soea\_EGA\_templet.py - Elitist Reservation GA templet(精英保留的遗传算法模板)

算法描述:

本模板实现的是基于杰出保留的单目标遗传算法。算法流程如下：

1) 根据编码规则初始化N个体的种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 独立地从当前种群中选取N-1个母体。

5) 独立地对这N-1个母体进行交叉操作。

6) 独立地对这N-1个交叉后的个体进行变异。

7) 计算当代种群的最优个体，并把它插入到这N-1个交叉后的个体的第一位，得到新一代种群。

8) 回到第2步。

self.selFunc = 'tour' # 锦标赛选择算子

soea\_SEGA\_templet : class - Strengthen Elitist GA templet(增强精英保留的遗传算法模板)

算法描述:

本模板实现的是增强精英保留的遗传算法。算法流程如下：

1) 根据编码规则初始化N个体的种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 独立地从当前种群中选取N个母体。

5) 独立地对这N个母体进行交叉操作。

6) 独立地对这N个交叉后的个体进行变异。

7) 将父代种群和交叉变异得到的种群进行合并，得到规模为2N的种群。

8) 从合并的种群中根据选择算法选择出N个个体，得到新一代种群。

9) 回到第2步。

该算法宜设置较大的交叉和变异概率，否则生成的新一代种群中会有越来越多的重复个体。

self.selFunc = 'tour' # 锦标赛选择算子

soea\_SGA\_templet : class - Simple GA templet(最简单、最经典的遗传算法模板)

算法描述:

本模板实现的是最经典的单目标遗传算法。算法流程如下：

1) 根据编码规则初始化N个体的种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 独立地从当前种群中选取N个母体。

5) 独立地对这N个母体进行交叉操作。

6) 独立地对这N个交叉后的个体进行变异，得到下一代种群。

7) 回到第2步。

self.selFunc = 'rws' # 轮盘赌选择算子

soea\_steadyGA\_templet : class - Steady State GA templet(稳态遗传算法模板)

算法描述:

本模板实现的是稳态遗传算法，算法流程如下：

1) 根据编码规则初始化N个体的种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 独立地从当前种群中选取2个母体。

5) 独立地对这2个母体进行交叉操作。

6) 独立地对这2个交叉后的个体进行变异。

7) 将这2个母体和由交叉变异得到的个体进行一对一生存者竞争选择。

8) 将第7步得到的个体替换父代中原母体所在位置的个体，形成新一代种群。

9) 回到第2步。

self.selFunc = 'etour' # 精英保留锦标赛选择算子

soea\_studGA\_templet.py - Stud GA templet(种马遗传算法模板)

算法描述:

本模板实现的是种马遗传算法。算法流程详见参考文献[1]。

参考文献:

[1] Khatib W , Fleming P J . The stud GA: A mini revolution?[C]// International

Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1998.

'tour' # 锦标赛选择算子

soea\_DE\_best\_1\_bin\_templet : class - 差分进化DE/best/1/bin算法模板

算法描述:

本模板实现的是DE/best/1/bin单目标差分进化算法。

为了实现矩阵化计算，本模板采用打乱个体顺序来代替随机选择差分向量。算法流程如下：

1) 初始化候选解种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 选择变异的基向量，对当前种群进行差分变异，得到变异个体。

5) 将当前种群和变异个体合并，采用二项式分布交叉方法得到试验种群。

6) 在当前种群和实验种群之间采用一对一生存者选择方法得到新一代种群。

7) 回到第2步。

self.selFunc = 'rws' # 基向量的选择方式，采用锦标赛选择

self.recOper = ea.Xovbd(XOVR = 0.5, Half = True) # 生成二项式分布交叉算子对象，这里的XOVR即为DE中的Cr

soea\_DE\_best\_1\_L\_templet : class - 差分进化DE/best/1/L算法模板

算法描述:

本模板实现的是经典的DE/best/1/L单目标差分进化算法。

为了实现矩阵化计算，本模板采用打乱个体顺序来代替随机选择差分向量。算法流程如下：

1) 初始化候选解种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 选择变异的基向量，对当前种群进行差分变异，得到变异个体。

