Geatpy 总览

1. Geatpy 结构分析

Geatpy 提供面向对象的简单封装的开放式进化算法框架，可以方便、自由地与其他算法或项目相结合。Geatpy 的文件结构如下图所示：

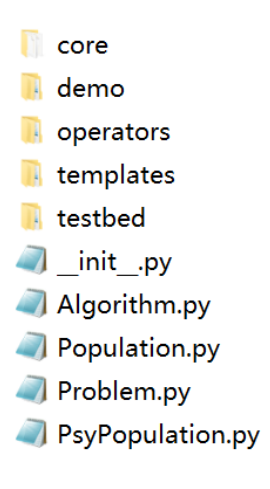

图 1　Geatpy 文件结构一览

其中“core”文件夹里面全部为 Geatpy 工具箱的内核函数；“templates”文件夹存放的是 Geatpy 的进化算法模板；“testbed”是进化算法的测试平台，内含多种单目标优化、多目标优化、组合优化的测试集。“demo”文件夹中包含了应用 Geatpy 工具箱求解问题的案例。“operators”是 2.2.2 版本之后新增的，里面存放着面向对象的重组和变异算子类，这些重组和变异算子类通过调用“core”文件夹下的重组和变异算子来执行相关的操作。

Geatpy 的面向对象进化算法框架有四个大类：Algorithm(算法模板顶级父类)、Population(种群类)、PsyPopulation(多染色体种群类) 和 Problem(问题类)，分别存放在“Algorithm.py”、“Population.py”、“Problem.py”文件中。其 UML 类图如下所示：

