Evolutionary Multitasking for Feature Selection in High-Dimensional Classification via Particle Swarm Optimization

Ke Chen, Member, IEEE, Bing Xue , Senior Member, IEEE, Mengjie Zhang , Fellow, IEEE, and Fengyu Zhou

摘要

在许多实际应用中，特征选择是提高特征集质量的重要预处理技术。粒子群算法以其高效、易于实现等优点被广泛应用于求解 FS。然而，现有的基于 pso 的 FS 方法在处理高维数据时，大多面临陷入局部最优和计算成本高的问题。进化多任务处理是一种通过相关任务间的知识转移来增强全局搜索能力和加速收敛的有效范式。受进化多任务处理的启发，本文提出了一种用于高维 FS 的多任务粒子群算法。该方法将一个高维 FS 任务转化为多个相关的低维 FS 任务，然后通过低维 FS 任务间的知识转移来寻找最优特征子集。具体而言，提出了一种基于特征重要性的任务生成策略，可以自适应地从数据集中生成高度相关的任务。此外，提出了一种新的知识转移机制，可以有效地实现相关任务之间的正向知识转移。结果表明，该方法能在较短的时间内进化出具有较高分类精度的特征子集。

关键词：进化多任务处理；特征选择 (FS)；高维分类；知识转移；粒子群优化 (PSO)。

1 引言

分类是学术界和工业界广泛研究的一项数据挖掘任务，目的是根据不可见数据的特征所呈现的信息对其进行分类。随着新兴技术的快速发展，高维数据在许多实际应用中变得普遍，例如作业车间调度和文本分类。在这样的数据中，许多特征 (即不相关的和冗余的特征) 对类预测没有用处，它们不仅增加了学习算法的计算复杂度，而且由于“维数的诅咒”而降低了分类性能。特征选择 (FS) 是一种有效的数据预处理技术，它能够通过从原始特征中选择相关特征的子集来解决这一难题。

通常，现有的 FS 方法可以根据不同的评价标准分为三类，即基于过滤器的方法、基于包装器的方法和基于嵌入式的方法。基于过滤器的方法使用数据的内在度量 (即，距离，相关性，一致性，信息增益) 来评估和确定选定的特征。基于过滤器的方法通常需要较少的计算资源，但通常获得较低的分类精度比基于包装器的方法。基于包装器的方法使用一种学习算法来评估所选的特性。由于特征子集是由学习算法决定的，基于包装器的方法通常可以获得比基于过滤器的方法更高的分类精度。然而，基于包装器的方法在计算上通常比基于过滤器的方法更昂贵，因为它们需要更多的计算资源来评估每个选定的特征子集。基于嵌入式的方法将 FS 融入到模型训练过程中，并与特定的分类算法相关联。一般来说，基于包装器和基于嵌入式的方法通常可以获得比基于滤波器的方法更好的分类精度，但代价是计算成本较高，泛化较少。

FS 是一个组合优化问题，有 2^n 个可能的特征组合，其中 n 表示数据集中所有可用特征的数量。因此，几乎不可能进行穷尽搜索来找到最好的特征子集，特别是当特征的数量很大的时候。大量的启

发式搜索方法 (如顺序前向或后向选择、顺序前向或后向浮动选择) 已被用于解决这一问题。遗憾的是，这些方法的主要缺陷是它们往往在 FS 过程中获得局部最优解。

近年来，基于进化计算 (EC) 的 FS 方法因其能够通过全局搜索策略在合理时间内找到满意解而受到广泛关注。粒子群优化 (PSO) 是一种基于群智能的 EC 算法。与许多其他 EC 算法相比，粒子群算法具有高效、易于实现、调整的参数较少等优点。此外，粒子群算法已被广泛应用于求解 FS 问题，其有效性已得到验证。因此，本研究采用粒子群算法作为一种有效的搜索技术。迄今为止，在某些论文中已经开发了许多基于 pso 的 FS 方法。然而，由于特征数量的快速增长，高维问题中的粒子群算法给粒子群算法带来了新的挑战。例如，其中一篇论文开发了一种基于 PSO 的无监督 FS，在 PSO 中引入了两种策略 (即空间缩减策略和局部过滤器搜索策略) 来选择特征子集。但是，由于在搜索过程中忽略了特征之间的相互作用，该方法仍然面临陷入局部最优的问题。在另一篇论文中，提出了一种基于 pso 的多目标 FS 方法，采用三种度量 (即模糊优势关系、模糊拥挤距离和容忍系数) 来确定 pareto 最优特征子集。但由于这三个度量的计算需要更多的计算资源，该方法仍然面临计算成本高的问题，特别是特征的数量较多。因此，粒子群算法解决高维 FS 问题的潜力还有待进一步探索。

进化多任务处理是 EC 领域一个新兴的研究课题。进化多任务处理的主要特点是将不同任务的知识相互共享，从而同时解决多个优化问题。进化多任务处理已被广泛用于解决实践中不同的优化问题 (例如，旅行推销员问题，作业车间调度问题和参数提取问题)。这是因为进化多任务处理有望通过相关任务之间的知识转移，增强算法的搜索能力，加速收敛，缩短计算时间。在以往的工作中，已有方法可以通过分析适应度情景的特征，或者简化原来研究的问题来生成相关的任务进行多任务优化。然而，基于健身景观特征的任务生成方法并不适用于 FS，因为健身景观的信息是不可用的。而基于简化原问题思想的任务生成方法，对所研究的问题 (如车辆路径问题、信号重构问题) 具有较强的针对性。本文提出构造具有不同特征子集的多个相关任务，形成多任务优化问题。换句话说，从原始特征的不同集合中选择特征可以被视为一个多任务处理问题，因为它们共享来自同一数据集的公共特征。例如，如果一个数据集包含十个特征 $\{F_1, F_2, \cdots, F_{10}\}$，从特征集 $\{F_1, F_2, F_3, F_5, F_6, F_8, F_{10}\}$ 或从特征集 $\{F_1, F_3, F_4, F_5, F_7, F_{10}\}$ 中选择特征的 FS 任务，可以被认为是多任务场景中的相关任务。