5) 将当前种群和变异个体合并，采用指数交叉方法得到试验种群。

6) 在当前种群和实验种群之间采用一对一生存者选择方法得到新一代种群。

7) 回到第2步。

self.selFunc = 'rws' # 基向量的选择方式，采用锦标赛选择

self.recOper = ea.Xovexp(XOVR = 0.5, Half = True) # 生成指数交叉算子对象，这里的XOVR即为DE中的Cr

soea\_DE\_rand\_1\_bin\_templet : class - 差分进化DE/rand/1/bin算法模板

算法描述:

本模板实现的是经典的DE/rand/1/bin单目标差分进化算法。

为了实现矩阵化计算，本模板采用打乱个体顺序来代替随机选择差分向量。算法流程如下：

1) 初始化候选解种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 选择差分变异的基向量，对当前种群进行差分变异，得到变异个体。

5) 将当前种群和变异个体合并，采用二项式分布交叉方法得到试验种群。

6) 在当前种群和实验种群之间采用一对一生存者选择方法得到新一代种群。

7) 回到第2步。

self.selFunc = 'rws' # 基向量的选择方式，采用轮盘赌选择

self.recOper = ea.Xovbd(XOVR = 0.5, Half = True) # 生成二项式分布交叉算子对象，这里的XOVR即为DE中的Cr

soea\_DE\_rand\_1\_L\_templet : class - 差分进化DE/rand/1/L算法模板

算法描述:

本模板实现的是经典的DE/rand/1/L单目标差分进化算法。

为了实现矩阵化计算，本模板采用打乱个体顺序来代替随机选择差分向量。算法流程如下：

1) 初始化候选解种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 选择差分变异的基向量，对当前种群进行差分变异，得到变异个体。

5) 将当前种群和变异个体合并，采用指数交叉方法得到试验种群。

6) 在当前种群和实验种群之间采用一对一生存者选择方法得到新一代种群。

7) 回到第2步。

self.selFunc = 'rws' # 基向量的选择方式，采用轮盘赌选择

self.recOper = ea.Xovexp(XOVR = 0.5, Half = True) # 生成指数交叉算子对象，这里的XOVR即为DE中的Cr

soea\_DE\_targetToBest\_1\_bin\_templet : class - 差分进化DE/target-to-best/1/bin算法模板

算法描述:

本模板实现的是经典的DE/target-to-best/1/bin单目标差分进化算法。

为了实现矩阵化计算，本模板采用打乱个体顺序来代替随机选择差分向量。算法流程如下：

1) 初始化候选解种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 采用target-to-best的方法选择差分变异的基向量，对当前种群进行差分变异，得到变异个体。

5) 将当前种群和变异个体合并，采用二项式分布交叉方法得到试验种群。

6) 在当前种群和实验种群之间采用一对一生存者选择方法得到新一代种群。

7) 回到第2步。

self.recOper = ea.Xovbd(XOVR = 0.5, Half = True) # 生成二项式分布交叉算子对象，这里的XOVR即为DE中的Cr

soea\_DE\_targetToBest\_1\_L\_templet : class - 差分进化DE/target-to-best/1/L算法模板

算法描述:

本模板实现的是经典的DE/target-to-best/1/L单目标差分进化算法。

为了实现矩阵化计算，本模板采用打乱个体顺序来代替随机选择差分向量。算法流程如下：

1) 初始化候选解种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 采用target-to-best的方法选择差分变异的基向量，对当前种群进行差分变异，得到变异个体。

5) 将当前种群和变异个体合并，采用二项式分布交叉方法得到试验种群。

6) 在当前种群和实验种群之间采用一对一生存者选择方法得到新一代种群。

7) 回到第2步。

self.recOper = ea.Xovexp(XOVR = 0.5, Half = True) # 生成指数交叉算子对象，这里的XOVR即为DE中的Cr

soea\_ES\_1\_plus\_1\_templet : class - (1+1)进化策略模板

算法描述:

本模板实现的是(1+1)进化策略。算法流程如下：

1) 根据编码规则初始化N个个体的种群。

2) 若满足停止条件则停止，否则继续执行。

3) 对当前种群进行统计分析，比如记录其最优个体、平均适应度等等。

4) 初始化控制高斯变异中的标准差Sigma。

5) 独立地对这种群个体进行高斯变异，得到试验种群。

6) 在当前种群和实验种群之间采用一对一生存者选择方法得到新一代种群，

同时统计新一代种群中有多少个个体继承自实验种群（即变异成功率）。

7) 根据变异成功率修改Sigma。

8) 回到第2步。