PlatEMO

进化多目标优化平台

用户手册 3.0

生物智能与知识发现（BIMK）研究所

2021 年 2 月 1 日

非常感谢使用由安徽大学生物智能与知识发现（BIMK）研究所开发的进化多目标优化平台 PlatEMO。本平台是一个开源免费的代码库，仅供教学与科研使用，不得用于商业用途。本平台中的代码基于作者对论文的理解编写而成，作者不对用户因使用代码产生的任何后果负责。包含利用本平台产生的数据的论文应在正文中声明对 PlatEMO 的使用，并引用以下参考文献：

Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin, “PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum],” IEEE Computational Intelligence Magazine, 2017, 12(4): 73-87.

如有任何意见或建议，欢迎联系 field910921@gmail.com（田野）。如想将您的代码添加进 PlatEMO 中并公开，也欢迎联系 field910921@gmail.com。您可以在 https://github.com/BIMK/PlatEMO 上获取 PlatEMO 的最新版本。

目　录

一 快速入门

PlatEMO 提供一系列的元启发式算法用于求解各类优化问题。为此，用户需要定义优化问题、选择求解算法并设置参数。PlatEMO 提供以下三种调用方式：

1) 带参数调用主函数 (需要 MATLAB R2012a 或更高版本)：

```
platemo('problem',@SOP_F1,'algorithm',@GA,'Name',Value,…);
```

可以利用指定的算法来求解指定的测试问题并设置参数，求解结果可以被显示在窗口中、保存在文件中或作为函数返回值（参阅*求解测试问题*章节）。

2) 带参数调用主函数 (需要 MATLAB R2012a 或更高版本)：

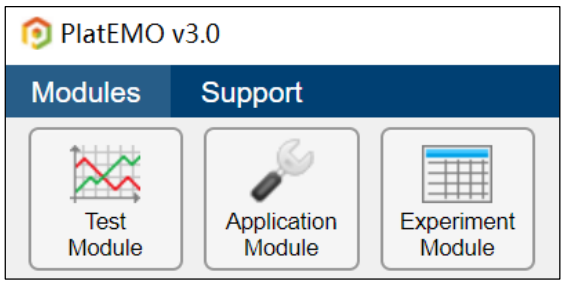
```
f1 = @(x,d)sum(x*d);
f2 = @(x,d)1-sum(x*d);
platemo('objFcn',f1,'conFcn',f2,'algorithm',@GA,…);
```

可以利用指定的算法来求解自定义的问题（参阅*求解自定义问题*章节）。

3) 不带参数调用主函数 (需要 MATLAB R2020b 或更高版本)：

```
platemo();
```

可以弹出一个带有三个模块的图形界面，其中测试模块用于可视化地研究单个算法在单个问题上的性能（参阅*测试模块*章节），应用模块用于求解自定义问题（参阅*应用模块*章节），实验模块用于统计分析多个算法在多个问题上的性能（参阅*实验模块*章节）。



二 通过命令行使用 PlatEMO

1. 求解测试问题

用户可以以如下形式带参数调用主函数 platemo() 来求解测试问题：

```
platemo('Name1',Value1,'Name2',Value2,'Name3',Value3,…);
```

其中所有可接受的参数列举如下：

参数名	数据类型	默认值	描述
'algorithm'	函数句柄或单元数组	不定	算法类
'problem'	函数句柄或单元数组	不定	问题类
'N'	正整数	100	种群大小
'M'	正整数	不定	问题的目标数
'D'	正整数	不定	问题的变量数
'maxFE'	正整数	10000	最大评价次数
'save'	整数	0	保存的种群数
'outputFcn'	函数句柄	@ALGORITHM.Output	每代开始前调用的函数

- 'algorithm'表示待运行的算法，它的值可以是一个算法类的句柄，例如@GA。它的值还可以是形如{@GA,p1,p2,…}的单元数组，其中 p1,p2,… 指定了该算法中的参数值。
- 'problem'表示待求解的测试问题，它的值可以是一个问题类的句柄，例如@SOP_F1。它的值还可以是形如{@SOP_F1,p1,p2,…}的单元数组，其中 p1,p2,… 指定了该算法中的参数值。
- 'N'表示算法的种群大小，它通常等于最终种群中解的个数。
- 'M'表示问题的目标数，它仅对一些多目标测试问题生效。
- 'D'表示问题的变量数，它仅对一些测试问题生效。
- 'maxFE'表示算法可使用的最大评价次数，算法将在产生的解的数目超过该值时立即停止。
- 'save'表示保存的种群数，该值大于零时结果将被保存在文件中，该值等

于零时结果将被显示在窗口中（参阅获取运行结果章节）。

- 'outputFcn'表示算法每代开始前调用的函数。该函数必须有两个输入和零个输出，其中第一个输入是当前的 ALGORITHM 对象、第二个输入是当前的 PROBLEM 对象。

例如，以下代码利用遗传算法求球面函数的最小值，其中种群大小为 50 且最终种群会被显示在窗口中：

```
platemo('algorithm',@GA,'problem',@SOP_F1,'N',50);
```

