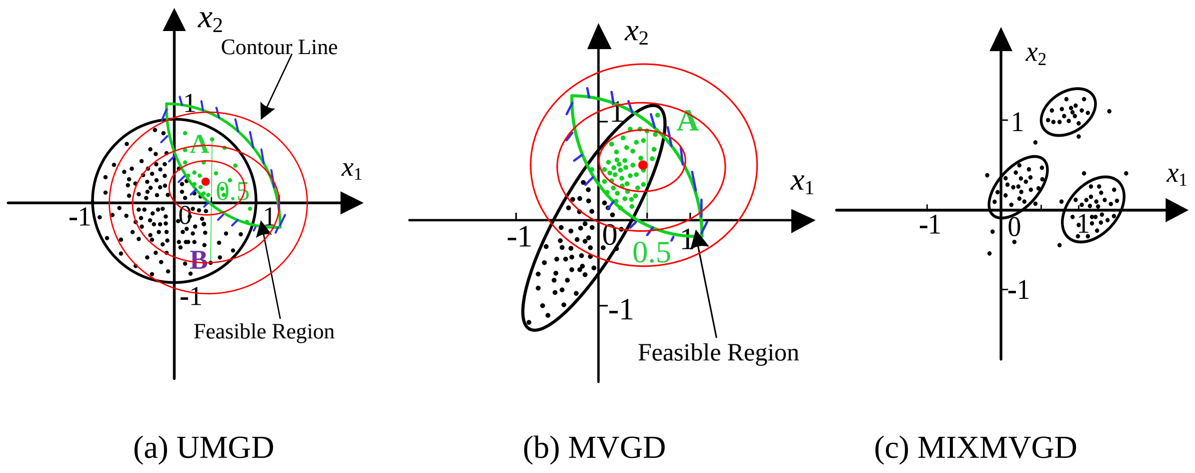


Figure 4.5 The distribution of three types of Gaussian probabilistic models.

Fig. 4.5(b) shows a two-dimensional multivariate Gaussian distribution, and the correlation coefficient between the two variables is taken to be 0.5. Here, when x_1 is increased from 0 to 0.5, it is known that x_2 must be positive. Then the offspring of x_2 are located at the upper right part of the coordinate and the density of offspring spreading in the feasible region is greater than that of the offspring sampled by UMGD. The offspring of UMGD can be located in both upper and lower right parts and the strength is dispersed, which decreases the probability of finding high quality solutions or the optimal solution represented by the thick red solid in the feasible region. Besides, when MVGD produces an offspring, each variable must be affected by the constraints from the other variables, echoing the equality or inequality constraints in COPs. For instance, suppose that the COPs contain equality constraints or the optimal solutions are located at the intersection of some constraints. Then when one variable varies, the other variables must vary correspondingly, and the Gaussian search cannot deviate from the feasible region or the optimal solution. From Table 4.2 and Fig. 4.4, it is seen that for most problems two EDAS with MVGD and MIXMVGD can outperform the two EDAs with UMGD and MIXUMGD.

Fig.4.5(c) shows an example of combined Gaussian distributions with three clusters. Although the combined Gaussian distribution may have a better correspondence to the actual distribution in the practical data analysis than the single Gaussian distribution, there is only one optimal solution for COPs in the present work. The center of the Gaussian distribution should approach the optimal solution as close as possible. More than one cluster means more than one center, and that may distract the effect of some high quality solutions. From Table 4.2 it is seen that MIXMVGD can have good performance in most problems, but overall MVGD outperforms MIXMVGD. Besides, the algorithms with combined Gaussian model have a high computational cost due to the clustering process. To obtain the same optimal solution for a problem, MIXMVGD takes approximately 4500 times longer than MVGD, in the present experiments. EGNA$_{BIC}$ considers the interaction between some variables in view of a penalized maximum likelihood score. If the score satisfies the criterion, a link exists between the two variables. It is seen from the experimental results that EGNA$_{BIC}$ can find the feasible solutions for most problems. But the solutions may just approach the optimal solutions, and hence the algorithm is not suitable for performing a fine search when solving COPs.

Fig. 4.6 through 4.8 show the convergence of the best fitness of each algorithm when solving some problems. Here the best fitness is the minimum value among the objective function values of all feasible solutions that are reached in one iteration. Fig. 4.6 corresponds to P1, which has 8 inequality constraints and 10 variables. Its objective value of the optimal solution is at 24.30621. The right figure is the enlarged view of the part in the red dashed ellipse in the left figure. As the EDAs of MVGD-TR or MVGD-TO cannot reach the best fitness of less than 80, their convergence curve is not shown in this figure.

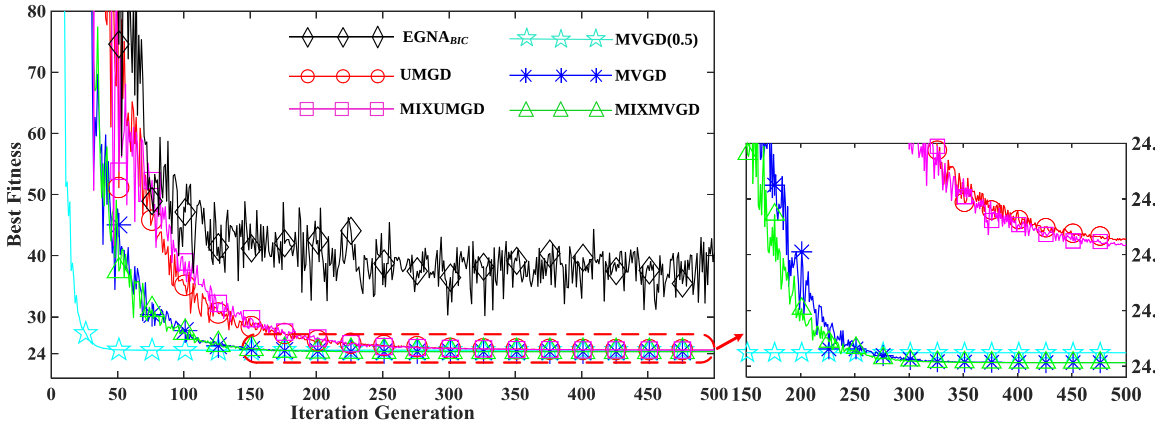


Figure 4.6 The best fitness convergence curves of each algorithm when solving P1.

It is seen that even though EGNA$_{BIC}$ can have a downward trend in the beginning, it does not converge satisfactorily. From the distribution of this algorithm, as given in Equation (4.8), it is seen that the mean of each variable consists of two parts. One is its own unconditional mean (μ_i), and the other comes from its parent variables ($\sum_{k=1}^{i-1} b_{ki}(x_k-\mu_k)$). So, the change in amplitude of the mean in this case can be bigger than those of UMGD and MVGD, and the convergence curve of EGNA$_{BIC}$ has larger oscillations as shown in Fig. 4.6 and 4.7. This is why EGNA$_{BIC}$ is not suitable for performing fine search. MVGD(0.5) has a smaller parent population and it also extrudes a few leading best solutions. Hence the parent population is less diverse, and the variance of each variable decreases rapidly. Hence the algorithm has the fastest convergence, but the final result is worse than those from MVGD and MIXMVGD. This implies that the parent population should maintain a moderate diversity. Besides, this figure shows that the three EDAs with MVGD converge to a smaller fitness value than the two EDAs with UMGD. This confirms that a multivariate model is more suitable than the univariate model to solve COPs. Fig. 4.7 shows the case of P5, which has 38 inequality constraints and 5 variables. Its objective value of the optimal solution is -1.90515526. It is seen that the convergence situation is similar to that of P1. The algorithms with multivariate model outperform the algorithms with univariate model and EGNA$_{BIC}$.

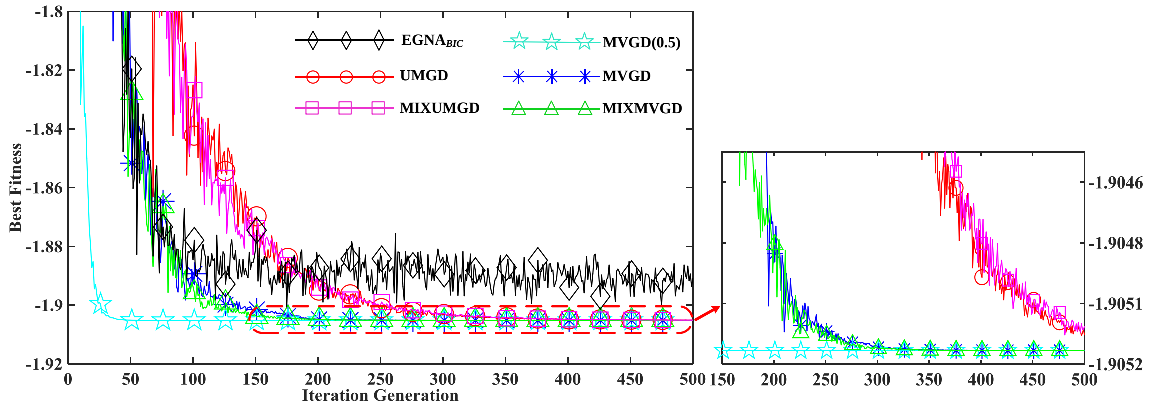


Figure 4.7 The best fitness convergence curves of each algorithm when solving P5.

Fig. 8 shows the case of P12, which has 2 inequality constraints, 3 equality constraints and 4 variables. The objective value of its best solution is 5126.49671. The feasible region is located at the intersections of 3 equalities and is very small. Hence, it is difficult for some algorithms to reach the feasible region, and only three algorithms can find some feasible solutions for the present problem. The upright lines indicate that the algorithms did not reach a feasible solution in some generations. The best fitness jumps to 10^6 if the algorithms cannot find any feasible solution in one iteration. The upper bound of the vertical axis is 5600, and hence the value of 10^6 does not appear in the figure. Besides, MVGD and MIXMVGD adopt a varying tolerance ($1/(1.02^g)$). The solutions obtained after the 582nd generation are valid feasible solutions since from that generation the tolerance is less than 0.00001. For the EDA of MVGD(CS ε), as long as the best fitness does not jump to the value of 10^6, the algorithm reaches a feasible solution.

However, the final convergence results of MVGD and MIXMVGD are better than that of MVGD(CS ε). This shows the advantage of using a varying tolerance, and some infeasible solutions can also provide useful information when solving COPs.

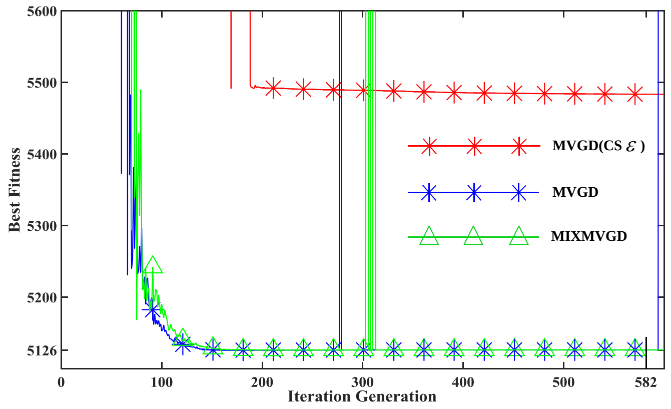


Figure 4.8 The best fitness convergence curves of each algorithm when solving P12.

4.4.1.4 Comparison of EDAs and some state-of-the-art algorithms for COPs

Table 4.3 shows the test results of some state-of-the-art approaches for these benchmark problems. These algorithms are introduced in the literature [50-53]. Here N/A donates that these techniques did not provide the data. It is seen the EDA based on MVGA can have better performance than other techniques for some problems. However, it should be mentioned that the EDA uses more function evaluations than other methods to obtain these test results. EDAs usually prefer a big population size since they adopt sampling to generate the offspring. More individuals mean more opportunities to obtain high quality solutions.

Table 4.3 The test results of some state-of-the-art algorithms for these benchmark problems.

Problem	Algorithm	Best	Median	Worst	Mean	STD
P1 24.30621	MVGD	**24.30621**	**24.30621**	**24.30622**	**24.30621**	1.43E-06
	TFGA	24.410977	26.735666	35.881930	N/A	2.61E+00
	ATMES	24.306	24.313	24.359	24.316	1.1E-02
	M-ABC	24.315	N/A	24.854	24.415	1.24E-01
	PSGA	24.360	N/A	24.999	24.738	2.30E-01

Problem	Algorithm	Best	Median	Worst	Mean	STD
P2 -30665.53867	MVGD	-30665.53867	-30665.53867	-30665.53867	-30665.53867	4.05E-09
	TFGA	-30665.5312	-30063.3642	-30651.9595	N/A	3.31E+00
	ATMES	-30665.539	-30665.539	-30665.539	-30665.539	7.4E-12
	M-ABC	-30665.539	-30665.539	-30665.539	-30665.539	2.22E-11
	PSGA	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	7.28E-12
P3 680.6300574	MVGD	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	4.44E-10
	TFGA	680.762228	681.706290	684.131429	N/A	7.44E-01
	ATMES	680.630	680.633	680.673	680.639	1.0E-02
	M-ABC	680.632	N/A	680.691	680.647	1.55E-02
	PSGA	680.630	N/A	680.725	680.658	2.48E-02
P4 -0.095825041	MVGD	-0.095825041	-0.095825041	-0.095825041	-0.095825041	2.82E-17
	TFGA	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	0
	ATMES	-0.095825	-0.095825	-0.095825	-0.095825	2.8E-17
	M-ABC	-0.095825	-0.095825	-0.095825	-0.095825	4.23E-17
	PSGA	-0.095825	-0.095825	-0.095825	-0.095825	3.84E-09
P5 -1.90515526	MVGD	-1.90515526	-1.90515525	-1.90515525	-1.90515525	2.09E-09
	TFGA	N/A	N/A	N/A	N/A	N/A
	ATMES	N/A	N/A	N/A	N/A	N/A
	M-ABC	**-1.905**	**-1.905**	**-1.905**	**-1.905**	4.52E-16
	PSGA	**-1.905**	**-1.905**	**-1.905**	**-1.905**	4.68E-15
P6 -6961.813876	MVGD	**-6961.813876**	**-6961.813876**	**-6961.813876**	**-6961.813876**	1.32E-09
	TFGA	-6961.17856	-6959.5683	-6954.3186	N/A	1.27E+00
	ATMES	-6961.814	-6961.814	-6961.814	-6961.814	4.6E-12
	M-ABC	-6961.814	-6961.814	-6961.814	-6961.814	0
	PSGA	-6961.814	-6961.814	-6961.814	-6961.814	9.26E-12
P7 7049.24802	MVGD	**7049.24802**	**7049.24802**	**7049.29318**	**7049.24975**	8.29E-03
	TFGA	7060.55288	7723.166720	12097.4078	N/A	7.99E+02
	ATMES	7052.253	7215.357	7560.224	7250.437	1.2E+02
	M-ABC	7051.706	N/A	7473.109	7233.882	1.10E+02
	PSGA	7049.255	N/A	7092.609	7059.107	1.26E+01
P8 -0.8660254	MVGD	**-0.8660254**	**-0.8660254**	**-0.8660254**	**-0.8660254**	8.70E-09
	TFGA	N/A	N/A	N/A	N/A	N/A
	ATMES	N/A	N/A	N/A	N/A	N/A
	M-ABC	-0.866006	N/A	-0.672216	-0.7950187	9.39E-02
	PSGA	-0.866025	N/A	-0.814956	-0.856956	2.13E-02
P9 -15.00	MVGD	-15.00	-13.38	-9.01	-13.21	1.48E+00
	UMGD	-15.00	-15.00	-15.00	-15.00	2.38E-05
	TFGA	-14.9999	-14.9997	-11.9999	N/A	8.51E-01
	ATMES	-15.000	-15.000	-15.000	-15.000	1.6E-14
	M-ABC	**-15**	**-15**	**-15**	**-15**	0
	PSGA	**-15**	**-15**	**-15**	**-15**	0
P10 -47.764888	MVGD	-47.761419	-47.761360	-47.760964	-47.761293	1.22E-04
	MIXMVGD	**-47.761417**	**-47.761374**	**-47.761138**	**-47.761335**	9.55E-05
	TFGA	N/A	N/A	N/A	N/A	N/A
	ATMES	N/A	N/A	N/A	N/A	N/A
	M-ABC	-47.641	N/A	-46.537	-47.271	2.64E-01
	PSGA	-47.738	N/A	-47.567	-47.679	3.74E-02

