arXiv:2304.06881v3 [math.OC] 9 Jan 2025

设计求解多目标仿真优化问题的框架

Tyler H. Chang[1] 和 Stefan M. Wild[2]

[1]阿贡国家实验室数学与计算机科学部,, 9700 S Cass Ave Bldg 240, Lemont, IL, 美国 60439, tchang@anl.gov[2]劳伦斯伯克利国家实验室应用数学与计算研究部,, 1 Cyclotron Rd, Berkeley,CA,美国 94720, wild@lbl.gov

2025年1月13日

摘要

多目标仿真优化（MOSO）问题是具有多个相互冲突目标的优化问题，其中至少一个目标的评估取决于黑箱数值代码或现实世界实验，我们将其称为仿真。尽管已有大量研究致力于开发用于求解此类及相关问题的新算法和方法，但将这些技术集成到可用于生产的现实世界求解器中仍颇具挑战且耗时。这部分是由于现代最先进的 MOSO 算法和方法种类繁多且复杂，部分是由于许多现实世界问题及其相应计算环境的复杂性和特殊性。引入可能复杂和/或特定领域的代理建模技术、问题表述、设计空间和数据采集函数，只会进一步增加这一问题的复杂性。本文全面考察了当前最先进的 MOSO 算法、技术和求解器，以及常见的 MOSO 应用问题类型和计算环境。随后，我们介绍了设计并行多目标仿真优化框架（ ParMOO ）时面临的若干关键挑战，以及应对这些挑战的方法。最后，我们提供了两个案例研究，展示如何快速构建并部署定制的 ParMOO 求解器，以解决现实世界的多目标仿真优化问题。

关键词： 多目标优化、仿真优化、工程设计优化、代理建模、开源软件设计

1 引言与动机

多目标优化问题（MOOP）是指涉及两个或多个可能相互冲突目标的优化问题。这类问题广泛存在于科学的众多领域，例如多学科工程设计 [1, 2], 科学模型校准 [3], 高性能计算（HPC）库自动调优 [4], 粒子加速器设计 [5], 神经网络架构搜索 [6, 7, 8], 以及计算化学 [9, 10]。这类问题的目标是从设计空间中找到一组相对于向量值目标函数 $\mathbf{F} : \mathcal{X} \to \mathbb{R}^o$而言“良好”的点。这里， $\mathcal{X}$ 称为可行设计空间，通常假设它是 $\mathbb{R}^n$ 的一个紧致、简单有界的子区域。本文假设每个目标 $F_i(\mathbf{x})$都有下界，因此可行目标空间 $\mathbf{F}(\mathcal{X})$在每个维度上都有下界。在标准表述中，目标是最小化每个目标（即$\mathbf{F}$的每个分量），该问题写作

$$\min_{\mathbf{x}\in\mathcal{X}} \mathbf{F}(\mathbf{x}). \tag{1}$$

由于通常无法找到一个能同时最小化$\mathbf{x} \in \mathcal{X}$ 的所有 o 分量的$\mathbf{F}$，因此，**(1)** 的解通常是一组设计点及其对应的目标值。与其

描述某个目标值何时对应问题 (1) 的一个解时，不妨先说明何时目标值不对应解。若存在$\mathbf{y} \in \mathcal{X}$使得$\mathbf{F(y)} \leq \mathbf{F(x)}$，则称目标值$\mathbf{F(x)} \in \mathbf{F}(\mathcal{X})$受支配。这里我们使用向量不等式“$\leq$”表示$\mathbf{F(y)}$逐分量小于或等于$\mathbf{F(x)}$，且在至少一个分量上严格小于 $F_i(\mathbf{y}) < F_i(\mathbf{x})i$。反之，若对于所有$\mathbf{F(y)} \in \Omega$，都有$\mathbf{F(y)} \not\leq \mathbf{F(x^\star)}$，则称目标值$\mathbf{F(x^\star)}$在目标值集合 $\Omega \subset \mathbb{R}^o$ 中是非支配的。

如果$\mathbf{F(x^\star)}$在$\mathbf{F}(\mathcal{X})$中是非支配的，则称$\mathbf{F(x^\star)}$是问题 (1) 的帕累托最优解，并称$\mathbf{x}^\star$ 是问题 (1) 的有效解。所有帕累托最优目标值构成的集合通常形成一个 $(o-1)$ 维权衡曲面，称为帕累托前沿（沿着$\mathbf{F}(\mathcal{X})$边界的某个子集）。关于一般多目标优化问题的更多细节，请参阅 [11] 的著作。一般来说，如果某个设计点与目标值对 $(\mathbf{x}^\star, \mathbf{F(x^\star)})$ 是有效的/帕累托最优的，则它们属于问题 (1) 的解集。在典型的多目标优化应用中，决策者会使用多目标优化求解器，从所有帕累托最优目标值中确定一个子集，并依据领域专业知识和个人偏好，从中选出一个或多个作为首选解。

一大类实际多目标优化问题源自计算成本高昂的数值模拟 [3, 5,1, 2], 计算机实验 [4, 6, 7, 8], 和实际实验 [12, 13, 9, 10]。我们将这类问题称为多目标仿真优化（MOSO）问题。通常，MOSO 问题中的仿真属于黑箱过程，评估成本高昂，且无法获得导数信息。尽管目标函数本身可能包含评估成本较低和/或能够获得导数信息的部分，但整个 MOSO 问题通常仍被视为黑箱，并使用黑箱优化技术求解 [14]。此外，由于仿真评估预计成本高昂，多目标仿真优化算法的性能通常按总仿真评估次数所对应的解的精度来衡量。本文重点关注 MOSO 问题求解器的设计。因此，本文约定“仿真”指的是仅提供零阶信息的高成本黑箱函数。正如本文后文所述，求解 MOSO 问题并不总是排斥使用某些偏导数信息。不过，我们仅考虑那些将计算流程中某处存在计算成本高昂的黑箱过程纳入考量的技术，并假设无法获得所有目标函数和约束的完整梯度。

从优化研究人员的角度来看，要有效地解决 MOSO 问题，需要协调来自多个研究领域的最先进技术与新兴数值技术，包括试验设计、代理建模、标量化、不确定性量化和优化算法。我们将在第2.1节详细介绍其中许多技术。

从实践者的角度来看，解决现实世界的多目标仿真优化问题涉及的不仅仅是数值算法。首先，即使是在范围较窄的多目标仿真优化问题领域内，问题定义也可能存在显著差异。这包括应用领域和领域特定建模技术、问题结构以及可行设计空间。其次，考虑问题求解所需的资源和计算环境也很重要。这包括湿实验室环境和高性能计算资源，而这些资源可能需要额外的步骤和技术才能有效运行。我们将在第2.2节详细阐述这些挑战。

将上述来自不同研究领域的所有技术协调整合为可用于解决实际问题的求解器，是一项庞大且耗时的工作，往往会限制新方法的采用速度。并行多目标仿真优化（ParMOO）库实现了一个基于代理建模、用于求解 MOSO 问题的框架，旨在提供易于使用且模块化的接口，以便实现和部署各种多目标仿真优化算法及新兴技术。ParMOO 最初在 [15], 中介绍，重点关注开源软件流程的质量、目标用户群体以及易用性。本文介绍了持续推动 ParMOO 设计与开发的考量因素，以及 ParMOO 目前支持的多目标仿真优化算法和技术。我们首先将基于代理模型的多目标仿真优化求解器拆解为由简单模块组成的系统，用于执行多种求解器架构中常见的子任务。随后，我们提出一个用于集成这些模块的抽象框架，同时降低复杂性。为了展示 ParMOO 利用现实世界问题结构的有效性，最后我们提供了两个计算成本较低的基准问题，体现了上述一些具有挑战性的特征。这些问题易于测试，因为其中昂贵的仿真已由基于现实世界数据集训练的机器学习模型取代。

本文其余部分的组织结构如下。第2节回顾 MOSO 中现有的技术、问题类型、发展趋势和软件，并阐述构建通用 MOSO 库时涉及的技术挑战与权衡，以及如何实现我们的所有设计目标。第3节介绍我们用于应对这些挑战的框架。第4节针对一个知名的多目标优化测试问题，开展简要的性能研究与验证。随后，我们展示两个案例研究，以说明该框架的灵活性和强大功能：第5节涉及物理模型的校准，第6节涉及在湿实验室环境中进行材料设计。最后，第7节简要讨论研究结果。

2 背景与设计原则

现在，我们将阐述设计通用框架以实现现代基于代理模型的多目标仿真优化算法时的主要目标和要求。这些目标和要求有赖于对多目标仿真优化领域当前最先进技术和现代趋势的了解，因此我们先对其进行回顾。

根据决策者在优化过程中何时表达对解的偏好，多目标优化算法可分为许多不同类别。本文关注事后方法，即决策者在问题求解完成后表达偏好。其他方法，如先验方法和交互式方法，则需要采用不同的技术。 [16]的综述对这些方法进行了总结。

将事后 MOSO 方法进一步划分为文献中常见的三大类很有帮助：多目标进化算法和元启发式算法（MOEAs），它们依靠对先前评估过的设计点所构成的所谓“池”“档案”或“种群”进行随机组合和/或扰动；多目标直接搜索（MDS）方法，它们通过对一个或多个当前迭代点附近有前景的设计点进行（通常是确定性的）评估，来寻找多目标下降方向；以及多目标贝叶斯优化（MBO），它专注于最小化多目标采集函数，以选择待评估的设计点。第2.1节总结了上述各类方法以及当前用于求解 MOSO 问题的代表性算法。此外，还有其他方法——主要基于通用代理建模或响应面法（RSM）——无法明确归入上述任何类别。我们也将在第2.1节探讨这些技术。

2.1 求解 MOSO 问题的基本方法

现在，我们总结多目标仿真优化算法中常见的关键技术，重点介绍与本文工作最相关的技术；完整综述请参见 [17]。

我们首先讨论标量化。这是一种经典技术，至今仍广泛应用于许多现代多目标仿真优化算法及其应用中。标量化几乎是所有先验方法的基础，因为它将一个多目标优化问题转化为一个单目标子问题，后者可以通过单目标优化求解器进行（近似）求解。在先验方法中，决策者可以提供一组固定的标量化函数，并经过仔细选择，使其只针对决策者预先指定的所需解点。然而，在后验方法中，可以使用一系列多个标量化函数，生成对有效集和/或帕累托前沿的离散近似。

标量化由标量化函数 $Q : \mathbb{R}^o \to \mathbb{R}$ 定义，使得

$$\arg\min_{\mathbf{x}\in\mathcal{X}} Q(\mathbf{F}(\mathbf{x})) \subset \arg\min_{\mathbf{x}\in\mathcal{X}} \mathbf{F}(\mathbf{x}).$$

标量化问题 $\min_{\mathbf{x}\in\mathcal{X}} Q(\mathbf{F}(\mathbf{x}))$ 可以使用任意无导数优化方法求解。最常用的标量化技术是加权和标量化 [11, Ch. 3], ，该方法通过加权平均将目标函数合并。其他常见的标量化技术包括 ε-约束法 [11, Ch. 4], Pascoletti-Serafini 标量化 [18], 参考点法 [19], 二次标量化方案 [20],以及（增广）加权切比雪夫标量化 [21]。

为了近似完整的帕累托前沿，上述每种方法都需要结合某种自适应方案，以遍历一系列标量化函数中的不同成员，从而生成大量覆盖帕累托前沿的解点[22, 23, 18, 24]。尽管标量化是一种经典技术，但它仍广泛应用于多目标进化算法中 [25, 26],

MDS [27, 23], 和多目标贝叶斯优化方法 [28]。即使是不明确依赖标量化的算法，也会间接使用其中一些标量化方法。例如，MDS方法 MOIF [29]和线搜索技术 DFMO [30]都会选择一个可从标量化角度理解的多目标最速下降方向。

许多较新的多目标仿真优化算法（尤其是多目标进化算法）已用指标函数取代标量化函数，该函数着重评估加入新点$\mathbf{F}(\mathbf{x})$后帕累托前沿近似的质量。给定由先前评估过的设计点和目标值组成的数据集 $\mathcal{D} = \{(\mathbf{x}_i,\ \mathbf{F}(\mathbf{x}_i))\}_{i=1}^{N}$，指标函数 $I(\mathcal{D})$会根据 $\mathcal{D}$中的非支配目标值，计算帕累托前沿近似的质量。设计一个良好的指标函数也并非易事，但人们普遍认为，良好的指标应能产生位于真实帕累托前沿上或接近该前沿的解点，充分覆盖整个帕累托前沿，并且分布均匀 [31, 32]。通过选择标量化函数 $Q_{\mathcal{D}}(\mathbf{x}) = I(\mathcal{D} \cup \{(\mathbf{x},\ \mathbf{F}(\mathbf{x}))\}$ 或 $Q_{\mathcal{D}}(\mathbf{x}) = -I(\mathcal{D} \cup \{(\mathbf{x},\ \mathbf{F}(\mathbf{x}))\}$，指标函数便可同时发挥标量化函数及其自适应方案的作用。这里的关键区别在于，由指标函数驱动的标量化除了取决于$\mathbf{F}(\mathbf{x})$的值外，还取决于数据集 $\mathcal{D}$。多目标仿真优化算法中最常用的指标之一是超体积改进指标，它衡量潜在目标值与先前观测到的目标值之间的超体积 [33, 34]。许多多目标算法都以此指标为基础，例如存档多目标模拟退火（AMOSA） [35]。

由于每个标量化函数都会将一个 MOSO 问题简化为单目标子问题，因此这种技术必须与单目标求解器结合使用。这类求解器可以实现单目标进化算法、启发式方法、直接搜索方法、贝叶斯优化求解器或任何其他单目标方法。也可以将另一种 MOSO 求解器（或其单次迭代）应用于该 MOSO 问题，然后事后选出最能使标量化函数最小化的解。关于可用于上述情境的无导数优化方法的最新综述，请参见 [36]。

对于仿真评估成本高昂的 MOSO 问题，如果需要进行过多的仿真评估，问题的总开销可能会难以承受。因此，许多现代 MOSO 求解器将上述一种或多种多目标算法与计算成本较低的代理模型相结合，以减少对真实仿真评估的需求 [37, 32, 38]。在这里，昂贵的目标函数 $\mathbf{F}(\mathbf{x}) = (F_1(\mathbf{x}) \dots, F_o(\mathbf{x}))$ 的每个分量，都根据当前数据集 $\mathcal{D}$ 建模为计算成本较低的代理函数 $\hat{}\mathbf{F}_{\mathcal{D}} = \left(\hat{}F_1, \dots, \hat{}F_o\right)$，且满足 $\hat{}\ F_1 \approx F_1, \dots, \hat{}F_o \approx F_o$。随后，可以通过寻找在代理问题中有效或帕累托最优的点，提出若干“候选解”

$$\min_{\mathbf{x} \in \mathcal{X}} \left(\hat{F}_1(\mathbf{x}), \dots, \hat{F}_o(\mathbf{x})\right). \tag{2}$$

$\hat{}$ 当从计算时间和资源的角度来看，评估 $\mathbf{F}$ 的成本远高于评估 $\mathbf{F}_{\mathcal{D}}$ 时，求得问题 (2) 的（近似）有效解/帕累托最优点的成本，可能远低于求得问题 (1) 的此类点的成本。实际应用中，这些代理模型可以是各种经典近似方法或机器学习 [39]模型。

使用代理模型的主要挑战在于，代理模型必须渐近准确，优化算法才能收敛到准确的解。通常，代理模型的准确性取决于数据集 $\mathcal{D}$ 在 $\mathcal{X}$中的几何形态和密度、底层函数 $\mathbf{F}$的分量光滑性，以及代理模型 [40]的选择。在大多数应用中，$\mathbf{F}$ 的光滑性是未知但固定的。因此，为确保代理模型具有可接受的准确性，我们必须依赖$\mathcal{D}$ 的密度和几何形态，或依赖代理模型返回的不确定性信息。