以下代码利用 NSGA-II 来求解 5 目标、40 变量的 DTLZ2 问题，其中最大评价次数为 20000 且最终种群会被保存在文件中：

```
platemo('algorithm',@NSGAII,'problem',@DTLZ2,'M',5,'D',40,'maxFE',20000,'save',10);
```

以下代码利用基于 Tchebycheff 方法的 MOEA/D 求解 ZDT1 问题十次，其中每次的结果会被保存在独立的文件中：

```
for i = 1 : 10
    platemo('algorithm',{@MOEAD,2},'problem',@ZDT1,'save',5);
end
```

注意每个参数均有一个默认值，用户可以在调用时省略任意参数。

2. 求解自定义问题

当不指定参数'problem'时，用户可以通过指定以下参数来自定义问题：

参数名	数据类型	默认值	描述
'encoding'	字符串	'real'	问题的编码方式
'objFcn'	函数句柄或单元数组	@(x,d)sum(x)	问题的目标函数
'conFcn'	函数句柄或单元数组	@(x,d)0	问题的约束
'lower'	行向量	0	变量的下界
'upper'	行向量	1	变量的上界
'initFcn'	函数句柄	[]	种群初始化函数
'decFcn'	函数句柄	[]	无效解修复函数
'parameter'	单元数组	{}	问题的数据集

- 'encoding'表示问题的编码方式，它的值可以是'real'（实数或整数变量）、'binary'（二进制变量）或'permutation'（序列变量）。算法针对不同的编码方式可能使用不同的算子来产生子代。
- 'objFcn'表示问题的目标函数，它的值可以是一个函数句柄（单目标）或一个单元数组（多目标）。每个目标函数必须有两个输入和一个输出，其中第一个输入是一个决策向量、第二个输入是由'parameter'指定的数据集、输出是目标值。所有目标函数均为最小化问题。
- 'conFcn'表示问题的约束，它的值可以是一个函数句柄（单约束）或一个单元数组（多约束）。每个约束函数必须有两个输入和一个输出，其中第一个输入是一个决策向量、第二个输入是由'parameter'指定的数据集、输出是约束违反值。当且仅当约束违反值小于等于零时，约束被满足。
- 'lower'表示决策变量的下界，它仅在'encoding'的值为'real'时生效。
- 'upper'表示决策变量的上界，它仅在'encoding'的值为'real'时生效。
- 'initFcn'表示种群初始化函数，它的值必须是一个函数句柄。该函数必须有两个输入和一个输出，其中第一个输入是种群大小、第二个输入是由'parameter'指定的数据集、输出是种群的决策向量构成的矩阵。该函数通常在算法开始时被调用。
- 'decFcn'表示无效解修复函数，它的值必须是一个函数句柄。该函数必须有两个输入和一个输出，其中第一个输入是一个决策向量、第二个输入是由'parameter'指定的数据集、输出是修复后的决策向量。该函数会在计算目标函数前被调用。
- 'parameter'表示问题的数据集，它作为函数'objFcn'、'conFcn'、'initFcn'和'decFcn'的第二个输入参数。

例如，以下代码利用差分进化算法求一个 10 变量的单峰函数的最小值：

```
platemo('objFcn',@(x,d)sum(x.^2),'lower',zeros(1,10)-10,
'upper',zeros(1,10)+10,'algorithm',@DE);
```

以下代码利用默认的算法求一个带旋转的 10 变量单峰函数的最小值：

```
platemo('objFcn',@(x,d)sum((x*d).^2),'lower',zeros(1,10)-
10,'upper',zeros(1,10)+10,'parameter',rand(10));
```

以下代码利用 NSGA-II 求一个带约束的 2 目标、20 变量的优化问题的最小值，其中种群大小为 50：

```
f1 = @(x,d)x(1)*sum(x(2:end));
f2 = @(x,d)sqrt(1-x(1)^2)*sum(x(2:end));
g1 = @(x,d)1-sum(x(2:end));
platemo('objFcn',{f1,f2},'conFcn',g1,'lower',zeros(1,20),'upper',ones(1,20),'algorithm',@NSGAII,'N',50);
```

3. 获取运行结果

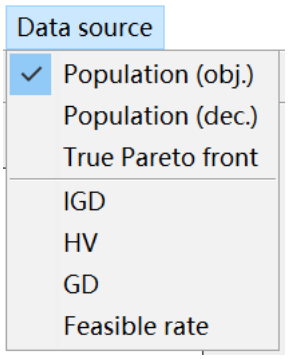
算法运行结束后得到的种群可以被显示在窗口中、保存在文件中或作为函数返回值。若按以下方式调用主函数：

```
[Dec,Obj,Con] = platemo(…);
```

则最终种群会被返回，其中 Dec 表示种群的决策向量构成的矩阵、Obj 表示种群的目标值构成的矩阵、Con 表示种群的约束违反值构成的矩阵。若按以下方式调用主函数：

```
platemo('save',Value,…);
```

