HSEA-HW4

匡亚明学院　张子谦　　191240076　　191240076@smail.nju.edu.cn

本人承诺该实验全程由本人独立完成，无抄袭或给予他人抄袭行为，代码在截至日期后上传至本人github

摘要： 本实验在特征选择的背景问题下，首先实现了Greedy算法作为baseline，其次复现了POSS, NSGA_II, MOEA/D三种遗传算法，通过实验比对效果，进行分析讨论。最后针对该问题进行改进尝试，复现了PORSS方法，提出了淘汰候选集，代理函数两种针对POSS/PORSS的改进方式，提升个体的多样性从而增强跳出局部最优解的能力。

1、引言

子集选择问题（subset selection problem）具有广泛的应用场景，而在遗传算法是解决子集选择问题较为广泛使用的方法。下面通过POSS, NSGA_II, MOEA/D等遗传算法，在sonar, ionosphere等数据集上进行实验，比对效果。

关于代码运行，细节见README.md。

2、实验内容

2.0 Greedy Baseline

基于前向Greedy的方式，构建一个Baseline用于比对效果。

dataset	R^2
sonar	0.4248
ionosphere	0.4926
svmguide3	0.2405
triazines	0.3066

2.1 POSS

在参考论文[1]以及MATLAB版本的代码上实现了POSS算法，位于POSS.py文件中。具体流程包括：

1、初始化种群population为全0的ndarray，对于每一位，为0表示不选择该feature，为1表示选择该feature进行回归。

2、随机从population中挑选个体s，进行bit-wise的mutation。

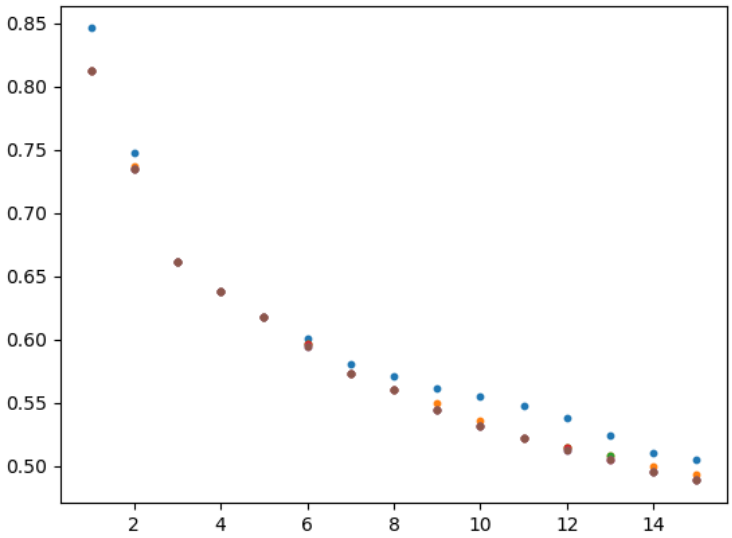
3、评估解的好坏，包括回归的mse以及所选择的feature的个数

4、更新种群，如果生成解所选择feature个数小于2*k，则与种群中的个体进行比较，删去其所支配的个体，如果其不被支配，则也加入种群。

5、重复2-5步2enk^2次，返回最后满足条件的population中的最优个体。

在细节上，缺失数据进行补零处理；通过sklearn的LinearRegression进行回归，得到mse loss（最小化目标），在最后计算1-mse即为极大化的目标R^2。

以其在sonar_scale数据集上的运行情况为例，得到每隔若干次绘制当前population的两个fitness如下，横轴为选取feature的数目，纵轴为mse loss的大小，不同颜色的散点代表不同时刻当前的不被支配的解。（关于其他图例，位于pics/文件夹中）



最后在四个数据集上运行得到的结果为：

dataset	R^2
sonar	0.4368 ± 0.0015
ionosphere	0.4991 ± 0.0000
svmguide3	0.2561 ± 0.0000
triazines	0.3114 ± 0.0010

