一种解决大规模稀疏多目标优化问题的进化算法

摘要

在过去的两十年里，进化计算领域对多目标优化问题（MOPs）进行了广泛研究，涵盖了各种不同类型的问题。然而，大多数现有的进化算法在处理 Pareto 最优解稀疏的 MOPs 时遇到了挑战，尤其是当决策变量的数量很大时。这种大规模稀疏 MOPs 的问题在各种应用中都普遍存在，例如，从众多候选特征中选择一个小子集以进行特征选择，或者优化稀疏连接以减少神经网络的过拟合问题。本文复现的是一种用于解决大规模稀疏 MOPs 的进化算法，这个算法采用了新的种群初始化策略和遗传算子，考虑到 Pareto 最优解的稀疏性，以确保生成的解也具有稀疏性。本文在此算法基础上进行改进，结果表明所改进的算法有更好的性能。

关键词：进化算法；大规模多目标优化（MOP）；稀疏帕累托最优解

1 引言

多目标优化问题（MOPs）是指那些需要同时优化多个目标的问题。这类问题在各个领域中都广泛存在，例如投资组合优化问题，旨在最大化回报和最小化风险；以及社区检测问题，旨在最大化内部链接密度和最小化外部链接密度。由于不同目标通常存在冲突，因此不存在一种解可以同时优化所有目标，取而代之的是存在一系列解，它们在不同目标之间进行权衡，被称为帕累托最优解。所有这些帕累托最优解构成了帕累托集，而在目标空间中的这些解的集合则称为帕累托前沿。自第一个多目标进化算法（MOEA）的实施以来，各种不同类型的 MOP 已被考虑用进化算法去解决。

存在现实世界中的许多优化问题包含大量的决策变量，这些问题被称为大规模优化问题（LOPs）。由于大规模优化问题的搜索空间呈指数增长，通常无法轻松通过通用进化算法解决，因此一些新颖的技术已被采用来应对特定类型的大规模优化问题，例如变量交互分析、关联学习、以及基于随机嵌入的贝叶斯优化。在许多重要领域的大规模优化问题，包括机器学习 [15]、数据挖掘 [26] 和网络科学 [19]。当帕累托最优解是稀疏的，即大多数最优解的决策变量为零时，称之为稀疏多目标问题（SOPs）。例如，分类中的特征选择问题旨在选择少量相关特征以实现最佳分类性能，通过同时最小化所选特征的数量和分类错误来实现。一般来说，帕累托最优解中的特征占所有候选特征的一小部分 [30]。因此，如果采用二进制编码（即，每个二进制决策变量表示是否选择某个特征），大多数帕累托最优解的决策变量将为零，换句话说，帕累托最优解是稀疏的。这种具有稀疏帕累托最优解的子集选择问题在许多其他应用中也很常见，例如稀疏回归 [23]、模式挖掘 [26] 和同时关键节点检测 [19]。然而，对于大规模稀疏多目标优化问题的研究，尚未有一个切实可行的算法能有效解决。

为了应对这一挑战，复现的论文提出了一种新的进化算法，旨在解决大规模稀疏多目标优化问题。该算法通过考虑帕累托最优解的稀疏性，提出了新的种群初始化策略和遗传算子，以确保生成的解也具有稀疏性。此外，本文在复现此算法的过程中对算法有所改进，通过实验结果表明，改进后的算法在解决大规模稀疏多目标优化问题方面表现优于原文的算法。这项研究为处理大规模稀疏多目标优化问题提供了新的方法和工具，有望在各种应用中提高决策的质量和效率。

2 相关工作

尽管大规模稀疏多目标优化问题在许多常见应用中广泛存在，但这类问题以前并没有受到专门的研究。实际上，很少有多目标进化算法（MOEAs）专门用于解决稀疏 MOPs，大多数工作只是采用通用的 MOEAs 来解决它们 [12,31]。根据文献报道，许多现有的 MOEAs 在处理稀疏 MOPs 时遇到困难，特别是当决策变量数量较大时 [29,33]。本章节将对大规模稀疏多目标优化问题的相关工作进行简要介绍。

首先介绍应用中的四个流行的稀疏多目标优化问题。分别是特征选择 [30]、模式挖掘 [26]、关键节点检测 [19] 和神经网络训练 [15]。

特征选择问题：对于分类任务，通常会引入大量特征到数据集中，其中许多特征是不相关或多余的。特征选择的目标是仅选择与分类相关的特征，这不仅可以减少特征数量，还可以提高分类性能 [5]。因此，特征选择本质上是一个双目标 MOP，并且已经有一些 MOEAs 用于解决它 [12,30]。

模式挖掘问题：这个问题旨在从交易数据集中找到最频繁和完整的模式（即一组项目），其中交易数据集包含一组交易，每个交易包含一组项目。例如，当顾客在网站上购买一件商品时，任务是通过从所有历史购物清单中挖掘出最佳模式，向顾客推荐匹配的商品。在 [33] 中，模式挖掘问题被视为一个 MOP，旨在最大化频率和完整性，其中模式的频率表示其出现在的交易比例，而模式的完整性表示其在出现的交易中的占用率。

关键节点检测问题：： 给定一个图 $G = (V\ E)$，关键节点检测问题 [19] 是选择一个点集 $x \subseteq V$，其删除最小化了剩余图 $G[V \setminus x]$ 中的成对连通性。为了在对图的破坏较大时选择较少的点，应最小化所选点的数量以及剩余图中的成对连通性。因此，关键节点检测问题也是一个子集选择的多目标优化问题（MOP），并已被一些多目标进化算法（MOEAs）解决 [10,20]。

