arXiv:2301.03310v1 [math.OC] 9 Jan 2023

改进的多目标优化前沿最速下降法

MATTEO LAPUCCI
全局优化实验室（GOL）信息工程系
佛罗伦萨大学圣玛尔塔街3号，50139，佛罗伦萨，意大利
matteo.lapucci@unifi.it

PIERLUIGI MANSUETO
全局优化实验室（GOL）信息工程系
佛罗伦萨大学圣玛尔塔街3号，50139，佛罗伦萨，意大利
pierluigi.mansueto@unifi.it

摘要

本文研究多目标优化中的前沿最速下降算法。我们指出，文献中的算法在设计上往往无法覆盖帕累托前沿的很大一部分。因此，我们对该算法进行了一些改进，旨在克服这一重大局限。我们证明了该算法的渐近收敛性质得以保留，并通过数值结果表明，所提出的方法明显优于原始方法。

关键词 多目标优化 · 最速下降 · 帕累托前沿

数学主题分类（***2020***） 90C29 · 90C30

1 引言

本文关注以下形式的优化问题

$$\min_{x\in\mathbb{R}^n} \; F(x) = (f_1(x), \ldots, f_m(x))^T, \tag{1}$$

其中 $F: \; \mathbb{R}^n \to \mathbb{R}^m$ 是一个连续可微的向量值函数。因此，我们研究的是光滑、无约束的多目标优化问题，其中必须同时最小化多个函数，并考虑帕累托有效性概念以确定最优性。关于多目标优化的入门介绍，请参阅 [8]。

多目标下降法 [9–11, 16] 是一类旨在解决此类问题的算法方法；这些方法基本上将用于标量优化的经典迭代优化算法扩展到多目标场景。下降法日益受到关注，并已逐渐成为标量化方法 [6, 7, 15] 和进化算法 [4]的重要替代方案。对于近期专门设计用于处理点集并构造整个帕累托前沿近似，而非仅求得单个解的下降法，这一点尤其如此。

在这篇简短的论文中，我们关注的是 [2]提出的前沿最速下降 （FSD）算法。具体而言，我们认为，尽管 FSD 远优于原始的多目标优化单点最速下降算法 [10], , 但 [2] 中定义的 FSD 探索能力有限，而且往往无法覆盖帕累托前沿的较大部分。

因此，我们对该算法提出了小幅但至关重要的修改，使其无论起始点集如何，都能极其有效地覆盖整个帕累托前沿。我们证明，所提出的方法仍保留了原始 FSD 良好的收敛保证。

本文其余部分安排如下：第2节概述 FSD 算法，并回顾其收敛性质；随后在第2.1节指出，在某些常见情况下，该算法无法覆盖帕累托前沿的较大

部分。在第3节中，我们介绍一种在 FSD 中生成非支配解的新策略，并在第3.1节对由此得到的算法进行收敛性分析。在第4节中，我们展示数值实验结果，表明所提出的改进显著提升了 FSD 算法的有效性和一致性。最后，我们在第5节给出一些总结性意见。

2 前沿最速下降算法

前沿最速下降算法 [2] 旨在根据帕累托最优性概念求解问题 (1)。给定 $\mathbb{R}^m$中的标准偏序，即

$$u \leq v \iff u_j \leq v_j,\ \forall\, j = 1, \ldots, m,$$
$$u < v \iff u_j < v_j,\ \forall\, j = 1, \ldots, m,$$
$$u \lneqq v \iff u \leq v \wedge u \neq v,$$

目标是找到满足以下性质的解 $\bar{x} \in \mathbb{R}^n$，这些性质按强度递减排列：

- 帕累托最优：$\nexists\, y \in \mathbb{R}^n$ 使得 $F(y) \lneqq F(\bar{x})$；
- 弱帕累托最优：$\nexists\, y \in \mathbb{R}^n$ 满足 $F(y) < F(\bar{x})$；
- 帕累托驻点性：最小化$_{d \in \mathbb{R}^n}$ 最大化$_{j=1,\ldots,m}$ $\nabla f_j(\bar{x})^T d = 0$。

事实上，通常存在许多帕累托最优解（即帕累托集），它们对应着相互冲突的目标之间不同的权衡；这些权衡在帕累托前沿中体现于目标空间，决策者可以事后对其进行评估，因此他们希望拥有尽可能广泛的可选方案。

FSD 方法旨在构造整个帕累托前沿的近似；该算法以迭代方式运行，在每次迭代中维护一个由 X^k 个相互非支配解组成的集合，即对于任意 $x \in X^k$，不存在 $y \in X^k$ 使得 $F(y) \lneqq F(x)$。

集合中的点 是从点 $\hat{x} \in X^k$ 出发，沿以下方向执行搜索步计算得到的：

- 最速共同下降方向 [10]:

$$v(\hat{x}) = \arg\min_{d \in \mathbb{R}^n} \max_{j=1,\ldots,m} \nabla f_j(\hat{x})^T d + \frac{1}{2}\|d\|^2; \tag{2}$$

