基于关系学习和预测的昂贵多目标优化

摘要

摘要——昂贵多目标优化问题对于进化算法来说具有巨大的挑战性，因为其评估成本较高。通过构建廉价的代理模型来替代昂贵的真实模型已被证明是减少昂贵评估次数的有效方法。通常借助机器学习技术构建回归模型（用于近似候选解的适应度值）或分类器（用于估计候选解的类别）作为替代模型。本文提出了一种新的关系模型，可用于解决昂贵多目标优化问题。与直接估计候选解的质量不同，关系模型预测一对解〈x, y〉之间的关系。具体实现步骤为：首先准备一个平衡的训练集，然后基于训练数据集构建一个分类器以学习这种关系，最后采用分类器来估计候选解与父代解之间的关系。通过这种方式选出优秀的候选解用于真实模型识别和评估。

关键词：昂贵优化；多目标优化；代理辅助进化算法；关系模型

1 引言

现实中很多真实问题的优化问题往往代价高昂，需要进行大量计算、模拟和实验。进化算法（Evolutionary algorithms, EAs）被广泛应用于复杂优化问题，但难以解决昂贵优化问题，因为它们的优化过程通常依赖大量的适应度评估 (Fitness evaluations, FEs) [2]。然而在进行昂贵问题优化时，可以进行的适应度评估次数非常有限，为解决这一困境，许多代理辅助进化算法 (SAEAs) 被提出 [5]。借助计算方便的辅助模型，SAEAs 可以在计算资源有限的情况下完成优化。SAEAs 使用建立的代理模型来评估新产生的候选解，只有少数被被认为有前景的解会使用真实目标函数进行评估，因此省去了大量昂贵的适应度估计。

设计一个 SAEA 算法的三个关键点为：训练数据的准备、辅助模型构建、模型的用法。这其中辅助模型的构建最为重要。现有的 SAEAs 可以被大致分为两类：

基于回归的 SAEAs：这类模型通过构建回归模型来模拟真实适应度函数，输入解向量并返回近似的目标值，这是在昂贵优化领域应用最广泛的策略。

基于分类的 SAEAs：这类模型通过构建分类模型来预测解所属的类别，这类模型近年来备受关注。

无论是通过适应度估计还是分类，其主要目的都是判断一个解的质量，因此有方法提出可以在不计算解的适应度的情况下比较两个解的优劣 [4]，根据这种思想，本文提出了另一个模型，称为关系模型，通过学习和预测昂贵的多目标优化问题的一对解之间的关系来间接预测解的质量。为简单起见，我们将此基于关系选择（Relation selection, RS）的进化多目标优化算法称为 REMO。为了实现 RS 多目标优化，本文将基于关系选择解决三个主要问题：训

练数据准备、关系模型构建和关系模型的使用。

2 相关工作

2.1 昂贵多目标优化问题

本文仅讨论如下所示无约束 MOPs，

$$\begin{aligned} min \quad & F(x) = (f_1(x), \cdots, f_m(x))^T \\ s.t \quad & x \in \prod_{i=1}^{n}[a_i, b_i] \end{aligned} \tag{1}$$

其中 $x = (x_1 \ \cdots \ x_n) \in R^n$ 为决策向量;$a_i < b_i (i = 1 \ \cdots \ n)$ 分别为解空间可行域的下界和上界;$F : R^n \rightarrow R^m$ 由 m 个目标函数 $f_i (i = 1 \ \cdots \ m)$ 组成，由于公式 (1) 中目标之间存在冲突性，通常没有一个解能够同时优化所有目标。这些权衡解称为帕累托最优解，构成了公式 (1) 的帕累托最优集。
当 $F(x)$ 的评估代价昂贵时，例如需要耗时的计算和高成本的试验，该问题被称为昂贵多目标问题。在面对昂贵多目标问题时，算法的目标变为使用有限的适应度评估来优化出一个 MOP 的帕累托最优解集。

2.2 二元关系学习

二元关系学习的基本思想是学习解对之间的关系。设 A、B、C 为三个解，它们的目标值由高到低依次递减（目标值越低越好)。我们可以构造一个正训练集 $<$ B, A $>$，$<$ C, A $>$，$<$ C, B $>$，每个元素的标签为' +1 '; 一个负训练集 $<$ A, B $>$，$<$ A, C $>$，$<$ B, C $>$，每个元素的标签为' -1 '。对于每一对 $< x_i, x_j > (x_i, x_j \in A, B, C)$，标签' +1 ' 表示 x_i 优于 x_j 的关系，标签' -1 ' 表示 x_i 劣于 x_j 的关系。

3 本文方法

3.1 本文方法概述

在多目标优化中，一对解 x_i, x_j 之间存在三种可能的关系，即 x_i 支配 x_j、x_i 被 x_j 支配，或者 x_i 与 x_j 互为非支配关系。此外，多目标优化的目标是找到一个近似解集，使其尽可能地接近帕累托最优集同时保持分布良好。因此，一对解之间的关系应考虑收敛性和多样性两个方面。基于此，本节提出了一种基于关系的选择（RS）方法，模型框架如图1所示。