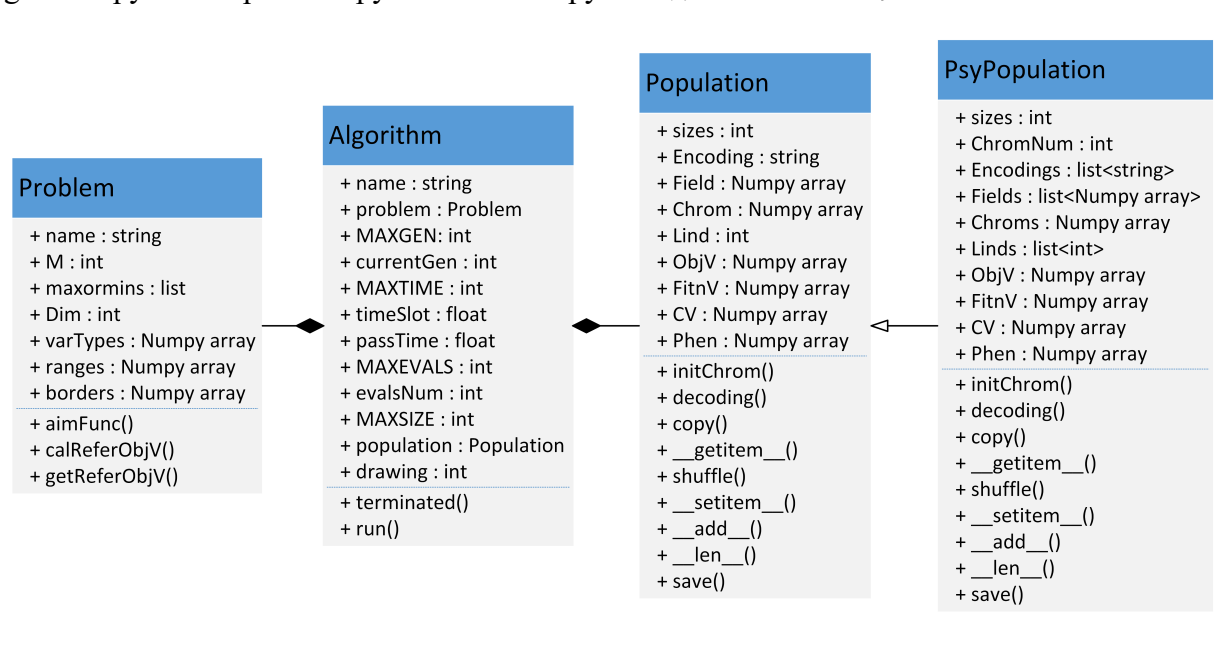

图 2　Geatpy UML 类图

Problem 类定义了与问题相关的一些信息，如问题名称 name、优化目标的维数 M、决策变量的个数 Dim、决策变量的范围 ranges、决策变量的边界 borders 等。maxormins 是一个记录着各个目标函数是最小化抑或是最大化的 list 类型列表，其中元素为 1 表示对应的目标是最小化目标；为-1 表示对应的是最大化目标。例如 M=3，maxormins=[1,-1,1]，此时表示有三个优化目标，其中第一、第三个是最小化目标，第二个是最大化目标。varTypes 是一个记录着决策变量类型的行向量，其中的元素为 0 表示对应的决策变量是连续型变量；为 1 表示对应的是离散型变量。待求解的目标函数定义在 aimFunc() 的函数中。calReferObjV() 函数则用于计算或读取目标函数参考值（一般用理论上的目标函数的最优值作为参考值），该参考值可以用于后续的指标分析。在实际使用时，不是直接在 Problem 类的文件中修改相关代码来使用的，而是通过定义一个继承 Problem 的子类来完成对问题的定义的。这些在后面的章节中会详细讲述。getReferObjV() 是 Problem 父类中已经实现了的一个函数，它先尝试读取特定文件夹中的目标函数值参考数据，如果读取不到，则调用 calReferObjV() 进行计算。对于 Problem 类中各属性的详细含义可查看 Problem.py 源码。

Population 类是一个表示种群的类。一个种群包含很多个个体，而每个个体都有一条染色体 (若要用多染色体，则使用 PsyPopulation 类)。除了染色体外，每个个体都有一个译码矩阵 Field(或俗称区域描述器) 来标识染色体应该如何解码得到表现型，同时也有其对应的目标函数值以及适应度。种群类就是一个把所有个体的这些数据统一存储起来的一个类。比如里面的 Chrom 是一个存储种群所有个体染色体的矩阵，它的每一行对应一个个体的染色体；ObjV 是一个目标函数值矩阵，每一行对应一个个体的所有目标函数值，每一列对应一个目标。对于 Population 类中各属性的详细含义可查看 Population.py 源码以及下一章“Geatpy 数据结构”。

PsyPopulation 类是继承了 Population 的支持多染色体混合编码的种群类。一个种群包含很多个个体，而每个个体都有多条染色体。用 Chroms 列表存储所有的染色体矩阵 (Chrom)；Encodings 列表存储各染色体对应的编码方式 (Encoding)；Fields 列表存储各染色体对应的译码矩阵 (Field)。

Algorithm 类是进化算法的核心类。它既存储着跟进化算法相关的一些参数，同时也在其继承类中实现具体的进化算法。比如 Geatpy 中的 moea_NSGA3_templet.py 是实现了多目标优化 NSGA-III 算法的进化算法模板类，它是继承了 Algorithm 类的具体算法的模板类。关于 Algorithm 类中各属性的含义可以查看 Algorithm.py 源码。这些算法模板通过调用 Geatpy 工具箱提供的进化算法库函数实现对种群的进化操作，同时记录进化过程中的相关信息，其基本层次结构如下图：