- 最速部分下降方向 [1,2]：给定 $I \subset \{1, \ldots, m\}$,

$$v^I(\hat{x}) = \arg\min_{d \in \mathbb{R}^n} \max_{j \in I} \nabla f_j(\hat{x})^T d + \frac{1}{2}\|d\|^2. \tag{3}$$

在最速下降方向的定义中使用等号是合理的，因为上述优化问题的解集具有唯一性（目标函数是强凸且连续的）。给定目标函数的任意子集I，当

$$\theta^I(\hat{x}) = \min_{d \in \mathbb{R}^n} \max_{j \in I} \nabla f_j(\hat{x})^T d + \frac{1}{2}\|d\|^2 < 0;$$

当然，当 $I = \{1, \ldots, m\}$ 时，会考虑最速共同下降方向 $v(\hat{x})$ 及其对应的 $\theta(\hat{x})$。两个映射 $v^I(\hat{x})$ 和 $\theta^I(\hat{x})$ 都是连续的 [10]。

FSD过程的步骤总结于算法1。简而言之，在每次迭代 k 中，都会考虑当前非支配解集合 X^k 中的所有点；对于其中的每个点 x_c，针对满足 $\theta^I(x_c) < 0$ 的任意目标子集 $I \subseteq \{1, \ldots, m\}$，沿最陡部分下降方向进行线搜索；此外，只有当某个点相对于目标子集 I 是非支配的时，才会针对 x_c 考虑该子集。

该线搜索采用一种 Armijo型过程，其方案见算法 2。给定一个非支配点以及相对于目标函数 I 的搜索方向，该算法会返回一个“充分非支配”的新点。所得点会被加入非支配点集合，同时筛除所有此时被该点支配的点。

算法 2 具有以下有限终止性质。

算法1： 前沿最速下降算法

输入：$F\colon \mathbb{R}^n \to \mathbb{R}^m$, X^0 关于 F 的相互非支配点集合。$k = 0$

当不满足停止准则时 **执行** $\hat{}X^k = X^k$ **对于** $x_c \in X^k$ **执行对于** $I \subseteq \{1, \ldots, m\}$ 满足

- $\nexists y \in \hat{}X^k$ 使得 $F_I(y) \lneqq F_I(x_c)$ 并且
- $\theta^I(x_c) < 0$ **执行** α =Armijo线搜索($F(\cdot)$ I, $\hat{}X^k$, x_c, $v^I(x_c)$ $\theta^I(x_c)$)$\hat{}X^k = \hat{}X^k \setminus \{y \in \hat{}X^k \mid F(x_c + \alpha v^I(x_c)) \lneqq F(y)\} \cup \{x_c + \alpha v^I(x_c)\}$ $X^{k+1} = \hat{}X^k$ $k = k + 1$

返回 X^k

算法 2： Armijo线搜索

输入：$F\colon \mathbb{R}^n \to \mathbb{R}^m$, $I \subseteq \{1, \ldots, m\}$, $\hat{}X$ 个关于 F, $x_c \in \hat{}X$, $v^I(x_c) \in \mathbb{R}^n$,$\theta^I(x_c) \in \mathbb{R}$, $\alpha_0 > 0$, $\delta \in (0, 1)$, $\gamma \in (0, 1)$ 的相互非支配点集合。

$\alpha = \alpha_0$ $\hat{}$ 令 X_I 为 X 中关于 F_I 相互非支配的点集**当** $\exists y \in \hat{}X_I$ 使得 $F_I(y) + \mathbf{1}\gamma\alpha\theta^I(x_c) < F_I(x_c + \alpha v^I(x_c))$ **时** $\alpha = \delta\alpha$

返回 α

命题1 ([2, 命题 4])。 设 $I \subseteq \{1, \ldots, m\}$, $\hat{}X$ 是包含 x_c 的相互非支配解集合； x_c 关于 F_I 也非支配，且满足 $\theta^I(x_c) < 0$。那么， $\exists \alpha > 0$ 足够小，使得

$$F_I(y) + \mathbf{1}\gamma\alpha\theta^I(x_c) \not< F_I(x_c + \alpha v^I(x_c)), \quad \forall\, y \in \hat{X}_I,$$

$\hat{}$ $\hat{}$其中， X_I 是 X 中关于 F_I彼此非支配的点集。此外，所生成的点 $x_c + \alpha v^I(x_c)$ 相对于 F不受$\hat{}X$ 中的任何点支配。

注1。 [2], 还提出了算法 2 的一个改进版本，该版本基于外推策略，能够沿搜索方向获得多个非支配解。在算法1中使用时，外推技术不会改变理论收敛结果，但据报道，改进后的算法效果显著。

现在，我们将回顾算法1的收敛性质，这些性质基于关联序列 [14]的概念。

定义 1。 设 $\{X^k\}$ 为算法1生成的非支配点集合序列。我们将满足以下条件的序列 $\{x_{j_k}\}$ 定义为关联序列：对于任意 $k = 1$， 2， $\ldots$， 点 $x_{j_k} \in X^k$ 由点 $x_{j_{k-1}} \in X^{k-1}$在算法1的第 $k - 1$ 次迭代中生成。

