性能目标对指导软件配置调优是否重要?

TAO CHEN,拉夫堡大学，英国MIQING LI, 伯明翰
大学，英国

arXiv:2301.03290v1 [cs.SE] 9 Jan 2023

可配置软件系统可以通过调优来提升性能。借助一些帕累托优化器，近期研究已从针对单一、与时间相关的性能目标进行调优，转向针对两个本质上不同的目标进行调优，这两个目标分别评估系统不同的性能方面，且各自有不同的待满足期望值，例如“延迟低于 10 秒”，同时“内存使用量不超过 1 GB”。在设计更好的优化器之前，其中一项至关重要的工程决策是：在调优过程中如何处理具有明确期望值的性能需求。为此，业界采用了两种不同的优化模型：一种是量化期望值并将其纳入指导调优的搜索目标；另一种是在搜索过程中不考虑期望值，而仅在后续决策过程中使用。然而，尽管这是一项至关重要的决策，它决定了优化器的设计方式及其定制方案，但对于在何种具体情况下应选择哪种优化模型，以及原因何在，人们仍知之甚少。

本文旨在弥合这一差距。首先，我们综述了文献中的426篇论文和14个真实需求数据集，并据此总结出四种性能需求模式，用于量化配置调优中的期望。基于这些内容，我们随后开展了一项全面的实证研究，涵盖最先进的性能需求模式的15种组合、四种期望空间类型、三种帕累托优化器，以及八种真实系统/环境，共研究了1,296个案例。研究结果表明：（1）期望的现实性是决定是否应使用期望来指导调优的关键因素；（2）给定的模式以及现实期望在目标景观中的位置，对选择的影响较小，但确实会影响改进幅度；（3）可用的调优预算也会影响对不现实期望值的选择，但在现实期望下影响并不显著。为促进开放科学实践，我们公开了代码和数据集，地址如下：
https://github.com/ideas-labo/aspiration-study。

CCS概念： • **软件及其工程 → 基于搜索的软件工程**； 实证软件验证； 软件性能。

附加关键词和短语：基于搜索的软件工程、软件配置调优、性能需求、性能期望、多目标优化。

ACM参考格式：
Tao Chen 和 Miqing Li。2023年。性能目标对指导软件配置调优是否重要? *ACM*软件工程与方法学汇刊 1，1，第1篇文章（2023年1月），共41页。https://doi.org/10.1145/3571853

1 引言

许多软件系统具有高度可配置性，因此拥有数量惊人的配置选项（例如 Apache Storm 中的 max_spout），软件工程师可以对其进行调优

作者地址：Tao Chen，t.t.chen@lboro.ac.uk，拉夫堡大学，英国拉夫堡；Miqing Li，伯明翰大学，英国伯明翰，m.li.8@bham.ac.uk。

允许免费为个人或课堂用途制作本作品全部或部分内容的数字副本或纸质副本，前提是这些副本不得用于营利或商业用途，亦不得复制或分发，并且每份副本首页均须附上本声明及完整引文。对于本作品中归作者以外的其他人所有的组件，必须尊重其版权。注明出处后，允许摘录。若要以其他方式复制、再版、发布到服务器或分发至邮件列表，则须事先获得明确许可和/或支付费用。请联系 permissions@acm.org 申请许可。© 2023年 版权归所有者/作者所有。出版权已授权给 ACM。1049-331X/2023/1-ART1 $15.00 https://doi.org/10.1145/3571853

以满足某些性能目标的需求，例如改善延迟、吞吐量和资源消耗 [12, 15, 36, 52, 78]。一项近期访谈显示，业界从业者已认识到，软件系统的配置调优在软件工程中发挥着重要作用，是软件产品成功的关键 [67]。事实上，据报道，全球有 59% 的软件性能问题（其中性能需求遭到严重违反）与配置不当有关，而非代码问题 [39], ，并由此造成严重后果。例如，2017 至 2018 年间，配置相关的性能问题每小时给全球超过 50% 的软件公司造成至少 40 万美元的损失[1]。

寻找良好的配置（即配置选项可能的组合设置）颇具挑战，因为：

- 默认配置往往远非理想。Jamshidi 和 Casale [44] 表明，Apache Storm 的默认配置可能导致性能比某些其他配置差 480 倍。
- 配置空间可能很大，而且测量往往成本高昂， [62], 这使得贪心搜索不切实际。

- 尽管传统上软件配置调优一直侧重于单一性能目标， [6, 11, 37, 58, 64, 66, 71, 77, 79], 近期研究却指出，同时针对多个性能目标进行调优十分必要。我们的综述（见第3节）发现，考虑两个性能目标是最常见的情况 [16, 32, 41, 62]。例如，对于 x264 这样的视频编码器，提高图像质量并降低能耗自然都至关重要；对于深度学习模型（如 Keras 等框架支持的深度神经网络），更高的准确率和更短的训练时间则是两个固有的性能目标。这进一步增加了调优过程的复杂性，因为各性能目标之间可能相互冲突，而且冲突程度往往事先未知 [16, 62]。

为自动调优软件配置以获得更好的性能，人们提出了多种方法，例如基于规则的方法 [31, 35], 、基于学习的方法 [3, 45], 以及基于搜索的方法 [8, 9, 16, 50,62, 70]。其中，基于搜索的方法主要依赖于基于搜索的软件工程（SBSE）范式中广泛使用的帕累托优化器 [40], ，是应对软件配置调优中上述所有挑战的一种有前景的方法，尤其适用于存在多个性能目标的情况 [8, 9, 16, 70]。简而言之，帕累托优化器通常会维护一个配置种群（或至少一个档案集），并反复生成这些配置的副本，通过直接分析软件来评估它们，旨在找到帕累托最优配置。其输出是一组彼此非支配的配置，可用于近似软件系统的帕累托前沿。

1.1 问题及其重要性

软件配置调优中的一个重要因素，是可能存在对性能目标有明确期望值的需求 [8, 9, 27, 65], ；本文将这类需求区分为两个重要概念：

- **期望值：** 用于量化性能被认为令人满意（或不满意）的程度的信息。

- **性能需求：** 定义性能偏好的情境。

例如，根据需求工程领域的研究 [4, 75], ，需求文档中包含性能需求并不少见，例如“延迟应小于 x”，而“内存使用量应不超过 y”，

[1]https://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html

其中，“小于 x”和“不超过 y”就是明确的期望值。值得注意的是，并非所有性能需求都包含期望值，例如，“延迟应较低”就是一个没有期望值的需求，因为无法量化其满意程度。事实上，在性能需求中包含明确期望值的场景下，人们普遍认为，所提供的信息可作为有用的指标，帮助软件工程师在调优完成后进行择优选择，从帕累托优化器 [55]生成的配置集合中提取满意配置。其背后的自然动机是：

- 在调优预算固定的情况下，找到最优性能并非总是可行，甚至未必符合利益相关者的期望。
- 明确的期望水平使利益相关者能够对相互冲突的性能目标进行隐式权衡或表达偏好。

无论使用哪种帕累托优化器，在调优过程中，现有研究采用两种本质上不同的优化模型来处理针对两个性能目标进行调优时的期望值，即：**带期望的帕累托搜索**（记为 PS-w）[8, 9, 33, 59] 和**不带期望的帕累托搜索**（记为 PS-w/o） [16, 49, 62, 70]。在 PS-w 中，带有期望值的性能需求会以特定形式量化（我们将在第3节详细阐述），进而作为调优中的新搜索目标。其动机很简单：由于期望值提供了关于满意程度的信息，因此可以利用这一优势来指导调优过程。相较之下，PS-w/o 更为传统，在调优时直接忽略期望值。这里的假设是，无论使用哪种帕累托优化器，其搜索本质上都是一个旨在寻找帕累托最优配置的优化过程，因此调优始终以实现尽可能最佳的性能为目标，从而无论期望值如何，都保留了追求满意化的倾向[2]。例如，找到延迟为=10s、内存使用量为=1GB 的帕累托最优配置，必然会满足“延迟应低于 20s”以及“内存使用量不得超过 2GB”的需求和期望。这与 Odhnoff 的观点一致，即“优化”和“满意化”仅在风格上有所不同，但在本质上是相同的 [63]。

尽管两个优化模型分别被相应的研究团队采用，但模型的选择大多是临时决定的，而且人们往往默认“它们差别不大，因此可以随意使用。”因此，我们对在何种具体情况下应选择哪种优化模型，以及原因是什么，仍然知之甚少。一些研究人员也充分印证了这一点。Ghanbari等人 [34] 指出，考虑模型选择十分重要，因为引导调优的函数形态，尤其是在达到期望值之后，可能会影响优化器的行为；但他们并未讨论这会带来什么蕴含关系。Fekry等人 [28] 近期的另一项研究则建议，研究是否应利用期望值来引导优化器，以及衡量其效果，是软件配置调优未来面临的一项重要挑战。事实上，**对此加以理解并非易事，但这有助于从业者做出更明智的决策，** 特别是对于可配置软件系统而言，测量成本高昂，因此逐个案例对比这两种模型并不现实。此外，**这些见解还可以启发软件配置调优未来的研究方向：** 如果 PS-w/o 更有前景，那么我们就可以大幅简化研究，将重点放在设计有效的优化器上，而不必考虑给定的需求，因为人工输入（即需求/期望值）在整体调优过程中并不那么重要。另一方面，如果 PS-w 整体上更有效，那么问题

[2]这假设了最常见的情况：最佳性能至少与其他某些取值同样受青睐。

就会变得更加复杂，但也会带来更多机遇。例如，未来的研究可以重点关注如何更好地量化这些性能需求和期望值，以及如何将它们更有效地融入更专业化的优化器中。

为了理解这一点，我们还关注了通用多目标优化领域的文献，尤其是偏好驱动的多目标优化 [5, 51, 74, 80]。然而，由于以下两个原因，我们没有找到与我们的情况直接相关的答案：（1）偏好驱动的多目标优化中偏好的表示方式（例如权重和排名）与我们从软件配置调优相关研究中总结出的需求模式不同；（2）相关研究主要开发针对特定偏好表示方式定制的算法/优化器，而软件配置调优通常依赖基础优化器 [8, 9]。

因此，我们的研究旨在回应学界对理解以下问题的需求：

> 我们是否应将需求和期望纳入软件配置调优过程的指导？如果应该，在哪些情境下这样做，原因是什么？

1.2 研究问题

本文通过一项实证研究来弥补上述空白，系统比较 PS-w 和 PS-w/o 在两个性能目标下调优软件配置的表现。假设存在一些现实的 期望（即所有期望都可以通过调优软件系统的配置实现），我们希望回答的第一个研究问题（RQ）是：

> ***RQ1***：在具有现实期望的性能需求下，PS-w 和PS-w/o哪种方法能找到更优的一组配置？

RQ1 旨在全面比较这两种优化模型。然而，各种可能的需求场景意味着，具体期望在目标景观中可能存在很大差异。例如，人们可能对延迟有较高的期望，但对吞吐量的要求较低，反之亦然。因此，我们想进一步了解的是：

> ***RQ2***：不同的现实期望如何影响结果？

RQ1 和**RQ2** 探讨的是给定期望合理且可实现的常规情境。然而，由于实际期望是由软件工程师和利益相关者事先协商确定的，因此它们可能会变得不现实 ，需要将关注点扩展到配置之外。也就是说，尽管单个性能目标可能达到相应的期望水平，但搜索空间中可能不存在任何配置，能同时满足所有性能目标的期望水平。这引出了下一个研究问题：

> 研究问题***3***：如果给定的期望值不切实际，会怎样？

虽然我们关注的是调优预算足以实现良好收敛的案例，但在现实场景中，由于需要快速发布或进行任务优先级排序等原因，用于软件配置调优的资源可能有限。因此，我们的最后一个问题旨在探讨：

> 研究问题***4***：给定的调优资源（调优预算）是否会影响在PS-w 和PS-w/o之间的选择？

1.3 贡献

为了解决这些研究问题，我们针对15种模式组合进行了广泛的实证研究，以量化期望值、目标景观中的四种期望空间、三种帕累托优化器，以及具有多样化性能目标的八个真实系统/环境，共开展了1,296个案例的研究。简而言之，本文的首项贡献 是总结出一组性能需求模式（针对单个性能目标），这些模式源自软件工程领域文献中的426篇论文以及14个广泛使用的真实世界需求数据集。这些模式包括：

(1) 没有给出期望值，但假设人们偏好尽可能达到最优性能，例如“延迟越低越好”，即人们偏好尽可能低的延迟。(2) 期望空间内的性能同样好，或者存在一定程度的容忍度，例如“理想情况下，最低延迟应为500毫秒”，这意味着优于*500*毫秒 的性能同样好，而比这更差的性能虽然可以接受，但并不理想。(3) 期望空间内的性能同样好，而空间之外的性能不可接受，例如“延迟应为500毫秒”。这意味着优于*500*毫秒 的延迟同样好，并且不容忍更差的性能。(4) 偏好最优性能，而期望空间之外的性能不可接受，例如“延迟最多应为500毫秒”，这表示不容忍高于*500*毫秒 的延迟，且延迟越低越好。

我们的第二项贡献 是基于1,296个案例得出的实践性发现，这些发现回答了上述研究问题：

- **针对RQ1：** 在84%的案例中，PS-w的表现显著优于或与PS-w/o相近；其中，超过60%的案例显示出具有统计显著性的改进。
- **针对RQ2：** 与PS-w/o相比，PS-w带来的改进往往很大程度上偏向目标景观中期望空间的某个特定位置，例如居中或向左偏移。
- **针对研究问题3：** 在70%的案例中，PS-w/o的表现不逊于PS-w；其中，超过85%的案例中，两者差异显著且具有统计显著性。
- **针对研究问题4：** 在现实期望下，PS-w在整个轨迹中始终优于PS-w/o，加速比最高可达 10×。相反，当期望不现实时，两种优化模型在调优初期不相上下，但很快PS-w/o便能带来更好的结果，且加速比相当高。

因此，我们推测，性能期望通常确实会影响双目标软件配置调优的指导。不过，具体效果取决于情境，可能有益，也可能有害。作为第三项贡献的一部分，我们深入分析并讨论了上述观察背后的原因。更重要的是，这些发现使我们得以提出第四项贡献：关于双目标软件配置调优中如何在 PS-w 与 PS-w/o 之间进行选择的关键经验教训，具体如下：

- **经验教训 1：** 是否利用期望值来指导调优，主要取决于这些期望值是否现实。
- **经验教训 2：** 模式组合不太可能改变调优时是否纳入期望值这一决策，但可能会影响基于期望值进行调优的益处或弊端。
- **经验教训 3：** 现实期望空间在目标空间中的位置，会极大影响在调优过程中考虑期望值所带来的益处，但不太可能影响选择。

- **第4课：** 给定的调优预算在期望值现实的情况下对选择影响甚微。然而，在期望值不现实的情况下，它可能是一个需要考虑的重要因素。

借鉴这些经验，我们的第五项贡献概述了该研究领域的三个未来机遇，具体如下：

- 可配置软件系统的景观分析。
- 面向配置调优的需求鲁棒优化器。
- 研究需求模式之间对调优的相对影响。

为推动开放科学实践，本研究所需的所有代码、数据集及补充文档均可公开获取：https://github.com/ideas-labo/aspiration-study。

本文其余部分安排如下：第2节对问题进行形式化，并介绍动机示例。第3节讨论用期望值量化性能需求的模式，以及这些模式的识别过程。第4节详细阐述我们的实证研究设计。第5节呈现并分析实验结果。之后，第6节讨论经验教训和未来机遇，第7节则介绍有效性威胁。最后，第8节和第9节分别综述相关工作并总结全文。

2 理论

在本节　　中，我们将介绍理解本研究目的所需的理论知识。

2.1 形式化定义

2.1.1 背景与问题形式化。在 DevOps 时代，软件配置调优涉及两个经常互动的基本角色 [67] ——利益相关者（其利益直接受到软件性能的影响）与软件工程师协商性能需求；随后，软件工程师作为操作人员调优配置，以满足这些需求。除了单一的性能关注点之外，近年来，对多个性能目标的考量需求日益增长 [19, 41]。其中，我们在第3节的文献综述表明，近期工作中有 90% 考虑了两个性能目标 [16, 32, 62]，例如 Storm 的延迟与吞吐量、x264 的图像质量与能耗。这使得以需求为导向的软件配置调优变得更加复杂。

不失一般性，我们假设可配置软件包含一组配置选项，其中第 i 个选项表示为 c_i，它可以是二元变量、整数变量或枚举变量。特定配置表示为 $\boldsymbol{c}$。搜索空间 $\mathcal{C}$ 是所有 c_i 的可能取值的笛卡尔积。形式化地，给定一个对两个性能目标具有明确期望值的需求场景，PS-w 在软件配置调优中的目标是找到能够实现以下目标的配置：

$$maximize\ p_x(f_1(\overline{c})), p_y(f_2(\overline{c})),\ \overline{c} \in \mathcal{C} \tag{1}$$

其中，f 是 $\boldsymbol{c}$ 达到的性能值的原始测量；p 是相应的需求模式，用于量化给定 f（$\boldsymbol{c}$）时的满意化程度（参见第3节）。在本研究中，我们考虑至少有一个 p 包含明确期望水平 [3] 的案例。

相比之下，PS-w/o 的目标是：

$$minimize\ f_1(\overline{c}), f_2(\overline{c}),\ \overline{c} \in \mathcal{C} \tag{2}$$

[3]我们使用 p_x 和 p_y 来区分两种性能需求模式。

4.3	Performance
4.3.1	*The system should support at least 1000 concurrent users.* This statement provides a general sense of reliability when the system is under load. It is important that a substantial number of actors be able to access the system at the same time, since a courseware system is important to the courses that employ it. The times when the system will be under the most stress are likely during midterm and finals weeks. Therefore, it must be able to handle at least 1,000 concurrent users.
	Priority: 1

图 1。PURE 数据集 [29]中一个真实项目需求文档中的性能需求片段。

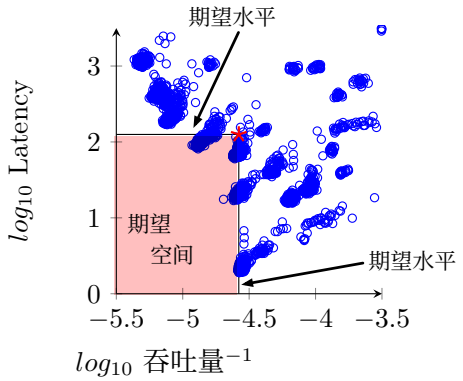


图 2。Rolling Sort 基准测试下 Storm 在双目标空间（延迟和吞吐量）中的期望空间（以颜色突出显示）和期望水平。

可以看出，PS-w 会明确利用给定需求及其清晰期望的信息来指导搜索和调优，而 PS-w/o 则假定基本的帕累托最优性[4]。

2.1.2 期望空间。遵循正常的软件工程需求协商实践，单个性能需求很可能会带有<style id='3'>明确的期望值；在这种情况下，我们将<style id='5'>期望空间定义为不劣于给定期望水平的性能点所构成的区域。图 1 展示了一个真实世界的示例。其中，“系统应至少支持 1,000 个并发用户 ” 包含一个明确的 <style id='7'> 期望水平，即 <style id='9'>1,000 个用户，这意味着期望空间涵盖从 1,000（含）到真实最优值之间的吞吐量（具体范围取决于实际情况）。除这种一维情况外，很容易知道，当涉及两个性能目标的期望水平时，期望空间可以推广到二维情况。例如，图 2 展示了 Storm 的以下需求所对应的期望空间：“系统应能够同时处理 39800 个用户”，且“延迟不得差于 160 秒”（使用对数变换值（log_{10}），并且由于我们考虑吞吐量的倒数，所有性能目标均需最小化）。这构成了下文分析的基础。

2.1.3 使用期望值和不使用期望值的帕累托搜索进行软件调优。为说明 PS-w 与 PS-w/o 的区别，算法 1 展示了以 NSGA-II 为底层优化器的伪代码

[4]我们假设所有性能目标都是需要最小化的；最大化目标也可以轻松转换 ed.