Problem	Algorithm	Best	Median	Worst	Mean	STD
P11 8853.539675	MVGD	**8853.539354**	**8928.481296**	**8958.393142**	**8917.109591**	3.24E+01
	TFGA	N/A	N/A	N/A	N/A	N/A
	ATMES	N/A	N/A	N/A	N/A	N/A
	M-ABC	8866.618	N/A	9165.219	8987.459	9.57E+01
	PSGA	8855.704	N/A	9009.484	8944.808	2.75E+01
P12 5126.496714	MVGD	5126.497980	5126.497983	5126.497989	5126.497984	2.91E-06
	MIXMVGD	**5126.497980**	**5126.497982**	**5126.497988**	**5126.497983**	2.53E-06
	TFGA	5126.5096	5170.5294	6112.2231	N/A	3.41E+02
	ATMES	5126.498	5126.776	5135.256	5127.648	1.8E+00
	M-ABC	5126.736	N/A	5317.196	5178.139	5.61E+01
	PSGA	5126.497	N/A	5166.438	5140.897	1.44E+01
P13 (Max) 1	MVGD	1	1	1	1	2.07E-06
	TFGA	1.00009	0.94899	0.7855820	N/A	4.89E-02
	ATMES	1	1	1	1	5.9E-05
	M-ABC	1	1	1	1	4.68E-05
	PSGA	**1**	**1**	**1**	**1**	2.76E-09

4.4.2 The application of EDAs in 4 mechanical engineering design problems

Table 4.4 presents the test results of each EDA for 4 mechanical design problems. The boldface rows show the best results among the EDAs. These engineering problems have been solved by Mohamed and Sabry [99], Ray and Liew [102], Zhang et al. [103], Baykasoğlu [104], and others. Some of their results are given here and others can be found in the literature.

Table 4.4 The test results of 4 mechanical engineering design problems.

Problem	Algorithm	Best	Median	Worst	Mean	STD
EG1 Welded beam	MVGD-TR	2.0567129	2.5349294	3.0014569	2.5301831	2.82E-01
	MVGD-TO	1.8149423	1.9234501	2.1016841	1.9436740	7.13E-02
	MVGD(0.5)	1.7248523	1.7248523	1.7248529	1.7248523	1.03E-07
	MVGD	1.7248523	1.7248523	1.7248523	1.7248523	8.26E-10
	MIXMVGD	1.7248523	1.7248523	1.7248523	1.7248523	8.66E-10
	UMGD	1.7714776	1.8997899	2.0182220	1.9003339	6.24E-02
	MIXUMGD	1.8788171	1.9377164	1.9909058	1.9391052	3.69E-02
	EGNA$_{BIC}$	1.7840900	1.8400507	1.8620323	1.8314880	3.01E-02
	Ray and Liew	2.3854347	3.0025883	6.3996785	3.2551371	9.59E-01
	Zhang et al.	2.3809566	2.3809566	2.3809566	2.3809566	3.19E-10
	Mohamed	1.724852	1.724852	1.724852	1.724852	1.60E-12
	Baykasoğlu	**1.724852**	**1.724852**	**1.724852**	**1.724852**	0

Problem	Algorithm	Best	Median	Worst	Mean	STD
EG2 Spring	MVGD-TR	0.012967676	0.013694716	0.020991583	0.014240053	1.75E-02
	MVGD-TO	0.013033612	0.013205664	0.014181892	0.013408580	3.37E-04
	MVGD(0.5)	**0.012665233**	**0.012665233**	**0.012666891**	**0.012665379**	3.98E-07
	MVGD	0.012665233	0.012670170	0.012704487	0.012673349	9.78E-06
	MIXMVGD	0.012665801	0.012666868	0.012702481	0.012670983	1.13E-05
	UMGD	0.013143677	0.013581479	0.014129107	0.013624537	2.68E-04
	MIXUMGD	0.013441285	0.013725969	0.014389966	0.013777575	2.56E-04
	EGNA$_{BIC}$	0.012734477	0.012783986	0.013041376	0.012814946	9.37E-05
	Ray and Liew	0.012669249	0.012922669	0.016717272	0.012922669	5.92E-04
	Zhang et al.	0.012665233	0.012665234	0.012738262	0.012669366	1.25E-05
	Mohamed	0.012665233	0.012665423	0.012676809	0.012667168	3.09E-06
	Baykasoğlu	0.0126653049	N/A	0.0127116883	0.0126770446	1.28E-05
EG3 Speed reducer	MVGD-TR	3027.2977376	3224.7498408	3791.4959411	3305.3976770	2.08E+02
	MVGD-TO	3039.6876304	3067.9651602	3108.2213346	3067.6236982	1.71E+01
	MVGD(0.5)	2994.4770466	2995.2688829	3002.6999255	2995.9885304	2.07E+00
	MVGD	2994.4710661	2994.4710661	2994.4710661	2994.4710661	1.32E-10
	MIXMVGD	2994.4710661	2994.4710661	2994.4710661	2994.4710661	1.60E-10
	UMGD	2994.4710932	2994.4711059	2994.4711256	2994.4711078	8.72E-06
	MIXUMGD	2994.4710992	2994.4711028	2994.4711132	2994.4711043	4.37E-06
	EGNA$_{BIC}$	3014.7467648	3029.7085596	3035.4105472	3026.4242401	7.28E+00
	Ray and Liew	2994.744241	3001.758264	3009.964736	3001.758264	4.01E+00
	Zhang et al.	2994.471066	2994.471066	2994.471066	2994.471066	3.58E-12
	Mohamed	**2994.4710661**	**2994.4710661**	**2994.4710661**	**2994.4710661**	1.54E-12
	Baykasoğlu	2996.372698	N/A	2996.669016	2996.514874	9.00E-02
EG4 Three-bar truss	MVGD-TO	263.9025921	263.9400049	264.2990881	263.9670276	8.01E-02
	MVGD-TR	263.8966927	263.9277427	264.0052677	263.9313738	2.63E-02
	MVGD(0.5)	**263.8958434**	**263.8958434**	**263.8958434**	**263.8958434**	0
	MVGD	**263.8958434**	**263.8958434**	**263.8958434**	**263.8958434**	0
	MIXMVGD	**263.8958434**	**263.8958434**	**263.8958434**	**263.8958434**	0
	UMGD	263.8958457	263.8962637	263.8978458	263.8964108	5.53E-04
	MIXUMGD	263.8961310	263.8963583	263.8991527	263.8969114	1.11E-03
	EGNA$_{BIC}$	368.1118398	377.4477835	381.2603528	376.7262392	4.16E+00
	Ray and Liew	263.8958	263.8989	263.96975	263.9033	1.26E-02
	Zhang et al.	263.8958434	263.8958434	263.8958498	263.8958436	9.72E-07
	Mohamed	263.8958434	263.8958434	263.8958434	263.8958434	5.34E-13
	Baykasoğlu	N/A	N/A	N/A	N/A	N/A

4.4.2.1 Welded beam design

This engineering problem here is to design a welded beam and manufacture it at the minimum cost. The objective and constrained function is given by:

Minimize: $f(\mathbf{X}) = 1.10471x_2x_1^2+0.04811x_3x_4(14.0+x_2)$ (4.10)

Subject to: $g_1(\boldsymbol{X}) = \tau(\boldsymbol{X})-\tau_{\max}\leq 0$ $g_2(\boldsymbol{X}) = \sigma(\boldsymbol{X})-\sigma_{\max}\leq 0$ $g_3(\boldsymbol{X}) = x_1-x_4\leq 0$

$g_4(\boldsymbol{X}) = \delta(\boldsymbol{X})-\delta_{\max}\leq 0$ $g_4(\boldsymbol{X}) = P-P_C(\boldsymbol{X})\leq 0$

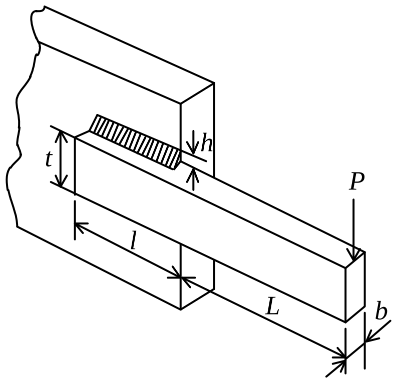
$\tau(\boldsymbol{X}) = [(\tau')^2+2\tau'\tau''x_2/(2R)+(\tau'')^2]^{1/2}$ $\tau'=P/(\sqrt{2}x_1x_2)$ $\tau''=MR/J$ $M=P(L+x_2/2)$ $R=[0.25x_2^2$ $+0.25(x_1+x_3)^2]^{1/2}$ $J=2\{(x_1x_2/2^{1/2})[x_2^2/12+0.25(x_1+x_3)^2]\}$ $\sigma(\boldsymbol{X}) = 6PL/(x_4x_3^2)$ $\delta(\boldsymbol{X})=4PL^3/(Ex_4x_3^3)$

$\sigma_{\max}$=30000psi P=6000lb $P_C(\boldsymbol{X})=4.013[(EGx_3^2x_4^6/36)^{1/2}/L^2](1-x_3(E/G)^{1/2}/(4L))$ L=14in

$E=30\times10^6$psi $G=12\times10^6$psi $\tau_{\max}$=13600psi $\delta_{\max}$=0.25in $0.125\leq x_1\leq 10.0$

$0.1\leq x_2\leq 10.0$ $0.1\leq x_3\leq 10.0$ $0.1\leq x_4\leq 10.0$

The constraints involve the shear stress ($\tau_{\max}$) and the bending stress in the beam ($\sigma_{\max}$), the side constraints, the end deflection of the beam ($\delta_{\max}$), and the buckling load of the bar (P_c). There are four design parameters, as shown in Fig. 4.9(a): $h(x_1)$, $l(x_2)$, $t(x_3)$ and $b(x_4)$. The best solution obtained by MVGD is [0.20572964; 3.47048868; 9.03662392; 0.20572964], the value of each constraint is as follows: -0.00007095, -0.00009522, 0, -0.23554032, -0.00002293, -3.43298378, -0.08072964, and the minimum cost is 1.7248523.



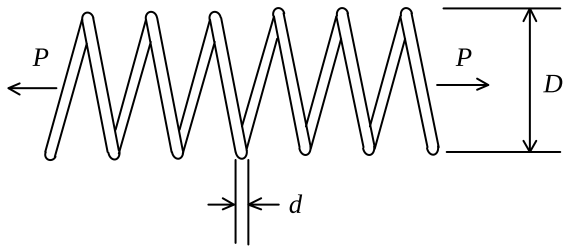


Figure 4.9 (a) A welded beam. (b) A compression spring.

4.4.2.2 Spring design

The second engineering problem seeks the minimum weight of a compression spring subjected to the constraints of minimum deflection, shear stress, surge frequency, and limits on the outside diameter, as shown in Fig. 4.9(b). The objective function is expressed by:

$$\text{Minimize: } f(\mathbf{X}) = (2+x_3)x_1^2x_2 \quad (4.11)$$

$$\text{Subject to: } g_1(\boldsymbol{X}) = 1- x_2^3x_3/71785x_1^4 \leq 0 \quad g_2(\boldsymbol{X}) = (4x_2^2-x_1x_2)/[12566(x_1^3x_2- x_1^4)]+1/(5108x_1^2)-1\leq 0$$

$$g_3(\boldsymbol{X}) = 1- 140.45x_1/(x_2^2x_3)\leq 0 \quad g_4(\boldsymbol{X}) =(x_1+x_2)/1.5-1\leq 0 \quad 0.05\leq x_1\leq 2 \quad 0.25\leq x_2\leq 1.3 \quad 2\leq x_3\leq 15$$

The design variables include the wire diameter $d(x_1)$, the mean coil diameter $D(x_2)$ and the number of active coils $N(x_3)$. MVGD(0.5) and MVGD both can achieve the best solution: [0.0516890609017513; 0.35671773544476; 11.2889660069144]. The constraint values are: 8.88E-16, 1.11E-15, -4.05378562, -0.72772880, and the minimum objective value is 0.012665233.