则当 Value 的值为零时（默认情况），得到的种群会被显示在窗口中，用户可以利用窗口中的 Data source 菜单来选择要显示的内容。



当 Value 的值大于零时，得到的种群会被保存在命名为 PlatEMO\Data\alg\alg_pro_M_D_run.mat 的 MAT 文件中，其中 alg 表示算法名、pro 表示问题名、M 表示目标数、D 表示变量数、run 是一个自动确定的正整数以保证不和已有文件重名。每个文件存储一个单元数组 result 和一个结构体 metric。算法的整个优化过程被等分为 Value 块，其中 result 的第一列存储每块最后一代时所消耗的评价次数、result 的第二列存储每块最后一代时的种群、metric 存储所有种群的指标值。以上操作均由默认的输出函数@ALGORITHM.Output 实现，用户可以通过指定'outputFcn'的值为其它函数来实现自定义的结果展示或保存方式。

```
result =
  6×2 cell array
    {[ 1650]}    {1×50 SOLUTION}
    {[ 3300]}    {1×50 SOLUTION}
    {[ 5000]}    {1×50 SOLUTION}
    {[ 6650]}    {1×50 SOLUTION}
    {[ 8300]}    {1×50 SOLUTION}
    {[10000]}    {1×50 SOLUTION}
```

```
metric =
  struct with fields:

    runtime: 0.3317
        IGD: [6×1 double]
```

此外，图形界面的实验模块可以自动计算种群的指标值并存储到 `metric` 中。若需要手动计算指标值，用户需获取问题的最优值并调用指标函数，例如

```
pro = DTLZ2();
IGD(result{end},pro.optimum);
```