命题 2 ([2, 命题 5])。 假设存在 $x_0 \in X^0$ 使得

- x_0 不是帕累托驻点；
- *the set* $\mathcal{L}(x_0) = \bigcup_{j=1}^m \{x \in \mathbb{R}^n \mid f_j(x) \leq f_j(x_0)\}$ *is compact.*

设 $\{X^k\}$ 为算法*1*生成的非支配点集合序列。设 $\{x_{j_k}\}$ 为一个关联序列，则该序列存在极限点，且每个极限点都是问题(1)的帕累托驻点。

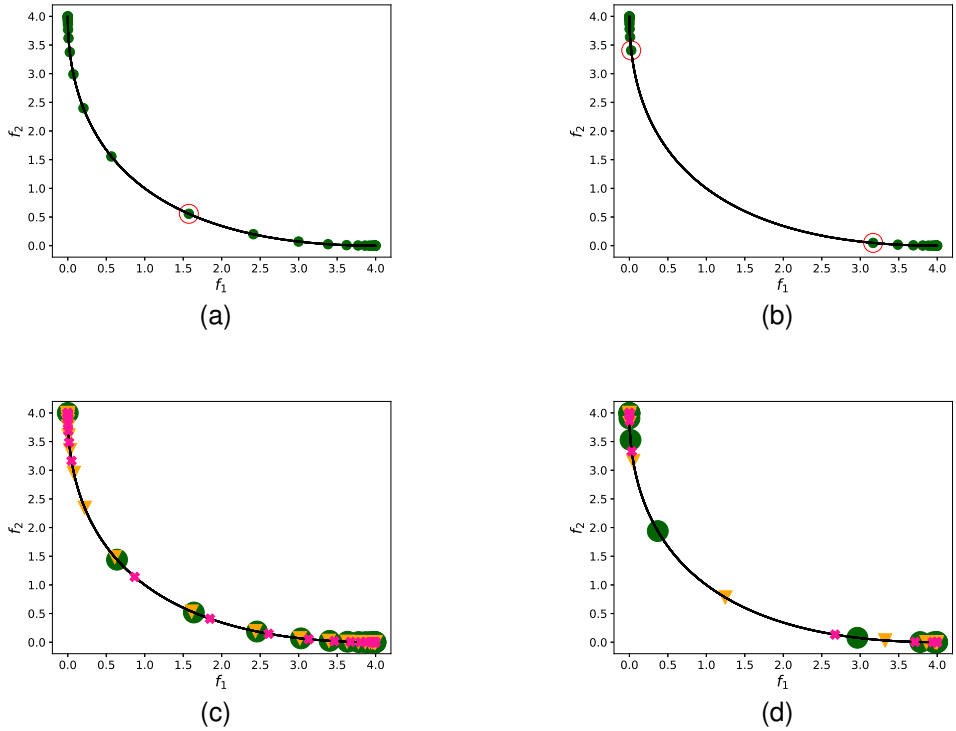


图 1：在凸 JOS 问题（$n = 5$）上通过 FSD 算法得到的帕累托前沿。(a) FSD 从 1 个帕累托点开始；(b) FSD 从 2 个帕累托点开始；(c) 从 3 个不同随机点开始的 3 次独立 FSD 运行；(d) 从与 (c) 中相同的 3 个随机点开始，采用外推策略进行的 3 次独立 FSD 运行。

2.1 FSD可能无法覆盖帕累托前沿

实际上，与用于多目标优化的简单多起点最速下降策略相比，FSD 算法有了显著改进。然而，在实验环境中，常常会观察到 FSD 无法获取帕累托前沿的很大一部分。

这里，我们着重指出这一缺陷，并认为它是算法设计直接导致的结果。具体而言，算法1第6步的第一个条件使算法的结果极大程度地取决于起始点。
当在算法1中考虑对点 x_c 进行探索时，只有当根据目标函数子集 $I \subseteq \{1, \dots, m\}$ 得到的部分下降方向满足 x_c 相对于 F_I 在 X^k 内非支配时，才会予以考虑；换言之，不存在 $y \in X^k$ 使得 $F_I(y) \lneqq F_I(x_c)$。算法2的作者要求满足这一条件，以确保线搜索具有有限终止性质。

遗憾的是，同一条件导致 X^k 中只有少数点会被用于启动部分下降搜索。在双目标情形下，可以看到这一现象的极端情况：事实上，当 $m = 2$时，对于两个指标真子集中的每一个，$I_1 = \{1\}$ 和 $I_2 = \{2\}$，都只有一个点满足（部分）非支配条件：$x_{I_1} = \text{argmin}_{x \in X^k} f_1(x)$ 和 $x_{I_2} = \text{argmin}_{x \in X^k} f_2(x)$。

因此，部分下降仅从帕累托前沿上的两个当前极端点开始执行。此外，这些部分下降步骤只能向外探索当前前沿近似的极端部分，而另一个下降步骤则主要使各点趋于帕累托驻点性；因此，即使当前解集中存在较大的空缺，也无法填补。

