Pyrimidine：基于面向对象的遗传算法 Python 框架

宋丛威
（浙江工业大学之江学院理学院数学系，浙江 绍兴 312030）

摘　要：Pyrimidine 是开发用于实现遗传算法的通用框架。它也可以实现任何迭代模型，如模拟退火、粒子群算法。它的设计基于面向对象，利用了 Python 的元编程功能把群体看成个体的容器，又把个体看成染色体的容器，而容器是一个元类，用来构造不同结构的个体类和群体类，因为它是高度面向对象的，所以它容易扩展、改进。

关键词：Pyrimidine 框架；遗传算法；Python 语言；面向对象编程；元编程

1　概述

遗传算法是一种通用优化方法。遗传算法模仿进化论中自然选择来解决优化问题。它是最早被开发出来的智能算法[1-4]，已经在多个领域得到广泛使用，并被改造成各种变体，并和新的算法相结合[5-6]。在此不回顾它的原理，关于更多内容请参考文献 [4] 以及其中罗列的文献。

目前有多种语言提供了实现遗传算法框架的库。这其中 Python 就提供了数个遗传算法框架库，如知名的 deap[7]，gaft，适合于机器学习参数优化的 tpot[8,9]，作为 scikit-learn[10]扩展的 scikit-opt 和 [gplearn]（https://github.com/trevorstephens/gplearn）等。介绍由设计的通用算法框架 Pyrimidine。它比其他库更加严格遵循面向对象编程模式，同时利用了 Python 的元编程功能。

2　框架设计

框架设计主要有两部分构成：实现遗传算法的基本概念的类，如个体，群体；用于构造这些类的元类。

在用 Python 具体实现时，容器就是对象列表（或者其他可行的迭代器）。如群体是个体列表；个体是染色体列表；染色体则是基因的数组。基因数组的具体实现是比较关键的。可以采用标准库 array，也可以采用著名的第三方数值计算库 numpy。后者在各种数值计算中比较方便，但是交叉操作比较慢。

2.1　容器元类

这个元类参考了函数式编程语言 Haskell 和类型理论[10]。容器主要有两种：列表和元组，其中列表表示容器中的元素有相同类型，元组则无此限制。不介意用户修改元类，因此不详细介绍元类内容。

下文的基类都是由元类构造的。利用元类提供的方法，构造一个由若干染色体 ExampleChromosome 组成的个体 ExampleIndividual，只需这样定义

```
ExampleIndividual =BaseIndividual [ExampleChromosome],
```

其中 BaseIndividual 是所有个体类的基类，这种写法受变量类型的影响，比如字符串列表，写作 List [String]。

2.2　基本类

Pyrimidine 中存在 3 个最基本的类。BaseChromosome, BaseIndividual, BasePopulation, 分别表示染色体、个体、种群。上文已经说明，个体是允许有多个染色体的，这和一般的遗传算法的设计不一样。设计算法时的一般顺序是，继承 BaseChromosome，构造用户的染色体类，然后继承 BaseIndividual，构造用于的个体类，最后用相同的方法构造种群类。为了方便用户，Pyrimidine 提供了一些常用子类，无需用户重复设置。继承了这些类，同时就拥有了，解的编码方案，染色体的杂交和变异方法。遗传算法一般采用二进制编码。Pyrimidine 提供了 BinaryChromosome 用于二进制情形。继承此类，就拥有了二进制编码，以及两点杂交，每位基因独立变异的算法组件。

一般，用户会从如下的子类构造开始算法设计，其中 MonoIndividual 强制个体只能有一条染色体（对算法来说，强制不是必须的，也可改用 BaseIndividual）。

```
class MyIndividual(MonoIndividual):
    element_class = BinaryChromosome
    def _fitness(self):
        # 此处写适应值的计算过程
```

MyIndividual 是一个以 BinaryChromosome 为染色体

作者简介：宋丛威（1986-），男，博士，讲师，研究方向：小波分析、调和分析、机器学习。

的个体类。因为 MonoBinaryIndividual 就是这样的类，所以等价的写法是，

```
class MyIndividual(MonoBinaryIndividual):
    def _fitness(self):
        ...
```