可以看见相较于Baseline，已经有了很不错的提升。

发现与论文中的数据对比发现存在一定差距，但是细看后注意到论文中是将数据集分为训练集和测试集后在测试集上验证的效果，所以些许的出入是被允许的。（注：其中svmguide3的数据是由libsvm官网中的svmguide3和svmguide3_t组合得到的数据集，与单独的svmguide3数据集的结果存在一些差距）

2.2 NSGA_II

参考论文[2]和PPT的基础上实现了NSGA-II 算法，位于NSGA_II.py文件中，具体流程如下：

1、随机生成大小为N的初始种群（参数默认为40），然后计算mse和选择feature数目作为种群的fitness，然后进行一次non_dominated_sorting，将每个个体rank作为fitness的第三个分量存储。

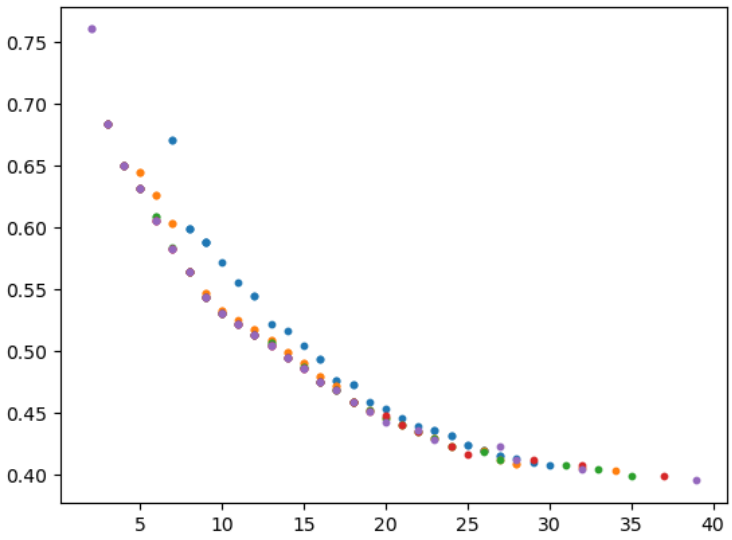
2、进行epoch次迭代（参数默认为400），重复3-4，迭代过程如下：

3、先通过binary tournament，基于rank选取2个父代，然后通过one-point crossover和mutation生成了2个子代，如此重复直至生成N个offspring，并计算offspring的fitness。

4、将offspring和上一代population合并成candidates，进行N+N selection，也即先通过non-dominated sorting计算rank，按rank从小到大的顺序向下一代中添加个体，当剩下的空间不足时，通过crowding_distance从同一rank中选出distance最大的若干个个体，添加入下一代，最终得到新的population。

5、最后选取符合条件的最优个体返回。

图中为其在sonar_scale数据集上的表现：



最后在四个数据集上得到的结果为：

dataset	R^2
sonar	0.4333 ± 0.0037
ionosphere	0.4991 ± 0.0000
svmguide3	0.2561 ± 0.0000
triazines	0.3111 ± 0.0008

2.3 MOEA/D

参考论文[3]和部分博客的基础上实现了MOEA/D算法，位于MOEA_D.py文件中，其中分别实现了weighted_sum和Tchbycheff两种方式的MOEA/D，下以Tchbycheff方式为例介绍，具体流程如下：

Step 1) Initialization:

Step 1.1) Set EP $= \emptyset$.

Step 1.2) Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i.

Step 1.3) Generate an initial population $x^1, \ldots, x^N$ randomly or by a problem-specific method. Set $FV^i = F(x^i)$.

Step 1.4) Initialize $z = (z_1, \ldots, z_m)^T$ by a problem-specific method.

Step 2) Update:

For $i = 1, \ldots, N$, do

Step 2.1) Reproduction: Randomly select two indexes k, l from $B(i)$, and then generate a new solution y from x^k and x^l by using genetic operators.