将这一推理推向极致，假设初始解集合已经位于帕累托前沿上；如果集合中只有一个点，那么通过反复针对 I_1 和 I_2 进行部分下降，就能相当均匀地遍历整个帕累托前沿；图 1a 展示了这种情况。另一方面，如果有两个初始解，且它们在目标空间中可能相距甚远，那么只能遍历前沿的两端部分，而中间的空缺

不会得到处理（图 1b）。当然，初始点多于两个时，同样的推理也适用。

该算法的行为看似矛盾：从远离帕累托前沿的位置开始，或许更为有利。这样一来，FSD 就有充足的迭代机会来扩大集合 X^k 的规模，并在目标空间中均匀铺展；随后，通过考虑 $I = \{1, 2\}$ 所采取的步长，将各点推进至帕累托驻点性。不过，初始解仍会以某种程度上随机的方式影响结果，如图 1c 所示。此外，前沿的极端部分总是比中间部分覆盖得密集得多。值得指出的是，由于前沿的中间区域往往能为用户提供最有趣的权衡，这在实践中是一个非常重要的问题。

[2] 中提出的外推技术或许能够在一定程度上缓解此处讨论的问题，因为每次迭代都能获得更多非支配解；然而，它再次主要加强并加快了对极端区域的探索，整体结果可能适得其反（图 1d）。

3 改进的前沿最速下降算法

算法3给出了改进的前沿最速下降（IFSD）算法方案，该算法克服了第2.1节中讨论的算法1的局限性。

算法3： 改进的前沿最速下降算法

输入：$F\colon \mathbb{R}^n \to \mathbb{R}^m$、$X^0$ 关于 F、$\alpha_0 > 0$的相互非支配点集合，$\delta \in (0, 1)$ $\gamma \in (0, 1)$。
$k = 0$
当停止准则未满足时**执行**
  $\hat{X}^k = X^k$
  对所有 $x_c \in X^k$ **执行**
    如果 $x_c \in X^k$ **则**
      如果 $\theta(x_c) < 0$ **则**
        $\alpha_c^k = \max_{h=0,1,\dots}\{\alpha_0\delta^h \mid F(x_c + \alpha_0\delta^h v(x_c)) \leq F(x_c) + \mathbf{1}\gamma\alpha_0\delta^h\theta(x_c)\}$
      else
        $\alpha_c^k = 0$
      $z_c^k = x_c + \alpha_c^k v(x_c)$
      $\hat{X}^k = \hat{X}^k \setminus \{y \in \hat{X}^k \mid F(z_c^k) \lneqq F(y)\} \cup \{z_c^k\}$
      对所有 $I \subseteq \{1, \dots, m\}$ **满足** $\theta^I(z_c^k) < 0$ **执行**
        如果 $z_c^k \in \hat{X}^k$ **那么**
          α_c^I = 最大值$_{h=0,1,\dots}\{\alpha_0\delta^h \mid \forall y \in \hat{X}^k \exists j \in \{1, \dots, m\}$ 使得 $f_j(z_c^k + \alpha_0\delta^h v^I(z_c^k)) < f_j(y)\}$
          $\hat{X}^k = \hat{X}^k \setminus \{y \in \hat{X}^k \mid F(z_c^k + \alpha_c^I v^I(z_c^k)) \lneqq F(y)\} \cup \{z_c^k + \alpha_c^I v^I(z_c^k)\}$
  $X^{k+1} = \hat{X}^k$
  $k = k + 1$
返回 X^k

算法3针对原始 FSD 方法做了一系列修改：

- 对于 X^k 中仍未被支配、正准备进行探索的任意点，先执行一个初步的最速下降步；该步骤采用经典的单点 Armijo 线搜索 [10];
- 只要所得点未被支配，就从该点开始针对目标子集进行进一步搜索；
- 对于部分下降搜索，我们要求所得点不被 $\hat{X}^k$ 中的任何其他点支配。

其基本思想是，通过这些修改，可以利用所有点基于部分下降进行探索；随后，通过“预备”最速下降步确保充分下降，从而迫使所有生成点趋向驻点性。下一节将证明该算法定义良好，并且确实会生成收敛序列。

3.1 收敛性分析

本节将对算法3进行正式的收敛性分析。

命题3. 算法*3*第*8*步的线搜索定义良好。

证明。 该结果由 [10, 引理 4] 以及第7步的 if 条件推出，该条件确保 $\theta(x_c) < 0$。 □

命题4. 若 z_c^k 相对于 $\hat{}X^k$ 中的点是非支配的，则算法*3*第*15*步定义良好。

证明。设 y 是 $\hat{}X^k$ 中的任意一点；若 $F(y) = F(z_c^k)$，则根据 [10, 引理 4] 和条件 $\theta^I(z_c^k) < 0$，存在 $\bar{\alpha} > 0$，使得对所有 $\alpha < \bar{\alpha}$，均有 $F_I(z_c^k + \alpha v^I(z_c^k)) < F_I(z_c^k) = F_I(y)$；因此，存在充分大的 h，使得对所有 $j \in I$，均有 $f_j(z_c^k + \alpha_0\delta^h v^I(z_c^k)) < f_j(y)$。另一方面，若存在 $j \in \{1, \ldots, m\}$，使得 $f_j(z_c^k) < f_j(y)$，则根据 F 的连续性，存在充分小的 $\alpha = \alpha_0\delta^h$，使得 $f_j(z_c^k + \alpha v^I(z_c^k)) < f_j(y)$。因此，对于所有 $y \in \hat{}X^k$，该条件均可满足，而 α_c^I 是 $\alpha_0\delta^h$ 对应取值中的最小值。 □