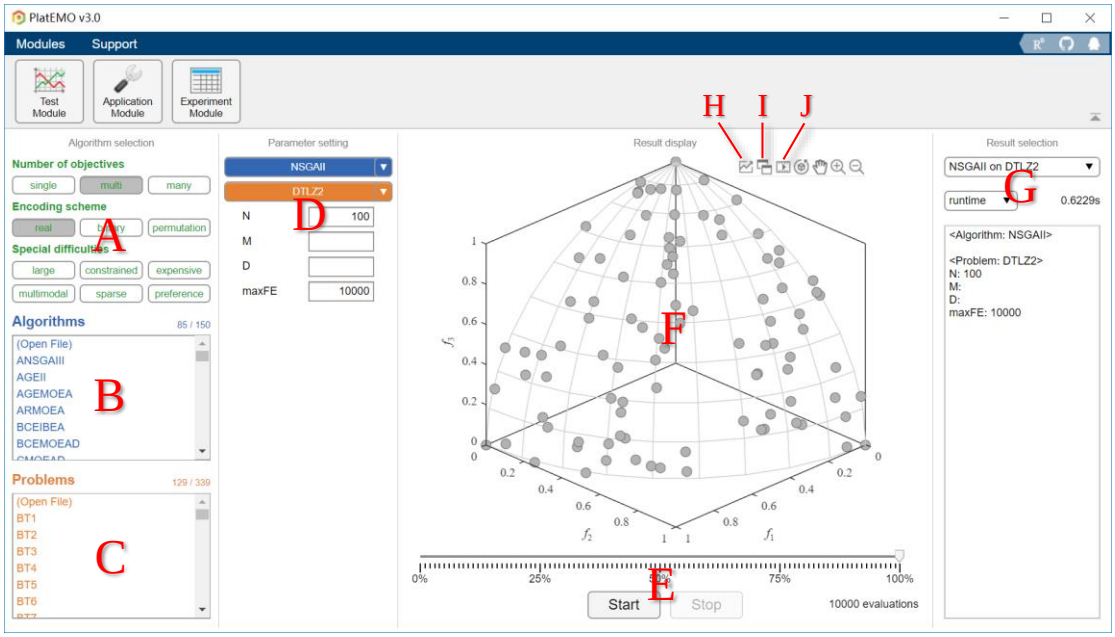
三　通过图形界面使用 PlatEMO

1. 测试模块

用户可以通过无参数调用主函数 `platemo()` 来使用 PlatEMO 的图形界面：

```
platemo();
```

图形界面的测试模块会被首先显示，它用于可视化地研究单个算法在单个问题上的性能。

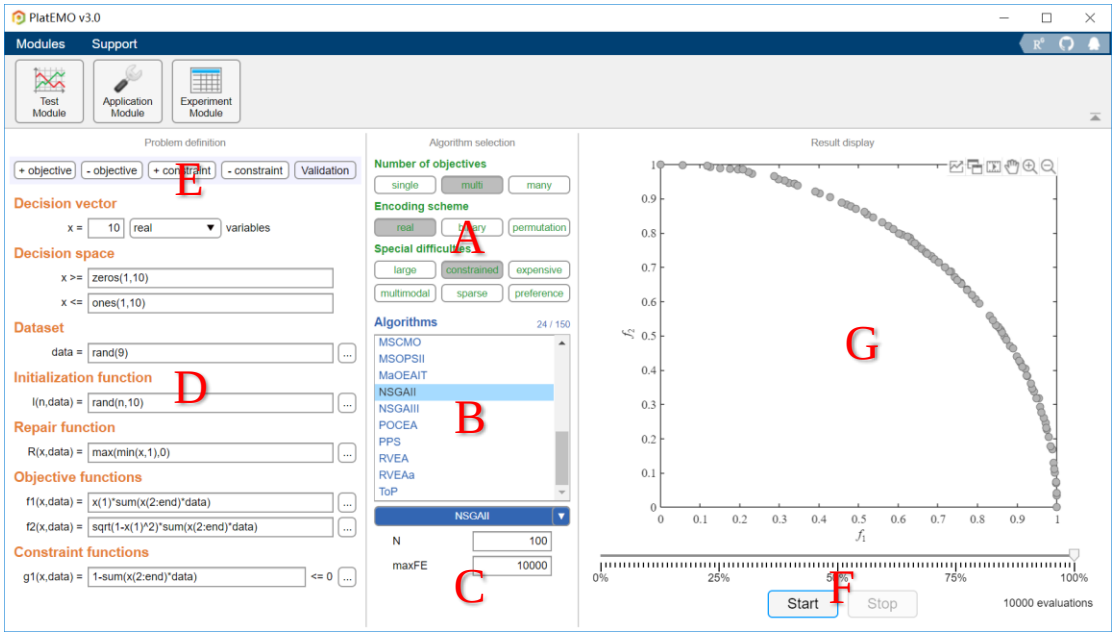


在该模块中，用户首先需要在区域 A 中选择问题类型（参阅*算法和问题的标签*章节），在区域 B 中选择一个算法，在区域 C 中选择一个测试问题，并在区域 D 中设定相关参数。之后，用户可以在区域 E 中控制算法的运行，在区域 F 中观察算法运行的实时结果，并在区域 G 中调取历史运行结果。

按钮 H 用于选择要显示的内容，按钮 I 用于将当前显示内容显示在一个新窗口中并存储至工作空间，按钮 J 用于将算法运行的动态过程保存为一个 20 帧的 GIF 图像。

2. 应用模块

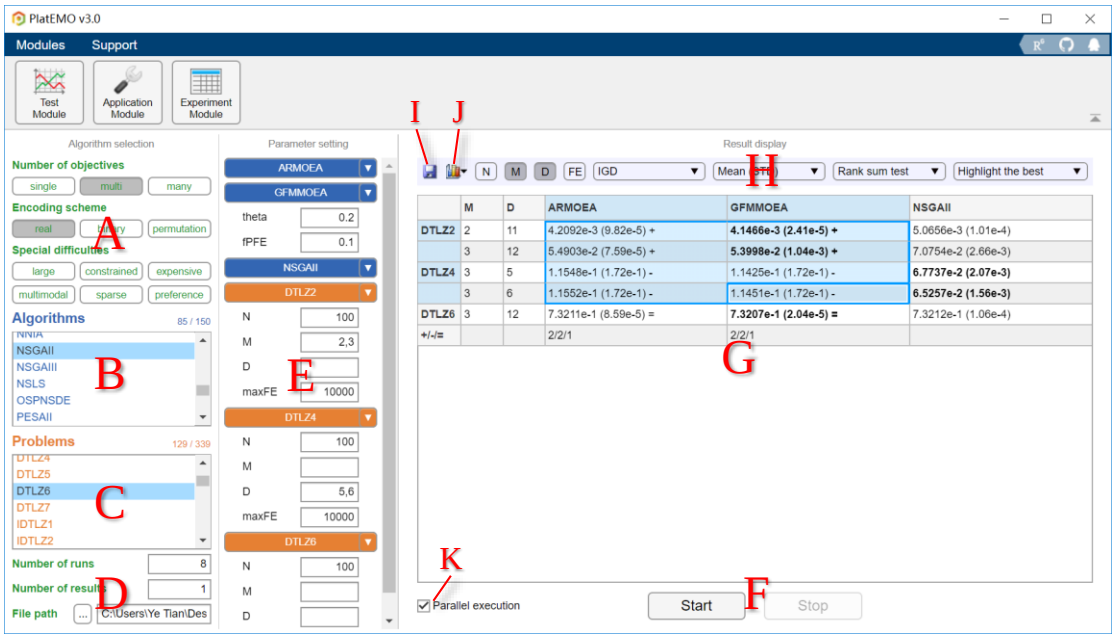
用户可以通过图形界面中的菜单切换至应用模块，它用于求解自定义问题。



在该模块中，用户首先需要在区域 D 中定义问题（参阅求解自定义问题章节），并在区域 E 中更改目标数量、约束数量和验证问题定义的合法性。之后，区域 A 中的问题类型可以被自动确定，用户需要在区域 B 中选择一个算法并在区域 C 中设定相关参数。最后，用户可以在区域 F 中控制算法的运行，并在区域 G 中观察算法运行的实时结果。

3. 实验模块

用户可以通过图形界面中的菜单切换至实验模块，它用于统计分析多个算法在多个问题上的性能。



在该模块中，用户首先需要在区域 A 中选择问题类型（参阅*算法和问题的标签*章节），在区域 B 中选择一个或多个算法，在区域 C 中选择一个或多个测试问题，在区域 D 中设定实验参数，并在区域 E 中设定相关参数；这里的目标数 M 和变量数 D 可以是向量。之后，用户可以在区域 F 中控制实验的运行，并在区域 G 中观察实验运行的实时结果。

