Ye Tian
School of Computer Science and Technology, Anhui University, Hefei, CHINA

Ran Cheng
School of Computer Science, University of Birmingham, Birmingham, U.K.

Xingyi Zhang
School of Computer Science and Technology, Anhui University, Hefei, CHINA

Yaochu Jin
Department of Computer Science, University of Surrey, Guildford, U.K

PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization

Abstract

Over the last three decades, a large number of evolutionary algorithms have been developed for solving multi-objective optimization problems. However, there lacks an up-to-date and comprehensive software platform for researchers to properly benchmark existing algorithms and for practitioners to apply selected algorithms to solve their real-world problems. The demand of such a common tool becomes even more urgent, when the source code of many proposed algorithms has not been made publicly available. To address these issues, we have developed a MATLAB platform for evolutionary multi-objective optimization in this paper, called PlatEMO, which includes more than 50 multi-objective evolutionary algorithms and more than 100 multi-objective test problems, along with several widely used performance indicators. With a user-friendly graphical user interface, PlatEMO enables users to easily compare several evolutionary algorithms at one time and collect statistical results in Excel or LaTeX files. More importantly, PlatEMO is completely open source, such that users are able to develop new algorithms on the basis of it. This paper introduces the main features of PlatEMO and illustrates how to use it for performing comparative experiments, embedding new algorithms, creating new test problems, and developing performance indicators. Source code of PlatEMO is now available at: http://bimk.ahu.edu.cn/index.php?s=/Index/Software/index.html.

Digital Object Identifier 10.1109/MCI.2017.2742868
Date of publication: 12 October 2017

I. Introduction

Multi-objective optimization problems (MOPs) widely exist in computer science such as data mining [1], pattern recognition [2], image processing [3] and neural network [4], as well as many other application fields [5]–[8]. An MOP consists of two or more conflicting objectives to be optimized, and there often exist a set of optimal solutions trading off between different objectives. Since the vector evaluated genetic algorithm (VEGA) was proposed by Schaffer in 1985 [9], a number of multi-objective evolutionary algorithms (MOEAs) have been proposed and shown their superiority in tackling MOPs during the last three decades. For example, several MOEAs based on Pareto ranking selection and fitness sharing mechanism including multi-objective genetic algorithm (MOGA) [10], non-dominated sorting genetic algorithm (NSGA) [11], and niched Pareto genetic algorithm (NPGA) [12] were proposed in the 1990s. From 1999 to 2002, some MOEAs characterized by the elitism strategy were developed, such as non-dominated sorting genetic algorithm II (NSGA-II) [13],

Corresponding Author: Xingyi Zhang (E-mail: xyzhanghust@gmail.com)

strength Pareto evolutionary algorithm 2 (SPEA2) [14], Pareto envelope-based selection algorithm II (PESA-II) [15] and cellular multiobjective genetic algorithm (cellular MOGA) [16]. Afterwards, the evolutionary algorithm based on decomposition (MOEA/D) was proposed in 2007 [17], and some other MOEAs following the basic idea of MOEA/D had also been developed since then [18]–[21].

In spite of a large number of MOEAs in the literature [22], there often exist some difficulties in applying and using these algorithms since the source code of most algorithms had not been provided by the authors. Besides, it is also difficult to make benchmark comparisons between MOEAs due to the lack of a general experimental environment. To address such issues, several MOEA libraries have been proposed to provide uniform experimental environments for users [23]–[30], which have greatly advanced the multi-objective optimization research and the implementation of new ideas. For example, the C-based multi-objective optimization library PISA [27] separates the implementation into two components, i.e., the problem-specific part containing MOPs and operators, and the problem-independent part containing MOEAs. These two components are connected by a text file-based interface in PISA. jMetal [23] is an object-oriented Java-based multi-objective optimization library consisting of various MOEAs and MOPs. MOEA Framework is another free and open source Java framework for multi-objective optimization, which provides a comprehensive collection of MOEAs and tools necessary to rapidly design, develop, execute and test MOEAs. OTL [25], a C++ template library for multi-objective optimization, is characterized by object-oriented architecture, template technique, ready-to-use modules, automatically performed batch experiments and parallel computing. Besides, a Python-based experimental platform has also been proposed as the supplement of OTL, for improving the development efficiency and performing batch experiments more conveniently. ParadisEO-MOEO [26] is also a notable MOEA library designed on the basis of reconfigurable components. It provides a wide range of archive-related features and fitness assignment strategies used in the most common Pareto-based evolutionary algorithms, such that users can use the library to generate a large number of new MOEAs by recombining these components. AutoMOEA [30] is another recently proposed MOEA template extended from ParadisEO-MOEO, which has a higher generality and more comprehensive coverage of algorithms and operators. In addition, a similar framework was also adopted in ParadisEO-MO for the design of single solution-based metaheuristics [31].

It is encouraging that there are several MOEA libraries dedicated to the development of evolutionary multi-objective optimization (EMO), but unfortunately, most of them are still far from useful and practical to most researchers. Besides, due to the lack of professional GUI for experimental settings and algorithmic configurations, these libraries are difficult to be used or extended, especially for beginners who are not familiar with EMO. In order to collect more modern MOEAs and make the implementation of experiments on MOEAs easier, in this paper, we introduce a MATLAB-based EMO platform called PlatEMO. Compared to existing EMO platforms, PlatEMO has the following main advantages:

❏ *Rich Library.* PlatEMO now includes 50 existing popular MOEAs published in important journals or conferences in evolutionary computation community as shown in Table 1, which cover a variety of different types, including multi-objective genetic algorithms, multi-objective differential evolution algorithms, multi-objective particle swarm optimization algorithms, multi-objective estimation of distribution algorithms, and surrogate-assisted multi-objective evolutionary algorithms. PlatEMO also contains 110 MOPs from 16 popular test suites covering various difficulties, which are listed in Table 2. In addition, there are a lot of performance indicators provided by PlatEMO for experimental studies, including Coverage [75], generational distance (GD) [89], hypervolume (HV) [90], inverted generational distance (IGD) [91], normalized hypervolume (NHV) [41], pure diversity (PD) [92], spacing [93], spread (Δ) [94], and the performance metric ensemble [95]. PlatEMO also provides a lot of widely-used operators for different encodings [96]–[102], which can be used together with all the MOEAs in PlatEMO.

❏ *Good Usability.* PlatEMO is fully developed in MATLAB language, thus any machines installed with MATLAB can use PlatEMO regardless of the operating system. Besides, users do not need to write any additional code when performing experiments using MOEAs in PlatEMO, as PlatEMO provides a user-friendly GUI, where users can configure all the settings and perform experiments on MOEAs via the GUI, and the experimental results can further be saved as a table in the format of Excel or LaTeX. In other words, with the assistance of PlatEMO, users can directly obtain the statistical experimental results to be used in academic writings by *one-click* operation via the GUI.

❏ *Easy Extensibility.* PlatEMO is not only easy to be used, but also easy to be extended. To be specific, the source code of all the MOEAs, MOPs and operators in PlatEMO are completely open source, and the length of the source code is very short due to the advantages of matrix operation in MATLAB, such that users can easily implement their own MOEAs, MOPs and operators on the basis of existing resources in PlatEMO. In addition, all new MOEAs developed on the basis of interfaces provided by PlatEMO can be also included in the platform, such that the library in PlatEMO can be iteratively updated by all users to follow state-of-the-arts.

❏ *Delicate Considerations.* There are many delicate considerations in the

TABLE 1 The 50 Multi-Objective Optimization Algorithms Included in the Current Version of PlatEMO.

ALGORITHM	YEAR OF PUBLICATION	DESCRIPTION
MULTI-OBJECTIVE GENETIC ALGORITHMS		
SPEA2 [14]	2001	STRENGTH PARETO EVOLUTIONARY ALGORITHM 2
PSEA-II [15]	2001	PARETO ENVELOPE-BASED SELECTION ALGORITHM II
NSGA-II [13]	2002	NON-DOMINATED SORTING GENETIC ALGORITHM II
ϵ-MOEA [32]	2003	MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON ϵ-DOMINANCE
IBEA [33]	2004	INDICATOR-BASED EVOLUTIONARY ALGORITHM
MOEA/D [17]	2007	MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON DECOMPOSITION
SMS-EMOA [34]	2007	S METRIC SELECTION EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION ALGORITHM
MSOPS-II [35]	2007	MULTIPLE SINGLE OBJECTIVE PARETO SAMPLING ALGORITHM II
MTS [36]	2009	MULTIPLE TRAJECTORY SEARCH
AGE-II [37]	2013	APPROXIMATION-GUIDED EVOLUTIONARY ALGORITHM II
NSLS [38]	2015	NON-DOMINATED SORTING AND LOCAL SEARCH
BCE-IBEA [39]	2015	BI-CRITERION EVOLUTION FOR IBEA
MOEA/IGD-NS [40]	2016	MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON AN ENHANCED INVERTED GENERATIONAL DISTANCE METRIC
MANY-OBJECTIVE GENETIC ALGORITHMS		
HypE [41]	2011	HYPERVOLUME-BASED ESTIMATION ALGORITHM
PICEA-g [42]	2013	PREFERENCE-INSPIRED COEVOLUTIONARY ALGORITHM WITH GOALS
GrEA [43]	2013	GRID-BASED EVOLUTIONARY ALGORITHM
NSGA-III [44]	2014	NON-DOMINATED SORTING GENETIC ALGORITHM III
A-NSGA-III [45]	2014	ADAPTIVE NSGA-III
SPEA2+SDE [46]	2014	SPEA2 WITH SHIFT-BASED DENSITY ESTIMATION
BiGE [47]	2015	BI-GOAL EVOLUTION
EFR-RR [20]	2015	ENSEMBLE FITNESS RANKING WITH RANKING RESTRICTION
I-DBEA [48]	2015	IMPROVED DECOMPOSITION BASED EVOLUTIONARY ALGORITHM
KnEA [49]	2015	KNEE POINT DRIVEN EVOLUTIONARY ALGORITHM
MaOEA-DDFC [50]	2015	MANY-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON DIRECTIONAL DIVERSITY AND FAVORABLE CONVERGENCE
MOEA/DD [51]	2015	MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON DOMINANCE AND DECOMPOSITION
MOMBI-II [52]	2015	MANY-OBJECTIVE METAHEURISTIC BASED ON THE R2 INDICATOR II
Two_Arch2 [53]	2015	TWO-ARCHIVE ALGORITHM 2
MaOEA-R&D [54]	2016	MANY-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON OBJECTIVE SPACE REDUCTION AND DIVERSITY IMPROVEMENT
RPEA [55]	2016	REFERENCE POINTS-BASED EVOLUTIONARY ALGORITHM
RVEA [56]	2016	REFERENCE VECTOR GUIDED EVOLUTIONARY ALGORITHM
RVEA* [56]	2016	RVEA EMBEDDED WITH THE REFERENCE VECTOR REGENERATION STRATEGY
SPEA/R [57]	2016	STRENGTH PARETO EVOLUTIONARY ALGORITHM BASED ON REFERENCE DIRECTION
θ-DEA [58]	2016	θ-DOMINANCE BASED EVOLUTIONARY ALGORITHM
MULTI-OBJECTIVE GENETIC ALGORITHMS FOR LARGE-SCALE OPTIMIZATION		
MOEA/DVA [59]	2016	MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON DECISION VARIABLE ANALYSES
LMEA [60]	2016	LARGE-SCALE MANY-OBJECTIVE EVOLUTIONARY ALGORITHM
MULTI-OBJECTIVE GENETIC ALGORITHMS WITH PREFERENCE		
g-NSGA-II [61]	2009	g-DOMINANCE BASED NSGA-II
r-NSGA-II [62]	2010	r-DOMINANCE BASED NSGA-II