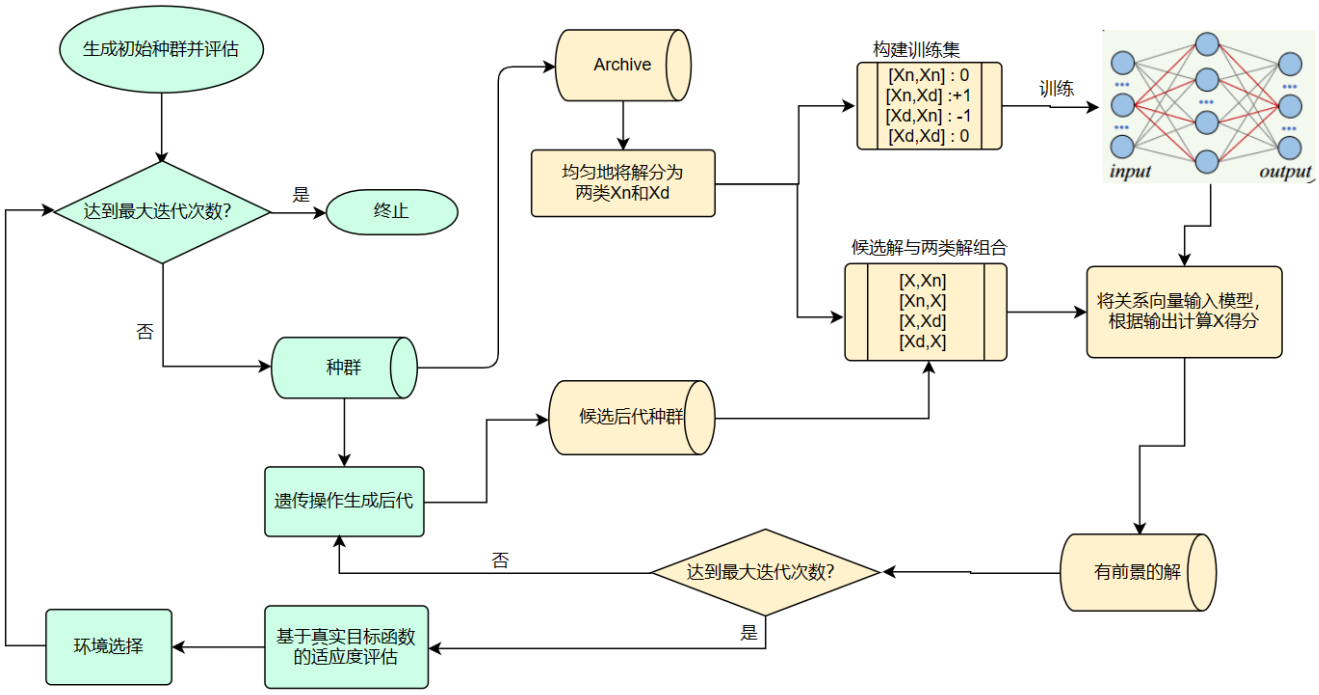

图 1. REMO 框架图

算法包括两个部分: 基本 MOEA 过程和本文提出的 RS 过程。MOEA 为基本的多目标进化过程，每轮循环由 RS 过程选择有前景的候选解交由实际目标函数评估。在 RS 部分，首先用新生成的解更新 Archive，然后准备训练集，接着用给定的数据集训练一个轻量级的神经网络，最后基于神经网络的输出和一种投票策略对候选解进行评估。

3.2 种群分割方法

对于多目标优化，帕累托支配关系是做数据划分最合适的方法。然而，将解划分为非支配解和支配解，既不能保证训练集中标签的平衡，也无法保证训练集的多样性。根据 CSEA [3] 中基于参考解构建分类边界的思想，本文提出了基于角度的支配的概念，该概念自适应地改变参考点的支配范围，以平衡支配和非支配解的数量。具体实现方法如下。

1）选择参考点：采用基于径向投影的选择方法 [1]，将 m 维目标向量映射到 2 维径向空间，然后将径向空间划分为小网格。通过这种方法，可以在保证多样性的多样性的情况下选择最优的 k 个解作为参考点。在 REMO 的环境选择过程中，我们也使用此方法选择有前景的解加入下一代种群。

2）自适应分类边界：为建立一个平衡的数据集。在本节中，我们使用带有自适应惩罚因子 δ 的 PBI 方法 [6] 将种群 P 划分为平衡的非支配解集 P_n 和支配解集 P_d。