算法 1： 采用 NSGA-II 的 PS-w 和 PS-w/o 统一代码。

输入： 配置空间 V；系统 F；由需求量化的适应度矩阵 Γ
输出： 非支配配置集合 S′

```
1  随机初始化一个由 n 个配置组成的种群 P
2  /* 在实际可配置系统上进行测量                                          */
3  测量(P，F)
4  /* 对于 PS-w，  用于引导搜索的适应度根据公式 (1) 计算                  */
5  如果 PS-w则 Γ ←getFitnessBasedonReqirements(P)
6  当 搜索预算尚未耗尽时
7      P′ = ∅
8      当 𝒫′ < n 执行时
9          /* 对于 PS-w，根据父代对需求的满足程度选择父代 */
10         如果 PS-w 则 {s_x， s_y} ←交配(𝒫 Γ)
11         否则 {s_x， s_y} ←交配(P)
12         {o_x， o_y} ←交叉与变异(V s_x， s_y)
13         测量(o_x， o_y， F)
14         如果PS-w 则 Γ ←getFitnessBasedonReqirements(o_x,o_y)
15         P′ ← P′ ⋃{o_x,o_y}
16     /* 对于 PS-w，  根据计算得出的适应度保留配置，适应度的计算依据是
           相对于需求                                                     */
17     如果PS-w 则U ←nondominatedSorting(P ⋃ P′， Γ)
18     否则 U ←非支配排序(P ⋃ P′)
19     从 U 中选择 P ← 个顶级配置 n
20 如果 PS-w 则返回 S′ ←非支配配置(P Γ)
21 否则返回S′ ←非支配配置(P)
```

。如图所示，PS-w 和 PS-w/o 的主要区别在于，前者受从给定需求和期望值中提取的信息（记为 Γ）引导，而后者不受此信息引导，即使用测得的性能目标原始值。这意味着，PS-w 中评估的所有配置的适应度都会利用 Γ，而 PS-w/o 中配置的适应度则不会。例如，按照原始性能指标，延迟为*500*毫秒 显然比*700*毫秒更受青睐。然而，根据“任何低于*900* 毫秒 的延迟都同样受青睐”这一需求和期望值，两者实际上等同，因此 PS-w 正是体现了这一点。

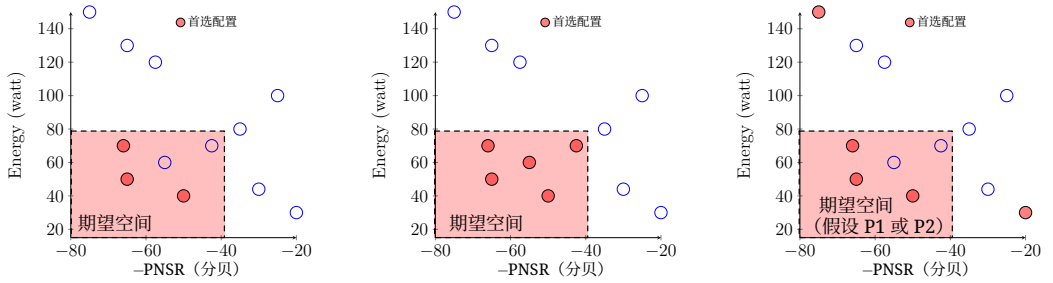
因此，上述做法在 PS-w 和 PS-w/o 之间产生了两点差异。首先，在决定选择哪两个配置作为父代以生成新配置时，两种方法采用了不同的指导方式（即第 10–11 行）。其次，决定下一次迭代保留哪些配置的环境选择，也受到不同适应度的指导（即第 17–18 行）。

正如我们将要展示的，即使只是这样简单的偏差，在不同情况下也可能导致截然不同的领先结果。

2.2 动机示例

以 x264——一种可配置视频编码器——为具体示例，一个可能的**需求场景**可能包含**性能需求**（记为 $\mathcal{P}_1$）：“PNSR[5] 应至少为 40 分贝”以及“能耗应至多为 80 瓦特”。这里，性能属性 PNSR 和能耗分别有明确的**期望水平***40* 分贝和*80* 瓦特，

[5]PNSR 是峰值信噪比的缩写，用于衡量图像的重建质量；PNSR 越大，质量越好。



(a) PS-w with $\mathcal{P}_1$ (b) PS-w with $\mathcal{P}_2$ (c) PS-w/o

图 3. 在不同需求场景下，PS-w/o 和 PS-w 的 x264 首选配置。

具体而言。事实上，视需求场景而定，即使期望水平明确，对性能偏离期望水平的偏好也可能有所不同（我们将在第3节中讨论）。例如，上述示例可能意味着，人们不会接受低于 *40* 分贝或高于 *80* 瓦特的任何性能表现，但会偏好 *PNSR* 和能耗表现更好的任何配置。这意味着，假设有三个配置 $A = \{65dB, 30watts\}$、$B = \{80dB, 25watts\}$ 和$C = \{35dB, 10watts\}$，尽管 C 的能耗表现最佳，但由于未达到 PNSR 的期望值，仍会被排除；在这样的需求场景下，B 肯定更理想，因为它在两个性能目标上的结果都优于 A。在另一种需求场景中，需求（记为 $\mathcal{P}_2$）可能变为“PNSR 不得低于 40 分贝”，同时“能耗不得高于 80 瓦特”，这意味着人们不会接受低于 *40* 分贝或高于 *80* 瓦特的任何性能表现，但对超过 *40* 分贝和 *80* 瓦特的表现一视同仁。在此，C 再次被排除，而 A 和 B 则会同样受偏好，因为它们的 PNSR 和能耗分别优于 40 分贝和 80 瓦特。当然，给定的*40* 分贝和/或*80* 瓦特也可能是不现实期望值，即没有任何配置能够达到这些指标（或者至少在可能的调优预算内找不到这样的配置）。

为了让 PS-w 和 PS-w/o 清楚地理解上述内容，图 3 展示了在 $\mathcal{P}_1$ 或 $\mathcal{P}_2$ 下进行调优时，使用 PS-w 和 PS-w/o 分别会偏好哪些配置。这里，需要根据需求评估生成配置的质量，而 PS-w 恰好偏好其中所需的配置。相比之下，PS-w/o 自然会偏好帕累托前沿上的所有配置。直观来看，我们注意到，PS-w/o 也会偏好一些其 PS-w 对应方法所偏好的配置。例如，比较图 3a 和 3c 时，PS-w 偏好的所有点也都受到 PS-w/o 的偏好（反之则不然），因此二者在 $\mathcal{P}_1$ 下应收敛到相同的可满足性。在图 3b 和 3c 中，尽管 PS-w/o 在期望空间中偏好的点与 PS-w 不同，但由于在通过 $\mathcal{P}_2$ 评估时，期望空间内的所有配置都被视为等价，因此二者应能达到相同的需求满足程度。事实上，如果 PS-w 和 PS-w/o 都能在空间中找到各自偏好的所有点，那么工程师只需从最终返回的配置集合中，根据给定的性能需求挑选出完全满足需求的配置即可。然而，尚未解答的问题是：上述假设是否成立，因此选择 PS-w 还是 PS-w/o 并无区别?

本文其余部分将通过实证研究深入理解上述困惑。

3 需求是如何处理的

在此，我们描述了挖掘、分类和分析包含期望值的现实世界性能需求的过程。我们使用科恩卡帕系数（κ） [60] 来减轻作者之间的偏差——当 $\kappa > 0.7$ 时，这种分类通常被认为是无偏且可持续的。在

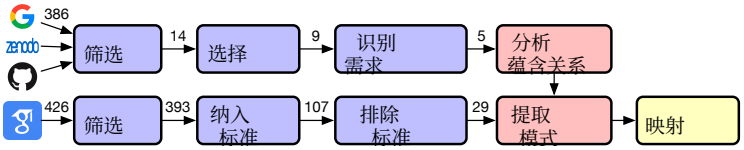


图 4. 数据集分析和文献综述概览。

简而言之，科恩卡帕系数通常被认为比评定者之间简单计算百分比一致率更稳健，因为它考虑了协议偶然发生的可能性。在这项工作中，我们从两个方面使用该系数：

- 衡量对某项需求所属含义类别的判断一致性（我们有$\kappa = 0.85$）。
- 衡量对论文所假设模式的判断一致性（我们有 $\kappa = 0.76$）。

3.1 带有期望的真实世界需求

为了解行业中常见的带有期望的现实世界性能需求及其蕴含关系，2021年1月，我们使用关键词“需求数据集”，从 Zenodo（归类于实证软件工程 标签下）、GitHub 和 Google 数据集搜索中挖掘了公开可用的需求数据集，如图4所示。搜索结果共得到386项，其中包括重复项和许多可通过标题轻松识别的不相关项。因此，我们将候选项筛选至14项，并按照以下标准从中提取与本研究最相关的数据集：

- 该数据集清晰地记录了待构建软件系统的需求陈述。
- 该数据集包含针对性能目标的标注需求，或者有现成的代码可用于完成此项工作。
- 为确保外部效度，该数据集包含来自不同领域系统的性能需求。

这一过程最终筛选出九个数据集。基于这些数据集，我们尝试按照以下规则识别性能需求陈述：

- 性能需求应包含可量化的期望水平，例如“系统应能同时服务1500名用户”。相比之下，“系统应运行得很快”过于模糊，无法量化。
- 为了确保与 PS-w/o 比较时的公平性，我们剔除了那些没有偏向目标某一极端值的性能需求，例如“显示屏应每 60 秒刷新一次”。这是因为这类需求希望性能达到一个明确的期望值（例如 60 秒），而非性能目标的最大值或最小值。因此，在这种情况下，应始终优先选择 PS-w，因为 PS-w/o 会自然地最大化或最小化目标，却不考虑期望值，因而没有使用它的意义[6]。

[6]需要注意的是，在某些案例中，确实可以通过推断上下文得出这类需求所隐含的偏好。沿用同一个例子，如果由于某些长时间运行的分析任务无法在60秒内终止，导致显示内容无法刷新，那么偏好就是确保每60秒或更短时间刷新一次的能力。然而，在我们的案例中，大多数需求都来自 PROMISE 数据集，其中除描述需求的句子外，没有其他额外信息。因此，我们很难准确推断出其中隐含的偏好。因此，在上面的例子中，我们遵循其字面含义：人们偏好且仅偏好60秒的刷新间隔，不多也不少。

表 1. 带有期望值的性能需求。

数据集	# 需求	Link
Do 等人 [23]	52	https://github.com/aqd14/ICSR-2019
PROMISE [61]	48	https://zenodo.org/record/268542
PURE [29]	28	https://zenodo.org/record/1414117
Shaukat 等人 [69]	13	https://zenodo.org/record/1209601
Dalpiaz 等人 [21]	10	https://zenodo.org/record/3309669

表2。采用愿景量化的已识别论文。

会议地点	# 论文数量	会议地点	# 论文	会议地点	# 论文
TSE（期刊）	3	JSS（期刊）	6	TAAS（期刊）	3
ASE（期刊）	2	ESE（期刊）	1	ICPE（会议）	1
ICSE（会议）	1	FSE（会议）	3	ASE（会议）	1
SEAMS（研讨会）	6	ICSA（会议）	1	MODELS（会议）	1

上述原因使我们排除了四个不包含合适需求的数据集。表 1 展示了本研究最终使用的五个数据集的详细信息（已去重）。

3.2 模式文献检索

如图4所示，我们还按照软件工程领域系统性文献综述的最佳实践进行了文献检索 [48], ，其中包含检索协议、纳入标准和排除标准。我们的目标是回答一个问题：正如第3.1节所确定的，现实世界的软件系统中普遍存在的带有期望值的性能需求，其蕴含关系在当前的双目标软件配置调优研究中是如何被具体量化的？请注意，我们的目的并非做到全面，而是收集具有代表性的论文。

2021年2月，我们在 Google Scholar 上对计算机科学领域自2010年以来发表的论文进行了全文检索（为更好地体现该领域的情况，我们排除了与系统相关的论文），使用以下针对性检索字符串：

“需求” AND (“多目标” OR “多目标” OR “多准则”) AND (“性-能” OR “非功能”)
AND (“可配置软件”OR “自适应软件”)
AND (“调优” OR “优化”)

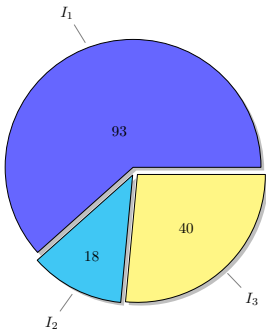
这为我们提供了426篇论文。随后，我们剔除了专利、无法获取的论文以及所有非英文文献，最终得到393篇论文。接下来，我们根据以下标题和摘要纳入标准进一步筛选论文：

- 论文与软件系统配置调优相关。
- 论文旨在提升或评估软件系统的性能目标。
- 论文涉及性能需求。
- 该论文经过同行评审，且不是综述或教程。

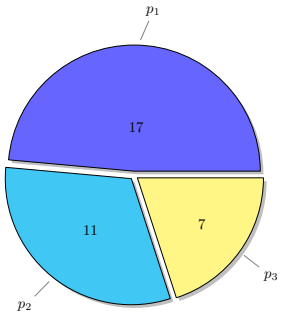
如果一篇论文不符合上述所有标准，则将其排除，共筛得107篇论文。随后，我们通过审阅论文内容，依据以下排除标准进一步剔除论文：

- 所考虑的性能需求没有明确的期望水平。
- 该论文仅处理单一性能目标。
- 该论文没有提供定量实验结果，也未清楚说明如何获得这些结果。

如果一篇论文符合上述任一标准，就会被排除。最终，我们筛选出29篇论文，如表2所示。



（a）每个蕴含关系的需求数



（b）每个模式的论文数

模式蕴含关系	
p_1	I_1, I_3
p_2	I_1, I_2, I_3
p_3	I_2, I_3

(c) 映射

图5. 蕴含关系、模式及其映射的分布（六篇论文考虑了不止一种模式）。

3.3 结果分析

3.3.1 性能目标的数量。综述发现，29篇论文中有26篇（90%）在调优过程中考虑了两个性能目标。其余三篇则考虑了三个或更多目标。这清楚地表明，两个性能目标仍是软件配置调优中最先进的设置，这与近期一项相关领域综述的发现一致 [19]。因此，本研究聚焦于双目标软件配置调优。

3.3.2 蕴含关系。我们分析了第3.1节中的全部151项带有期望值的性能需求，并发现对于给定的性能目标，期望空间可能存在三种蕴含关系：

- I_1：**期望空间中的任何内容都同等可取**。在没有其他信息时，这给出了明确的期望上限，例如“服务器将支持最多1,000名并发用户”；或者存在一个期望下限，但已明确说明性能达到期望空间时的情形，例如“结果应在15秒内返回”。