(Continued)

TABLE 1 The 50 Multi-Objective Optimization Algorithms Included in the Current Version of PlatEMO. *(Continued)*

ALGORITHM	YEAR OF PUBLICATION	DESCRIPTION
WV-MOEA-P [63]	2016	WEIGHT VECTOR BASED MULTI-OBJECTIVE OPTIMIZATION ALGORITHM WITH PREFERENCE
MULTI-OBJECTIVE DIFFERENTIAL ALGORITHMS		
GDE3 [64]	2005	GENERALIZED DIFFERENTIAL EVOLUTION 3
MOEA/D-DE [18]	2009	MOEA/D BASED ON DIFFERENTIAL EVOLUTION
MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION ALGORITHMS		
MOPSO [65]	2002	MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION
SMPSO [66]	2009	SPEED-CONSTRAINED MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION
dMOPSO [67]	2011	DECOMPOSITION-BASED PARTICLE SWARM OPTIMIZATION
MULTI-OBJECTIVE MEMETIC ALGORITHMS		
M-PAES [68]	2000	MEMETIC ALGORITHM BASED ON PARETO ARCHIVED EVOLUTION STRATEGY
MULTI-OBJECTIVE ESTIMATION OF DISTRIBUTION ALGORITHMS		
MO-CMA [69]	2007	MULTI-OBJECTIVE COVARIANCE MATRIX ADAPTATION
RM-MEDA [70]	2008	REGULARITY MODEL-BASED MULTI-OBJECTIVE ESTIMATION OF DISTRIBUTION ALGORITHM
IM-MOEA [71]	2015	INVERSE MODELING MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM
SURROGATE-ASSISTED MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS		
ParEGO [72]	2005	EFFICIENT GLOBAL OPTIMIZATION FOR PARETO OPTIMIZATION
SMS-EGO [73]	2008	S-METRIC-SELECTION-BASED EFFICIENT GLOBAL OPTIMIZATION
K-RVEA [74]	2016	KRIGING ASSISTED RVEA

TABLE 2 The 110 Multi-Objective Optimization Problems Included in the Current Version of PlatEMO.

PROBLEM	YEAR OF PUBLICATION	DESCRIPTION
MOKP [75]	1999	MULTI-OBJECTIVE 0/1 KNAPSACK PROBLEM AND BEHAVIOR OF MOEAS ON THIS PROBLEM ANALYZED IN [76]
ZDT1–ZDT6 [77]	2000	MULTI-OBJECTIVE TEST PROBLEMS
mQAP [78]	2003	MULTI-OBJECTIVE QUADRATIC ASSIGNMENT PROBLEM
DTLZ1–DTLZ9 [79]	2005	SCALABLE MULTI-OBJECTIVE TEST PROBLEMS
WFG1–WFG9 [80]	2006	SCALABLE MULTI-OBJECTIVE TEST PROBLEMS AND DEGENERATE PROBLEM WFG3 ANALYZED IN [81]
MONRP [82]	2007	MULTI-OBJECTIVE NEXT RELEASE PROBLEM
MOTSP [83]	2007	MULTI-OBJECTIVE TRAVELING SALESPERSON PROBLEM
Pareto-Box [84]	2007	PARETO-BOX PROBLEM
CF1–CF10 [85]	2008	CONSTRAINED MULTI-OBJECTIVE TEST PROBLEMS FOR THE CEC 2009 SPECIAL SESSION AND COMPETITION
F1–F10 FOR RM-MEDA [70]	2008	THE TEST PROBLEMS DESIGNED FOR RM-MEDA
UF1–UF12 [85]	2008	UNCONSTRAINED MULTI-OBJECTIVE TEST PROBLEMS FOR THE CEC 2009 SPECIAL SESSION AND COMPETITION
F1–F9 FOR MOEA/D-DE [18]	2009	THE TEST PROBLEMS EXTENDED FROM [86] DESIGNED FOR MOEA/D-DE
C1_DTLZ1, C2_DTLL2, C3_DTLZ4 IDTLZ1, IDTLZ2 [45]	2014	CONSTRAINED DTLZ AND INVERTED DTLZ
F1–F7 FOR MOEA/D-M2M [19]	2014	THE TEST PROBLEMS DESIGNED FOR MOEA/D-M2M
F1–F10 FOR IM-MOEA [71]	2015	THE TEST PROBLEMS DESIGNED FOR IM-MOEA
BT1–BT9 [87]	2016	MULTI-OBJECTIVE TEST PROBLEMS WITH BIAS
LSMOP1–LSMOP9 [88]	2016	LARGE-SCALE MULTI-OBJECTIVE TEST PROBLEMS

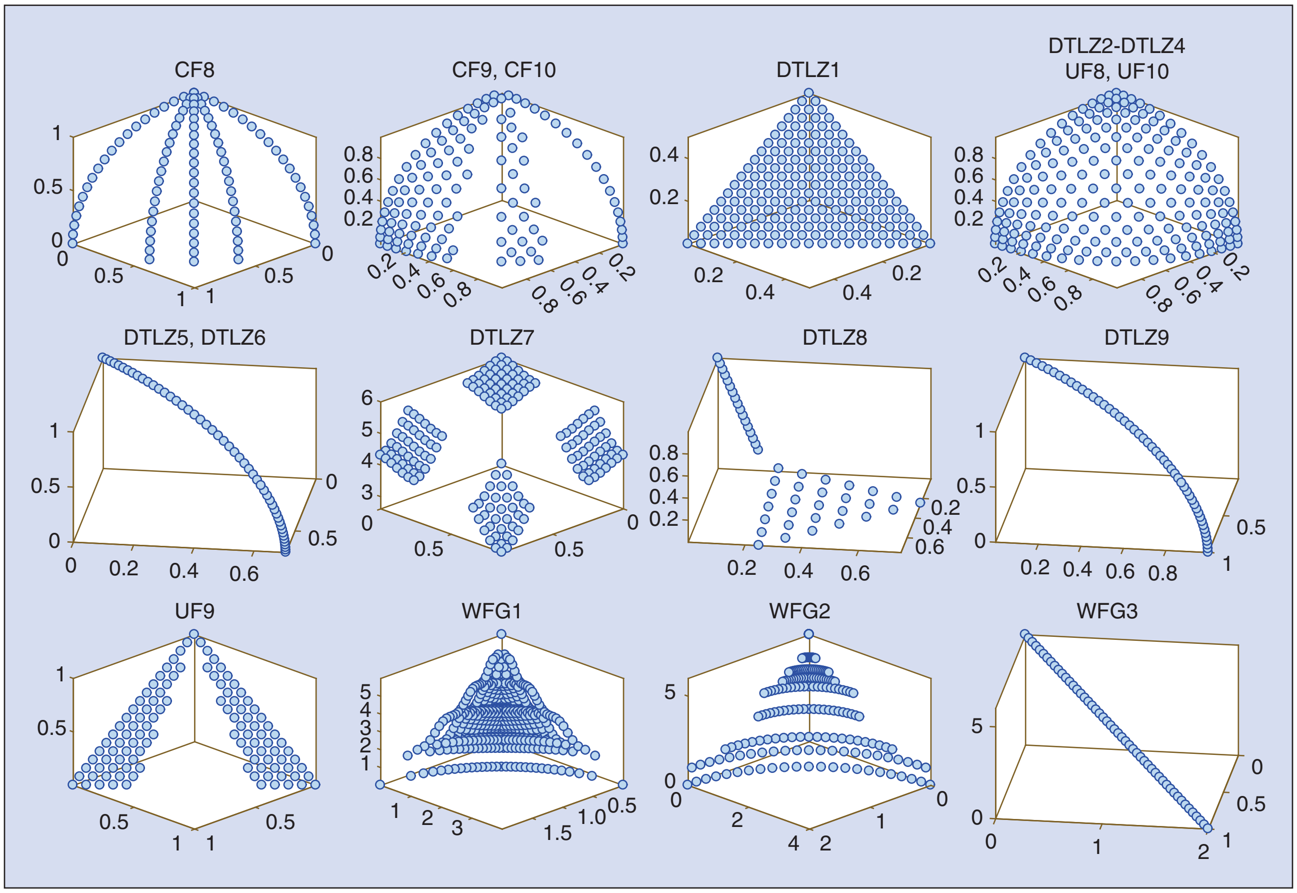


FIGURE 1 The reference points generated by PlatEMO on the Pareto fronts of CF8–10, DTLZ1–9, UF8–10 and WFG1–3 with 3 objectives. Note that the reference points do not cover the whole true Pareto front of WFG3, since the geometry of the true Pareto front is still unknown [81].

implementation of PlatEMO. For example, PlatEMO provides different figure demonstrations of experimental results, and it also provides well-designed sampling methods which are able to generate an arbitrary number of uniformly distributed points on the Pareto optimal fronts with an arbitrary number of objectives. Fig. 1 shows the reference points sampled by PlatEMO on the Pareto optimal fronts of some MOPs with 3 objectives, while such reference points have not been provided by any other existing EMO libraries. It is also worth noting that, since the efficiency of most MOEAs is subject to the non-dominated sorting process, PlatEMO employs the efficient non-dominated sort ENS-SS [103] for two- and three-objective optimization and the tree-based ENS termed T-ENS [60] for optimization with more than three objectives as the non-dominated sorting approaches, which have been demonstrated to be much more efficient than the widely adopted fast non-dominated sorting approach [13] as well as other popular non-dominated sorting approaches [104].