设 $Z = (z_1, z_2 \cdots z_m)$ 为当前总体的理想点 (z_i 为各目标值最小值)。设 $P_{ref} = x^1_{ref}, x^2_{ref} \cdots x^k_{ref}$ 为 k 个参考点的集合。定义如下:

$$
\begin{aligned}
g^{pbi}(F(x)|F(x^i_{ref}, Z) &= d_1 + \delta d_2 \\
d_1 &= \frac{(F(x) - Z)^T (F(x^i_{ref}) - Z)}{|F(x^i_{ref}) - Z|} \\
d_2 &= |F(x) - Z| \sqrt{1 - (d_1 |(F(x) - Z)|)^2}
\end{aligned} \tag{2}
$$

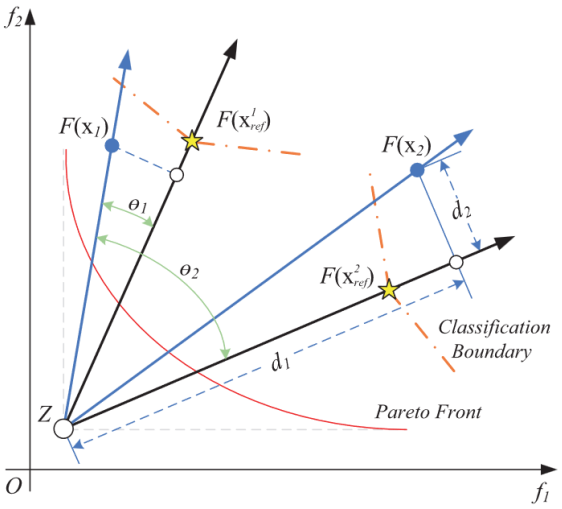

图 2. 自适应分类边界示意图

如图2所示，d_1 表示线段 $ZF(x)$ 在直线 $ZF(x^i_{ref})$ 上的投影长度，d_2 表示 $F(x)$ 与直线 $ZF(x^i_{ref})$ 之间的垂直距离，δ 是一个惩罚因子，用于控制分类边界的夹角。对于一个解 x$\in P$ $g^{pbi}(F(x)|F(x^i_{ref}, Z) < |F(x^{i^*}_{ref})-Z|$ $i^* = argmax_{i\in 1\ldots k} cos(\theta_i)$。
根据建立的分类边界，我们可以将种群 P 划分为非支配解 P_n 和被支配解 P_d。图3所示的算法阐述了种群分割方法，包括 δ 的调整过程，该算法不断调整 δ 的大小来控制某一参考点对应的分类边界角度，以此来将种群均分。

Algorithm 2: PartitionPopulation(P_{ref}, P)

Output: P_n,P_d (partitioned subpopulations)

$\delta_l \leftarrow 0, \delta_u \leftarrow 0$
Increase and/or decrease δ_l,δ_u with step size $\Delta = 10$, to find a bracket such that $\frac{|P_n(\delta_u)|}{|P|} \le 0.5 \le \frac{|P_n(\delta_l)|}{|P|}$;
while $\frac{|P_n(\delta_u)|}{|P|} \le \frac{|P_n(\delta)|}{|P|} \le \frac{|P_n(\delta_l)|}{|P|}$ **do**
  $\delta \leftarrow 0.5(\delta_l + \delta_u)$
  if $\frac{|P_n(\delta)|}{|P|} > 0.5$ **then** $\delta_l \leftarrow \delta$;
  else $\delta_u \leftarrow \delta$;
end
$\delta \leftarrow \arg\min_{\delta\in\{\delta_l,\delta_u\}} |\frac{|P_n(\delta)|}{|P|} - 0.5|$
$P_n \leftarrow P_n(\delta), P_d \leftarrow P \backslash P_n(\delta)$

图 3. 种群分割算法

3.3 关系模型的建立

本文使用神经网络构建一个三分类分类器来学习解对关系。
1）训练集构建：训练数据的形式为 $D = (< x_i, x_j >, l)|x_i, x_j \in P, < x_i, x_j >$ 是由两个解组成

的特征向量，l 为该向量的标签，表示该 x_i, x_j 之间的支配关系,l 取值如下所示。

$$l(< x_i, x_j >) = \begin{cases} +1, & x_i \in P_n, x_j \in P_d \\ -1, & x_i \in P_d, x_j \in P_n \\ +0, & x_i \in P_n, x_j \in P_n \\ -0, & x_i \in P_d, x_j \in P_d \end{cases} \tag{3}$$

其中的"+0" 和 "-0" 归为同一类标签，因此为保证训练数据均衡，将随机保留部分训练数据以保证三类训练数据数量均等。

2）模型训练：为构建一个三分类分类器，本文所使用的神经网络输入为 2n 个变量，隐藏层这里采用三层分别为 3n、2n、n 维，输出为 3 维向量。因此采用独热编码将标签转换为三维向量，例如'+1','0','-1' 分别编码为 [1,0,0]，[0,1,0] 和 [0,0,1]。训练后的模型输出值 $[s_1, s_2, s_3]$ 分别代表解对属于三种关系的概率值。例如 s_1 表示 $x_i \in P_n, x_j \in P_d$，的概率。