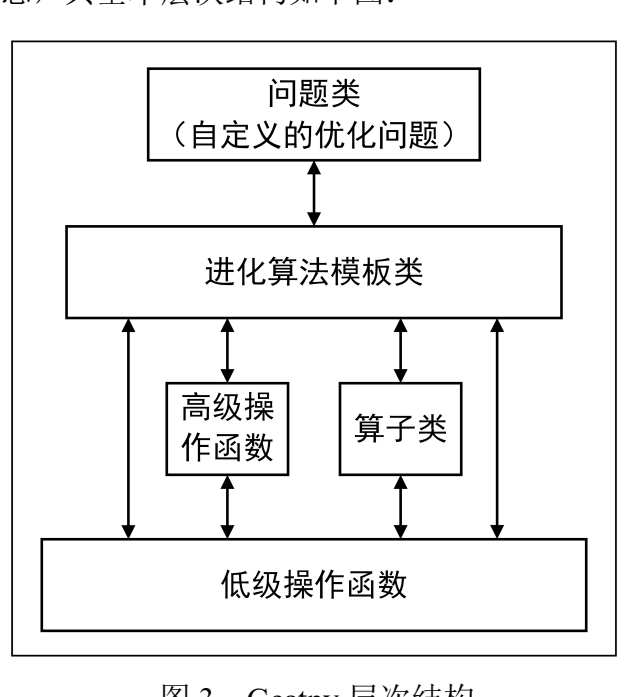

图 3　Geatpy 层次结构

其中“算子类”是 2.2.2 版本之后增加的新特性，通过实例化算子类来调用低级操作函数，可以利用面向对象编程的优势使得对传入低级操作函数的一些参数的修改变得更加方便。目前内置的“算子类”有“重组算子类”和“变异算子类”。用户可以绕开对底层的低级操作函数的修改而直接通过新增“算子类”来实现新的算子例如自适应的重组、变异算子等。

对于未接触过面向对象编程的用户而言，可以不采用上面所说的 Geatpy 提供的面向对象进化算法框架，可以通过直接调用工具箱的库函数来解决实际问题 (详见第三章“快速入门”)。这样做的不好之处是需要比较熟悉工具箱的内核用法。

Geatpy 工具箱提供了大量的跟进化算法有关的内核函数，涵盖了单目标优化、多目标优化、组合优化等方面的众多算子，同时提供指标评价、绘图等功能性函数。下图展示了 Geatpy 的函数调用关系。中心是进化算法模板，它调用高级的运算函数 (selecting, recombin, mutate) 来实现进化优化。高级函数进一步调用相关的低级运算函数 (即实现选择、交叉、变异等底层算法的函数)。这种层级调用关系使 Geatpy 的结构十分清晰，更重要的是，你可以自定义更多低级运算函数来轻松自由地扩展 Geatpy。