神经网络训练问题：进化算法已经深入研究了训练人工神经网络的方法，如优化网络的拓扑结构 [24]、权重 [16] 和超参数 [31]。在多目标方法方面，已经考虑了训练神经网络中的各种目标 [1,4,15]。为了控制神经网络的复杂性，在本文中采用了 [15] 建议的定义，即优化前馈神经网络的权重以最小化复杂性和错误。

尽管上述四个 MOPs 来自不同领域并包含不同的目标，但它们可以被归为相同类别的 MOPs，因为它们的帕累托最优解都是稀疏的。对于特征选择、关键节点检测和神经网络训练等问题，解的稀疏性被定义为直接优化的目标之一，因此帕累托最优解肯定是稀疏的。而对于模式挖掘问题，虽然它没有明确控制解的稀疏性，但由于目标的定义，帕累托最优解也是稀疏的。一般来说，每个交易中的项目数量要远少于数据集中所有项目的总数，因此每个帕累托最优解中的项目应该只占所有项目的一小部分，否则该解将不会出现在任何交易中。

对于上述的稀疏多目标优化问题（Sparse MOPs），大部分研究使用现有的多目标进化算

法（MOEAs）来解决，如 NSGA-II [8]、SMS-EMOA [3] 和 MOEA/D [32]，而一些研究也通过采用定制的搜索策略来增强性能。例如，顾等人 [11] 修改了竞争性群体优化器，并将其嵌入到特征选择的包装方法中，张等人 [33] 建议在 NSGA-II 中采用了一种新颖的种群初始化策略来解决模式挖掘问题，Aringhieri 等人 [2] 在遗传算子中采用了适当的贪婪规则，以改善关键节点检测的后代质量，而 Sun 等人 [25] 假设了一组正交向量，以大幅减少在训练无监督深度神经网络时的决策变量数量。然而，值得注意的是，这些定制的搜索策略不能直接用于解决其他稀疏 MOPs，因为它们与目标函数和数据集的结构相关，只能解决特定类型的稀疏 MOPs。

另一方面，专门为大规模多目标优化问题（Large-scale MOPs）定制的许多 MOEAs 不能应用于具有二进制变量的特征选择问题、模式挖掘问题和关键节点检测问题，因为这些 MOEAs 大多基于决策变量的划分。例如，MOEA/DVA [21] 和 LMEA [34] 分别通过控制属性分析和变量聚类来划分决策变量，这两种方法都扰动了连续决策空间中的每个决策变量。WOF [36] 将决策变量划分为若干组，并优化了一个权重向量，而不是在同一组中的所有决策变量。因此，这些 MOEAs 只能解决具有实数变量的 MOPs。此外，划分决策变量需要大量的函数评估，这对于解决具有大型数据集的稀疏 MOPs 来说效率低下。

3 本文方法

为了解决上述问题，文章提出了一种用于解决大规模稀疏多目标优化问题（MOPs）的多目标进化算法（MOEA）。所提出的算法无需了解目标函数或数据集的具体信息，能够同时有效地解决具有实数变量和二进制变量的 MOPs。

3.1 本文方法概述

如算法 1所示，所提出的针对大规模稀疏 MOPs 的进化算法，称为 *SparseEA*，具有与 NSGA-II [8] 相似的框架。首先，初始化一个大小为 N 的 P 种群，并计算 P 中各解的非主导前沿数 [28] 和拥挤距离 [8]。在主循环中，根据 P 中每个解的非优势数和拥挤距离，通过二元锦标赛选择 $2N$ 个亲本，然后生成 N 个后代（即两个亲本生成一个后代），然后结合 P，删除组合种群中的重复解，合并种群中非优势数和拥挤距离较好的 N 个解存活到下一代。

简而言之，*SparseEA* 中的交配选择和环境选择与 NSGA-II 中的类似，但 *SparseEA* 使用新策略生成初始种群和后代，这可以确保生成的解具有稀疏性。*SparseEA* 中的种群初始化策略和遗传算子将在接下来的两个部分分别详细说明。

Algorithm 1 Framework of the Proposed SparseEA

Input: N (population size)
Output: P (final population)
$[P, Score] \leftarrow$ Initialization(N) ▷ Algorithm 2
$[F_1, F_2, \ldots] \leftarrow$ Do non-dominated sorting on P
$CrowdDis \leftarrow$ CrowdingDistance(F)
while termination criterion not fulfilled **do**
    $P' \leftarrow$ Select $2N$ parents via binary tournament selection
        according to the nondominated front number and $CrowdDis$
        of solutions in P;
    $P \leftarrow P \cup$ Variation$(P', Score)$ ▷ Algorithm 3
    Delete duplicated solutions from P;
    $[F_1, F_2, \ldots] \leftarrow$ Do non-dominated sorting on P;
        // F_i denotes the solution set in the i-th
        // non-dominated front
    $CrowdDis \leftarrow$ CrowdingDistance(F);
    $k \leftarrow \arg\min_i |F_1 \cup \ldots \cup F_i| \geq N$;
    Delete $|F_1 \cup \ldots \cup F_k| - N$ solutions from F_k with the
        smallest $CrowdDis$;
    $P \leftarrow F_1 \cup \ldots \cup F_k$;
end while
return P;