在基于模型的无导数优化（DFO）中，人们关注的是 $\mathcal{D}$ 围绕当前迭代点 $\mathbf{x}^{(k)}$的局部性质。随着优化算法在运行过程中评估越来越多的设计点，局部极小值附近的 $\mathcal{D}$ 密度会迅速增加。因此，重点是在此类极小值邻域内，保持代理建模问题具有可接受的条件性，这由 $\mathcal{D}$ [41] 中设计点的局部几何决定。偶尔要求优化算法评估改进模型的设计点，而不是严格评估模型最小化点 [42]，即可维持这种几何结构。由于这些技术关注代理模型的局部质量，因此代理问题（2）必须受到局部信赖域（LTR）的约束，使优化算法仅提出 LTR 内的候选解。目前已有少数代理驱动的多目标单目标优化

据我们所知，完全遵循此方法的算法 [43, 44]，还没有配套软件。其他现有算法则采用了此方法的某些要素，或在理念上与之相似 [32, 45]。在单目标问题中，已知这种“基于模型的无导数优化”方法能够产生收敛速度快、可扩展性强的算法，但只能保证局部收敛 [46]。不过，如果需要全局精度，也可以将此方法与全局搜索或多起点技术结合使用 [47]。

另一方面，贝叶斯优化利用代理模型特有的不确定性模型。某些代理模型（如高斯过程）可以给出不确定性函数$\boldsymbol{\Sigma}_{\hat{\mathbf{F}},\,\mathcal{D}}:\ \mathcal{X}\to\mathbb{R}^{o}_{+}$使得$\boldsymbol{\Sigma}_{\hat{\mathbf{F}},\,\mathcal{D}}(\mathbf{x})\propto\left|\mathbf{F}(\mathbf{x})-\hat{}\mathbf{F}_{\mathcal{D}}(\mathbf{x})\right|$（其中“$\propto$”可理解为“逐分量成比例”） [48]。在多目标贝叶斯优化中，标量化函数/指标函数会结合$\boldsymbol{\Sigma}_{\hat{\mathbf{F}},\,\mathcal{D}}$的值，以评估对潜在候选设计点$\mathbf{x}$进行评估的整体效用，既考虑改善帕累托前沿近似，也考虑维持代理模型的全局精度 [49, 50]。值得注意的是，ParEGO算法利用高斯过程模型的不确定性信息，计算增广切比雪夫标量化中的期望改进 [28]，；而(q)EHVI算法则利用超体积指标中的期望改进（通过准随机近似计算） [51]。多目标贝叶斯优化能够产生全局收敛的方法，但缺点是收敛速度较慢，尤其是在高维设计空间中。因此，许多面向高维设计空间的方法会采用局部化策略，例如LTR方法，这与上述基于模型的无导数优化方法类似 [52]。

统计学文献中的另一种框架是多目标响应面方法（RSM）。RSM 从“搜索”步骤开始，该步骤会生成并评估大量试验设计 [53] 或拟随机样本[54]，以探索设计空间并收集用于代理建模的初始数据集。收集到足够的数据、确保代理建模精度达到要求后，典型的多目标 RSM 方法会采用多项式代理模型、多重标量化和优化过程，来求解所有标量化代理问题 [13, 第 7] 章。若干现有的多目标代理建模算法采用了这一方法的变体或多轮迭代，包括 VTMOP [32], SOCEMO [38], 和 PAWS [45]。在多目标设置下，多目标 RSM 可能较为有效，因为 RSM 的大部分计算成本来自初始搜索步骤，而该步骤的结果可供所有标量化共用。因此，通过应用多重标量化来求解多个目标，其增量成本并不比用 RSM 求解单个标量化高出多少。这种方法的缺点是，在高维空间中，可能无法生成足够大的试验设计来保证代理模型具有足够的全局精度。尽管如此，上述所有方法也都需要初始代理建模数据集，而该数据集通常通过与 RSM 类似的技术收集；例如，贝叶斯优化库 BoTorch [55]所采用的搜索技术。

实际上，上文讨论的各类算法之间的界限往往并不分明。此外，并非所有多目标仿真优化算法都会采用上文所述的全部方法。相比之下，ParMOO 侧重于采用以下技术：

- 用于在目标空间中设定目标的采集函数或类似技术；
- 用于根据采集函数生成下一个迭代点的优化求解器（或其单次迭代）；
- 用于近似高开销函数的代理模型，以及用于维持其准确性的不确定性函数或模型改进过程；以及
- 用于探索设计空间并生成代理建模初始样本的搜索技术。

在 ParMOO 发布之前，还没有开源软件能够实现遵循上述结构的通用 MOSO 求解器。 [15] 关注的是 ParMOO 的使用，而非其支持的底层算法、框架和技术。本文介绍了通过符合上述框架的任意算法求解 MOSO 问题所采用的设计策略和架构，并重点探讨如何将这类组合最有效地应用于特定领域应用。

2.1.1 多目标仿射可分优化中的其他技术。

如前所述　简而言之，并非所有多目标仿真优化算法都会采用第2节讨论的所有技术　.1.

与第2.1节的最大不同之一在于，并非所有多目标仿真优化算法都会使用可归入采集函数范畴的技术。不过，大多数算法都需要一种技术来选择用于迭代的点或点集。例如，MDS方法MODIR [56] 通过多目标识别过程，将单目标DIRECT算法推广到多目标情形。MDS方法DMulti-MADS [57], 、DMS [58], 和 MultiGLODS [59] 则结合使用指标函数和/或信赖域半径来选择每个迭代点。在多目标进化算法中，种群选择通常会采用基于支配关系的排序指标 [60, 第2.3章3]。著名的多目标进化算法NSGA-II [61]便采用了这种方法。尽管这些选择指标中有许多可以根据采集函数可用的信息（尤其是数据集 $\mathcal{D}$）计算得出，但这些方法与第2.1节讨论的技术存在显著差异。此外，值得注意的是，上述许多算法并不以代理建模为核心特征，这也是它们与第2.1节末尾介绍的技术存在的另一项重大差异。

尽管第2.1节所述的技术并未涵盖所有情况，但这一类技术为许多多目标仿真优化算法提供了核心要素。本文主要关注这类算法。

2.2 构建多目标仿真优化求解器时的设计挑战

现在我们来探讨 ParMOO 所要应对的关键设计挑战。ParMOO 旨在运用第2.1节介绍的技术，特别是采集函数/指标函数/标量化函数、代理建模和试验设计，解决种类繁多的科学多目标单目标优化应用。因此，我们重点关注与这些技术相关的挑战，以及如何在实际应用和科学计算环境中部署这些技术。

2.2.1 最先进的技术与领域专用技术。

首先，如第2.1节所述，最先进的代理驱动MOSO求解器需要整合多种不同技术，包括采集函数、优化求解器、代理模型和试验设计。若再纳入特定领域和问题的技术，这一挑战将更加复杂，例如带有物理信息约束的机器学习代理模型 [62] ，或旨在针对极罕见事件的采集函数 [63]。

在 ParMOO 中，我们力求支持大多数此类技术，包括我们目前可能尚不了解的特定领域方法。这需要为用户提供清晰的应用程序编程接口（应用程序接口），以便实现新的特定领域方法。进而，这要求对每种方法的应用程序接口进行标准化，并明确定义这些方法如何相互作用；要为第2.1节中的所有技术都做到这一点，可能并不容易。

2.2.2 结构化问题表述。

除了使用特定领域和特定问题的算法外，还可以更进一步，在 MOSO 问题表述中显式建模特定领域和特定问题的结构。这包括对复合问题定义进行建模，如 [64, 65, 66], 所述，其中目标函数可以表示为一个或多个函数的复合。具体而言，我们可以使用以下形式修改式（1）中的 MOOP问题定义：

$$\min_{\mathbf{x}\in\mathcal{X}} \mathbf{F}(\mathbf{x}, \mathbf{S}(\mathbf{x})) \tag{3}$$

其中 $\mathbf{F}$： $\mathcal{X} \times \mathbb{R}^m \to \mathbb{R}^o$ ，且 $\mathbf{S}$： $\mathcal{X} \to \mathbb{R}^m$。在此表述中，假设 $\mathbf{F}$ 是一个计算成本低廉的代数函数，且其梯度函数可能已知；而 $\mathbf{S}$ 是一个计算成本高昂的仿真函数，如前文所述。此表述将计算中的“仿真”部分与目标函数解耦，从而能够使用结构化非黑箱优化求解器。

$\mathbf{F}$ 的一个常见示例是平方和函数 $F_i(\mathbf{x},\mathbf{S}(\mathbf{x})) = \sum_{j\in J} S_j(\mathbf{x})^2$, , 它由一个或多个分量函数 F_i 和一个索引集 J构成。在单目标情形下，可以利用这种结构，通过受高斯-牛顿启发的方法实现超线性收敛速率 [67], ，无导数最小二乘求解器 POUNDERS [68]便是如此。尽管理论优势可能并不糟糕。

同样强大，显式建模式(3) 中的组合通常能以更少的仿真评估实现收敛，适用于各种其他代数**F**-函数 [64, 65, 66]。

F 多目标场景所特有的另一种结构是异构问题定义[43], ，其中一个或多个 F_i 是代数函数，不依赖于昂贵的仿真输出，但其他 F_i 要么是仿真输出的直接结果，要么是根据仿真输出计算得出。以往研究已提及这类问题定义 [32, 43], ，但我们尚未发现对此进行正式分析的文献。在基于代理模型的多目标仿真优化中，典型做法是对依赖仿真输出的部分应用代理模型，并在可用时改用代数形式。这种方法很适合式 (3) 中的抽象表示。

在 ParMOO 中，我们支持形式为（3）的问题定义，这与标准的黑箱优化方法略有不同。然而，需要注意的是，这种表述仍然涉及一个黑箱过程，因此无法仅通过基于梯度的方法求解。

2.2.3 设计空间的变化——混合变量和约束。

到目前为止，我们假设 $\mathcal{X}$ 是 $\mathbb{R}^n$ 的一个紧致且仅受简单界限约束的子区域。然而，广泛使用的多目标仿真优化软件通常支持范围广泛的设计变量和约束，包括实数（连续取值）、离散（整数取值）和/或类别型（非序数）变量的各种组合，以及线性和非线性约束 [55, 69, 70, 71, 6, 72]。

对于混合类型的设计变量，需要注意，改变定义域 $\mathcal{X}$ 可能会导致问题的分类完全不同，因此需要采用不同的技术。例如，当 $\mathcal{X}$ 仅由二元决策变量组成时，通常会使用多目标排序与选择算法 [73], ，这与第2.1节所述的连续型多目标仿真优化算法有显著差异。然而，许多以连续变量为主的 MOSO 问题也包含少量离散变量，这通常可以得到合理支持，而无需显著改变所用技术的类型。

由于我们关注的是为连续型基于代理模型的多目标仿真优化而设计的方法，因此主要挑战在于处理离散设计变量。对于多目标进化算法，可以通过为任何现有的多目标进化算法 [71]提供混合变量交配过程来支持这一点。对于多目标贝叶斯优化和许多多目标设计空间技术，定义一个支持整数变量和分类变量的距离函数、核函数或邻域函数或许就足够了[74]。然而，对于通用基于代理模型的多目标仿真优化算法，则需要一个函数 $E: x_{i'} \to [0,1]^\ell$ ，将离散设计变量 $x_{i'}$ 嵌入连续潜在空间 $[0,1]^\ell$，其中 ℓ 是嵌入的维数。潜在空间嵌入最常见的示例或许是独热编码，它将每个类别映射为 **0** 或 **1**，并且仅当相应的 $x_{i'}$表示对应类别时，才赋值为 **1**。然后，代理模型会对这个潜在空间建模，并可用于将嵌入后的设计变量松弛为连续表示。这种方法已广泛应用于现有的代理建模软件 [39, 75]。对于上述任何一种技术，当类别设计变量数量较多时，通用技术都难以扩展。因此，许多领域专用软件包和算法会采用领域专用设计空间嵌入器或距离函数 [6, 10]。以材料设计为例，贝叶斯优化求解器 EDBO [10] 使用分子描述符计算器 Mordred [76] 计算分子网络的三维潜在空间编码，然后在这个潜在空间中求解贝叶斯优化问题。

同样，引入线性约束和非线性约束后，我们需要放宽先前的假设，即 $\mathcal{X}$ 仅受边界限制。在实践中，这类约束可能是多种形式的组合：已知的（例如，问题定义的一部分）或隐藏的（例如，运行时发现的）、代数形式的或依赖于仿真输出的、可放宽的（即违反约束时仍可进行仿真），以及可量化的（即可以量化约束违反的程度） [77]。显然，我们无法支持上述所有组合，但会尽可能支持更多合理的组合。

根据式 (3) 中复合目标函数的定义，采用复合约束定义 $\mathbf{G(x,S(x))} \leqq \mathbf{0}$ 是自然而简洁的，其中“$\leqq$”表示逐分量小于或等于。基于这一点，我们将同时支持基于仿真的不等式约束和代数不等式约束，并立即排除隐式约束。这样，我们便可将可行域 $\mathcal{X}$ 分解为简单有界区域 $\mathcal{B}$ （包含 n 个维度，变量类型也可能混合），其中边界约束不可放宽；以及

可松弛的非线性 由 $\mathbf{G}$ 定义的非线性约束。这就得到了 ParMOO 的最终问题表述 n

$$\min_{\mathbf{x}\in\mathcal{B}} \mathbf{F}(\mathbf{x}, \mathbf{S}(\mathbf{x})) \text{ subject to } \mathbf{G}(\mathbf{x}, \mathbf{S}(\mathbf{x})) \leqq \mathbf{0}. \tag{4}$$

对于一般的非线性黑箱约束，单目标优化文献中已有多种处理非线性约束$\mathbf{G}$的方法，包括屏障函数 [78] 和增广拉格朗日方法 [79]。这两种方法都可以视为构造一个描述约束违反程度的惩罚函数，并将其加到目标函数中，以促使问题满足可行性要求。不过，实现方式有所不同。要将这些方法扩展到多目标情形，只需对所有目标函数应用累积惩罚函数 [80]。在这些方案中，屏障函数通常更灵活，无需对优化算法进行重大修改即可应用。在屏障方法中，常见的有两种：极端屏障，即每当违反约束时便施加无穷大的惩罚；以及渐进屏障，即根据约束违反程度施加惩罚，并随时间逐步增大该惩罚。

在 ParMOO 中，我们致力于支持数量有限的混合变量和可松弛非线性约束$\mathbf{G}$。我们通过嵌入和屏障函数来实现这一点，这只需对第2.1节所述技术进行有限修改。

2.2.4 并行计算与其他计算环境。

多目标仿射可分优化面临的另一项挑战，是如何在多种计算环境中有效利用资源。许多现有的优化求解器仅通过一种范式提供并行性（例如 Python 多进程、OpenMP 或 MPI） [71, 32, 72]。然而，在处理多种计算环境时，这可能并不足够；这些环境包括分布式系统、高性能计算集群、实验室集群，甚至湿实验室环境，而其中任意一种都可能不支持一种或多种常见的并行计算范式。

为应对这一复杂性，人们开发了许多平台 [81, 12] 和库/框架 [82, 83] ，用于协调并行评估、复杂仿真、数据存储以及异构资源的利用。此外，用于管理仿真、建模与优化之间相互作用的领域专用工具，也负责将任务分配到并行资源上 [84]。然而，如果优化软件在设计之初未充分考虑这些范式，那么有效利用这些平台并非易事[85, 86]。

在 ParMOO 中，我们在设计时充分考虑了并行性和仿真环境的灵活性，并确保将仿真评估与求解器和技术解耦。

2.2.5 可维护性与易用性。

近年来，科学计算领域普遍开始通过改进科学软件开发实践来减少技术债务、提高科学家的工作效率 [87]。许多科学软件库正在采用此类实践 [82, 46], ，而且近期已有若干现有的多目标优化库经过重构，以提高可维护性 [88, 69]。

设计 ParMOO 的一项关键目标，是构建一个软件包，提供用于实现前两节所讨论技术的应用程序接口，同时不牺牲可维护性和易用性。

2.2.6 设计目标。

总而言之， 本节列出的挑战和要求之上，我们现在列出 e

ParMOO 设计中的五个关键目标。

1. 我们希望提供一个高度可定制的软件框架，用于构建和部署基于代理模型的多目标仿真优化求解器。
2. 我们希望让优化求解器能够利用仿真输出在定义目标函数时所采用的任何可用结构。 3. 我们希望灵活支持各种问题类型，包括混合变量和约束。