Step 2.2) Improvement: Apply a problem-specific repair/ improvement heuristic on y to produce y'.

Step 2.3) Update of z: For each $j = 1, \ldots, m$, if $z_j < f_j(y')$, then set $z_j = f_j(y')$.

Step 2.4) Update of Neighboring Solutions: For each index $j \in B(i)$, if $g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$, then set $x^j = y'$ and $FV^j = F(y')$.

Step 2.5) Update of EP:

Remove from EP all the vectors dominated by $F(y')$.

Add $F(y')$ to EP if no vectors in EP dominate $F(y')$.

Step 3) Stopping Criteria: If stopping criteria is satisfied, then stop and output EP. Otherwise, go to **Step 2**.

1、随机生成大小为N的lambdas（参数默认为40），$\lambda[i] = (\frac{(N-i-1)}{(N-1)}, \frac{i}{N-1})$对应一个子问题。然后为每个子问题，根据λ的范数距离得到距离最近的neighbourSize个邻居（默认为3），并随机生成N个repair后的个体作为population，每个个体是每个子问题当前的最优解。

2、为每个个体计算target值（即mse和选择的feature数目），根据target初始化EP为当前population的所有非支配个体，然后根据Tchbycheff函数计算每个个体的函数值，作为fitness。

3、进行epoch次迭代（参数默认为400），重复4-6，迭代过程如下：

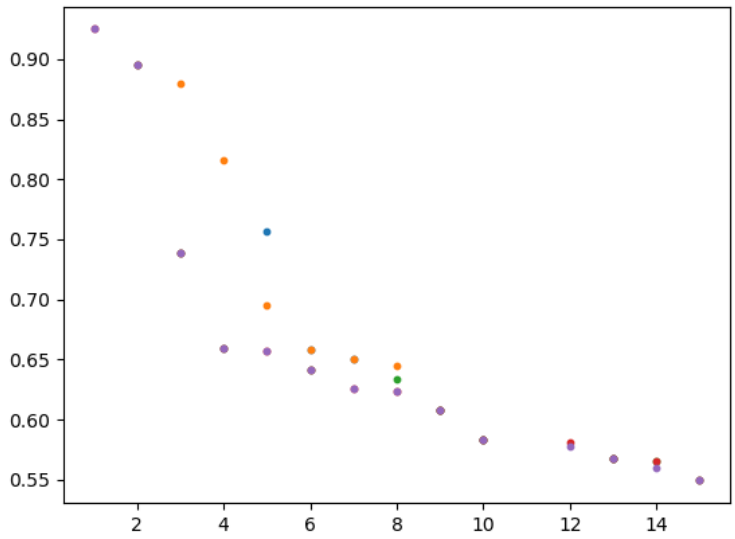
4、遍历种群中的N个个体，对每个个体，从邻居中随机挑选出两个，通过mutation和uniform_crossover操作生成1个新的子代个体，并进行repair并计算它的target值。

5、根据target值，对选中个体的所有邻居问题，计算子代个体的Tchbycheff值，如果更优，则进行替换更新。

6、根据子代个体的target值，更新EP。

注：这里的repair函数设置为，如果选择feature数目大于2*k，则以90%的概率，从已选择的feature中有放回的挑选2k个；以10%的概率从已选择的feature中有放回的挑选k个，通过np.unique去除相同的feature，以保证最后的解能满足条件。如果不进行repair，有一定可能无法找到满足约束条件的解。

图中为其在sonar_scale数据集上的表现：



最后尝试不同参数和方法在四个数据集上得到的结果为：

dataset	ws(40,5)	tc(40,5)	ws(100,5)	tc(100,5)
sonar	0.3912 ± 0.0095	0.3744 ± 0.0183	0.3943 ± 0.0112	0.3912 ± 0.0179
ionosphere	0.4628 ± 0.0085	0.4572 ± 0.0214	0.4797 ± 0.0114	0.4720 ± 0.0057
svmguide3	0.2426 ± 0.0000	0.2384 ± 0.0104	0.2463 ± 0.0055	0.2356 ± 0.0160
triazines	0.2412 ± 0.0196	0.2300 ± 0.0238	0.2601 ± 0.0196	0.2344 ± 0.0259