- I_2：**不在期望空间内的任何值都同样不受偏好**。例如，“系统应允许至少有6名用户同时使用”，在这种情况下，仅包含关于明确的下限期望界限的信息。

- I_3：**没有关于期望空间的信息可用**。这通常指具有明确期望水平的需求，但没有迹象表明该水平是上限期望界限还是下限期望界限，同时也没有其他可用信息。例如，“系统应满足10名并发用户的需求”。

蕴含关系的分布可见图5a，我们为此获得了卡帕系数$\kappa = 0.85$。

3.3.3 模式。接下来，考虑到上述蕴含关系，我们试图了解这些蕴含关系在所识别的29篇论文中是如何量化的。由此，我们归纳出三种最先进的函数模式，用于量化带有期望水平的需求（假设下界为最优值）。设 α 和 β 分别表示性能目标的下界和上界； d 为期望水平。图6展示了这些模式及其量化方式，具体说明如下：

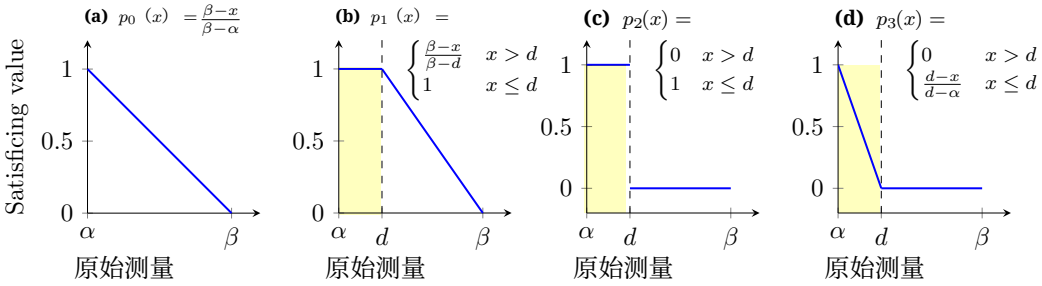


图6. 文献中带有（及不带）期望值的需求模式。 α 和 β 分别表示性能目标的下界和上界。 d 为期望水平，期望空间以阴影标示。

- p_1：期望空间内的性能同样出色，否则则存在一定程度的容忍度（图6b）。该函数可表示为：

$$p_1(x) = \begin{cases} \frac{\beta - x}{\beta - d} & x > d \\ 1 & x \le d \end{cases} \tag{3}$$

- p_2：期望空间内的性能均同样良好，而空间之外的任何表现都不可接受（图 6c），因此：

$$p_2(x) = \begin{cases} 0 & x > d \\ 1 & x \le d \end{cases} \tag{4}$$

- p_3：优先选择最优性能，而期望空间之外的任何表现都不可接受（图 6d），其定义如下：

$$p_3(x) = \begin{cases} 0 & x > d \\ \frac{d - x}{d - \alpha} & x \le d \end{cases} \tag{5}$$

同样地，我们也可以将不涉及明确期望水平的需求（例如“延迟应当较小”）形式化，记为 p_0，如图 6a 所示，其形式化如下：

$$p_0(x) = \frac{\beta - x}{\beta - \alpha} \tag{6}$$

这些模式的分布如图 5b 所示，其中我们有 $\kappa = 0.76$，这是可持续的 [60]。

通过对这些模式进行归一化，性能目标的原始测量值会转化为相对于给定期望空间（如果有）的满意度，取值范围为 0 到 1，其中 1 表示完全满足。因此，这种转化取决于对期望空间所包含或排除的测量值进行满意化的假设，而这一区别正是各模式不同之处。此时，我们可以立即看出这些模式与从真实世界数据集中提取的蕴含关系之间的映射。这些映射已在图 5c 中展示；从中可以看出，除 p_0外，每种模式都至少能与真实世界需求中的两种蕴含关系相匹配。例如， p_1 可以与 I_1 和 I_3，相匹配，因为前者偏好期望空间内的任何内容，但对另一端没有任何规定；而后者完全没有提供信息，因此在量化 I_3 以指导搜索时，需要依赖某种假设，这意味着它有可能与这三种模式都相匹配。

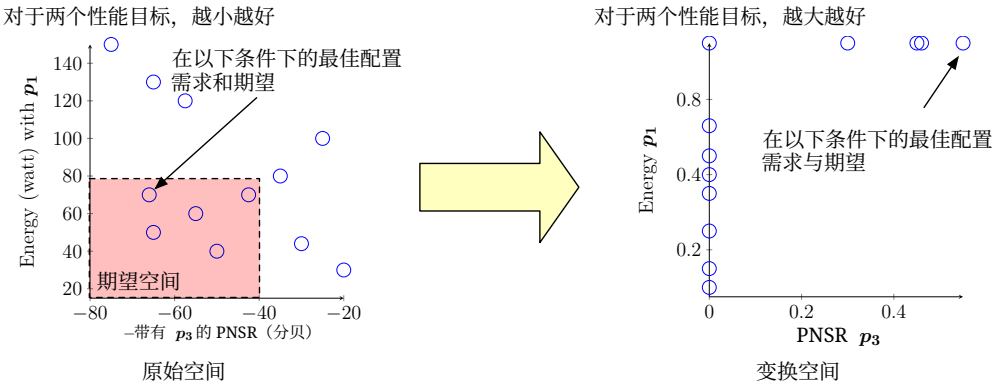


图 7. 使用这些模式评估配置的优劣并在变换空间中引导 PS-w 的示例。

由此可确认，当前工作中的确存在能够反映现实世界性能需求及其期望所蕴含关系的模式。因此，我们将在实证研究中考察所有这些模式。

3.4 软件配置调优中的需求与期望

虽然上述需求模式是评估任何帕累托优化器和优化模型所生成的配置集合中哪些配置“更好”或“更差”的关键，但它们会直接影响 PS-w 的行为（它们对应于公式 (1) 中的 p_n），却不会影响 PS-w/o 的行为。

最重要的是，这些模式使我们能够在给定一组需求和期望的情况下，精确量化这两种优化模型所生成的配置中，哪些配置是最优的。图 7 展示了一个示例：在考虑需求和期望时，也就是能耗的 p_1 和 80 瓦特的期望值，以及 PNSR 的 p_3 和 40 分贝的期望值，如何在变换空间中评估配置（并引导 PS-w）。在此，我们显然更偏好期望空间内的点，而非空间外的点。然而，对于期望空间内的点，我们只偏好 PNSR 更好的点，而能耗则视为相同（这是由于 p_1 和 p_3 的蕴含关系）。

在原始空间中（图7左侧），上述情况难以评估和量化，因为当不涉及需求和期望值时，彼此互不支配的点（就原始目标值而言）自然会被视为等价。因此，实际最受偏好的点（箭头所示）并未被视为最佳点。相比之下，在变换空间中，最佳点一目了然；其中，能耗和PNSR分别通过针对 p_1 和 p_3的方程进行转换。现在，很明显，最受偏好的点是该空间中唯一的非支配点（图7右侧）。

4 实证研究设计

如图8所示，我们的方法包括以下步骤：

步骤1： 假设软件工程师和利益相关者已经就一个需求场景达成一致，我们将需求量化，使其适用于帕累托优化器，即将其表示为第3节中的模式及其期望空间的组合形式。具体而言，为构成一个需求场景，将给定的模式组合表示为二维向量 $\mathcal{P}$，其中至少有一个模式具有明确的期望水平，例如 $\mathcal{P} = \{p_0,\ p_3\}$。在本研究中，我们考察模式的所有可能组合（包括 p_0）。对于每种组合，我们

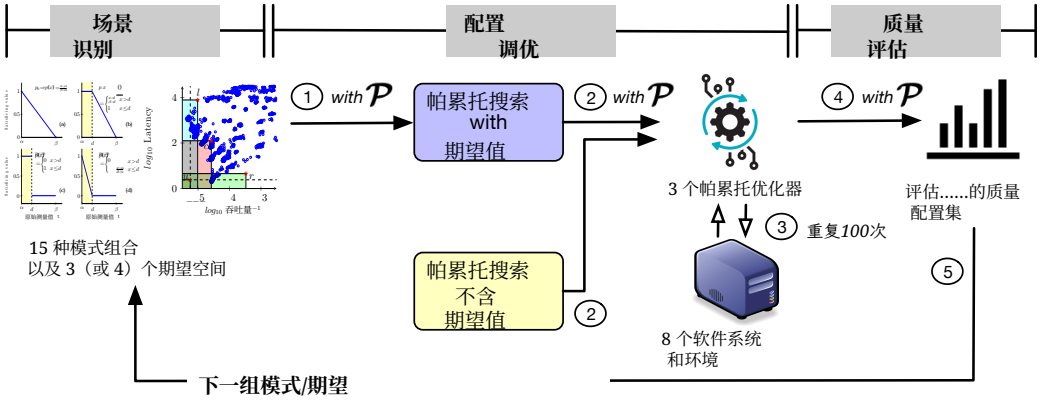


图8. 实证研究概览。

还会针对研究问题考虑不同的期望空间；这将在第4.2节中进一步阐述。

步骤2： 在不同的软件系统上分别运行 PS-w/o 和 PS-w。具体而言，在制定性能目标时，PS-w/o 仅由原始测量值引导[7]，而 PS-w 则旨在由给定的模式向量 $\mathcal{P}$ 作为新目标来引导。为确保公平，我们使用相同的优化器来评估这两种优化模型，并在本研究中考虑三种具有代表性的优化器，即 NSGA-II [22], 、IBEA [85],和 MOEA/D [82]。**步骤3：** 随着搜索的进行测量系统，直至调优预算耗尽；重复100次。**步骤4：** 随后使用 $\mathcal{P}$ 评估配置集，作为质量评估阶段的一部分。**步骤5：** 如果还有更多模式组合和期望需要考察，则返回步骤1。

值得注意的是，尽管第3节中的模式针对单一性能目标，但在步骤 1 [8, 9, 33, 59]，的场景识别 阶段，可以像表3所示那样任意组合这些模式，用于双目标软件配置调优。

在配置调优 阶段（步骤 2 和 3）中，这些模式需要使用下界和/或上界进行归一化（ $\boldsymbol{p_0}$ 和 $\boldsymbol{p_2}$ 除外）。然而，由于这些界限通常未知，我们采用一种动态方法：随着调优的进行，使用迄今为止发现的最大值和最小值对原始测量值进行归一化。这是基于搜索的软件工程中进行软件配置调优的常见做法 [7, 68]。我们会在整个调优过程中记录每种配置的原始测量值，以便高效利用调优预算（第4.3.2节）。

为减少随机偏差，我们将每项实验重复运行 100 次。研究在一个计算机集群上开展，集群中的每台机器均配备 Intel i5 六核中央处理器，主频为 2.9GHz，内存为 8GB，并在五个月内并行运行了大量实验（24×7）。

4.1 研究对象：软件系统

我们在一组现实世界中的高度可配置软件系统和环境上开展研究，这些系统和环境已在现有研究中得到广泛探讨 [17, 44, 45, 62]。这些研究对象根据以下标准选定：

(1) 为确保搜索空间并非过于简单、以至于无法进行探索，系统应包含二元型和枚举型配置选项。

[7]这实际上等同于对所有性能目标使用 $\boldsymbol{p_0}$。

表3。所考虑的需求场景（根据第3节中识别的模式组合）及其示例解读。解读基于以下假设：性能目标为 $\{latency,\ throughput\}$，其可能的期望水平分别为 d_1 和 d_2。

可能的 $\mathcal{P}$	示例解读
$\{p_0,\ p_1\}$	优先选择延迟更低且吞吐量优于 d_2的配置，但所有优于 d_2 的配置都同样优先选择；愿意接受吞吐量差于 d_2的配置。
$\{p_1,\ p_0\}$	更倾向于选择延迟和吞吐量优于 d_1的配置，但任何优于 d_1 的配置都同样优先选择；愿意接受延迟差于 d_1的配置。
$\{p_0,\ p_2\}$	更倾向于选择延迟和吞吐量优于 d_2的配置，但任何优于 d_2 的配置都同样优先选择；不接受吞吐量低于 d_2的配置。
$\{p_2,\ p_0\}$	优先选择吞吐量更高且延迟更低的配置，但任何优于 d_1 的配置都同样优先选择；不接受延迟高于 d_1的配置。
$\{p_0,\ p_3\}$	优先选择延迟更低且吞吐量更高的配置；不接受吞吐量低于 d_2的配置。
$\{p_3,\ p_0\}$	优先考虑更低的延迟和更高的吞吐量；不接受延迟差于 d_1的情况。
$\{p_1,\ p_1\}$	优先考虑延迟优于 d_1 且吞吐量优于 d_2的配置，但任何优于 d_1 且 d_2 的配置优先级相同；愿意接受延迟和吞吐量分别差于 d_1 和 d_2的情况。
$\{p_2,\ p_2\}$	倾向于选择优于 d_1 的延迟和优于 d_2的吞吐量，但任何优于 d_1 的配置以及 d_2 均同等受偏好；不接受延迟和吞吐量分别劣于 d_1 和 d_2的配置。
$\{p_3,\ p_3\}$	倾向于选择更优的延迟和吞吐量；不接受延迟和吞吐量劣于 d_1 和 d_2的配置，分别。
$\{p_1,\ p_2\}$	更偏好延迟优于 d_1 且吞吐量优于 d_2的配置，但任何优于 d_1 且 d_2 的配置都同样受偏好；愿意接受延迟差于 d_1，但不接受吞吐量更差于 d_2的配置。
$\{p_2,\ p_1\}$	更偏好延迟优于 d_1 且吞吐量优于 d_2的配置，但任何优于 d_1 且 d_2 的配置都同样受偏好；愿意接受吞吐量差于 d_2，但不接受延迟更差优于 d_1。
$\{p_1,\ p_3\}$	更倾向于吞吐量更高且延迟优于 d_1的配置，但任何优于 d_1 的配置都同样更受青睐；愿意接受延迟劣于 d_1 的配置，但不接受吞吐量劣于 d_2的配置。
$\{p_3,\ p_1\}$	更倾向于延迟更低且吞吐量优于 d_2的配置，但任何优于 d_2 的配置都同样更受青睐；愿意接受吞吐量低于 d_2，但不能接受延迟高于 d_1。
$\{p_2,\ p_3\}$	优先选择更高的吞吐量，且延迟优于 d_1；但所有优于 d_1 的配置都同样受更受青睐；不能接受延迟高于 d_1，吞吐量也不能低于 d_2。
$\{p_3,\ p_2\}$	优先选择更低的延迟，且吞吐量优于 d_2；但所有优于 d_2 的配置都同样受首选；不接受延迟低于 d_1 或吞吐量低于 d_2。

(2) 无法对搜索空间进行完整探索，也就是说，这项工作无法在 24 小时内完成。(3) 关于如何设置用于测量系统的基准测试，有明确的说明。(4) 如果同一环境中的同一系统曾使用另一组配置选项，则选择复杂度相对较高的选项，即搜索空间更大、配置选项更多的选项。例如，Storm 可在不同的工作负载基准测试下进行调优，我们选择 WordCount 和 RollingSort 作为符合上述标准的两个基准测试。

我们首先从 [62], 中剔除了 LLVM，因为它违反了标准（1）。同样，sort-256 和 noc-CM-log 也被排除在外，因为它们的搜索空间相当小，可在 24 小时内穷举探索，这违反了标准（2）。由于没有明确说明应使用哪种基准测试对 SaC 进行分析，因此我们无法将其纳入考量，这违反了标准（3）。我们还注意到，Storm 和 Keras（搭配深度神经网络或长短期记忆网络）的使用远比其他系统普遍，但其配置选项和环境各不相同。因此，根据标准（4），我们采用能够形成更大搜索空间且提供更多选项的设置。如表4所示，所选软件系统涵盖多个不同领域，例如视频编码、流处理以及深度学习/机器学习，同时具有不同的性能目标、规模和搜索空间。

表4. 研究的可配置软件系统。我们在各系统的标准基准测试下运行所有软件系统。Storm 和 Keras（使用深度神经网络）分别使用两个基准测试和三个数据集。

软件	领域	性能目标	# 选项	搜索空间	使用者
Trimesh	网格求解器	延迟和迭代次数	13	239,260	[17, 62]
x264	视频编码	峰值信噪比和能耗	17	53,662	[17, 62]
Storm/WC	流处理	延迟和吞吐量	6	2,880	[17, 44, 45, 62]
Storm/RS	流处理	延迟和吞吐量	6	3,839	[17, 44, 45, 62]
Keras/Adiac	深度学习	曲线下面积和推理时间	13	3.99×10^{13}	[45]
Keras/DSR	深度学习	曲线下面积和推理时间	13	3.32×10^{13}	[17, 45]
Keras/SA	深度学习	曲线下面积和推理时间	13	2.66×10^{13}	[45]
XGBoost	机器学习	准确率和训练时间	13	2.88×10^{10}	[45]

它们的测量成本也很高[8], , 例如，XGBoost 需要 2,807 小时才能探索不到 1% 的搜索空间。

我们沿用先前研究中采用的性能目标、配置选项及其取值范围，例如 [17, 44, 45, 62],, 因为这些已被证明是相关环境下这些软件系统的关键因素。因此，尽管软件系统相同，实际的搜索空间却有所不同，例如 Storm/WC 和 Storm/RS。具体而言，我们沿用先前研究中的做法，考虑的环境/工作负载如下：