需要在表格中显示的统计信息可以在区域 H 中选择。按钮 I 用于将当前表格保存为 Excel、TeX、TXT 或 MAT 文件，按钮 J 用于将所选单元格中的结果显示在一个新窗口中，按钮 K 用于选择实验在单处理器上运行（串行）或多处理器上运行（并行）。

所有结果将会被以 MAT 文件的形式保存在区域 D 中指定的文件夹中。如果存在同名的结果文件，该文件将会被读取以代替算法运行。

4. 算法和问题的标签

每个算法或测试问题需要被添加上标签，这些标签以注释的形式添加在主函数代码的第二行。例如在 PSO.m 代码的开头部分：

```
classdef PSO < ALGORITHM
% <single> <real> <large/none> <constrained/none>
```

通过多个标签指定了该算法可求解的问题类型。所有的标签列举如下：

标签	描述
<single>	单目标优化：问题含有一个目标函数
<multi>	多目标优化：问题含有两或三个目标函数
<many>	超多目标优化：问题含有三个以上目标函数
<real>	连续优化：决策变量为实数或整数
<binary>	二进制优化：决策变量为二进制数
<permutation>	序列优化：决策变量构成一个序列
<large>	大规模优化：问题含有 100 个以上的决策变量
<constrained>	约束优化：问题含有至少一个约束
<expensive>	昂贵优化：目标函数的计算非常耗时，即最大评价次数非常小
<multimodal>	多模优化：存在多个目标值接近但决策向量差异很大的最优解，它们都需要被找到
<sparse>	稀疏优化：最优解中大部分的决策变量均为零
<preference>	偏好优化：仅需寻找前沿面上指定区域的最优解
<none>	空标签

每个算法可能含有多个标签集合，这些集合的笛卡尔积构成该算法可求解的所有的问题类型。例如当标签集合为<single><real><constrained/none>时，表示该算法可求解带或不带约束的单目标连续优化问题；若标签集合为<single><real>，表示该算法只能求解无约束的问题；若标签集合为<single><real><constrained>，表示该算法只能求解带约束的问题；若标签集合为<single><real/binary>，表示该算法可以求解连续或二进制优化问题。

每个算法和测试问题都需要被添加至少一个标签，否则它将不会在图形界面的列表中出现。当用户在图形界面的区域 A 中选择问题类型后，所有可求解该类型问题的算法将会出现在图形界面的区域 B 中，且所有符合该类型的测试问题将会出现在图形界面的区域 C 中。

四　扩展 PlatEMO

1. 算法类

每个算法需要被定义为 ALGORITHM 类的子类并保存在 PlatEMO\Algorithms 文件夹中。算法类包含的属性与方法如下：

属性	赋值方式	描述
parameter	用户	算法的参数
save	用户	每次运行中保存的种群数
outputFcn	用户	在 NotTerminated() 中调用的函数
pro	Solve()	当前运行中求解的问题对象
result	NotTerminated()	当前运行中保存的种群
metric	NotTerminated()	当前保存的种群的指标值
方法	是否可重定义	描述
ALGORITHM	不可	设定由用户指定的属性值
Solve	不可	通过调用 alg.Solve(pro) 来利用算法 alg 求解问题 pro
main	必须	算法的主体部分
NotTerminated	不可	main() 中每次迭代前调用的函数
ParameterSet	不可	根据 parameter 设定算法参数

每个算法需要继承 ALGORITHM 类并重定义方法 main()。例如 GA.m 的代码为：

```
1  classdef GA < ALGORITHM
2  % <single><real/binary/permutation><large/none><constrained/none>
3  % Genetic algorithm
4  % proC ---  1 --- Probability of crossover
5  % disC --- 20 --- Distribution index of crossover
6  % proM ---  1 --- Expectation of the number of mutated variables
7  % disM --- 20 --- Distribution index of mutation
8
9  %------------------------- Reference --------------------------
10 % J. H. Holland, Adaptation in Natural and Artificial Systems,
11 % MIT Press, 1992.
12 %--------------------------------------------------------------
13
```

```
14      methods
15          function main(Alg,Pro)
16              [proC,disC,proM,disM] = Alg.ParameterSet(1,20,1,20);
17              P = Pro.Initialization();
18              while Alg.NotTerminated(P)
19                  P1 = TournamentSelection(2,Pro.N,FitnessSingle(P));
20                  O = OperatorGA(P(P1),{proC,disC,proM,disM});
21                  P = [P,O];
22                  [~,rank] = sort(FitnessSingle(P));
23                  P = P(rank(1:Pro.N));
24              end
25          end
26      end
```