2.4 Empirical Evaluation

dataset	Baseline	POSS	NSGA-II	MOEA/D
sonar	0.4248	0.4368 ± 0.0015	0.4333 ± 0.0037	0.3943 ± 0.0112
ionosphere	0.4926	0.4991 ± 0.0000	0.4991 ± 0.0000	0.4797 ± 0.0114
svmguide3	0.2405	0.2561 ± 0.0000	0.2561 ± 0.0000	0.2463 ± 0.0055
triazines	0.3066	0.3114 ± 0.0010	0.3111 ± 0.0008	0.2601 ± 0.0196

其中POSS运行为2ek^2n，NSGA-II和MOEA/D都分别迭代400次，但是由于每次迭代中需要计算新生成的至少N个个体的target，加上non-dominated sorting等操作所以运行时间没有太大区别，甚至更快。

以sonar为例，单次运行时间分别为7.1 sec，11.4 sec， 11.9 sec。

POSS无论是运行时间还是最后的结果上，表现都最好，NSGA-II相差不是太大。但是MOEA/D并没有太好的处理该问题（仅针对本人实现的代码而言），一开始认为可能是分解后mse和feature数目在量级上存在一定差距，所以经过normalize将两个targets都放缩到了mean 0, std 1，但是改进仍然不是明显。

2.5 改进尝试

1、PORSS复现

参考了[4]，与POSS区别在于引入了one-point或uniform的crossover。

为更好对比算法运行效率的提升，我们再控制运行次数为2ekn

dataset	POSS	PORSS_o	PORSS_u
sonar	0.4283 ± 0.0055	0.4343 ± 0.0019	0.4357 ± 0.0017
ionosphere	0.4898 ± 0.0053	0.4978 ± 0.0013	0.4989 ± 0.0008
svmguide3	0.2531 ± 0.0027	0.2552 ± 0.0013	0.2557 ± 0.0011
triazines	0.3037 ± 0.0046	0.3110 ± 0.0008	0.3103 ± 0.0016

可以看见PORSS在同等时间复杂度上的运行效果都优于POSS。

2、基于POSS/PORSS的改进

下面基于POSS/PORSS提出了一些改进的方式。对于POSS，其种群中对于每个特征数量，只保留一个对应的个体解，虽然在保证收敛性的同时极大提高了运行效率，但是只依赖于mutation, recombination操作有时无法跳出局部最优。所以希望能够增加种群的多样性，保留一部分当前的sub-optimal的个体解，增强多样性，提高跳出局部最优的能力。依次，提出了两种方式：

2.1 淘汰候补集

代码位于 ./POSS_candidate.py中

在POSS的基础上，扩增一个外部集合Q，用于存储在被种群P剔除的个体中的最优解（为防止集合规模过大，存储方式与P相同，对于固定的选取特征数量，只存储一个对应的sub-optimal解）。在选取父代解的时候，以p_i的概率选择P中的解作为父代解，其中概率设定为：$p_i = \frac{1}{1+e^{i-T}}$，其中$T = 2enk^2$为迭代总次数。也即在运行前期，希望主要从按照POSS的方式从P中选取父代解，在后期可能困于局部最优的时候，希望通过外部集合Q中的解进行mutation和recombination产生新的解来跳出局部最优。