Table 3 lists the main characteristics of PlatEMO in comparison with other existing MOEA libraries. As shown in the table, PlatEMO provides a variety of MOEAs and MOPs with different types, where the types of MOEAs cover genetic algorithm, differential algorithm, particle swarm optimization, memetic algorithm, estimation of distribution algorithm, surrogate-assisted evolutionary algorithm and the types of MOPs have multi-objective, many-objective, combinatorial, large-scale and expensive optimization problem. The PlatEMO shows significant superiority in usability of MOEAs and extendibility of adding new MOEAs, while ParadisEO-MOEO has the best configurability in terms of generating new MOEAs by combining different components.

The rest of this paper is organized as follows. In the next section, the architecture of PlatEMO is presented on several aspects, i.e., the file structure of PlatEMO, the class diagram of PlatEMO, and the sequence diagram of executing algorithms by PlatEMO. Section III introduces how to use PlatEMO for analyzing the performance of algorithms and performing comparative experiments. The methods of extending PlatEMO with new MOEAs, MOPs, operators and performance indicators are described in Section IV. Finally, conclusion and future work are given in Section V.

II. Architecture of PlatEMO

After opening the root directory of PlatEMO, users can see a lot of *.m* files organized in the structure shown in

TABLE 3 Comparison Between PlatEMO and Five Existing MOEA Libraries.

MOEA LIBRARY	LANGUAGE	TYPES OF MOEAS AVAILABLE	TYPES OF MOPS AVAILABLE	USABILITY	COMPONENTS CONFIGURABILITY	EXTENDIBILITY
PARADISEO-MOEO [26]	C++	GENETIC ALGORITHM, SIMULATED ANNEALING, TABU SEARCH	MULTI-OBJECTIVE, COMBINATORIAL	NORMAL	HIGH	NORMAL
PISA [27]	C	GENETIC ALGORITHM	MULTI-OBJECTIVE, MANY-OBJECTIVE, COMBINATORIAL	NORMAL	LOW	NORMAL
jMETAL [23]	Java	GENETIC ALGORITHM, DIFFERENTIAL ALGORITHM, PARTICLE SWARM OPTIMIZATION	MULTI-OBJECTIVE, MANY-OBJECTIVE, COMBINATORIAL	NORMAL	NORMAL	NORMAL
OTL [25]	C++, Python	GENETIC ALGORITHM, DIFFERENTIAL ALGORITHM	MULTI-OBJECTIVE, MANY-OBJECTIVE, COMBINATORIAL	LOW	NORMAL	NORMAL
MOEA FRAMEWORK	Java	GENETIC ALGORITHM, DIFFERENTIAL ALGORITHM, PARTICLE SWARM OPTIMIZATION	MULTI-OBJECTIVE, MANY-OBJECTIVE, COMBINATORIAL	NORMAL	NORMAL	NORMAL
PlatEMO	MATLAB	GENETIC ALGORITHM, DIFFERENTIAL ALGORITHM, PARTICLE SWARM OPTIMIZATION, MEMETIC ALGORITHM, ESTIMATION OF DISTRIBUTION ALGORITHM, SURROGATE-ASSISTED EVOLUTIONARY ALGORITHM	MULTI-OBJECTIVE, MANY-OBJECTIVE, COMBINATORIAL, LARGE-SCALE, EXPENSIVE	HIGH	NORMAL	HIGH

*ParadisEO-MOEO: http://paradiseo.gforge.inria.fr (last updated Nov. 4, 2012).
*PISA: http://www.tik.ee.ethz.ch/pisa (last updated Oct. 13, 2014).
*jMetal: http://jmetal.sourceforge.net/index.html (last updated Sep. 29, 2015).
*OTL: https://github.com/O-T-L/OTL (last updated Nov. 1, 2015).
*MOEA Framework: http://moeaframework.org/index.html (last updated Jan. 4, 2017).
*PlatEMO: http://bimk.ahu.edu.cn/index.php?s=/Index/Software/index.html (last updated Apr. 10, 2017).

Fig. 2, where it is very easy to find the source code of specified MOEAs, MOPs, operators or performance indicators. As shown in Fig. 2, there are six folders and one interface function *main.m* in the root directory of PlatEMO. The first folder *\Algorithms* is used to store all the MOEAs in PlatEMO, where each MOEA has an independent subfolder and all the relevant functions are in it. For instance, as shown in Fig. 2, the subfolder *\Algorithms\NSGA-II* contains three functions *NSGAII.m*, *CrowdingDistance.m* and *EnvironmentalSelection.m*, which are used to perform the main loop, calculate the crowding distances, and perform the environmental selection of NSGA-II, respectively. The second folder *\Problems* contains a lot of subfolders for storing benchmark MOPs. For example, the subfolder *\Problems*\DTLZ contains 15 DTLZ test problems (i.e., DTLZ1–DTLZ9, C1_DTLZ1, C2_DTLZ2, C3_DTLZ4, IDTLZ1, IDTLZ2 and CDTLZ2), and the subfolder *\Problems*WFG contains 9 WFG test problems (i.e., WFG1–WFG9). The folders *\Operators* and *\Metrics* store all the operators and performance indicators, respectively. The next folder *\Public* is used to store some public classes and functions, such as *GLOBAL.m* and *INDIVIDUAL.m*, which are two classes in PlatEMO representing settings of parameters and definitions of individuals, respectively. The last folder *\GUI* stores all the functions for establishing the GUI of PlatEMO, where users need not read or modify them.

PlatEMO also has a simple architecture, where it only involves two classes, namely *GLOBAL* and *INDIVIDUAL*, to store all the parameters and joint all the components (e.g., MOEAs, MOPs and operators). The class diagram of these two classes is presented in Fig. 3. The first class *GLOBAL* represents all the parameter setting, including the handle of MOEA function *algorithm*, the handle of MOP function *problem*, the handle of operator function *operator* and other parameters about the environment (the population size, the number of objectives, the length of decision variables, the maximum number of fitness evaluations, etc.). Note that all the properties in *GLOBAL* are read-only, which can only be assigned by users when the object is being instantiated. *GLOBAL* also provides several methods to be invoked by MOEAs, where MOEAs can achieve some complex operations

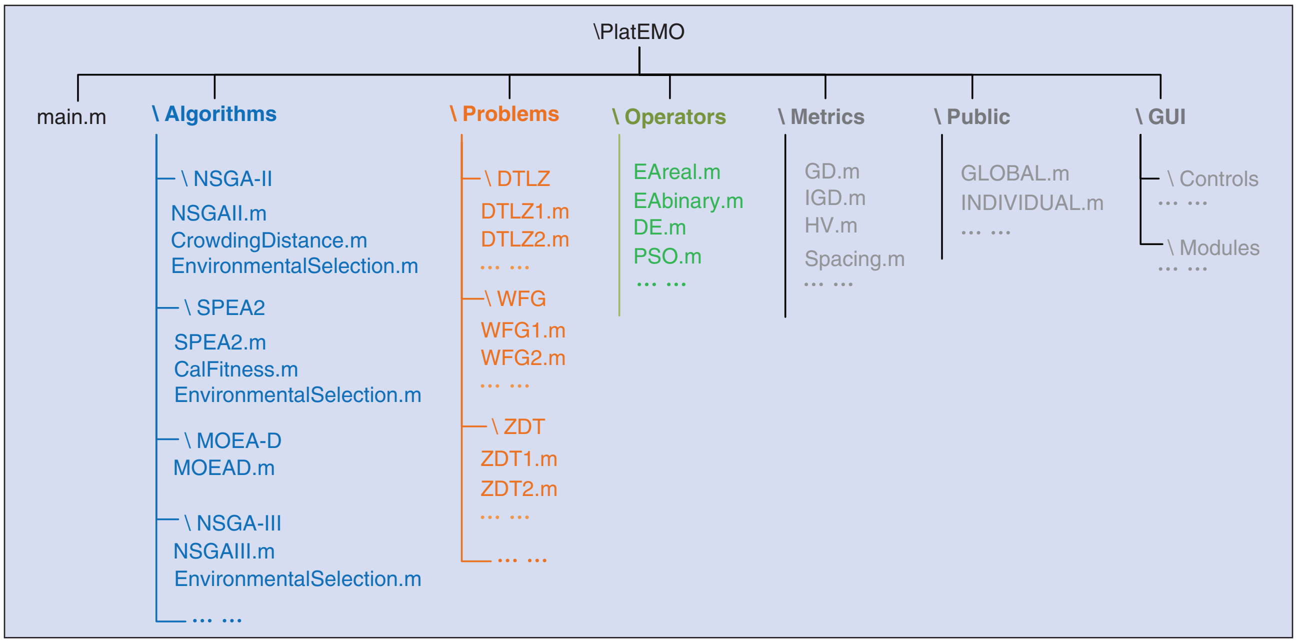


FIGURE 2 Basic file structure of PlatEMO.