- **Trimesh：** 我们使用包含 51,300 个独特 3D 模型的 Shapenet 数据集。在本研究中，我们随机抽取 100 个模型作为标准基准。
- **x264：** 对此，采用的基准测试是一个大小为 1 GB 的标准视频，该视频是随机选取的。
- **Storm/WC：** 我们使用 WordCount 作为基准测试。这是一个典型的简易流式处理示例，Storm 用于跟踪流入的单词及其计数。WordCount 会产生 CPU 密集型工作负载。
- **Storm/RS：** 与 Storm/WC 类似，这里我们使用 RollingSort 作为基准测试。与 WordCount 不同，RollingSort 会产生内存密集型工作负载。
- **Keras/Adiac：** 我们使用 Keras 软件中的深度神经网络（DNN），并在 Adiac 数据集上运行。通常，该数据集包含一项硅藻自动识别任务，目标是从 31 个类别中识别硅藻（单细胞藻类）；训练集和测试集的样本量分别为 390 和 391。
- **Keras/DSR：** 我们使用 Keras 软件中的深度神经网络，并在 DiatomSizeReduction 数据集上运行。该数据集用于预测四种硅藻，训练集和测试集的样本量分别为 16 和 306。
- **Keras/SA：** 我们使用 Keras 软件中的深度神经网络，并在 ShapesAll 数据集上运行。该数据集通常用于测试轮廓/图像和基于骨架的描述符；共有 60 个类别，训练集和测试集的样本量各为 600。
- **XGBoost：** 我们使用 Covertype 数据集，该数据集仅包含根据制图变量划分的 54 种森林覆盖类型。数据集大小为 581,012，我们采用 70%-30% 的训练集与测试集划分。

事实上，第3节中分析的数据集和文献可能并非专门针对本研究所考虑的软件系统。不过，提取出的蕴含关系和模式相当通用，因此可应用于不同案例。此外，一些受到广泛研究的性能目标（来自数据集和文献）具有极强的普适性。例如，延迟和

[8]每次测量包含 5 个重复样本，并使用其中位数值。

表 5. 所研究的可配置软件系统的期望水平和空间（适用于所有模式组合）。 l、 r、 c 和 u 分别表示左移、右移、居中和不现实的期望。

软件	性能目标	l	r	c	u
Trimesh	{延迟（秒），迭代次数}	{81，4}	{461，15}	{135，7}	{37，501}
x264	{峰值信噪比（分贝），能耗（W）}	{50，3680}	{37，462}	{46，1260}	{100，34}
Storm/WC	{吞吐量（消息/分钟），延迟（毫秒）}	{16473，15677}	{994，5}	{8982，101}	{34740，3}
Storm/RS	{吞吐量（消息/分钟），延迟（毫秒）}	{1.	{3006，5}	$\{3.7 \times 10^4, 126\}$	{
Keras/Adiac	{曲线下面积，推理时间（毫秒）}	{0.030, 44}	{0。017，0。05}	{0.028, 3}	{0。292，0。03}
Keras/DSR	{曲线下面积，推理时间（毫秒）}	{0。307，123}	{0.107, 0.12}	{0.300, 25}	{0。581，0。031}
Keras/SA	{曲线下面积，推理时间（毫秒）}	{0。167，21}	{0.157, 0.07}	{0。160，6}	{0。325，0。04}
XGBoost	{准确率（%），训练时间（秒）}	{80, 42}	{54, 3}	{72，8}	{92、1}

吞吐量相关需求（具有不同的期望水平）广泛存在于各类软件 [62] 中。

4.2 期望空间

为了提高外部效度，我们考虑在两个性能目标下形成两类期望空间的期望水平：现实的和不现实的。为此，对于每个软件系统，我们分别运行所有帕累托优化器三小时，以获得包含近似帕累托前沿的景观。我们通过确保获得的前沿已合理收敛来实现这一点，即增加预算只会使结果发生微小变化。随后，我们根据这样的前沿设定期望空间，如表 5 所示。

4.2.1 现实期望空间。对于具有两个性能目标的软件配置调优，如果至少存在一个配置能够同时达到两个性能目标的期望水平，我们就称该期望空间是现实的。

以图 9 中的 Storm/RS 为例，对于每种模式组合下的现实期望空间，我们根据其在目标空间中的位置设定三个期望空间：左移（l）、右移（r）和居中（c）。具体而言， l将吞吐量的 $20th$ 百分位数值和延迟的 $80th$ 百分位数值分别作为对应的期望水平；同理， r将延迟的 $20th$ 百分位数值和吞吐量的 $80th$ 百分位数值分别作为对应的期望水平；最后， c则将两个性能目标各自的 $50th$ 百分位数值作为期望水平。显然，尽管这些空间覆盖了性能目标整体空间中的不同区域，但每个空间至少包含一个点（配置）。请注意，只要至少有一个性能目标具有明确的期望水平，这种期望空间就适用于有 $\boldsymbol{p_0}$ 和没有 $\boldsymbol{p_0}$ 的任意模式组合（分别见图 9a 和图 9b）。

4.2.2 不现实期望空间。 由于期望水平/空间是事先协商确定的，因此可能并不现实。在本研究中，我们将不现实期望空间定义为这样一种情况：两个性能目标的期望水平最多只能分别达到，但无法同时达到。例如，对于图 9a 中的两个性能目标， u构成一个不现实期望空间：如虚线所示，两个目标各自都能达到相应水平，但无法同时达到，因为该空间内不存在任何点（配置）。因此，当只有一个性能目标包含明确的期望值时，这一定义并不适用，例如图 9b。为定义这样的空间，我们将两个性能目标的 $5th$ 百分位数值设为相应的期望水平；我们发现，这足以为每个系统构造一个不现实期望空间。

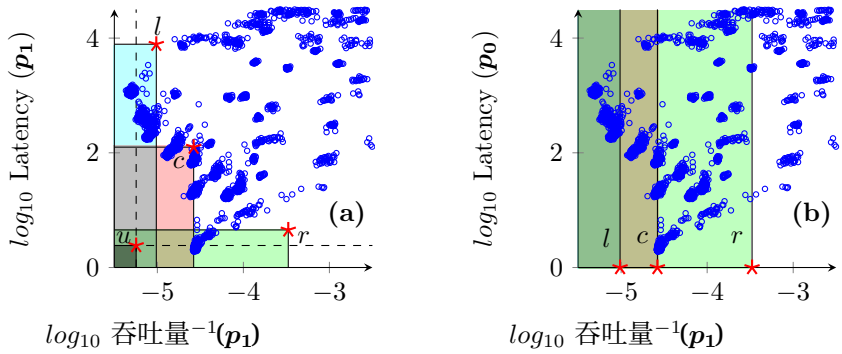


图9. Storm/RS 在不同模式组合下的远期期望空间（以不同颜色着色）。

表6. 种群规模和测量调优预算。

软件	种群规模	# 测量	软件	种群规模	# 测量
Trimesh	10	500	x264	50	1,500
Storm/WC	50	500	Storm/RS	30	700
Keras/Adiac	50	700	Keras/DSR	60	500
Keras/SA	60	500	XGBoost	30	300

4.3 调优设置

4.3.1 帕累托优化器。 我们考虑三种帕累托优化器，即 NSGA-II [22], IBEA [85], 和 MOEA/D [82], , 原因如下:

- 先前研究中已广泛将它们用于软件配置调优 [8, 9, 16, 33, 59,70]。
- 它们代表了三种本质上不同的帕累托搜索框架 [26]
- 它们可以依赖代理模型，但并非必须如此 [16],, 这大大降低了实证研究中的噪声。

基于 jMetal 中的实现 [25]，上述所有优化器均用于 PS-w 和 PS-w/o。

4.3.2 调优预算。 在本研究中，我们为每次运行设置了一小时的预算，这也是处理高成本基于搜索的软件工程（SBSE）问题时的常见做法 [53]。然而，直接以时间作为终止条件，在调优过程中可能受到严重干扰，因为需要并行运行大量实验。为避免这种情况，对于每个软件系统，我们采用以下方法将一小时的调优预算转换为唯一测量的数量:

（1）以每次增加100个配置的方式，在专用机器上通过随机抽样逐步测量不同的配置，直至用尽一小时的时间预算。（2）将上述过程重复5次，并记录测量数量。（3）这5次重复的中位数，作为之后调优的关键终止准则（见表6）。

请注意，在每次调优运行中，我们都会缓存每个不同配置的测量结果，以便直接复用。因此，只有不同的配置会消耗预算。

4.3.3 参数。对于所有案例中的三种优化器，我们均采用先前研究 [16, 20] 使用的二元锦标赛选择、边界变异和均匀交叉。变异率和交叉率分别设为 0.1 和 0.9，这也符合软件

配置调优 [16, 20] 中最常见的设置。IBEA 和 MOEA/D 的其他具体设置均保留默认值，已有研究表明这些设置有效 [16]。

对于每个系统，我们通过以下方式务实地设置种群规模：

（1）在表6所示的预算下，通过试运行考察不同的规模，即在所有优化器、模式组合和期望值设置下（同时针对 PS-w 和 PS-w/o）测试{10、20、.....、100}。（2）使用 $g = \frac{1}{k} \times \sum_{i=0}^{k} \frac{c_i}{s}$ 记录最后10%代数中种群的平均变化率，其中 k 是最后10%代数的数量； c_i 表示第 i 代中与第 i-1代相比的不同配置数量； s 是种群规模。（3）在所有条件下，满足 $g \leq 0.1$ 的最大种群规模将被采用（如果没有规模满足上述约束条件，则采用10）。

结果也显示在表6中。通过这种方式，我们力求在给定的调优预算下，在收敛性（较小的种群变化）和多样性（较大的种群规模）之间取得平衡。也就是说，增加预算不太可能改变结果。这一点已在 [17, 32]中得到实践。

4.4 分析与比较

4.4.1 度量指标。 为了进行比较并确定在本研究中哪种优化模型更好，我们需要根据以下两个条件来衡量“最佳”：

- **条件 1：** 该度量指标需要能够全面比较帕累托优化器生成的不同配置集，涵盖收敛性和多样性等多个质量方面。

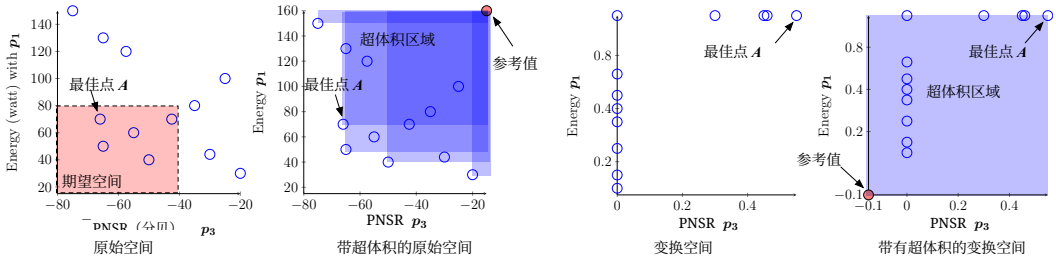
- **条件 2：** 该度量指标应能够反映给定的需求场景，即在进行比较和评估时考虑第3节中识别出的给定模式。

为此，我们采用超体积（HV） [84, 86] 作为基本度量指标，评估每次运行所生成配置集的质量。简而言之，HV 衡量配置集中所有点与一个参考值（通常为最低点）之间的体积；体积越大，该配置集实现的收敛性和多样性就越好。本研究选择 HV 的原因如下：

- HV 是一项综合指标，涵盖配置集的所有质量方面，即收敛、均匀性、分布范围和基数 [55, 57],，符合**条件 1**。
- HV 也不需要参考帕累托前沿，并且具有帕累托一致性[9]；由于真实的帕累托前沿未知，这一特性适用于我们的情况。
- 遵循 [54, 55], 提出的指南后，我们确定 HV 是适用于我们基于搜索的软件工程问题的指标。

由于我们关注的是目标空间中具有特定模式组合和期望空间（$\mathcal{P}$）的需求场景，因此，始终偏向于靠近整个帕累托前沿的配置的原始超体积（HV）已不再适用。因此，我们需要按照 Li 等人 [55] 的指导，并利用第3节中的模式和量化方法（以满足**条件2**），将这些偏好转换到 HV 中。以第2.2节中的同一示例为例，如图10所示，该需求场景为：利益相关者偏好更高的 *PNSR*，并偏好能耗低于 *80* 瓦特；但对于能耗低于 *80* 瓦特的任何配置，他们都一视同仁；他们愿意接受能耗高于 *80* 瓦特，但不接受 *PNSR* 低于 *40* 分贝。这意味着，对于期望空间中的任意点，PNSR 更高的点会更受偏好。

[9]一般来说，质量指标具有帕累托一致性，意味着其评估结果不会与两个解集之间的帕累托支配关系相冲突。更严格地说，如果解集 A 比 [87] B 更优 （即，对于 B中的任意解，都存在 A 中的一个解能够覆盖它（支配它或与之等价），并且 A 中至少有一个解不被 B中的任何解覆盖），那么该指标对 A 的评估结果始终优于对 B 的评估结果。



（a）在原始空间中使用超体积进行评估　　（b）在变换空间中使用超体积进行评估

图 10. 有无需求/期望时对超体积的评估。

因此，根据需求，点 A是最佳点，应对所选指标贡献最大。然而，直接应用 HV 会导致某些不太符合需求偏好的配置对 HV 值产生显著贡献（图10a）。这会造成误导，使包含许多非偏好点的某些集合看起来具有非常高的 HV 值。相比之下，在使用 HV 之前转换模式信息（即在变换空间中）后，就能更好地满足上述需求和期望；显然， A对 HV 的贡献最大，而其他非偏好点往往没有贡献或贡献很小。（图10b）

为此，我们在本文中扩展了超体积。假设有 m 个性能目标（本文中有 $m = 2$ 个），且 $\mathcal{A}$ 是一个生成的配置集，其中某个配置的原始测量值向量为 $\boldsymbol{x_i} = \{x_1, \ x_2, \ ...x_m\}$，我们根据给定的 $\mathcal{P}$，基于$\boldsymbol{x_i}$转换后的满意化值计算超体积。我们将其称为愿望感知超体积（简称 A-HV ），其公式如下：

$$A\text{-}HV(\mathcal{A}) = \lambda(\bigcup_{\overline{x}_i \in \mathcal{A}} \{\boldsymbol{v} | \mathcal{P}(\overline{x}_i) \prec \boldsymbol{v} \prec \boldsymbol{r}\}) \tag{7}$$

其中， λ 是用于量化体积 [86] 的勒贝格测度，与原始 HV 中的定义相同； $\boldsymbol{r}$ 是参考最差点，通常取为非支配集 [55], 各维度取值范围的 1.1 倍，因此在我们的案例中，该点为 $\{-0.1, -0.1\}$，因为 $\mathcal{P}(\boldsymbol{x_i})$会将输出转换为 $[0, 1]$。与 HV 一样，A-HV 值越高越好。为确保与 A-HV 的比较公平，我们使用所有实验中各性能目标的最小值和/或最大值，对各模式进行后验归一化。

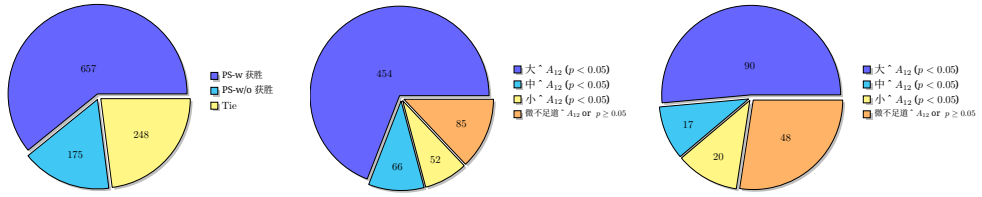
为便于直观说明，我们报告在每次运行中，调优时考虑需求和期望所得的 A-HV（即 PS-w）相较于 PS-w/o 的增益百分比，定义如下：

$$\%\ \text{Gain} = \frac{x_i - y_i}{y_i} \times 100 \tag{8}$$

其中， x_i 和 y_i 分别是 PS-w 和 PS-w/o 在各自排序列表中第 i次运行所得的 A-HV 值。显然，增益百分比为正表示期望有帮助（PS-w 表现更好），而负值则表示期望有害（PS-w/o 表现更好）；增益为零表示结果相同。

4.4.2 统计验证。 我们使用标准方法来解释每种情况下 100 次运行的结果显著性 [1, 47]:

- **Wilcoxon 检验：** 我们使用 Wilcoxon 检验 [76] with $a = 0.05$ [1] 来考察全部 100 次运行中 A-HV 比较的统计显著性。该检验是一种非参数统计检验，对数据分布的假设较少，并且已被推荐用于软件工程研究中的成对比较 [1]。



(a) Distribution on all 1,080 cases (b) Distribution on the 657 cases that `PS-w` win (c) Distribution on the 175 cases that `PS-w/o` win

图 11. PS-w 和 PS-w/o 获胜情况的总结，以及详细的统计验证结果。

- **ˆA12 效应量：** 为确保 a $p < 0.05$ 并非由样本量过小导致，我们采用 ˆ A_{12} [72]来衡量效应量。在本研究中，ˆ $A_{12} > 0.5$ 表示 PS-w 获胜，即其 A-HV 在超过 50% 的运行中更优。 $A_{12} \geq 0.6$ 或 $A_{12} \leq 0.4$ 表示效应量并非微不足道。由于运行次数为 100 次（而非通常采用的 30 次），我们采用更严格的 ˆ ˆ ˆ ˆ 判定标准： $0.6 \leq A_{12} < 0.7$ ($0.3 < A_{12} \leq 0.4$)、$0.7 \leq A_{12} < 0.8$ ($0.2 < A_{12} \leq 0.3$) 和 $A_{12} \geq 0.8$ ($A_{12} \leq 0.2$)分别表示小、中和大效应。