命题5。 如果 X^k 相对于 F包含相互非支配点，那么在迭代 k的任何时刻，$\hat{}X^k$ 都包含非支配点；因此步骤*15*始终有定义，并且 X^{k+1} 最终是一个非支配解集合。

证明。 在迭代 k时，集合$\hat{}X^k$ 初始化为非支配点 X^k；此后，仅在步骤12和16对其进行更新。在步骤12中，要么 $z_c^k = x_c$，集合不作修改；要么根据 α_c^k的定义， z_c^k支配 x_c，而 x_c原本是非支配点。因此，新增点 z_c^k是非支配点，同时移除所有新近被支配的点。在步骤16中，根据 α_c^I的定义，新增点 $z_c^k + \alpha_c^I v^I$（z_c^k）不受支配；所有新近被支配的点均被移除。因此，$\hat{}X^k$ 始终包含相互非支配解。由命题4可知，步骤15始终有定义。此外，由于在迭代结束时 $X^{k+1} = \hat{}X^k$， X^{k+1} 继承了$\hat{}X^k$的非支配性质。

□

引理1。 算法*3*的第*12*步之后， z_c^k属于$\hat{}X^k$。此外，对于所有 $\tilde{k} > k$，都存在 $y \in X^{\tilde{k}}$ 使得$F(y) \leq F(z_c^k)$。

证明。 命题的第一个断言显然可由第12步中$\hat{}X^k$的更新规则推出。现在，或者 $z_c^k \in X^k$，或者 $z_c^k \notin X^k$；在前一种情况下，显然有 $y = z_c^k$；否则，我们可以注意到，根据算法的指令，任意集合 $X^{\tilde{k}}$、$\tilde{k} > k$都是从某个满足 $z_c^k \in \hat{}X^k$的时刻的$\hat{}X^k$ 出发，反复执行第12步和第16步的结果。当 z_c^k从集合中移除时，必然插入了一个点 y^1，使得 $F(y^1) \leq F(z_c^k)$。然后，或者 $y^1 \in X^k$，或者 y^1 是在加入一个满足 $F(y^2) \leq F(y^1)$的点 y^2时被移除的。递归应用上述推理可知，必然存在一个点 $y^t \in X^k$，使得 $F(y^t) \leq F(y^{t-1}) \leq \ldots \leq F(y^2) \leq F(y^1) \leq F(z_c^k)$。证毕。 □

命题6. 设 X^0 是一组相互非支配点， $x_0 \in X^0$ 是一个点，且集合 $\mathcal{L}(x_0) = \bigcup_{j=1}^m \{x \in \mathbb{R}^n \mid f_j(x) \leq f_j(x_0)\}$ 是紧致的。设 $\{X^k\}$ 是由算法*3*生成的非支配点集序列。设 $\{x_{j_k}\}$ 是一个关联序列，则它存在极限点，并且每个极限点都是问题 (1) 的帕累托驻点。

证明. 对于任意 k，要么 $x_0 \in X^k$，要么 $x_0 \notin X^k$。在前一种情况下，由于 X^k 中的所有点都相互非支配，我们显然有 $x_{j_k} \in \mathcal{L}(x_0)$。否则，按照引理1证明中的类似推理可知，存在一点 $y_k \in X^k$，使得 $F(y_k) \leq F(x_0)$；由于 y_k 不支配 x_{j_k}，因此存在 $h \in \{1, \ldots, m\}$，使得 $f_h(x_{j_k}) \leq f_h(y_k) \leq f_h(x_0)$；于是，同样有 $x_{j_k} \in \mathcal{L}(x_0)$。因此，整个序列 $\{x_{j_k}\}$ 属于紧集 $\mathcal{L}(x_0)$，从而存在极限点。

现在，令u 我们考虑关联序列 $\{x_{j_k}\}$的一个极限点 $\bar{x}$，即存在 $K \subseteq \{1, 2, \ldots\}$ 使得 at

$$\lim_{\substack{k \to \infty \\ k \in K}} x_{j_k} = \bar{x}.$$

我们通过反证法假设 $\theta(\bar{x}) < 0$，因此存在 $\varepsilon > 0$，使得对于所有充分大的 $k \in K$，都有$\theta(x_{j_k}) \leq -\varepsilon < 0$。令 $z_{j_k} = x_{j_k} + \alpha_{j_k} v(x_{j_k})$ 为从 x_{j_k} 出发，算法在第 11 步得到的点。现在，$\alpha_{j_k} \in [0, \alpha_0]$ 是一个紧集，因此存在一个进一步的子序列 $K_1 \subseteq K$，使得 $\alpha_{j_k} \to \bar{\alpha} \in [0, \alpha_0]$。