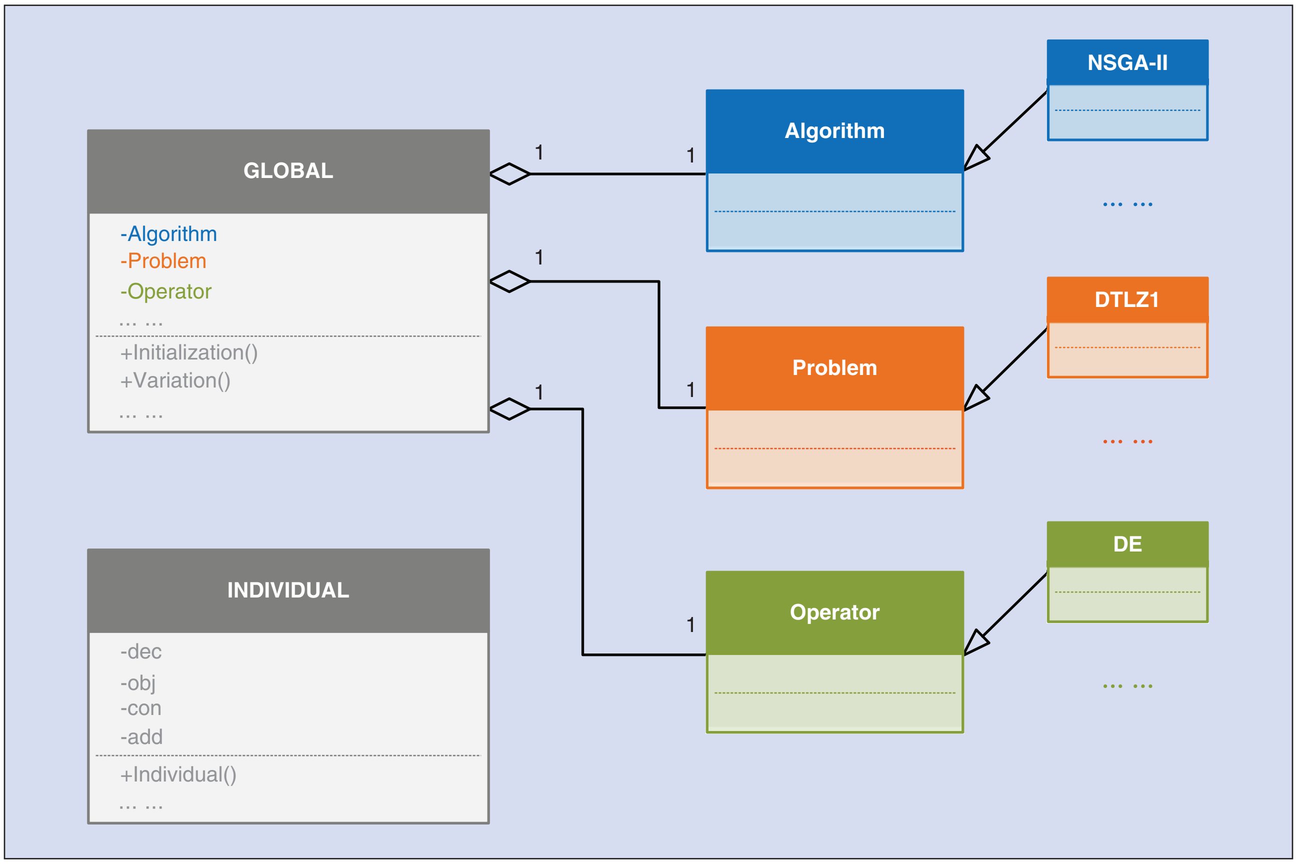


FIGURE 3 Class diagram of the architecture of PlatEMO.

via these methods. For instance, the method *Initialization*() can generate a randomly initial population with specified size, and the method *Variation*() can generate a set of offsprings with specified parents.

The other class in PlatEMO is *INDIVIDUAL*, where its objects are exactly individuals in MOEAs. An *INDIVIDUAL* object contains four properties, i.e., *dec*, *obj*, *con* and *add*, all of which are also read-only. *dec* is the array of decision variables of the individual, which is assigned when the object is being instantiated. *obj* and *con* store the objective values and the constraint values of the individual, respectively, which are calculated after *dec* has been assigned. The property *add* is used to store additional properties of the individual for some special operators, such as the 'speed' property in PSO operator [101].

In order to better understand the mechanism of PlatEMO, Fig. 4 illustrates the sequence diagram of running an MOEA by PlatEMO without GUI. To begin with, the interface *main.m* first invokes the algorithm function (e.g., *NSGAII.m*), then the algorithm obtains an initial population (i.e., an array of *INDIVIDUAL* objects) from the *GLOBAL* object by invoking its method *Initialization*(). After that, the algorithm starts the evolution until the termination criterion is fulfilled, where the maximum number of fitness evaluations is used as the termination criterion for all the MOEAs in PlatEMO. In each generation of a general MOEA, it first performs mating pool selection for selecting a number of parents from the current population, and the parents are used to generate offsprings by invoking the method *Variation*() of *GLOBAL* object. *Variation*() then passes the parents to the operator function (e.g., *DE.m*), which is used to calculate the decision variables of the offsprings according to the parents. Next, the operator function invokes the *INDIVIDUAL* class to instantiate the offspring objects, where the objective values of offsprings are calculated by invoking the problem function (e.g., *DTLZ1.m*). After obtaining the offsprings, the algorithm performs environmental selection on the current population and the offsprings to select the population for next generation. When the number of instantiated *INDIVIDUAL* objects exceeds the maximum number of fitness evaluations, the algorithm will be terminated and the final population will be output.

As presented by the above procedure, the algorithm function, the problem function and the operator function do

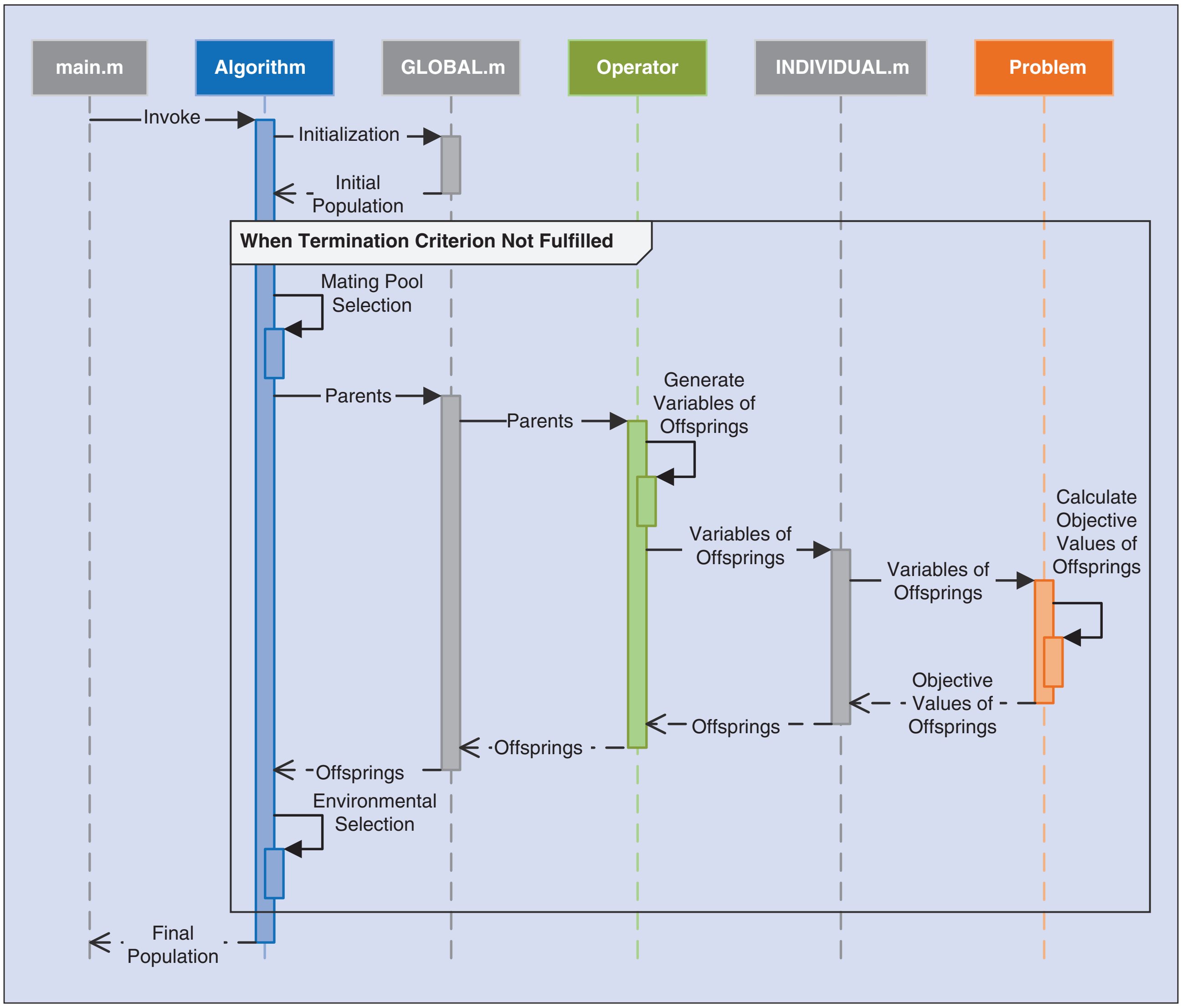


FIGURE 4 Sequence diagram of running a general multi-objective optimization algorithm by PlatEMO without GUI.

not invoke each other directly; instead, they are connected to each other by the *GLOBAL* class and the *INDIVIDUAL* class. This mechanism has two advantages. First, MOEAs, MOPs and operators in PlatEMO are independent mutually, and they can be arbitrarily combined with each other, thus improved the flexibility of PlatEMO. Second, users need not consider the details of the MOP or the operator to be involved when developing a new MOEA, thus significantly improving the development efficiency.

III. Running PlatEMO

As mentioned in Section I, PlatEMO provides two ways to run it: first, it can be run with a GUI by invoking the interface *main*() without input parameter, then users can perform MOEAs on MOPs by simple *one-click* operations; second, it can be run without GUI, and users can perform one MOEA on an MOP by invoking *main*() with input parameters. In this section, we elaborate these two ways of running PlatEMO.

A. Running PlatEMO without GUI

The interface *main*() can be invoked with a set of input parameters by the following form: *main*(*'name1', value1, 'name2', value2,...*), where *name1, name2,...* denote the names of the parameters and *value1, value2,...* denote the values of the parameters. All the acceptable parameters together with their data types and default values are listed in Table 4. It is noteworthy that every parameter has a default value so that users need not assign all the parameters. As an example, the command *main('-algorithm',@NSGAII,'-problem',@DTLZ2,'-N',100,'-M',3,'-D',10,'-evaluation',10000)* is used to perform NSGA-II on DTLZ2 with a population size of 100, an objective number of 3, a decision variable length of 10, and a maximum fitness evaluation number of 10000.