4. 我们希望让我们的框架易于部署在各种各样的科学工作流中。5. 我们的软件框架和工作流必须易于使用、维护和扩展。

目标 1、2 和 3 要求我们仔细考虑支持哪些类型的多目标仿真优化算法、问题类型和结构，以及这些技术将如何相互作用。目标 4 和 5 属于工程问题，这限制了我们在实现目标 1、2 和 3 时能够合理支持的方法范围。尽管这些问题会限制我们对可用技术的选择，但 ParMOO 的工作流和总体项目结构（目标 5 的一些主要组成部分）此前已在[15]中进行了详尽讨论。因此，我们仅讨论目标 5 中会影响其他设计目标决策的部分。

2.3 现有多目标仿真优化软件

除了下文介绍的软件外，我们也承认还存在其他针对 MOSO 问题领域特定变体的专用求解器。不过，在本研究中，我们不考虑与特定应用绑定的求解器，例如神经架构搜索 [8] 和化学实验设计 [10], ；也不考虑针对问题（1）其他变体的求解器，包括在线多目标优化 [89]和多目标强化学习 [90]。我们也承认，基于代理模型的多目标仿真优化与主动学习 [91], 之间存在相似之处；我们认为，主动学习是一个比基于代理模型的多目标仿真优化更通用的问题。

本节仅列出用于解决 MOSO 问题的开源生产级软件包。这里的“生产级”是指软件实现足够成熟，非专业人士也能在非学术应用中使用。此类软件的标志包括具备详细的用户文档或 API文档、在软件期刊上发表，以及拥有庞大的用户群体等特征的组合。虽然这一点常常未加区分，但同样重要的是要明确，并非每种知名的 MOSO 算法都有与之对应的一一匹配的软件实现。

现有软件中很大一部分实现了多目标进化算法。广泛使用的多目标进化算法求解器包括 ParEGO[28], （它还集成了高斯过程代理建模）和 SPEA2 [92]。此外还有许多库，包括 Platypus [93], 、pymoo [71], 、PlatEMO [94], 、jMetal/jMetalPy [95, 69], 以及 pagmo/pygmo[70]。这些库大多实现了所有知名的多目标进化算法，例如 NSGA-II [61] 和 NSGA-III [25, 26]。虽然 DEAP [96] 框架并非专为多目标优化问题设计，但也常用于在 Python 中实现分布式进化算法，包括分布式多目标进化算法。

采用多目标直接搜索方法的软件包包括以下几种。DFO-lib [97] 包含 MODIR 和 DFMO 的串行 Fortran 实现，以及 MOIF 的 MATLAB 和 Python 实现。VTMOP [32], 提供串行和并行 Fortran 求解器，结合使用响应面方法、自适应权重和信赖域方法。NOMAD 是一个支持并行的 C++ 库，包含工业级的单目标和多目标直接搜索方法；直到最近，它仅包含双目标求解器 BiMADS [72]。在较新版本的 NOMAD v4 [88], 中，还新增了 DMulti-MADS 的实现。BoostDFO[98] 是一个 MATLAB 库，包含单目标和多目标直接搜索求解器的并行实现，包括 DMS 和 MultiGLODS。PyMOSO [99] 是一个主要面向整数值问题的 Python 框架。

近年来，出现了若干支持多目标贝叶斯优化的贝叶斯优化库和框架。Python库 Dragonfly [6], 旨在解决神经架构搜索问题，但也可用于通用的多目标贝叶斯优化问题。与本文目标最相关的或许是 BoTorch [55] 框架，它用于实现和部署并行、可用于生产环境的通用贝叶斯优化算法（包括多目标算法），并构建于自动微分框架 PyTorch[100]之上。

为总结本节，我们在表 1 中列出了迄今讨论过的求解器和库。该表展示了这些库与 ParMOO 设计目标的相关性。因此，我们重点关注这些库所提供的、与第 2.2.6 节设计目标 1–4 相关的功能。值得注意的是，表中未展示对领域特定和应用特定问题定义的支持（与目标 2 相关），因为在所列方法中，目前只有 BoTorch 和 ParMOO 支持这种问题形式。我们承认，这些软件包中的许多仍在积极开发中，未来可能会添加新功能，因此我们的分类依据是本文发表时有文档记录的功能。

Name	Type	语言	方法	约束	变量类型	代理模型
NOMAD v4	L	C++	MDS	yes	混合	yes
BoostDFO	L	MATLAB	MDS	some	real	yes
BoTorch	L	Python	MBO	yes	混合	yes
DEAP	L	Python	MOEA	yes	混合	no
DESDEO	L	Python	any	yes	real	yes
Dragonfly	L	Python	MBO	yes	混合的	yes
jMetal/jMetalPy	L	Java/Python	MOEA	yes	混合	no
DFO-lib	L	Fortran/Python/Matlab	MDS	some	实数或整数	no
ParEGO	S	C	多目标进化算法/多目标贝叶斯优化	no	real	yes
pagmo	L	C++	MOEA	some	混合	no
ParMOO	L	Python	MDS/MBO	yes	混合	yes
PlatEMO	L	MATLAB	MOEA	some	混合	some
Platypus	L	Python	MOEA	yes	混合	no
pygmo	L	Python	MOEA	some	混合	no
pymoo	L	Python	MOEA	some	混合	no
PyMOSO	L	Python	MDS	yes	int	no
SPEA2	S	C	MOEA	no	real	no
VTMOP	S	Fortran	MDS	no	real	yes

表 1：通用开源多目标仿真优化软件。对于每个软件包，各列的含义如下："类型"表示这是单个求解器（S）还是求解器库（L）；"语言"说明主要的开发语言；"方法"将每个软件主要采用的方法归类为多目标进化算法（MOEA）、MDS 或多目标贝叶斯优化（MBO）；"约束"表示是否支持非线性约束；"变量类型"表示支持的变量类型；"代理模型"表示是否使用代理建模。

3 多目标仿真优化框架

现在，我们概述用于求解 MOSO 问题的框架。具体而言，我们将说明该框架如何应对第 2.2.6 节中提出的设计目标和挑战。该框架在软件库 ParMOO 中实现 [15], ，可按照其在线文档 [101]中的安装说明获取。

3.1 面向对象的模块化设计

尽管实现目标 1 需要相当高的复杂度，但这项工作的最大挑战是在管理这些相互作用技术的复杂性的同时，保持应用程序接口易用、易维护且可扩展（目标 5）。为此，我们采用模块化设计，让每个模块通过通用接口处理更大问题中的特定组成部分。面向对象编程（OOP）是实现此类框架的自然范式。为此，我们为各个求解器组件实现了抽象基类（ABCs）。回顾第 2.1 节，ParMOO 的四个主要抽象基类是

- GlobalSearch 类，它抽象了用于试验设计及其他采样技术的应用程序接口；
- SurrogateFunction 类，该类对用于拟合、更新和评估通用代理函数的应用程序接口进行了抽象，并根据具体情况，提供评估不确定性函数或推荐模型改进点的功能；
- AcquisitionFunction 类，它抽象了用于实现采集函数（包括标量化和指标函数）以及选择下一个迭代点（如适用）的应用程序接口；以及
- SurrogateOptimizer 类接受一个标量化的可微或不可微函数，并使用适当的方法生成下一个迭代点，或（如适用）调用模型进行改进。

为了让用户无需自行实现这些技术中的每一种就能使用 ParMOO，我们提供了一个内置库，其中包含各类别中若干标准技术的实现。

这一切的核心是 MOOP 类，它存储着一个可能较为复杂、形式如式（4）所示的 MOSO 问题定义，并协调各抽象基类的实现以求解 MOSO 问题。这是通过“建造者”设计模式实现的 [102]。MOOP 类本身也具有高度模块化和可扩展性，其用于执行私有和公共任务的方法均有清晰的文档说明。开发者可以轻松覆盖某些功能，例如分配高成本仿真评估的方法，以适应新颖的架构和用例。由此，模块化面向对象框架也实现了目标 5。

这种方法的缺点是，面向对象的应用程序接口增加了额外的复杂度，类似于学习一种新的建模语言，例如 JuMP [103] 或 Pyomo [104]。用户还需要了解基于代理模型的多目标求解器的基本组成部分，才能正确使用我们的组件库和框架。归根结底，这种学习负担远高于正确使用其他 Python 多目标求解器库（如 pymoo[71] 和 pagmo/pygmo [70]）所需的负担。然而，我们认为，为实现第 2.2.6 节所述的目标 1–3，这种方法是必要的折中。

因此，ParMOO 与表 1 中列出的许多库和单独的求解器并不可直接比较。在这些软件包中，ParMOO 也许最接近 DEAP [96]和 BoTorch [55], ，它们实现了类似的框架，采用不同类别的技术来求解略有不同的问题类别。

3.2 区分仿真、目标和约束 ts

我们需要直接应对的下一个挑战是目标 2：对于问题 (3) 和 (4) 中定义的问题，需要让优化求解器能够访问仿真输出和函数评估。这里的关键思路是，将仿真输出与目标值和约束惩罚分开管理和建模。图 1 展示了 ParMOO 如何处理多目标优化问题：黑箱仿真函数 $\mathbf{S}$先映射到中间的“仿真输出空间” $\mathcal{S}$，然后 $\mathbf{F}$ 再将其映射到可行目标空间。

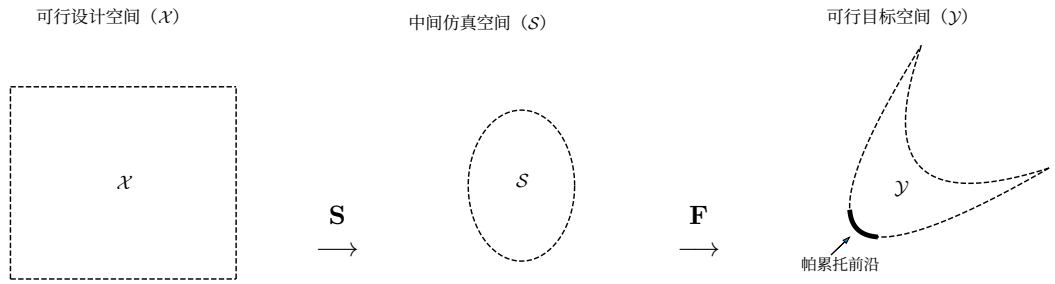


图 1：可行设计空间 $\mathcal{X}$ （左）通过黑箱仿真 $\mathbf{S}$映射到中间的仿真输出空间 $\mathcal{S}$（中），再通过代数目标函数 $\mathbf{F}$映射到可行目标空间 $\mathcal{Y}$ （右）。

图 1 所示的抽象使我们能够处理式 (3) 和 (4) 中的问题表述。我们不再使用由设计点与目标值组成的数据集 $\mathcal{D}$ 对目标进行代理建模，也就是拟合 $\mathbf{F}_{\mathcal{D}}(\mathbf{x}_i) \approx \mathbf{F}(\mathbf{x}_i)$；而是使用由设计点与仿真输出 组成的数据集 $\mathcal{D}$ ，为中间仿真空间 $\mathbf{S}_{\mathcal{D}}(\mathbf{x}_i) \approx \mathbf{S}(\mathbf{x}_i)$建立代理模型。接着，与式 (2) 类似，我们使用求解器近似最小化代理问题的采集函数或标量化形式。

$$\min_{\mathbf{x}\in\mathcal{B}} A_{\mathcal{D}}(\mathbf{F}(\mathbf{x}, \hat{\mathbf{S}}_{\mathcal{D}}(\mathbf{x})), \mathbf{\Sigma}_{\hat{\mathbf{S}},\mathcal{D}}(\mathbf{x})) \text{ subject to } \mathbf{G}(\mathbf{x}, \hat{\mathbf{S}}_{\mathcal{D}}(\mathbf{x})) \leqq \mathbf{0} \tag{5}$$

ˆ 其中，$\mathbf{\Sigma}_{\hat{\mathbf{S}},\mathcal{D}}$ 是给定 $\mathcal{D}$ 时$\mathbf{S}_{\mathcal{D}}$ 的不确定性函数。

在每次迭代中，SurrogateOptimizer 对象都可以访问标量化输出 $A_{\mathcal{D}}(\mathbf{F}(\mathbf{x},\ \hat{}\mathbf{S}_{\mathcal{D}}(\mathbf{x}))^{\mathbf{\Sigma}_{\hat{\mathbf{S}},\ \mathcal{D}}(\mathbf{x}))}$、ˆ 目标函数 $\mathbf{F}$、仿真代理模型 $\mathbf{S}_{\mathcal{D}}$ 以及约束违反函数 $\mathbf{G}$。这些信息足以实现各种充分利用复合结构的优化求解器。

3.3 通过问题嵌入管理复杂性

为实现目标3，我们需要处理涉及连续变量、整数值变量和分类变量以及非线性约束的问题，同时不要求使用任何特定于问题的技术，也不对第3.1节所述的结构进行大幅修改。为保持可用性和可维护性（目标4），我们必须谨慎选择所采用的技术。

如第2.2.3节所述，为了处理混合变量，我们选择将问题嵌入连续潜在空间，然后求解该问题的松弛形式。ParMOO 通过创建一个隐藏的嵌入层，将这一思想抽象化；该嵌入层基于内置或用户提供的嵌入器/提取函数 $\mathbf{E}_{\text{in}}$ 和$\mathbf{E}_{\text{out}}$，满足$\mathbf{E}_{\text{in}}(\mathbf{x}) : \mathcal{X} \to [0,1]^\ell$ 且$\mathbf{E}_{\text{out}}(\mathbf{E}_{\text{in}}(\mathbf{x})) = \mathbf{x}$；参见图 2。输入时，所有设计变量都会经过各自的嵌入器 $\mathbf{E}_{\text{in}}$，任何取离散值的变量都会通过代理模型进行松弛。ParMOO 求解代理优化问题后，所有候选设计点都会通过$\mathbf{E}_{\text{out}}$ 从潜在空间提取回原始设计空间，并在必要时归入最接近的合法值。
ParMOO 为分类变量提供默认嵌入方式，该方式采用独热编码，并随后执行降维过程，以消除潜在空间中未使用的维度（即，表示同一分类变量的两个不同潜在变量不能同时取非零值）。但是，这种技术无法扩展到大量分类变量的情况。因此，对于包含大量分类变量的问题，ParMOO 还允许用户提供自定义嵌入过程。对于连续和整数设计变量，默认嵌入基于简单的缩放，将其映射到范围 $[0,1]$。

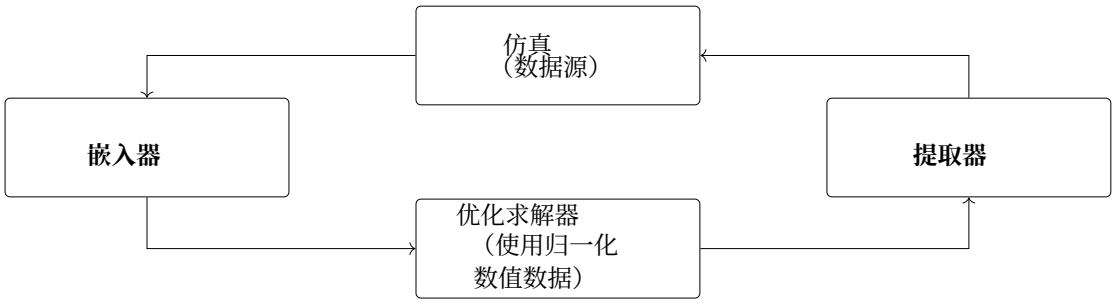


图 2：ParMOO 中数据流的示意图。嵌入器模块接收设计空间中可能为离散的取值，并将其映射到归一化的潜在空间 $[0,1]^\ell$。提取器模块接收来自优化模块、范围为 $[0,1]^\ell$ 的候选设计，并将其映射回 $\mathcal{X}$。

这种方法的主要优势在于，它使应用程序接口保持简洁，并且无论问题类型如何，我们都可以专注于连续优化技术。具体而言，求解混合变量问题时，无需使用定制搜索、代理建模或优化技术（尽管这样做可能会提升性能），因为默认技术旨在对重新缩放后的潜在空间进行操作。接受自定义嵌入工具的灵活性，对于分子工程等领域也很有用，因为这些领域已有可用的嵌入工具 [76]。如前所述，对于包含大量分类变量、但没有任何领域专用嵌入工具的问题，默认方法可能会变得低效，此时采用一种完全不同的、不基于连续型多目标仿真优化算法的方法或许更有优势。