特别注意：自定义的进化算法模板最好不要与 Geatpy 自带的模板名称重复，否则会出现一些难以解决的冲突问题。

2. Geatpy 调用关系

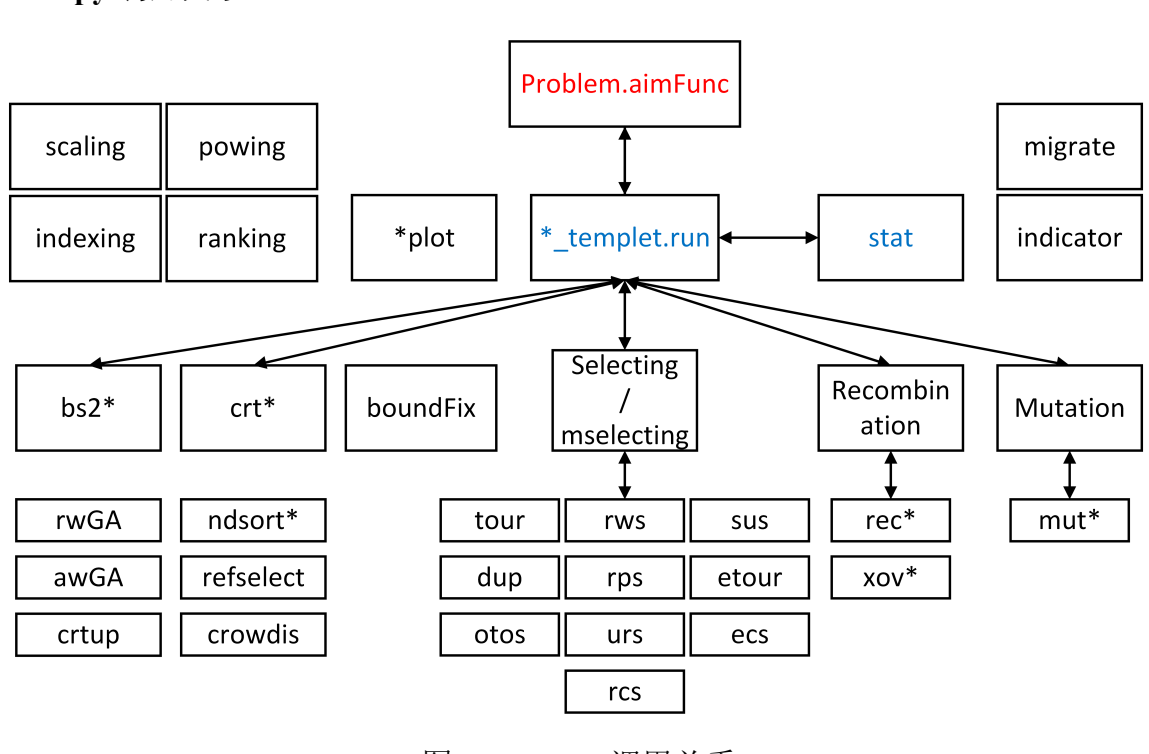

图 4　Geatpy 调用关系

图中黑色文字的方框 Geatpy 内置的函数 (可以添加自定义的函数)。其中，crt* 用于创建种群染色体矩阵 (* 表示名字是多样的)；bs2* 用于二进制/格雷码种群染色体矩阵的解码；ranking, powing, scaling, indexing 均是用于计算单目标优化种群个体适应度的函数；selecting、recombin、mutate 分别是高级的选择、重组和变异函数，对应下方的都是低级的选择、重组和变异函数；*plot 是以 plot 名字结尾的用于可视化输出函数。ndsort* 是与多目标优化的非支配排序函数。rwGA、awGA、refselect、crowdis 均是多目标进化优化算法中需要用到的函数，功能分别为随机权重目标聚合、自适应权重目标聚合、基于参考点关联的个体选择、目标空间拥挤距离的计算。indicator 是一个评价指标模块，里面包含众多评价指标的计算，如 GD、IGD、HV、Spacing 等。

红色文字的方框表示 Problem 类中的自定义目标函数，假设 myProblem 为一个继承了 Problem 类的自定义问题类的对象，则通过 myProblem.aimFunc() 即可调用自定义的目标函数。

蓝色文字的方框表示 Algorithm 类的相关方法，run 是算法模板的执行方法，stat 是 Algorithm 类中的用于统计和分析处理的函数。

这些函数所用的数据都统一遵循 Geatpy 的数据结构规范 (详见下一章节“Geatpy 数据结构”)。

3. Geatpy 核心模块一览

以下是 Geatpy 的核心模块，其详细用法可以通过 help() 命令查看或翻阅 Geatpy 的 API 文档。其中的算法模板等模块的源码均有详细的注释。例如：

```python
import geatpy as ea
help(ea.crtfld)
```

命名规范：并不遵循 Python 推荐的命名规范，而是采用与知名的 Matlab 进化算法工具箱先驱者“GEATbx”类似的命名规范，即采用英文短语缩写的方式，例如：“crtfld”意为：Create Field.

3.1 与初始化种群染色体相关的函数

- crtfld (生成译码矩阵，俗称“区域描述器”)
- crtbp (创建二进制种群染色体矩阵)
- crtip (创建元素是整数的种群染色体矩阵)
- crtpp (创建排列编码种群染色体矩阵)
- crtrp (创建元素是实数的种群染色体矩阵)
- meshrng (网格化决策变量范围)

3.2 进化迭代

当完成了种群染色体的初始化后，就可以进行进化迭代了。这部分是在进化算法模板里调用。迭代过程中包括：

- 调用 ranking 或 scaling 等计算种群适应度。
- 调用 selecting 进行选择操作 (也可以直接调用低级选择函数)。
- 调用 recombin 进行重组操作 (也可以直接调用低级重组函数)。
- 调用 mutate 进行重组操作 (也可以直接调用低级变异函数)。