3.4 关系模型的运用

该部分介绍了一种投票-积分策略，用于评估候选解的水平。

1）投票和积分：每个新产生的解 u 都将与原种群中所有解 $x, x \in P$ 结合形成若干解对 $< u, x >, < x, u >$，根据 x 所属的类别，可以形成四种组合，取每种组合的平均值参与积分计算，公式如下:

$$\begin{aligned} [s_1^I, s_2^I, s_3^I] &= mean_{x \in P_n}(net(< x, u >)) \\ [s^I I_1, s^I I_2, s^I I_3] &= mean_{x \in P_n}(net(< u, x >)) \\ [s^I II_1, s^I II_2, s^I II_3] &= mean_{x \in P_d}(net(< x, u >)) \\ [s^I V_1, s^I V_2, s^I V_3] &= mean_{x \in P_d}(net(< u, x >)) \end{aligned} \tag{4}$$

积分统计方法:

$$\begin{aligned} s(u) = (s^I I_1 + s^I V_1 + s_2^I + s^I I_2 + s_3^I + s^I II_3) - \\ (s_1^I + s^I II_1 + s^I II_2 + s^I V_2 + s^I I_3 + s^I V_3) \end{aligned} \tag{5}$$

2）基于关系的选择：使用前面提出的投票-积分策略，计算每一个候选解的评分 s, 选择高评分的解加入种群，算法伪代码如下。

Algorithm 3: SearchSolution(P_n, P_d, P_{Ref}, $\mathfrak{C}_m$, FE'_{max}, N')

Output: Q (promising solutions)

Set $Q \leftarrow P_n \cup P_d$ and $t \leftarrow 0$;
while $t < FE'_{max}$ **do**
  $Q \leftarrow$ Generation($Q \cup P_{ref}$)
  Evaluate each $u \in Q$ by Eq.(5) with P_n and P_d;
  $Q \leftarrow$ Select $|P_{Ref}|$ solutions from Q with the largest s values;
  Set $t \leftarrow t + |Q|$;
end
Set $Q \leftarrow$ solutions in Q with the max s value;
if $|Q| < N'$ **then**
  $Q \leftarrow$ Select N' solutions from Q with the largest s values.
end

图 4. 解搜索算法

4 复现细节

4.1 与已有开源代码对比

作者已将源代码开源在 github 和 PlatEMO 上，本复现工作参考了本文在 PlatEMO 上的代码，代码结构如图5所示。

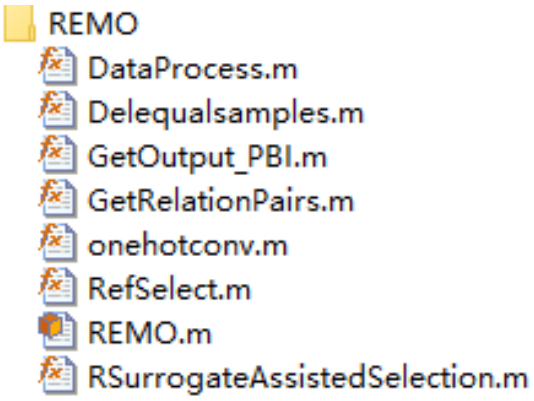

图 5. 源代码文件

其中，REMO.m 为算法主函数，RefSelect.m 为参考点选择函数，Getoutput_PBI.m 用于种群分割并标记解的类别，GetRelationPairs.m 为图3中的种群分割算法，DataProcess.m 用于将解对和标签分为训练集和测试集，onehotconv.m 用于将标签编码为三维独热向量，RSurrogateAssistedSelection.m 为图4中的解搜索算法，

为验证该方法的实现效果，我在 PlatEMO 上进行实验，将该算法与其他 6 个多目标优化算法进行对比，实验的参数设置与原文中相同；由于硬件条件限制，省去了部分测试实例，实验结果如下：

TABLE I: The statistical results of IGD metric values obtained by 7 algorithms with n=10 on 21 test instances over 10 independent runs. The best mean results in each row are highlighted.