为了纳入一个 p维非线性约束函数 $\mathbf{G}$，我们对代理问题应用累积惩罚函数 $P(\mathbf{x}) = \lambda \sum_{i=1}^{p} \max(G_i(\mathbf{x}, \mathbf{S}(\mathbf{x})), 0)$ 或 $P(\mathbf{x}) = \lambda \sum_{i=1}^{p} \max(G_i(\mathbf{x}, \hat{\mathbf{S}}_{\mathcal{D}}(\mathbf{x})), 0)$。在求解 (5) 时，该惩罚项会添加到所有目标函数中。结合上述所述的嵌入器/提取函数，我们可以将非线性约束、部分离散的可行设计空间 $\mathcal{X}$ 缩减为一个边界约束、条件良好且连续的 MOSO 问题

$$\min_{\mathbf{x} \in [0,1]^\ell} A_{\mathcal{D}}(\mathbf{F}(\mathbf{E}_{\text{out}}(\mathbf{x}), \hat{\mathbf{S}}_{\mathcal{D}}(\mathbf{x})) + \lambda \sum \max(\mathbf{G}(\mathbf{E}_{\text{out}}(\mathbf{x}), \hat{\mathbf{S}}_{\mathcal{D}}(\mathbf{x})), \mathbf{0}), \mathbf{\Sigma}_{\hat{\mathbf{S}}, \mathcal{D}}(\mathbf{x})), \tag{6}$$

其中，最大值按元素计算。当预计式（6）的解不可行时，惩罚参数 $\lambda > 0$ 会按照指数计划逐步增大。这与 [78]，中描述的渐进屏障方法类似，并已成功应用于基于仿真的约束问题[5]。研究表明，当约束难以满足时，这种方法是有效的 [105]。

这种方法的一个缺点是，惩罚函数使用约束违反量，这就隐含要求$\mathbf{G}$ 中的约束函数既可松弛又可量化。因此，在 ParMOO 中，每个设计变量的上界/下界约束是唯一可以不可松弛的约束，这是在约束繁多的问题上实现合理性能所必须做出的妥协。

3.4 对新型工作流的灵活性和可扩展性

目标 4 是使我们的框架足够灵活，以便轻松部署到各种科学工作流中。关键示例包括高性能计算（HPC）和湿实验室环境。

为实现这一目标，我们设计 ParMOO，使其仅通过包装函数 MOOP.evaluateSimulation() 发出仿真评估，而该函数仅在求解器例程 MOOP.solve() 内部调用。在只需更改仿真命令以便与现有工作流技术集成的情况下，可以通过扩展 MOOP 类并重写 MOOP.evaluateSimulation() 方法来实现。在其他情况下，如果需要控制仿真评估的分发频率（例如，对仿真评估进行批处理），则可以重写整个 MOOP.solve() 方法。ParMOO 的 libE MOOP 类采用了后一种方法；该类扩展 MOOP 类，并重写 solve 方法，以使用 libEnsemble 库 [82, 83] 在 HPC 系统上动态分发仿真评估。当前，libE MOOP 类是使用 ParMOO 实现可扩展并行性的推荐方法。

3.5 最终框架

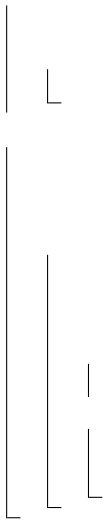
综合以上内容，我们介绍在 ParMOO 中实现的求解 MOSO 问题的框架。在每次迭代中，ParMOO 都会使用算法 1生成下一批候选项（通过 MOOP.iterate() 方法实现）。

要填充算法 1 的各个组成部分，用户必须创建并完整填充一个 MOOP 对象。初始化 MOOP 时，用户必须指定一个 SurrogateOptimizer，用于提出求解或迭代 (6) 的候选项，并提供一个超参数字典。接下来，用户必须添加问题详情，包括

- n 设计变量字典，每个字典都为问题指定一个设计变量，并在适用时提供自定义的嵌入器和提取器例程；
- s仿真字典，每个字典都定义问题的仿真/数据源，以及中间输出的数量（m_i）、用于对该仿真的输出空间进行采样的全局搜索技术，以及用于对该仿真输出建模并评估不确定性或执行模型改进迭代的代理函数；
- o 个目标函数字典，每个字典都指定了一个代数函数，可用于根据设计变量和仿真输出计算目标值，并可选择性地计算该目标函数的梯度；
- p 约束字典，与目标函数字典相同，但用于评估可松弛约束的违反程度（即 $\mathbf{G}$ 函数）；以及
- q 采集函数字典，每个字典都指定一个可用于将问题标量化的 AcquisitionFunction，其中 q 还决定了并行评估的批大小。

图 3 展示了概述该框架的统一建模语言图。请注意，图中仅显示了本节讨论的相关公共方法。在 MOOP 类的实现中，还包含若干其他公共和私有辅助方法，以及用于保存、加载、记录日志和创建检查点 [101] 的其他“设置器”方法。

算法 1：ParMOO 的 MOOP.iterate() 方法的单次迭代输入： $k \geq 0$ 是当前迭代点。输入： $q_0 \geq 0$ 是初始搜索预算。输入： q 是批大小，即 AcquisitionFunctions 的数量。输入： n 是设计空间的维度，ℓ 是潜在空间的维度。输入：$\mathrm{E}_{\mathrm{in}}: \mathcal{X} \to [0,1]^\ell$ 是所有设计变量的累积嵌入函数。输入：$\mathrm{E}_{\mathrm{out}}: [0,1]^\ell \to \mathcal{X}$ 是提取函数（E_{in} 的逆函数）。输入：$\mathrm{S}: \mathcal{X} \to \mathbb{R}^m$ 是仿真函数。输入：$\mathrm{F}: \mathcal{X} \times \mathbb{R}^m \to \mathbb{R}^o$ 是目标函数。输入：$\mathrm{G}: \mathcal{X} \times \mathbb{R}^m \to \mathbb{R}^p$ 是约束函数。输入： $\mathcal{D}^{(k)}$ 是第 k 次迭代之前评估的所有 (x, S(x)) 对的集合。输入：GlobalSearch.search(q_0, ℓ) 是一个过程，用于在 $[0,1]^\ell$ 中生成一个试验设计或大小为 q_0 的样本。输入：SurrogateFunction.fit($\mathcal{D}$) 用于拟合代理模型 $\hat{\mathrm{S}}_{\mathcal{D}}: [0,1]^\ell \to \mathbb{R}^m$，以及（如适用）其不确定性函数 $\boldsymbol{\Sigma}_{\hat{\mathrm{s}},\mathcal{D}}: [0,1]^\ell \to \mathbb{R}^m$。输入：SurrogateFunction.improve($\mathcal{T}$) 为 $\hat{\mathrm{S}}_{\mathcal{D}}$ 在 $\mathcal{T} \subset [0,1]^\ell$ 中生成一个模型改进步骤（如适用）。输入： $\{A_{\mathcal{D}}^{(i)}\}_{i=1}^q$ 是一组 q AcquisitionFunctions。输入：SurrogateOptimizer.solve($A_{\mathcal{D}}$, F, G, $\hat{\mathrm{S}}_{\mathcal{D}}$, $\boldsymbol{\Sigma}_{\hat{\mathrm{s}},\mathcal{D}}$, $\mathrm{x}^{(0)}$)) 求解 (6)，从 $\mathrm{x}^{(0)}$ 开始迭代逼近 (6) 的解，或调用 SurrogateFunction.improve($\mathcal{T}$) 方法并提供一个 LTR $\mathcal{T}$。输出：第 k 次迭代后的候选设计点批次 $\mathcal{C}^{(k)}$。**1** $\mathcal{C}^{(k)} \leftarrow \emptyset$； 2 如果 $k = 0$ 则**3** $\mathcal{C}' \leftarrow$GlobalSearch.search(q_0, ℓ)； 4 对每个 $\mathrm{y} \in \mathcal{C}'$ 执行 **5** $\mathcal{C}^{(0)} \leftarrow \mathcal{C}^{(0)} \cup \{\mathrm{E}_{\mathrm{out}}(\mathrm{y})\}$； 6 否则 **7** $\mathcal{D}' \leftarrow \{(\mathrm{E}_{\mathrm{in}}(\mathrm{x}), \mathrm{S}(\mathrm{x})) : \text{对于所有 } (\mathrm{x}, \mathrm{S}(\mathrm{x})) \in \mathcal{D}^{(k)}\}$； 8 $\hat{\mathrm{S}}_{\mathcal{D}}, \boldsymbol{\Sigma}_{\hat{\mathrm{s}},\mathcal{D}} \leftarrow$ SurrogateFunction.fit($\mathcal{D}'$)； 9 对 $i \leftarrow 1$ 至 q 执行 **10** $\mathrm{x}^{(0,i)} \leftarrow \arg\min_{(\mathbf{x},\mathbf{S}(\mathbf{x}))\in\mathcal{D}'} A_{\mathcal{D}'}(\mathrm{F}(\mathrm{x}, \mathrm{S}(\mathrm{x})), 0)$)； 11 $\mathrm{y}^{(k,i)} \leftarrow$ SurrogateOptimizer.solve($A_{\mathcal{D}'}$, $\mathrm{F} \circ \mathrm{E}_{\mathrm{in}}$, $\mathrm{G} \circ \mathrm{E}_{\mathrm{in}}$, $\hat{\mathrm{S}}_{\mathcal{D}}$, $\boldsymbol{\Sigma}_{\hat{\mathrm{s}},\mathcal{D}}$, $\mathrm{x}^{(0,i)}$)； 12 如果 $\mathrm{y}^{(k,i)} \notin \mathcal{C}^{(k)}$ 则**13** $\mathcal{C}^{(k)} \leftarrow \mathcal{C}^{(k)} \cup \{\mathrm{E}_{\mathrm{out}}(\mathrm{y}^{(k,i)})\}$； 14 否则 15 $\mathrm{y}' \leftarrow$ SurrogateFunction.improve($[0,1]^\ell$)；**16** $\mathcal{C}^{(k)} \leftarrow \mathcal{C}^{(k)} \cup \{\mathrm{E}_{\mathrm{out}}(\mathrm{y}')\}$； 17 返回 $\mathcal{C}^{(k)}$。



3.6 框架的局限性

我们力求涵盖大量多目标仿真优化算法和技术，并使其能够轻松应用于各种应用场景。尽管我们已成功覆盖范围广泛的算法，但我们的框架仍有其局限性。对于第2.1.1节中讨论的、不使用采集函数或代理模型的多目标仿真优化算法，将其实现于 ParMOO 的框架中，并强行将其迭代过程纳入算法 1，可能效率不高。此外，正如第2.2.3节所承认的，ParMOO 的框架从根本上基于针对连续、有界约束的多目标仿真优化的技术，并通过将混合变量和非线性约束问题转化为有界约束问题来处理，

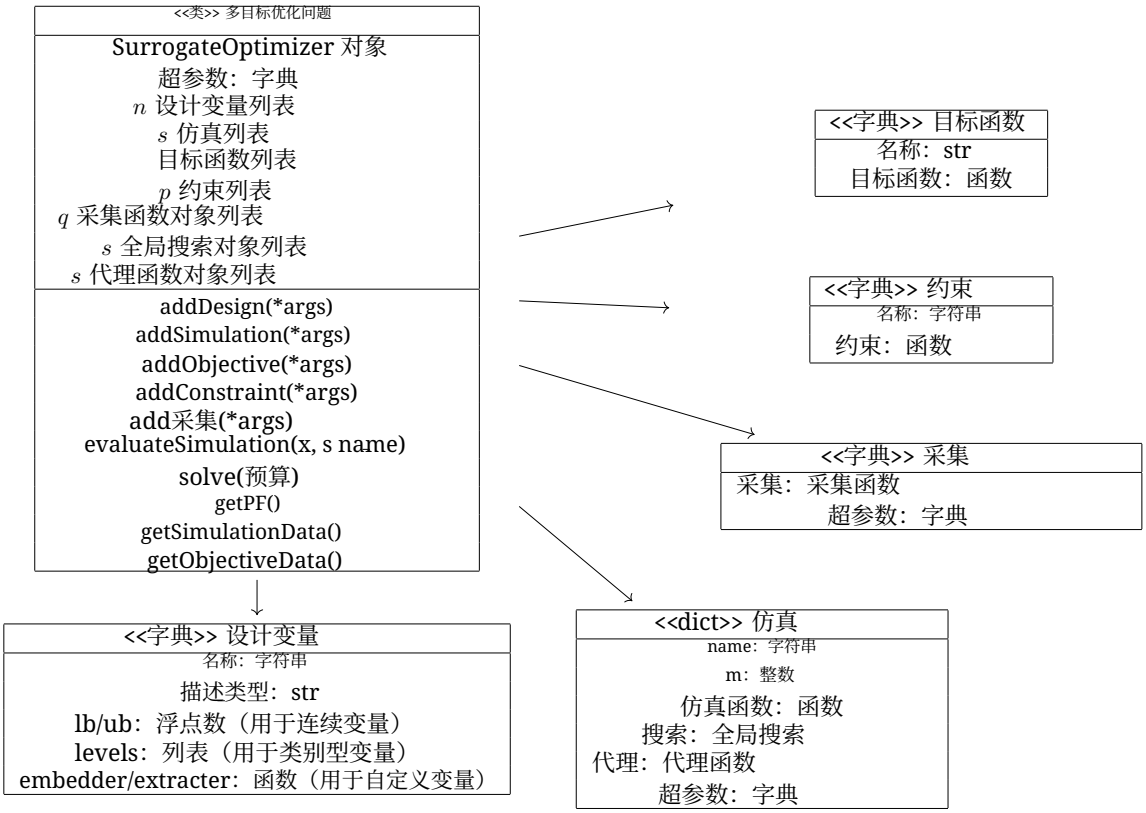


图3：统一建模语言图，概述了构成一个多目标优化问题对象及其内容的关键字典、组件和方法。

连续多目标单目标优化问题。对于不含连续变量或不可松弛约束的问题，这可能不是最有效的方法，其他框架和算法类别可能更为适用。

尽管多目标仿射可分优化通常是一项无导数的工作，我们仍努力提供有限的偏导数信息支持，因为代数函数$\mathbf{F}$ 和$\mathbf{G}$ 可能是可微的，而且在有梯度信息时，ParMOO 的优化求解器可以利用这些信息。不过，这可能会造成误导，因为 ParMOO 的设计目标是求解至少涉及一些计算成本高昂的黑箱仿真的问题。如果问题定义中不包含黑箱仿真函数，并且目标函数和约束定义的所有组成部分都有梯度，那么此处未涉及的其他方法可能会更有效。

最后值得指出的是，ParMOO 本质上是一个框架，而非算法。因此，利用问题结构并实现高质量算法的许多工作都留给了用户。尽管 ParMOO 的框架和开发已相当成熟，但可用技术种类仍在积极发展中，目前尚未提供第2.1节所列的全部技术。要在利用大多数问题结构时获得理论优势，需要仔细分析并设计巧妙的算法；不过，我们将在第5节和第6节展示，即使只投入相对少量的工作，ParMOO 在大多数应用中仍能获得一些明显优势。

4 性能摘要

在本节中，我们将 ParMOO中构建的信赖域求解器应用于文献中的一个常见学术测试问题，以评估其并行扩展性。在此之前，我们想提醒读者注意本研究的局限性。首先，如第3节所述，ParMOO 是一个框架和软件库，用于

实现求解器，而非其本身的一种算法；ParMOO 的性能可能因所用方法不同而有显著差异。其次，本文和 ParMOO 的设计重点是实现其在真实多目标单目标优化应用中的性能。由于此处使用的学术问题未必能反映真实世界中的多目标单目标优化应用，因此，对该问题的性能应理解为验证 ParMOO 在可求解问题上能够取得合理的性能，而非表明本文所述的特定 ParMOO 求解器比现有方法“更好”或“更差”。因此，我们特意从多目标优化文献中选择了一个相对简单的问题，即广泛使用的 DTLZ 测试套件 [106] 中的 DTLZ2问题。我们采用该问题的一个变体，其中包含 3 个目标函数和 10 个设计变量。该问题没有约束、没有混合变量，也没有可供利用的图 1 所示形式的复合结构。这个简单问题使我们能够在通用条件下，专注于实现的正确性和并行扩展性。