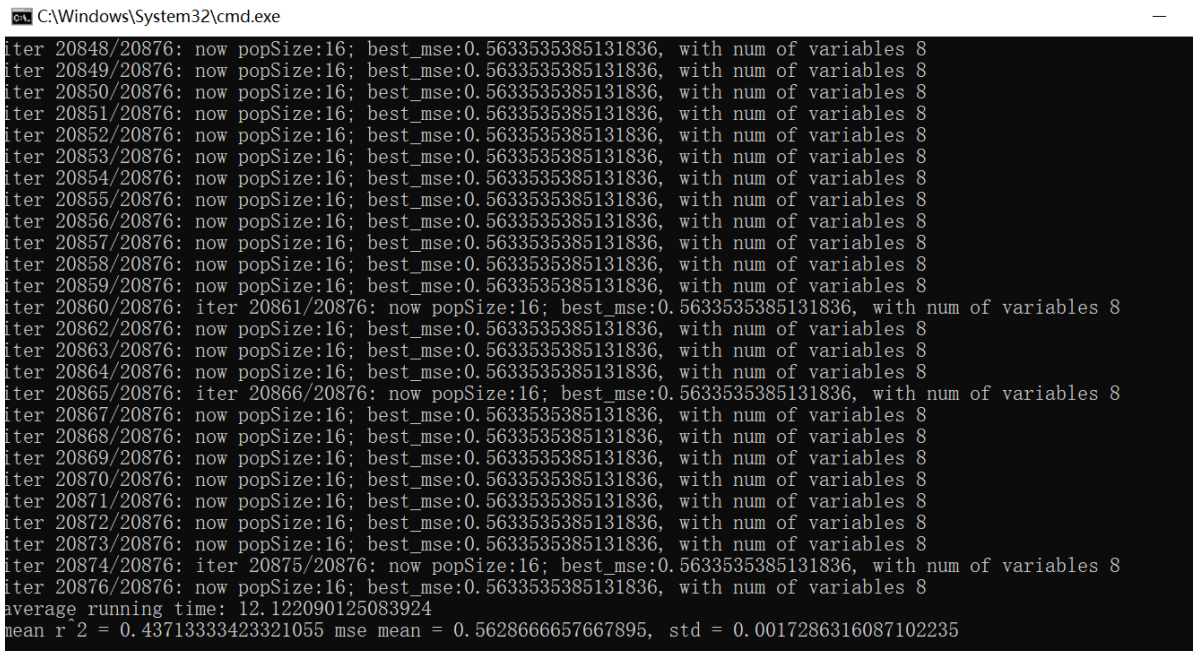
伪代码如下：

Algorithm 1: POSS_candidate

Input: all Variables $V = \{X_1, ...X_n\}$, a given criterion f and an integer parameter $k \in [1, n]$, iteration numbers T
Output: a subset of V with at most k variables

Let $s = \{0\}^n$ and $P = Q = \{s\}$
Let $t = 0$
Define $dominate(a, b)$ is True if $a.o1 \leq b.o1$ and $a.o2 < b.o2$ or $a.o1 < b.o1$ and $a.o2 \leq b.o2$
while $t<T$ **do**
    $p_i = schedule(t, T)$
    Select s from P uniformly at random with prob. p_i or from Q with prob. $1 - p_i$
    Generate s' from s by applying bit-wise mutation
    $candidates \leftarrow \emptyset$
    $to\_add \leftarrow True$
    for $z \in P$ **do**
        if $dominate(z, s')$ **then**
            $to\_add \leftarrow False$
        end
        if $dominate(s', z)$ **then**
            $candidates = candidates \cup \{z\}$
        end
    end
    $P = P \backslash candidates$
    if to_add **then**
        $P = P \cup \{s'\}$
    end
    for $candidate \in candidates$ **do**
        for $z \in Q$ **do**
            if $candidate.o1 < z.o1$ *and* $candidate.o2 = z.o2$ **then**
                $Q = (Q \backslash \{z\}) \cup \{candidate\}$
                break
            end
        end
    end
    $t = t + 1$
end
return $\arg\max_{x \in P, |x| \leq k} f(x)$