By invoking *main*() with parameters, one MOEA can be performed on an MOP with the specified setting, while the GUI will not be displayed. After the MOEA has been terminated, the final population will be displayed or saved, which is determined by the parameter *-mode* shown in Table 4. To be specific, if *-mode* is set to 1, the objective values or decision variable values of the final population will be displayed in a new figure, and users can also observe the true Pareto front and the evolutionary trajectories of performance indicator values. For example, Fig. 5 shows the objective values of the population obtained by NSGA-II on DTLZ2 with 3 objectives, where users can select the figure to be displayed on the rightmost menu. If *-mode* is set to 2, the final population will be saved in a *.mat* file, while no figure will be displayed. If *-mode* is set to 3, users can specify any operation to be performed on the final population, e.g., displaying and saving the population concurrently.

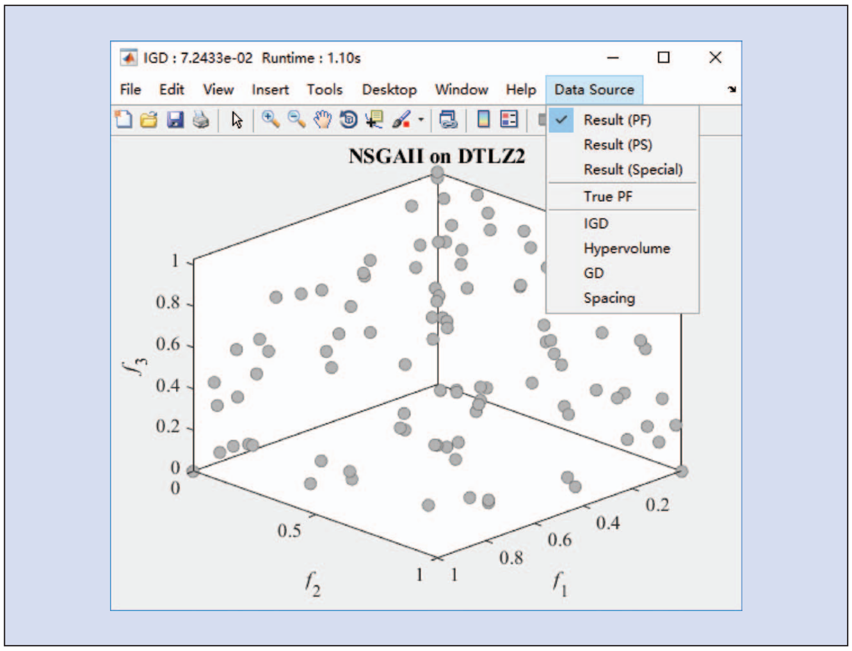


FIGURE 5 The objective values of the population obtained by NSGA-II on DTLZ2 with 3 objectives by running PlatEMO without GUI.

TABLE 4 All the Acceptable Parameters for the Interface of PlatEMO.

PARAMETER NAME	TYPE	DEFAULT VALUE	DESCRIPTION
-algorithm	FUNCTION HANDLE	@NSGAII	ALGORITHM FUNCTION
-problem	FUNCTION HANDLE	@DTLZ2	PROBLEM FUNCTION
-operator	FUNCTION HANDLE	@EAreal	OPERATOR FUNCTION
-N	POSITIVE INTEGER	100	POPULATION SIZE
-M	POSITIVE INTEGER	3	NUMBER OF OBJECTIVES
-D	POSITIVE INTEGER	12	NUMBER OF DECISION VARIABLES
-evaluation	POSITIVE INTEGER	10000	MAXIMUM NUMBER OF FITNESS EVALUATIONS
-run	POSITIVE INTEGER	1	RUN NO.
-mode	1, 2 OR 3	1	RUN MODE: 1. DISPLAY RESULT, 2. SAVE RESULT, 3. SPECIFIED BY USER
-outputFcn	FUNCTION HANDLE	-	SPECIFIED OPERATION ON THE RESULT WHEN *-mode* IS SET TO 3
-X_parameter	CELL	-	THE PARAMETER VALUES FOR FUNCTION X

Generally, as listed in Table 4, four parameters related to the optimization can be assigned by users (i.e., the population size *-N*, the number of objectives *-M*, the number of decision variables *-D*, and the maximum number of fitness evaluations *-evaluation*); however, different MOEAs, MOPs or operators may involve additional parameter settings. For instance, there is a parameter *rate* denoting the ratio of selected knee points in *KnEA* [49], and there are four parameters *proC*, *disC*, *proM* and *disM* in *EAreal* [96], [97], which denote the crossover probability, the distribution index of simulated binary crossover, the number of bits undergone mutation, and the distribution index of polynomial mutation, respectively. In PlatEMO, such function related parameters can also be assigned by users via assigning the parameter *-X_parameter*, where *X* indicates the name of the function. For example, users can use the command *main(…,'-KnEA_parameter',{0.5},…)* to set the value of *rate* to 0.5 for *KnEA*, and use the command *main(…,'-EAreal_parameter',{1,20,1,20},…)* to set the values of *proC*, *disC*, *proM* and *disM* to 1, 20, 1 and 20 for *EAreal*, respectively. Besides, users can find the acceptable parameters of each MOEA, MOP and operator in the comments at the beginning of the corresponding function.

B. Running PlatEMO with GUI

The GUI of PlatEMO currently contains two modules. The first module is used to analyze the performance of each MOEA, where one MOEA on an MOP can be performed in this module each time, and users can observe the result via different figure demonstrations. The second module is designed for statistical experiments, where multiple MOEAs on a batch of MOPs can be performed at the same time, and the statistical experimental results can be saved as Excel table or LaTeX table.

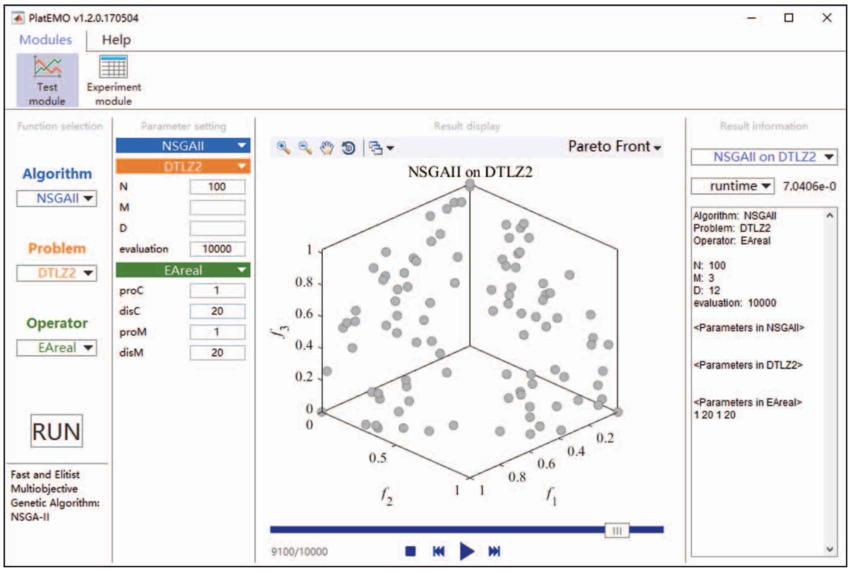
The interface of the first module, i.e., test module, is shown in Fig. 6. As can be seen from the figure, the main panel is divided into four parts. The first subpanel from left provides three pop up menus, where users can select the MOEA, MOP and operator to be performed. The second subpanel lists all the parameters to be assigned, which depends on the selected MOEA, MOP and operator. The third subpanel displays the current population during the optimization, other figures such as the true Pareto front and the evolutionary trajectories of performance indicator values can also be displayed. In addition, users can observe the populations in previous generations by dragging the slider at the bottom. The fourth subpanel stores the detailed information of historical results. As a result, the test module provides similar functions to the PlatEMO without GUI, but users do not need to write any additional command or code when using it.

The other module on the GUI is the experimental module, which is shown in Fig. 7. Similar to the test module, users should first select the MOEAs, MOPs and operators to be performed in the leftmost subpanel. Note that multiple MOEAs and MOPs can be selected in the experimental module. After setting the number of total runs, folder for saving results, and all the relevant parameters, the experiment can be started and the statistical results will be shown in the rightmost subpanel. Users can select any performance indicator to calculate the results to be listed in the table, where the mean and the standard deviation of the performance indicator value are shown in each grid. Furthermore, the best result in each row is shown in blue, and the Wilcoxon rank sum test result is labeled by the signs '+', '–' and '≈', which indicate that the result is significantly better, significantly worst and statistically similar to the result in the control column, respectively. After the experiment is finished, the data shown in the table can be saved as Excel table (*.xlsx* file) or LaTeX table (*.tex* file). For example, after obtaining the experimental results shown in the table in Fig. 7, users can press the 'saving' button on the GUI to save the table in the format of LaTeX, where the corresponding LaTeX table is shown in Table 5.

It can be concluded from the above introduction that the functions provided by PlatEMO are modularized, where two modules (i.e., the test module and the experimental module) are included in the current version of PlatEMO. In the future, we also plan to develop more modules to provide more functions for users.