此外，函数 $v(\cdot)$ 是连续的，因此对于 $k \to \infty$， $k \in K_1$，有 $v(x_{j_k}) \to v(\bar{x})$。因此，沿着 K_1 取极限，我们还得到 $z_{j_k} \to \bar{x} + \bar{\alpha} v(\bar{x}) = \bar{z}$。
根据 α_{j_k} 和 z_{j_k} 的定义（步骤 8-11），我们有

$$F(z_{j_k}) \leq F(x_{j_k}) + \mathbf{1}\gamma\alpha_{j_k}\theta(x_{j_k}).$$

对 $k \in K_1$、 $k \to \infty$取极限，并回顾 $\theta(\cdot)$的连续性，可得

$$F(\bar{z}) \leq F(\bar{x}) + \mathbf{1}\gamma\bar{\alpha}\theta(\bar{x}) \leq F(\bar{x}) - \mathbf{1}\gamma\bar{\alpha}\varepsilon. \tag{4}$$

现在，给定 $k \in K_1$，令 $k_1(k)$ 为 K_1 中满足 $k_1(k) > k$的最小索引。根据引理1，存在 $y_{j_{k_1(k)}} \in X^{k_1(k)}$，使得 $F(y_{j_{k_1(k)}}) \leq F(z_{j_k})$；此外， $x_{j_{k_1(k)}} \in X^{k_1(k)}$；根据命题5， $X^{k_1(k)}$ 中的点两两非支配，因此存在 $h(k) \in \{1, \ldots, m\}$，使得

$$f_{h(k)}(x_{j_{k_1(k)}}) \leq f_{h(k)}(y_{j_{k_1(k)}}) \leq f_{h(k)}(z_{j_k}).$$

考虑一个进一步的子序列 $K_2 \subseteq K_1$，使得对所有 $k \in K_2$ 均有 $h(k) = h$，并取极限，可得

$$f_h(\bar{x}) \leq f_h(\bar{z}).$$

将这一最后结果与（4）结合起来，我们得到

$$f_h(\bar{x}) \leq f_h(\bar{z}) \leq f_h(\bar{x}) - \gamma\bar{\alpha}\varepsilon.$$

由于 $\bar{\alpha} \in [0, \alpha_0]$、 $\varepsilon > 0$ 和 $\gamma > 0$，上述不等式链只有在 $\bar{\alpha} = \lim_{k\to\infty, k\in K_2} \alpha_{j_k} = 0$ 时才能成立。对于所有充分大的 $k \in K_2$，都有 $\theta(x_{j_k}) < 0$，因此在第 8 步定义了 α_{j_k}。由于 $\alpha_{j_k} \to 0$，对于任意$q \in \mathbb{N}$，当 $k \in K_2$ 足够大时，我们必然有 $\alpha_{j_k} < \alpha_0\delta^q$；因此， $\alpha = \alpha_0\delta^q$ 不满足 Armijo条件 $F(x_{j_k} + \alpha v(x_{j_k})) \leq F(x_{j_k}) + \mathbf{1}\gamma\alpha\theta(x_{j_k})$，即存在 $h(k)$ 使得

$$f_{\tilde{h}(k)}(x_{j_k} + \alpha_0\delta^q v(x_{j_k})) > f_{\tilde{h}(k)}(x_{j_k}) + \gamma\alpha_0\delta^q\theta(x_{j_k}).$$

~ ~沿着一个合适的子序列取极限，使得 $h(k) = h$，我们得到

$$f_{\tilde{h}}(\bar{x} + \alpha_0\delta^q v(\bar{x})) \geq f_{\tilde{h}}(\bar{x}) + \gamma\alpha_0\delta^q\theta(\bar{x}).$$

Now，因为 q 是任意的且 $\theta(\bar{x}) < 0$，根据 [10, 引理 4]，这显然不成立。因此证明已完成 ete. □

4 数值结果

本节展示计算实验的结果，以支持第 2 至第 3 节中的讨论。代码使用 Python3 编写，并在一台具有以下配置的计算机上运行：Ubuntu 22.04、Intel Xeon ProcessorE5-2430v2 6 核 2.50 GHz、16GB RAM。为求解问题 (2)–(3) 的实例，使用了 Gurobi 优化器（版本 9.5）。

我们将所提出的方法（IFSD）与原始 FSD（算法1）进行了比较，后者采用基础线搜索（算法 2）或外推策略（EFSD）。线搜索采用了以下参数设置： $\alpha_0 = 1$， $\delta = 0.5$， $\gamma =10^{-4}$。

关于算法3中的概念方案，我们在 IFSD 中采用了一种策略，以限制用于部分下降搜索的点数，从而提高整体流程的效率，并避免产生过多彼此非常接近的解。具体而言，我们添加了一个基于拥挤距离 [4] 的条件，以判断一个点在最速下降步之后是否应继续探索。