3.3 适应度计算

- ranking (基于等级划分的适应度分配计算)
- scaling (线性尺度变换适应度计算)
- indexing (指数尺度变换适应度计算)
- powing (幂尺度变换适应度计算)

对于多目标进化优化，由于各种多目标优化算法所采用的适应度计算方法门类有很多，因此此时的适应度计算交由继承了 Algorithm 的具体算法模板类中实现，详见相关源码。

3.4 选择

selecting 和 mselecting 是高级选择函数，它们调用下面的低级选择函数。其中 selecting 是用于单一种群的个体选择，而 mselecting 是用于多种群个体的选择。

- dup (Duplication，基于适应度排序的直接复制选择)
- ecs (Elite Copy Selection，精英复制选择)
- etour (精英保留锦标赛选择)
- otos (One-to-One Survivor Selection，一对一生存者选择)
- rcs (Random Compensation Selection，随机补偿选择)
- rps (Random Permutation Selection，随机排列选择)
- rws (Roulette Wheel Selection，轮盘赌选择)
- sus (Stochastic Universal Sampling，随机抽样选择)
- tour (Tournament，锦标赛选择)
- urs (Uncommitted Random Selection，无约束随机选择)

3.5 重组

重组包括了交叉。recombin 是高级的重组函数，它调用下面的低级重组函数：

- recdis (离散重组)
- recint (中间重组)
- reclin (线性重组)
- recndx (正态分布交叉)
- recsbx (模拟二进制交叉)
- xovbd (二项式分布交叉)
- xovdp (两点交叉)
- xovexp (指数交叉)
- xovmp (多点交叉)
- xovox (顺序交叉)
- xovpmx (部分匹配交叉)
- xovsec (洗牌指数交叉)
- xovsh (洗牌交叉)
- xovsp (单点交叉)
- xovud (均匀分布交叉)

注意：所有重组算子都不会检查重组结果是否满足所设边界范围。下面的变异算子则是内置对边界范围的检查和修复。因此如果在进化算法中要单是使用重组算子，则需要调用“ea.boundfix”函数进行边界修复。详见“help(ea.boundfix)”。

在重组过程中，种群的前一半个体会与后一半个体的染色体按照个体顺序进行一一配对。这些重组算子可通过设置传入参数“Half”的值为 True，来使得重组后只保留一半的个体，此时将保留上面所说的一一配对重组后的第一条染色体。

Geatpy2.2.2 版本之后在进化算法框架中新增了面向对象的 Recombination(重组算子接口)，上述的低级重组算子均有与之对应的重组算子类来进行参数的设置及调用，这些新增的类命名为首字母大写的对应低级重组算子的名称：

- Recdis (离散重组算子类)
- Recint (中间重组算子类)
- Reclin (线性重组算子类)
- Recndx (正态分布交叉算子类)
- Recsbx (模拟二进制交叉算子类)
- Xovbd (二项式分布交叉算子类)
- Xovdp (两点交叉算子类)
- Xovexp (指数交叉算子类)
- Xovmp (多点交叉算子类)
- Xovox (顺序交叉算子类)
- Xovpmx (部分匹配交叉算子类)
- Xovsec (洗牌指数交叉算子类)
- Xovsh (洗牌交叉算子类)
- Xovsp (单点交叉算子类)
- Xovud (均匀分布交叉算子类)

在调用某个重组算子时，可以直接调用低级重组算子进行重组；也可以利用高级重组算子 recombine 通过指定低级重组算子的名称来调用低级重组算子进行重组，如 recombine(’xovdp’, ...)；也可以通过实例化重组算子类的对象，然后执行该对象的 do() 函数进行重组，例如：recOper = Xovdp(...)，recOper.do(...)。具体结构详见这些类的源码。