与以前的 FS 方法相比，本文的主要贡献可以描述如下。

1) 提出了一种新的基于 pso 的高维数据 FS 多任务处理方法。通过几个相关的低维 FS 问题的进化多任务处理来解决高维 FS 问题的思想显著增强了粒子群算法在处理高维数据时的可扩展性。

2) 根据高维数据的特点，设计一种新的相关任务生成策略。该策略可以自适应地从高维 FS 问题生成几个低维相关的 FS 任务，而不会丢失太多关于类标签的信息。

3) 建立有效的知识传递机制，在相关任务之间共享信息。在该策略中，应用来自其他相关任务的知识来帮助当前任务跳出局部最优，从而提高当前任务解决方案的质量。

2 相关工作

2.1 特征选择

假设一个标记的数据集包含 N 个样本和 m 个特征 (或属性)。因此，FS 问题可以表示为：从原始特征集中选取 G 个特征 (G<M)，优化给定性能评价函数 F（•）的值。例如，在分类问题中，F（•）通

常表示分类错误率。FS 过程中，解 X 可表示为:

$$\begin{aligned} &X = (x_1, x_2, ..., x_M) \\ &x_m \in 0, 1 \quad \forall m \in 1, 2, ..., M \end{aligned} \tag{1}$$

其中 x_m = 1 表示应选择第 m 维的特征; 否则，应该放弃相应的特征。因此，一个具有 M 特征的 FS 问题可以用以下公式描述:

$$\begin{aligned} &minF(x) \\ &s.t.\ X = (x_1, x_2, ..., x_M) \\ &x_m \in 0, 1 \quad \forall m \in 1, 2, ..., M \end{aligned} \tag{2}$$

2.2 进化多任务优化

进化多任务是优化和 EC 领域一个比较新的研究课题，它的提出是通过在进化过程中同时优化多个相关任务来提高全局搜索性能。在多任务优化框架中，解决一个问题可能有助于解决其他问题。这是因为一些有用的知识在他们的问题之间是互补或共享的。数学上，一个有 K 个任务的多任务优化框架可以描述如下:

$$\begin{aligned} &\{x_1^*, x_2^*, ..., x_K^*\} \\ &= \{argminf_1(x_1), argminf_2(x_2), ..., argminf_K(x_K)\} \end{aligned} \tag{3}$$

其中 x_K^* 表示任务 k 的最优解。

多因子优化 (MFO) 是进化多任务处理中的一个新范式，它通过基于群体的优化算法的隐式并行性，有效地搜索多个决策空间，同时处理多个相关任务。在 MFO 中，不同的任务通过统一的随机密钥方案编码到统一的搜索空间中。关于 MFO 的更多细节可以在中找到。

通过扩展多因子进化算法 (MFEA)，提出了求解不同优化问题的若干进化多任务算法。例如，为了进行进化多任务处理，有学者开发了一种改进的 MFEA 算法，其中基于排列的统一表示和基于分裂的解码算子被应用到 MFEA 算法中，以解决 NP-hard 有能力车辆路由问题。在另一篇文献中，利用具有多种群进化框架的 MFEA 变量求解数值优化问题。此外还提出了一种多因子遗传规划 (MFGP)，设计了一种新的可扩展染色体编码方案和一种新的进化机制来解决回归问题。然而，它们大多还停留在基准函数上，只有少数尝试用于解决实际问题。在本研究中，我们旨在发展一种新的多任务优化方法来解决高维 FS 问题。

2.3 粒子群优化

在粒子群优化中，每个粒子表示问题的一个候选解。在 D 维搜索空间中，第 t 次迭代的第 i 个粒子有两个向量引导搜索，分别是位置向量 $X_i^t = [X_{i1}^t, X_{i2}^t, ..., X_{iD}^t]$ 和速度矢量 $V_i^t = [V_{i1}^t, V_{i2}^t, ..., V_{iD}^t]$。在运动过程中，每个粒子的位置和速度根据两个位置 (或解) 进行更新，一个是自身获得的个人最佳位置，用 $pbest_i = [pbest_{i1}, pbest_{i2}, ..., pbest_{iD}]$，另一个是整个种群获得的全局最佳位置用 $gbest_i = [gbest_{i1}, gbest_{i2}, ..., gbest_{iD}]$。根据 pbest 和 gbest，第 i 个粒子在 (t+1) 次迭代时的速度和位置按以下公式更新:

$$V_{id}^{t+1} = \omega * V_{id}^{t} + c_1 * r_1 * (pbest_{id}^{t} - X_{id}^{t}) + c_2 * r_2 * (gbest_{d}^{t} - X_{id}^{t}) \tag{4}$$

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1} \tag{5}$$

其中 t 为当前迭代次数;ω 表示惯性权重;c_1 和 c_2 分别表示加速常数，称为认知参数和社会参数;r_1 和 r_2 是在区间 [0,1] 上的均匀分布值。

2.4 基于 pso 的高维数据分类算法

在过去的几十年里，研究人员为高维分类问题开发了大量基于 pso 的 FS 方法。在这些方法中，粒子群算法结合了不同的搜索策略，增强了所选特征子集对高维分类数据的辨别能力。

在求解高维分类问题时，先缩小搜索空间再使用粒子群算法已成为一种流行方法。其中，针对 FS 提出了一种变尺度协同协同进化 PSO (VS-CCPSO) 方法。该算法设计了一种空间划分策略，首先根据数据的特征将一个大的特征空间划分为多个小的子空间。然后，利用多群粒子群算法对这些特征子空间进行协同搜索。最后，采用基于适应度的精英组合策略，从这些子群中构造一个完整的特征子集。结果表明，该方法在分类精度和选择特征数量方面优于其他比较算法。然而，该方法在演化过程中由于忽略了不同特征子空间之间的相互作用，有可能陷入局部最优。

在粒子群算法中，根据特征间的关联信息设计特定的搜索策略是避免搜索过程中无效搜索的有效方法。在中，针对高维分类问题的 FS，提出了一种变长粒子群优化算法。该方法提出了一种基于特征相关性的变长度表示策略，以提高粒子群算法在不同搜索空间中的搜索能力。实验结果表明，该策略能有效提高粒子群算法在搜索过程中的有效性。然而，这种具有不同长度表示的策略可能会在进化过程中忽略一些有潜力的区域。