用于比较的基准测试集由以下无约束问题组成：CEC09_2、CEC09_3 [17]、JOS_1 [12]、MAN [13] ($m = 2$) 和 CEC09_10 ($m = 3$) [17]。对于所有问题，我们考虑了 n 取值属于 $\{5, 10, 20, 30, 40, 50, 100, 200\}$的实例。此外，每个问题均采用不同的初始点策略进行两次测试：a) 从适当的盒体超对角线上均匀采样 n 个点；b) 仅选取超对角线的中点。超对角线指构成 CEC 和 MAN 问题有界版本约束的盒体的超对角线；对于 JOS 问题，它则是 $[-100, 100]^n$。
为了评估这些方法的相对性能和稳健性，我们采用了性能剖面 [5]。简而言之，该工具展示了在某个问题中，一种方法得到的指标值在最佳值的 $\tau \in \mathbb{R}$ 倍范围内的概率；这里的最佳值是该问题中所有算法所得到的最佳值。我们采用了多目标优化中的经典指标：纯度、 Γ–扩散度、 Δ–扩散度 [3]和超体积 [18]。纯度和超体积

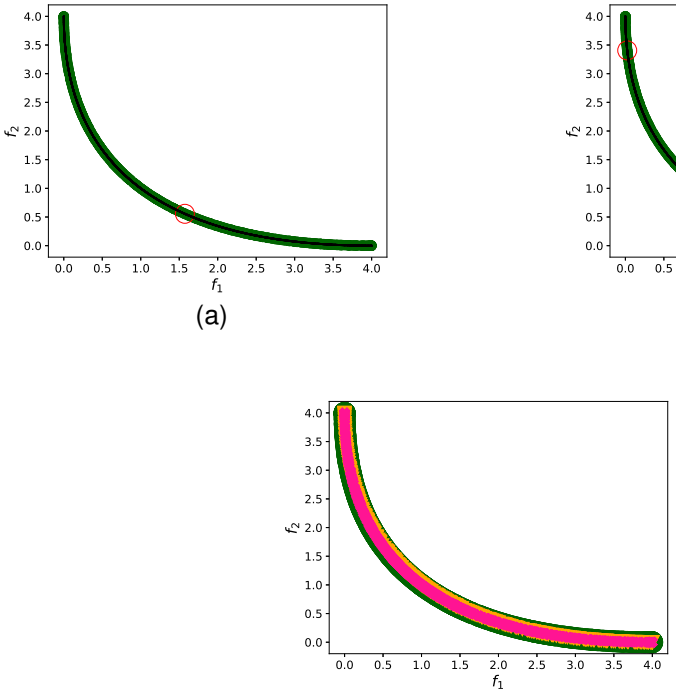


图 2：IFSD 算法在凸 JOS 问题（$n = 5$）上从不同初始点出发得到的帕累托前沿：(a) 1 个帕累托点，如图 1a所示；(b) 2 个帕累托点，如图1b所示；(c) 从与图 1(c)-(d) 中相同的随机点出发进行的 3 次独立运行。

对于更优的解，其值会增大：因此，生成相应的性能剖面时，会使用所得值的倒数。

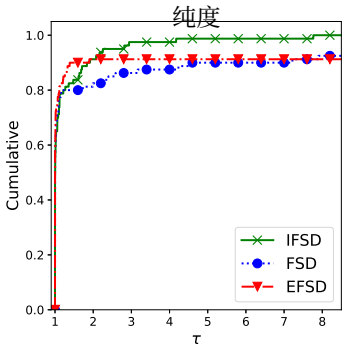
图 2 展示了所提出方法在与图 1 相同设置下的表现。在这个例子中，我们可以观察到，无论起始点如何，整个帕累托前沿现在都得到了有效覆盖，甚至没有微小的空缺。

为了更一致地评估算法的性能，我们在图 3 中报告了 IFSD、FSD 和 EFSD 算法在包含 80 个问题实例的完整基准测试集上的性能剖面。
与该算法的原始变体相比，我们观察到所提出的方法具有显著优势，尤其是在分布度指标方面，这表明帕累托前沿确实得到了更广泛、更均匀的覆盖。出色的超体积性能也支持这一结果。至于纯度指标，三种算法的表现似乎较为接近，但我们仍观察到 IFSD 略占优势。

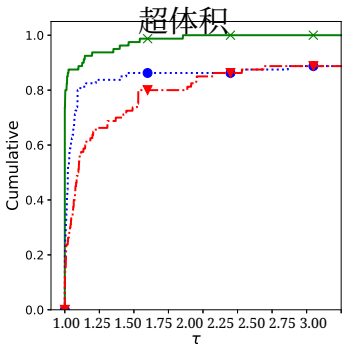
5 结论

本文提出了一种改进的前沿最速下降算法，其渐近收敛保证与原方法类似。该新算法旨在克服 FSD 的一些明显的经验局限，即 FSD 往往无法覆盖帕累托前沿的较大部分。数值结果表明，所提出的方法有效地实现了这一目标。