4.4.2.3 Speed reducer design

This problem seeks the minimum weight of a speed reducer as given by:

$$\text{Minimize: } f(\mathbf{X}) = 0.7854(3.3333x_3^2+14.9334x_3-43.0934)x_2^2x_1-1.508x_1(x_6^2+x_7^2)$$
$$+7.4777(x_6^3+x_7^3)+0.7854(x_4x_6^2+ x_5x_7^2) \quad (4.12)$$

$$\text{Subject to: } g_1(\boldsymbol{X}) = 27/(x_1x_3x_2^2)-1\leq 0 \quad g_2(\boldsymbol{X}) = 397.5/(x_1x_2^2x_3^2)-1\leq 0 \quad g_3(\boldsymbol{X}) = 1.93x_4^3/(x_2x_3x_6^4)-1\leq 0$$

$$g_4(\boldsymbol{X}) = 1.93x_5^3/(x_2x_3x_7^4)-1 \leq 0 \quad g_5(\boldsymbol{X}) = \{[745x_4/(x_2x_3)]^2+16.9\times 10^6\}^{1/2}/(110.0x_6^3)-1 \leq 0$$

$$g_6(\boldsymbol{X}) = \{[745x_5/(x_2x_3)]^2+157.5\times 10^6\}^{1/2}/(85.0x_7^3)-1\leq 0 \quad g_7(\boldsymbol{X}) = x_2x_3/40-1\leq 0 \quad g_8(\boldsymbol{X}) = 5x_2/x_1-1\leq 0$$

$$g_9(\boldsymbol{X}) = x_1/12x_2-1\leq 0 \quad g_{10}(\boldsymbol{X}) = (1.5x_6+1.9)/x_4-1\leq 0 \quad g_{11}(\boldsymbol{X}) = (1.1x_7+1.9)/x_5-1\leq 0$$

$$2.6\leq x_1\leq 3.6 \quad 0.7\leq x_2\leq 0.8 \quad 17\leq x_3\leq 28 \quad 7.3\leq x_4\leq 8.3 \quad 7.3\leq x_5\leq 8.3 \quad 2.9\leq x_6\leq 3.9 \quad 5.0\leq x_7\leq 5.5$$

It is subject to 11constraints such as the bending stress of the gear teeth, the surface stress, the transverse deflections of the shafts, and the stresses in the shafts. There are 7 design variables including the face width (x_1), the module of the teeth (x_2), the number of teeth in the pinion (x_3), the length of the first shaft between bearings (x_4), the length of the second shaft between bearings (x_5), and the diameters of the first (x_6) and the second shafts (x_7). The best solution obtained by MVGD is [3.50000000000008; 0.700000000000009; 17; 7.3; 7.71531991147984; 3.35021466609646; 5.2866544649803]. The constraint values are: -0.07391528, -0.19799853, -0.49917225, -0.90464390, -1.14E-14, -9.99E-15, -0.58333333, -0.05132575, -1.95E-13, and the best objective function value is 2994.4710661. Fig. 4.10 shows the convergence curve of this problem. The best fitness of other EDAs cannot reach 2997 and they are not presented in the figure. It is seen that the situation is similar to the problems presented in Section 4.4.1.3. A big parent population is better than a small one and a multivariate model is better than a univariate model.

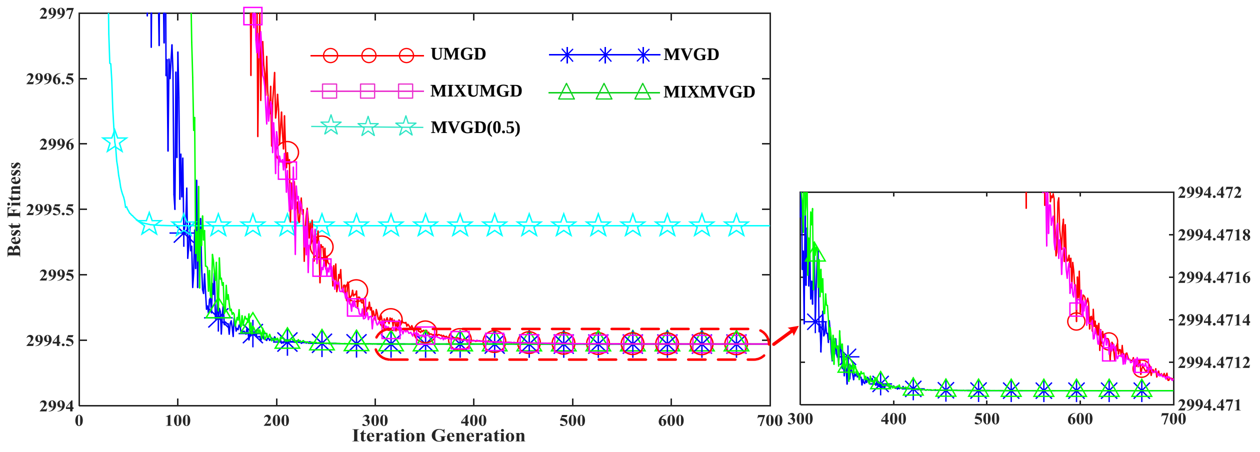


Figure 4.10 The best fitness convergence curves of each algorithm when solving EG3.

4.4.2.4 Three-bar truss design

This design problem concerns finding the minimum volume of a three-bar truss structure under the constraints of stress. Its expression is given by:

Minimize: $f(\boldsymbol{X}) = (2\sqrt{2}x_1 + x_2) \times l$ (4.13)

Subject to: $g_1(\boldsymbol{X}) = (\sqrt{2}x_1 + x_2)P / (\sqrt{2}x_1^2 + 2x_1x_2) - \sigma \leq 0$ $g_2(\boldsymbol{X}) = P / (\sqrt{2}x_2 + x_1) - \sigma \leq 0$

$g_3(\boldsymbol{X}) = x_2P / (\sqrt{2}x_1^2 + 2x_1x_2) - \sigma \leq 0$ l=100cm P=2KN/cm^2 σ=2 KN/cm^2 $0 \leq x_1 \leq 1$ $0 \leq x_2 \leq 1$

The design variables are the two cross-sectional areas x_1 and x_2. Three EDAs can find the best solution as [0.788675136783631 0.408248284272951]. The constraint values are: -6.82E-13, -1464.1016222, -535.8983778 and the minimum objective value is 263.8958434.

4.5 Summary

In this chapter, estimation distribution algorithms (EDAs) based on five different Gaussian distributions were used to solve a set of benchmark constrained optimization problems. The extreme elitism selection method could assist EDAs handle various types of constraints like inequality, equality, linear or nonlinear, which could further improve the application of EDAs in practical engineering optimization problems. The experimental results showed that the multivariate Gaussian distribution model was more suitable than the univariate marginal Gaussian distribution model for COPs with different types of constraints. A single Gaussian model was somewhat better than the combined Gaussian models. Finally, the EDA based on the probabilistic model of a single multivariate Gaussian distribution with the modified extreme elitism selection could have better performance than other EDAs for most benchmark problems.

Chapter 5: Estimation Distribution Algorithms for Multi-objective Optimization Problems

5.1 Introduction

The multi-objective optimization problems (MOPs) are the problems with two or more objectives. It is important to study MOPs since they are very common in the science and engineering areas. The mathematical description of MOPs can be stated as:

$$\begin{aligned} \text{minimize} \quad & F(\boldsymbol{X}) = (f_1(\boldsymbol{X}), f_2(\boldsymbol{X}), \cdots, f_{\mathrm{m}}(\boldsymbol{X}))^T \\ \text{subject to} \quad & g_j(\boldsymbol{X}) \le 0, \qquad j = 1, \cdots, q \\ & h_j(\boldsymbol{X}) = 0, \qquad j = q+1, \cdots, n \\ & x_{iL} \le x_i \le x_{iU} \quad i = 1, \cdots, D \end{aligned} \tag{5.1}$$

Here, *m* donates the number of real-valued objective functions, $g_j(\boldsymbol{X})$ and $h_j(\boldsymbol{X})$ represent the inequality and equality constraints, *n* denotes the total number of constraints, x_{iL} and x_{iU} are the lower and upper bounds, respectively, and *D* donates the number of problem variables.

Traditionally, MOPs are transformed into single objective problems and handled by techniques like weighted-sum approach, ϵ-constraint method, goal programming, and so on. The advantage of these methods is that they are convenient and have low computational cost. But they generally require prior knowledge and experience. So, more recently evolutionary approaches have been suggested to solve MOPs. These methods explore a group of solutions simultaneously and finally seek a set of uniformly distributed Pareto optimal solutions. Compared to the traditional methods, these approaches can provide more alternative choices for decision makers and strike trade-offs between different objectives.

Pareto solutions are also called non-dominated solutions. The definition of non-domination may be presented as follows: Let $\boldsymbol{a}$, $\boldsymbol{b} \in R^m$ (R^m is the objective value space). Then, $\boldsymbol{a}$ is said to dominate $\boldsymbol{b}$, if and only if $\boldsymbol{a}_i \le \boldsymbol{b}_i$, for every $i \in \{1, 2, \cdots, m\}$ and $\boldsymbol{a}_i < \boldsymbol{b}_i$ for at least one index i. An instance of a minimization problem is shown in Fig.5.1 to illustrate non-domination. In the f_1 direction $\boldsymbol{a}_1 < \boldsymbol{b}_1$ and in the f_2 direction $\boldsymbol{a}_2 < \boldsymbol{b}_2$ as well, so $\boldsymbol{a}$ can dominate $\boldsymbol{b}$. Similarly, $\boldsymbol{c}$ can also dominate $\boldsymbol{b}$. Hence, solutions $\boldsymbol{a}$ and $\boldsymbol{c}$ are both better than $\boldsymbol{b}$. For $\boldsymbol{a}$ and $\boldsymbol{c}$, in the f_1 direction $\boldsymbol{a}_1 < \boldsymbol{c}_1$, but in the f_2 direction $\boldsymbol{a}_2 > \boldsymbol{c}_2$. Hence, $\boldsymbol{a}$ cannot dominate $\boldsymbol{c}$ and $\boldsymbol{c}$ cannot dominate $\boldsymbol{a}$, either. The solution $\boldsymbol{a}$ and $\boldsymbol{c}$ are both non-dominated solutions while the solution $\boldsymbol{b}$ is a dominated solution.

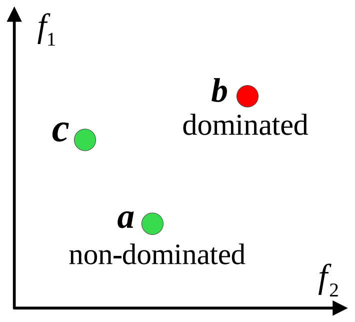


Figure 5.1 The definition of non-domination.

The Pareto optimal solutions in the objective value space constitute a Pareto front seen in Fig. 5.2. The solutions in the Pareto front are all non-dominated solutions and no solution can dominate these front solutions. In Fig. 5.2, every red solution is dominated by at least one solution (If one solution is dominated by any other solution, this solution cannot be a non-dominated solution or in Pareto optimal front). The evolution algorithms attempt to push the Pareto front to move towards the original point and axis in a run of every generation run. In a minimization problem it is better for the Pareto front to be closer to the original point and axis, but these solutions in the front must locate in the feasible region. Besides, these solutions should spread uniformly in the Pareto front. So there will be more alternative solutions for decision makers and a tradeoff among the sub-objectives can be maintained.

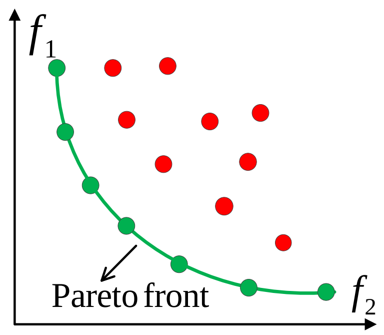


Figure 5.2 Pareto front.

In the past decades, a number of evolutionary algorithms based on searching the Pareto optimal solutions have been developed. Zitzler and Thiele introduced a strength Pareto evolution algorithm (SPEA) that used an external population to store the non-domination solutions. This algorithm evaluated a new individual’s fitness depending on the number of external non-dominated points that dominate it. Also, a clustering technique was used to ensure diversity among non-dominated solutions [56]. Schaffer proposed a vector evaluated genetic algorithm (VEGA), which first selected promising *m* sub-group solutions according to the fitness of the *m* sub-objectives. Then the *m* sub-group solutions were shuffled together to form the parent population. After that, the operation of genetic crossover and mutation were applied on the parent individuals to produce the offspring for the next generation population. VEGA can produce good performance for some MOPs, but it has the weakness of bias towards some solutions and cannot obtain a set of uniformly distributed solutions [57]. Knowles and Corne proposed the Pareto archived evolution strategy (PAES), which used a single-parent single-offspring (1+1) evolution technique. Furthermore, this strategy used the archive to save the non-dominated solutions for the first time. PAES is strictly confined to local search and it effectively solved the optimization problem of finding the minimum communication costs and congestion in circuit switched networks [58]. Deb et al. proposed a fast and elitist multi-objective genetic algorithm called NSGA-II. The improved algorithm decreased the complexity from $O(MN^3)$ to $O(MN^2)$ (M is the number of objectives and N is the population size). Moreover, in order to maintain the diversity of the parent population, they chose the solution with the larger crowding distance in the same layer [59]. Zhang and Li proposed MOEA/D— multi-objective evolutionary algorithm based on decomposition. This algorithm decomposes a multi-objective optimization problem into a number of scalar optimization sub-problems and optimizes them simultaneously. Then each sub-problem is optimized by using information only from its several neighboring sub-problems. Hence, MOEA/D has lower computational complexity than the algorithms based on non-dominated sorting in each generation. This method provides a new research direction for solving multi-objective optimization problems [60]. Recently, Jain and Deb proposed NSGA-III. Compared to the NSGA-II, the main change of this new algorithm is that it introduced a reference-point-based method to select the parent solution in the same layer to keep uniform the solution distribution in the Pareto front. Besides, it can solve problems having more (more than 3) objectives [61].

All the above strategies used the genetic operations of crossover and mutation to produce the offspring. Some other evolution techniques like artificial immune system (AIS) [62], particle swarm optimization (PSO) [63], evolution strategies (ES) [64], differential evolution (DE) [65], and so on also have been suggested to solve MOPs. Also, EDAs have been proposed to handle MOPs. Okabe et al. developed a voronoi-based EDA (VEDA) to solve MOPs. This algorithm can adjust its reproduction process based on the problem structure. When estimating the search distribution, not only the selected individuals, but also those that are not selected are taken into account [106]. Zhang et al. proposed a regularity model-based multi-objective EDA. They considered the Pareto set of a continuous MOP as a piecewise continuous (m-1) (m is the number of objectives) dimensional manifold in the decision space. So, in their probabilistic model, the (m-1) dimensional manifold was used as the centroid of the model. The test results have shown that this algorithm can produce good performance for some MOPs with the variable linkages [107]. Shim et al. suggested an EDA based on restricted Boltzmann machines to handle multi-objective optimization problems in a noisy environment [108]. In this chapter, EDAs based on Gaussian distribution are combined with deferential mutation to solve MOPs.

5.2 The algorithm of combining EDA with DM for MOPs

5.2.1 The flowchart of EDA-DM and domination sorting

Differential mutation is used in differential evolution [79]. Here the proposed algorithm does not involve the operator of differential crossover. Fig.5.3 shows the flowchart of the algorithm of EDA combining DM (EDA-DM) for MOPs. First, the initial population is generated randomly; then the solutions are sorted from the best to the worst. In a single objective problem, the objective values need to be sorted from the minimum to the maximum (or reverse), and this operation is very easy and fast. However, for the MOPs, the domination relationship between each solution should be considered. Here according to the number of individuals, which can dominate one solution, the solutions are sorted from the best to the worst. Fig. 5.4 gives an instance to indicate how to sort the solutions from the best to the worst, according to the domination rank. Here there are 5 solutions (these solutions are in the objective value space, not the decision variable space) in the population and they form the 5×2 matrix ***POP*** = [1 2; 3 5; 2.5

3.6; 4 0.6; 0.2 0.3]. The number of rows corresponds to the number of solutions or the population size; the number of columns corresponds to the number of objectives.