3.6 突变

mutate 是高级的突变函数，它调用下面的低级突变函数：

- mutbga (Mutation for Breeder Genetic Algorithm，Breeder GA 算法突变算子)
- mutbin (Mutation for Binary Chromosomes，二进制变异算子)
- mutde (Mutation for Differential Evolution，差分变异算子)
- mutgau (Gaussian Mutation，高斯突变算子)
- mutinv (Invertion Mutation，染色体片段逆转变异算子)
- mutmove (Mutation by Moving，染色体片段移位变异算子)
- mutpolyn (Polynomial Mutation，多项式变异)
- mutpp (Mutation of Permutation Chromosomes，排列编码变异算子)
- mutswap (Two Point Swapping Mutation，染色体两点互换变异算子)
- mutuni (Uniform Mutation，均匀变异算子)

注意：对于 mutbga、mutde、mutgau、mutpolyn、mutuni，变异是先按实数值来变异，然后对于标记了是离散型变量采用四舍五入等方法转化为整数。因此结果往往会是浮点“float”类型的，此时如果要把这些离散值用作其他变量的下标，需要对其进行强制类型转换。

Geatpy2.2.2 版本之后在进化算法框架中新增了面向对象的 Mutation(变异算子算子接口)，上述的低级变异算子均有与之对应的变异算子类来进行参数的设置及调用，这些新增的类命名为首字母大写的对应低级变异算子的名称：

- Mutbga (Mutation for Breeder Genetic Algorithm，Breeder GA 算法突变算子类)
- Mutbin (Mutation for Binary Chromosomes，二进制变异算子类)
- Mutde (Mutation for Differential Evolution，差分变异算子类)
- Mutgau (Gaussian Mutation，高斯突变算子类)
- Mutinv (Invertion Mutation，染色体片段逆转变异算子类)
- Mutmove (Mutation by Moving，染色体片段移位变异算子类)
- Mutpolyn (Polynomial Mutation，多项式变异类)
- Mutpp (Mutation of Permutation Chromosomes，排列编码变异算子类)
- Mutswap (Two Point Swapping Mutation，染色体两点互换变异算子类)
- Mutuni (Uniform Mutation，均匀变异算子类)

在调用某个变异算子时，可以直接调用低级变异算子进行重组；也可以利用高级变异算子 mutate 通过指定低级变异算子的名称来调用低级变异算子进行重组，如 mutate(’mutgau’, ...)；也可以通过实例化变异算子类的对象，然后执行该对象的 do() 函数进行变异，例如：mutOper = Mutgau(...)，mutOper.do(...)。具体结构详见这些类的源码。

3.7 染色体解码

对于二进制/格雷编码的种群，我们要对其进行解码才能得到其表现型。

- bs2int (二进制/格雷码转整数)
- bs2real (二进制/格雷码转实数)
- bs2ri (二进制/格雷码转实整数)

3.8 可视化

- moeaplot (多目标优化目标空间绘图函数)
- soeaplot (单目标优化绘图函数)
- trcplot (进化记录器绘图函数)
- varplot (决策变量绘图函数)

如果是在 iPython 控制台中调用可视化绘图函数（例如使用 Spyder 开发工具），一般图像会默认显示在控制台中。此时可以在控制台下执行%matplotlib 来设置把图像显示在一个独立窗口中。

3.9 多目标相关

- awGA (适应性权重法多目标聚合函数)
- rwGA (随机权重法多目标聚合函数)
- ndsortDED (基于排除法的帕累托层级划分)
- ndsortESS (基于 ENS_SS 的快速非支配层级划分)
- ndsortTNS (基于 T_ENS 的快速非支配层级划分)
- crtgp (创建在单位超空间中均匀的网格点集)
- crtup (创建在单位超平面内均匀分布的点集)
- crowdis (拥挤距离计算)
- refgselect (利用参考点引导的个体选择)
- refselect (基于参考点的“入龛”个体选择)

3.10 多种群相关

- migrate (种群间个体迁移)
- mselecting (多种群个体选择)

在面向对象的进化算法框架中对应 migrate 有一个 Migrate 类，它通过输入种群列表，然后调用底层的 migrate 函数，最后进行一些后续处理并返回迁移后的种群列表，使得种群迁移的功能更加容易被使用。