采用标准遗传算法作为迭代方法，直接令 MyPopulation=SGAPopulation [MyIndividual]。这种写法受 Python 的 typing 模块（以及 Haskell 的类型概念）启发，由元编程实现，表示 MyPopulation 是 MyIndividual 对象构成的列表。

如果要重新设计交叉和变异算法，那么重写个体类的 mutate 和 cross 方法，也可以重写染色体类的方法，因为个体类对象的遗传操作方法都是简单调用每个染色体类对象的方法。

这些类同时继承了基类 BaseIterativeModel，用来规范迭代格式，包括导出用于可视化的数据。开发新算法，关键是重载方法 transit，所有迭代算法都是在重复调用这个方法。对于遗传算法来说，transit 主要就是相继执行各种遗传操作。

3 实例

举一个简单的例子说明 Pyrimidine 的基本用法：n=50 维经典背包问题：

$$\max \sum_i c_i x_i,$$

$$\sum_i w_i x_i \le W, x_i = 0,1.$$

这个问题直接用二进编码，无需解码，非常适合于测试遗传算法。该问题已定义在子模块 pyrimidine.benchmarks.optimization 下，其中 Knapsack.random 会随机生成适合的参数 c_i, w_i, W。用户可以通过重写 _fitness 方法替换成自己的优化问题。

3.1 算法构造

利用 Pyrimidine 提供的类，可以非常方便构造一个含 20 个个体，其中每个个体只含一个 50 维染色体。种群将迭代 100 次，最后一代的最优个体便是优化问题的解。

```
from pyrimid ine import MonoBinaryIndividual, SGAPopulation
from pyrimidine.benchmarks.optimization import *
n=50
_evaluate = Knapsack.random(n) # 将 n 维二进编
# 码映射为目标函数值
class MyIndividual(MonoBinaryIndividual):
    def _fitness(self):
        # 返回值必须是一个数
        return _evaluate(self.chromosome)
class MyPopulation(SGAPopulation):
    element_class = MyIndividual
    default_size = 20
pop = MyPopulation.random(size=n)  # size: 染色体长度
pop.evolve(n_iter=100)
```

最后，通过 pop.best_individual 找出最优个体作为解。设置 verbose=True 可打印迭代过程。下面是等价写法。

```
MyPopulation = SGAPopulation[MonoBinaryIndividual.set_fitness(lambda o: _evaluate(o.chromosome))]
pop = MyPopulation.random (n_individuals =20, size=n)
pop.evolve()
```

等价写法不再显式依赖类的继承和 class 语法，使得程序更加“代数化”，也变得更简短。

3.2 可视化

为了考察算法性能，通常要绘制适应值曲线，或者其他量值的迭代序列，可改用方法 history。该方法能返回一个 pandas.DataFrame 对象，记录了关于每一代种群的统计结果。用户可以用它来自由绘制性能曲线。一般用户要提供一个“统计量字典”：键是统计量的名称，值是从种群到数值的函数（字符串只限于已经定义好的种群方法或属性，而且以数量为返回值）。参看下述代码。

```
stat ={'Mean Fitness':'mean_fitness', 'Best Fitness':'best_fitness'}
data = pop.history(stat=stat, n_iter=100)
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111)
data[['Mean Fitness', 'Best Fitness']].plot(ax=ax)
ax.set_xlabel('Generations')
ax.set_ylabel('Fitness')
plt.show()
```

这里字符串 mean_fitness，best_fitness 分别表示种群的平均适应值和最优个体适应值。当然，它们最终都是由已经定义好的对象方法来实现统计功能的，比如 best_fitness 对应的是映射 pop→pop.best_individual.fitness。