利用特征排序方法去除无关和冗余特征，提高粒子群算法在高维数据中的可扩展性。其中，针对高维基因分类问题，提出了一种滤波和包装方法的混合模型。该方法采用基于相关的滤波方法确定初始特征子集，然后应用粒子群算法选择来自初始特性子集的特性。实验结果表明，该方法在分类精度和计算时间方面优于其他比较方法。然而，仅仅基于每个特征与类标签的相关性是不足以确定最佳初始子集的，因为它们忽略了特征的相互作用。

开发局部搜索策略，挖掘有潜力的区域，是提高粒子群算法在高维问题上搜索能力的有效途径。其中，针对高维 FS 问题，提出了具有 pbest 局部搜索机制的粒子群算法。该方法将局部搜索机制作为基于 pso 的全局搜索的补充搜索机制，利用个体最佳位置附近的空间。结果表明，在大多数情况下，该策略能显著提高高维 FS 问题的分类精度。然而，这种 FS 方法通常需要更多的计算资源来增强粒子的局部搜索，特别是在搜索空间较大的情况下。

综上所述，尽管许多基于 pso 的方法已经被提出，并在高维分类问题上显示出了前景，但高维数据的 FS 由于计算成本高，容易陷入局部最优，仍然是一个开放的挑战。多任务优化已经在许多具有挑战性的问题上显示出了前景。在本研究中，我们研究了如何使用多任务优化来提高 FS 在高维数据上的性能。

3 本文方法

3.1 本文方法概述

进化多任务处理是一种通过在多个相关任务之间进行知识转移来处理复杂问题的有效技术。受此启发，我们利用多任务优化方法的优点，设计了一种新的 FS 方法 MTPSO，将高维 FS 分类作为一个多任务问题来解决。

MTPSO 的主要框架如图 1所示。输入是训练数据，包括所有可用的特征。输出是由多个相关任务的子种群获得的特征子集。MTPSO 由两个部分组成，即根据高维数据的特征生成多个相关 FS 任务，并通过知识转移解决这些相关 FS 任务。当接收到高维数据时，首先采用特征排序的方法来衡量特征的重要性，然后根据特征的重要性，采用一种生成多个相关 FS 任务的方法。然后，为每个 FS 任务分配一个子种群，利用本文提出的多任务优化方法通过知识转移对相关 FS 任务进行搜索，直到满足停止条件。最后从多个特征子集中选取分类精度最高的特征子集作为最终特征子集。如果多个特征子集达到相同的最高分类精度，则选择所选特征数量最少的特征子集作为最终特征子集。

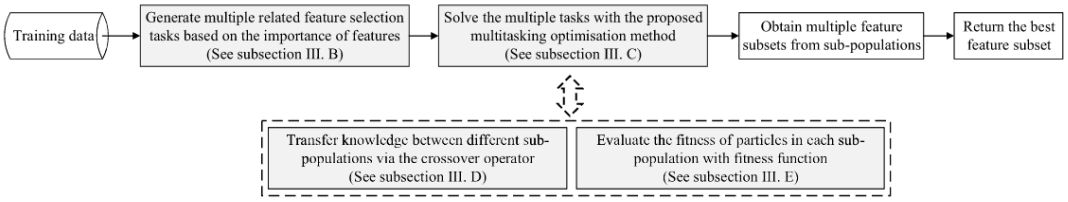

图 1: 算法的基本框架

本研究共涉及四个研究问题。第一个问题是如何开发一种策略来生成相关的 FS 任务，同时又不会丢失太多关于类标签的信息。二是如何通过多任务优化设计一种有效处理多个相关 FS 任务的方法。三是如何在多任务优化方法中设计有效的知识传递机制，实现不同任务间的正向信息共享。最后是如何确定一个有效的适应度函数来准确评价所选特征子集。解决这四个研究问题的方法细节将在下一节中描述。

3.2 任务生成策略

在 FS 任务中，有用的信息主要是特征之间的相关性和特征之间的相互作用。在该方法中，任务是根据特征的相关性来生成的，即高度相关的特征被选择形成任务 (作为候选特征集) 的概率很高。然而，由于特征的相互作用，最优特征子集通常是一组互补的特征，包括高度相关和弱相关的特征。因此，弱相关特征也有机会 (低概率) 被选择来保存任务中的特征交互信息。虽然不是原始任务的所有信息都被保存，但最重要的信息很可能被保存在不同的任务中。在那些生成的任务上使用进化多任务处理有望提高性能。生成任务策略的详细描述如下。

首先，我们使用 ReliefF 评估特征的相关性信息。这是因为 ReliefF 比其他相关度量方法能更好地确定变量之间的非线性关系，它通过为每个特性分配权重已被广泛采用来确定特征对类标签的重要性。在 ReliefF 中，根据特征对近距离样本的区分能力确定权重。在使用 ReliefF 处理多类问题时，首先从训练数据集中随机选择一个样本 R，然后从相同和不同的类中寻找样本 R 的 h 个最近邻，分贝记为 H_j(j = 1,2，...，h) 和 M_j(c) (j = 1,2,...,h)。利用式 (6) 计算特征 a 的权重，权重越大，表示特征越重要。

$$W(a)^{t+1} = W(a)^t - \sum_{j=1}^{h} f(a, R, H_j)/(u * h) + \sum_{c \notin s(R)} [\frac{p(c)}{1 - p(s(R))} * \sum_{j=1}^{h} f(a, R, H_j(c))]/(u * h) \tag{6}$$

其中 W(a) 为特征 a 的权重;T 为当前迭代次数;u 为采样次数;s(R) 为样品 R 的类标签;p(c) 和 p(s(R)) 分别表示 c 类和 R 类的比例。$f(a, R_1, R_2)$ 表示样本 R_1 和 R_2 的特征值 a 之差。