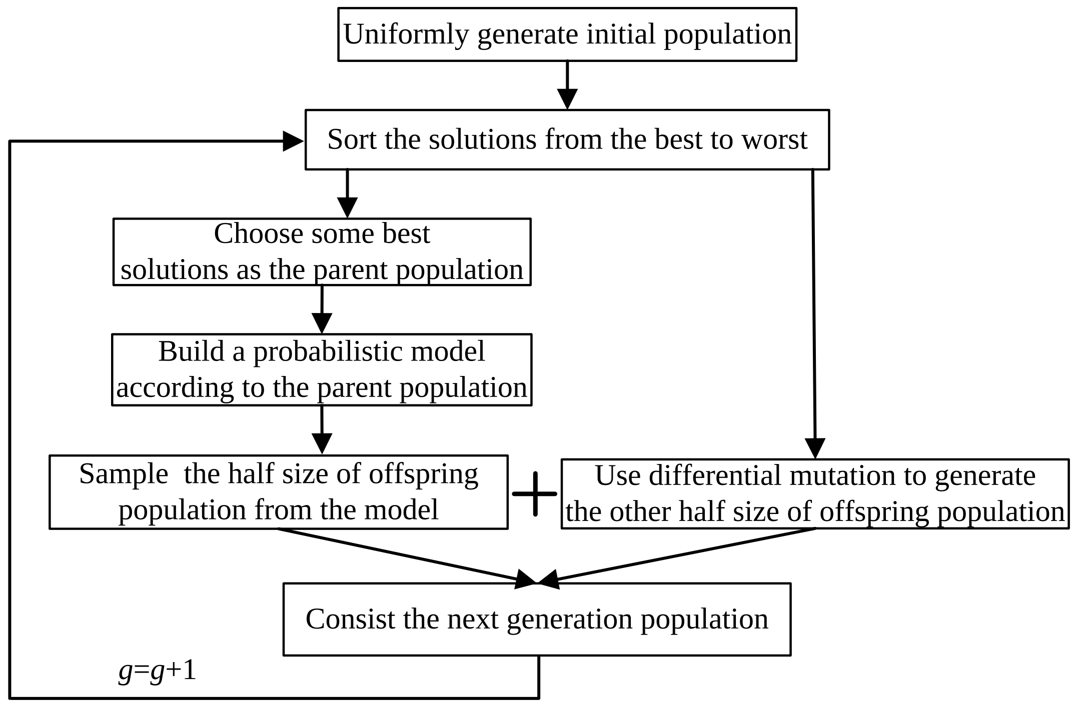
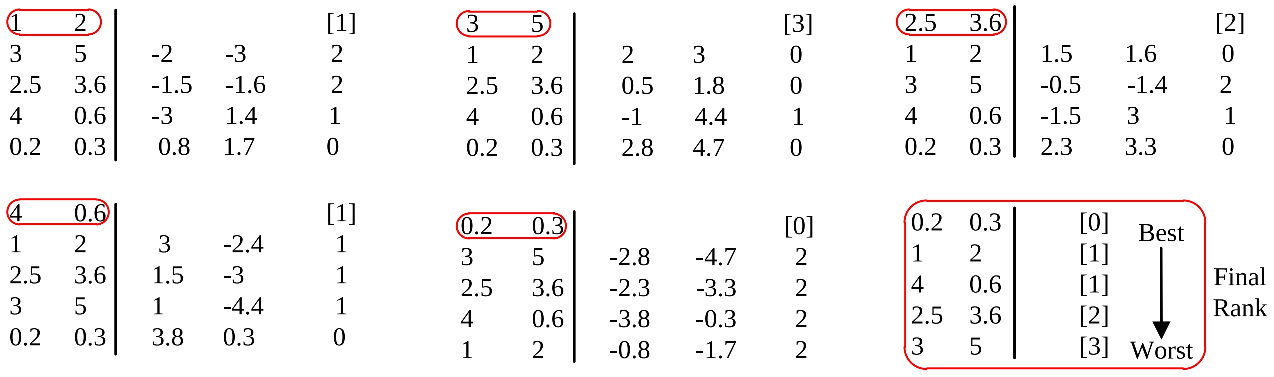

Uniformly generate initial population
Sort the solutions from the best to worst
Choose some best solutions as the parent population
Build a probabilistic model according to the parent population
Sample the half size of offspring population from the model
Use differential mutation to generate the other half size of offspring population
Consist the next generation population
g=g+1

Figure 5.3 Flowchart of EDA-DM for MOPs.

First, every solution subtracts all other solutions. For example, [1 2]-[3 5]=[-2 -3], [1 2]-[2.5 3.6]=[-1.5 -1.6] and so on. Second, the number with a difference less than 0 is counted; for example, for the difference vector [-2 -3], the number is 2. If the number is equal to the number of objectives, it means the solution dominates the other solution. Here [1 2]-[3 5]=[-2 -3], and the 2 differences are both less than 0. Hence, the solution [1 2] can dominate [3 5] (for the minimization problems). If the difference is all more than 0, it indicates that the solution is dominated by the other solution. Here [1 2]-[0.2 0.3]=[0.8 1.7], and the number with the difference less than 0 is equal to 0. Hence, the solution [1 2] is dominated by [0.2 0.3]. Then the number of solutions that can dominate this solution is counted. For example, 1 solution can dominate [1 2]; 3 solutions can dominate [3 5], and so on. Finally, these solutions are sorted from the best to the worst according to how many other solutions can dominate them. Here no solution can dominate [0.2 0.3], so it is located at the top while 3 solutions can dominate [3 5], so it is located at the bottom. It should be mentioned that [1 2] and [4 0.6] are both dominated by only one solution, but the solution [1 2] is in front of [4 0.6]. This is because [1 2] can dominate more solutions (2) than [4 0.6] (0).



1	2			[1]
3	5	-2	-3	2
2.5	3.6	-1.5	-1.6	2
4	0.6	-3	1.4	1
0.2	0.3	0.8	1.7	0

3	5			[3]
1	2	2	3	0
2.5	3.6	0.5	1.8	0
4	0.6	-1	4.4	1
0.2	0.3	2.8	4.7	0

2.5	3.6			[2]
1	2	1.5	1.6	0
3	5	-0.5	-1.4	2
4	0.6	-1.5	3	1
0.2	0.3	2.3	3.3	0

4	0.6			[1]
1	2	3	-2.4	1
2.5	3.6	1.5	-3	1
3	5	1	-4.4	1
0.2	0.3	3.8	0.3	0

0.2	0.3			[0]
3	5	-2.8	-4.7	2
2.5	3.6	-2.3	-3.3	2
4	0.6	-3.8	-0.3	2
1	2	-0.8	-1.7	2

0.2	0.3	[0]
1	2	[1]
4	0.6	[1]
2.5	3.6	[2]
3	5	[3]

Figure 5.4 Domination rank.

After the sorting, the flowchart is divided to two tracks. In one track, EDA uses the probabilistic model to sample the half size of offspring. Here the Gaussian multivariate model is adopted and its joint probability density function is given by

$$f_X(x_1,\cdots,x_D)=\frac{1}{(2\pi)^{D/2}\,|\Sigma|^{1/2}}\exp[-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})] \tag{5.1}$$

Here, D is the dimension of the decision space, $\sum$ is the covariance matrix, $\boldsymbol{X}=[x_1,\cdots,x_D]^T$ is the variable vector, and $\boldsymbol{\mu}=[\mu_1,\cdots,\mu_D]^T$ is the mean vector.

In the other track, DM generates the remaining half offspring. In Chapter 3, DM was combined with EDA to solve the inverse displacement problem of a 7-DOF robotic arm. The difference there is that DM was only operated in one or two generations when EDA did not obtain a satisfactory solution; here DM is more effective in the evolution. There are several popular DM strategies. The present hybrid algorithm adopts the strategy of “DE/rand/1” whose expression is given by

$$\boldsymbol{V}_j^g = \boldsymbol{X}_{r_1}^g + F \cdot (\boldsymbol{X}_{r_2}^g - \boldsymbol{X}_{r_3}^g) \tag{5.2}$$

Here, $\boldsymbol{V}_j^g$ is an offspring; $F \in (0, 1)$ is the scaling factor, which is randomly generated from (0, 1) for every offspring for all problems in the present study; $\boldsymbol{X}_{r_1}^g$, $\boldsymbol{X}_{r_2}^g$ and $\boldsymbol{X}_{r_3}^g$ are the solutions randomly selected from the population, and $r_1 \neq r_2 \neq r_3$. Compared to other DM strategies, this method can generate a larger diversified offspring population.

The difference between EDA and DM in the flowchart is that EDA selects some best solutions to constitute the parent population while DM does not. This is because the responsibility of EDA in the hybrid algorithm is to guide the evolutionary direction while for DM it is to enrich the population diversity and make the algorithm avoid premature convergence.

5.2.2 A modified extreme elitism selection for MOPs

The extreme elitism selection is used to choose the best solutions as the parent for EDA. But this selection is designed originally for single objective problems. Hence, it should be modified for MOPs. Usually, in the early generations, there are just a few non-dominated solutions in the population, so the top 5 solutions are directly chosen as the elites and take more items than other solutions in the parent population. However, after some iteration, there may be more and more non-dominated solutions in the population and in this situation the top 5 solutions cannot be directly extruded, since these 5 solutions may be very close to each other. If they still take more items than other solutions, the final obtained Pareto solutions may gather in some areas and cannot spread uniformly. So if there are more than 10 (experience-based threshold value) non-dominated solutions in the population, a procedure is activated to remove some very close solutions from the non-dominated solutions group. The removing action is only operated on the non-dominated solutions in the population, as they are already at the top and have higher probability of taking more items than other solutions. After the removing procedure, the parent population is selected from the new population.

5.2.3 A new sampling offspring method for MOPs

A new sampling method is introduced to generate the offspring of EDA for MOPs. When there are less than 10 non-dominated solutions in the population, there is only one Gaussian model to sample the offspring. When there are more than 10 non-dominated solutions in the population and some close solutions are removed, each leading non-dominated solution is directly set as the mean vector of one Gaussian model. This means some offspring solutions are specially sampled around these non-dominated solutions seen in Fig.5.5.

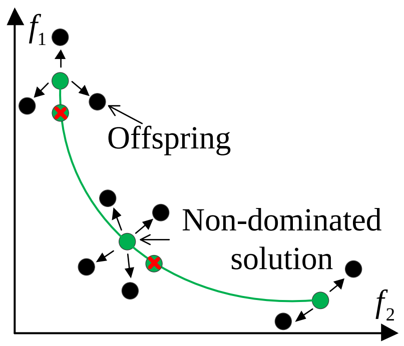


Figure 5.5 Sample offspring around leading best non-dominated solutions.

The Gaussian model with the 1st best non-dominated solution as the center produces 15 offspring, next the 2nd producing 12, and so on. Only the non-dominated solutions can be set as the mean vector. After these Gaussian models generate some offspring, the remaining offspring are all sampled by one Gaussian model. This model is built from the parent population in which the individuals are chosen by the modified extreme elitism selection. These individuals are selected from the population where the close top solutions have been removed. Compared to using only one Gaussian model to sample the offspring, this new sampling method could generate more diversified offspring and is more likely to produce uniformly distributed non-dominated solutions for some problems.

5.2.4 The performance metric of the algorithms for MOPs

In the present study, two metrics are utilized to measure the performance of the proposed algorithm for MOPs. These two metrics have been introduced by Deb et al. and are widely used to evaluate the performance of other algorithms for MOPs [59]. The first metric is called the distance metric Υ. It is illustrated in Fig. 5.6.

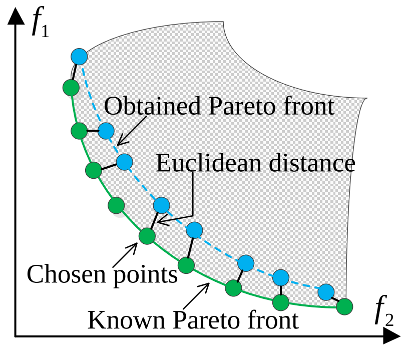


Figure 5.6 Distance metric.

The green curve is the Pareto optimal front, which is known for some benchmark test problems. The blue curve is the Pareto front obtained by the algorithms. First, a set of uniformly spaced solutions (H=500 or more) from the known Pareto-optimal front are found in the objective space. For each solution in the obtained Pareto front, the minimum Euclidean distance of it from the chosen solutions for the known Pareto optimal front is computed. The average of these distances is used as the first metric Υ. This metric measures the extent of convergence of the algorithms. For an algorithm, if the value of this metric is smaller, the convergence toward the Pareto optimal front is better.

The second metric is the diversity metric Δ, as shown in Fig.5.7. The two green solutions are extreme solutions, which have the minimum value of one objective in the feasible region. The blue solutions are the Pareto solutions obtained by the algorithms. d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions, respectively, of the obtained Pareto set. With N solutions in the obtained Pareto front, there are N-1 consecutive distances. $\bar{d}$ is the average of all the consecutive distances. The distance metric can be calculated by

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \tag{5.3}$$

An ideal distribution would make all distances equal to $\bar{d}$ and would make $d_f = d_l = 0$ (with the existence of extreme solutions in the obtained Pareto set). Then, the value of this metric is equal 0. A smaller value of this metric is better for an algorithm.

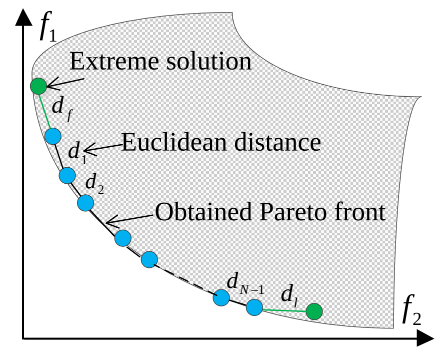


Figure 5.7 Diversity metric.

5.3 Experiments and analysis

Some benchmark MOPs are introduced now to test the algorithm proposed in the present study. Deb et al. have used these problems to measure the performance of the NSGA-II, SPEA and PAES [59]. Their experiment results will be compared with the results of the new algorithm.