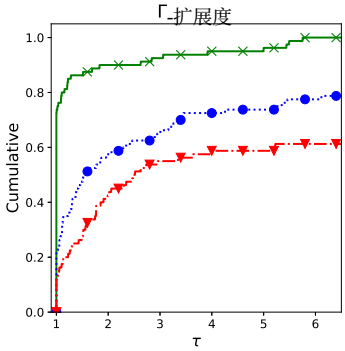
未来的工作应重点关注将本文提出的方法与 [2] 中提出的外推策略相结合。此外，还可以考虑在全局多目标优化 [13] 的模因算法流程中采用本文提出的方法。最后，本文定义的算法可以扩展用于处理约束优化问题。



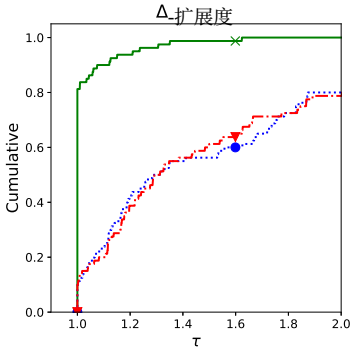
(a) 纯度曲线



(b) 超体积曲线



(c) Γ-扩散曲线



(d) Δ-扩散曲线

图 3：性能 IFSD、FSD 和 EFSD 算法在包含 80 个多目标问题的基准测试集上的性能概况 .

利益冲突

作者声明不存在利益冲突。

参考文献

[1] G. Cocchi、M. Lapucci 和 P. Mansueto。通过多目标增广拉格朗日方法近似帕累托前沿。《*EURO*计算优化期刊》，第 100008 页，2021 年。[2] G. Cocchi、G. Liuzzi、S. Lucidi 和 M. Sciandrone。关于多目标优化最速下降法的收敛性。《计算优化与应用》，第 1–27 页，2020 年。[3] A. L. Custódio、J. F. A. Madeira、A. I. F. Vaz 和 L. N. Vicente。用于多目标优化的直接多重搜索。*SIAM*优化期刊，21(3):1109–1140，2011 年。[4] K. Deb、A. Pratap、S. Agarwal 和 T. Meyarivan。一种快速且具精英策略的多目标遗传算法：NSGA-II。《*IEEE*进化计算汇刊》，6(2):182–197，2002 年。[5] E. D. Dolan 和 J. J. Moré。利用性能剖面对优化软件进行基准测试。《数学规划》，91(2):201–213，2002 年。[6] L. G. Drummond、N. Maculan 和 B. F. Svaiter。向量优化中加权法参数的选择。《数学规划》，111(1-2):201–216，2008 年。[7] G. Eichfelder。多目标优化中的一种自适应标量化方法。*SIAM*优化期刊，19(4):1694–1718，2009 年。[8] G. Eichfelder。二十一世纪连续多目标优化的二十年发展。《*EURO*计算优化期刊》，9:100014，2021 年。

[9] J. Fliege、L. G. Drummond 和 B. F. Svaiter。多目标优化的牛顿法。*SIAM*优化期刊，20(2):602–626，2009 年。[10] J. Fliege 和 B. F. Svaiter。多准则优化的最速下降法。运筹学数学方法，51(3):479–494，2000 年。[11] M. L. N. Gonçalves、F. S. Lima 和 L. F. Prudente。用于多目标优化的全局收敛牛顿型方法。《计算优化与应用》，83(2):403–434，2022 年 11 月。[12] Y. Jin、M. Olhofer 和 B. Sendhoff。进化多目标优化的动态加权聚合：为何有效以及如何实现？载于遗传与进化计算会议论文集，第 1042–1049 页，2001 年。[13] M. Lapucci、P. Mansueto 和 F. Schoen。用于全局多目标优化的模因算法过程。*Mathe-matical ProgrammingComputation*，2022 年。[14] G. Liuzzi、S. Lucidi 和 F. Rinaldi。一种用于约束多目标非光滑优化的无导数方法。*SIAM*优化期刊，26(4):2744–2774，2016 年。[15] A. Pascoletti 和 P. Serafini。向量优化问题标量化。优化理论与应用期刊，42(4):499–524，1984 年。[16] H. Tanabe、E. H. Fukuda 和 N. Yamashita。多目标优化的近端梯度法及其应用。《计算优化与应用》，72(2):339–361，2019 年 3 月。[17] Q. Zhang、A. Zhou、S. Zhao、P. Suganthan、W. Liu 和 S. Tiwari。CEC 2009特别专题与竞赛的多目标优化测试实例。机械工程，2008 年 1 月。[18] E. Zitzler 和 L. Thiele。使用进化算法进行多目标优化——一项比较案例研究。载于 A. E. Eiben、T. Bäck、M. Schoenauer 和 H.-P. Schwefel 编，并行自然问题求解——*PPSN V*，第 292–301 页，柏林、海德堡，1998 年。施普林格·柏林·海德堡。