其次，我们提出了一种基于特征重要性的膝点选择策略，将所有可用特征划分为两个特征集 (即前景特征集和剩余特征集)。Das 首先提出了用到极值线的最大距离来表征膝点的方法，该方法在许多优化问题中被广泛应用于确定分界点。受此启发，我们提出了一种基于特征权重的膝点选择策略，该策略可以减少所提方法中的一个参数 (即划分特征的阈值)。将所有特征划分为两个集合的伪代码如算法 1 所示。在膝点选择策略中，特征根据由 ReliefF 计算 (从第 1 和第 2 行) 出来的权重降序排序，然后得到关于权重的曲线。接下来，在具有最大和最小特征权重的两点之间生成一条线 (L)(从第 3-6 行开始)。计算曲线上每个点与 L 之间的距离，并检测到距离 L 最大的点为膝点 (从第 7-13 行)。图 2是一个例子来说明膝盖点的确定，其中数据集有 100 个特征。最后，选择权重大于膝关节点权重的特征形成有希望的特征集 (即 ProFeatures)，其余的特征被分配到剩余的特征集 (即 RemFeatures)(从第 14-21 行)。

Procedure 1 Knee Point Selection Strategy

Input: The all available features.
Output: Two feature sets *ProFeatures* and *RemFeatures.*
Calculate feature weights *W* using *ReliefF.*
  Sort *W* based on descending order.
  maxPoint (1,*W* (1)).
  minPoint (*numFeature,W*(*numFeature*)).
  Set *maxDistance* ← 0 and *kneePoint* ← 0.
  Get a line *L* based on *maxPoint* and *minPoint* points.
  for *i = 1 to numFeature* **do**
    Calculate distance *d* from *Point*(*i, W*(*i*)) to line *L.*
     if *d* > *maxDistance* **then**
      maxDistance ← *d.*
       kneePoint ← *i.*
    end
end
for *i = 1 to numFeature* **do**
    if *i* <= *kneePoint* **then**
      ProFeatures ← *Feature(i).*
    end
    else
      RemFeatures ← *Feature(i).*
    end
end
return *ProFeatures, RemFeatures.*

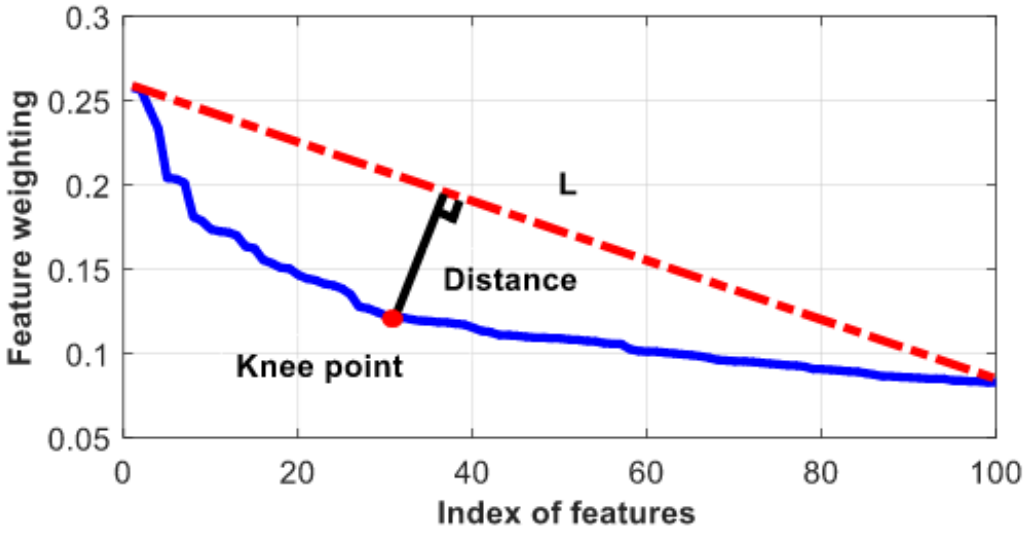

图 2: 确定膝点的例图

第三，基于有前途特征集和剩余特征集制定任务生成策略，形成相关的 FS 任务。直观地说，有希望的特征集比剩下的特征集更有利于 FS。由于 ReliefF 不能有效地去除冗余的特征，因此有希望的特征集仍然包含冗余的特征。此外，由于特征之间的交互，剩余特征集中的一些特征是有用的。因此，利用概率从有前途的特征集中和剩余的特征集中选择特征，形成相关的 FS 任务。在该策略中，利用有前途特征集和剩余特征集的平均特征权重来确定选择概率。选择概率的计算公式如下:

$$p = \frac{W_p}{W_p + W_r} \tag{7}$$

其中 W_p 和 W_r 分别表示有希望特征集和剩余特征集中特征的平均权重。

假设一个数据集有 n 个特征，其中 m 个特征属于有希望的特征集。图 3显示了使用所提出的任务生成策略生成 FS 任务的过程。如果特征在有希望的特征集中，当条件 rand <= p 成立时，相应的特征被选择到 task 中，否则不选择。如果特征在剩余的特征集中，当条件 rand < (1−p) 被激活时，相应的特征被选中到 Task 中，否则不选中，其中 rand 表示 [0,1] 之间的随机值。通过重复上述过程 T 次，可以获得 T 个不同的 FS 任务。由于选择概率是通过考虑特征的相关性来确定的，因此单个有用的特征更有可能被保留。特别是, 有前途的特征集中的特征在生成的 FS 任务中有很高的概率被选中，而剩余特征集中的特征仍然有机会被选中。该策略通过将巨大的搜索空间转化为多个小而有前景的空间 (即多个相关任务)，避免浪费计算资源去探索不需要的区域 (例如特征不相关的区域)。