5 结果与发现

本节将介绍实证研究的结果，并回答第 1 节提出的研究问题。

5.1 RQ1：在现实期望下，哪种方法更符合需求?

5.1.1 方法。 为回答 **RQ1**，我们在 15 种模式组合、三个现实期望空间（l、 r 和 c）、三个优化器和八个被测系统上比较 PS-w 与 PS-w/o，共计$15 \times 3 \times 3 \times 8 = 1{,}080$ 个案例。由于我们关注每个案例下 A-HV 的成对比较，因此使用 Wilcoxon 检验和 A_{12} 来验证 100 次运行结果的统计显著性。

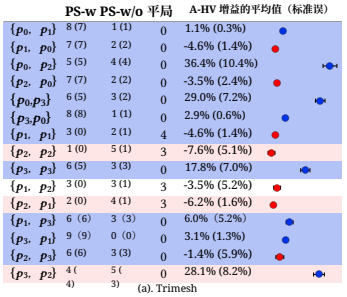
ˆ*5.1.2* 结果。 总体而言，图 11 汇总了各案例的 A_{12} 结果。显然，PS-w 的表现远胜于 PS-w/o。具体而言，PS-w 在 61%（657/1080）的案例中获胜，在 16%（175/1080）的案例中落败，另有 23%（248/1080）的案例打平。换言之，与使用 PS-w/o 时的 39% ˆ（423/1080）相比，PS-w 在 84%（905/1080）的案例中表现更好或相当。从统计角度看，PS-w 在 $A_{12} \geq 0.6$ 和 $p < 0.05$条件下赢得了 572 个案例，而落败时只有 127 个案例具有统计显著性。

为更全面地了解不同系统和需求场景，我们从表7中可以看到，PS-w 总体表现明显更好：在大多数案例中都取得了相当可观的正增益（A-HV 的平均提升最高达 145%），且统计显著的获胜次数通常更多。值得注意的是，我们观察到 PS-w 在 Storm 下的增益尤其高（表7c 和表 7d）。这归因于该系统不同配置之间性能差异很大，这一点先前研究中也有报告 [17,44, 62]。

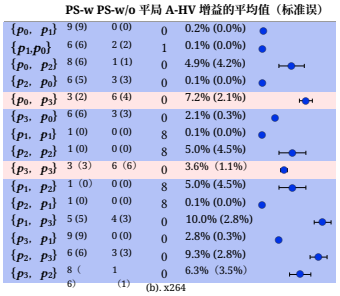
令人欣喜的是，PS-w 的优势在给定的需求模式中始终如一——这清楚地表明，需求能够为调优提供重要指导。然而，正如预期的那样，当场景仅要求 $\boldsymbol{p_1}$ 或 $\boldsymbol{p_0}$ 时，PS-w 和 PS-w/o 的表现大多相同（或非常相近）。至于 PS-w 劣于 PS-w/o 的少数案例，结果可能是由偶然遇到的局部最优问题造成的，我们将在下文中进一步讨论。

因此，我们得出：

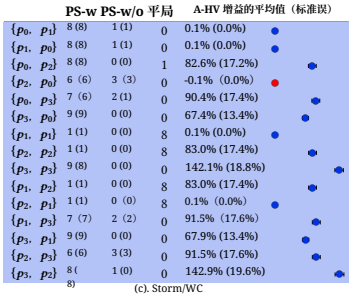
表7. 在现实需求和期望值下，运行100次，对比PS-w 与PS-w/o 。● 和● 分别表示正向和负向增益百分比的平均值（Avg）和标准误（SE）。 ○表示总体增益为零。列“PS-w”和“PS-w/o”显示相应优化模型获胜的案例数。9（6）表示其中一个模型在9个案例中获胜，其中6个案例具有统计显著性，即$\hat{A}_{12} \geq 0.6$ （或$\hat{A}_{12} \leq 0.4$）以及 $p < 0.05$ （每种需求模式组合总计有9个案例，因为共有3种期望空间和3种优化器）。蓝色单元格表示PS-w 获胜次数更多，而红色单元格表示其失利次数更多。



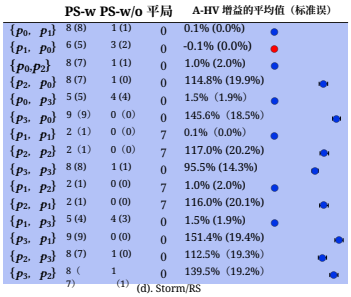
(a). Trimesh



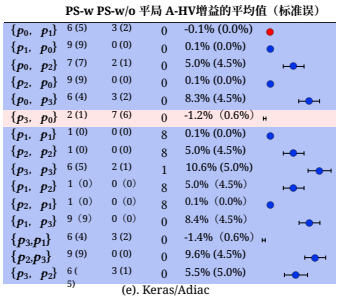
(b). x264



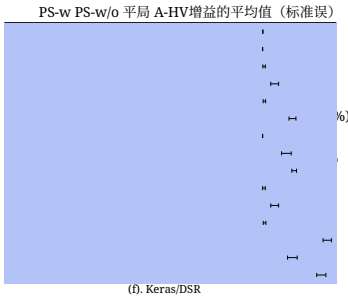
(c). Storm/WC



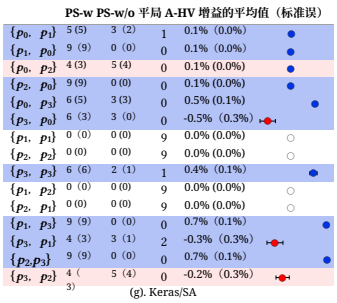
(d). Storm/RS



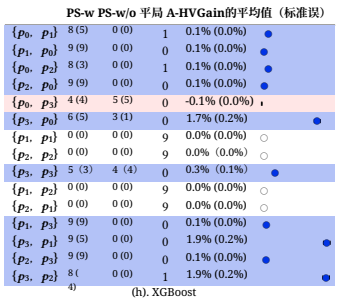
(e). Keras/Adiac



(f). Keras/DSR



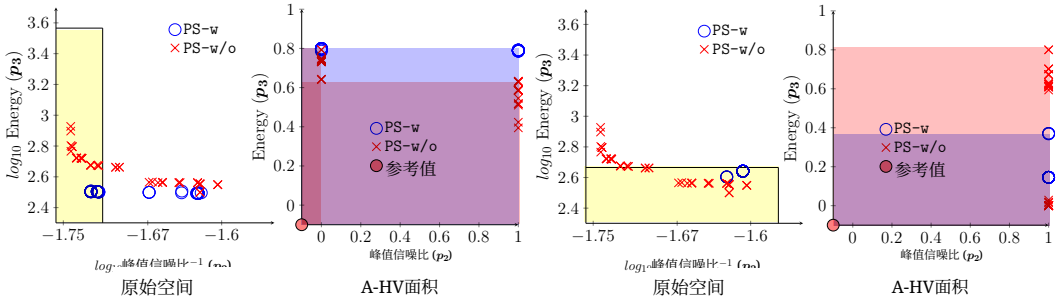
(g). Keras/SA



(h). XGBoost

> ***RQ1***： 在现实期望空间下，PS-w 有 *84%* 的情况下与 PS-w/o 效果相当或更好，并取得了显著改进，这表明在这种情况下，需求和期望有助于指导调优。然而，效果可能因某些特定的模式组合而异；例如，仅给出 p_1 和/或 p_0 时，效果往往不明显。

5.1.3 讨论。为了解现实期望空间下结果产生的原因，图 12 展示了 x264 中一个常见示例：所有优于其期望值的峰值信噪比（PSNR）值都同样受到青睐，且不接受更差的结果（p_2）；与此同时，能耗越低越理想，即使已经超出其期望值也是如此（p_3）。图 12a 是 PS-w 的一个优势案例，我们可以看到，期望值促使调优更聚焦于目标空间内的局部区域，因此 PS-w 的点分布范围远小于 PS-w/o（如原始空间中所示）。



(a) x264，PS-w 获胜　　(b) x264，PS-w/o 获胜

图12. 在现实期望空间下（由阴影区域表示），最终配置集（使用 NSGA-II）的示例运行结果。

这种“聚焦压力”通常足以帮助场景在固定调优预算下找到更多受青睐的区域，因此 PS-w 的 A-HV 优于 PS-w/o（体积更大，如*A-HV*面积中所示）。

然而，PS-w 并非总是有益。正如 Chen、Li [17], Nair 等人 [62], 以及其他人 [38, 44], 所报告的那样，可配置软件系统已知具有高度稀疏性，即相近的配置也可能表现出截然不同的性能，因此只有少量配置可能达到特定的性能范围，导致目标点相当稀疏（例如，图 9）。例如，在 Storm 中切换等待_策略会对性能产生巨大影响，尽管这只是对一个选项进行的单一更改。这是因为等待_策略会根据等待时间是固定间隔，还是在运行时依据队列长度逐步确定，来节省 CPU 使用率，因此会对延迟和吞吐量产生重大影响。然而，在调优过程中，它被表示为一个配置选项，其取值为 {0，1，2，3}，每个取值代表一种不同的等待策略。高度稀疏性会加剧局部最优陷阱问题——优化器难以逃离的一些非理想区域。在高稀疏性下进行集中搜索，有时确实会导致 PS-w 过分强调较不理想的局部最优，从而损害结果。这就是某些案例中 PS-w 没有优势的原因，如图 12b 所示：与 PS-w/o 的点相比，PS-w 的点过于密集（见原始空间），导致 PS-w 覆盖的体积小于 PS-w/o，A-HV 也更小（见*A-HV*面积）。

有趣的是，在某些模式组合下，即仅包含 $\boldsymbol{p_0}$ 和/或 $\boldsymbol{p_1}$ 时，两个优化模型的表现相似。这是合理的，因为在这些案例中，需求对不同配置的区分能力与 PS-w/o（其本质上由{$\boldsymbol{p_0}$, $\boldsymbol{p_0}$}引导）相似，因此生成的配置在给定需求下同样受青睐。

5.2 RQ2：不同期望如何影响比较?

5.2.1 方法。 为了解答**RQ2**，我们采用**RQ1**所用的流程，但重点关注与所使用的三个期望空间（即 l、 c和 r）相关的结果。

5.2.2 结果。 图13绘制了 A-HV 对不同期望空间的敏感性。尽管对于不同模式和系统，整体结论与**RQ1** 一致，但我们发现，期望空间位于某些位置时，收益往往会明显偏向特定位置。例如，在 x264 和 Keras/Adiac 上，PS-w 在期望空间位于目标空间中心区域时，提升尤其显著。相比之下，在左移

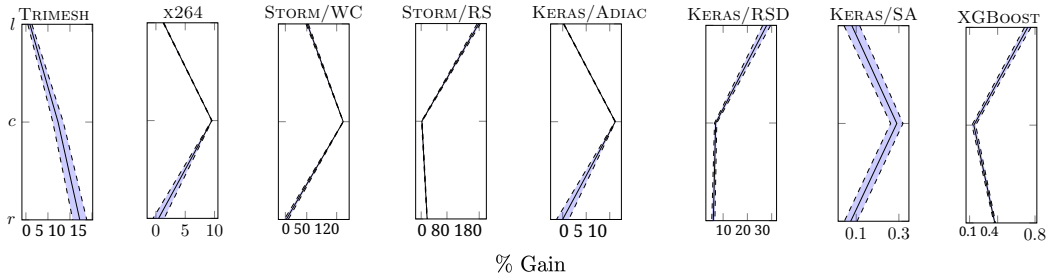


图 13. PS-w 相较于 PS-w/o 的收益百分比对现实期望空间不同位置的敏感性。每个数据点表示所有模式组合和优化器的平均值及标准误。 *l*、 *c*和 *r*分别表示性能景观中的左移、居中和右移位置。

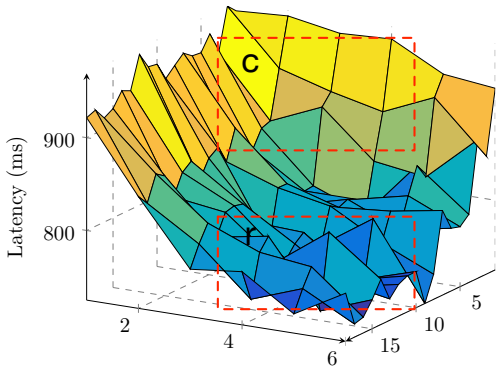


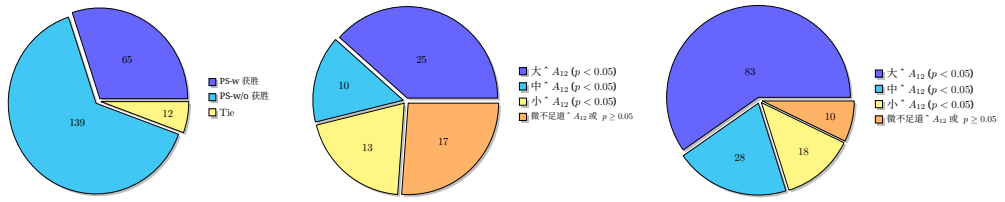
图 14. Storm/WC 中性能目标“延迟”相对于配置选项“分流器”和“计数器”的投影景观。 *c*和 *r*分别表示居中和右移的期望空间。请注意，期望空间是有界的，因为同时考虑了吞吐量目标；为简化说明，此处未展示该目标。

期望空间位于 Storm/RS 的期望空间，以及 Storm/WC 的中心 期望空间，收益尤其显著；这可能取决于系统的景观（下文将对此进行讨论）。事实上，某些期望空间容易使 PS-w 陷入局部最优，导致其相较于 PS-w/o 的改进效果变得不明显。例如，在 Storm/WC 上，当期望空间右移时，这种影响会造成很大的负面作用，从而严重削弱 PS-w 的优势。

总而言之，我们得出以下结论：

> ***RQ2***： PS-w 相较于PS-w/o 的改进往往很大程度上偏向期望空间在性能景观中的某些位置，例如中心或左移位置。然而，总体而言，PS-w 仍然表现更优。

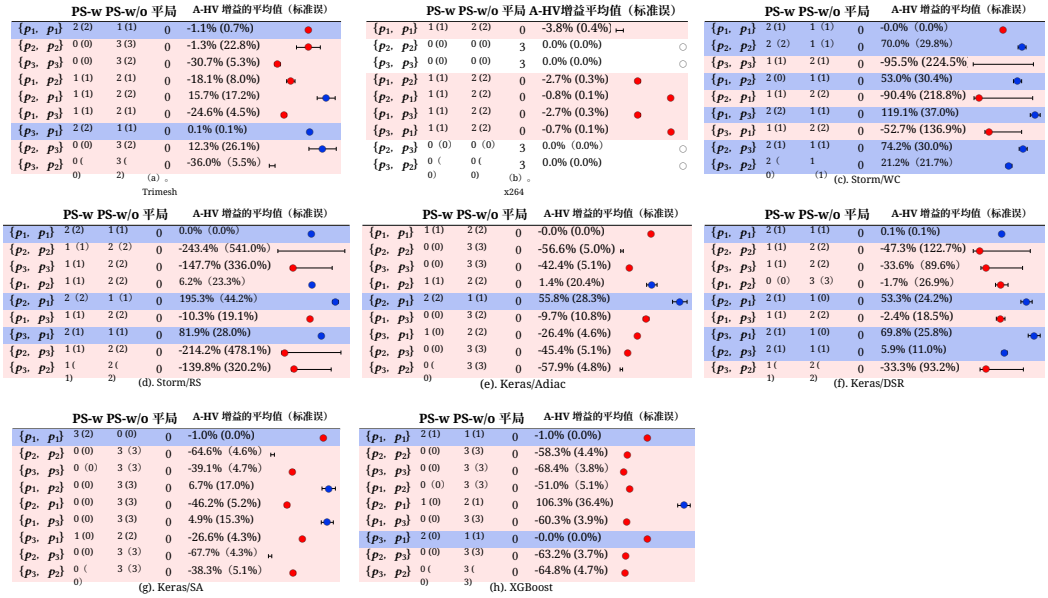
5.2.3 讨论。 如**RQ1**所述，PS-w 优于 PS-w/o 的主要原因是“聚焦搜索压力”。然而，如果调优遇到优化器难以跳出的复杂局部最优，这种方法未必总是有益。收益对期望空间位置表现出的高度敏感性表明，局部最优在景观中的分布可能并不均匀。如果期望空间覆盖了许多局部最优区域，那么 PS-w 带来的收益当然就微乎其微。



(a) Distribution on all 216 cases (b) Distribution on the 65 cases that PS-w win (c) Distribution on the 139 cases that PS-w/o win

图15. PS-w 和 PS-w/o 的获胜情况汇总及其详细统计验证结果。

表 8. 在不切实际的需求和期望下，对 PS-w 与 PS-w/o 进行比较，运行次数为100次。格式与表7相同。

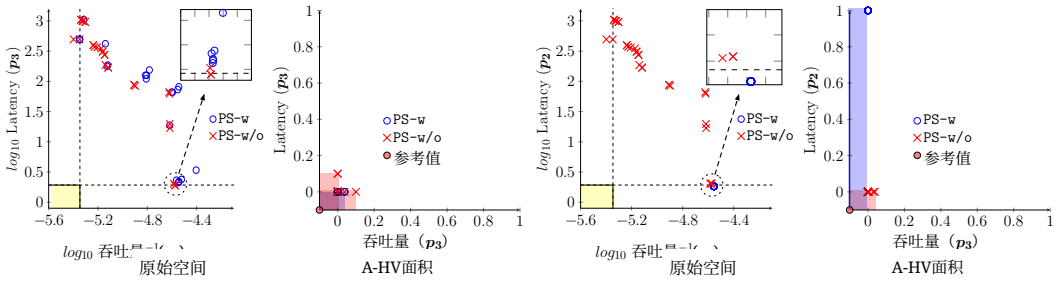


例如，在图14中，显然，期望空间 c （涵盖延迟和吞吐量的需求）在景观中的某些区域会受到边界限制，而这些区域的延迟表面要平滑得多。然而，对于 r，该区域会变得崎岖陡峭，其中包含一些非常难以应对的局部最优。遗憾的是，我们没有在不同可配置软件系统中观察到这种敏感性呈现一致的模式。这也合乎情理，因为这些系统的性能景观可能大不相同。

5.3 研究问题3：如果期望不切实际怎么办?