IV. Extending PlatEMO

PlatEMO is an open platform for scientific research and applications of EMO, hence it allows users to add their own MOEAs, MOPs, operators and performance indicators to it, where users should save the new MOEA, MOP, operator or performance indicator to be



Algorithm: NSGAII

Problem: DTLZ2

Operator: EAreal

NSGAII	
DTLZ2	
N	100
M	
D	
evaluation	10000
EAreal	
proC	1
disC	20
proM	1
disM	20

Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II

NSGAII on DTLZ2

runtime 7.0406e-0

```
Algorithm: NSGAII
Problem: DTLZ2
Operator: EAreal

N: 100
M: 3
D: 12
evaluation: 10000

<Parameters in NSGAII>


<Parameters in DTLZ2>


<Parameters in EAreal>
1 20 1 20
```

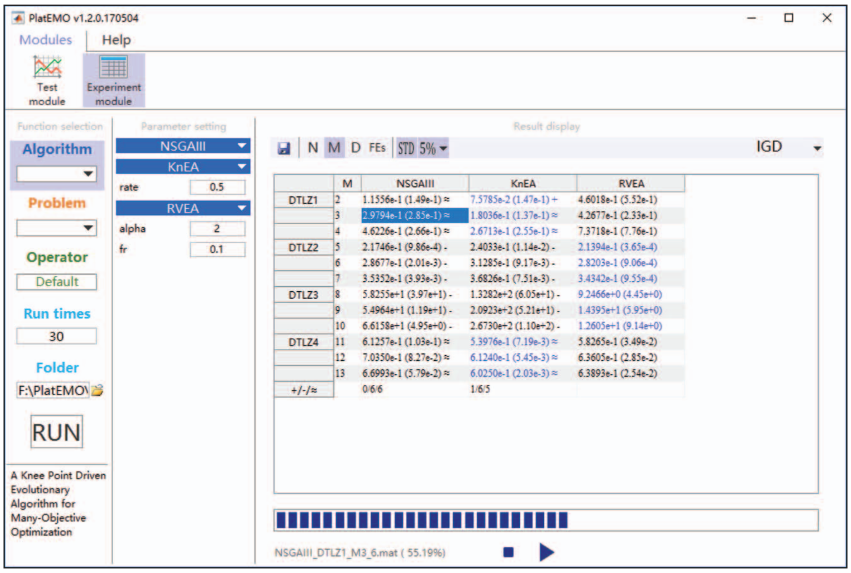
FIGURE 6 The test module of PlatEMO.

added as a MATLAB function (i.e., a *.m* file) with the specified interface and form, and put it in the corresponding folder. In the following, the methods of extending PlatEMO with a new MOEA, MOP, operator and performance indicator are illustrated by several cases, respectively.

A. Adding New Algorithms to PlatEMO

In order to add a new MOEA to PlatEMO, users only need to slightly modify the input and output of their MATLAB function of the algorithm as required by PlatEMO, and put the function (*.m* file) in the folder *\Algorithms* of the root directory. For example, as shown in the file structure in Fig. 2, the three *.m* files for NSGA-II (i.e., *NSGAII.m*, *CrowdingDistance.m* and *EnvironmentalSelection.m*) are all in the subfolder *\Algorithms\NSGA-II*. A case study including the source code of the main function of NSGA-II (i.e. *NSGAII.m*) is given in Fig. 8, where the logic of the function is completely the same to the one shown in Fig. 4.

To begin with, the main function of an MOEA has one input parameter and zero output parameter, where the only input parameter denotes the *GLOBAL* object for the current run. Then an initial population *Population* is generated by invoking *Global.Initialization*(), and the non-dominated front number and the crowding distance of each individual are



rate 0.5
alpha 2
fr 0.1

A Knee Point Driven Evolutionary Algorithm for Many-Objective Optimization

	M	NSGAIII	KnEA	RVEA
DTLZ1	2	1.1556e-1 (1.49e-1) ≈	7.5785e-2 (1.47e-1) +	4.6018e-1 (5.52e-1)
	3	2.9794e-1 (2.85e-1) ≈	1.8036e-1 (1.37e-1) ≈	4.2677e-1 (2.33e-1)
	4	4.6226e-1 (2.66e-1) ≈	2.6713e-1 (2.55e-1) ≈	7.3718e-1 (7.76e-1)
DTLZ2	5	2.1746e-1 (9.86e-4) -	2.4033e-1 (1.14e-2) -	2.1394e-1 (3.65e-4)
	6	2.8677e-1 (2.01e-3) -	3.1285e-1 (9.17e-3) -	2.8203e-1 (9.06e-4)
	7	3.5352e-1 (3.93e-3) -	3.6826e-1 (7.51e-3) -	3.4342e-1 (9.55e-4)
DTLZ3	8	5.8255e+1 (3.97e+1) -	1.3282e+2 (6.05e+1) -	9.2466e+0 (4.45e+0)
	9	5.4964e+1 (1.19e+1) -	2.0923e+2 (5.21e+1) -	1.4395e+1 (5.95e+0)
	10	6.6158e+1 (4.95e+0) -	2.6730e+2 (1.10e+2) -	1.2605e+1 (9.14e+0)
DTLZ4	11	6.1257e-1 (1.03e-1) ≈	5.3976e-1 (7.19e-3) ≈	5.8265e-1 (3.49e-2)
	12	7.0350e-1 (8.27e-2) ≈	6.1240e-1 (5.45e-3) ≈	6.3605e-1 (2.85e-2)
	13	6.6993e-1 (5.79e-2) ≈	6.0250e-1 (2.03e-3) ≈	6.3893e-1 (2.54e-2)
+/-/≈		0/6/6	1/6/5	

NSGAIII_DTLZ1_M3_6.mat (55.19%)

FIGURE 7 The experimental module of PlatEMO.

calculated (line 2–4). In each generation, the function *Global.NotTermination*() is invoked to check whether the termination criterion is fulfilled, and the variable *Population* is passed to this function to be the final output (line 5). Afterwards, the mating pool selection, generating offsprings, and environmental selection are performed in sequence (line 6–9).

The common code required by any MOEA is underlined in Fig. 8. In addition to the interface of the function, one MOEA needs to perform at least the following three operations: obtaining an initial population via *Global.Initialization*(), checking the optimization state and outputting the final population via *Global.NotTermination*(), and generating offsprings via *Global.Variation*(), where all these three functions are provided by the *GLOBAL* object. Apart from the above three common operations, different MOEAs may have different logics and different functions to be invoked.

B. Adding New Problems to PlatEMO

All the *.m* files of MOP functions are stored in the folder *\Problems*, and one *.m*

TABLE 5 IGD Values of NSGA-III, KnEA and RVEA on DTLZ1–DTLZ4. The LaTeX Code of This Table is Automatically Generated by PlatEMO.

PROBLEM	*M*	NSGA-III	KnEA	RVEA
DTLZ1	2	1.1556E−1 (1.49E−1) ≈	7.5785E−2 (1.47E−1) +	4.6018E−1 (5.52E−1)
	3	2.9794E−1 (2.85E−1) ≈	1.8036E−1 (1.37E−1) ≈	4.2677E−1 (2.33E−1)
	4	4.6226E−1 (2.66E−1) ≈	2.6713E−1 (2.55E−1) ≈	7.3718E−1 (7.76E−1)
DTLZ2	5	2.1746E−1 (9.86E−4) −	2.4033E−1 (1.14E−2) −	2.1394E−1 (3.65E−4)
	6	2.8677E−1 (2.01E−3) −	3.1285E−1 (9.17E−3) −	2.8203E−1 (9.06E−4)
	7	3.5352E−1 (3.93E−3) −	3.6826E−1 (7.51E−3) −	3.4342E−1 (9.55E−4)
DTLZ3	8	5.8255E+1 (3.97E+1) −	1.3282E+2 (6.05E+1) −	9.2466E+0 (4.45E+0)
	9	5.4964E+1 (1.19E+1) −	2.0923E+2 (5.21E+1) −	1.4395E+1 (5.95E+0)
	10	6.6158E+1 (4.95E+0) −	2.6730E+2 (1.10E+2) −	1.2605E+1 (9.14E+0)
DTLZ4	11	6.1257E−1 (1.03E−1) ≈	5.3976E−1 (7.19E−3) ≈	5.8265E−1 (3.49E−2)
	12	7.0350E−1 (8.27E−2) ≈	6.1240E−1 (5.45E−3) ≈	6.3605E−1 (2.85E−2)
	13	6.6993E−1 (5.79E−2) ≈	6.0250E−1 (2.03E−3) ≈	6.3893E−1 (2.54E−2)
+/−/≈		0/6/6	1/6/5	

```
1. function NSGAII(Global)
2.   Population = Global.Initialization();
3.   FrontNo    = NDSort(Population.objs,inf);
4.   CrowdDis   = CrowdingDistance(Population.objs,FrontNo);
5.   while Global.NotTermination(Population)
6.     MatingPool = TournamentSelection(2,Global.N,FrontNo,-CrowdDis);
7.     Offspring    = Global.Variation(Population(MatingPool));
8.     [Population,FrontNo,CrowdDis] = ...
9.     EnvironmentalSelection([Population,Offspring],Global.N);
10.  end
11.end
```

FIGURE 8 The source code of the main function of NSGA-II. The common code required by any MOEA is underlined.

```
1. function varargout = DTLZ2(Operation,Global,input)
2.   switch Operation
3.     case 'init'
4.       Global.M          = 3;
5.       Global.D          = Global.M + 9;
6.       Global.lower      = zeros(1,Global.D);
7.       Global.upper      = ones(1,Global.D);
8.       Global.operator = @Eareal;
9.       PopDec     = rand(input,Global.D);
10.      varargout = {PopDec};
11.    case 'value'
12.      PopDec = input;
13.      M          = Global.M;
14.      PopObj  = zeros(size(PopDec,1),M);
15.      g          = sum((PopDec(:,M:end)-0.5).^2,2);
16.      for m = 1 : M
17.        PopObj(:,m) = (1+g).*prod(cos(PopDec(:,1:M-m)*pi/2),2);
18.        if m > 1
19.          PopObj(:,m) = PopObj(:,m).*sin(PopDec(:,M-m+1)*pi/2);
20.        end
21.      end
22.      PopCon    = [];
23.      varargout = {PopDec,PopObj,PopCon};
24.    case 'PF'
25.      f = UniformWeight(input,Global.M);
26.      f = f./repmat(sqrt(sum(f.^2,2)),1,Global.M);
27.      varargout = {f};
28.  end
29.end
```

FIGURE 9 The source code of DTLZ2. The common code required by any MOP is underlined.

file usually indicates one MOP. Fig. 9 gives the source code of DTLZ2, where the common code required by any MOP is underlined. It can be seen from the source code that, the interface of DTLZ2 is more complex than the one of NSGA-II, where the function *DTLZ2*() includes three input parameters and one output parameter. The input parameter *Operation* determines the operation to be performed; the parameter *Global* denotes the *GLOBAL* object; and the parameter *input* has different meanings when *Operation* is set to different values, so does the output parameter *varargout*.