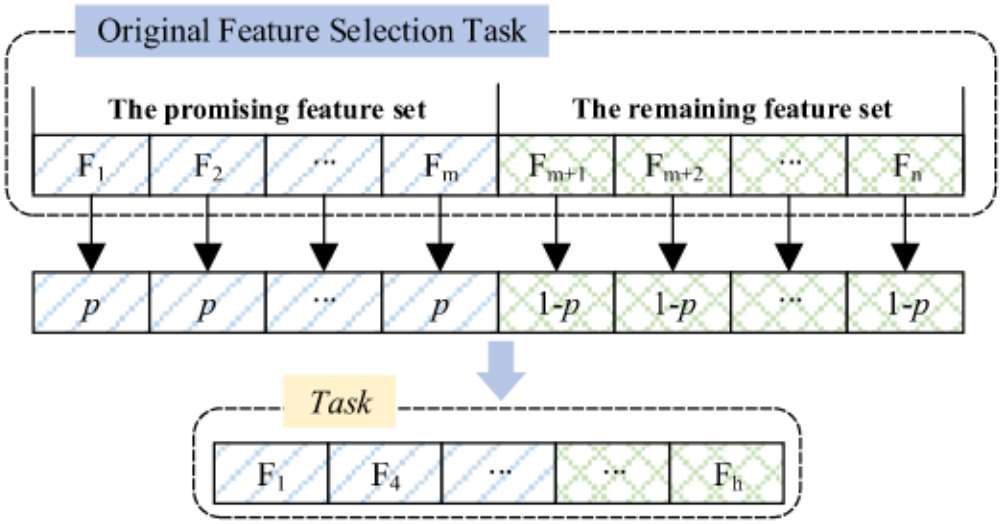

图 3: 通过提出的任务生成策略生成任务的过程

虽然不能保证最优特征子集包含在这些生成的相关 FS 任务中，但由于以下原因，提出的方法仍然很有可能找到最优特征子集。首先，由于冗余特征和特征交互的存在，FS 问题往往有多个最优解。其次，在提出的任务生成策略中，高相关性和弱相关性特征都有机会被包含到生成的 FS 任务中，其中高相关性特征更有可能形成最优特征子集，更有可能被包含。第三，在这些相关的任务中候选特征之间存在重叠，因此可以大部分保留和发现有用特征之间的重要交互。最后但并非最不重要的是，这些相关的 FS 任务并不是独立求解的，而是在相关任务之间共享有用的信息，以利用多任务处理的思想来提高 FS 性能。

3.3 多任务优化方法

多种群是一种将整个种群划分为多个亚种群，同时搜索多个不同区域的有效技术。提出了多种群结合不同 EC 算法求解多任务优化。基于多种群技术，提出了一种求解高维 FS 问题的多任务优化方法。在该方法中，每个子种群对应一个任务。与其他多种群多任务优化方法不同，设计了一种新的知识传递机制来实现不同任务 (子种群) 之间的通信。所设计的知识转移机制的细节将在下一节中介绍。由于粒子群算法具有较强的全局搜索能力和对 FS 的自然表示，提出了基于粒子群算法的多任务优化方法。

图 4显示了所提出的多任务优化方法的演化过程，包括初始化和演化两个阶段。假设生成 T 个具有不同特征空间的 FS 任务 ($Task_1$,$Task_2$,...,$Task_T$)，通过这些相关 FS 任务之间的知识转移，为所需的高维分类问题找到最优特征子集。在初始化过程中，为 FS 任务生成子种群 (例如,$Subpop_1$,$Subpop_2$,...,$Subpop_T$)，每个子种群的目标是处理其对应的 FS 任务。在进化过程中，位于不同亚种群中的 FS 任务会通过知识转移相互帮助。因此，提出的多任务优化方法将输出 T 个特征子集 (例如，$BestFS_1$, $BestFS_2$,..., $BestFS_T$) 来自 T 亚群体。