由于整个 DTLZ2问题都是代数形式的，我们必须创建一个人工仿真。因此，我们为 DTLZ2 函数创建了一个包装器，通过人为延长运行时间，将其转化为黑盒仿真。具体做法是从一到三秒之间均匀采样一个运行时长，然后使用 Python 内置的 time.sleep() 函数等待相应的时间。目标（**F**函数）随后被设为从该人工仿真的三个输出字段映射而来的恒等映射。

作为示例求解器，我们通过实例化 libE MOOP构建器类，为该问题创建了一个并行 ParMOO 求解器。该求解器采用 SurrogateOptimizer 的 TR LBFGSB 实现，使用 L-BFGS-B 求解信赖域约束代理问题；采集函数则采用随机化ε约束标量化 RandomConstraint 与固定权重加权和标量化 FixedWeight 的组合；此外还使用 GaussRBF（高斯径向基函数）代理函数以及拉丁超立方全局搜索。通过改变 AcquisitionFunction 对象的数量，我们能够创建该 ParMOO 求解器的不同变体，使其每次迭代生成批大小为 8、16 和 32 的候选仿真。由于 ParMOO 会并行执行仿真评估，因此该批大小决定了单次迭代中可用的最大并行性。

由于拉丁超立方抽样和默认的模型改进流程都带有随机性，我们使用不同的随机种子，对上述定义的 ParMOO 求解器进行了 5 次运行。对于每个随机种子和每种批大小，我们分别采用串行方式，以及在仿真评估期间采用 2、4 和 8 路并行性进行求解，总预算为 1,000 次仿真评估。

为验证结果，我们在每次运行结束时计算了广泛使用的超体积性能指标 [31], ，对这些数值取平均，并比较各次运行的结果。对于超体积指标，数值越大越好；其最大值为帕累托前沿与参考点之间的总体积（本例中参考点为 (1，1，1)）。超体积数值向来难以解读，因此，为便于比较，我们还使用 pymoo 中广泛采用的 NSGA-II 实现，在相同预算下求解了 DTLZ2[61, 71]。

比较结果见表 2。即使未利用任何结构，这些结果也表明，在进行 1,000 次仿真评估后，ParMOO 求解器在 DTLZ2 上的表现（依据超体积指标衡量）始终优于 pymoo，说明 ParMOO 在学术基准问题上至少能达到合理的性能。不同批大小的 ParMOO 求解器性能略有差异；不过，我们不建议基于如此有限的数据得出结论，尤其是因为 DTLZ2 并非特别能代表 ParMOO 的应用场景。

方法	pymoo	ParMOO-8	ParMOO-16	ParMOO-32
超体积	0.28	0.33	0.33	0.37

表 2：pymoo 和 ParMOO 在批次大小为 8、16、32 时的超体积指标。数值越大越好。

ParMOO 在并行运行时不会改变算法 1，仅将批量中的仿真评估并行化。因此，除了由于随机数生成导致的轻微且不相关的波动外，当增加线程数时，我们没有观察到其收敛性（以超体积指标衡量）发生任何变化。因此，如果我们能够使用上述任一求解器缩短执行 1,000 次仿真的运行时间，这将体现出明显的并行优势。在图 4 中，我们考察了线程数增加时运行每种 ParMOO 批大小所需的总运行时间。

请注意，由于平均仿真时间为 2 秒，因此串行执行总计 1,000 次仿真评估时，预计总运行时间约为 2,000 秒，此外还需加上 ParMOO 产生的任何迭代开销。

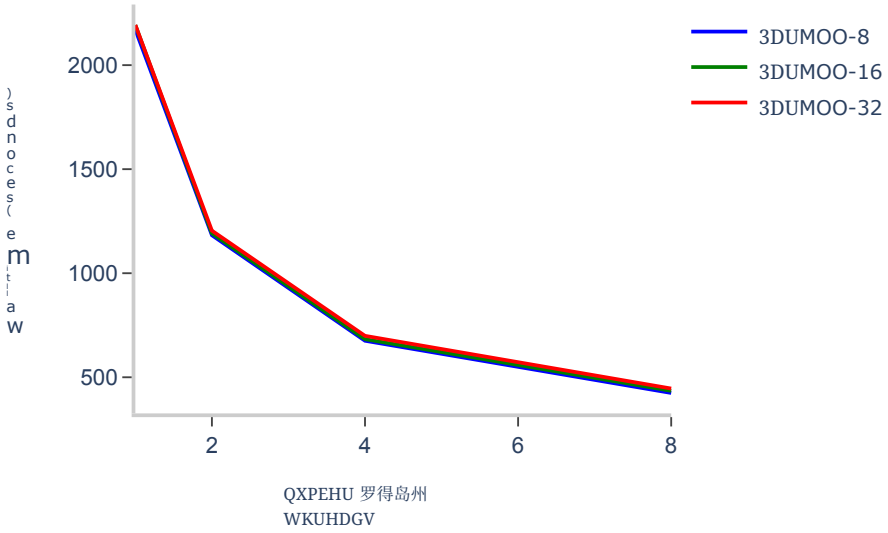


图 4：在 ParMOO 中执行 1,000 次耗时 1–3 秒的仿真评估时，随着线程数量增加，不同批次大小（8、16 和 32）对应的运行时间。

如图 4 所示，对于此求解器配置和问题规模，ParMOO 迭代计算会产生少量开销（请注意，单线程运行时间略超过 2,000 秒），但解的运行时间会随着线程数增加而成比例缩短。这种并行性略逊于理想的强扩展性，并受候选批次大小限制。尽管如此，对于大多数合理的求解器配置，在并行执行 8 次仿真且每次仿真平均耗时 2 秒（± 1 秒变化）时，预计可以在略少于 500 秒内使用 ParMOO 完成 1,000 次仿真。

5 案例研究：求解多目标逆问题

本节将 ParMOO 应用于一个多目标逆问题，其目标是根据实验数据调整 Fayans 能量密度泛函（EDF）模型的参数。该问题的完整描述见 [3]。前向模型的计算成本很高，且未公开。因此，为了确保结果可复现，我们针对上述数据集上训练的神经网络模型，优化一个合成问题。

5.1 Fayans 能量密度泛函校准背景

现在我们回顾 Fayans 能量密度泛函校准的重要方面。令 $T_{\mathbf{x}}:\ s \to t$ 表示 Fayans 能量密度泛函模型。对 $T_{\mathbf{x}}$ 进行一次前向计算时，输入归一化参数向量 $\mathbf{x} \in \mathbb{R}^{13}$ 和输入 s，并生成物理观测值 t。给定 198 对观测数据 $(s_1,\ t_1)\ \ldots,\ (s_{198},\ t_{198})$，Fayans 能量密度泛函模型校准的目标是找到“良好”的参数向量 $\mathbf{x}^\star \in \mathbb{R}^{13}$，使得相对于给定的标准误差 $\sigma_i > 0$，$T_{\mathbf{x}^\star}(s_i) \approx t_i$，其中 $i = 1,\ \ldots,\ 198$。

在传统的 Fayans 能量密度泛函校准中，假设所有观测值均已通过标准误差 $\boldsymbol{\sigma}$ 进行归一化，从而具有相等且相互独立的归一化误差。因此，通过单目标公式最小化全部 198 个观测值的 χ^2 损失是合理的。

$$\min_{\mathbf{x}\in\mathbb{R}^{13}} \sum_{i=1}^{198} \left(\frac{T_{\mathbf{x}}(s_i) - t_i}{\sigma_i} \right)^2. \tag{7}$$

在 [3], 中，然而，值得注意的是， χ^2 损失的各项贡献可能因 9 种不同的观测类型而异。由于求解九目标公式的计算成本过高，我们考虑采用一种简化的三目标公式，将相关的观测类型合并，从而仅保留三个观测类别。这样，我们便可以定义以下 Fayans 能量密度泛函校准的多目标公式：

$$\min_{\mathbf{x}\in\mathbb{R}^{13}} F_{\mathbf{x},j}, \qquad j=1,2,3, \tag{8}$$

其中，$F_{\mathbf{x},\ j}=\sum_{i\in\Phi_j}\left(\frac{T_{\mathbf{x}}\ (s_i\ -t_i}{\sigma_i}\right)^2)$ 以及 Φ_1， Φ_2， Φ_3 是基于上述三个观测类别对 $\{1,\ldots,198\}$ 进行的划分。

在本节中，我们求解 Fayans 能量密度泛函校准问题（8）的多目标公式。在该问题中，对全部 198 个观测值 $[T_{\mathbf{x}}(s_1)\ldots,T_{\mathbf{x}}(s_{198})]$ 的前向模型进行评估，被视为一个计算代价高昂的黑箱。三个损失函数 $F_{\mathbf{x},j}$ ($j=1$,2,3) 均具有复合结构，具体而言，是平方和结构。

5.2 神经网络残差模型

对于我们的实验，由于 $T_{\mathbf{x}}$ 成本高昂，而且我们力求结果可复现，因此我们拟合了一个多层感知机（MLP），用来近似标准化残差函数 $R_i(\mathbf{x})\ =\ ^{T_{\mathbf{x}}}{}_{(s_i\ -t_i)}$, $i\ =\ 1,\ldots,198$。我们使用一个包含残差函数 $R_1(\mathbf{x})\ldots,R_{198}(\mathbf{x})$ 带标签观测值的数据集来完成这一工作，这些观测值对应于 $\mathbf{x}$ 的 52,079 个不同取值。该数据集由 [3] 收集；正如该文所述，数据集来源于对问题（7）的单目标公式运行多个初始设计点，并使用若干种不同的单目标求解器。因此，我们指出，该数据集并非均匀分布，在单目标问题（7）的若干局部解邻域内密度较高。

本节所用多层感知机的训练面临的主要挑战，是确保多目标问题 (8) 的最优值具有较高的预测准确度。具体而言，单目标公式中有大片参数空间被确定为不稳定区域，因此没有可用的观测数据。此外，在稳定区域内，对于若干个 i 值，$R_i(\mathbf{x})$ 的观测数据量级可能从小于 1 到超过 10^{20} 不等。对于这些较大的残差，量级大于 1 的预测误差是可以接受（且在预料之中）的；但为忠实于原始问题，我们必须确保在真实解附近，预测误差保持在 10^{-2} 量级。

为处理没有可用观测数据的参数空间不可行区域，我们依据 [3], 在表3中给出的约束，为 ParMOO 施加边界约束。为应对观测值范围较大的问题，我们先对 $R_i(\mathbf{x})$ 的较大值应用双对数变换，再应用双曲正切变换。由于这种变换会压缩极大的误差值，因此在训练期间，它能确保多层感知机更加重视匹配残差函数 $R_i(\mathbf{x})$ 得分较低处的观测值。

在网络架构方面，我们定义了一个包含 13 个输入、198 个输出和 2 个隐藏层的多层感知机，每层有 256 个节点。随后，我们对网络的每一层（包括输出层）应用 tanh 激活函数，并使用上述转换后的数据训练网络。验证时，我们首先根据（8）中定义的全部三种观测类别，将完整的观测数据库按残差高低分层，然后从各个分层中留出 5% 的数据作为验证集。该网络在 Keras [107] 中使用 RMS-prop 训练了 5,000 个周期。我们验证了训练后的模型在验证集的所有残差分层范围内均具有较低的相对误差。最重要的是，将输出反缩放至原始范围后，在残差绝对值在全部三种观测类别中均最低的底部分层中，验证集上的平均绝对误差（MAE）仅为 0.036，对于这个问题而言，这是可接受的精度。

在本节余下部分中，我们使用上述训练好的 Keras 模型，作为原始 Fayans 能量密度泛函校准问题的合成表示。

5.3 利用 ParMOO 中的结构性信息

平方和结构在单目标黑盒优化文献中已有充分研究。在这种策略的典型软件实现中 [68], , 会特别注意确保所有几何给主人留下些什么吧

变量名称	下界	上界
ρ_{eq}	0.146	0.167
E/A	-16.21	-15.50
K	137.2	234.4
J	19.5	37.0
L	2.20	69.6
h^v_{2-}	0.0	100.0
a^s_+	0.418	0.706
h^s_∇	0.0	0.516
κ	0.076	0.216
κ'	-0.892	0.982
f^ξ_{ex}	-4.62	-4.38
h^ξ_+	3.94	4.27
h^ξ_∇	-0.96	3.66

表3：Fayans EDF模型$T_{\mathbf{x}}(s)$参数空间稳定区域的上界和下界，据 [3]报告。我们只能在这些范围内获取观测数据，因此训练得到的多层感知机仅在这些界限内准确。

插值节点的条件得以满足，因此算法偶尔需要进行改善几何形状的评估，并采用一系列局部（信赖域）近似。然而，研究也表明，仅对这种复合结构进行建模，通常就能在实际应用中减少仿真评估次数 [64, 65, 66]。

在 ParMOO 内置的 LocalGaussRBF 代理模型实现中，信赖域半径会根据当前迭代点的第 $(n+1)$ 个最近邻的距离自适应选择。随后，每当信赖域约束代理问题的解无法使标量化目标值充分下降时，就会触发模型改进步骤；该步骤会从一个分布中抽取随机样本，该分布在当前仿真数据集中方差较低的方向上具有最高方差。这种方法无法保证渐近收敛速率方面的优势（事实上，在多目标场景中，甚至很难定义这样的收敛速率）。然而，我们将通过实证看到，将**R(x)**与**F(x)**分别建模，能够显著加快利用结构的多目标求解器在实际中的收敛速度。

5.4 ParMOO Fayans EDF求解器

为了展示其利用并行函数评估的能力（如果评估真实的 Fayans EDF模型而非合成多层感知机模型，我们就需要这种能力），我们使用 ParMOO 的 libEnsemble 接口，以批大小为 10 分发仿真评估。具体做法是使用 9 个 ε约束采集函数，分别以式（8）的解为目标，并使用一个固定权重，以式（7）的单目标公式的解为目标，因为它等价于 χ^2 loss。为了定义问题，我们在 ParMOO 中创建一个 libE MOOP对象，并添加 13 个连续设计变量，其名称和边界如表3所示。

为了按照上一节所述定义一个利用结构的变体，我们接着添加一个仿真函数，使用 LTR 约束高斯径向基函数（RBF）模型对合成多层感知机残差函数的全部 198 个分量进行建模，并使用包含 2,000 个点的拉丁超立方抽样（LHS）初始化其观测数据库。接下来，我们添加三个可微代数目标函数，分别计算三个不同观测类别中仿真输出的平方和。然后，为了让 ParMOO 能够聚焦于设计空间中有趣的区域（在这些区域中，没有任何观测类别的性能会很差），我们添加三个非线性约束，分别要求特定观测类别中的残差平方不超过 $10\times$ 其最优值（报告于[3]）。随后，ParMOO 在 LTR 中使用 SciPy 中的 L-BFGS-B [108]实现 [109] 求解代理优化问题。

为了与未利用结构时的性能进行比较（其他现有多目标软件便属于这种情况），我们定义了该问题的一个“黑箱”变体。对于黑箱变体，上文所述的 198 输出仿真被替换为一个 3 输出“黑箱”仿真，其中已计算出三个观测类别各自的平方和。然后，对于每个目标函数，我们都提供一个从各仿真输出到对应目标函数的恒等映射。除此之外，我们按照与结构化变体相同的方式定义该问题。

在这两种情况下，随后都运行 ParMOO 800 次迭代。由于我们从 2,000 点的拉丁超立方抽样搜索开始，并且每次迭代提供 10 个点的批大小，因此每个求解器的总预算为 10,000 次仿真评估。我们指出，对于典型的仿真优化问题而言，10,000 次仿真评估的预算大得不切实际。事实上，由于我们的高斯 RBF 代理模型较为复杂，而且求解代理优化问题的成本较高，在如此大的预算下，每次迭代的成本都极高。不过，运行到如此大的仿真预算仍然值得，这有助于了解求解器在极限情况下的性能。

请注意，由于求解代理优化问题的复杂性，以及评估 Keras 模型所需的成本不可忽略，因此求解过程需要耗费大量计算时间。为考虑 LHS 搜索和权重初始化中的随机性，我们对每种求解器分别运行 5 次，并对其性能取平均值。