3.11 内置的进化算法模板类

下面是内置已实现的进化算法模板，其中带“psy”字样的是多染色体版本，带“multi”字样的是多种群（单一染色体）版本。

- soea_DE_best_1_bin_templet (差分进化 DE/best/1/bin 算法模板)
- soea_DE_best_1_L_templet (差分进化 DE/best/1/L 算法模板)
- soea_DE_rand_1_bin_templet (差分进化 DE/rand/1/bin 算法模板)
- soea_DE_rand_1_L_templet (差分进化 DE/rand/1/L 算法模板)
- soea_DE_targetToBest_1_bin_templet (差分进化 DE/targetToBest/1/bin 算法模板)
- soea_DE_targetToBest_1_L_templet (差分进化 DE/targetToBest/1/L 算法模板)
- soea_ES_1_plus_1_templet ((1+1) 进化策略模板)
- soea_EGA_templet (精英保留的遗传算法模板)
- soea_SEGA_templet (增强精英保留的遗传算法模板)
- soea_SGA_templet (最简单、最经典的遗传算法模板)
- soea_GGAP_SGA_templet (带代沟的简单遗传算法模板)
- soea_studGA_templet (种马遗传算法模板)
- soea_steady_GA_templet (稳态遗传算法模板)
- soea_psy_EGA_templet (精英保留的多染色体遗传算法模板)
- soea_psy_SEGA_templet (增强精英保留的多染色体遗传算法模板)
- soea_psy_SGA_templet (最简单、最经典的多染色体遗传算法模板)
- soea_psy_GGAP_SGA_templet (带代沟的多染色体简单遗传算法模板)
- soea_psy_studGA_templet (多染色体种马遗传算法模板)
- soea_psy_steady_GA_templet (多染色体稳态遗传算法模板)
- soea_multi_SEGA_templet (增强精英保留的多种群协同遗传算法模板)
- moea_awGA_templet (基于 awGA 算法的多目标进化算法模板)
- moea_NSGA2_DE_templet (基于 NSGA-II-DE 算法的多目标进化算法模板)
- moea_NSGA2_archive_templet (带全局存档的多目标进化 NSGA-II 算法模板)
- moea_NSGA2_templet (基于 NSGA-II 算法的多目标进化算法模板)
- moea_NSGA3_DE_templet (基于 NSGA-III-DE 算法的多目标进化算法模板)
- moea_NSGA3_templet (基于 NSGA-III 算法的多目标进化算法模板)
- moea_RVEA_templet (基于 RVEA 算法的多目标进化算法模板)
- moea_RVEA_RES_templet (基于带参考点再生策略的 RVEA 算法的多目标进化算法模板)
- moea_psy_awGA_templet (基于 awGA 算法的多染色体多目标进化算法模板)
- moea_psy_NSGA2_archive_templet (带全局存档的多染色体多目标进化 NSGA-II 算法模板)
- moea_psy_NSGA2_templet (基于 NSGA-II 算法的多染色体多目标进化算法模板)
- moea_psy_NSGA3_templet (基于 NSGA-III 算法的多染色体多目标进化算法模板)
- moea_psy_RVEA_templet (基于 RVEA 算法的多染色体多目标进化算法模板)
- moea_psy_RVEA_RES_templet (基于带参考点再生策略的多染色体 RVEA 算法的多目标进化算法模板)

3.12 指标评价

评价指标计算函数基本上集中在 indicator 模块中，主要有以下几个评价指标：

- indicator.GD (多目标优化世代距离的计算)
- indicator.IGD (多目标优化反转世代距离的计算)
- indicator.HV (多目标优化超体积指标的计算)
- indicator.Spacing (多目标优化分布性指标 Spacing 的计算)
- indicator.moea_tracking (多目标优化进化过程指标追踪分析)

其中的 moea_tracking() 函数可用于多目标优化进化过程的指标变化追踪分析，如果在进化过程中把历代种群对象保存到了 pop_trace 中，那么就可以利用该函数来计算各代种群的指标，然后调用 trcplot() 函数绘制指标追踪分析图。