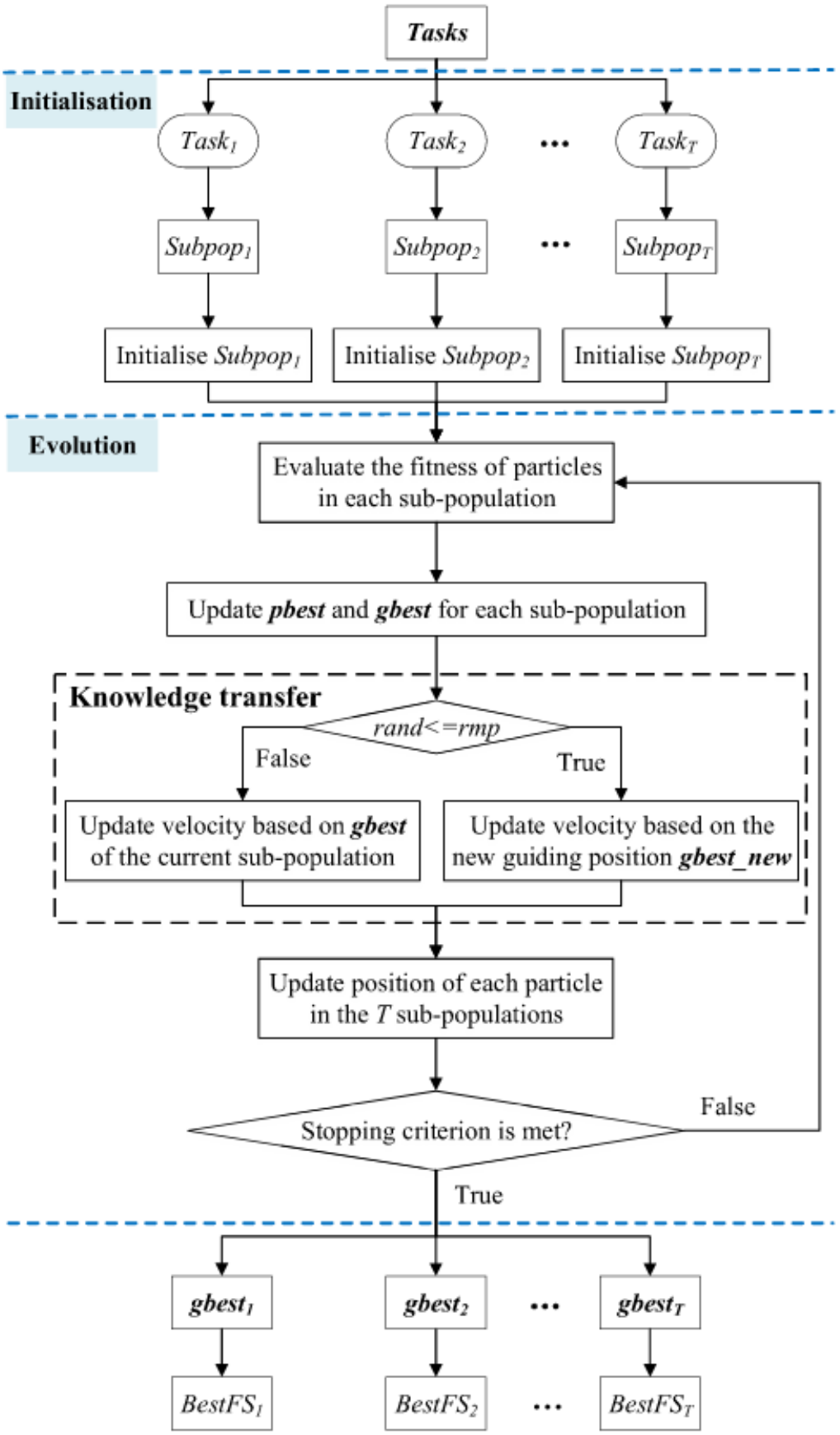

图 4: 多任务处理方法的演化过程

使用所开发的多种群多任务优化方法来实现高维分类问题的 FS 有几个潜在的好处。首先，所开发的多任务处理框架可以通过多个相关 FS 任务之间的知识转移来解决高维分类问题。其次，利用来自其他具有不同特征空间 FS 任务的知识，帮助子种群跳出可能的局部最优，使当前 FS 任务能够找到更好的特征子集。第三，这种处理高维分类的方法可以大大减少计算时间，因为这些子种群的搜索空

间比原来的搜索空间小得多。

3.4 知识转移机制

粒子群优化算法是一种有效的基于群体的优化算法，通常用于求解单一任务。多任务处理是同时解决多个任务的有效方法。因此，在为多任务扩展粒子群算法时，需要考虑两个关键问题。首先是如何确定哪些知识可以在不同的任务之间传递。二是如何在不同任务之间实现有效的知识转移。针对上述问题，本研究提出了一种新的知识传递机制，用于相关任务之间的信息共享。具体情况如下。

如何转移: 将知识转移到其他任务的方式对最终解决方案的质量有很大影响。如果采用较差的转移算子来实现相关任务之间的信息共享，将降低进化过程中利用知识转移提高搜索质量的目的。交叉算子是一种重要的算子，广泛应用于多任务优化实现信息共享，其有效性已被证明。此外，利用交叉技术实现知识转移不会增加太多额外成本。在本研究中，使用粒子位置的交叉算子作为不同任务 (即子种群) 之间的知识转移方式。跨界共享信息不仅可以实现不同任务之间的知识转移，还可以增强搜索过程中的多样性。

何时转移: 在进化过程中，我们定义一个随机配对概率 (rmp) 来控制何时从其他任务转移知识。较大的 rmp 鼓励从其他任务中转移知识，反之亦然。在每一代中，如果知识转移策略被激活 (rand<=rmp)，来自其他任务的知识将被用于协助更新粒子的位置。否则 (rand>rmp)，粒子的位置将由来自当前任务的知识更新。

转移什么: 确定什么样的知识有利于共享是至关重要的。在粒子群算法中，全局最优位置所携带的知识在速度更新中起着重要的作用，在搜索过程中，它被用来影响每个粒子的搜索方向。因此，我们利用全局最优位置上的交叉来实现不同任务之间的知识转移。具体而言，对于 $Task_t$ 中的每个粒子，利用竞赛选择机制从其他子种群的所有 gbest 中选择一个 gbest，并将选中的 gbest 记录为 $gbest^m$。基于这两个全局最优解，通过算术交叉生成一个新的引导位置。对于 $Task_t$，生成新的导向位置的过程如下所示:

$$gbest\_new^t = cr * gbest^t + (1 - cr) * gbest^m \tag{8}$$

其中，$gbest^t$ 表示 $Task_t$ 当前全局最优位置。cr 表示随机向量，各维范围在 0 到 1 之间。

在进化过程中，如果 $gbest^t$ 在给定代 (G) 内不发生变化，$Task_t$ 中的 PSO 可能陷入潜在的局部最优。在本研究中，我们提出了一个新的算子来生成新的引导位置 ($gbest\_new^t$) 基于 T 个全局最优位置的平均值。该操作符被期望找到一个新的搜索区域，从而增加了在搜索过程中逃离局部最优的可能性。式 (9) 为 $Task_t$ 的该算子公式。

$$gbest\_new^t = \frac{1}{T} * \sum_{i=1}^{T} gbest^i \tag{9}$$

其中 $gbest^i$ 表示 $Task_i$ 中到目前为止的全局最优位置。T 表示 Task 的个数。

这样，不同任务中的粒子就有机会通过 gbest 互相分享知识。在进化过程中，如果一个随机事件 (rand<=rmp) 被激活，gbest_new 将取代原来的 gbest 来指导速度更新。

3.5 适应度函数

特征评价是对所选特征子集的质量进行评估，以指导进化过程的关键组成部分。分类算法主要有两个目标，一是提高分类精度，二是减小特征子集的大小。在本研究中，结合这两个目标的适应度函数被用来评价所选择的特征。式 (10) 为适应度函数，该函数使用一个权重将分类错误率与所选特征数量结合起来。

$$fitness = \alpha * ErrorRate + (1-\alpha) * \frac{\#Selected}{\#All} \tag{10}$$

其中 ErrorRate 表示学习算法的分类错误率，#Selected 表示选中特征的数量，#All 表示训练数据集中可用特征的总数。α 是控制分类错误率和所选特征数量贡献的权重, 范围为 [0,1]。由于分类性能优于特征子集大小，本文设 α = 0.9。