Problem	M	D	NSGAIII	CPSMOEA	KRVEA	KTA2	MOEADEGO	CSEA	REMO
DTLZ1	3	10	8.5106e+1 (1.98e+1) -	9.1420e+1 (1.66e+1) -	9.0194e+1 (2.09e+1) -	5.2112e+1 (1.93e+1) =	8.6576e+1 (1.38e+1) -	6.7394e+1 (2.01e+1) =	5.8200e+1 (1.39e+1)
	6	10	2.1339e+1 (8.85e+0) =	2.9838e+1 (9.39e+0) =	3.1658e+1 (1.12e+1) -	1.8870e+1 (1.50e+1) =	3.1735e+1 (7.51e+0) -	1.7450e+1 (4.85e+0) =	2.0797e+1 (9.68e+0)
	10	10	5.5014e-1 (3.12e-1) -	6.5586e-1 (2.54e-1) -	3.6475e-1 (9.73e-2) -	3.6767e-1 (2.09e-1) =	4.7403e-1 (1.73e-1) -	3.3499e-1 (9.37e-2) -	2.5333e-1 (4.27e-2)
DTLZ2	3	10	2.7190e-1 (2.12e-2) -	2.6852e-1 (2.27e-2) -	1.5222e-1 (2.89e-2) +	6.5325e-2 (4.34e-3) +	3.4083e-1 (1.80e-2) -	2.8698e-1 (4.23e-2) -	2.0641e-1 (2.98e-2)
	6	10	5.0186e-1 (3.53e-2) -	6.8784e-1 (3.44e-2) -	3.2220e-1 (1.77e-2) +	2.9166e-1 (1.06e-2) +	4.7260e-1 (1.49e-2) -	4.4757e-1 (2.70e-2) -	3.8502e-1 (4.82e-2)
	10	10	6.8191e-1 (3.36e-2) -	7.4995e-1 (3.76e-2) -	5.2181e-1 (2.96e-2) =	4.8288e-1 (2.97e-2) +	5.4814e-1 (1.85e-2) =	6.5113e-1 (3.21e-2) -	5.5617e-1 (3.92e-2)
DTLZ3	3	10	2.2123e+2 (3.44e+1) -	1.9213e+2 (2.76e+1) -	2.2433e+2 (6.48e+1) -	1.3562e+2 (2.86e+1) =	2.0268e+2 (3.47e+1) -	1.7475e+2 (2.72e+1) -	1.4727e+2 (2.52e+1)
	6	10	8.1877e+1 (2.54e+1) =	1.3630e+2 (3.65e+1) -	7.4154e+1 (2.13e+1) =	3.7691e+1 (2.14e+1) +	9.0088e+1 (1.93e+1) -	5.8966e+1 (1.77e+1) =	6.5959e+1 (1.91e+1)
	10	10	1.7388e+0 (1.13e+0) =	5.9539e+0 (5.79e+0) -	1.2966e+0 (3.50e-1) -	1.4292e+0 (1.09e+0) =	1.1198e+0 (2.62e-1) =	9.6062e-1 (2.71e-1) =	1.0009e+0 (2.41e-1)
DTLZ4	3	10	6.7196e-1 (2.02e-1) -	5.7248e-1 (5.20e-2) -	4.1752e-1 (7.62e-2) -	3.4467e-1 (3.42e-1) =	6.0795e-1 (6.68e-2) -	4.2084e-1 (1.65e-1) -	1.9979e-1 (3.69e-2)
	6	10	7.2842e-1 (5.18e-2) -	6.8212e-1 (3.81e-2) -	5.3860e-1 (2.67e-2) =	5.6326e-1 (1.13e-1) =	6.7712e-1 (3.34e-2) -	5.3289e-1 (3.85e-2) =	5.2042e-1 (5.48e-2)
	10	10	7.4242e-1 (1.56e-2) -	7.6658e-1 (1.48e-2) -	6.1447e-1 (4.06e-2) +	6.0578e-1 (4.88e-2) +	6.5125e-1 (9.18e-3) =	6.5513e-1 (2.38e-2) =	6.6117e-1 (2.33e-2)
DTLZ5	3	10	1.7313e-1 (3.47e-2) -	1.9396e-1 (2.78e-2) -	8.3051e-2 (2.36e-2) +	1.0885e-2 (2.06e-3) +	2.5614e-1 (2.95e-2) -	1.6670e-1 (3.89e-2) -	1.1101e-1 (2.40e-2)
	6	10	1.4445e-1 (1.87e-2) -	2.3020e-1 (5.41e-2) -	4.0455e-2 (9.96e-3) +	6.7936e-2 (1.26e-2) =	1.4601e-1 (1.61e-2) -	8.8144e-2 (1.48e-2) -	6.6030e-2 (1.62e-2)
	10	10	1.4847e-1 (1.99e-2) -	1.4789e-1 (4.93e-2) -	1.2749e-2 (1.32e-3) =	1.7288e-2 (3.23e-3) =	1.8574e-2 (1.77e-3) -	1.4511e-2 (1.35e-3) =	1.5501e-2 (3.91e-3)
DTLZ6	3	10	5.9373e+0 (3.19e-1) -	3.2780e+0 (6.57e-1) +	2.6106e+0 (4.57e-1) +	1.7066e+0 (3.22e-1) +	2.3350e+0 (7.71e-1) +	4.3950e+0 (4.02e-1) -	3.8802e+0 (5.15e-1)
	6	10	3.6083e+0 (4.83e-1) -	2.8462e+0 (8.31e-1) =	1.1330e+0 (3.13e-1) +	1.2304e+0 (4.34e-1) +	1.1506e+0 (3.81e-1) +	2.3608e+0 (5.67e-1) =	2.5002e+0 (8.62e-1)
	10	10	6.6578e-1 (3.10e-1) -	3.4129e-1 (1.45e-1) -	5.2966e-2 (1.93e-2) =	1.1334e-1 (4.21e-2) =	1.8967e-1 (7.90e-2) =	1.0491e-1 (1.03e-1) =	1.5644e-1 (1.86e-1)
DTLZ7	3	10	5.1011e+0 (8.34e-1) -	3.8869e+0 (8.22e-1) -	1.1842e-1 (1.50e-2) +	5.4087e-1 (2.85e-1) =	3.4388e-1 (1.26e-1) =	1.7102e+0 (1.17e+0) -	4.3966e-1 (2.39e-1)
	6	10	9.9998e+0 (3.50e+0) -	3.2158e+0 (2.45e+0) =	5.7450e-1 (4.19e-2) +	8.0352e-1 (4.54e-1) +	9.0098e-1 (5.46e-2) +	3.7373e+0 (5.44e-1) -	2.3800e+0 (6.44e-1)
	10	10	9.3806e+0 (7.14e+0) -	1.8749e+0 (1.52e-1) =	1.0404e+0 (2.81e-2) +	1.3985e+0 (3.88e-1) +	1.2310e+0 (3.63e-2) +	2.2050e+0 (5.65e-1) =	2.0880e+0 (3.81e-1)
+/-/=			0/18/3	1/16/4	10/6/5	10/0/11	4/12/5	0/11/10	