各行代码的功能如下：

第 1 行：　继承 ALGORITHM 类；

第 2 行：　为算法添加标签（参阅*算法和问题的标签*章节）；

第 3 行：　算法的全称；

第 4-7 行：　参数名 --- 默认值 --- 参数描述，将会显示在图形界面的参数设置列表中；

第 9-12 行：算法的参考文献；

第 15 行：　重定义算法主体流程的方法；

第 16 行：　获取用户指定的参数设置，其中 1,20,1,20 分别表示参数 proC,disC,proM,disM 的默认值。

第 17 行：　调用 PROBLEM 类的方法获得一个初始种群；

第 18 行：　保存当前种群并检查评价次数是否超过最大值；若超过则终止算法；

第 19 行：　调用公共函数实现基于二元联赛的交配池选择；

第 20 行：　调用公共函数产生子代种群；

第 21 行：　将父子代种群合并；

第 22 行：　调用公共函数计算种群中解的适应度，并依此对解进行排序；

第 23 行：　保留适应度较好的一半解进入下一代。

在以上代码中，函数 ParameterSet()和 NotTerminated()是 ALGORITHM 类的方法，函数 Initialization()是 PROBLEM 类的方法，而函数 TournamentSelection()、FitnessSingle()和 OperatorGA()是在 PlatEMO\Algorithms\Utility functions 文件夹中的公共函数。所

有可被算法调用的方法及公共函数列举如下，详细的调用方式参阅代码中的注释：

函数名	描述
ALGORITHM.NotTerminated	算法每次迭代前调用的函数
ALGORITHM.ParameterSet	根据用户的输入设定算法参数
PROBLEM.Initialization	初始化一个种群
CrowdingDistance	计算解的拥挤距离（用于多目标优化）
FitnessSingle	计算解的适应度（用于单目标优化）
NDSort	非支配排序
OperatorDE	差分进化算子
OperatorFEP	进化规划算子
OperatorGA	遗传算子
OperatorGAhalf	遗传算子（仅产生前一半的子代）
OperatorPSO	粒子群优化算子
RouletteWheelSelection	轮盘赌选择
TournamentSelection	联赛选择
UniformPoint	产生均匀分布的参考点

2. 问题类

每个问题需要被定义为 PROBLEM 类的子类并保存在 PlatEMO\Problems 文件夹中。问题类包含的属性与方法如下：

属性	赋值方式	描述
N	用户	求解该问题的算法的种群大小
M	用户和Setting()	问题的目标数
D	用户和Setting()	问题的变量数
maxFE	用户	求解该问题可使用的最大评价次数
FE	SOLUTION()	当前运行中已消耗的评价次数
encoding	Setting()	问题的编码方式
lower	Setting()	决策变量的下界
upper	Setting()	决策变量的上界
optimum	GetOptimum()	问题的最优值，例如目标函数的最小值（单目标优化）和前沿面上一组均匀参考点（多目标优化）

PF	GetPF()	问题的前沿面，例如 1 维曲线（双目标优化）、2 维曲面（三目标优化）和可行区域（约束优化）
parameter	用户	问题的参数
方法	是否可重定义	描述
PROBLEM	不可	设定由用户指定的属性值
Setting	必须	设定默认的属性值
Initialization	可以	初始化一个种群
CalDec	可以	修复种群中的无效解
CalObj	必须	计算种群中解的目标值；所有目标函数均为最小化问题
CalCon	可以	计算种群中解的约束违反值；当且仅当约束违反值小于等于零时，约束被满足
GetOptimum	可以	产生问题的最优值并保存在 optimum 中
GetPF	可以	产生问题的前沿面并保存在 PF 中
DrawDec	可以	显示一个种群的决策向量
DrawObj	可以	显示一个种群的目标向量
Current	不可	用来设定或获取当前 PROBLEM 对象的静态方法
ParameterSet	不可	根据 parameter 设定问题参数

每个算法需要继承 PROBLEM 类并重定义方法 Setting()和 CalObj()。例如 SOP_F1.m 的代码为：

```
classdef SOP_F1 < PROBLEM
% <single><real><expensive/none>
% Sphere function

%--------------------------- Reference --------------------------
% X. Yao, Y. Liu, and G. Lin, Evolutionary programming made
% faster, IEEE Transactions on Evolutionary Computation, 1999, 3
% (2): 82-102.
%----------------------------------------------------------------

    methods
        function Setting(obj)
            obj.M = 1;
            if isempty(obj.D); obj.D = 30; end
            obj.lower = zeros(1,obj.D) - 100;
            obj.upper = zeros(1,obj.D) + 100;
            obj.encoding = 'real';
```

```
18          end
19          function PopObj = CalObj(obj,PopDec)
20              PopObj = sum(PopDec.^2,2);
21          end
22      end
```

各行代码的功能如下：

第 1 行：　继承 PROBLEM 类；

第 2 行：　为问题添加标签（参阅算法和问题的标签章节）；

第 3 行：　问题的全称；

第 5-9 行：问题的参考文献；

第 12 行：　重定义设定默认属性值的方法；

第 13 行：　设置问题的目标数；

第 14 行：　设置问题的变量数（若未被用户指定）；

第15-16行：设置决策变量的上下界；

第 17 行：　设置问题的编码方式；

第 19 行：　重定义计算目标函数的方法；

第 20 行：　计算种群中解的目标值。

除以上代码外，默认的方法 Initialization()用于随机初始化一个种群，用户可以重定义该方法来指定特殊的种群初始化策略。例如 Sparse_NN.m 将初始化的种群中随机一半的决策变量置零：

```
function Population = Initialization(obj,N)
    if nargin < 2; N = obj.N; end
    PopDec = (rand(N,obj.D)-0.5)*2.*randi([0 1],N,obj.D);
    Population = SOLUTION(PopDec);
end
```