为确保结果可复现，完整的实验设置（包括用作合成问题表示的训练后多层感知机）可在 https://github.com/parmoo/parmoo-solver-farm/tree/ main/fayans-model-calibration-2022 获取。请注意，出于易用性和兼容性考虑，我们已将 Keras 模型的权重转移至等效的 Torch 模型 [100], ，这一点也体现在 GitHub 仓库中。

5.5 结果

这里展示了基于 Keras/Torch 残差模型求解 (8) 的结果。为考虑初始试验设计中的随机性所带来的变异，所有性能结果均取五个随机种子的平均值。首先，借鉴 [3], 中使用的指标，我们在图 5 中展示了所有可观测类别的 χ^2 损失。请注意，对于 χ^2 损失，数值越小越好。接下来，为估计我们的方法在多目标方面的性能，我们在图 6 中展示了超体积性能指标的重缩放结果。对于超体积指标，数值越大越好。

请注意，超体积指标对问题缩放和 Nadir 点的选择极其敏感，因此其原始值难以解读。为对数值进行归一化，我们将超体积改进表示为相对于初始 2,000 点拉丁超立方抽样设计所支配的超体积的百分比，即当前超体积相对于该初始设计的增幅。然而，即使经过归一化，超体积改进的绝对值仍然难以解读，因为它仍受真实帕累托前沿与 Nadir 点之间总超体积的影响。从图 6 可以看出，总超体积的增幅不到 0.7%。这是因为 Nadir 点是由 ParMOO 对感兴趣值范围施加的下界约束确定的，而这些约束有意设置得过于悲观。不过，两种方法的超体积改进仍可用于衡量彼此的相对表现。

对于 χ^2 损失和超体积改进，结构化求解器的收敛速度都明显快于黑箱求解器。就 χ^2 损失而言，仅经过 200 次迭代（4,000 次仿真评估），结构化求解器的表现就优于黑箱求解器在全部 800 次迭代（10,000 次仿真评估）中所能达到的表现。这一结果在预料之中：由于我们的一个采集函数专门针对采用固定标量化权重的这一解，ParMOO 在这一性能指标上的行为与利用结构信息的单目标求解器相似。

就百分比超体积改进（一种真正的多目标性能指标）而言，ParMOO 的结构化求解器也显著提升了性能。不过，其收敛速度似乎略慢于 χ^2 loss。这是意料之中的，因为求解完整的多目标问题远比求解单个标量化问题困难。然而，尤其是在计算预算有限的情况下，结构化求解器的性能仍有大幅提升。

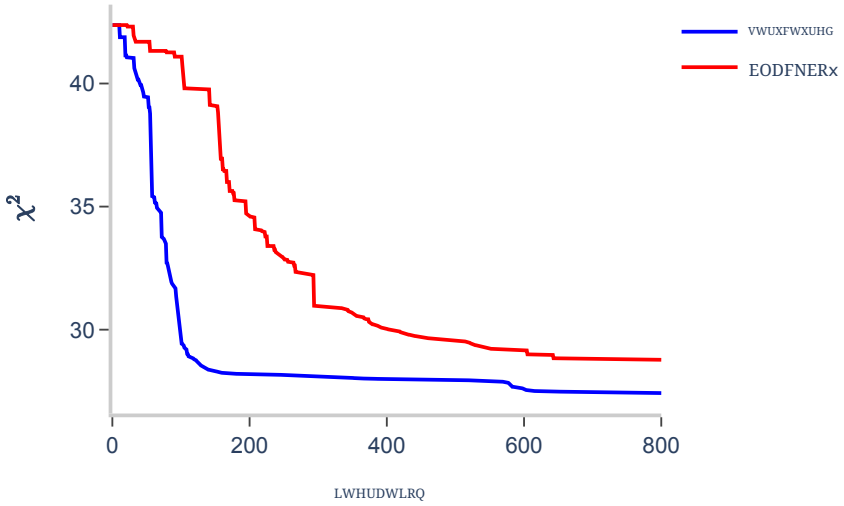


图 5：使用 ParMOO 求解 Fayans 能量密度泛函校准时，利用平方和结构（结构化方法）和采用标准方法（黑盒方法）的迭代次数与 χ^2 损失的关系。迭代至 k 结束时使用的仿真总数计算为 $2000 + 10k$。

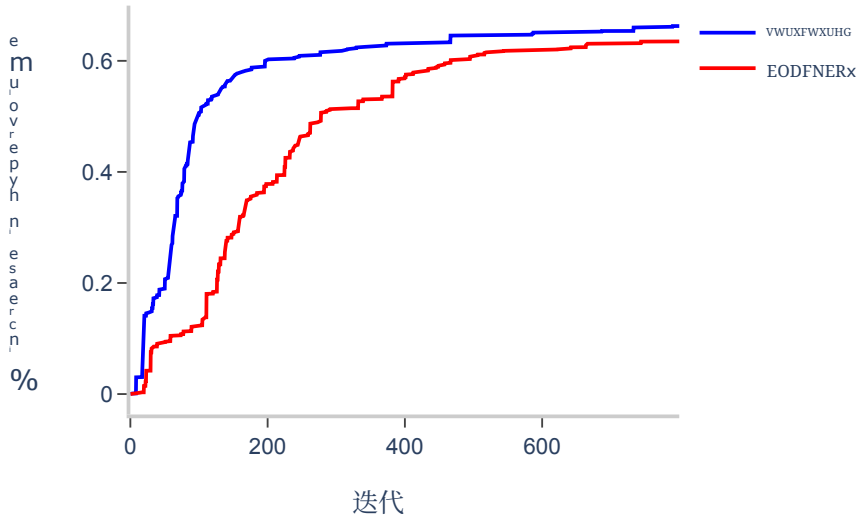


图 6：使用 ParMOO 求解 Fayans 能量密度泛函校准时，利用平方和结构（结构化方法）和采用标准方法（黑盒方法）的迭代次数与超体积改进百分比的关系。迭代至 k 结束时使用的仿真总数计算为 $2000 + 10k$。

6 案例研究：多目标化学设计

为展示 ParMOO 在处理异构目标方面的有效性，以及支持多样化科学工作流的灵活性，本节将使用 ParMOO 求解一个材料工程问题。其中，三个目标中的两个源自一项在湿实验室环境中开展的连续流化学实验，第三个目标则是代数形式的。与第5节一样，为使我们的结果开放且可复现，本节采用基于真实世界数据的化学响应面非线性模型，

使用核磁共振（NMR）光谱法对来自连续流反应器（CFR）的溶液进行测量所收集的数据。

6.1 连续流反应器材料设计的背景

在材料制造应用中，我们的目标是提出一种化学品制造技术，用于大规模生产具有高纯度的目标材料。具体而言，在此应用中，我们通过将两种预先选定的潜在溶剂之一与两种预先选定的潜在碱之一混合，来制造电解质 2,2,2-三氟乙基甲基碳酸酯（TFMC）。请注意，出于保密原因，本节不公开这些潜在溶剂和碱的真实名称；它们分别仅标记为 S1、S2 和 B1、B2。这不会影响我们结果的可复现性，因为在本节用于表示该问题的非线性模型中，它们也采用了这些标记。

我们的目标是找到溶剂、碱及反应条件（如反应时间、当量比和温度）的最佳组合，以便在连续流反应器中制备纯净的 TFMC 溶液。然而，除了制备纯净溶液外，我们还希望反应时间尽可能短，从而能够大规模生产大量 TFMC。我们预计，这需要采用较高的温度，而这可能会引发副反应并产生不需要的副产物，从而降低纯度。

6.2 化学响应面模型

在 ParMOO 的背景下，该问题包含三个连续设计变量和两个类别设计变量。这些变量的名称及其取值范围/潜在取值列于表 4 中。我们感兴趣的响应值是通过核磁共振波谱法测量得到的，在固定长度的时间窗口内 TFMC 和副产物生成量的积分。

变量名称	变量类型	下界	上界	合法值
温度（T）	连续	35 C	150 °C	N/A
反应时间（RT）	连续	60 秒	300 秒	N/A
当量比（EQR）	连续型	0.8	1.5	N/A
溶剂	类别型	N/A	N/A	S1,S2
base	类别型	N/A	N/A	B1,B2

表4：变量类型、范围以及本节讨论的连续流反应器材料优化问题的合法值。请注意，连续变量具有边界，而分类变量具有合法值。

由于我们无法为实验提供连续流反应器（CFR）的访问权限，本节使用一个非线性响应面来表示实际的连续流化学实验。这些模型利用现实世界的数据拟合而成；这些数据是通过将实际的 CFR/核磁共振（NMR）装置作为 ParMOO 可在闭环中查询的仿真系统来收集的。为此，ParMOO 的可扩展应用程序接口构建在 MDML 工具 [12], 之上。该工具使用 Apache Kafka 后端，将实验评估请求分发给 CFR，并收集 NMR 结果，再将结果直接返回到一个在线数据库，供 ParMOO 查询仿真结果。有关这些数据的收集方式，请参见 [110], ，其中介绍了如何使用相同的实验装置收集一个规模较小的数据集。

在达到 62 次实验的预算后，ParMOO 已收敛到若干个实际问题的近似解。由于现实世界实验材料的成本不可忽视，继续进行实验在经济上已不再可行。本节使用由设计点/积分值对组成的实验数据库拟合上述非线性模型。由于该数据集在五维输入空间中相对稀疏，我们特别注意确保所得模型不会呈现非物理行为。具体而言，为验证模型，我们确保两个响应面均

- 以较低的平均绝对误差（MAE）近似底层数据，尤其是针对接近最优的设计点；
- 在可行设计空间中的任何位置都不会取负值，因为输出代表材料生产的时间积分，而这在物理上是不可能的；

- 对于设计空间中的任何输入，其总和不得超过所提供的溶剂和碱的总量；以及

- 不要在设计空间的边界处出现意料之外的最大值/最小值，因为这些现象可能是该区域缺乏数据造成的假象，进而使模型过度外推。

为达到这些标准，我们根据对反应化学性质的预期，手工为响应面构建了少量具有物理意义的非线性项。随后，我们结合使用 scikit-learn[111]进行广义线性回归，将这些项的系数拟合到两个积分上。结果发现，平均绝对误差处于可接受范围内。通过使用 scipy.optimize.minimize [109], 中的 Powell 求解器进行超参数调优，我们还确保了每个模型各自的全局最小值均为非负值，且全局最大值之和处于可接受范围内。随后，我们验证了每个模型各自的最小值和最大值均位于设计空间中可接受的区域，这与我们的物理直觉和经验数据相符。

在本节余下部分，我们将使用这些训练好的响应面模型，作为 TFMC 和副产物积分各自的真实化学响应面。

6.3 异构问题结构

在这个问题中，三个目标中有两个来自真正的黑箱实验，必须在湿实验室环境中使用真实材料进行，且预算极其有限。然而，第三个目标表示反应时间，而反应时间是可直接控制的输入之一。因此，ParMOO 能够直接控制这一输出，并利用这一能力加快实际收敛速度。

6.4 ParMOOCFR 材料设计求解器

为了定义该问题，我们将表 4 所示的五个设计变量及其各自的变量类型和上下界/取值提供给 ParMOO。为处理两个类别型变量，我们使用 ParMOO 默认的分类变量嵌入器，将类别型变量的四种不同组合嵌入到三维连续潜在空间中。加上三个连续设计变量后，得到一个六维的有效优化空间。

对于该问题的结构化变体，ParMOO 使用第6.2节中描述的化学响应面的预训练非线性模型。ParMOO 被配置为将其视为具有两个输出的单个仿真，使用包含 50 次评估的拉丁超立方抽样（拉丁超立方抽样）生成初始数据库，并使用高斯径向基函数进行代理建模。接下来，为 ParMOO 指定三个目标，其中两个是仿真输出的恒等映射，第三个是反应时间设计变量的恒等映射。然后，添加两个 ε-约束函数和一个固定加权，使每次迭代的仿真评估批大小为三。

为了进行比较并展示利用异构问题结构的优势，我们还提供了针对该问题的 ParMOO 相同实现，其中第三个“反应时间”目标函数作为第三个黑箱仿真输出提供给 ParMOO。这样一来，ParMOO 会将第三个输出建模为黑箱函数，并忽略异构结构，这与使用大多数现成的多目标黑箱优化求解器时的情况相同。其他所有设置均与上述结构化变体相同。

在材料制造领域开展自动化实验的一项重大挑战，是原材料成本以及进行现实世界实验的成本。尽管我们并不掌握材料响应面计算模型的这些成本，但我们仍使用 ParMOO 进行了运行，预算设定得较为紧张：初始设计仅包含 50 个点，随后进行 30 轮迭代，每轮使用 3 个采集函数（共 140 次实验）。通过利用异构问题结构，我们希望在预算有限的情况下仍能取得良好表现（尤其是在成本较低的目标函数上）。

上述响应面模型以及用于复现本文所述实验的代码可在以下网址获取：
https://github.com/parmoo/parmoo-solver-farm/tree/main/cfr-material-design-2022。相关结果将在下一节中介绍。

6.5 结果

此处展示了基于我们的化学响应面对制造条件进行调优的结果。同样，所有性能结果均为 5 个不同随机种子的平均值。

首先，为了评估我们在唯一一个低成本目标函数上的表现，我们在图 7 中展示了满足 75% 纯度约束时观测到的平均最小反应时间（秒）。请注意，对于这些反应时间，数值越小越好。接下来，与第5节 5.5 中一样，我们在图 8 中展示超体积改进占初始超体积与总可能超体积（相对于理想点）之间差距的比例。同样，对于超体积改进，数值越大越好。

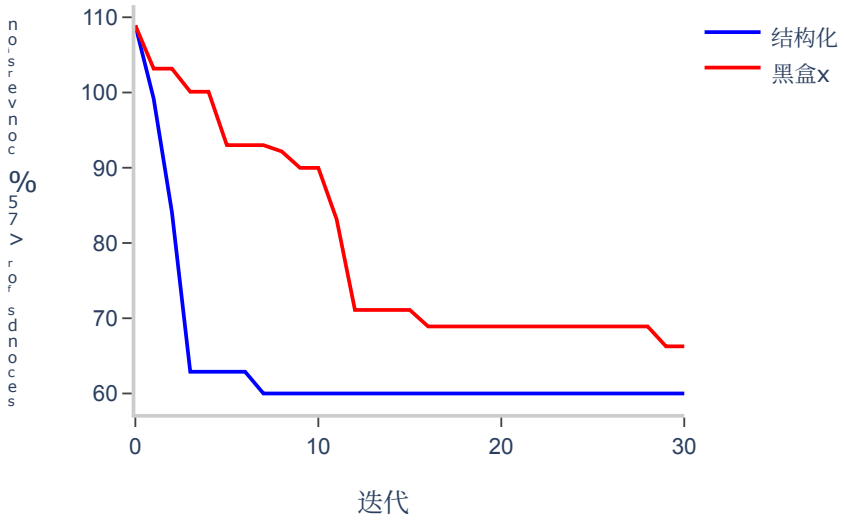


图 7：使用 ParMOO 求解连续流反应器化学品制造问题时，利用异构结构（结构化）和采用标准（黑盒）方法，迭代次数与达到至少 75% 转化率所需的最小反应时间之间的关系。到第 k 次迭代结束时使用的仿真总数计算为 $50 + 3k$。

与第5.5节中的情况一样，结构化求解器在两项指标上均大幅优于黑箱方法。我们注意到，尽管达到超过75%转化率所需的时间是一个具有明确物理意义的收敛指标，也体现了 ParMOO 直接控制反应时间（即代数目标）的能力，但这一次我们并未提供任何旨在明确针对该解的固定标量化方法，从而将问题简化为单目标优化问题。

值得注意的是，ParMOO 的结构利用型求解器在这个问题和第5节5.5中的 Fayans 问题上都表现出色，而用户几乎无需额外工作，尽管这两个问题的结构差异相当显著。