TABLE II: The statistical results of IGD metric values obtained by K-RVEA, KTA2, and REMO on 21 test instances over 5 independent runs. The best mean results in each row are highlighted.

Problem	M	D	KRVEA	KTA2	REMO
DTLZ1	3	50	1.2274e+3 (4.65e+1) -	9.6542e+2 (1.84e+2) =	8.6054e+2 (7.40e+1)
	6	50	8.7550e+2 (3.07e+1) -	8.0678e+2 (4.05e+1) -	6.7329e+2 (8.50e+1)
	10	50	7.8707e+2 (4.19e+1) -	6.1114e+2 (6.30e+1) =	5.9357e+2 (7.71e+1)
DTLZ2	3	50	2.7568e+0 (6.80e-2) -	1.6575e+0 (2.16e-1) =	1.9890e+0 (1.46e-1)
	6	50	2.7753e+0 (1.34e-1) -	2.4824e+0 (1.21e-1) -	1.9311e+0 (2.27e-1)
	10	50	2.7432e+0 (9.43e-2) -	2.3441e+0 (3.10e-1) =	2.1926e+0 (2.93e-1)
DTLZ3	3	50	3.6714e+3 (2.36e+2) -	2.9065e+3 (1.17e+2) =	2.9397e+3 (3.88e+2)
	6	50	3.5383e+3 (1.36e+2) -	2.9727e+3 (3.77e+2) =	2.6992e+3 (3.62e+2)
	10	50	2.9573e+3 (2.84e+2) -	2.5085e+3 (3.14e+2) =	2.5654e+3 (1.33e+2)
DTLZ4	3	50	3.0280e+0 (1.47e-1) -	3.0394e+0 (6.90e-1) -	1.7958e+0 (1.60e-1)
	6	50	3.0963e+0 (8.62e-2) -	2.5039e+0 (4.28e-1) =	1.9357e+0 (2.20e-1)
	10	50	2.9678e+0 (1.27e-1) -	2.3494e+0 (3.00e-1) -	1.8797e+0 (1.85e-1)
DTLZ5	3	50	2.7406e+0 (7.93e-2) -	1.8105e+0 (1.72e-1) =	1.7757e+0 (2.30e-1)
	6	50	2.5604e+0 (4.98e-2) -	2.3764e+0 (1.07e-1) -	1.9455e+0 (1.79e-1)
	10	50	2.2218e+0 (1.00e-1) -	2.0765e+0 (2.38e-1) -	1.4635e+0 (1.32e-1)
DTLZ6	3	50	4.2025e+1 (1.28e-1) -	3.6292e+1 (1.22e+0) =	3.7283e+1 (1.02e+0)
	6	50	3.9234e+1 (7.46e-2) -	3.7493e+1 (2.81e-1) =	3.7116e+1 (1.06e+0)
	10	50	3.5415e+1 (6.12e-1) -	3.4897e+1 (4.78e-1) -	3.3008e+1 (6.95e-1)
DTLZ7	3	50	9.3479e+0 (1.57e-1) -	4.3439e+0 (8.56e-1) =	5.0063e+0 (9.86e-1)
	6	50	1.9461e+1 (7.11e-1) -	1.2811e+1 (1.37e+0) +	1.5969e+1 (1.34e+0)
	10	50	3.2668e+1 (2.30e+0) =	2.0137e+1 (3.46e+0) +	3.2106e+1 (2.45e+0)
+/-/=			0/20/1	2/7/12	