由于收集到的大部分数据是不平衡的，本研究采用平衡精度来评价适应度函数的第一个分量。对训练样本进行 5 倍交叉验证，以避免训练过程中的 FS 偏差。分类错误率的计算方法如下:

$$ErrorRate = 1 - \frac{1}{c} * \sum_{i=1}^{c} TPR_i \tag{11}$$

其中，c 表示数据中类的数量，TPR_i 表示真阳性率，即第 i 类中正确识别实例的比例。由于平衡精度对分类问题中的每个类都没有偏见，因此将每个类的权重设为 1/c。

4 复现细节

4.1 代码

代码部分为自己复原，原作并未公开代码。复现采用 matlab 实现，具体的代码已放在 github 上。接下来会分三个部分来展示自己的复现过程和思路，分别是任务生成策略，多任务优化方法和适应度函数。

4.2 任务生成策略

在任务生成策略上，采用的是膝点分类的原则，首先是用 ReliefF 函数评估特征的相关信息，这是因为 ReliefF 比其他相关度量方法能更好地确定变量之间的非线性关系，因其能为每个特性分配权重已被广泛采用来确定特征对类标签的重要性。如图 5这里采用的是 matlab 自带的 relieff 函数进行分类。

```
%使用ReliefF函数作为分类器对特征值进行分类
[ranks,weights] = relieff(data,label,1,'method','classification');
```

图 5: relieff 函数分类实现

在膝点选择策略中，特征根据由 ReliefF 计算出来的权重降序排序，然后得到关于权重的曲线。接下来，在具有最大和最小特征权重的两点之间生成一条线 (L)。计算曲线上每个点与 L 之间的距离，并检测到距离 L 最大的点为膝点。图 6为具体实现的代码。

```
%确定膝点
maxpoint = [1,weights(ranks(1))];
minpoint = [size(ranks,2),weights(ranks(size(ranks,2)))];
maxDistance = 0;
kneepoint = 0;
for i=1:size(ranks,2)
    point = [i,weights(ranks(i))];
    d = abs(det([minpoint-maxpoint;point-maxpoint]))/norm(minpoint-maxpoint);
    if d > maxDistance
        maxDistance = d;
        kneepoint = i;
    end
end
```

图 6: 确定膝点实现代码

最后将原本的特征集根据膝点分成有希望的特征集（即 ProFeatures）和剩余的特征集 (即 Rem-Features)。基于已经分好的两个特征集，我们引入一个超参数 p，利用概率从有前途的特征集中和剩余的特征集中选择特征，形成相关的 FS 任务。在该策略中，利用有前途特征集和剩余特征集的平均特征权重来确定选择概率 p。概率 p 可以保证生成的任务中有希望的特征以较高的概率加入子集，剩余的特征以较低的概率加入子集。如果特征在有希望的特征集中，当条件 rand <= p 成立时，相应的特征被选择到 task 中，否则不选择。如果特征在剩余的特征集中，当条件 rand < (1−p) 被激活时，相应的特征被选中到 Task 中，否则不选中，其中 rand 表示 [0,1] 之间的随机值。通过重复上述过程 T 次，可以获得 T 个不同的 FS 任务。图 7为实现根据膝点分类生成子任务的核心代码。

```
for i = 1:T
    temp = [];
    for j = 1:size(prodata,2)
        n = rand;
        if n <= p
        temp(end+1) = prodata(j);
        end
    end
    for j = 1:size(remdata,2)
        n = rand;
        if n < 1-p
        temp(end+1) = remdata(j);
        end
    end
    task{1,i} = temp;
end
```

图 7: 子任务生成实现代码

该策略通过将巨大的搜索空间转化为多个小而有前景的空间 (即多个相关任务)，避免浪费计算资源去探索不需要的区域 (例如特征不相关的区域)。

4.3 多任务优化方法

在该方法中，每个子种群对应一个任务。与其他多种群多任务优化方法不同，设计了一种新的知识传递机制来实现不同任务 (子种群) 之间的通信。在本研究中，使用粒子位置的交叉算子作为不同任务 (即子种群) 之间的知识转移方式。跨界共享信息不仅可以实现不同任务之间的知识转移，还可以增强搜索过程中的多样性。在进化过程中，我们定义一个随机配对概率 (rmp) 来控制何时从其他任务转移知识。较大的 rmp 鼓励从其他任务中转移知识，反之亦然。在每一代中，如果知识转移策略被激活 (rand<=rmp)，来自其他任务的知识将被用于协助更新粒子的位置。否则 (rand>rmp)，粒子的位置将由来自当前任务的知识更新。在粒子群算法中，全局最优位置所携带的知识在速度更新中起着重要的作用，在搜索过程中，它被用来影响每个粒子的搜索方向。因此，我们利用全局最优位置上的交叉来实现不同任务之间的知识转移。图 8为具体实现的核心代码。

```
%开始迭代
for iter = 1:max_iter

    disp(['******************run for: ' num2str(iter) '***************']);
    w = 0.9-0.5*iter/max_iter;

    % 更新速度
    for task = 1:T
        % 这里迁移的时候对于不同搜索空间只迁移了重叠部分
        if rand()< rmp
            if taskSet{task}.notChangeIters >= maxIterUpdateVelocity
                % 这里因为只考虑重叠部分的平均，所以实现的相对繁琐一点
                    gBest = zeros(1,featureNum);
                    for i = 1:featureNum
                        if taskSet{task}.searchSpace(1,i)
                           posList = [];
                           for j = 1:T
                               if taskSet{j}.searchSpace(1,i)
                                    posList(end+1) = taskSet{j}.gBest.pos(1,i);
                               end
                           end
                           gBest(1,i) = mean(posList);
                        end
                    end
            else
                    %这里文中锦标赛排序的参数没有交代，设置成2
                    TSNum = 2;
                    gBestList = zeros(1,T);
                    for i = 1:T
                       gBestList(i) = taskSet{i}.gBest.fit;
                    end
                    % 防止锦标赛选择选到自己
                    index = 0;
                    while index ~= task
                        index = TournamentSelection(TSNum,1,gBestList);
                    end
                    overlapSpace = taskSet{task}.searchSpace & taskSet{index}.searchSpace;
                    gBest = taskSet{task}.gBest.pos;
                    cr = rand(1,sum(overlapSpace));
                    gBest(overlapSpace) = cr.*taskSet{task}.gBest.pos(overlapSpace) + (1-cr)
            end
        else
            gBest = taskSet{task}.gBest.pos;
        end
        % 更新速度
        taskSet{task}.velocity = w*taskSet{task}.velocity + c1*rand(popNum,featureNum).*(tas
                c2*rand(popNum,featureNum).*(gBest-taskSet{task}.position);
        taskSet{task}.velocity(taskSet{task}.velocity<Vlimit(1)) = Vlimit(1);
        taskSet{task}.velocity(taskSet{task}.velocity>Vlimit(2)) = Vlimit(2);
    end
```