5.3.1 方法。 为研究**研究问题3**，我们省略了包含 p_0 的场景，因为它们无法构建不切实际的期望空间。因此还剩下九种模式组合；这九种组合与三个帕累托优化器及八个研究对象一起，共形成 $9 \times 3 \times 8 = 216$ 个案例。其他所有设置均与**RQ1**相同。

5.3.2 结果。 从图15的汇总结果可以看出，PS-w/o 总体上在所有案例中表现更好：其获胜率为 64%（139/216），而败率为 30%（65/216）。另有 6%（12/216）的情况为平局。这意味着，与 PS-w 的 36%（77/216）相比，PS-w/o 在 70%（151/216）的案例中表现更好或相近



（a）Storm/RS，PS-w/o获胜 (b) Storm/RS，PS-w 获胜

图 16. 在不现实期望空间（阴影区域所示）下，最终配置集（采用 NSGA-II）的示例运行。

ˆ 对于 PS-w。在这些案例中，PS-w/o 在 $A_{12} \leq 0.4$ 和 $p < 0.05$ 的比较中获胜 129 次，而在其落败的此类案例中有 41 次。

检查具体的系统和需求场景时，可以在表 8 中看到类似结果。尽管 PS-w 在少数案例中仍有优势，但更常见的情况是毫无改进，甚至造成相当大的负增益，平均值最高可达 −243%。此外，PS-w 在所有案例中的统计显著获胜次数总体也少得多。特别是，我们发现，在所有系统中采用 $\{\boldsymbol{p}_1, \boldsymbol{p}_1\}$ 时，两种优化模型的表现相近，但 PS-w 往往获胜更多。这是因为这种组合模式是不现实的期望值不会导致过多不可比较配置的唯一情况。

总而言之，我们得出以下结论：

> 研究问题***3***：当期望空间不现实时，PS-w/o 更为稳妥：在 *70%* 的情况下，其表现与 PS-w 相近或明显更好，这意味着在这种情况下，需求和期望值对调优的指导作用弊大于利。不过，唯一的例外是 $\{\boldsymbol{p}_1, \boldsymbol{p}_1\}$。

5.3.3 讨论。在不现实期望空间下，最常见的案例与图 16 中的 Storm/RS 示例相似：当多样性趋于较高时，PS-w 通常不如 PS-w/o（图 16a）；但在多样性有限时，PS-w 有时优于 PS-w/o （图 16b ）。这是因为在大多数案例中，使用具有不现实期望值的需求进行转换后，PS-w 往往从一开始就找到过多不可比较配置（如 $\{\boldsymbol{p}_1, \boldsymbol{p}_1\}$ 以外的案例中，大多数配置至少在一个性能目标上完全不满足需求），这意味着不现实期望空间所提供的指导作用被大幅削弱。尽管这种不可比较性可能会略微提升多样性，帮助搜索逃离局部区域，但它往往会严重削弱搜索朝着更受偏好的配置（达到或超过期望值）的倾向，导致其 A-HV（较小的体积）低于图 16a 中的 PS-w/o。这是因为在这种情况下无法产生选择压力（即区分能力）。这也解释了为何在 $\{\boldsymbol{p}_1, \boldsymbol{p}_1\}$ 下，PS-w 并未因不现实期望值而退化，因为这仍能确保配置之间具有可比性。然而，有时（图 16b），如此高的不可比较性确实会帮助 PS-w 偶然找到一个良好配置（例如，其延迟优于期望值），而这比 PS-w/o 找到的配置更受青睐，因此 HV（较大的体积）更高。因此，尽管由于稀疏性较高，调优很容易被困在该配置上，PS-w 在某些案例中仍然表现更好。

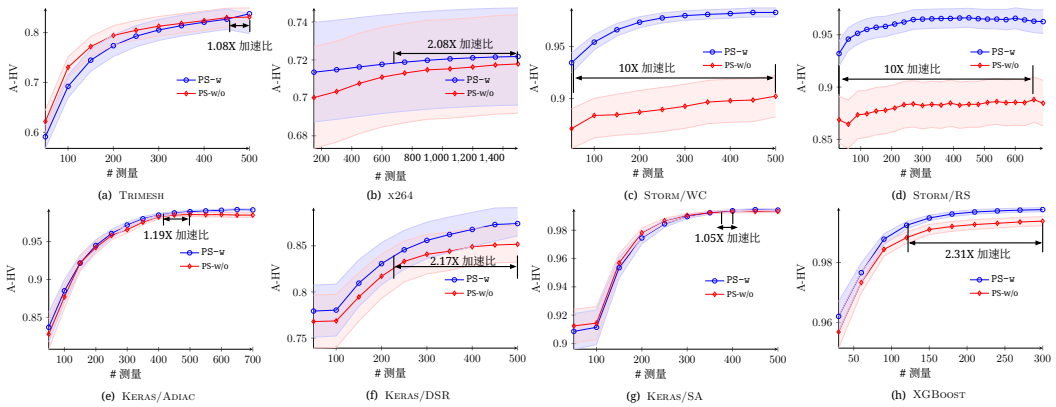


图 17. 现实期望下 PS-w 相较于 PS-w/o 的加速比（每个点表示所有模式组合、期望空间、优化器及其运行次数的平均值和标准误）。

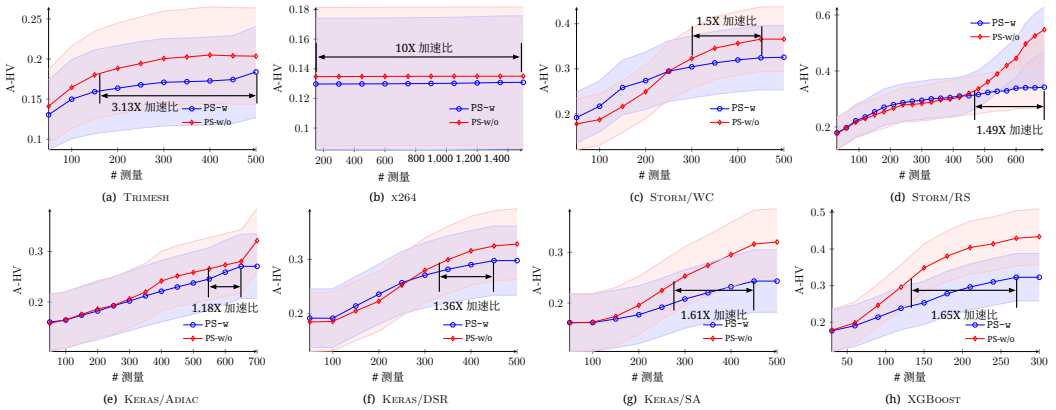


图 18. 不现实期望值下 PS-w/o 相较于 PS-w 的加速比（每个点表示所有模式组合、优化器及其运行次数的平均值和标准误）。

5.4 研究问题4：给定的调优资源是否重要?

5.4.1 方法。 为了解研究问题4中两种优化模型的资源效率，对于每个系统，我们采用以下流程:

(1) 绘制 PS-w 和 PS-w/o 的 A-HV 随测量次数变化的轨迹，其中每个点是所有需求模式、期望空间和优化器的平均值。(2) 确定一个基线， b，取基线模型达到其最佳 A-HV（设为 T）所消耗的最少测量次数。(3) 对于另一个模型，找出其平均 A-HV 等于或优于 T 时所需的最少测量次数，记为 m。(4) 根据 Gao 等人 [30] 使用的指标，计算相对于基线模型的加速比，即 $s = \frac{b}{m}$。

—

由于我们发现，整体表现更好的优化模型取决于期望空间是否现实，因此我们分别使用 PS-w/o 和 PS-w 作为现实期望和不现实期望情形的基线。

5.4.2 结果。从图 17 所示的现实期望下的结果可以看出，在不同可配置系统的整个调优过程中，PS-w 始终优于 PS-w/o，这进一步强化了我们关于**RQ1**的发现。资源效率的提升十分显著：加速比介于 1.05× 和 10×之间。相比之下，当给定的期望不现实（图 18）时，PS-w/o 的资源效率高得多，因为它能实现从 1.18× 到 10×的加速比。这再次符合我们关于**研究问题3**的发现。然而，在不现实期望下，PS-w/o 的优势在调优初期可能并不明显；在某些系统中（例如图 18c 和图 18f），直到测量了约 250 种配置后，它的表现甚至仍不如 PS-w。总而言之

y, 我们发现：

研究问题***4***：在现实期望下，PS-w在整个轨迹中通常都能获得持续优于PS-w/o的*A-HV*，加速比最高可达 10×。相比之下，当期望不切实际时，这两种优化模型在调优初期表现相当，但很快PS-w/o便会带来更好的结果，且加速比相当高。

5.4.3 讨论。在现实期望下，PS-w 在整个轨迹中都能获得更好的 A-HV，且加速比显著更高，原因有二：首先，正如我们在**RQ1**中所讨论的，需求和期望提供的指导往往有助于让调优更具针对性，从而更好地利用资源探索更有希望的区域。其次，PS-w/o 会将宝贵的调优预算浪费在探索它偏好的配置上，但这些配置在给定需求下永远不会被优先选择，因为它天然关注整个帕累托前沿。因此，上述差异使 PS-w 成为某些系统（如 Storm）中特别有吸引力的模型，因为不同配置的性能可能存在巨大差异。

当给定的期望值不现实时，情况则截然不同，这主要是由于 PS-w 中不可比性较高，如**研究问题3** 中所述——使用具有不现实期望值的需求进行转换后，许多配置彼此不可比。事实证明，这种情况可能会给任何帕累托优化器带来严重问题 [56], ，因为资源将主要耗费在探索上。然而，这种不可比性有时也能偶然帮助探索某些首选配置，尤其是在调优初期，此时 PS-w/o 尚未探索足够的空间来逼近帕累托前沿。因此我们看到，在初期，PS-w 的表现与 PS-w/o 相近，有时甚至更好。

6 经验教训

本节结合经验教训和发现的未来机遇，探讨我们的研究发现如何帮助该领域的从业者。

第***1***课：是否利用期望值来指导调优，主要取决于这些期望是否现实。

有趣的是，对于具有两个性能目标的软件配置调优，我们不能像现有研究普遍假设的那样，随意得出应在 PS-w 和 PS-w/o 之间择一的结论。相反，根据**RQ1**、**RQ3**和**研究问题4**，我们发现，给定期望的现实程度是选择的关键：当期望切合实际时，PS-w 更有益；而当期望不切实际时，PS-w/o 更稳妥（前提是调优预算也足够）。这凸显了在选择合适的优化模型来调优具有两个性能目标的软件配置之前，了解给定的需求和期望是否切合实际，或其中所隐含的假设，至关重要。

经验教训 ***2***：少量模式组合就可能改变是否在调优中纳入期望的决策，但也会影响愿望引导的调优所带来的益处或弊端。

尽管从 **RQ1** 中我们注意到，当给定的模式组合仅包含 p_1 和/或 p_0 时，PS-w 的益处并不明显，但这并不会改变选择，因为 PS-w 仍然优于 PS-w/o。唯一明确的情况是，当期望值不现实且模式为 $\{p_1, p_1\}$ 时，应选择 PS-w。因此，我们认为，给定模式对 PS-w 与 PS-w/o 之间选择的敏感性很小，而且我们还发现了其他更重要的因素。不过，我们确实发现，*PS-w* 带来的改进或退化程度可能对给定模式较为敏感。

经验教训 ***3***：现实期望空间在目标空间中的位置会在很大程度上影响在调优过程中考虑期望值所带来的益处，但不太可能影响选择。

从**RQ2** 中意外发现，当给定现实期望时，期望空间的位置会在很大程度上影响 PS-w 的益处。虽然这不太可能影响 PS-w 与 PS-w/o 之间的选择，但确实凸显了系统分析期望空间与系统配置景观之间相关性的必要性，尤其是分析期望空间覆盖某些难以达到的局部最优的可能性及其蕴含关系。

第***4***课：当期望值现实合理时，给定的调优预算对选择的影响微乎其微。然而，在期望值不切实际时，它可能是一个需要考虑的重要因素。

根据 **研究问题4**，我们还发现，在现实期望下，PS-w 与 PS-w/o 之间的选择对调优预算仅有轻微敏感性；但在期望值不现实的情况下，预算可能会影响这一选择。这为不现实期望值增添了一个需要考虑的层面。一般而言，我们观察到，调优预算较小时，采用 PS-w/o 的优势远没有那么明显，因此使用这两种优化模型中的任意一种，都可能不会产生显著不同的结果。然而，在预算充足的情况下，PS-w/o 很可能优于 PS-w。遗憾的是，根据目前的证据，我们仍很难准确量化要实现这种区分，调优预算需要“小”到什么程度或“大”到什么程度。

上述经验不仅揭示了从业者在为双目标软件配置调优选择 PS-w 和 PS-w/o 时需要考虑的重要因素，也暗示了该领域未来的一些研究机遇。这些机遇包括：

- **可配置软件系统的景观分析：**我们发现，期望空间的现实性、其在目标景观中的位置以及调优预算，可能是选择 PS-w 还是 PS-w/o 时需要考虑的关键因素。这些因素都与对可配置系统本身进行景观分析有关。事实上，通过系统地分析收集到的任何数据，我们能够进一步了解上述因素，从而更有依据地决定是否在调优中纳入需求。

- **用于配置调优的需求鲁棒优化器：**期望值的现实性无疑是选择 PS-w 还是 PS-w/o 的关键因素。然而，我们未必总能预先获得这方面的知识，因此决策存在不确定性。鉴于此，理想的做法是结合 PS-w 和 PS-w/o 的优势，设计一种能够应对需求不确定性的优化器。同样，前文所述的景观分析可以为这类设计提供思路。

- **对需求模式及其与调优之间关系的严谨分析：**虽然需求模式对 PS-w 与 PS-w/o 之间的选择几乎没有蕴含作用，但我们仍需深入了解它们为何会在某些模式上产生更大的差异，以及它们究竟如何影响 PS-w 的性能。事实上，在

理论层面，第3节中的量化为模式切换提供了理论推理基础，这对于需求放宽这一主题十分重要。例如，这可以从两个方面实现：

– 通过量化这些模式，可以正式展示它们之间的关系。例如，由于 p_1 中所有点的满意化值都高于 p_3 中的点，因此我们可以说，p_1 是 p_3 的一种“放宽”形式。– 同样，我们也可以量化具有两个不同期望水平的模式之间的关系。

基于上述理解，我们让软件工程师在调优过程中能够针对给定需求获得更强的可解释性。例如，调优完成后，人们便能知道如何放宽或收紧需求，从而在这些需求的约束下找到最受青睐的配置。这可以成为一个统一流程，将需求协商与调优本身结合起来。

• **交互式配置调优：** 在实证方面，我们的发现为交互式调优过程中如何应对不同情况提供了一些见解。例如，如果软件工程师发现，在 PS-w 下调优从未（或很少）产生满足需求/期望的配置，那么在担心底层优化器是否合适之前，可以立即切换到 PS-w/o。同样，通过改变期望空间的位置，也可以影响 PS-w（或 PS-w/o）产生的结果。

7 有效性威胁

与软件工程领域的其他实证研究一样，我们的工作在以下方面可能存在构念效度威胁：

• **度量指标：** 帕累托搜索会生成一组配置，因此比较时需要针对配置集，而非单个配置进行。我们采用了超体积（HV），这是一种用于评估解集的综合指标，并遵循 Li 等人 [55] 提出的方法。由于给定的需求及其期望值可能构成不同的集合，因此应在此类场景下比较配置集。为此，我们扩展了超体积，使其能够明确考虑需求模式，详见第4节 4.4.1。
• **统计：** 帕累托优化器的随机性可能威胁结果的稳定性。为降低这种影响，我们重复进行了100次运行，并采用 Wilcoxon 检验以及 A_{12} 验证所有成对比较。上述方法均已被推荐，并广泛应用于软件工程研究 [1]。

在我们的研究中，有两个因素可能对**内部效度**构成威胁：

• **调优预算：** 考虑到研究规模，我们为每个案例设定了一小时的预算，这对于基于搜索的软件工程中的高开销问题而言是常见的设置 [53]。为减轻实验干扰，我们随后按照系统化步骤将其转换为独立测量的数量（第4.3.2节）。在第5.4节中，我们还分析了 A-HV 的变化轨迹，展示使用较小预算时会出现什么情况。诚然，研究更大的调优预算可能会影响部分结果，但要验证这一点，需要投入更多的计算资源和时间（因为调优开销很高）；我们计划将其作为未来工作的一部分。