默认的方法 CalDec()将大于上界的决策变量设为上界值、将小于下界的决策变量设为下界值，用户可以重定义该方法来指定特殊的解修复策略。例如 MOKP.m 修复了超过背包容量限制的解：

```
function PopDec = CalDec(obj,PopDec)
    C = sum(obj.W,2)/2;
    [~,rank] = sort(max(obj.P./obj.W));
    for i = 1 : size(PopDec,1)
        while any(obj.W*PopDec(i,:)'>C)
            k = find(PopDec(i,rank),1);
```

```
            PopDec(i,rank(k)) = 0;
        end
    end
end
```

默认的方法 CalCon()返回零作为解的约束违反值（即解都是满足约束的），用户可以重定义该方法来指定问题的约束。例如 MW1.m 添加了一个约束：

```
function PopCon = CalCon(obj,X)
    PopObj = obj.CalObj(X);
    l = sqrt(2)*PopObj(:,2) - sqrt(2)*PopObj(:,1);
    PopCon = sum(PopObj,2) - 1 - 0.5*sin(2*pi*l).^8;
end
```

用户可以重定义方法 GetOptimum()来指定问题的最优值。例如 SOP_F8.m 指定了目标函数的最小值：

```
function R = GetOptimum(obj,N)
    R = -418.9829*obj.D;
end
```

DTLZ2.m 生成了一组前沿面上均匀分布的参考点：

```
function R = GetOptimum(obj,N)
    R = UniformPoint(N,obj.M);
    R = R./repmat(sqrt(sum(R.^2,2)),1,obj.M);
end
```

用户可以重定义方法 GetPF()来指定多目标优化问题的前沿面或可行区域。例如 DTLZ2.m 生成了 2 维和 3 维的前沿面数据：

```
function R = GetPF(obj)
    if obj.M == 2
        R = obj.GetOptimum(100);
    elseif obj.M == 3
        a = linspace(0,pi/2,10)';
        R = {sin(a)*cos(a'),sin(a)*sin(a'),cos(a)*ones(size(a'))};
    else
        R = [];
    end
end
```

MW1.m 生成了可行区域的数据：

```
function R = GetPF(obj)
```

```
    [x,y] = meshgrid(linspace(0,1,400),linspace(0,1.5,400));
    z = nan(size(x));
    fes = x+y-1-0.5*sin(2*pi*(sqrt(2)*y-sqrt(2)*x)).^8 <= 0;
    z(fes&0.85*x+y>=1) = 0;
    R = {x,y,z};
end
```

默认的方法 DrawDec()显示种群的决策向量（用于图形界面中），用户可以重定义该方法来指定特殊的显示方式。例如 TSP.m 显示了种群中最优解的路径：

```
function DrawDec(obj,P)
    [~,best] = min(P.objs);
    Draw(obj.R(P(best).dec([1:end,1]),:),'-k','LineWidth',1.5);
    Draw(obj.R);
end
```

默认的方法 DrawObj()显示种群的目标向量（用于图形界面中），用户可以重定义该方法来指定特殊的显示方式。例如 Sparse_CD.m 添加了坐标轴的标签：

```
function DrawObj(obj,P)
    Draw(P.objs,{'Kernel k-means','Ratio cut',[]});
end
```

其中 Draw()用于显示数据，它位于 PlatEMO\GUI 文件夹中。以上方法的详细调用方式参阅代码中的注释。

3. 个体类

一个 SOLUTION 类的对象表示一个个体（即一个解），一组 SOLUTION 类的对象表示一个种群。个体类包含的属性与方法如下：

属性	赋值方式	描述
dec	用户	解的决策向量
obj	SOLUTION()	解的目标值
con	SOLUTION()	解的约束违反值
add	adds()	解的额外属性值（例如速度）
方法	描述	
SOLUTION	接受一个或多个解的决策向量并计算相应的目标值与约束违反值；PROBLEM.FE 的值会自动加上生成的 SOLUTION 对象数目	
decs	获取多个解的决策向量构成的矩阵	
objs	获取多个解的目标值构成的矩阵	

cons	获取多个解的约束违反值构成的矩阵
adds	获取多个解的额外属性值构成的矩阵
best	获取种群中可行且最好的解（单目标优化）或可行且非支配的解（多目标优化）