3.2 SparseEA 的种群初始化策略

所提出的 SparseEA 旨在解决具有实数变量和二进制变量的稀疏多目标优化问题（sparse MOPs）。因此需要采用混合表示解的方法，以整合两种不同的编码方式，这在解决许多问题中已被广泛使用，例如神经网络训练 [17] 和投资组合优化 [7]。具体而言，SparseEA 中的解 x 由两个部分组成，即表示决策变量的实向量 dec 和表示掩码的二进制向量 mask。用于计算 x 的最终决策变量的公式如下：

$$(x_1, x_2, \ldots) = (\text{dec}_1 \times \text{mask}_1, \text{dec}_2 \times \text{mask}_2, \ldots). \tag{1}$$

例如，对于一个实向量 dec $= (0.5, 0.8, 0.3, 0.6, 0.1)$ 和一个二进制向量 mask $= (1, 0, 0, 1, 0)$，决策变量为 $(0.5, 0, 0, 0.6, 0)$。在进化过程中，每个解的实向量 dec 可以记录到目前为止找到的最佳决策变量，而二进制向量 mask 可以记录应该被设置为零的决策变量，从而控制解的稀疏性。请注意，如果决策变量是二进制数，那么实向量 dec 始终设置为全为 1 的向量，因此最终的决策变量可以根据二进制向量 mask 确定为 0 或 1。在 SparseEA 中，每个解的 dec 和 mask 由不同的策略生成和演化。

Algorithm 2 Initialization(N)

Input: N (population size)
Output: P (initial population), $Score$ (scores of decision variables)
$D \leftarrow$ Number of decision variables;
if the decision variables are real numbers **then**
    $Dec \leftarrow D \times D$ random matrix;
else if the decision variables are binary numbers **then**
    $Dec \leftarrow D \times D$ matrix of ones;
end if
$Mask \leftarrow D \times D$ identity matrix;
$Q \leftarrow$ A population whose i-th solution is generated by the i-th rows of Dec and $Mask$ according to (1);
$[F_1, F_2, \ldots] \leftarrow$ Do non-dominated sorting on Q;
for $i = 1$ to D **do**
    $Score_i \leftarrow k, s.t. Q_i \in F_k$; // Q_i denotes the i-th solution in Q
end for
if the decision variables are real numbers **then**
    $Dec \leftarrow$ Uniformly randomly generate the decision variables of N solutions;
else if the decision variables are binary numbers **then**
    $Dec \leftarrow N \times D$ matrix of ones;
end if
$Mask \leftarrow N \times D$ matrix of zeros;
for $i = 1$ to N **do**
    for $j = 1$ to $rand() \times D$ **do**
        $[m, n] \leftarrow$ Randomly select two decision variables;
        if $Score_m < Score_n$ **then**
            Set the m-th element in the i-th binary vector in $Mask$ to 1;
        else
            Set the n-th element in the i-th binary vector in $Mask$ to 1;
        end if
    end for
end for
$P \leftarrow$ A population whose i-th solution is generated by the i-th rows of Dec and $Mask$ according to (1);
return P and $Score$;

SparseEA 的种群初始化策略如算法 2所示，包括两个步骤，即计算决策变量的得分和生成初始种群。首先，生成一个包含 D 个解的种群 Q，其中 D 表示决策变量的数量。具体而言，每个解的实向量 dec 中的元素被设置为随机值，而二进制向量 mask 中的元素被设置为 0，除了第 i 个解的掩码中的第 i 个元素设置为 1 外，其余都设置为 0。因此，种群 Q 中所有

解的二进制向量 mask 构成了一个 $D \times D$ 的单位矩阵。然后，种群 Q 按非支配排序进行排序，第 i 个解的非支配前沿数被视为第 i 个决策变量的得分。因此，决策变量的得分表示它应该被设置为零的概率，较小的得分可能表示决策变量应该被设置为零的概率较低，因为较小的非支配前沿数表示解的质量较好。决策变量的得分将在 SparseEA 中的种群初始化策略和遗传算子中都起作用，根据决策变量的得分通过二进制锦标赛选择来选择要翻转的掩码元素。值得注意的是，决策变量的得分分配策略可以更精确地考虑变量之间的相互作用 [22]，并在演化过程中更新得分。然而，提出的得分分配策略在解决稀疏多目标优化问题方面经验上被证明是有效的，并且我们根据以下两点考虑在 SparseEA 中使用这种得分分配策略。首先，提出的得分分配策略非常高效，只需要 D 次函数评估。其次，二进制向量 mask 是根据决策变量的得分通过二进制锦标赛选择生成的，二进制锦标赛选择中的随机性有助于得分分配策略的有效性。

另一方面，一些现有的算法也使用实值来表示变量为 0 或 1 的概率 [13,18]，但 SparseEA 中的得分完全不同。具体而言，现有算法的解中直接编码了每个变量为 0 或 1 的概率，这些概率在进化过程中进行优化。相比之下，SparseEA 中每个变量的得分是事先估计的，用于引导解的优化。

在计算决策变量的得分之后，初始种群 P 被生成。对于 P 中的每个解，实向量 dec 中的每个元素都被设置为随机值，而二进制向量 mask 中的每个元素都被设置为 0。然后，根据决策变量的得分（得分越小越好）从 mask 中选择 $\text{rand}() \times D$ 个元素，并将它们设置为 1，其中 rand() 表示在 $[0, 1]$ 范围内均匀分布的随机数。