5.3.1 MOPs with a few variables

Table 5.1 shows the description of some benchmark problems. The parameters of EDA-DM for these problems are different, since each problem has its own character. Every problem is tested 30 times independently. The population size and parent population size of these problems are both set at 100; and the time of iteration is set differently (SCH: 15, FON: 20, POL: 100 and KUR: 100). In every generation some non-dominated solutions are produced. Some of them may not be non-dominated in all generations. So, an archive is used to collect these non-dominated solutions from some generations (SCH: 4th, FON: 10th, POL: 20th and KUR: 20th). In the last generation, all these solutions in the archive are compared with each other to choose the final non-dominated solutions, which form the obtained Pareto front.

Table 5.1 Some benchmark MOPs with a few variables.

Problem	D	Interval	Objective functions	
SCH	1	$[-10^3, 10^3]$	$f_1(\boldsymbol{X})=x^2$	$f_2(\boldsymbol{X})=(x-2)^2$
FON	3	[-4, 4]	$f_1(\boldsymbol{X})=1-\exp(-\sum_{i=1}^{3}(x_i-1/\sqrt{3})^2)$	$f_2(\boldsymbol{X})=1-\exp(-\sum_{i=1}^{3}(x_i+1/\sqrt{3})^2)$
POL	2	$[-\pi, \pi]$	$f_1(\boldsymbol{X})=1+(A_1-B_1)^2+(A_2-B_2)^2$ $A_1=0.5\sin 1-2\cos 1+\sin 2-1.5\cos 2$ $A_2=0.5\sin x_1-2\cos x_1+\sin x_2-1.5\cos x_2$	$f_2(\boldsymbol{X})=(x_1+3)^2+(x_2+1)^2$ $B_1=1.5\sin 1-\cos 1+2\sin 2-0.5\cos 2$ $B_2=1.5\sin x_1-\cos x_1+2\sin x_2-0.5\cos x_2$
KUR	3	[-5, 5]	$f_1(\boldsymbol{X})=\sum_{i=1}^{D-1}(-10\exp(-0.2\sqrt{x_i^2+x_{i+1}^2}))$	$f_2(\boldsymbol{X})=\sum_{i=1}^{D}(\lvert x_i\rvert^{0.8}+5\sin x_i^3)$

Compared to NSGA-II and other algorithms, EDA/DM has fewer function evaluations. NSGA-II used the *PS* of 100 and *G* of 250. The total function evaluation time is 25,000 for all problems. Here EDA-DM uses less function evaluations to get a better result. For problem SCH, the function evaluation time is 2×100×15=3,000. EDA-DM has two populations: one is produced by EDA; the other is produced by DM. So, the actual population should be equal 2×100. For problems FON, POL and KUR, the function evaluations are 4,000, 20,000 and 20,000, respectively. Table 5.2 shows the test results of the distance metric Υ for these problems. For each problem, the first row is the mean value of the distance metric of 30 independent tests; the second row is the variance. EDA-DM can find a smaller distance metric than with other algorithms for SCH, FON and POL. Table 5.3 presents the mean and the variance of the diversity metric. EDA-DM can have the best diversity for SCH and FON.

Table 5.2 The mean and variance of the distance metric Υ**.**

	EDA-DM	NSGA-II (R)	NSGA-II (B)	SPEA	PAES
SCH	**0.000068**	0.003391	0.002833	0.003403	0.001313
	0.004648	0	0.000001	0	0.000003
FON	**0.000714**	0.001931	0.002571	0.125692	0.151263
	0.000796	0	0	0.000038	0.000905
POL	**0.011078**	0.015553	0.017029	0.037812	0.030864
	0.008225	0.000001	0.000003	0.000088	0.000431
KUR	0.060865	0.028964	**0.028951**	0.045617	0.057323
	0.045157	0.000018	0.000016	0.000050	0.011989
ZDT1	0.005424	0.033482	**0.000894**	0.001799	0.082085
	0.001644	0.004750	0	0.000001	0.008679
ZDT2	0.003867	0.072391	**0.000824**	0.001339	0.126276
	0.001466	0.031689	0	0	0.036877
ZDT3	**0.020623**	0.114500	0.043411	0.047517	0.023872
	0.020186	0.007940	0.000042	0.000047	0.000001
ZDT4	0.977169	**0.513053**	3.227636	7.340299	0.854816
	0.236948	0.118460	7.30763	6.572516	0.527238
ZDT6	**0.000435**	0.296564	7.806798	0.221138	0.085469
	0.000262	0.013135	0.001667	0.000449	0.006664

Table 5.3 The mean and the variance of the diversity metric Δ**.**

	EDA-DM	NSGA-II (R)	NSGA-II (B)	SPEA	PAES
SCH	**0.276182**	0.477899	0.449265	1.021110	1.063288
	0.004648	0.003471	0.002062	0.004372	0.002868
FON	**0.267022**	0.378065	0.395131	0.792352	1.162528
	0.000232	0.000639	0.001314	0.005546	0.008945
POL	1.020100	**0.452150**	0.503721	0.972783	1.020007
	0.001512	0.002868	0.004656	0.008475	0

	EDA-DM	NSGA-II (R)	NSGA-II (B)	SPEA	PAES
KUR	0.658487 0.000055	**0.411477** 0.000992	0.442195 0.001498	0.852990 0.002619	1.079838 0.013772
ZDT1	0.419483 0.009913	**0.390307** 0.001876	0.463292 0.041622	0.784525 0.004440	1.229794 0.004839
ZDT2	0.508422 0.035599	0.430776 0.004721	0.435112 0.024607	**0.001339** 0	0.126276 0.036877
ZDT3	0.658033 0.010500	0.114500 0.007940	0.043411 0.000042	0.047517 0.000047	**0.023872** 0.000001
ZDT4	0.711919 0.009192	0.702612 0.064648	**0.479475** 0.009841	0.798463 0.014616	0.870458 0.101399
ZDT6	1.450446 0.002192	0.668025 0.009923	**0.644477** 0.035042	0.849389 0.002713	1.153052 0.003916

Fig.5.8 and 5.9 show the obtained Pareto front (pink circles) by EDA-DM for SCH, FON, POL and KUR. The curve is the true Pareto optimal front.

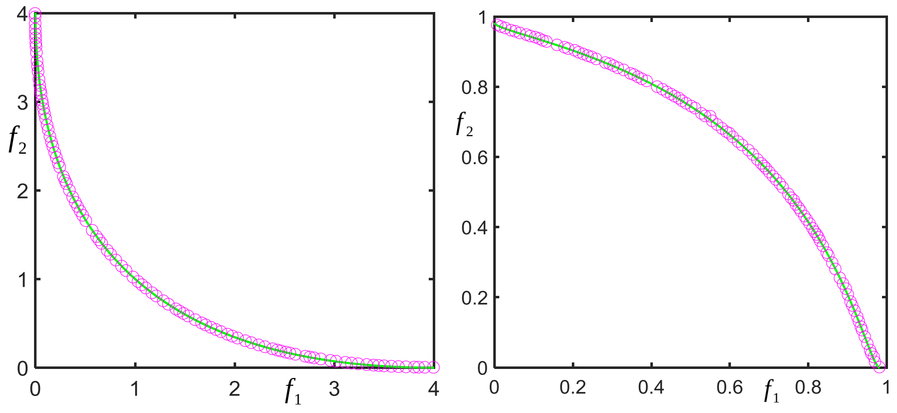


Figure 5.8 Obtained Pareto front by EDA-DM for SCH and FON.

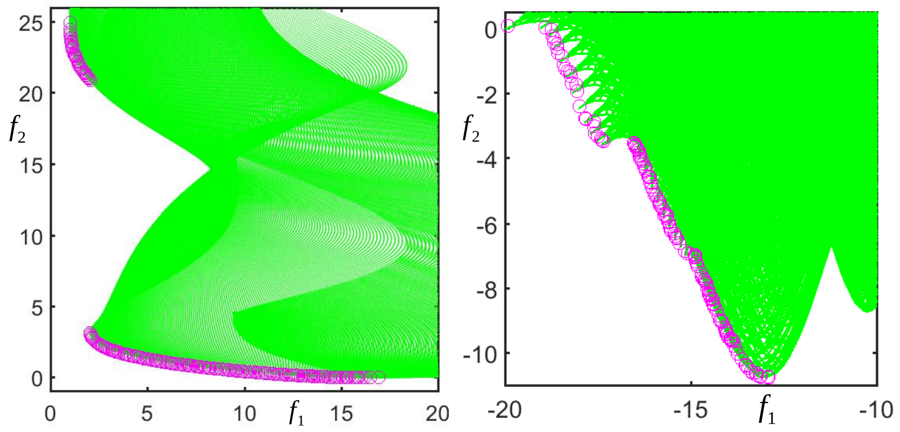


Figure 5.9 Obtained Pareto front by EDA-DM for POL and KUR.

It is seen that the final obtained non-dominated solutions can converge to the true Pareto front very closely. Besides, these solutions can uniformly spread in the front. In Fig.5.9, the green surfaces are the mapping of the feasible region of the function POL in the objective value space. The non-dominated solutions found by EDA-DM can spread the front of the mapping surface.

5.3.2 MOPs with more variables

Table 5.4 Some benchmark MOPs with more variables.

Problem	D	Interval	Objective functions	
ZDT1	30	[0, 1]	$f_1(\boldsymbol{X})=x_1$ $g(\boldsymbol{X})=1+9(\sum_{i=2}^{D} x_i)/(D-1)$	$f_2(\boldsymbol{X})=g(\boldsymbol{X})(1-\sqrt{x_1/g(\boldsymbol{X})})$
ZDT2	30	[0, 1]	$f_1(\boldsymbol{X})=x_1$ $g(\boldsymbol{X})=1+(9\sum_{i=2}^{D} x_i)/(D-1)$	$f_2(\boldsymbol{X})=g(\boldsymbol{X})\{1-[x_1/g(\boldsymbol{X})]^2\}$
ZDT3	30	[0, 1]	$f_1(\boldsymbol{X})=x_1$ $g(\boldsymbol{X})=1+9(\sum_{i=2}^{D} x_i)/(D-1)$	$f_2(\boldsymbol{X})=g(\boldsymbol{X})[1-\sqrt{x_1/g(\boldsymbol{X})}-x_1\sin(10\pi x_1)/g(\boldsymbol{X})]$
ZDT4	10	$x_1\in[0, 1]$ $x_i\in[0, 5]$, $i=2,\cdots,D$	$f_1(\boldsymbol{X})=x_1$ $g(\boldsymbol{X})=1+10(D-1)+\sum_{i=2}^{D}[x_i^2-10\cos(4\pi x_i)]$	$f_2(\boldsymbol{X})=g(\boldsymbol{X})[1-\sqrt{x_1/g(\boldsymbol{X})}]$
ZDT6	10	[0, 1]	$f_1(\boldsymbol{X})=1-\exp(-4x_1)\sin^6(6\pi x_1)$ $g(\boldsymbol{X})=1+9[(\sum_{i=2}^{D} x_i)/(D-1)]^{0.25}$	$f_2(\boldsymbol{X})=g(\boldsymbol{X})\{1-[f_1(\boldsymbol{X})/g(\boldsymbol{X})]^2\}$

ZDT1-ZDT6 has more problem variables as seen in Table 5.4. The problems are named using the first letter of the names of the authors Zitzler, Deb and Thiele [109]. In view of the No Free Lunch theorem, there is no universal algorithm that can always have good performance for all the problems. So EDA-DM is modified for these problems. The univariate marginal distribution algorithm (UMDA) [22] in binary code is adopted for ZDT1 and ZDT2. The probabilistic model of this EDA does not consider the interaction between variables. The joint probability density function of this model can be expressed by

$$f_{\boldsymbol{X}}(x_1,\cdots,x_D)=\prod_{i=1}^{D}\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp(-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}) \tag{5.4}$$

Each variable can be presented by a string of binary values 1 or 0. Fig.5.10 presents an example, which has 5 solutions in the parent population and the length of the string is set as 10. Then the number 1s in every column is counted. For example, the first position of the string of variable x_1 has three 1s, so the percentage (fraction) of 1s equals 3/5=0.6. The probabilistic model vector consists of all percentages [0.6 0.6 ⋯ 0.2; 0.8 0.4 ⋯ 0.6; ⋯; 0.6 0.4 ⋯ 0.6].

$$\begin{array}{cccccccccccccc}
 & \multicolumn{4}{c}{\overbrace{\qquad\qquad\qquad\qquad}^{x_1}} & \multicolumn{4}{c}{\overbrace{\qquad\qquad\qquad\qquad}^{x_2}} & \cdots & \multicolumn{4}{c}{\overbrace{\qquad\qquad\qquad\qquad}^{x_D}} \\
1 & 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 1 & \cdots & 0 & 1 & \cdots & 1 \\
2 & 0 & 1 & \cdots & 1 & 1 & 0 & \cdots & 0 & \cdots & 1 & 0 & \cdots & 0 \\
3 & 1 & 1 & \cdots & 0 & 0 & 1 & \cdots & 1 & \cdots & 0 & 1 & \cdots & 1 \\
4 & 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 1 & \cdots & 1 & 0 & \cdots & 0 \\
5 & 0 & 1 & \cdots & 0 & 1 & 1 & \cdots & 0 & \cdots & 1 & 0 & \cdots & 1 \\
\hline
 & 3 & 3 & \cdots & 1 & 4 & 2 & \cdots & 3 & \cdots & 3 & 2 & \cdots & 3 \\
 & 0.6 & 0.6 & \cdots & 0.2 & 0.8 & 0.4 & \cdots & 0.6 & \cdots & 0.6 & 0.4 & \cdots & 0.6
\end{array}$$

Figure 5.10 Binary probability vector.

When producing the offspring, the value 1 at each position in the binary string is sampled from the probability vector. Compared to a real code, the binary code can do microcosmic mutation and it is more suitable for some problems. When selecting the parent individuals, as usual, the solutions are sorted from the best to the worst. Some too close solutions in the non-dominated solution group are removed when there are more than 10 non-dominated solutions in the population. After that, the first top solution takes 15 items in the parent population, the second one takes 12 items, and so on. The size of parent population is the same as the population size and there is only one Gaussian centre to generate the offspring all the time for these two problems. The population size of these two problems is 200; the iteration time is 100 and the non-dominated solutions are saved in the archive from the 85th generation for ZDT1; the iteration time is 180 and the non-dominated solutions are saved from the 60th generation for ZDT2.

Fig.5.11 shows the convergence results of the two problems. It is seen that the finial non-dominated solution set could approach the true Pareto front represented by the curve very closely. Moreover, EDA-DM is able to obtain many non-dominated solutions, which can uniformly spread in the obtained Pareto front.