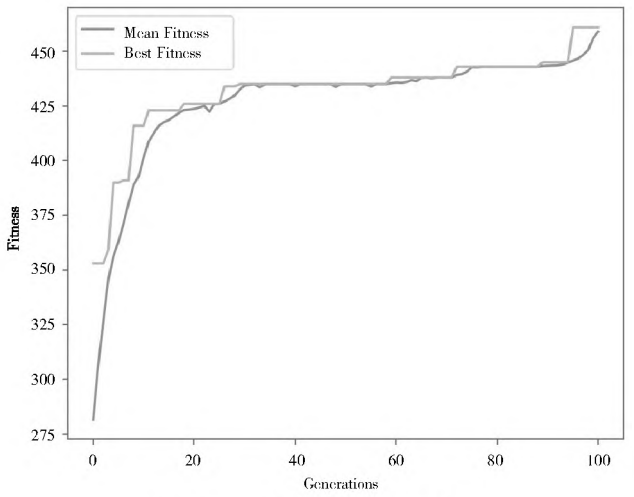

图 1　生成的适应值曲线

3.3　算法扩展

用 Pyrimidine 开发新的算法非常简单。在经典遗传算法中，整个种群的变异率和交叉率是一致的，但 Pyrimidine 可以轻易将它们编码到每个个体中，使它们随着迭代而发生变化。

```
    class NewIndividual(MixIndividual):
            element_class = (BinaryChromosome,
FloatChromosome)
        def mutate(self):
            # 根据第二个染色体第一位 self[1][0]变异
        def cross(self, other):
            # 根据第二个染色体第二位 self[1][1]交叉
        def _fitness(self):
            # 适应值只和第一个染色体有关
            f(self[0])
    class NewPopulation(SGAPopulation):
        element_class = NewIndividual
        default_size = 20
    # 8 表示变量的编码长度,2 代表变异率和交叉率
    pop = NewPopulation.random(sizes=(8, 2))
```

FloatChromosome 已经配备了用于浮点数的遗传操作，无需用户定义。这样就可开发出一种变异率和交叉率具有演化特点的遗传算法。

4　结语

进行大量的实验和改进，证明 Pyrimidine 是一个可用于实现多种遗传算法的通用框架。比起其他框架，它的设计具有很强的可扩展性，可以实现任何迭代模型，如模拟退火，粒子群算法。如果用户开发新算法，那么 Pyrimidine 会是一个不错的选择。

目前，Pyrimidine 还在开发中，但是大部分 API 已经固定，用户不用担心变动。Pyrimidine 要求适应值是一个数，因此还不能解决多目标问题，除非把它们归结为单目标问题。但已经考虑实现多目标优化。Pyrimidine 采用了 numpy 的数值类，交叉操作比 deap 慢，但不难改用其他实现方案。当然，还有其他需要改进的地方。此外，Pyrimidine 的文档还不够丰富，正在制作中。

参考文献

[1] Holland, J. Adaptation in Natural and Artificial Systems [M] . The Univ. of Michigan, 1975.

[2] 汪定伟，王俊伟，王洪峰，张瑞友，郭哲．智能优化方法 [M] ．北京：高等教育出版社，2007.

[3] D. Simon. 进化优化算法——基于仿生和种群的计算机智能方法 [M] ．北京：清华大学出版社，2018.

[4] 玄光男，程润伟．遗传算法与工程优化 [M] ．北京：清华大学出版社，2004.

[5] 叶志伟，王明威，王春枝．自然计算及其图像处理与分析应用 [M] ．北京：中国水利水电出版社，2019.

[6] 李士勇，李妍，林永茂．智能优化算法与涌现计算 [M] ．北京：清华大学出版社，2019.

[7] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau and Christian Gagné, DEAP: Evolutionary Algorithms Made Easy [J] . Journal of Machine Learning Research, 2012, 13: 2171-2175.

[8] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. Automating biomedical data science through tree-based pipeline optimization [J] . Applications of Evolutionary Computation, 2016: 123-137.

[9] Trang T. Le, Weixuan Fu and Jason H. Moore. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. Bioinformatics [J] . 2020, 36 (1) : 250-256.

[10] 韩冬．魔力 Haskell [M] ．北京：人民邮电出版社，2016.