3.3 SparseEA 的遗传算子

算法 3详细描述了 SparseEA 中的遗传算子，每次从 P 中随机选择两个父代 p 和 q 来生成一个后代 o。后代 o 的二进制向量掩码首先设置为与 p 相同，然后以相同概率执行以下两种操作之一：根据决策变量的得分（得分越大越好）从 $p.\text{mask} \cap q.\text{mask}$ 中的非零元素中进行二进制锦标赛选择，将 o 的掩码中的此元素设置为 0 [7]；或者根据决策变量的得分（得分越小越好）从 $p.\text{mask} \cap q.\text{mask}$ 中的非零元素中进行二进制锦标赛选择，将 o 的掩码中的此元素设置为 1 [7]。之后，o 的掩码通过以下两种操作之一以相同概率进行变异：根据决策变量的得分（得分越大越好）从 $o.\text{mask}$ 中的非零元素中进行二进制锦标赛选择，将此元素设置为 0；或者根据决策变量的得分（得分越小越好）从 $o.\text{mask}$ 中的非零元素中进行二进制锦标赛选择，将此元素设置为 1。o 的实向量 dec 是由许多现有的多目标进化算法采用的相同操作生成的，即模拟二进制交叉和多项式变异。请注意，如果决策变量是二进制数字，o 的 dec 总是设置为全 1 的向量。总之，提议的 SparseEA 采用了用于实数变量的现有遗传算子，而对于二进制变量则采用了新的遗传算子 [9,14]。事实上，一些专门设计的二进制变量遗传算子已经存在 [9,14]。但值得注意的是，SparseEA 中的遗传算子专门针对稀疏多目标优化问题而定制。具体而言，提出的遗传算子以相同的概率翻转二进制向量掩码中的零元素或非零元素，翻转的元素是基于决策变量的得分进行选择的。因此，提出的 SparseEA 生成的后代不会有相同数量的 0 和 1，并且可以确保后代的稀疏性。

Algorithm 3 Variation(P', Score)

Input: P' (a set of parents), $Score$ (scores of decision variables)
Output: O (a set of offsprings)
$O \leftarrow \varnothing$;
while P' is not empty **do**
    $[p, q] \leftarrow$ Randomly select two parents from P';
    $P' \leftarrow P' - \{p, q\}$;
    $o.mask \leftarrow p.mask$; ▷ p.mask denotes the binary vector mask of solution p
    if rand() < 0.5 **then**
        $[m, n] \leftarrow$ Randomly select two variables from the nonzero elements in $p.mask \cap q.mask$;
        if $Score_m > Score_n$ **then**
            Set the m-th element in $o.mask$ to 0;
        else
            Set the n-th element in $o.mask$ to 0;
        end if
    else
        $[m, n] \leftarrow$ Randomly select two variables from the nonzero elements in $p.mask \cap q.mask$;
        if $Score_m < Score_n$ **then**
            Set the m-th element in $o.mask$ to 1;
        else
            Set the n-th element in $o.mask$ to 1;
        end if
    end if
    if rand() < 0.5 **then**
        $[m, n] \leftarrow$ Randomly select two variables from the nonzero elements in $o.mask$;
        if $Score_m > Score_n$ **then**
            Set the m-th element in $o.mask$ to 0;
        else
            Set the n-th element in $o.mask$ to 0;
        end if
    else
        $[m, n] \leftarrow$ Randomly select two variables from the nonzero elements in $o.mask$;
        if $Score_m < Score_n$ **then**
            Set the m-th element in $o.mask$ to 1;
        else
            Set the n-th element in $o.mask$ to 1;
        end if
    end if
    if the decision variables are real numbers **then**
        $o.dec \leftarrow$ Perform crossover and mutation based on $p.dec$ and $q.dec$;
    else
        $o.dec \leftarrow$ Vector of ones;
    end if
    $O \leftarrow O \cup \{o\}$;
end while
return O;

4 复现细节

4.1 与已有开源代码对比

图 1为 SparseEA 中使用的 OperatorGAhalf 函数用于生成子代 *Dec*,图 2为改进的 SparseEA 使用的 Croup_Operate 函数用于生成子代 *Dec*。

```
%% Crossover and mutation for dec
if any(Problem.encoding~=4)
    OffDec = OperatorGAhalf(Problem,ParentDec);
    OffDec(:,Problem.encoding==4) = 1;
else
    OffDec = ones(N/2,D);
end
```

图 1. SparsEA 中使用 OperatorGAhalf 函数生成子代 *Dec*

```
%% dec的杂交和变异
if any(Problem.encoding ~= 4)
    [off_dec, groupIndex, chosengroups] = Group_Operate(Problem, p1_dec, p2_dec, 4); % 4为组数
    off_dec(:, Problem.encoding == 4) = 1;
else
    off_dec = ones(size(p1_dec));
end
```

图 2. SparsEA 中使用 Croup_Operate 函数生成子代 *Dec*

如图 3是截取自 OperatorGAhalf 的部分代码，图 4是截取自 Croup_Operate 函数的部分代码。对比 SparseEA 的 OperatorGAhalf 和改进的 SparseEA 的 Group_Operate 函数的执行步骤，主要差异如下：

SparseEA 的 OperatorGAhalf:

- 使用了标准的遗传算法操作，包括交叉和变异。
- 根据变量类型（实数、二进制、排列等）应用不同的遗传算法操作。
- 针对每种类型变量使用了特定的交叉和变异策略。

改进的 SparseEA 的 Group_Operate:

- 也包含交叉和变异操作，但引入了变量分组的概念。
- 通过 *CreateGroups* 函数将变量分组，然后针对这些组进行交叉和变异。
- 使用了分组索引和选定组（*out_index_list* 和 *chosen_groups*）来指导变异操作。

简而言之，SparseEA 使用了更传统的交叉和变异策略，而改进的 SparseEA 引入了变量分组的概念，以增加算法处理复杂性和提高效率。这种分组概念在 SparseEA 中是一个显著的改进，它允许算法更有针对性地处理变量，可能在处理大规模优化问题时更为有效。

```
%% Simulated binary crossover
[N,D] = size(Parent1);
beta  = zeros(N,D);
mu    = rand(N,D);
beta(mu<=0.5) = (2*mu(mu<=0.5)).^(1/(disC+1));
beta(mu>0.5)  = (2-2*mu(mu>0.5)).^(-1/(disC+1));
beta = beta.*(-1).^randi([0,1],N,D);
beta(rand(N,D)<0.5) = 1;
beta(repmat(rand(N,1)>proC,1,D)) = 1;
Offspring = (Parent1+Parent2)/2+beta.*(Parent1-Parent2)/2;

%% Polynomial mutation
Lower = repmat(lower,N,1);
Upper = repmat(upper,N,1);
Site  = rand(N,D) < proM/D;
mu    = rand(N,D);
temp  = Site & mu<=0.5;
Offspring       = min(max(Offspring,Lower),Upper);
Offspring(temp) = Offspring(temp)+(Upper(temp)-Lower(temp)).*((2.*mu(temp)+(1-2.*mu(temp)).*...
                  (1-(Offspring(temp)-Lower(temp))./(Upper(temp)-Lower(temp))).^(disM+1)).^(1/(disM
temp = Site & mu>0.5;
Offspring(temp) = Offspring(temp)+(Upper(temp)-Lower(temp)).*(1-(2.*(1-mu(temp))+2.*(mu(temp)-0.5).
                  (1-(Upper(temp)-Offspring(temp))./(Upper(temp)-Lower(temp))).^(disM+1)).^(1/(disM
```

图 3. OperatorGAhalf 函数代码片段 *Dec*

```
function [out_index_array, number_of_groups_array] = CreateGroups(number_of_groups, x_prime, number_of_variables)
    % 使用按序分组

    out_index_array = [];
    number_of_groups_array = [];
    no_of_solutions = size(x_prime, 1);
    for sol = 1:no_of_solutions
        vars_per_group = floor(number_of_variables / number_of_groups);
        vars = x_prime(sol, :);
        [~, I] = sort(vars);
        out_index_list = ones(1, number_of_variables);
        for i = 1:number_of_groups-1
            out_index_list(I(((i-1) * vars_per_group) + 1:i * vars_per_group)) = i;
        end
        out_index_list(I(((number_of_groups - 1) * vars_per_group) + 1:end)) = number_of_groups;
        out_index_array = [out_index_array; out_index_list];
        number_of_groups_array = [number_of_groups_array; number_of_groups];
    end
end
```

图 4. Croup_Operate 函数代码片段 *Dec*

4.2 实验环境搭建

实验平台：MATLAB R2021b 和 PlatEMO [27]

4.3 界面分析与使用说明

参考 MARLAB 用户使用手册，PlatEMO 可以从 GitHub 中下载：
https://github.com/BIMK/PlatEMO

4.4 创新点

原 SparseEA：采用简单的交叉和变异策略；针对决策变量的交叉和变异仅当问题编码不是二进制时进行；在掩码上进行交叉和变异，以便选择性地启用或禁用某些基因。

改进的 SparseEA：引入了按序分组策略，通过此策略，算法能够更有效地处理大规模和复杂的优化问题。分组允许算法针对性地对特定变量组进行交叉和变异操作，从而提高了搜索效率和解的质量。

5 实验结果分析

如图 5 所示的在 SMOP1_100，即决策变量维数为 100 的 SMOP1 问题，获得的Pareto前沿中进一步观察到的，拥有 100 个决策变量的 SMOP1 对于现有的多目标演化算法来说都是相当具有挑战性的，所有四个比较的多目标演化算法都无法收敛到 Pareto 前沿，而 SparseEA 和SparseEA_Mine可以获得良好收敛的种群。且，SparseEA_Mine 算法获得的解略优于 SparseEA 算法获得的解。

表格 1呈现了由 NSGA-II、CMOPSO、MOEA/D-DRA、WOF-NSGA-II 、SparseEA 和 SparseEA_Mine 在 SMOP1-SMOP8 上获得的中位数 IGD 值和四分位距（IQR）。可以看到，SparseEA_Mine 在所有测试实例上表现出最佳性能。但相比 SparseEA 性能提升并不是很明显，在大多数测试问题上性能高出的不会超过一个数量级。