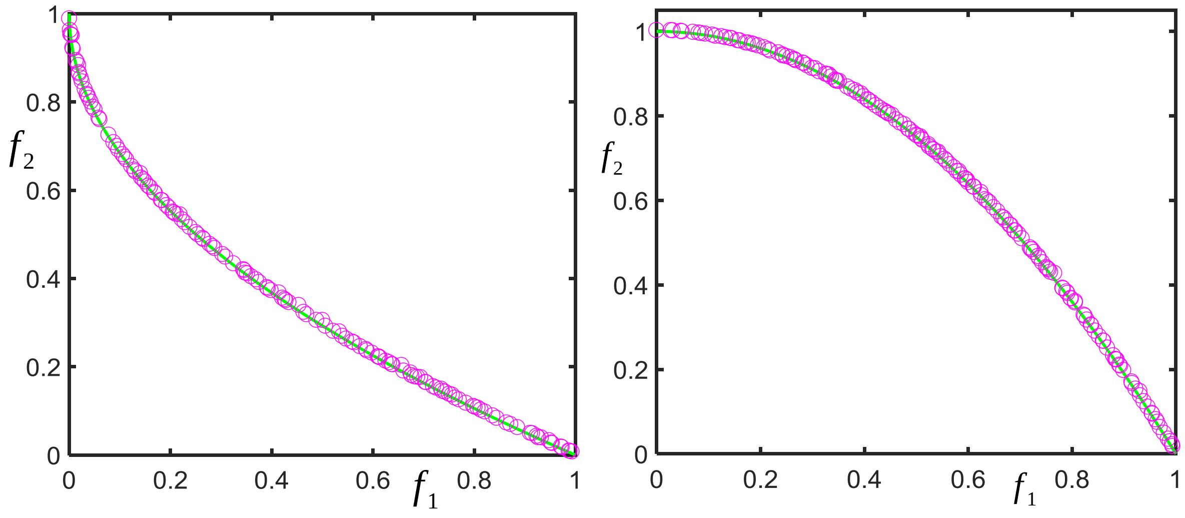


Figure 5.11 Obtained Pareto front by EDA-DM for ZDT1 and ZDT2.

ZDT3 and ZDT4 still adopt the univariate marginal distribution model, but use the real code to program them. The modified extreme elitism selection and the new sampling offspring method are adopted. But DM is not combined to produce the half size of offspring. It is found that the test result of only EDA is better than that of the hybrid algorithm. This may be since these two problems need a less diversified population. The *PS* of the two problems is set at 200 and the parent population size is 160 for ZDT3, 100 for ZDT4; *G* is 80 for ZDT3 and 100 for ZDT4; from the 50th generation, the non-dominated solutions are saved in the archive for them. Fig.5.12 shows the convergence results of the two problems. ZDT3 can obtain a group of widely distributed solutions; for ZDT4, EDA cannot obtain a Pareto front very close to the true one.

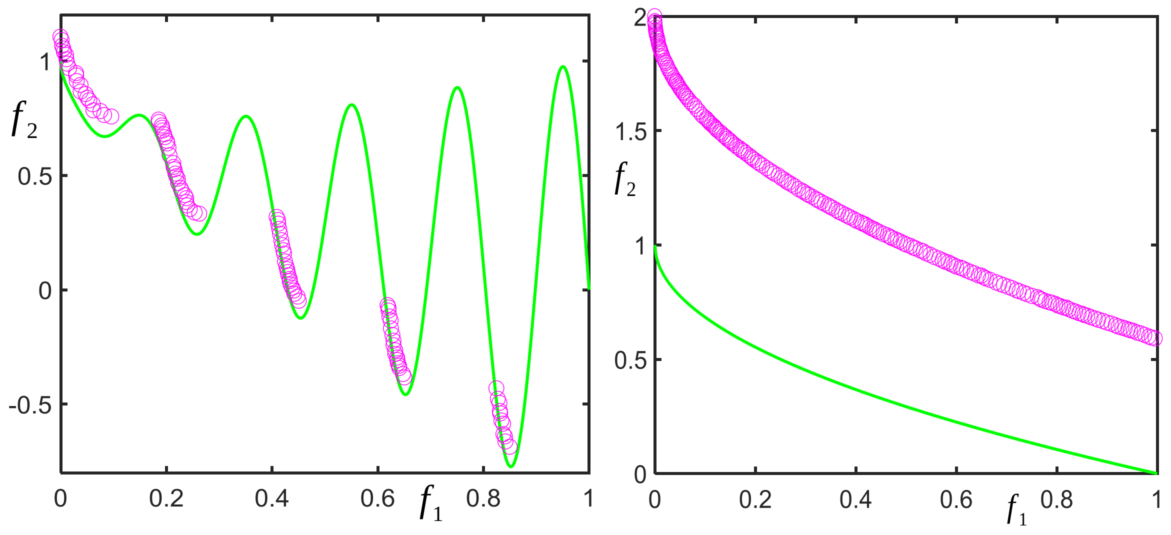


Figure 5.12 Obtained Pareto front by EDA-DM for ZDT3 and ZDT4.

ZDT6 uses the binary code to program and the selection method developed in NASA-II to choose the parent population. The method first divides the population into different levels according to the domination relationship. The non-dominated solutions among the entire population are put in the first level and then these solutions are removed from the population. The remaining solutions do the domination sorting again, so some new non-dominated solutions are found and they are put in the second level, and so on. Then the solutions in the same level are ranked according to the crowding distance. Finally, binary tournament selection is adopted. The solution in the higher level wins over the ones at the lower level; if two solutions are in the same level, the solution with a smaller crowding distance wins over the ones with bigger crowding distance. After the parent population is chosen, two EDAs: population based incremental learning (PBIL) [24] and univariate marginal distribution algorithm (UMDA in binary code) are integrated to produce the probabilistic model. PBIL uses the concept of learning rate to control the converge speed. In every generation, the probabilistic model vector is updated according to

$$p_{g+1}(\boldsymbol{X}) = (1-\gamma)\, p_g(\boldsymbol{X}) + \gamma\, p_{\text{current}}(\boldsymbol{X}) \tag{5.5}$$

Here, $p_{g+1}(\boldsymbol{X})$ is the joint probability vector in every generation, $p_g(\boldsymbol{X})$ is the joint probability vector in the last generation and $p_{\text{current}}(\boldsymbol{X})$ is the joint probability vector that is obtained from the parent population. Also, $\gamma \in (0, 1)$ is the learning rate, and a bigger γ leads to faster convergence. However, if the convergence is too fast, the algorithm may prematurely converge. The probability vector of $p_{\text{current}}(\boldsymbol{X})$ is obtained by the algorithm of UMDA and the procedure is seen in Fig. 5.10. DM cooperates to generate the other half size population. The *PS* for ZDT6 is 200, *G* is 70 and the non-dominated solutions are saved from the 60th generation. The pool (parent population) size is 150 and the learning rate α is set at 0.01. Fig. 5.13 shows the obtained Pareto front of ZDT6 by EDA-DE. It is seen that this algorithm can find a Pareto front that could approach the true one very closely, but the diversity is not good enough. From Table 5.2 it is seen that EDA and EDA-DM can find a more accurate Pareto front for ZDT3 and ZDT6 than with other algorithms. For other problems, these two algorithms can also have good performance, except ZDT4.

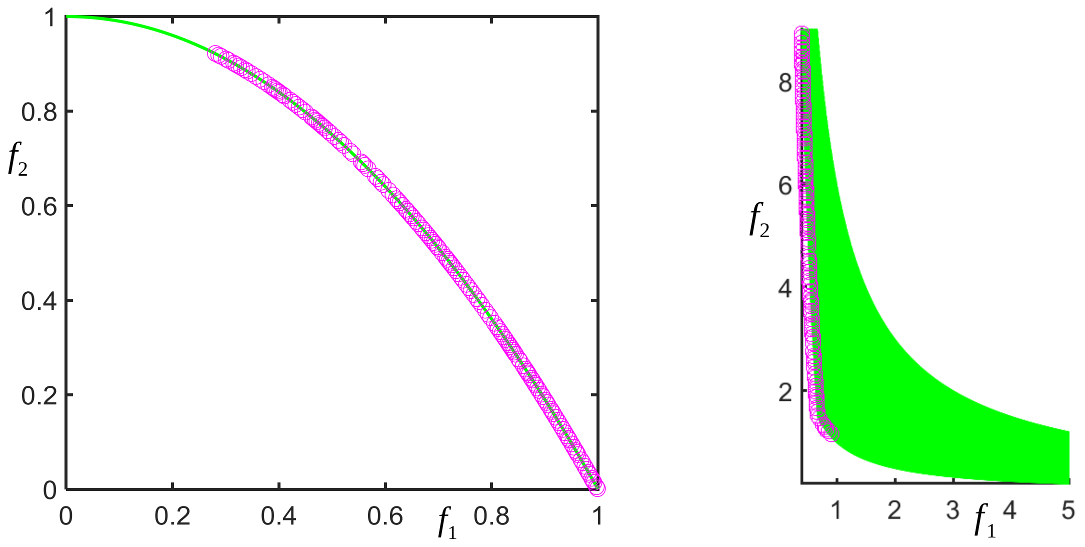


Figure 5.13 Obtained Pareto front by EDA-DM for ZDT6 and CONSTR.

5.3.3 MOPs with inequality constraints

The MOPs: CONSTR, SRN and TNK have some inequality constraints seen in Table 5.5. When the promising solutions are selected as the parent, the feasible non-dominated solutions are given the priority. First, the feasible solutions are found in the population. Second, the domination sorting is operated among these feasible solutions. These feasible solutions, which are dominated by the fewest other solutions are on the top. After that, the solutions are sorted from the best to the worst. Similarly, the role of these leading five best solutions is highlighted in the evolution. If there are less than 10 non-dominated feasible solutions in the parent population, there is only one multivariate Gaussian model to sample the offspring. When there are more than 10 non-dominated feasible solutions in the population, after removing some very close solutions, there will be more multivariate Gaussian models to generate the offspring. The *PS* for these problems is 100 and the parent population size is the same. The *G* of CONSTR, SRN and TNK is 70, 15 and 100, respectively; all of them save the non-dominated solutions in the archive from 5th generation. The function evaluation time for these problems is 14,000, 3,000 and 20,000, respectively. Fig. 5.13 and 5.14 present the convergence of EDA-DM for these problems. It is seen that EDA-DM can find some non-dominated solutions and they can evenly spread along the border of the mapping surface of the feasible region. It should be mentioned that all the MOPs have two objective functions in the present research. However, some real-world problems may have more than two objective functions, and the proposed algorithm should be improved to handle them. EDA-DM selects the solutions as the parents in light of two criteria. One is the domination relationship. The non-domination solutions are always ahead. Among these solutions, the solutions that can dominate more solutions are placed ahead. The other is the distance between two solutions. If two solutions are very close in a non-dominated group, one solution will be removed from the group. Future work may employ more criteria to select the parents and improve the population diversity, in multi-objective problems.

Table 5.5 Some benchmark MOPs with inequality constraints.

Problem	D	Interval	Objective functions	Constraints
CONSTR	2	$x_1 \in [0.1, 1]$ $x_2 \in [0, 5]$	$f_1(\boldsymbol{X})=x_1$ $f_2(\boldsymbol{X})=(1+x_2)/x_1$	$g_1(\boldsymbol{X})= x_2+9x_1 \geq 6$ $g_2(\boldsymbol{X})= -x_2+9x_1 \geq 1$
SRN	2	$x_i \in [0, 20]$ i=1, 2	$f_1(\boldsymbol{X})=(x_1-2)^2+(x_2-1)^2+2$ $f_2(\boldsymbol{X})=9x_1-(x_2-1)^2$	$g_1(\boldsymbol{X})= x_1^2+ x_2^2 \leq 225$ $g_2(\boldsymbol{X})= x_1-3x_2 \leq -10$
TNK	2	$x_i \in [0, \pi]$ i=1, 2	$f_1(\boldsymbol{X})=x_1$ $f_2(\boldsymbol{X})=x_2$	$g_1(\boldsymbol{X})= -x_1^2- x_2^2 +1+0.1\cos[16\arctan(x_1/x_2)] \leq 0$ $g_2(\boldsymbol{X})= (x_1-0.5)^2-(x_2-0.5)^2\leq 0.5$

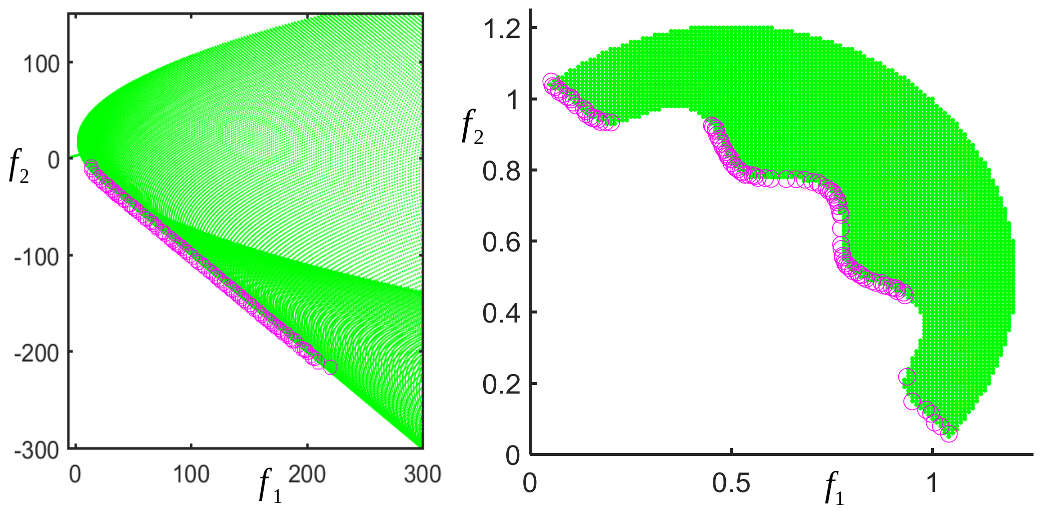


Figure 5.14 Obtained Pareto front by EDA-DM for SRN and TNK.

5.4 Summary

In this chapter, estimation distribution algorithm combing differential mutation (EDA-DM) was proposed to handle multi-objective optimization problems (MOPs). EDAs based on the Gaussian distribution with the extreme elitism selection had a fast convergence rate while DM could improve the population diversity. In view of the no-free-lunch theorem, a universal optimization strategy is impossible. So, some problems used a univariate marginal Gaussian model while some adopted a multivariate Gaussian model; some preferred real code while some others preferred binary code. In addition, more Gaussian models were adopted to sample the offspring for some problems and this operation could advance the diversity of the obtained non-dominated solution set. The experiment results indicated that the hybrid algorithm (EDA-DM) could use fewer function evaluations than some typical algorithms to get good performance for some MOPs.