运行效果对比如下



dataset	POSS	POSS_candidate
sonar	0.4368 ± 0.0015	0.4371 ± 0.0017
triazines	0.3114 ± 0.0010	0.3113 ± 0.0009

其中ionosphere和svmguide3由于本身特征的数目较少，往往都能在固定时间以标准差为0收敛到最优解，所以如不在表格中列出，则默认效果相同，达到最优。

对于PORSS，实现方式基本相同，在选择父代解时，由于需要选择两个个体，所以以$\frac{1-p_i}{2}$的概率从P,Q中分别选取，$\frac{1-p_i}{2}$的概率从Q中选取，p_i的概率从P中选取。

运行$2enk^2$次的结果如下：

dataset	PORSS_o	PORSS_u	PORSS_co	PORSS_cu
sonar	0.4374 ± 0.0019	0.4371 ± 0.0017	$.4375 \pm .0018$	0.4371 ± 0.0017
triazines	0.3117 ± 0.0008	0.3122 ± 0.0005	0.3118 ± 0.0005	0.3119 ± 0.0007

2.2 代理函数

代码位于 ./POSS_surrogate.py中

参考[5]中作者提出结合了$c(x)$和f构建出代理函数的思想，希望能引入一个外部集合，以代理函数g和$|X|$ 作为最小化的目标函数存储个体解。但是对于本实验中的特征选择问题，并不像作者提出的EAMC中一般的cost constrained 子集选择问题在种群中既能考虑到选择特征的数量也利用到cost function。所以这里提出了一个“伪"代理函数，$g_t(x)=\alpha_t f(x)$用于外部集合Q的淘汰筛选。

f仍然是fitness evaluation function，而α_t是一个real-valued number，我们以初始值为1，步长$\sigma=0.1$，$\alpha_{t+1}=\alpha_t+\sigma\mathcal{N}(0,1)$ 的方式更新。注意到，如果Q内部和外部的集合都是用$g_t(x)$的话，本质与使用$f(x)$衡量没有区别，所以对于Q内部的集合，可以有两种方式计量：1、仍然使用$f(x)$，2、记录其进入Q集合时的α_t。这里采用了第一种计量的方式。

流程在POSS的基础上，对于新生成的个体解，以$g_t(x)$，在Q中与具有和x相同选择特征数量的个体比较竞争。在选取父代解时，以$1-p_i$的概率从Q中选择，p_i的计算方式采取了和上一小节中淘汰代理相同的方式。

伪代码如下：

Algorithm 2: POSS_surrogate

Input: all Variables $V=\{X_1,...X_n\}$, a given criterion f and an integer parameter $k\in[1,n]$, iteration numbers T
Output: a subset of V with at most k variables

Let $s=\{0\}^n$ and $P=Q=\{s\}$
Let $t=0$
Let $\alpha_0=1.0,\sigma=0.1$
Define $dominate(a,b)$ is True if $a.o1\leq b.o1$ and $a.o2<b.o2$ or $a.o1<b.o1$ and $a.o2\leq b.o2$
while $t<T$ **do**
  $p_i=schedule(t,T)$
  Select s from P uniformly at random with prob. p_i or from Q with prob. $1-p_i$
  Generate s' from s by applying bit-wise mutation
  $knockouts\leftarrow\varnothing$
  $to\_add\leftarrow True$
  for $z\in P$ **do**
    if $dominate(z,s')$ **then**
      $to\_add\leftarrow False$
    end
    if $dominate(s',z)$ **then**
      $knockouts=knockouts\cup\{z\}$
    end
  end
  $P=P\backslash knockouts$
  if to_add **then**
    $P=P\cup\{s'\}$
  end
  for $z\in Q$ **do**
    if $\alpha_t s'.o1<z.o1$ *and* $s'.o2=z.o2$ **then**
      $Q=(Q\backslash\{z\})\cup\{s'\}$
      break
    end
  end
  $\alpha_{t+1}=\alpha_t+\sigma\mathcal{N}(0,1)$
  $t=t+1$
end
return $\arg\max_{x\in P,|x|\leq k}f(x)$