例如，以下代码产生一个具有十个解的种群，并获取其中最好的解的目标值矩阵：

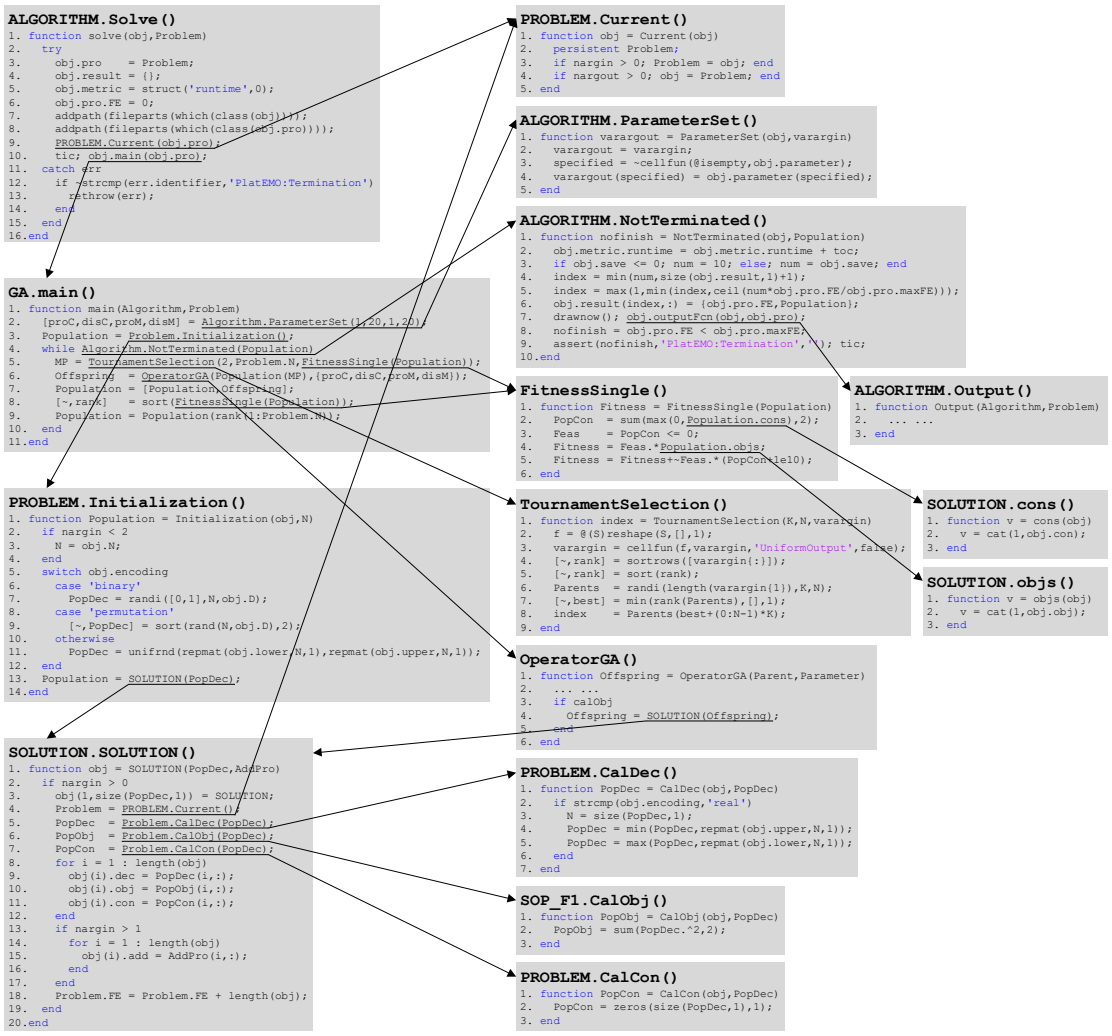
```
Population = SOLUTION(rand(10,5));
BestObjs = Population.best.objs
```

4. 一次完整的运行过程

以下代码利用遗传算法求球面函数的最小值：

```
Alg = GA();
Pro = SOP_F1();
Alg.Solve(Pro);
```

其中代码 `Alg.Solve(Pro)` 执行时所涉及的函数调用过程如下图所示。



5. 指标函数

每个性能指标需要被定义为一个函数并保存在 PlatEMO\Metrics 文件夹中。例如 IGD.m 的代码为：

```
1  function score = IGD(Population,PF)
2  % <min>
3  % Inverted generational distance
4
5  %--------------------------- Reference --------------------------
6  % C. A. Coello Coello and N. C. Cortes, Solving multiobjective
7  % optimization problem using an artificial immune system, Genetic
8  % Programming and Evolvable Machines, 2005, 6(2): 163-190.
9  %----------------------------------------------------------------
10
11     PopObj = Population.best.objs;
12     if size(PopObj,2) ~= size(PF,2)
13       score = nan;
14     else
15       score = mean(min(pdist2(PF,PopObj),[],2));
16     end
17 end
```

各行代码的功能如下：

第 1 行：　函数声明，其中第一个输入为一个种群、第二个输入为问题的最优值、输出为种群的指标值；

第 2 行：　为指标添加标签，其中<min>表示指标值越小越好、<max>表示指标值越大越好；

第 3 行：　指标的全称；

第 5-9 行：　指标的参考文献；

第 11 行：　获取种群中最好的解（可行且非支配的解）的目标值矩阵；

第12-13行：若种群不存在可行解则返回 nan；

第14-15行：否则返回可行且非支配的解的指标值。