7 讨论与后续工作

在本文中，我们介绍了多目标仿真优化库 ParMOO 背后的设计原则。总而言之，我们的五个主要设计目标是

1. 求解器的可定制性；2. 在 MOSO 问题表述中利用复合结构；3. 灵活支持各种设计空间；4. 易于用于科学工作流；以及

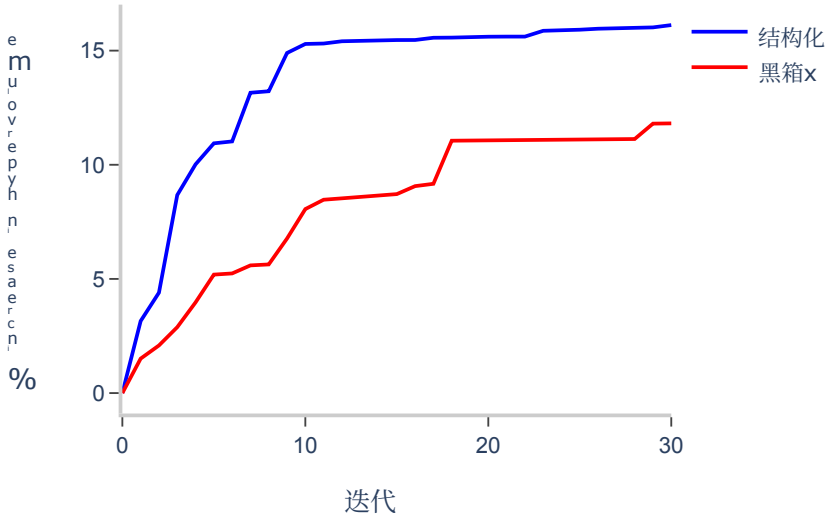


图 8：使用 ParMOO 求解连续流反应器化学品制造问题时，迭代次数与超体积改进百分比的关系，分别采用利用异质结构的方法（结构化）和标准方法（黑箱）。截至第 k 次迭代，总仿真次数为 $50 + 3k$。

5. 作为开源软件包的易用性、可扩展性和可维护性。

我们通过一种高度模块化的面向对象设计实现了这些目标。该设计采用中间仿真输出空间，并将复杂问题嵌入连续的潜在输入空间。我们已在两个现实问题的开源模型上展示了这一框架。这些模型以及用于复现我们结果的代码已通过 parmoo-solver-farm 代码库共享，出版时所有相关代码的快照可在 [112] 获取。

尽管这两个问题呈现出完全不同的结构，ParMOO 仍能在两种情况下有效利用其结构，并相较于不利用已知结构的黑箱方法显著改善收敛。值得注意的是，定义这样一种利用结构的求解器，用户几乎无需付出额外工作或进行定制。

致谢

我们感谢 Jared O'Neal、Witold Nazarewicz 和 Paul-Gerhardt Reinhard 提供 Fayans 泛函数据，也感谢 Joseph Libera、Jakob Elias、Santanu Chaudhuri 和 Trevor Dzwiniel 分别参与收集连续流反应器材料数据。我们还感谢 Jeff Larson、John-Luke Navarro 和 Steve Hudson 就科学软件开发最佳实践提供建议，并在我们使用 libEnsemble 时给予支持。我们感谢四位匿名审稿人的评论，这些意见改进了本文的呈现。

本研究部分由美国能源部科学办公室高级科学计算研究办公室的 SciDAC 计划资助，资助合同编号为 DE-AC02-05CH11231 和 DE-AC02-06CH11357。

参考文献

[1] J. Sobieszczanski-Sobieski, A. Morris, M. Van Tooren, 《知识工程支持的多学科设计优化》，约翰·威立父子公司，英国奇切斯特，2015年。

[2] W. Zhao, R. K. Kapania，采用 SpaRibs 和多个操纵面的复合材料飞翼多目标优化，载于：2018年多学科分析与优化会议论文集，美国航空航天学会，2018年，第3424页。doi:10.2514/6.2018-3424。[3] R. Bollapragada, M. Menickelly, W. Nazarewicz, J. O'Neal, P.-G. Reinhard, S. M. Wild，无导数数值物理模型拟合的优化与监督式机器学习方法，《物理学杂志G：核与粒子物理》48（2020年）024001。doi:10.1088/1361-6471/abd009。[4] T. H. Chang, J. Larson, L. T. Watson，高性能 LINPACK 求解器可变性的多目标优化，载于：2020年冬季仿真会议（WSC 2020）论文集，IEEE，2020年，第3081–3092页。doi:10.1109/WSC48552.2020.9383875。[5] N. Neveu, T. H. Chang, P. Franz, S. Hudson, J. Larson，LCLS-II光阴极注入器多目标优化方法比较，《计算物理通信》283（2023年）。doi:10.1016/j.cpc.2022.108566。[6] K. Kandasamy, K. R. Vysyaraju, W. Neiswanger, B. Paria, C. R. Collins, J. Schneider, B. Poczos, E. P. Xing，无需研究生调参：借助 Dragonfly 实现可扩展且稳健的贝叶斯优化，《机器学习研究杂志》21（2020年）1–27。网址：http://jmlr.org/papers/v21/18-223.html。[7] F. Karl, T. Pielok, J. Moosbauer, F. Pfisterer, S. Coors, M. Binder, L. Schneider, J. Thomas, J. Richter, M. Lang, E. C. Garrido-Merch´an, J. Branke, B. Bischl，机器学习中的多目标超参数优化：综述，技术报告4，2023年。doi:10.1145/3610536。[8] M. Parsa, J. P. Mitchell, C. D. Schuman, R. M. Patton, T. E. Potok, K. Roy，面向准确、快速且有效的神经网络加速器设计的贝叶斯多目标超参数优化，《神经科学前沿》14（2020年），文章编号667。doi:10.3389/fnins.2020.00667。[9] A. M. Schweidtmann, A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne, A. A. Lapkin，机器学习与连续流化学相结合：面向多个目标帕累托前沿的自动优化，《化学工程杂志》352（2018年）277–282。doi:10.1016/j.cej.2018.07.031。[10] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, A. G. Doyle，贝叶斯反应优化作为化学合成工具，《自然》590（2021年）89–96。doi:10.1038/s41586-021-03213-y。[11] M. Ehrgott，《多准则优化》，经济学与数学系统讲义系列，第2版，Springer Verlag，德国海德堡，2005年。doi:10.1007/3-540-27659-9。[12] J. R. Elias, R. Chard, J. A. Libera, I. T. Foster, S. Chaudhuri，制造数据与机器学习平台：借助物联网实现科学实验的实时监测与控制，2020年IEEE第六届世界物联网论坛（WF-IoT）（2020年）1–2。doi:10.1109/WF-IoT48130.2020.9221078。[13] R. H. Myers, D. C. Montgomery, C. M. Anderson-Cook，《响应面法：基于试验设计的过程与设计优化》，第4版，John Wiley & Sons, Inc.，美国新泽西州霍博肯，2016年。[14] C. Audet, W. Hare，《无导数与黑箱优化》，运筹学与金融工程系列，施普林格国际出版社，瑞士沙尔姆，2017年。doi:10.1007/978-3-319-68913-5。[15] T. H. Chang, S. M. Wild，ParMOO：用于并行多目标仿真优化的 Python 库，《Journal of Open Source Software》8（2023年）4468。doi:10.21105/joss.04468。[16] T. R. Marler, J. S. Arora，工程领域多目标优化方法综述，《结构与多学科优化》26（2004年）369–395。doi:10.1007/s00158-003-0368-6。

[17] S. R. Hunter, E. A. Applegate, V. Arora, B. Chong, 《多目标仿真优化导论》, ACM Transactions on Modeling and Computer Simulation 29 (2019) 1–36。doi:10. 1145/3299872。[18] G. Eichfelder, 《用于自适应求解多目标优化问题的标量化方法》, Computa- tional Optimization and Applications 44 (2009) 249–273。doi:10.1007/s10589-007-9155-4。[19] A. P. Wierzbicki, 参考点方法, 载于: T. Gal、T. J. Stewart、T. Hanne（编）, 《多准则决策: MCDM模型、算法、理论与应用进展》, Springer US, 波士顿, 马萨诸塞州, 1999年, 第237–275页。doi:10.1007/978-1-4615-5025-9_9。[20] B. Dandurand, M. M. Wiecek, 用于分解多目标优化的二次标量化方法, OR Spectrum 38 (2016) 1071–1096。doi:10.1007/s00291-016-0453-z。[21] R. E. Steuer, E.-U. Choo, 用于多目标规划的交互式加权切比雪夫程序, 《数学规划》26 (1983) 326–344。doi:10.1007/BF02591870。[22] I. Das, J. E. Dennis, 法向边界交点法: 一种生成非线性多准则优化问题帕累托曲面的新方法, SIAM Journal on Optimization 8 (1998) 631–657。doi:10.1137/S1052623496307510。[23] S. Deshpande, L. T. Watson, R. A. Canfield, 采用自适应权重方案的多目标优化, Optimization Methods and Software 31 (2016) 110–133。doi:10.1080/10556788.2015. 1048861。[24] M. Laumanns, L. Thiele, E. Zitzler, 基于ε-约束法的元启发式算法高效自适应参数变化方案, European Journal of Operational Research 169 (2006) 932–942。doi:10.1016/j.ejor.2004.08.029。[25] K. Deb, H. Jain, 采用基于参考点的非支配排序方法的进化多目标优化算法, 第一部分: 求解带盒约束的问题, IEEE 进化计算汇刊 18 (2013) 577–601。doi:10.1109/TEVC.2013.2281535。[26] H. Jain, K. Deb, 采用基于参考点的非支配排序方法的进化多目标优化算法, 第二部分: 处理约束并扩展为自适应方法, IEEE 进化计算汇刊 18 (2013) 602–622。doi:10.1109/TEVC. 2013.2281534。[27] C. Audet, G. Savard, W. Zghal, 一种用于多目标优化的网格自适应直接搜索算法, European Journal of Operational Research 204 (2010) 545–556。doi:10.1016/j.ejor.2009.11. 010。[28] J. Knowles, ParEGO: 一种采用在线景观近似的混合算法, 用于计算代价高昂的多目标优化问题, IEEE 进化计算汇刊 8 (2006) 1341–66。doi:10.1109/tevc.2005.851274。[29] G. Cocchi, G. Liuzzi, A. Papini, M. Sciandrone, 一种用于带盒约束无导数多目标优化的隐式过滤算法, Computational Optimization and Applications 69 (2018) 267–296。doi:10.1007/s10589-017-9953-2。[30] G. Liuzzi, S. Lucidi, F. Rinaldi, 一种用于约束多目标非光滑优化的无导数方法, SIAM Journal on Optimization 26 (2016) 2744–2774。doi:10.1137/15M1037810。[31] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, L. Salomon, 多目标优化中的性能指标, European Journal of Operational Research 292 (2021) 397–422。doi:10.1016/j.ejor. 2020.11.016。[32] T. H. Chang, L. T. Watson, J. Larson, N. Neveu, W. I. Thacker, S. Deshpande, T. C. H. Lux, 算法 1028: VTMOP: 黑箱多目标优化问题求解器, ACM Transactions on Mathematical Software 48 (2022) 文章编号: 36。doi:10.1145/3529258。

[33] K. Bringmann, T. Friedrich，超体积指标的逼近质量，Artificial Intelligence 195 (2013) 265–290。doi:10.1016/j.artint.2012.09.005。[34] K. Shang, H. Ishibuchi, L. He, L. M. Pang，进化多目标优化中超体积指标综述，IEEE 进化计算汇刊 25 (2020) 1–20。doi:10. 1109/TEVC.2020.3013290。[35] S. Bandyopadhyay, S. Saha, U. Maulik, K. Deb，一种基于模拟退火的多目标优化算法：AMOSA，IEEE 进化计算汇刊 12 (2008) 269–283。doi:10.1109/TEVC.2007.900837。[36] J. Larson, M. Menickelly, S. M. Wild，无导数优化方法，《数值学报》28 (2019) 287–404。doi:10.1017/S0962492919000060。[37] E. Bradford, A. M. Schweidtmann, A. Lapkin，采用高斯过程、谱采样和遗传算法的高效多目标优化，《全球优化期刊》71 (2018) 407–438。doi:10.1007/s10898-018-0609-2。[38] J. M¨uller，SOCEMO：计算代价高昂的多目标问题的代理优化，《INFORMS计算杂志》29 (2017) 581–596。doi:10.1287/ijoc.2017.0749。[39] M. A. Bouhlel, J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, J. R. Martins，带导数的 Python 代理建模框架，《工程软件进展》135 (2019) 102–662。doi:10.1016/j.advengsoft.2019.03.005。[40] E. W. Cheney, W. A. Light，《逼近理论教程》，数学研究生教材系列，美国数学会，普罗维登斯，罗得岛州，美国，2009年。[41] A. R. Conn, K. Scheinberg, L. N. Vicente，无导数优化中插值集的几何，Mathematical programming 111 (2008) 141–172。doi:10.1007/s10107-006-0073-5。[42] A. R. Conn, K. Scheinberg, L. N. Vicente，无导数优化导论，MPS-SIAM优化丛书，SIAM，费城，宾夕法尼亚州，美国，2009年。doi:10.1137/1.9780898718768。[43] J. Thomann, G. Eichfelder，一种用于异质多目标优化的信赖域算法，SIAM Journal on Optimization 29 (2019) 1017–1047。doi:10.1137/18m1173277。[44] M. Berkemeier, S. Peitz，采用径向基函数代理模型的无导数多目标信赖域下降法，《数学与计算应用》26 (2021) 31。doi:10.3390/mca26020031。[45] J.-H. Ryu, S. Kim，一种用于双目标优化的无导数信赖域方法，SIAM Journal on Optimization 24 (2014) 334–362。doi:10.1137/120864738。[46] T. M. Ragonneau, Z. Zhang，PDFO：Powell无导数优化求解器的跨平台接口，2021年。网址：https://github.com/pdfo/pdfo。[47] J. Larson, S. M. Wild，用于寻找多个极小值的异步并行优化求解器，Mathematical Programming Computation 10 (2018) 303–332。doi:10.1007/s12532-017-0131-4。[48] R. Garnett，《贝叶斯优化》，剑桥大学出版社，2023年。网址：https://bayesoptbook. com。[49] M. Emmerich, K. Yang, A. Deutz, H. Wang, C. M. Fonseca，贝叶斯全局优化的一种多准则推广，载于：P. M. Pardalos, A. Zhigljavsky, J. ˇZilinskas（编），《随机与确定性全局优化进展》，Springer，2016年，第229–242页。doi:10.1007/978-3-319-29975-4。[50] P. Feliot, J. Bect, E. Vazquez，一种用于约束单目标和多目标优化的贝叶斯方法，《全球优化期刊》67 (2016) 97–133。doi:10.1007/s10898-016-0427-3。