Different from the MOEA functions which are invoked only once in each run, an MOP function may be invoked many times for different operations. As shown in Fig. 9, an MOP function contains three independent operations: generating random decision variables (line 3–10), calculating objective values and constraint values (line 11–23), and sampling reference points on the true Pareto front (line 24–27). To be specific, if *Operation* is set to *'init'*, the MOP function will return the decision variables of a random population with size *input* (line 9–10). Meanwhile, it sets *Global.M*, *Global.D*, *Global.lower*, *Global.upper* and *Global.operator* to their default values, which denote the number of objectives, number of decision variables, lower boundary of each decision variable, upper boundary of each decision variable, and the operator function, respectively (line 4–8). When *Operation* is set to *'value'*, the parameter *input* will denote the decision variables of a population, and the objective values and constraint values of the population will be calculated and returned according to the decision variables (line 14–23). And if *Operation* is set to *'PF'*, a number of *input* uniformly distributed reference points will be sampled on the true Pareto front and returned (line 25–27).

C. Adding New Operators or Performance Indicators to PlatEMO

Fig. 10 shows the source code of evolutionary operator based on binary coding (i.e. *EAbinary.m*), where the *.m* files of the operator functions are all stored in the folder *Operators*. An operator function has two input parameters, one denoting the *GLOBAL* object (i.e. *Global*) and the other denoting the parent population (i.e. *Parent*), and it also has one output parameter denoting the generated offsprings (i.e. *Offspring*). As can be seen from the source code in Fig. 10, the main task of an operator function is to generate offsprings according to the values of *Parent*, where *EAbinary*() performs the single-point crossover in line 6–11 and the bitwise mutation in line 12–13 of the code. Afterwards, the *INDIVIDUAL* objects of the offsprings are generated and returned (line 14).

Fig. 11 shows the source code of IGD, where all these performance indicator functions are stored in the folder *Metrics*. The task of a performance indicator is to calculate the indicator value of a population according to a set of reference points. The input parameters of *IGD*() consists of two parts: the objective values of the population (i.e. *PopObj*), and the reference points sampled on the true Pareto front (i.e. *PF*). Correspondingly, the output parameter of *IGD*() is the IGD value (i.e. *score*). Thanks to the merits of matrix operation in MATLAB, the source code of IGD is quite short as shown in Fig. 11, where the calculation of the mean value of the minimal distance of each point in *PF* to the points in *PopObj* can be performed using a built-in function *pdist2*() provided by MATLAB.

D. Adding Acceptable Parameters for New Functions

All the user-defined functions can have their own parameters as well as the functions provided by PlatEMO, where these parameters can be either assigned by invoking *main*(…,'-*X_parameter*',{…},…) with *X* denoting the function name, or displayed on the GUI for assignment. In order to add acceptable parameters for an MOEA, MOP, operator or performance indicator function, the comments in the head of the function should be written in a specified form. To be specific, Fig. 12 shows the comments and the source code in the head of the function of evolutionary operator based on real value coding (i.e. *EAreal.m*).

The comment in line 2 of Fig. 12 gives the two labels of this function, which are used to make sure this function can be identified by the GUI. The comment in line 3 is a brief introduction about this function; for an MOEA or MOP function, such introduction should be the title of the relevant literature. The parameters *proC*, *disC*, *proM* and *disM* for this function are given by the comments in line 4–7, where the names of the parameters are in the first column, the default values of the parameters are in the second column, and the introductions about the parameters are given in the third column. The columns in each row are divided by the sign '—'.

The comments define the parameters and their default values for the function, and invoking *Global.ParameterSet*() can make these parameters assignable to users. As shown in line 9 of Fig. 12, the function invokes *Global.ParameterSet*() with four inputs denoting the default values of the parameters, and sets the four parameters to the outputs. More specifically, if users have not assigned the parameters, they will equal to their default values (i.e. 1, 15, 1 and 15). Otherwise, if users assign the parameters by invoking *main*(…,'-*EAreal_parameter*',{*a*,*b*,*c*,*d*},…), the parameters *proC*, *disC*, *proM* and *disM* will be set to *a*, *b*, *c* and *d*, respectively.

```
function Offspring = EAbinary(Global,Parent)
  parentDec = Parent.decs;
  [N,D]       = size(parentDec);
  Parent1Dec          = parentDec(1:N/2,:);
  Parent2Dec          = parentDec(N/2+1:end,:);
  k                   = repmat(1:D,N/2,1) > repmat(randi(D,N/2,1),1,D);
  Offspring1Dec       = Parent1Dec;
  Offspring2Dec       = Parent2Dec;
  Offspring1Dec(k)    = Parent2Dec(k);
  Offspring2Dec(k)    = Parent1Dec(k);
  OffspringDec        = [Offspring1Dec;Offspring2Dec];
  site                = rand(N,D) < 1/D;
  OffspringDec(site)  = ~OffspringDec(site);
  Offspring = INDIVIDUAL(OffspringDec);
end
```

FIGURE 10 The source code of evolutionary operator based on binary coding. The common code required by any operator is underlined.

V. Conclusion and Future Work

This paper has introduced a MATLAB-based open source platform for evolutionary multi-objective optimization, namely PlatEMO. The current version of PlatEMO includes 50 multi-objective optimization algorithms and 110 multi-objective test problems, having covered the majority of state-of-the-arts. Since PlatEMO is developed on the basis of a light architecture with simple relations between objects, it is very easy to be used and extended. Moreover, PlatEMO provides a user-friendly GUI with a powerful experimental module, where engineers and researchers can use it to quickly perform their experiments without writing any additional code. This paper has described the architecture of PlatEMO, and it has also introduced the steps of running PlatEMO with and without the GUI. Then, the ways of adding new algorithms,

```
function score = IGD(PopObj,PF)
  score = mean(min(pdist2(PF,PopObj),[],2));
end
```

FIGURE 11 The source code of IGD. The common code required by any performance indicator is underlined.

```
1. function Offspring = EAreal(Global,Parent)
2. % <operator> <real>
3. % Simulated binary crossover and polynomial mutation
4. % proC ---  1 --- The probability of doing crossover
5. % disC --- 15 --- The distribution index of SBX
6. % proM ---  1 --- The expectation of number of bits doing mutation
7. % disM --- 15 --- The distribution index of polynomial mutation
8.
9.    [proC,disC,proM,disM] = Global.ParameterSet(1,15,1,15);
… …
```

FIGURE 12 The comments and the source code in the head of the function of evolutionary operator based on real value coding.

problems, operators and performance indicators to PlatEMO have been elaborated by several cases.

It is worth noting that, despite that we have performed careful validation and test on the source code of PlatEMO, there may still exist incorrect re-implementations or bugs. To address this issue, we have carefully followed the advice in dealing with algorithm implementations in [105]. We have performed unit tests on every algorithm in PlatEMO thoroughly, highlighted the source of the re-using code, and reported the version number and the change log on the website after each update as suggested in [105]. In addition, since all of the algorithms in PlatEMO are completely open-source, we sincerely hope that users of the software could also join us to make further improvements.

We will continuously maintain and develop PlatEMO in the future. On one hand, we will keep following the state-of-the-arts and adding more effective algorithms and new problems into PlatEMO. On the other hand, more modules will be developed to provide more functions for users, such as preference optimization, dynamic optimization, noisy optimization, etc. We hope that PlatEMO is helpful to the researchers working on evolutionary multi-objective optimization, and we sincerely encourage peers to join us to shape the platform for better functionality and usability.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (Grant No. 61672033, 61272152, 615012004, 61502001), and the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (Grant No. 61428302). This manuscript was written during Y. Tian's visit at the Department of Computer Science, University of Surrey. The authors would like to thank Mr. Kefei Zhou and Mr. Ran Xu for their valued work in testing the PlatEMO.