Chapter 6: Conclusions and Future Work

6.1 Conclusions

Optimization plays a very important role in many areas in the human society. This dissertation proposed innovative estimation distribution algorithms (EDAs) to solve various types of optimization problems, particularly in the domain of engineering. EDAs first randomly generate an initial population, then select some promising solutions according to the fitness function, constituting a parent population. Based on the parent population, a probabilistic model is built and the offspring is sampled from the model. After that, the selection and sampling of offspring repeat for some generations until the termination requirement is satisfied. In this manner, the optimal or some high quality solutions are obtained. The main conclusions of the present research are listed below.

1. The EDAs developed in this thesis can search a group of solutions in parallel, in a large space. They do not require such analytical information as differentiability and continuity of the optimization problem. Also, they are not sensitive to the convexity or connectivity of feasible regions. In the Gaussian distribution, the mean vector determines the search direction. The extreme elitism selection can improve the performance of the EDA based on univariate marginal Gaussian distribution (UMGD) for solving a set of low-dimensional and high-dimensional benchmark problems. In particular, this EDA can still find the optimal solutions for problems with 100 and 200 variables.

2. The EDA based on UMGD with extreme elitism selection, as developed in the present work, did not rely on the structure of the robotic arms or require initial guesses when solve the inverse displacement problem (IDP) of the robotic arms. This EDA was able to obtain

satisfactory solutions for IDP of the 4-DOF Barrett WAM Arm more rapidly and steadily than the EDA methods with truncation, tournament and proportional selection. The developed EDA in combination with the differential mutation was implemented to handle the IDP of the 7-DOF Barrett WAM Arm. The differential mutation was found to improve the stability of the EDA in obtaining satisfactory solutions. The developed algorithm provides an effective method to solve the IDP of robotic arms having high degrees freedom (even more than 7-DOF).

3. Existing EDAs have focused on solving the optimization problems having only the constraints of variable boundaries. The present thesis proposed and developed EDAs with extreme elitism selection, which was able to handle optimization problems with inequality, equality, linear, nonlinear, continuous, and discontinuous constraints. The developed EDA based on a multivariate Gaussian distribution model with extreme elitism selection can outperform other existing EDAs and state-of-the-art algorithms for most benchmark problems. In addition, the developed EDA was shown to have good performance in four mechanical design problems. The present research promotes and facilitates the practical application of EDAs in many real-world optimization problems with different types of constraints.

4. The combination of EDAs with differential mutation (DM), as developed in the present thesis, sought to find a set of uniformly distributed Pareto optimal solutions for multi-objective optimization problems (MOPs). It was able to provide more alternative choices for decision makers and strike a trade-off between different objectives to some extent. In the evolution, the EDA guides the search direction while DM is able to enrich the population diversity. The new sampling method that uses more Gaussian distribution models to generate the offspring can facilitate the obtained non-dominated solutions to evenly spread in the Pareto front. The

algorithm proposed in the present work uses fewer function evaluations than in some well-known algorithms, to get better performance for some benchmark MOPS.

6.2 Possible future work

There are still some limitations of EDAs. EDAs usually need a large population size since they adopt sampling to generate the offspring. More individuals mean more opportunities to obtain high quality solutions. However, a large population may lead to a high computational cost, especially, when solving optimization problems that have many variables. In future work, a dynamic population may be designed for EDAs. Sometimes it is not necessary to maintain a big population all the time during the evolution process. In the beginning, sampling more offspring is beneficial; but after some iterations, the search direction may already approach the optimal solution closely and the search scope can be reduced. Hence, fewer offspring may be adequate to find the optimal solution. The population can adjust its size dynamically and does not need to always maintain a large population. As a result, the computing cost will decrease.

This thesis used 4-DOF and 7-DOF robotic arms to test the performance of the proposed EDA. The developed algorithm for the robot inverse displacement problem still needs to be tested on high degree freedom (more than 7-DOF) robotic arms to evaluate its performance. When solving optimization problems having various types of constraints, the proposed EDA sometimes is unable to find the optimal solutions for the problems having equality constraints. This is because these problems usually have a very small feasible region. An effective approach to handle equality constraints should be investigated. Besides, compared to some other state-of-the-art algorithms, this developed algorithm used more function evaluations to find the optimal solutions. Hence, a dynamic population may be also implemented in the problems with different types of constraints. Concerning multi-objective optimization problems, the proposed EDAs solved the problems only with two objectives. It is important to promote EDAs to handle problems with more objectives. Finally, EDAs should be extended to more and diverse optimization applications in various engineering fields.

Bibliography

[1] A. Andreas, W. S. Lu. *Practical optimization algorithms and engineering applications*. New York: Springer Science + Business Media, LLC, pp.1-2, 2007. Print.

[2] H. Wang, Z. J. Wu, S. Rahnamayan, Y. Liu, M. Ventresca. Enhancing particle swarm optimization using generalized opposition-based learning. *Information Sciences*, 181(20): 4699-4714, 2011.

[3] B. Moradabadi, M. M. Ebadzadeh, M. R. Meybodi. A new real-coded stochastic Bayesian optimization algorithm for continuous global optimization. *Genet Program Evolvable Mach*, 17:145-167, 2016.

[4] Q. Yang, W. N. Chen, Y. Li, C. L. P. Chen, X. M. Xu, J. Zhang. Multimodal estimation of distribution algorithms. *IEEE Transactions on Cybernetics ON CYBERNETICS*, 47(3): 636-650, 2017.

[5] L. F. Wang, J. C. Zeng. Estimation of Distribution Algorithm Based on Copula Theory. *Exploitation of Linkage Learning in Evolutionary Algorithms*, vol.3, pp.139-162, 2010.

[6] Yu. Wang, B. Li, T. Weise. Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. *Information Sciences*, 180(12):2405-2420, 2010.

[7] W. S. Dong, X. Yao. NichingEDA: utilizing the diversity inside a population of EDAs for continuous optimization. *IEEE World Congress on Computational Intelligence*, pp.1260-1267, 2008.

[8] W. S. Dong, X. Yao. Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms. *Information Sciences*, 178(15):3000-3023, 2008.

[9] P. Larrañaga, J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell: Kluwer Academic Publishers, pp.181-193, 2001. Print.

[10] H. Müehlenbein, T. Mahnig. Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology*, 7(1):19-32, 1999.

[11] M. Pelikan, D. E. Goldberg, E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp.525-532, 1999.

[12] Y. Gao, J. Culberson. Space Complexity of Estimation of Distribution Algorithms. *Evolutionary Computation*, 13(1):125-143, 2005.

[13] J. Q. Gan, E. Oyama, E. M. Rosales, H. Hu. A Complete Analytical Solution to the Inverse Kinematics of the Pioneer 2 Robotic Arm. *ROBOTICA*, 23(1): 123-129, 2005.

[14] M. Shimizu, H. Kakuya, W. Yoon, K. Kitagak, K. Kosuge. Analytical inverse kinematic computation for 7-DOF redundant manipulators with joint limits and its application to redundancy resolution. *IEEE Transactions on Robotics*, 24(5):1131-1142, 2008.

[15] I. A. Vasilyev, A. M. Lyashin. Analytical solution to inverse kinematic problem for 6-DOF robot-manipulator. *Automation and Remote Control*, 71(10): 2195–2199, 2010.

[16] Y. J. Zhao, T. Huang, Z. Y. Yang. A new numerical algorithm for the inverse position analysis of all serial manipulators. *ROBOTICA*, vol.24, pp.373–376, 2006.

[17] S. Kucuk, Z. Bingul. Inverse kinematics solutions for industrial robot manipulators with offset wrists. *Applied Mathematical Modelling*, vol.38, pp.1983-1999, 2014.

[18] V. Kumara, S. Sena, S. S. Royb, S. K Dasa, S. N. Shomea. Inverse kinematics of redundant manipulator using interval newton method. *International Journal of Engineering and Manufacturing*, doi: 10.5815/ijem.2015.02.03, 2015.

[19] J. Grahl, F. Rothlauf. PolyEDA: Combining estimation of distribution algorithms and linear

inequality Constraints. In: *Proceedings of the Genetic and Evolutionary Computation–GECCO*, pp.1174-1185, 2004.

[20] P. A. Simionescu, D. Beale, G. V. Dozier. Constrained optimization problem solving using estimation of distribution algorithms. In: *Proceedings of the Congress on Evolutionary Computation (IEEE Cat.* No. *04TH8753) Conference: Evolutionary Computation*, pp.296-302, 2004.

[21] Z. P. Wan, L. J. Mao, G. M. Wang. Estimation of distribution algorithm for a class of nonlinear bi-level programming problems. *Information Sciences*, Vol.256, pp.184-196, 2014.

[22] H. Mühlenbein, G. PaaB. From recombination of genes to the estimation of distributions I. binary parameters. In: *Parallel Problem Solving from Nature*, pp. 178-187, 1996.

[23] M. Hauschild, M. Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, vol.1, pp.111-128, 2011.

[24] S. Baluja. Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. *Technical report*, No.CMU-CS-94-163, *Carnegie Mellon University*, 1994.

[25] G. R. Harik, F. G. Lobo, D. E. Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4): 287-297, 1999.

[26] J. S. Bonet, C. L. Isbell, P. Viola. MIMIC: finding optima by estimating probability Densities. *Advances in Neural Information Processing Systems 9*, vol.9, pp.424-431, 1997.

[27] M. Pelikan, H. Mühlenbein. The bivariate marginal distribution algorithm. *Advances in Soft Computing*, vol.9, pp.521-535, 1999.

[28] P. Larrañaga, R. Etxeberria, J. A. Lozano, J. M. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In: *Proceedings of the Workshop in*

Optimization by Building and Using Probabilistic Models, pp.201-204, 2000.

[29] P. Larrañaga, J. A. Lozano, E. Bengoetxea. Estimation of distribution algorithms based on multivariate normal and Gaussian networks. *Technical Report KZZA-IK-1-01, Department of Computer Science and Artificial Intelligence, University of the Basque Country*, 2001.

[30] P. A. N. Bosman, D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In: *Workshop Proceedings of the Genetic and Evolutionary Computation Conference*, pp.197-200, 2000.

[31] C. W. Ahn, R. S. Ramakrishna, D. E. Goldberg. Real-coded Bayesian optimization algorithm: bringing the strength of BOA into the continuous world. In: *Proceedings of the Genetic and Evolutionary Computation – GECCO*, pp.840-851, 2004.

[32] H. Handa. The effectiveness of mutation operation in the case of Estimation of Distribution Algorithms. *BioSystems*, 87(2-3): 243-251, 2007.

[33] S. H. Chen, M. C. Chen, P. C. Chang, Q. F. Zhang, Y. M. Chen. Guidelines for Developing effective estimation of distribution algorithms in solving single machine scheduling problems. *Expert Systems with Applications*, 37(9): 6441–6451, 2010.

[34] H. Karshenas, R. Santana, C. Bielza, P. Larrañaga. Regularized continuous estimation of distribution algorithms. *Applied Soft Computing*, 13(5): 2412-2432, 2013.

[35] C. W. Ahn, J. An, J. Yoo. Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs. *Information Sciences*, vol.192, pp.109-119, 2012.

[36] Q. B. Zhang, S. Kang, J. X. Gao, S. Wu, Y. P. Tian. An artificial immune univariate marginal distribution algorithm. *Computational Intelligence and Intelligent Systems*, vol.51, pp.66-75, 2009.

[37] C. F. Lima, F. G. Lobo, M. Pelikan, D. Goldberg. Model accuracy in the Bayesian optimization algorithm. *Soft Computing*, 15(7): 1351-1371, 2010.

[38] Q. F. Zhang, H. Mühlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):127-136, 2004.

[39] T. S. Chen, K. Tang, G. L. Chen, X. Yao. Analysis of computational time of simple estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):1-22, 2010.

[40] J. Grahl, S. Minner, F. Rothlauf. Behavior of UMDA$_C$ with Truncation Selection on Monotonous Functions. *IEEE Congress on Evolutionary Computation*, vol.3, pp.2553-2559, 2005.

[41] R. Rastegar. On the optimal convergence probability of univariate estimation of distribution algorithms. *Evolutionary Computation*, 19(2): 225-248, 2011.

[42] S. Muelas, A. Mendiburu, A. LaTorre, J. M. Peña. Distributed estimation of distribution algorithms for continuous optimization: How does the exchanged information influence their behavior? *Information Sciences*, vol.268, pp.231-254, 2014.

[43] J. Ceberio, E. Irurozki, A. Mendiburu, J. A. Lozano. A Distance-Based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Transactions on Evolutionary Computation*. 18(2): 286-300, 2014.

[44] X. C. Hao, J. Z. Wu, C. F. Chien, M. Gen. The cooperative estimation of distribution algorithm: a novel approach for semiconductor final test scheduling problems. *Journal of Intelligent Manufacturing*, 25(5): 867-879, 2014.

[45] S. Bashir, M. Maeem, A. A. Khan, S. Shah. An application of univariate marginal distribution algorithm in MIMO communication systems. *International Journal of Communication Systems*, 23(1):109-124, 2009.

[46] W. Gu, Y. G. Wu, G. Y. Zhang. A hybrid univariate marginal distribution algorithm for dynamic economic dispatch of units considering valve-point effects and ramp rates. *International transactions on electrical energy systems*, 25(2): 374-392, 2013.

[47] I. Cruz-Aceves, J. G. Avina-Cervantes, J. M. Lopez-Hernandez, M. G. Garcia-Hernandez, M. Torres-Cisneros, H. J. Estrada-Garcia, A. Hernandez-Aguirre. Automatic image segmentation using active contours with univariate marginal distribution. *Mathematical Problems in Engineering*, doi:10.1155/2013/419018, 2013.

[48] S. C. Shital, N. R. Babu. Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach. *Engineering Applications of Artificial Intelligence*, 23(7): 1083-1092, 2010.

[49] M. Ayyıldız, K. Çetinkaya. Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator. *Neural Computing and Applications*, 27(4): 825-836, 2016.

[50] S. Venkatraman, G. G. Yen. A generic framework for constrained optimization using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(4): 424-435, 2005.

[51] Y. Wang, Z. Cai, Y. Zhou, W. Zeng. An adaptive trade-off model for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 12(1): 80-92, 2008.

[52] E. M. Montes, O. C. Domínguez. Empirical analysis of a modified Artificial Bee Colony for constrained numerical optimization. *Applied Mathematics and Computation*, 218(22): 10943-10973, 2012.

[53] M. K. Dhadwal, S. N. Jung, C. J. Kim. Advanced particle swarm assisted genetic algorithm for constrained optimization problems. *Computational Optimization and Application*, vol.58, pp. 781-806, 2014.

[54] P. Yang, K. Tang, X. F. Lu. Improving estimation of distribution algorithm on multimodal problems by detecting. *IEEE Transactions on Cybernetics*, 45(8): 1438-1449, 2015.