• **优化器设置：** 在本研究中，我们遵循文献中已证明对基于搜索的软件工程问题有效的做法，因为我们的目标是比较最常见的实践。我们无法确切确定的唯一部分是种群规模，因为它高度依赖于具体问题。为此，我们遵循了精心设计的标准（第4.3.3节），在合理的收敛效果与调优预算内所需的时间之间取得平衡。不过，

我们确实认同，探索其他参数设置是一个值得进一步探讨的方向，这将留待未来工作。

对**外部效度** 的威胁可能来自多种因素，包括：

• 软件系统： </style}在本研究中，我们依据经过仔细编纂的规则（第4.1节），从现有软件配置调优研究中选取了八个最具代表性的系统/环境。这些被测系统来自不同领域，在规模、性能目标和搜索空间方面也各不相同。值得注意的是，所提取的需求包括针对更复杂系统的需求，例如信息物理系统，而我们考察的对象主要是软件系统。这并不会严重削弱我们的结论，因为提取出的蕴含关系和模式相当通用，可以应用于不同案例；同时，某些性能属性与范围广泛的系统相关，例如延迟和吞吐量相关需求（具有不同的期望水平） [62]。尽管如此，我们同意所研究系统的清单并不全面，可能遗漏了一些只有在更复杂系统中才会显现的特殊情况。针对更多不同类型的系统进行实验或许会有所助益。另一个相关问题是，我们没有在跨越软件和硬件层的高度复杂软件系统上检验结果。在这类情况下，跨层配置选项之间的交互可能更加复杂，从而形成一些不同的配置景观 [42]。因此，考察这些高度复杂的系统或许能带来新的见解，并进一步巩固我们的研究发现。

值得注意的是，比较不同类型软件系统的结果可能尤其有吸引力。然而，遗憾的是，我们尚未观察到能够根据系统所属领域得出的一致结果模式，因此无法据此得出普遍结论。这可能有两个原因：

– 各系统运行时所采用的工作负载和基准测试大不相同，因此形成了各异的配置景观。

– 正因如此，即使是属于同一领域的系统，所采用的合适期望值（级别）也各不相同。

再次说明，使用更多软件系统或许有助于实现这一目标，这当然也是我们未来工作的计划。不过，这并不影响我们关于 PS-w 与 PS-w/o 对比所得出的结论。

• **配置选项：** 每个配置选项的离散化级别可能会对调优产生不可忽视的影响。在本研究中，我们使用的配置选项及其取值与先前研究 [17, 44, 45, 62] 完全相同。不过，需要指出的是，改变离散化级别可能会揭示新的见解，我们将把对此的研究纳入未来工作。

• **需求模式与期望空间：** 为了模拟现实世界中的需求场景（如第3节所示），我们通过调查相关数据集和论文，归纳了这些蕴含关系及其量化方式。这使我们能够专注于涵盖广泛情形的四种模式，由此形成15种模式组合。至于期望空间，我们涵盖了现实期望和不现实期望值；对于前者，我们设置了三种期望空间，包括两种偏斜空间和一种更均衡的空间。不过，必须承认，始终可能存在一些被无意忽略的案例。

• **优化器：** 在本研究中，我们采用了三种常见的基于进化搜索的帕累托优化器，它们分别代表各自类别中的不同类型。当然，基于搜索的软件工程中还采用了其他流行的多目标优化方法，例如精确方法和贝叶斯优化方法。

当给定的多目标优化问题具有某些特殊特征（例如，目标函数和约束条件均为线性，且规模较小或适中）时，精确方法（例如整数线性规划）可能是非常好的选择。通过指定不同的权重或期望值／容许值，可以迭代地求得帕累托前沿上的解。此类优化问题在下一版本问题中十分常见，而成熟的精确方法，例如 ϵ-约束法和增强切比雪夫法，已展现出良好的结果 [24, 73]。

另一类流行的优化器，尤其常用于软件配置调优的是贝叶斯优化。近年来，出现了多目标贝叶斯优化器（例如， [43]），用于搜索问题的整个帕累托前沿。与进化算法相比，贝叶斯优化器通常具有更高的样本效率 [44, 46]。然而，这类方法的一个问题是，与进化算法相比，它们可能不那么容易直接纳入利益相关者的性能期望（例如，对于贝叶斯优化，这可能需要对采集函数进行谨慎设计）。不同的纳入方式可能导致不同的结果，从而很可能影响对有无性能期望的帕累托搜索进行直接比较时所得结论的可靠性。

此外，有必要指出，从多目标进化算法中得出的结论可能不适用于其他优化方法。那些受期望值“强烈”引导的优化器，如果期望值是现实的，可能会迅速找到它们；但如果期望值不现实，则可能最终得到不理想的解。在有限预算下能够在开发与探索之间取得良好平衡的优化器（如贝叶斯优化）可能会带来不同的结果，尽管这取决于期望值在优化过程中的纳入方式。因此，研究不同的优化方法以考察我们研究发现的可推广性是值得的，尤其是那些具有丰富理论结果的方法（例如，关于响应面光滑性的收敛速率，以及贝叶斯优化中的遗憾界），这些理论结果可能有助于支持我们的实证结论。这将是我们未来工作的重要组成部分。

- **目标数量：** 我们的研究涵盖了软件配置调优中涉及两个性能目标的情况。正如我们在第3节的综述中发现的那样，在考虑多个目标时，这往往是最常见的情况。不过，研究结果可能无法推广到目标维度更高的情况。未来工作可以将研究扩展到更多目标，但需要考虑的因素也会呈指数级增加，例如模式组合的数量。本研究的结果是迈出的第一步，旨在强调研究是否应使用期望值来指导涉及多个性能目标的软件配置调优。

总的来说，上述设置为我们提供了 1,000 多个案例，以便将本研究的发现推广到更广泛的情况。

8 相关工作

在此，我们结合本实证研究的目的讨论相关工作。

8.1 用于软件配置调优的 PS-w/o

基于搜索的方法已广泛采用多种优化器来研究软件配置调优，例如随机搜索 [64], 爬山法 [77], 遗传算法 [68],和蚁群优化 [14]。当目标不止一个时，帕累托搜索已被证明非常有效。Chen 等 [16]和 Singh 等 [70]等人使用不同的多目标进化算法来搜索帕累托最优配置。Nair

等 [62] 也致力于此，但其方法采用贝叶斯优化，并以类似 MOEA/D 的方式处理两个性能目标。最近，Zhu 等 [30] 也提出了一种扩展的贝叶斯优化方法，在同时考虑延迟和资源消耗的情况下达到给定的性能期望；然而，这类信息尚未被用于明确指导搜索。

上述工作有一个共同点：它们忽略了搜索过程中的期望。其假设是，优化这一概念能够获得满足任意需求模式的最优配置。

8.2 用于软件配置调优的 PS-w

相比之下，Calinescu 等人 [8, 32] 明确地将期望量化为目标的一部分（如 $\boldsymbol{p}_1$），以指导帕累托搜索。Martens 等人 [59] 也有类似的想法，但他们的模式与 $\boldsymbol{p}_3$匹配。Ghanbari 等人 [34] 也一直依靠期望来指导调优，并且假设需求模式呈现出一条更平滑的曲线。然而，尚无研究论证期望对于指导双目标软件配置调优中的帕累托搜索的重要性。事实上，Fekry 等人 [28] 最近发表的一篇讨论性论文指出，研究用于指导优化器的期望并衡量其有效性，是软件配置调优未来面临的一项重要挑战。

这些是 PS-w 优化模型的典型示例，其中性能需求在搜索和调优过程中被精确量化。还值得注意的是，实际采用的需求模式可能因假设而异，而且现有的 PS-w 相关研究都未考虑我们在研究中总结的所有模式。

8.3 使用或不使用代理进行调优

从另一个角度来看，现有的基于搜索的方法无论考虑多少个性能目标，都可以分为基于模型的方法（例如，EvoChecker [32], FLASH [62], 和 BOCA [10]）和基于测量的方法（例如，FEMOSAA [16], eQual [68], 和 Plato [66]）。前者依靠代理模型 [13] 引导搜索，后者则通过直接测量软件来进行搜索。这两者的关键区别在于搜索所处的景观：基于测量的方法直接在配置景观上搜索，而基于模型的方法则在代理景观上搜索，后者是对真实配置景观的近似。

本研究着重于通过直接测量系统来进行调优，而不使用代理模型，原因如下：

- 我们希望避免代理模型带来的噪声，因为它们不可避免地会引入误差；正如 Zhu 等人 [83], 所证明的，这些误差会严重影响搜索和调优行为。
- 事实上，我们将这两类方法视为互补而非互斥。例如，效果良好的基于测量的方法也可以应用于基于模型的贝叶斯优化，以搜索采集函数。这意味着，我们关于期望是否重要的发现同样适用于后者。

8.4 通用多目标优化

从概念上讲，我们的工作与偏好驱动的多目标优化这一主题相关。对此，正如 Wang 等人 [74],在综述中所述，对目标值（即本研究中的性能需求）的偏好可分为以下几类：

- **权重：**在这种情况下，给定一个表示各目标相对重要性的权重向量。通常，这会通过某种聚合方式（例如加权和）将多目标问题转化为单目标问题。事实上，本研究中的权重和期望都属于偏好。但二者截然不同：权重需要在各性能目标之间指定，以明确表示权衡 [68]。相比之下，期望水平是对单一目标的预期，通常更容易设定，而且不需要明确权衡。事实上，权重可以视为本研究所考虑的偏好的一种特殊情况：采用权重时，最佳配置通常是帕累托前沿上的某个特定点。相比之下，在所考虑的需求和期望下，最佳点往往不止一个，其中包括帕累托前沿上的一部分点。有趣的是，研究表明，给定一组权重后，帕累托搜索（不使用权重运行）通常能找到比权重引导的搜索更好的配置 [18]。

- **目标关系：** 这指的是已对目标进行完整或部分排序的情况。例如，人们可能会优先满足吞吐量需求，然后再考虑延迟。相比之下，本文中的模式组合与期望之间并未对目标进行直接排序。
- **目标空间区域：** 人们可以大致指定目标空间中特别偏好的区域，例如膝点或极端点。与第3节中的模式组合相比，这种偏好较为模糊——膝点或极端点是相对于已找到的解而言的，而模式仍由明确定义的期望空间所引导。

- **参考点 [51, 80]：** 在此，会给出一个表示各目标期望值的向量。从这一点来看，这一概念确实与本文讨论的期望水平相似。不过，主要区别在于，基于参考点的多目标优化总是假设必须首先考虑帕累托最优性；也就是说，优先选择接近参考点的帕累托最优解（沿着决策者指定的某个方向） [2]。此外，还需要额外的参数来指定优选解的分布范围 [51]。相比之下，我们为软件配置调优发现的模式组合可能与上述情况大不相同：期望空间内（外）的解可能同样受到偏好（不受偏好），而且并不一定偏好接近期望向量的解。此外，除期望水平外，无需其他参数。

因此，正如我们在第3节中总结的那样，以上所有偏好形式都不同于用于软件配置调优的需求期望和模式。这一点很重要，因为总结出的模式源自针对问题特征的实证发现——它们未必能推广到其他问题，但对软件配置调优意义重大。

通用优化领域也有一项关于在搜索中考虑偏好重要性的实证研究 [51, 81]。不过，这项研究在两个方面与我们的工作不同：

- 他们关注的是参考值，而正如我们所讨论的，参考值与本研究考察的模式截然不同。
- 所研究的优化器是会考虑特定偏好形式的专用算法。相比之下，我们的研究考察的是配备需求模式以引导搜索的经典优化器，这也是软件配置调优中普遍采用的做法。

总而言之，据我们所知，本研究是首个实证研究，旨在根据该问题中需求模式的特征及实践总结，探究期望值是否、何时以及为何对引导双目标软件配置调优很重要。

9 结论

在本文中，我们开展了一项全面的实证研究，深入了解性能期望是否会影响可配置软件系统的双目标调优。我们的研究涵盖了 15 种用于量化期望的模式组合、四种期望空间、三种搜索算法，以及八种软件系统/环境，共调查了 1,296 个案例。研究结果挑战了“调优时是否使用期望值无关紧要”这一观点，并揭示了：

- 当期望合理时，它们通常有助于指导调优。
- 但是，当这些期望不合理时，它们往往会对调优产生负面影响。
- 不同的模式和期望空间的位置不会改变上述结论，但会影响由此产生的收益或损害程度。

- 当期望切合实际时，调优预算对调优收益的影响微乎其微。相反，当期望不切实际时，调优预算则是一个重要因素。

我们的研究发现为这一特定研究领域的从业者提供了有益的见解，尤其是关于回答“调优过程中何时应考虑期望？”这一问题所需的因素和信息。

我们概述了一些令人振奋的未来研究机会：

- 为了更好地理解给定期望的现实可能性及其所处位置，我们应分析可配置软件系统的景观，即使分析仅基于有限的样本。

- 为应对给定需求和期望的不确定性（例如，它们可能不切实际），我们应研究用于软件配置调优的需求鲁棒优化器。
- 为了解释需求模式对调优指导的影响，我们应为严谨分析它们之间的关系，以及它们如何相互转换以实现松弛，奠定基础。

参考文献

[1] Andrea Arcuri 和 Lionel C. Briand。2011。使用统计检验评估软件工程中随机化算法的实用指南。载于第*33*届国际软件工程会议（*ICSE 2011*）论文集，威基基，檀香山，夏威夷州，美国，*2011*年*5*月*21*日至*28*日。1–10。[2] Anne Auger、Johannes Bader、Dimo Brockhoff 和 Eckart Zitzler。2009。通过加权超体积采样阐明多目标问题中的用户偏好。载于第*11*届遗传与进化计算年会论文集。555–562。[3] Liang Bao、Xin Liu、Fangzheng Wang 和 Baoyin Fang。2019。ACTGAN：利用生成对抗网络为软件系统自动配置调优。载于第*34*届*IEEE/ACM*国际自动化软件工程会议（*ASE*） *2019,* *11 15, 2019* 465 476 10 1109 2019 00051 美国加利福[4]尼亚州圣迭戈，*11*月-。IEEE，–。https://doi.org/。/ASE。。Luciano Baresi、Liliana Pasquale 和 Paola Spoletini。。，。面向需求驱动适应的模糊目标。载于*RE 2010*，第*18*届*IEEE*国际需求工程会议，澳大利亚新南威尔士州悉尼，*9*月*27*日至*10*月 *1, 2010* 125 134 10 1109 2010 25 。IEEE计算机学会，–。[5]https://doi.org/。/RE。。Slim Bechikh、Marouane Kessentini、Lamjed Ben Said 和 Khaled Ghédira。。，。第四章——进化多目标优化中的偏好融入：最先进技术综述。计算进展98（2015），141–207。https://doi.org/10 1016 2015 03 001 。/bs.adcom。。Babak Behzad、Huong Vu Thanh Luu、Joseph[6]Huchette、Surendra Byna、Prabhat、Ruth A. Aydt、Quincey Koziol 和 Marc Snir。2013。通过自动调优驾驭并行I/O复杂性。载于高性能计算、网络、存储与分析国际会议（*SC'13*），美国科罗拉多州丹佛，*2013*年*11*月*17*日至*21*日，William Gropp 和 Satoshi Matsuoka（编）。ACM，68:1–68:12。https://doi.org/10.1145/2503210.2503278