图 8: 多任务优化实现代码

4.4 适应度函数

特征评价是对所选特征子集的质量进行评估，以指导进化过程的关键组成部分。分类算法主要有两个目标，一是提高分类精度，二是减小特征子集的大小。在本研究中，结合这两个目标的适应度函数被用来评价所选择的特征。这里使用 5 折交叉验证对实验结果进行适应度分析。图 9为具体实现的核心代码。

```
function [preLabel] = KNN(trainX,trainY,testX)
    neighborNum = 5;

    Dis = pdist2(testX,trainX,'cityblock');

    testNum = size(testX,1);
    neighborPosition = zeros(testNum,neighborNum);

    for i= 1:neighborNum
        [~,minPos]  = min(Dis,[],2);
        neighborPosition(:,i) = minPos;
        for j = 1:testNum
           Dis(j,minPos(j)) = inf;
        end
    end
    y = trainY(neighborPosition);


    preLabel = mode(y,2);
end
```

图 9: 适应度函数实现代码

5 实验结果分析

这里使用的是原作使用的数据集之一—PCMAC 进行实验，PCMAC 特征数据集是一个大小为 1943*3289 的一个特征数据集，特征维度为 3289 个特征，数据量为 1943 个，这是一个二分类数据集。首先在任务生成策略阶段，原本的数据集有高达 3289 个维度的特征，在分类之后每个子相关任务的维度减少到 327-389 个维度这个范围内，减少了将近 89.11% 的冗余维度。图 10和图 11分别为原始数据集的规模大小和生成任务子集后的子任务规模大小。

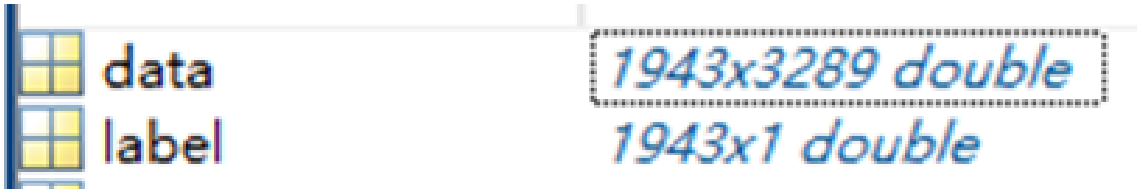

图 10: 初始数据集规模

task

1x8 cell

	1	2	3	4	5	6	7	8
1	1x344 dou...	1x382 dou...	1x375 dou...	1x374 dou...	1x343 dou...	1x362 dou...	1x327 dou...	1x389 dou...

图 11: 子任务规模大小

如图 12是最终的优化结果。其中 Featurenum 为最后选择的特征数量，fit 为子集的适应度，是用 5 折 KNN 算法进行交叉验证算出来的结果，从结果上来看第一个子集的选择结果最好，冗余特征排

除率也达到了 94.9%。

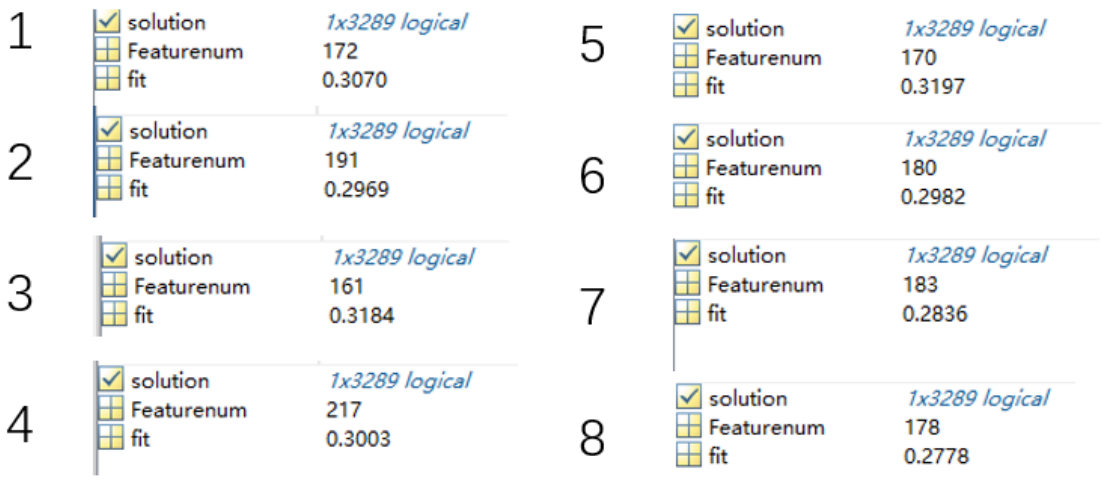

图 12: 最终优化结果

6 总结与展望

本文的目标是为高维分类问题开发一种有效的 FS 方法。通过提出一种有效的 FS 多任务粒子群算法，通过相关任务间的知识转移有效地选择出好的特征子集，成功地实现了该目标。

在不久的将来可以进一步研究一些有趣的方向。首先，如何通过考虑数据本身的特点，有效、高效地决定任务的数量。其次，如何在考虑特征相关性的前提下设计有效的知识转移机制，进一步提高所提方法的性能。第三，如何结合领域专业知识 (如基因表达领域)，进一步减少所选特征的数量，提高在实践中验证特征的效率。此外，将 FS 看作一个多任务多目标优化问题是一个很有前途的研究方向，它可以为决策者提供更多的折衷解决方案。