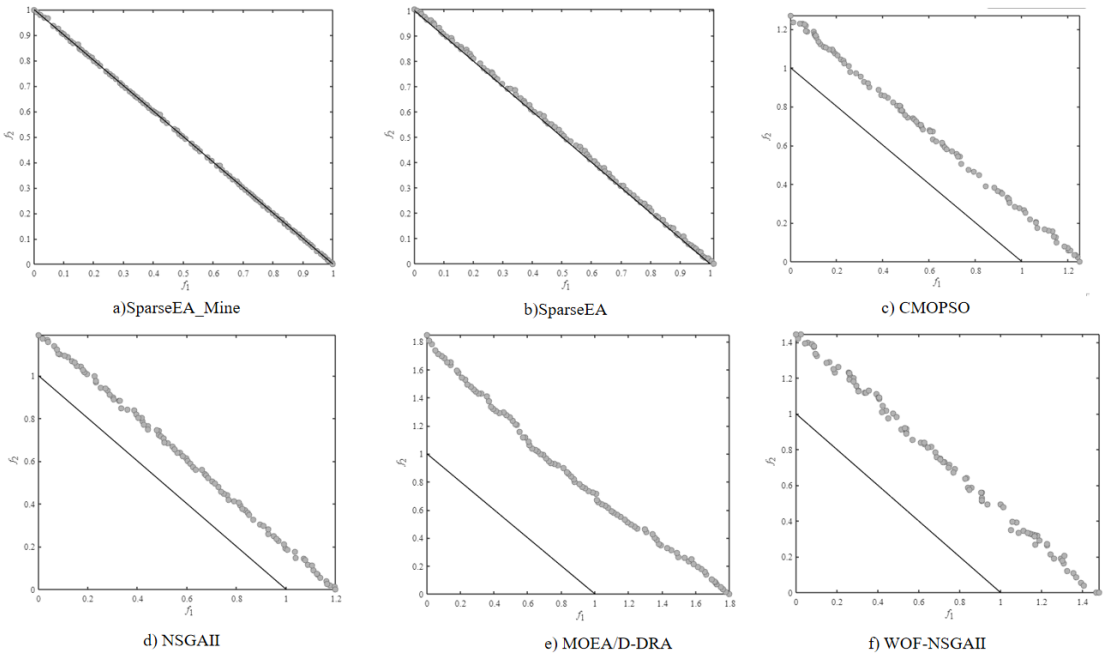

图 5. 各个算法在 SMOP1_100 上获得的 Pareto 前沿面

表 1. 各个算法在SMOP1-SMOP8上获得的中位数 IGD 值和 IQRS

Problem	D	NSGA-II	CMOPSO	MOEA/D-DRA	WOF-NSGA-II	SparseEA	SparseEA_Mine
SMOP1	100	3.3810e-1 (7.70e-2)	5.1484e-1 (2.63e-2)	1.4091e-1 (1.47e-2)	1.6121e-1 (2.25e-2)	9.9508e-3 (2.43e-3)	**6.1001e-3 (1.45e-3)**
SMOP2	100	1.2075e+0 (1.22e-1)	1.7319e+0 (5.21e-2)	5.3101e-1 (5.71e-2)	1.0815e+0 (6.35e-2)	3.0395e-2 (7.97e-3)	**1.1996e-2 (5.99e-3)**
SMOP3	100	8.3813e-1 (1.03e-1)	1.6102e+0 (1.87e-1)	8.8289e-1 (2.86e-2)	1.2283e+0 (1.01e-1)	1.7868e-2 (4.08e-3)	**6.5741e-3 (1.50e-3)**
SMOP4	100	5.7263e-1 (6.28e-2)	8.7936e-1 (4.13e-2)	1.8468e-1 (2.64e-2)	6.0840e-1 (6.72e-2)	**4.7928e-3 (2.12e-4)**	4.8044e-3 (2.56e-4)
SMOP5	100	4.0199e-1 (1.69e-2)	4.2419e-1 (4.54e-3)	3.7766e-1 (2.94e-3)	3.9738e-1 (3.99e-3)	6.1391e-3 (3.87e-4)	**5.6236e-3 (3.26e-4)**
SMOP6	100	9.8021e-2 (2.19e-2)	1.7277e-1 (8.65e-3)	5.0388e-2 (3.97e-3)	1.0477e-1 (6.63e-3)	7.6269e-3 (8.25e-4)	**6.6850e-3 (4.16e-4)**
SMOP7	100	5.9067e-1 (9.39e-2)	5.2877e-1 (2.22e-2)	3.7313e-1 (5.01e-2)	3.0551e-1 (2.99e-2)	4.3252e-2 (1.10e-2)	**1.4996e-2 (6.81e-3)**
SMOP8	100	2.5139e+0 (2.17e-1)	2.7171e+0 (9.76e-2)	1.6203e+0 (1.10e-1)	2.5003e+0 (2.88e-1)	1.6910e-1 (2.30e-2)	**1.3930e-1 (2.47e-2)**
SMOP1	500	5.1990e-1 (7.94e-2)	6.5857e-1 (1.62e-2)	4.7326e-1 (1.60e-2)	5.6528e-1 (2.37e-2)	6.2288e-2 (2.57e-3)	**5.7325e-2 (3.28e-3)**
SMOP2	500	1.3730e+0 (1.11e-1)	1.9748e+0 (1.95e-2)	1.3343e+0 (2.42e-2)	1.7585e+0 (4.06e-2)	1.3253e-1 (4.49e-3)	**1.2103e-1 (5.81e-3)**
SMOP3	500	1.0423e+0 (1.90e-1)	1.6893e+0 (2.55e-2)	1.5572e+0 (3.17e-2)	1.7562e+0 (4.19e-2)	6.7283e-2 (2.12e-3)	**6.1338e-2 (2.06e-3)**
SMOP4	500	6.7799e-1 (7.61e-2)	1.0181e+0 (1.17e-2)	6.9906e-1 (1.27e-2)	9.5649e-1 (2.41e-2)	4.8063e-3 (2.33e-4)	**4.7633e-3 (2.09e-4)**
SMOP5	500	4.4132e-1 (3.05e-2)	4.4975e-1 (2.83e-3)	4.7658e-1 (4.75e-3)	4.5892e-1 (4.02e-3)	2.4094e-2 (4.82e-3)	**1.2442e-2 (6.31e-3)**
SMOP6	500	1.5127e-1 (2.58e-2)	2.1064e-1 (4.58e-3)	1.6575e-1 (3.74e-3)	2.0060e-1 (5.53e-3)	2.5590e-2 (5.66e-3)	**1.7322e-2 (6.99e-3)**
SMOP7	500	7.3721e-1 (1.20e-1)	6.8747e-1 (2.93e-2)	9.2551e-1 (3.35e-2)	7.8174e-1 (3.09e-2)	1.7104e-1 (5.86e-3)	**1.4525e-1 (9.98e-3)**