References

[1] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 4–19, 2014.
[2] J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE Trans. Evol. Comput.*, vol. 11, no. 1, pp. 56–76, 2007.
[3] B. Lazzerini, F. Marcelloni, and M. Vecchio, "A multi-objective evolutionary approach to image quality/compression trade-off in JPEG baseline algorithm," *Appl. Soft Computing*, vol. 10, no. 2, pp. 548–561, 2010.
[4] F. Pettersson, N. Chakraborti, and H. Saxén, "A genetic algorithms based multi-objective neural net applied to noisy blast furnace data," *Appl. Soft Computing*, vol. 7, no. 1, pp. 387–397, 2007.
[5] S. H. Yeung, K. F. Man, K. M. Luk, and C. H. Chan, "A trapeizform U-slot folded patch feed antenna design optimized with jumping genes evolutionary algorithm," *IEEE Trans. Antennas Propagat.*, vol. 56, no. 2, pp. 571–577, 2008.
[6] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 321–344, 2013.
[7] J. G. Herrero, A. Berlanga, and J. M. M. López, "Effective evolutionary algorithms for many-specifications attainment: Application to air traffic control tracking filters," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 151–168, 2009.
[8] H. Ishibuchi and T. Murata, "Multiobjective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Systems, Man, Cybernet. C*, vol. 28, no. 3, pp. 392–403, 1998.
[9] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, 1985, pp. 93–100.
[10] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, vol. 93, pp. 416–423.
[11] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1995.
[12] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1994 IEEE Congr. Evolutionary Computation*, 1994, pp. 82–87.
[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.
[14] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. 5th Conf. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2001, pp. 95–100.
[15] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region-based selection in evolutionary multi-objective optimization," in *Proc. 2001 Conf. Genetic and Evolutionary Computation*, 2001, pp. 283–290.
[16] T. Murata, H. Ishibuchi, and M. Gen, "Specification of genetic search directions in cellular multiobjective genetic algorithms," in *Proc. 1st Int. Conf. Evolutionary Multi-Criterion Optimization*, 2001, pp. 82–95.
[17] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007.
[18] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, 2009.
[19] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Trans. Evol. Computat.*, vol. 18, no. 3, pp. 450–455, 2014.
[20] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "Balancing convergence and diversity in decomposition-based many-objective optimizers," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 180–198, 2016.
[21] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 169–190, 2017.
[22] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.
[23] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.
[24] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich, "Opt4J: A modular framework for meta-heuristic optimization," in *Proc. 13th Annu. Conf. Genetic and Evolutionary Computation*, 2011, pp. 1723–1730.
[25] R. Shen, J. Zheng, and M. Li, "A hybrid development platform for evolutionary multi-objective optimization," in *Proc. IEEE Congr. Evolutionary Computation*, 2015, pp. 1885–1892.
[26] A. Liefooghe, M. Basseur, L. Jourdan, *and* E.-G. Talbi, "ParadisEO-MOEO: A framework for evolutionary multi-objective optimization," in *Proc. Int. Conf. Evolutionary Multi-Criterion Optimization*, 2007, pp. 386–400.
[27] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA-a platform and programming language independent interface for search algorithms," in *Proc. Int. Conf. Evolutionary Multi-Criterion Optimization*, 2003, pp. 494–508.
[28] C. Igel, V. Heidrich-Meisner, and T. Glasmachers, "Shark," *J. Mach. Learn. Res.*, no. 9, pp. 993–996, 2008.
[29] D. Izzo, "PyGMO and PyKEP: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization)," in *Proc. 5th Int. Conf. Astrodynamics Tools and Techniques*, 2012.
[30] L. C. Bezerra, M. López-Ibánez, and T. Stützle, "Automatic component-wise design of multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 403–417, 2016.
[31] J. Humeau, A. Liefooghe, E.-G. Talbi, and S. Verel, "ParadisEO-MO: From fitness landscape analysis to efficient local search algorithms," *J. Heuristics*, vol. 19, no. 6, pp. 881–915, 2013.
[32] K. Deb, M. Mohan, and S. Mishra, "Towards a quick computation of well-spread Pareto-optimal solutions," in *Proc. 2003 Int. Conf. Evolutionary Multi-Criterion Optimization*, 2003, pp. 222–236.
[33] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Int. Conf. Parallel Problem Solving from Nature*, 2004, pp. 832–842.
[34] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, 2007.
[35] E. J. Hughes, "MSOPS-II: A general-purpose many-objective optimiser," in *Proc. 2007 IEEE Congr. Evolutionary Computation*, 2007, pp. 3944–3951.
[36] L.-Y. Tseng and C. Chen, "Multiple trajectory search for unconstrained/constrained multi-objective optimization," in *Proc. 2009 IEEE Congr. Evolutionary Computation*, 2009, pp. 1951–1958.
[37] M. Wagner and F. Neumann, "A fast approximation-guided evolutionary multi-objective algorithm," in *Proc. 15th Annu. Conf. Genetic and Evolutionary Computation*, 2013, pp. 687–694.
[38] B. Chen, W. Zeng, Y. Lin, and D. Zhang, "A new local search-based multiobjective optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 50–73, 2015.
[39] M. Li, S. Yang, and X. Liu, "Pareto or non-Pareto: Bi-criterion evolution in multi-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 645–665, 2015.
[40] Y. Tian, X. Zhang, R. Cheng, and Y. Jin, "A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric," in *Proc. IEEE Congr. Evolutionary Computation*, 2016, pp. 5222–5229.
[41] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.
[42] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithms for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 474–494, 2013.
[43] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization,"

IEEE Trans. Evol. Comput., vol. 17, no. 5, pp. 721–736, 2013.
[44] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, 2014.
[45] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, 2014.
[46] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for pareto-based algorithms in many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 348–365, 2014.
[47] M. Li, S. Yang, and X. Liu, "Bi-goal evolution for many-objective optimization problems," *Artif. Intell.*, vol. 228, pp. 45–65, 2015.
[48] M. Asafuddoula, T. Ray, and R. Sarker, "A decomposition based evolutionary algorithm for many objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 445–460, 2015.
[49] X. Zhang, Y. Tian, and Y. Jin, "A knee point driven evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 761–776, 2015.
[50] J. Cheng, G. Yen, and G. Zhang, "A many-objective evolutionary algorithm with enhanced mating and environmental selections," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 592–605, 2015.
[51] K. Li, K. Deb, Q. Zhang, and S. Kwong, "Combining dominance and decomposition in evolutionary many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, 2015.
[52] R. Hernández Gómez and C. A. Coello, "Improved metaheuristic based on the R2 indicator for many-objective optimization," in *Proc. Genetic and Evolutionary Computation Conf.*, 2015, pp. 679–686.
[53] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 524–541, 2015.
[54] Z. He and G. G. Yen, "Many-objective evolutionary algorithm: Objective space reduction and diversity improvement," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 145–160, 2016.
[55] Y. Liu, D. Gong, X. Sun, and Z. Yong, "Many-objective evolutionary optimization based on reference points," *Appl. Soft Comput.*, vol. 50, pp. 344–355, 2017.
[56] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, 2016.
[57] S. Jiang and S. Yang, "A strength Pareto evolutionary algorithm based on reference direction for multi-objective and many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 329–346, 2017.
[58] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 16–37, 2016.
[59] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 275–298, 2016.
[60] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Trans. Evol. Comput.*, to be published.
[61] J. Molina, L. V. Santana, A. G. Hernández-Díaz, C. A. C. Coello, and R. Caballero, "G-Dominance: Reference point based dominance for multiobjective metaheuristics," *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 685–692, 2009.
[62] L. B. Said, S. Bechikh, and K. Ghédira, "The r-dominance: A new dominance relation for interactive evolutionary multicriteria decision making," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 801–818, 2010.
[63] X. Zhang, X. Jiang, and L. Zhang, "A weight vector based multi-objective optimization algorithm with preference," *Acta Electron. Sin.*, vol. 44, no. 11, pp. 2639–2645, 2016.
[64] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Proc. IEEE Congr. Evolutionary Computation*, 2005, vol. 1, pp. 443–450.
[65] C. C. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. 2002 IEEE Congr. Evolutionary Computation*, 2002, vol. 2, pp. 1051–1056.
[66] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. C. Coello, F. Luna, and E. Alba, "SMPSO: A new PSO-based metaheuristic for multi-objective optimization," in *Proc. IEEE Symp. Computational Intelligence in Multi-Criteria Decision-Making*, 2009, pp. 66–73.
[67] S. Zapotecas Martínez and C. A. Coello, "A multi-objective particle swarm optimizer based on decomposition," in *Proc. 13th Annu. Conf. Genetic and Evolutionary Computation*, 2011, pp. 69–76.
[68] J. D. Knowles and D. W. Corne, "M-PAES: A memetic algorithm for multiobjective optimization," in *Proc. IEEE Congr. Evolutionary Computation*, 2000, pp. 325–332.
[69] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evol. Comput.*, vol. 15, no. 1, pp. 1–28, 2007.
[70] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, 2008.
[71] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process based inverse modeling," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 838–856, 2015.
[72] J. Knowles, "ParEGO: A hybrid algorithm with online landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, 2006.
[73] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection," in *Proc. 2008 Int. Conf. Parallel Problem Solving from Nature*, 2008, pp. 784–794.
[74] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, "A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization," *IEEE Trans. Evol. Comput.*, to be published.
[75] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, 1999.
[76] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 264–283, 2015.
[77] K. D. E. Zitzler and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[78] J. Knowles and D. Corne, "Instance generators and test suites for the multiobjective quadratic assignment problem," in *Proc. 2003 Int. Conf. Evolutionary Multi-Criterion Optimization*, 2003, pp. 295–310.
[79] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Proc. Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, 2005, pp. 105–145.
[80] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, 2006.
[81] H. Ishibuchi, H. Masuda, and Y. Nojima, "Pareto fronts of many-objective degenerate test problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 807–813, 2016.
[82] Y. Zhang, M. Harman, and S. A. Mansouri, "The multi-objective next release problem," in *Proc. 9th Annu. Conf. Genetic and Evolutionary Computation*, 2007, pp. 1129–1137.
[83] D. W. Corne and J. D. Knowles, "Techniques for highly multiobjective optimisation: Some nondominated points are better than others," in *Proc. 9th Conf. Genetic and Evolutionary Computation*, 2007, pp. 773–780.
[84] M. Köppen and K. Yoshida, "Substitute distance assignments in NSGA-II for handling many-objective optimization problems," in *Proc. 2007 Int. Conf. Evolutionary Multi-criterion Optimization*, 2007, pp. 727–741.
[85] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," Univ. Essex, Colchester, U.K., and Nanyang Technol. Univ., Tech. Rep. CES-487, 2008.
[86] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff, "On test functions for evolutionary multi-objective optimization," in *Proc. 2004 Int. Conf. Parallel Problem Solving from Nature*, 2004, pp. 792–802.
[87] H. Li, Q. Zhang, and J. Deng, "Biased multiobjective optimization and decomposition algorithm," *IEEE Trans. Cybernetics*, vol. 47, no. 1, pp. 52–66, 2017.
[88] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Trans. Cybernetics*, to be published.
[89] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Dept. Electr. Computer Eng. Graduate School of Eng., Air Force Inst. Technol, Wright Patterson, Tech. Rep. TR-98-03, 1998.
[90] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 29–38, 2006.
[91] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion," in *Proc. IEEE Congr. Evolutionary Computation*, 2006, pp. 892–899.
[92] H. Wang, Y. Jin, and X. Yao, "Diversity assessment in many-objective optimization," *IEEE Trans. Cybernet.*, vol. 47, no. 6, pp. 1510–1522, 2017.
[93] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," M.S. thesis, Massachusetts Inst. Technol., Cambridge, 1995.
[94] Y. Wang, L. Wu, and X. Yuan, "Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure," *Soft Comput.*, vol. 14, no. 3, pp. 193–209, 2010.
[95] Z. He and G. G. Yen, "Performance metric ensemble for multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 131–144, 2014.
[96] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York: Wiley, 2001.
[97] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Sci. Informat.*, vol. 26, no. 4, pp. 30–45, 1996.
[98] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proc. Int. Joint Conf. Artificial Intelligence*, vol. 1, 1985, pp. 162–164.
[99] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biol. Cybernet.*, vol. 60, no. 2, pp. 139–144, 1988.
[100] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer Science & Business Media, 2006.
[101] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.
[102] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.
[103] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to non-dominated sorting for evolutionary multi-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, 2015.
[104] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "Empirical analysis of a tree-based efficient non-dominated sorting approach for many-objective optimization," in *Proc. IEEE Symp. Series on Computational Intelligence*, 2016, pp. 1–8.
[105] D. Brockhoff, "A bug in the multiobjective optimizer IBEA: Salutary lessons for code release and a performance re-assessment," in *Proc. Int. Conf. Evolutionary Multi-Criterion Optimization*, 2015, pp. 187–201.