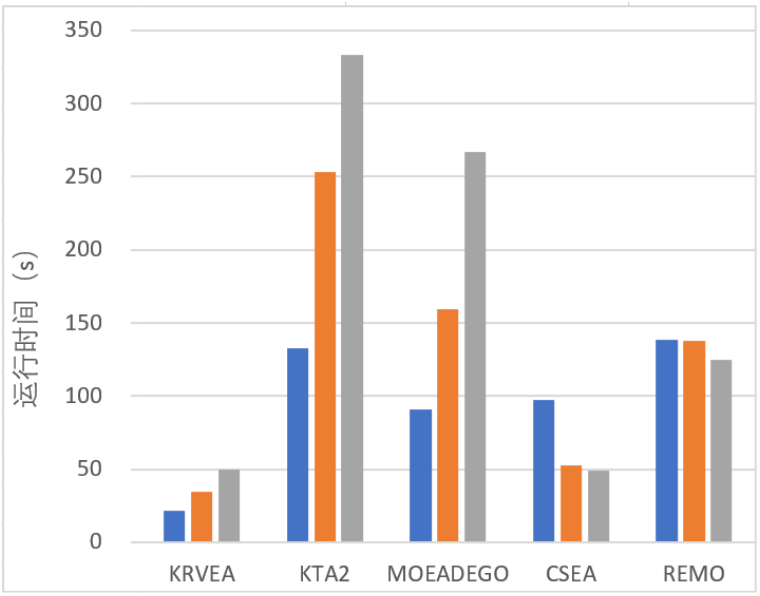

图 6. m=3,6,10 时，5 种算法在 DTLZ1-7 上的平均运行时间柱状图。

4.2 创新点

	NN1	NN2	NN3	NN4	NN5
Structure	$[3n, 2n]$	$[2n, n]$	$[3n]$	$[n]$	$[3n, 2n, n]$

如前表所示针对模型所使用的神经网络，原文中测试了五种不同的隐藏层结构下的模型性能，最终选择了性能最好的 NN5 结构作为默认设置，即三个隐藏层神经元数量分别为 3n，2n 和 n。为了实现本算法的最佳性能，我额外测试了三种网络结构，如下表：

	NN6	NN7	NN8
Sructure	[2n]	[4n,2n]	[4n,2n,n]

初步实验结构表明，NN6 在高维问题上 (D=50) 与 NN5 具有竞争力，于是进一步将二者进行对比试验：

TABLE III: The statistical results of IGD metric values and runtime on 21 test instances over 10 independent runs. The best mean results in each row are highlighted.

			IGD		runtime	
Problem	M	D	REMO(NN5)	REMO(NN6)	REMO(NN5)	REMO(NN6)
DTLZ1	3	50	9.3364e+2 (7.77e+1) +	9.8511e+2 (7.04e+1)	5.8119e+1 (5.78e+0) +	8.8453e+1 (2.62e+1)
	6	50	7.0363e+2 (5.72e+1) =	6.8396e+2 (9.16e+1)	6.5754e+1 (1.28e+1) +	7.9319e+1 (1.92e+1)
	10	50	6.0122e+2 (7.51e+1) +	6.3296e+2 (8.87e+1)	5.6561e+1 (1.17e+1) +	7.5619e+1 (1.61e+1)
DTLZ2	3	50	1.9708e+0 (2.96e-1) -	1.8043e+0 (1.69e-1)	7.1022e+1 (2.05e+1) -	6.7357e+1 (6.08e+0)
	6	50	2.1105e+0 (2.22e-1) -	2.0095e+0 (1.35e-1)	9.6890e+1 (1.42e+1) -	7.2599e+1 (8.31e+0)
	10	50	1.9052e+0 (2.74e-1) =	1.9406e+0 (1.93e-1)	8.6351e+1 (1.23e+1) -	6.0295e+1 (7.16e+0)
DTLZ3	3	50	2.8643e+3 (2.32e+2) -	2.7772e+3 (2.42e+2)	9.2382e+1 (1.16e+1) -	7.0499e+1 (8.48e+0)
	6	50	2.6852e+3 (3.20e+2) =	2.7230e+3 (2.46e+2)	9.1654e+1 (5.74e+0) -	7.5635e+1 (9.93e+0)
	10	50	2.4553e+3 (2.06e+2) -	2.3710e+3 (2.70e+2)	1.0879e+2 (2.20e+1) -	7.9274e+1 (1.60e+1)
DTLZ4	3	50	1.7486e+0 (1.97e-1) =	1.7386e+0 (1.88e-1)	7.8196e+1 (8.90e+0) -	6.5482e+1 (8.16e+0)
	6	50	1.8956e+0 (1.81e-1) =	1.9446e+0 (1.07e-1)	8.8179e+1 (1.34e+1) -	6.4374e+1 (7.14e+0)
	10	50	1.9387e+0 (2.02e-1) =	1.8687e+0 (1.90e-1)	8.4661e+1 (7.14e+0) -	6.7019e+1 (6.45e+0)
DTLZ5	3	50	1.8282e+0 (2.83e-1) =	1.8631e+0 (2.56e-1)	8.5180e+1 (5.38e+0) -	6.0960e+1 (5.01e+0)
	6	50	1.8820e+0 (2.57e-1) +	1.9844e+0 (2.23e-1)	8.9682e+1 (1.46e+1) -	4.8735e+1 (2.40e+0)
	10	50	1.6398e+0 (1.92e-1) =	1.6019e+0 (2.23e-1)	8.3718e+1 (8.17e+0) -	5.3657e+1 (6.22e+0)
DTLZ6	3	50	3.8800e+1 (7.34e-1) =	3.8904e+1 (1.08e+0)	1.2912e+2 (2.59e+1) -	8.6100e+1 (2.06e+1)
	6	50	3.6484e+1 (1.36e+0) =	3.7484e+1 (8.36e-1)	1.7306e+2 (1.56e+0) -	1.0874e+2 (7.97e-1)
	10	50	3.4002e+1 (8.56e-1) =	3.3821e+1 (5.13e-1)	1.7155e+2 (2.77e+0) -	1.0759e+2 (2.38e+0)
DTLZ7	3	50	5.1311e+0 (1.17e+0) -	4.9721e+0 (8.46e-1)	8.3807e+1 (1.12e+1) -	4.9317e+1 (3.47e+0)
	6	50	1.5873e+1 (1.27e+0) =	1.5923e+1 (1.35e+0)	7.4593e+1 (5.78e+0) -	5.0168e+1 (3.26e+0)
	10	50	2.9488e+1 (1.87e+0) =	2.9545e+1 (2.30e+0)	7.8454e+1 (6.59e+0) -	5.2336e+1 (4.28e+0)
+/-/=			3/5/13		3/18/0	