SMOP8	500	2.8242e+0 (1.65e-1)	3.1276e+0 (8.00e-2)	2.9715e+0 (9.27e-2)	3.3160e+0 (9.54e-2)	3.6381e-1 (1.96e-2)	**3.3824e-1 (1.40e-2)**
SMOP1	1000	5.5290e-1 (8.30e-2)	6.7586e-1 (1.08e-2)	6.6465e-1 (1.54e-2)	7.3338e-1 (2.04e-2)	7.4849e-2 (7.25e-4)	**7.4273e-2 (8.99e-4)**
SMOP2	1000	1.4238e+0 (1.39e-1)	2.0201e+0 (2.04e-2)	1.6327e+0 (1.45e-2)	1.9409e+0 (2.23e-2)	1.5628e-1 (1.27e-3)	**1.5423e-1 (1.66e-3)**
SMOP3	1000	1.0265e+0 (1.62e-1)	1.7251e+0 (1.86e-2)	1.9100e+0 (1.53e-2)	1.9009e+0 (2.61e-2)	7.6312e-2 (2.64e-3)	**7.4560e-2 (6.12e-4)**
SMOP4	1000	6.9368e-1 (8.00e-2)	1.0343e+0 (1.23e-2)	8.3937e-1 (9.09e-3)	1.0157e+0 (6.51e-3)	**4.7205e-3 (2.25e-4)**	**4.7805e-3 (2.53e-4)**
SMOP5	1000	4.5493e-1 (4.52e-2)	4.5402e-1 (1.78e-3)	5.6423e-1 (7.01e-3)	4.8520e-1 (7.22e-3)	3.4170e-2 (9.96e-4)	**2.8050e-2 (8.31e-3)**
SMOP6	1000	1.4870e-1 (1.81e-2)	2.1537e-1 (3.28e-3)	2.3140e-1 (3.75e-3)	2.4366e-1 (6.15e-3)	3.8522e-2 (8.59e-4)	**3.3193e-2 (8.76e-3)**
SMOP7	1000	7.8725e-1 (1.24e-1)	7.0884e-1 (2.53e-2)	1.2642e+0 (3.45e-2)	9.9765e-1 (3.98e-2)	2.1331e-1 (2.76e-3)	**2.0999e-1 (2.43e-3)**
SMOP8	1000	2.8804e+0 (1.87e-1)	3.1723e+0 (6.02e-2)	3.3152e+0 (5.49e-2)	3.4523e+0 (5.25e-2)	4.7054e-1 (7.39e-3)	**4.6099e-1 (6.67e-3)**
+/-/=		0/8/0	0/8/0	0/8/0	0/8/0	0/22/2	

6 总结与展望

许多实际应用中的多目标优化问题具有稀疏的 Pareto 最优解，然而，这类问题以前并没有得到专门的研究，大多数工作只是采用现有的多目标演化算法来解决它们。为了填补这一空白，复现文章提出了一种用于解决大规模稀疏多目标优化问题的多目标演化算法，称为 *SparseEA*。所提出的 *SparseEA* 使用了一种新的种群初始化策略和遗传算子来生成稀疏解，经实验证明它对于稀疏多目标优化问题比现有的多目标演化算法更加有效。本文在此基础上对 SparseEA 算法进行改进，称为 SparseEA_Mine, 在子代生成的过程中加入了按序分组的策略，经实验证明，这个算法的表现优于其他算法。

由于稀疏多目标优化问题具有广泛的应用场景，因此仍然需要进一步研究这个主题。未来的工作可以从以下几个方面进一步推进，首先，将 *SparseEA* 与定制的搜索策略结合起来，用于解决特定的稀疏多目标优化问题在应用中具有趣味性。其次，对于具有许多大规模稀疏多目标优化问题，除了 *SparseEA* 中采用的基于非支配排序和拥挤度距离的环境选择策略，还可以采用其他环境选择策略在 *SparseEA*。最后，由于 *SparseEA* 在种群进化时不考虑决策变量之间的相互作用，因此可以通过采用其他策略如决策变量聚类 [35] 和贝叶斯优化 [6] 来增强其性能。最后，决策变量的得分可以在演化过程中动态更新，以使其更加准确。

参考文献

[1] Hussein A Abbass. Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. In *The 2003 Congress on Evolutionary Computation,*

2003. CEC'03., volume 3, pages 2074–2080. IEEE, 2003.

[2] Roberto Aringhieri, Andrea Grosso, Pierre Hosteins, and Rosario Scatamacchia. A general evolutionary framework for different classes of critical node problems. *Engineering Applications of Artificial Intelligence*, 55:128–145, 2016.