[55] H. Karshenas, R. Santana, C. Bielza, P. Larrañaga. Regularized continuous estimation of distribution algorithms. *Applied Soft Computing*, 13(5): 2412-2432, 2013.

[56] E. Zitzler, L. Thiele. Multi-objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4): 252-271, 1999.

[57] J. D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: *Proceedings of the 1st International Conference on Genetic Algorithms*, pp.93-100, 1985.

[58] J. D. Knowles, D. W. Corne. Approximating the Non-dominated Front Using the Pareto Archived Evolution Strategy. *Evolution Computation*, 8(2): 149-172, 2006.

[59] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolution Computation*, 6(2): 182-197, 2002.

[60] Q. F. Zhang, H. Li. MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decompositiong. *IEEE Transactions on Evolution Computation*, 11(6): 713-731, 2007.

[61]H. Jain, K. Deb. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Non-dominated Sorting Approach, Part II: Handling Constraints and

Extending to an Adaptive Approach. *IEEE Transactions on Evolution Computation*, 18(4): 602-622, 2014.

[62] C. A. Coello, N. C. Cortés. Solving Multiobjective Optimization Problems Using an Artificial Immune System. *Genetic Programming and Evolvable Machines*, 6(2): 163-190, 2005.

[63] C. Dai, Y. P. Wang, M. Ye. A new multi-objective particle swarm optimization algorithm based on decomposition. *Information Sciences*, vol.325, pp. 541-557, 2015.

[64] S. B. Andersen, I. F. Santos. Evolution strategies and multi-objective optimization of permanent magnet motor. *Applied Soft Computing*, 12(2): 778-792, 2012.

[65] J. M. Chaves-González, M. A. Vega-Rodríguez. DNA strand generation for DNA computing by using a multi-objective differential evolution algorithm. *Biosystems*, vol.116, pp. 49-64, 2014.

[66] V. A. Shim, K. C. Tan, C. Y. Cheong. A Hybrid Estimation of Distribution Algorithm with Decomposition for Solving the Multi-objective Multiple Traveling Salesman Problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* (*Applications and Reviews*), 42(5): 682-691, 2012.

[67] H. Karshenas, R. Santana, C. Bielza, P. Larrañaga. Multi-objective Estimation of Distribution Algorithm Based on Joint Modeling of Objectives and Variables. *IEEE Transactions on Evolution Computation*, 18(4): 519-542, 2014.

[68] R. Santana, C. Echegoyen, A. Mendiburu, C. Bielza, J. A. Lozano, P. Larrañaga, R. Armañanzas, S. Shakya. MATEDA: A suite of EDA programs in Matlab. *Research Report EHU-KZAA-IK-2/09, Department of Computer Science and Artificial Intelligence University of the Basque Country*, 2009.

[69] H. Mühlenbein, D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evolutionary Computation*, 1(1): 25-49, 1993.

[70] J. H. Holland. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. *MIT Press Cambridge, MA, USA, ISBN*: *0262082136*, 1992.

[71] D. E. Goldberg, B. Korb, K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5): 493-530, 1989.

[72] T.-L. Yu, K. Sastry, D. E. Goldberg, M. Pelikan. Population sizing for entropy-based model building in genetic algorithms (IlliGAL Report No. 2006020). *Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory*, 2006.

[73] C. F. Lima, M. Pelikan, D. E. Goldberg, F. G. Lobo, K. Sastry, M. Hauschild. Influence of Selection and Replacement Strategies on Linkage Learning in BOA (MEDAL Report No. 2007005). *University of Missouri–St. Louis, Missouri Estimation of Distribution Algorithms Laboratory*, 2007.

[74] Y. Hong, S. Kwon, Q. S. Ren, X. Wang. Over-Selection: An Attempt to Boost EDA under Small Population Size. *IEEE Congress on Evolutionary Computation*, pp. 1075-1082, 2007.

[75] A. E. I. Brownlee, J. A. W. McCall, Q. F. Zhang, D. F. Brown. Approaches to selection and their effect on fitness modelling in an Estimation of Distribution Algorithm. *IEEE World Congress on Computational Intelligence,* pp. 2621-2628, 2008.

[76] S. I. Valdez1, A. Hernández, S. Botello. Efficient Estimation of Distribution Algorithms by Using the Empirical Selection Distribution. *New Achievements in Evolutionary Computation*, ISBN 978-953-307-053-7, Chapter 11, pp. 229-250, 2010.

[77] R. Santana, A. Mendiburu, J. A. Lozano. Customized Selection in Estimation of Distribution Algorithms. *Simulated Evolution and Learning*, pp. 94-105, 2014.

[78] Y. H. Shi, R. Eberhart. A Modified Particle Swarm Optimizer. *IEEE World Congress on Computational Intelligence*, pp. 69-73, 1998.

[79] S. Das, P. N. Suganthan. Differential Evolution: A Survey of the State-of-the-art. IEEE *Transactions on Evolution Computation*, 15(1): 4-31, 2011.

[80] T. Bäck, H. P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1): 1-23, 1993.

[81] H. P. Schwefel. Collective phenomena in evolutionary systems. In: *Preprints of the* 31st *Annual Meeting of the International Society for General System Research*, vol.2, pp.1025-1032, 1987.

[82] S. Baluja, R. Caruana. Removing the Genetics from the Standard Genetic Algorithm. *Tech. report CMU-CS-95-141*, *Computer Science Department*, *Carnegie Mellon University*, 1995.

[83] X. Yao, Y. Liu, G. M. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolution Computation*, 3(2): 82-102, 1999.

[84] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-Adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolution Computation*, 10(6): 646-657, 2006.

[85] Y. Shi, H. C. Liu, L. Gao, G. H. Zhang. Cellular particle swarm optimization. *Information Sciences*, 181(20): 4460-4493, 2011.

[86] H. Wang, S. Rahnamayan, H. Sun, G. H. O. Mahamed. Gaussian bare-Bones differential evolution. *IEEE Transactions on Cybernetics*, 43(2): 634-647, 2013.

[87] J. A. Rezaee. Enhanced Leader PSO (ELPSO): a new PSO variant for solving global optimization problems. *Applied Soft Computing*, vol.26, pp. 401-417, 2015.

[88] D. H. Wolpert, W. G. Macready. No free lunch theorems for search. *Santa Fe Institute, Tech. Rep.* SFI-TR-05-010, 1995.

[89] D. H. Wolpert, W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolution Computation*, 1(1): 67-82, 1997.

[90] Y. C. Ho, D. L. PEPYNE, M. A. Simaan. Simple Explanation of the No-Free-Lunch Theorem and Its Implications. *Journal of Optimization on Theory and Applications*, 115(3): 549-570, 2002.

[91] J. Slezak, T. Gotthans. Design of passive analog electronic circuits using hybrid modified UMDA algorithm. *Radioengineering*, 24(1): 161-170, 2015.

[92] M. H. Chen, P. C. Chang, J. L. Wu. A population-based incremental learning approach with artificial immune system for network intrusion detection. *Engineering Applications of Artificial Intelligence*, vol.51, pp. 171-181, 2016.

[93] X. C. Hao, M. Gen, L. Lin, G. A. Suer. Effective multi-objective EDA for bi-criteria stochastic job-shop scheduling problem. *Journal of Intelligent Manufacturing*, vol.28, pp. 833–845, 2017.

[94] L. T. Wang and C. C. Chen, A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4): 489-499, 1991.

[95] B. Balaguer, S. Carpin. Kinematics and Calibration for a Robot Comprised of Two Barrett WAMs and a Point Grey Bumblebee2 Stereo Camera, School of Engineering Technical Report. (University of California, Merced, CA, USA, 2012)

[96] H. G. Beyer and H. P. Schwefel. Evolution strategies – A comprehensive introduction. *Natural Computing*, 1(1): 3-52, 2002.

[97] Y. Z. Zhou, X. Y. Li, L. Gao. A differential evolution algorithm with intersect mutation operator. *Applied Soft Computing*, vol.13, pp. 390–401, 2013.

[98] R. Köker. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Information Sciences*, vol.222, pp. 528-543, 2013.

[99] A. W. Mohamed, H. Z. Sabry. Constrained optimization based on modified differential evolution algorithm. *Information Sciences*, vol.194, pp.171-208, 2012.

[100] T. P. Runarsson, X. Yao. Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolution Computation, 4(3): 284-294, 2000.

[101] C. H. Lin. A rough penalty genetic algorithm for constrained optimization. Information Sciences, vol.241, pp. 119-137, 2013.

[102] T. Ray, K. M. Liew. Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolution Computation*, 7(4): 386-396, 2003.

[103] M. Zhang, W. J. Luo, X. F. Wang. Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, 178(15): 3043–3074, 2008.

[104] A. Baykasoğlu, F. B. Ozsoydan. Adaptive firefly algorithm with chaos for mechanical design optimization Problems. *Applied Soft Computing*, vol.36, pp. 152-164, 2015.

[105] D. Geiger, D. Heckerman. Learning Gaussian Networks. In: *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, USA*, pp. 235-243, 1994.

[106] T. Okabe, Y. C. Jin, B. Sendhoff, M. Olhofe. Voronoi-based Estimation of Distribution Algorithm for Multi-objective Optimization. In: *Proceedings of the 2004 Congress on Evolutionary Computation* (*IEEE* Cat. No.04TH8753), pp. 1594-1601, 2004.

[107] Q. F. Zhang, A. Zhou, Y. C. Jin. RM-MEDA: A Regularity Model-Based Multi-objective Estimation of Distribution Algorithm. *IEEE Transactions on Evolution Computation*, 12(1): 41-63, 2008.

[108] V. A. Shim, K. C. Tan, J. Y. Chia, A. A. Mamun. Multi-Objective Optimization with Estimation of Distribution Algorithm in a Noisy Environment. Evolutionary Computation 21(1): 149-177, 2013.

[109] E. Zitzler, K. Deb, L. Thiele. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2): 173-195, 2000.

Appendix

This appendix presents the expressions of Δo and Δp for a numerical example of the 7-DOF Barrett WAM arm. Here, $c_i=\cos(\theta_i)$, $s_i=\sin(\theta_i)$, $i=1, 2, \cdots, 7$.

$$
\begin{aligned}
&\Delta o(\theta_1,\theta_2,\cdots,\theta_7) = (0.55c_1s_2 - 0.045s_1s_3 + 0.045c_3(s_1s_3 - c_1c_2c_3) - 0.3s_4(s_1s_3 - c_1c_2c_3) - 0.06s_6(c_5(c_4(s_1s_3 - c_1c_2c_3)\\
&+ c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3))) - 0.06c_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) + 0.3c_1c_4s_2 + 0.045c_1s_2s_3 + 0.045c_1c_2c_3 - 0.2401)^2\\
&+(0.7788c_7(s_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) - c_5(c_1c_3 - c_2s_1s_3)) + 0.4040c_7(s_5(c_4(s_1s_3 - c_1c_2c_3) - c_1s_2s_4) - c_5(c_3s_1 + c_1c_2s_3))\\
&+ 0.4799s_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)) + 0.4799c_7(s_5(c_2s_4 + c_3c_4s_2) + c_5s_2s_3) + 0.404s_7(c_6(c_5\\
&(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3)) - s_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2)) + 0.7788s_7(c_6(c_5(c_4(c_1s_3 + c2c_3s_1) - s_1s_2s_4)\\
&+ s_5(c_1c_3 - c_2s_1s_3)) - s_6(s_4(c_1s_3 + c_2c_3s_1) + c_4s_1s_2)) + 1.0)^2 + (0.4150s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3)\\
&-0.8771s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + 0.4150c_6(s_4(c_1s_3 + c_2c_3s_1) + c_4s_1s_2) + 0.8771c_6(c_2c_4 - c_3s_2s_4) + 0.2419s_6\\
&(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3)) + 0.2419c_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) - 1.0)^2 + (0.4704s_7(s_5(c_4(c_1s_3\\
&+ c_2c_3s_1) - s_1s_2s_4) - c_5(c_1c_3 - c_2s_1s_3)) - 0.8822s_7(s_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) - c_5(c_3s_1 + c_1c_2s_3)) + 0.0207c_7(c_6(c_5(c_2s_4\\
&+ c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)) + 0.8822c_7(c_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3)) - s_6(s_4(s_1s_3 - c_1c_2c_3)\\
&- c_1c_4s_2)) - 0.0207s_7(s_5(c_2s_4 + c_3c_4s_2) + c_5s_2s_2s_3) - 0.4704c_7(c_6(c_5(c_4(c_1c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))\\
&- s_6(s_4(c_1s_3 + c_2c_2c_3s_1) + c_4s_1s_2)) + 1.0)^2 + (0.045c_1s_3 + 0.55s_1s_2 + 0.06s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2c_2s_1s_3))\\
&+ c_6(s_4(c_1s_3 + c_2c_3s_1) + c_4s_1s_2) - 0.045c_3(c_1c_1s_3 + c_2c_2c_3s_1) + 0.3s_4(c_1s_3 + c_2c_3c_3s_1) + 0.045c_2c_3s_1 + 0.3c_4s_1s_2 + 0.045s_1s_2s_3\\
&- 0.2486)^2 + (0.55c_2 + 0.3c_2c_4 + 0.045c_2s_3 - 0.045c_3s_2 - 0.06s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + 0.06c_6(c_2c_4 - c_3s_2s_4)\\
&+ 0.045c_3^2s_2 - 0.3c_3s_2s_4 - 0.8382)^2
\end{aligned}
$$

$$
\begin{aligned}
&\Delta p(\theta_1,\theta_2,\cdots,\theta_7) = (0.55c_1s_2 - 0.045s_1s_3 + 0.045c_3(s_1s_3 - c_1c_2c_3) - 0.3s_4(s_1s_3 - c_1c_2c_3) - 0.06s_6(c_5(c_4(s_1s_3 - c_1c_2c_3)\\
&+ c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3)) - 0.06c_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) + 0.3c_1c_4s_2 + 0.045c_1s_2s_3 + 0.045c_1c_2c_3 - 0.2401)^2\\
&+ (0.045c_1s_3 + 0.55s_1s_2 + 0.06s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3)) + 0.06c_6(s_4(c_1s_3 + c_2c_3s_1) + c_4s_1s_2)\\
&- 0.045c_3(c_1s_3 + c_2c_3s_1) + 0.3s_4(c_1s_3 + c_2c_3s_1) + 0.045c_2c_3s_1 + 0.3c_4s_1s_2 + 0.045s_1s_2s_3 - 0.2486)^2 + (0.55c_2 + 0.3c_2c_4\\
&+ 0.045c_2s_3 - 0.045c_3s_2 - 0.06s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + 0.06c_6(c_2c_4 - c_3s_2s_4) + 0.045c_3^2s_2 - 0.3c_3s_2s_4 - 0.8382)^2
\end{aligned}
$$