运行效果对比如下：

```
iter 20870/20876: now popSize:16; best_mse:0.5602747477017916, with num of variables 8
iter 20871/20876: now popSize:16; best_mse:0.5602747477017916, with num of variables 8
iter 20872/20876: now popSize:16; best_mse:0.5602747477017916, with num of variables 8
iter 20873/20876: now popSize:16; best_mse:0.5602747477017916, with num of variables 8
iter 20874/20876: now popSize:16; best_mse:0.5602747477017916, with num of variables 8
iter 20875/20876: now popSize:16; best_mse:0.5602747477017916, with num of variables 8
iter 20876/20876: now popSize:16; best_mse:0.5602747477017916, with num of variables 8
average running time: 11.222725820541381
mean r^2 = 0.4378025275010329 mse mean = 0.5621974724989671, std = 0.0019439571709305473
```

```
iter 20868/20876: now popSize:16; best_mse:0.6874332633069766, with num of variables 8
iter 20869/20876: now popSize:16; best_mse:0.6874332633069766, with num of variables 8
iter 20870/20876: now popSize:16; best_mse:0.6874332633069766, with num of variables 8
iter 20871/20876: iter 20872/20876: now popSize:16; best_mse:0.6874332633069766, with num of variables 8
iter 20873/20876: now popSize:16; best_mse:0.6874332633069766, with num of variables 8
iter 20874/20876: now popSize:16; best_mse:0.6874332633069766, with num of variables 8
iter 20875/20876: iter 20876/20876: now popSize:16; best_mse:0.6874332633069766, with num of variables 8
average running time: 11.23890974521637
mean r^2 = 0.3117046356201172 mse mean = 0.6882953643798828, std = 0.0009471757065576652
```

dataset	POSS	POSS_surrogate
sonar	0.4368 ± 0.0015	0.4378 ± 0.0019
triazines	0.3114 ± 0.0010	0.3117 ± 0.0009

2.3 Discussion

提出的两种改进方式在重复10次的情况下都达到了不错的效果，甚至相较于POSS，最终结果有了一定的提升（由于算法本身的随机性，尤其是第二种方法代理函数中的α_t有较大的随机性，结果可能并不稳定，单次的运行结果如上图所示）。但是对外部集合Q的维护也明显带来了运行效率上的下降，在这样的小规模问题上还不突出，如果特征数量更加庞大，那么维护Q的成本也会提升，但是好处在于并没有引入新的fitness evaluation这样最expensive的过程。

3、一些其他的尝试

此外参考了一些以PSO，ACO等方法进行特征选择的算法，这里直接借用了网上现有的matlab代码运行尝试，但并没有在相近运行时间上实现效果的提升，故不再赘述。

3、总结

本次实验完整实现POSS, NSGA-II, MOEA/D在子集选择问题上的应用，并进行了算法的改进尝试。通过本次实验对限制条件优化，多目标优化的演化算法求解有了更加深入的认识了解。

References

[1] *Chao Qian, Yang Yu, and Zhi-Hua Zhou. 2015. Subset selection by Pareto optimization. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (**NIPS'15**). MIT Press, Cambridge, MA, USA, 1774–1782.*

[2] *K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002, doi: 10.1109/4235.996017.*

[3] *Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," in IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, pp. 712-731, Dec. 2007, doi: 10.1109/TEVC.2007.892759.*

[4] *Qian, C., Bian, C., & Feng, C. (2020). Subset Selection by Pareto Optimization with Recombination. Proceedings of the AAAI Conference on Artificial Intelligence, 34(03), 2408-2415.*

[5] *Bian, C., Feng, C., Qian, C., & Yu, Y. (2020). An Efficient Evolutionary Algorithm for Subset Selection with General Cost Constraints. Proceedings of the AAAI Conference on Artificial Intelligence, 34(04), 3267-3274.*