[51] S. Daulton、M. Balandat、E. Bakshy，面向并行多目标贝叶斯优化的可微期望超体积改进，载于：H. Larochelle、M. Ranzato、R. Hadsell、M. Balcan、H. Lin（编），神经信息处理系统进展，第33卷，Curran Associates, Inc.，2020年，第9851–9864页。网址：https://proceedings.neurips.cc/paper/2020/file/ 6fec24eac8f18ed793f5eaad3dd7977c-Paper.pdf。[52]D. Eriksson、M. Pearce、J. Gardner、R. D. Turner、M. Poloczek，通过局部贝叶斯优化实现可扩展的全局优化，《神经信息处理系统进展》第32卷（2019年）第1–12页。网址：https://proceedings.neurips.cc/paper/2019/file/ 6c990b7aca7bc7058f5e98ea909e924b-Paper.pdf。[53] S. S. Garud、I. A. Karimi、M. Kraft，用于代理模型开发的智能采样算法，《计算机与化学工程》96（2017年）103–114页。doi:10.1016/j.compchemeng.2016.10.006。[54] P. T. Roy、A. B. Owen、M. Balandat、M. Haberland，Python中的准蒙特卡罗方法，Journal of Open Source Software 8（2023年）5309。doi:10.21105/joss.05309。[55] M. Balandat、B. Karrer、D. Jiang、S. Daulton、B. Letham、A. G. Wilson、E. Bakshy，BoTorch：用于有效蒙特卡罗贝叶斯优化的框架，载于：H. Larochelle、M. Ranzato、R. Hadsell、M. Balcan、H. Lin（编），神经信息处理系统进展，第33卷，Curran Associates, Inc.，2020年，第21524–21538页。网址：https://proceedings.neurips.cc/paper/2020/file/ f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf。[56] E. F. Campana、M. Diez、G. Liuzzi、S. Lucidi、R. Pellegrini、V. Piccialli、F. Rinaldi、A. Serani，一种用于船体优化的多目标DIRECT算法，Computational Optimization and Applications 71（2018年）53–72页。doi:10.1007/s10589-017-9955-0。[57] J. Bigeon、S. Le Digabel、L. Salomon，DMulti-MADS：用于黑箱多目标优化的网格自适应直接多重搜索，Computational Optimization and Applications 79（2020年）301–338页。doi:10. 1007/s10589-021-00272-9。[58] A. L. Cust´odio、J. F. A. Madeira、A. I. F. Vaz、L. N. Vicente，用于多目标优化的直接多重搜索，SIAM Journal on Optimization 21（2011年）1109–1140页。doi:10.1137/10079731x。[59] A. L. Cust´odio、J. F. A. Madeira，MultiGLODS：使用直接搜索进行全局与局部多目标优化，《全球优化期刊》72（2018年）323–345页。doi:10.1007/s10898-018-0618-1。[60] A. Abraham、L. Jain、R. Goldberg（编），《进化多目标优化：理论进展与应用》，高级信息与知识处理丛书，Springer Verlag，伦敦，英国，2005年。doi:10.1007/1-84628-137-7。[61] K. Deb、A. Pratap、S. Agarwel、T. Meyarivan，一种快速且精英化的多目标遗传算法：NSGA-II，IEEE 进化计算汇刊6（2002年）182–197页。doi:10.1109/4235. 996017。[62] G. E. Karniadakis、I. G. Kevrekidis、L. Lu、P. Perdikaris、S. Wang、L. Yang，物理信息机器学习，《自然·物理评论》3（2021年）422–440页。doi:10.1038/s42254-021-00314-5。[63] E. Pickering、S. Guth、G. E. Karniadakis、T. P. Sapsis，通过神经算子中的主动学习发现和预测极端事件，《自然·计算科学》2（2022年）823–833页。[64] R. Astudillo、P. I. Frazier，深入探究黑箱：灰箱贝叶斯优化教程，载于：2021年冬季仿真会议（WSC 2021）论文集，IEEE，2021年。doi:10.1109/WSC52266.2021. 9715343。[65] K. A. Khan、J. Larson、S. M. Wild，对由光滑分量分段线性复合而成的非凸函数进行优化的流形采样，SIAM Journal on Optimization 28（2018年）3001–3024页。doi:10.1137/17m114741x。

[66] J. Larson、M. Menickelly，面向高成本无导数非光滑复合优化的结构感知方法，Mathematical Programming Computation 16（2024年）1–36页。doi:10.1007/ s12532-023-00245-5。[67] H. Zhang、A. R. Conn，最小二乘最小化无导数算法的局部收敛性，Computational Optimization and Applications 51（2012年）481–507页。doi:10.1007/ s10589-010-9367-x。[68] S. M. Wild，使用POUNDERS求解无导数非线性最小二乘问题，载于：T. Terlaky、M. F. Anjos、S. Ahmed（编），Advances and Trends in Optimization with Engineering Applications，SIAM，2017年，第529–540页。doi:10.1137/1.9781611974683.ch40。[69] A. Ben´ıtez-Hidalgo、A. J. Nebro、J. Garc´ıa-Nieto、I. Oregi、J. Del Ser，jMetalPy：用于元启发式多目标优化的Python框架，Swarm and Evolutionary Computation 51（2019年）100598。doi:10.1016/j.swevo.2019.100598。[70] F. Biscani、D. Izzo，用于优化的并行全局多目标框架：pagmo，Journal of Open Source Software 5（2020年）2338。doi:10.21105/joss.02338。[71] J. Blank、K. Deb，pymoo：Python中的多目标优化，IEEE Access 8（2020年）89497–89509页。doi:10.1109/ACCESS.2020.2990567。[72] S. Le Digabel，算法 909：NOMAD：使用MADS算法进行非线性优化，ACM Transactions on Mathematical Software 37（2011年），文章编号44。doi:10.1145/1916461.1916468。[73] G. Feldman、S. R. Hunter，用于双目标排序与选择的SCORE分配，ACM Transactions on Modeling Computer and Simulation 28（2018年）。doi:10.1145/3158666。[74] C. Audet、E. Hall´e-Hannan、S. Le Digabel，含元变量和类别变量的约束混合变量黑箱优化问题的一般数学框架，载于：Operations Research Forum，第4卷，Springer，2023年，第12页。doi:10.1007/s43069-022-00180-6。[75] P. Saves、R. Lafage、N. Bartoli、Y. Diouane、J. Bussemaker、T. Lefebvre、J. T. Hwang、J. Morlier、J. R. R. A. Martins，SMT 2.0：专注于层级与混合变量高斯过程的代理建模工具箱，技术报告，arXiv预印本，2023年。doi:10.48550/arXiv.2305. 13998。[76] H. Moriwaki、Y.-S. Tia、N. Kawashita、T. Takagi，Mordred：分子描述符计算器，Journal of Cheminformatics 10（2018年）。doi:10.1186/s13321-018-0258-y。[77] S. Le Digabel、S. M. Wild，黑箱仿真优化中的约束分类，技术报告2，2024年。doi:10.1007/s11081-023-09839-3。[78] C. Audet、J. E. Dennis，用于无导数非线性规划的渐进屏障，SIAM Journal on Optimization 20（2009年）445–472页。doi:10.1137/070692662。[79] R. M. Lewis、V. Torczon，一种用于含一般约束和简单界限的优化问题的全局收敛增广拉格朗日模式搜索算法，SIAM Journal on Optimization 12（2002年）1075–1089页。doi:10.1137/S1052623498339727。[80] G. Cocchi、M. Lapucci，一种用于多目标优化的增广拉格朗日算法，Computational Optimization and Applications 77（2020年）29–56页。doi:10.1007/s10589-020-00204-z。[81] R. Chard、Y. Babuji、Z. Li、T. Skluzacek、A. Woodard、B. Blaiszik、I. Foster、K. Chard，funcX：面向科学的联邦函数服务平台，载于：第29届高性能并行与分布式计算国际研讨会（HPDC ’20）论文集，ACM，2020年，第65–76页。doi:10.1145/3369583.3392683。[82] S. Hudson、J. Larson、J.-L. Navarro、S. M. Wild，libEnsemble：用于动态计算集成的完整Python工具包，技术报告92，2023年。doi:10.21105/joss.06031。

[83] S. Hudson, J. Larson, J.-L. Navarro, S. M. Wild，libEnsemble：用于协调动态计算集成并发评估的库，《IEEE并行与分布式系统汇刊》33 (2022年) 977–988。doi:10.1109/tpds.2021.3082815.[84]R. M. Kolonay, M. Sobolewski，用于大规模、分布式、动态保真度气动弹性分析的面向服务的计算环境 (SORCER)，载于：国际气动弹性与结构动力学论坛（IFASD 2011），《优化》，Citeseer，2011年，第26–30页。URL: http: //citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.656.7539.[85] T. H. Chang, J. Larson, L. T. Watson, T. C. H. Lux，使用 libEnsemble 管理计算成本高昂的黑箱多目标优化问题，载于：2020年春季仿真会议论文集（SpringSim 2020），第28届高性能计算研讨会（HPC '20），SCS，2020年，第31号文章。doi:10.22360/SpringSim.2020.HPC.001.[86] C. Raghunath, T. H. Chang, L. T. Watson, M. Jrad, R. K. Kapania, R. M. Kolonay，面向服务的架构中的全局确定性与随机优化，载于：2017年春季仿真会议论文集（SpringSim 2017），第25届高性能计算研讨会（HPC '17），SCS，美国弗吉尼亚州弗吉尼亚海滩，2017年，第7号文章。doi:10.22360/springsim.2017.hpc.023.[87] M. A. Heroux, L. McInnes, D. E. Bernholdt, A. Dubey, E. Gonsiorowski, O. Marques, J. D. Moulton, B. Norris, E. Raybourn, S. Balay, R. A. Bartlett, L. Childers, T. Gamblin, P. Grubel, R. Gupta, R. Hartman-Baker, J. C. Hill, S. Hudson, C. Junghans, A. Klinvex, R. Milewicz, M. Miller, H. Ah Nam, J. O'Neal, K. Riley, B. Sims, J. Schuler, B. F. Smith, L. Vernon, G. R. Watson, J. Willenbring, P. Wolfenbarger，通过更优质的科学软件提升科学生产力：开发者生产力与软件可持续性报告，技术报告 ORNL TM-2020 1459 / ECP-U-RPT-2020-0001，橡树岭国家实验室，美国田纳西州橡树岭，2020年。doi:10.2172/1606662.[88] C. Audet, S. Le Digabel, V. Rochon Montplaisir, C. Tribes，算法 1027：NOMAD 4版：使用 MADS 算法进行非线性优化，《ACM Transactions on Mathematical Software》48 (2022年)。doi:10.1145/3544489.[89]S. Mannor, V. Perchet, G. Stoltz，未知博弈中的可接近性：在线学习与多目标优化相遇，载于：第27届学习理论会议论文集（PMLR），机器学习研究论文集第35卷，PMLR，西班牙巴塞罗那，2014年，第339–355页。URL: https://proceedings.mlr.press/v35/mannor14.html.[90] C. F. Hayes, R. R˘adulescu, E. Bargiacchi, J. K¨allstr¨om, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz 等，多目标强化学习与规划实用指南，《自主智能体与多智能体系统》36 (2022年) 1–59。doi:10.1007/s10458-022-09552-y.[91] T. P. Sapsis, A. Blanchard，使用高斯过程回归进行数据驱动降阶建模时数据选择的最优准则及其渐近形式，《英国皇家学会哲学汇刊A辑》380 (2022年) 20210197。doi:10.1098/rsta.2021.0197.[92] E. Zitzler, M. Laumanns, L. Thiele，SPEA2：改进强度帕累托进化算法，TIK 报告 103 (2001年)。doi:10.3929/ethz-a-004284029.[93] D. Hadka，Platypus——Python 中的多目标优化，技术报告 1.0.4版，GitHub，2015年。URL: https://platypus.readthedocs.io/en/latest.[94] Y. Tian, R. Cheng, X. Zhang, Y. Jin，PlatEMO：用于进化多目标优化的 MATLAB 平台 [教育论坛]，《IEEE计算智能杂志》12 (2017年) 73–87。doi:10. 1109/MCI.2017.2742868.[95] J. J. Durillo, A. J. Nebro，jMetal：用于多目标优化的 Java 框架，《工程软件进展》42 (2011年) 760–771。doi:10.1016/j.advengsoft.2011.05.014.

[96] F.-A. Fortin、F.-M. De Rainville、M.-A.Gardner、M.Parizeau、C. Gagn´e ”，DEAP：让进化算法变得简单，《机器学习研究杂志》13（2012）2171–2175。网址：https: // www.jmlr.org/papers/v13/fortin12a.html。[97] G. e. a. Liuzzi，Dfo-lib，2024年。网址：https://github.com/DerivativeFreeLibrary。[98] S. Tavares、C.P. Br´as、A. L. Cust´odio、V. Duarte、P. Medeiros，直接多重搜索的并行策略，《数值算法》92（2022年）1757–1788。doi:10.1007/s11075-022-01364-1。[99] K. Cooper、S. R. Hunter，PyMOSO：使用R-PERLE和R-MinRLE进行多目标仿真的软件，《INFORMS计算杂志》32（2020年）1101–1108。doi:10.1287/ijoc. 2019.0902。[100] A. Paszke、S. Gross、F. Massa、A. Lerer、J.Bradbury、G. Chanan、T. Killeen、Z. Lin、N. Gimelshein、L. Antiga、A.Desmaison、A. Kopf、E. Yang、Z. DeVito、M. Raison、A. Tejani、S. Chil-amkurthy、B. Steiner、L.Fang、J. Bai、S.Chintala，PyTorch：命令式风格的高性能深度学习库，载于：H.Wallach、H. Larochelle、A.Beygelzimer、F. d'Alch´e-Buc、E.Fox、R. Garnett（编），《神经信息处理系统进展》，第32卷，Curran Associates，Inc.，2019年，第1–12页。网址：https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf。[101] T. H. Chang、S.M. Wild、H. Dickinson，ParMOO：用于并行多目标仿真的Python库，《技术报告》版本0.3.1，阿贡国家实验室，Lemont，IL，美国，2023年。网址：https://parmoo.readthedocs.io/en/latest。[102] E. Gamma、R. Helm、R. Johnson、J. Vlissides，设计模式：可复用面向对象软件的要素，Addison-Wesley，Reading，MA，美国，1995年。[103] I. Dunning、J.Huchette、M. Lubin，JuMP：用于数学优化的建模语言，《SIAM评论》59（2017年）295–320。doi:10.1137/15M1020575。[104] W. E.Hart、C. D. Laird、J.-P. Watson、D. L. Woodruff、G. A. Hackebeil、B. L. Nicholson、J.D. Siirola，Pyomo——使用Python进行优化建模，Springer优化及其应用，第2版，Springer Cham，Cham，瑞士，2017年。doi:10.1007/978-3-319-58821-6。[105] G. Chen、T. H.Chang、J.Power、C.Jing，粒子加速器设计的集成多物理场优化框架，载于：2023年冬季仿真会议（WSC 2023）论文集，工业应用专题，2023年。doi:10.48550/arXiv.2311.09415。[106] K. Deb、L.Thiele、M. Laumanns、E. Zitzler，可扩展的多目标优化测试问题，载于：2002年IEEE进化计算大会（CEC’02）论文集，第1卷，IEEE，2002年，第825–830页。doi:10.1109/CEC.2002.1007032。[107] F.Chollet等，Keras，https://keras.io，2015年。[108] C.Zhu、R.H. Byrd、P.Lu、J.Nocedal，算法778：L-BFGS-B：用于大规模边界约束优化的Fortran子程序，《ACM Transactionson MathematicalSoftware》23（1997）550–560。doi:10.1145/279232.279236。[109] P. Virtanen、R. Gommers、T. E.Oliphant、M. Haberland、T.Reddy、D. Cournapeau、E. Burovski、P.Peterson、W. Weckesser、J. Bright、S. J.van der Walt、M. Brett、J.Wilson、K.JarrodMillman，N. Mayorov、A.R. J. Nelson、E. Jones、R. Kern、E. Larson、C. Carey、I. Polat、Y.Feng、E. W.Moore、J. VanderPlas、D.Laxalde、J. Perktold、R. Cimrman、I. Henriksen、E. A. Quintero、C. R. Harris、A. M. Archibald、A. H.Ribeiro、F. Pedregosa、P. vanMulbregt、S. Contributors，SciPy 1.0：Python科学计算的基础算法，《自然·方法》17（2020年）261–272。doi:10.1038/s41592-019-0686-2。

[110] T. H. Chang、J. R. Elias、S. M. Wild、S. Chaudhuri、J. A. Libera，《通过多目标优化和主动学习实现材料全自主设计的框架：挑战与后续步骤》，载于：第 11 届国际学习表征会议（ICLR 2023），材料机器学习研讨会（ML4Materials），2023年，第 1–10 页。网址：https://openreview.net/forum?id=8KJS7RPjMqG，即将发表。

[111] F. Pedregosa、G. Varoquaux、A. Gramfort、V. Michel、B. Thirion、O. Grisel、M. Blondel、P. Prettenhofer、R. Weiss、V. Dubourg 等，《Scikit-learn：Python 中的机器学习》，《机器学习研究杂志》12（2011年），2825–2830 页。网址：https://www.jmlr.org/papers/volume12/ pedregosa11a/pedregosa11a.pdf。

[112]T. H. Chang、S. M. Wild，《设计一个用于解决多目标仿真优化问题的框架》，2024年。doi:10.1287/ijoc.2023.0250.cd，可从 https://github.com/INFORMSJoC/2023.0250 下载。

这份提交的稿件由芝加哥大学阿贡有限责任公司（阿贡国家实验室的运营方，以下简称“阿贡”）编制。阿贡是美国能源部科学办公室下属的实验室，根据合同编号 DE-AC02-06CH11357 运营。美国政府为自身及代表其行事的其他机构保留一项已付费、非独占、不可撤销的全球许可，可复制该文章、制作其衍生作品、向公众分发副本，并代表政府或由政府公开表演和展示该文章。能源部将根据 DOE 公共访问计划，向公众开放这些由联邦资助研究产生的成果。 http://energy.gov/downloads/doe-public-access-plan