[3] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[4] Arjun Chandra and Xin Yao. Divace: Diverse and accurate ensemble learning algorithm. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 619–625. Springer, 2004.

[5] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156, 1997.

[6] Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. 2016.

[7] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[9] Andrej Dobnikar, Nigel C Steele, David W Pearson, Rudolf F Albrecht, A Simões, and E Costa. Transposition: A biological-inspired mechanism to use with genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Portorož, Slovenia, 1999*, pages 178–186. Springer, 1999.

[10] Luca Faramondi, Gabriele Oliva, Stefano Panzieri, Federica Pascucci, Martin Schlueter, Masaharu Munetomo, and Roberto Setola. Network structural vulnerability: a multiobjective attacker perspective. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(10):2036–2049, 2018.

[11] Shenkai Gu, Ran Cheng, and Yaochu Jin. Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing*, 22:811–822, 2018.

[12] Tarek M Hamdani, Jin-Myung Won, Adel M Alimi, and Fakhri Karray. Multi-objective feature selection with nsga ii. In *Adaptive and Natural Computing Algorithms: 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11-14, 2007, Proceedings, Part I 8*, pages 240–247. Springer, 2007.

[13] Kuk-Hyun Han and Jong-Hwan Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE transactions on evolutionary computation*, 6(6):580–593, 2002.

[14] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Diversity improvement by non-geometric binary crossover in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(6):985–998, 2010.

[15] Yaochu Jin, Tatsuya Okabe, and Bernhard Sendhoff. Evolutionary multi-objective optimization approach to constructing neural network ensembles for regression. In *Applications of Multi-Objective Evolutionary Algorithms*, pages 635–673. World Scientific, 2004.

[16] Yaochu Jin and Bernhard Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(3):397–415, 2008.

[17] Yaochu Jin and Bernhard Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(3):397–415, 2008.

[18] James Kennedy and Russell C Eberhart. A discrete binary version of the particle swarm algorithm. In *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, volume 5, pages 4104–4108. IEEE, 1997.

[19] Mohammed Lalou, Mohammed Amin Tahraoui, and Hamamache Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117, 2018.

[20] Juan Li, Panos M Pardalos, Bin Xin, and Jie Chen. The bi-objective critical node detection problem with minimum pairwise connectivity and cost: theory and algorithms. *Soft Computing*, 23(23):12729–12744, 2019.

[21] Xiaoliang Ma, Fang Liu, Yutao Qi, Xiaodong Wang, Lingling Li, Licheng Jiao, Minglei Yin, and Maoguo Gong. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Transactions on Evolutionary Computation*, 20(2):275–298, 2015.

[22] Martin Pelikan, David E Goldberg, and Erick Cantu-Paz. Linkage problem, distribution estimation, and bayesian networks. *Evolutionary computation*, 8(3):311–340, 2000.

[23] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. *Advances in neural information processing systems*, 28, 2015.

[24] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[25] Yanan Sun, Gary G Yen, and Zhang Yi. Evolving unsupervised deep neural networks for learning meaningful representations. *IEEE Transactions on Evolutionary Computation*, 23(1):89–103, 2018.

[26] Linpeng Tang, Lei Zhang, Ping Luo, and Min Wang. Incorporating occupancy into frequent pattern mining for high quality pattern recommendation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 75–84, 2012.

[27] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017.

[28] Ye Tian, Handing Wang, Xingyi Zhang, and Yaochu Jin. Effectiveness and efficiency of non-dominated sorting for evolutionary multi-and many-objective optimization. *Complex & Intelligent Systems*, 3:247–263, 2017.

[29] Mario Ventresca, Kyle Robert Harrison, and Beatrice M Ombuki-Berman. An experimental evaluation of multi-objective evolutionary algorithms for detecting critical nodes in complex networks. In *European Conference on the Applications of Evolutionary Computation*, pages 164–176. Springer, 2015.

[30] Bing Xue, Mengjie Zhang, and Will N Browne. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE transactions on cybernetics*, 43(6):1656–1671, 2012.

[31] Chong Zhang, Pin Lim, A Kai Qin, and Kay Chen Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, 28(10):2306–2318, 2016.

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

[33] Xingyi Zhang, Fuchen Duan, Lei Zhang, Fan Cheng, Yaochu Jin, and Ke Tang. Pattern recommendation in task-oriented applications: A multi-objective perspective [application notes]. *IEEE Computational Intelligence Magazine*, 12(3):43–53, 2017.

[34] Xingyi Zhang, Ye Tian, Ran Cheng, and Yaochu Jin. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on evolutionary Computation*, 22(1):97–112, 2016.

[35] Xingyi Zhang, Ye Tian, Ran Cheng, and Yaochu Jin. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on evolutionary Computation*, 22(1):97–112, 2016.

[36] Heiner Zille, Hisao Ishibuchi, Sanaz Mostaghim, and Yusuke Nojima. A framework for large-scale multiobjective optimization based on problem transformation. *IEEE Transactions on Evolutionary Computation*, 22(2):260–275, 2017.