5 实验结果分析

TABLE I 中的结果与原文结果近似，REMO 的表现远远优于除 K-RVEA 和 KTA2 以外的其他算法。

由于本结果是在 D=10 的低维问题下得出的，为验证本算法在较高维问题上的表现，进一步将 REMO 与 K-RVEA,KTA2 在高维 (D=50) 问题上进行对比试验，结果在 TABLE II 中列出；可以看出，在高维问题上，REMO 的表现要由于另外两个算法，这与原文中的实验结果相符。

图6为五个 SAEAs 完成 300 FEs 所需的运行时间对比，结果显示虽然 REMO 的运行速度不是最优的，但随目标数增加，REMO 完成运行所需的时间无明显变化，证明 REMO 可以很好的应对多目标问题。

TABLE III 中的结果表明，虽然在 IGD 指标上 (衡量最终种群的收敛性和多样性的指标) 我所采用的 NN6 网络结构与原文的默认设置 NN6 对比无显著优势，但采用 NN6 网络结构的 REMO 运行时间在大部分问题上优于原 REMO，证明对于高维问题，使用大小为 2n(n 为决策向量维度) 的单隐藏层神经网络结构优于原文默认使用的三隐藏层神经网络结构。

6 总结与展望

本文介绍一种基于关系学习和预测的昂贵多目标优化方法，并根据文字提供的代码复现了论文中的实验，在此基础上对原文方法进行了部分改进。
本文第一部分引出了论文的相关背景，许多代理辅助算法 SAEAs 被提出用于解决昂贵多目标优化问题，本文基于分类的想法提出了一种基于关系选择 (RS) 的模型；第二部分介绍了昂贵多目标优化问题和二元关系学习的思想；第三部分先介绍了 REMO 方法的框架，然后详细

介绍了该算法的方法细节；第四部分对本文方法进行复现，并通过调整模型所使用的神经网络结构取得了更好的实验结果。第五部分是对实验结果的分析。
未来，我计划以我从这篇论文中获得的灵感设计一种新的基于分类的代理辅助进化算法，并确保其应用于昂贵多目标优化问题的性能优于当前最新的代理辅助进化算法。

参考文献

[1] Y. Jin X. Zhang C. He, Y. Tian and L. Pan. A radial space division based evolutionary algorithm for many-objective optimization. *Applied Soft Computing*, 61:603–621, 2017-12.

[2] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011-06.

[3] Y. Tian H. Wang X. Zhang L. Pan, C. He and Y. Jin. A classificationbased surrogate-assisted evolutionary algorithm for expensive manyobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(1):74–88, 2019-02.

[4] S. Ruette S. Gelly and O. Teytaud. Comparison-based algorithms are robust and randomized algorithms are anytime. *Evolutionary Computation*, 15(4):441–434, 2007-12.

[5] J. Hakanen T. Chugh, K. Sindhya and K. Miettinen. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23(8):3137–3166, 2019-05.

[6] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007-12.