[7] Kate M. Bowers、Erik M. Fredericks 和 Betty H. C. Cheng。2018。自适应系统中加权非功能目标的自动化优化。载于《基于搜索的软件工程——第十届国际研讨会（SSBSE 2018）》，法国蒙彼利埃，2018年9月8日至9日，会议论文集（计算机科学讲义，第11036卷），Thelma Elita Colanzi 和 Phil McMinn（编）。施普林格，182–197。[8] Radu Calinescu、Milan Ceska、Simos Gerasimou、Marta Kwiatkowska 和 Nicola Paoletti。2017。通过参数化马尔可夫链综合设计鲁棒软件系统。载于IEEE国际软件架构会议。[9] Radu Calinescu、Milan Ceska Jr.、Simos Gerasimou、Marta Kwiatkowska 和 Nicola Paoletti。2018。随机系统鲁棒模型的高效综合。《系统与软件杂志》 143 （ 2018 ），140–158。[10] Junjie Chen 、Ningxin Xu、Peiqi Chen 和 Hongyu Zhang。2021。通过贝叶斯优化实现高效的编译器自动调优。载于第43届IEEE/ACM国际软件工程会议（ICSE 2021），西班牙马德里，2021年5月22日至30日。IEEE, 1198 1209 10 1109 43902 2021 00110 –。https://doi.org/。/ICSE。。Tao Chen。，。自适应系统[11] 2022的终身动态优化：事实还是虚构？载于IEEE国际软件分析、演化与再工程会议（SANER 2022），美国夏威夷州檀香山，2022年3月15日至18日。IEEE, 78 89 10 1109 53432 2022 00022 –。https://doi.org/。/SANER。。Tao Chen 和 Rami Bahsoon。，。[12] 2015迈向更智能的云：云配置与资源的自感知自动扩缩容。《计算机》 48, 9 2015 93 96 10 1109 2015 278 ( ), –。https://doi.org/。/MC。。Tao Chen 和[13] 2017Rami Bahsoon。，。基于云的软件服务的自适应在线服务质量建模。《IEEE软件工程汇刊》 43, 5 2017 453 475 10 1109 2016 2608826 ( ), –。https://doi.org/。/TSE。。Tao[14] 2017Chen 和 Rami Bahsoon。，。基于云的服务自动扩缩容中的自适应权衡决策。《IEEE服务计算汇刊》10，4（2017），618–632。[15] Tao Chen、Rami Bahsoon 和 Xin Yao。2018。自感知与自适应云自动扩缩容系统综述与分类。《ACM计算综述》 51, 3 2018 61 1 61 40 10 1145 3190507 ( ), : – :。https://doi.org/。/[16] 2018Tao Chen 、 Ke Li 、 Rami Bahsoon 和 Xin Yao。，。FEMOSAA：面向自适应软件的特征引导与膝点驱动多目标优化。《ACM软件工程与方法学汇刊》27，2（2018）。[17] Tao Chen 和 Miqing Li。2021。软件配置调优的多目标化。载于ESEC/FSE '21：第29届ACM联合欧洲软件工程会议暨软件工程基础研讨会，希腊雅典，2021年8月23日至28日，Diomidis Spinellis、Georgios Gousios、Marsha Chechik 和 Massimiliano Di Penta（编）。ACM, 453 465 10 1145 3468264 3468555 –。https://doi.org/。/。Tao Chen 和 Miqing Li。，。权重可能有害：基[18] 2022于搜索的软件工程中的帕累托搜索与加权搜索。《ACM软件工程与方法学汇刊》（2022年1月）。https://doi.org/ 10 1145 3514233 。/ Tao Chen、Miqing Li、Ke Li 和 Kalyanmoy Deb。2020。面向自适应系统的基于搜[19]索的软件工程：综述、失望之处、建议与机遇。CoRR abs/2001.08236（2020）。[20] Tao Chen、Miqing Li 和 Xin Yao。2019。站在巨人的肩膀上：利用先验知识为软件服务组合中的基于搜索的多目标优化提供种子。《信息与软件技术》114（2019），155–175。https: //doi.org/ 10 1016 2019 05 013 。/j.infsof。。Fabiano Dalpiaz、Davide Dell' [21]Anna、Fatma Basak Aydemir 和 Sercan Çevikol。2019。利用可解释机器学习和依存句法分析进行需求分类。载于第27届IEEE国际需求工程会议（RE 2019），韩国济州岛，2019年9月23日至27日，Daniela E. Damian、Anna Perini 和 Seok-Won Lee（编）。IEEE, 142 152 10 1109 2019 00025 –。https://doi.org/。/RE。。[22] 2002K. Deb、A. Pratap、S. Agarwal 和 T. Meyarivan。，。快速且精英的多目标遗传算法：NSGA-II。《IEEE进化计算汇刊》6，2（2002），182–197。[23] Quoc Anh Do、Surendra Raju Chekuri 和 Tanmay Bhowmik。2019。通过需求复用自动支持捕获创造性需求。载于《大数据时代的复用——第18届国际软件与系统复用会议（ICSR 2019）》，美国俄亥俄州辛辛那提，2019年6月26日至28日，会议论文集（计算机科学讲义，第11602卷），Xin Peng、Apostolos Ampatzoglou 和 Tanmay Bhowmik（编）。施普林格，47–63。https://doi.org/10.1007/ 978-3-030-22888-0 4_[24] Miguel Ángel Domínguez-Ríos、Francisco Chicano、Enrique Alba、Isabel del Águila 和 José del Sagrado。2019。求解双目标下一版本问题的高效随时算法。《系统与软件杂志》156（2019），217–231。[25] Juan José Durillo 和 Antonio J. Nebro。2011。jMetal：用于多目标优化的Java框架。《高级工程软件》 42, 10 2011 760 771 10 1016 2011 05 014 ( ), –。https://doi.org/。/j.advengsoft。。[26] 2018Michael TM Emmerich 和 André H Deutz。，。多目标优化教程：基础与进化方法。《自然计算》17，3（2018），585–609。[27] Naeem Esfahani、Ehsan Kouroshfar 和 Sam Malek。2011。驾驭自适应软件中的不确定性。载于SIGSOFT-/FSE'11第19届ACM SIGSOFT软件工程基础研讨会（FSE-19）暨ESEC'11：第13届欧洲软件工程会议（ESEC-13），匈牙利塞格德，2011年9月5日至9日，Tibor Gyimóthy 和 Andreas Zeller

(编者)。ACM, 234 244 10 1145 2025113 2025147 – 。https://doi.org/ 。 / 。Ayat Fekry、Lucian Carata、Thomas F. J.-M. Pasquier、Andrew Rice 和 Andy Hopper。2019。迈向大数据分析的无缝配置调优。载于第39届IEEE分布式计算系统国际会议（ICDCS *2019,* *7 10, 2019* 1912 1919 10 1109 2019 00189 ），美国得克萨斯州达拉斯，7月 - 。IEEE，– 。https://doi.org/ 。 /ICDCS. 。Alessio Ferrari、Giorgio Oronzo Spagnolo 和 Stefania Gnesi。 。PURE：公开需求文档数据集。载于第25届IEEE国际需求工程会议（RE 2017），葡萄牙里斯本，2017年9月4日至8日，Ana Moreira、João Araújo、Jane Hayes 和 Barbara Paech（编者）。IEEE计算机学会，502–505。https: //doi.org/ 10 1109 2017 29 . / RE. 。Yanjie Gao、Yonghao Zhu、Hongyu Zhang、Haoxiang Lin 和 Mao Yang。2021。面向深度学习模型的资源引导配置空间缩减。载于第43届IEEE/ACM国际软件工程会议（ICSE 2021），西班牙马德里，*22 30* *2021* 175 187 10 1109 43902 2021 00028 - 5月 。IEEE，– 。https://doi.org/ 。 / ICSE 。 。David Garlan、S-W Cheng、A-C Huang、Bradley Schmerl 和 Peter Steenkiste。 。Rainbow：基于架构的自适应与可复用基础设施。《计算机》37卷10期（2004），46–54。[32] Simos Gerasimou、Radu Calinescu 和 Giordano Tamburrelli。2018。面向服务质量软件工程的概率模型综合。《自动化软件工程》25卷4期（2018），785–831。[33] Simos Gerasimou、Giordano Tamburrelli 和 Radu Calinescu。2016。面向服务质量软件工程的概率模型基于搜索的综合（T）。载于IEEE/ACM自动化软件工程国际会议。319–330。[34] Hamoun Ghanbari、Bradley Simmons、Marin Litoiu 和 Gabriel Iszlai。2012。私有云的基于反馈的优化。《未来一代计算机系统》 28, 1 2012 104 111 10 1016 2011 05 019 （），– 。https://doi.org/ 。 / j.future. 。 。Alim Ul Gias、Giuliano Casale 和 Murray Woodside。 。ATOM：面向微服务的模型驱动自动扩缩。载于第39届IEEE分布式计算系统国际会议（ICDCS 2019），美国得克萨斯州达拉斯，2019年7月7日至10日。IEEE, 1994 2004 10 1109 2019 00197 – 。https://doi.org/ 。 /ICDCS. 。Jingzhi Gong 和 Tao Chen。 。配置编码是否会影响软件性能学习？关于编码方案的实证研究。载于第19届IEEE/ACM国际软件仓库挖掘会议（MSR *2022,* *23 24, 2022* 482 494 10 1145 3524842 3528431 ），美国宾夕法尼亚州匹兹堡，5月 - 。ACM，– 。https://doi.org/ 。 / 。Jichi Guo、Qing Yi 和 Apan Qasem。 。评估优化专用搜索启发式方法在有效自动调优中的作用。技术报告（2010）。[38] Huong Ha 和 Hongyu Zhang。2019。DeepPerf：利用深度稀疏神经网络预测可配置软件的性能。载于第41届国际软件工程会议（ICSE 2019）会议论文集，加拿大魁北克省蒙特利尔，2019年5月25日至31日，Joanne M. Atlee、Tevfik Bultan 和 Jon Whittle（编者）。IEEE/ACM，1095–1106。https: //doi.org/ 10 1109 2019 00113 . /ICSE. 。Xue Han 和 Tingting Yu。 。关于高度可配置软件系统性能缺陷的实证研究。载于第10届ACM/IEEE实证软件工程与度量国际研讨会（ESEM *2016,* *8 9, 2016* 23 1 23 10 10 1145 2961111 2962602 ），西班牙雷阿尔城，9月 - 。ACM，：– ：。https://doi.org/ 。 / 。Mark Harman、S Afshin Mansouri 和 Yuanyuan Zhang。 。基于搜索的软件工程：趋势、技术与应用。《ACM计算调查》（CSUR）45卷1期（2012），11。[41] Max Hort、Maria Kechagia、Federica Sarro 和 Mark Harman。2021。移动应用性能优化综述。《IEEE软件工程汇刊》（ 2021 1 1 10 1109 2021 3071193 ），– 。https://doi.org/ 。/TSE. 。Md Shahriar Iqbal 、 Rahul Krishna 、 Mohammad Ali Javidian 、 Baishakhi Ray 和 Pooyan Jamshidi。 。Unicorn：从因果关系的角度推理可配置系统性能。载于EuroSys '22：第十七届欧洲计算机系统会议，法国雷恩，2022年4月5日至8日，Yérom-David Bromberg、Anne-Marie Kermarrec 和 Christos Kozyrakis（编者）。ACM, 199 217 10 1145 3492321 3519575 – 。https://doi.org/ 。 / 。Md Shahriar Iqbal、Jianhai Su、Lars Kotthoff 和 Pooyan Jamshidi。 。Flexibo：深度神经网络的成本感知多目标优化。arXiv预印本 arXiv:2001.06588（2020）。[44] Pooyan Jamshidi 和 Giuliano Casale。2016。流处理系统最优配置的不确定性感知方法。载于第24届IEEE计算机与电信系统建模、分析与仿真国际研讨会（MASCOTS 2016），英国伦敦，2016年9月19日至21日。IEEE计算机学会，39–48。[45] Pooyan Jamshidi、Miguel Velez、Christian Kästner 和 Norbert Siegmund。2018。学习采样：利用不同环境间的相似性，为可配置系统学习性能模型。载于2018年ACM欧洲软件工程会议与软件工程基础研讨会联合会议（ESEC/SIGSOFT FSE 2018）会议论文集，美国佛罗里达州布埃纳维斯塔湖，2018年11月4日至9日，Gary T. Leavens、Alessandro Garcia 和 Corina S. Pasareanu（编者）。ACM, 71 82 10 1145 3236024 3236074 – 。https://doi.org/ 。 / 。Pooyan Jamshidi 、 Miguel Velez 、 Christian Kästner 、 Norbert Siegmund 和 Prasad Kawthekar。2017。利用迁移学习改进高度可配置软件中的模型预测。载于2017年IEEE/ACM第十二届自适应与自管理系统软件工程国际研讨会（SEAMS）。IEEE, 31–41。

[47] Vigdis By Kampenes、Tore Dybå、Jo Erskine Hannay 和 Dag I. K. Sjøberg。2007。软件工程实验中效应量的系统性综述。*Information & Software Technology* 49卷11-12期（2007），1073–1086。[48] Barbara A. Kitchenham、Pearl Brereton、David Budgen、Mark Turner、John Bailey 和 Stephen G. Linkman。2009。软件工程中的系统性文献综述——一项系统性文献综述。*Inf. Softw. Technol.* 51卷1期（2009），7 15 10 1016 2008 09 009 – 。https://doi.org/ 。 /j.infsof. 。 。[49] Anne Koziolek、Heiko Koziolek 和 Ralf H. Reussner。2011。PerOpteryx：在多目标软件架构优化中自动应用策略。载于第七届国际软件架构质量会议（*QoSA 2011*）暨第二届国际关键系统架构研讨会（*ISARCS 2011*），美国科罗拉多州博尔德，*2011*年*6*月*20*日至*24*日，会议论文集，Ivica Crnkovic、Judith A. Stafford、Dorina C. Petriu、Jens Happe 和 Paola Inverardi（编者）。ACM，33 42 10 1145 2000259 2000267 – 。https://doi.org/ 。 / 。[50] Satish Kumar、Tao Chen、Rami Bahsoon 和 Rajkumar Buyya。2020。DATESSO：利用债务感知双层约束推理实现自适应服务组合。载于*SEAMS '20*：*IEEE/ACM*第*15*届自适应与自管理系统软件工程国际研讨会，韩国首尔，*2020*年*6*月*29*日至*7*月*3*日，Shinichi Honiden、Elisabetta Di Nitto 和 Radu Calinescu（编者）。ACM，96 107 10 1145 3387939 3391604 – 。https://doi.org/ 。 / 。[51] K. Li、M. Liao、K. Deb、G. Min 和 X. Yao。2020。偏好是否总有帮助？基于参考点的偏好型进化多目标优化的全面研究。*IEEE*进化计算汇刊 24卷6期（2020），1078–1096，10.1109 2020。https://doi.org/ 。 /TEVC. 。[52] 2020 Ke Li、Zilin Xiang、Tao Chen 和 Kay Chen Tan。 。BiLO-CPDP：用于跨项目缺陷预测自动化模型发现的双层规划。载于第*35*届*IEEE/ACM*自动化软件工程国际会议（*ASE 2020*），澳大利亚墨尔本，*2020*年*9*月*21*日至*25*日。IEEE，573–584。https://doi.org/10.1145/3324884. 3416617[53] Ke Li、Zilin Xiang、Tao Chen、Shuo Wang 和 Kay Chen Tan。2020。理解迁移学习中用于跨项目缺陷预测的自动参数优化：一项实证研究。载于*ICSE '20*：第*42*届国际软件工程会议，韩国首尔，*2020*年*6*月*27*日至*7*月*19*日，Gregg Rothermel 和 Doo-Hwan Bae（编者）。ACM，566 577 10 1145 3377811 3380360 – 。https://doi.org/ 。 / 。[54] Miqing Li、Tao Chen 和 Xin Yao。2018。对《基于搜索的软件工程中评估帕累托搜索算法的质量指标选择实用指南》的批判性评述：关于基于搜索的软件工程质量指标选择的论文。载于*2018*年*IEEE/ACM*第*40*届国际软件工程会议：新构想与新兴技术成果。17–20。[55] Miqing Li、Tao Chen 和 Xin Yao。2022。如何评估基于帕累托的基于搜索的软件工程中的解？批判性评述与方法论指导。*IEEE*软件工程汇刊 48卷5期（2022），1771–1799。https://doi.org/ 10 1109 2020 3036108 . /TSE. 。[56] Miqing Li、Shengxiang Yang、Ke Li 和 Xiaohui Liu。2014。采用基于分段搜索的进化算法求解多目标优化问题。*IEEE*控制论汇刊 44卷8期（2014），1295–1313。[57] Miqing Li 和 Xin Yao。2019。多目标优化中解集的质量评估：综述。计算机调查 52卷2期（2019）。[58] Min Li、Liangzhao Zeng、Shicong Meng、Jian Tan、Li Zhang、Ali Raza Butt 和 Nicholas C. Fuller。2014。MRONLINE：MapReduce在线性能调优。载于第*23*届高性能并行与分布式计算国际研讨会（*HPDC'14*），加拿大不列颠哥伦比亚省温哥华，*2014*年*6*月*23*日至*27*日，Beth Plale、Matei Ripeanu、Franck Cappello 和 Dongyan Xu（编者）。ACM，165 176 10 1145 2600212 2600229 – 。https://doi.org/ 。 / 。[59] Anne Martens、Heiko Koziolek、Steffen Becker 和 Ralf H. Reussner。2010。利用进化算法自动改进软件架构模型的性能、可靠性和成本。载于第一届*WOSP/SIPEW*性能工程国际会议论文集，美国加利福尼亚州圣何塞，*2010*年*1*月*28*日至*30*日，Alan Adamson、Andre B. Bondi、Carlos Juiz 和 Mark S. Squillante（编者）。ACM，105 116 10 1145 1712605 1712624 – 。https://doi.org/ 。 / 。[60] Mary L McHugh。2012。评定者间信度：卡帕统计量。*Biochemia medica* 22, 3 2012 276 282 （），– 。[61] Tim Menzies、Bora Caglayan、Ekrem Kocaguneli、Joe Krall、Fayola Peters 和 Burak Turhan。2012。实证软件工程数据的PROMISE资料库。[62] Vivek Nair、Zhe Yu、Tim Menzies、Norbert Siegmund 和 Sven Apel。2020。利用FLASH寻找更快的配置。*IEEE*软件工程汇刊 46卷7期（2020）。[63] Jan Odhnoff。1965。论优化与满意化技术。瑞典经济学杂志 67卷1期（1965），24–39。[64] Jeho Oh、Don S. Batory、Margaret Myers 和 Norbert Siegmund。2017。通过随机抽样在产品线中寻找近最优配置。载于*2017*年第*11*届软件工程基础联合会议（*ESEC/FSE 2017*）会议论文集，德国帕德博恩，*2017*年*9*月*4*日至*8*日，Eric Bodden、Wilhelm Schäfer、Arie van Deursen 和 Andrea Zisman（编者）。ACM，61 71 10 1145 3106237 3106273 – 。https://doi.org/ 。 / 。[65] Andres J. Ramirez 和 Betty H. C. Cheng。2011。自动推导用于监控软件需求的效用函数。载于模型驱动工程语言与系统，第*14*届国际会议（*MODELS 2011*），

新西兰惠灵顿，*2011*年*10*月*16*日至*21*日。会议论文集（计算机科学讲义，第*6981*卷），Jon Whittle、Tony Clark 和 Thomas Kühne（编）。施普林格，
501 516 10 1007 978 3 642 24485 8_37 – 。https://doi.org/ 。 /
[66] 2009
- - - - Andres J. Ramirez 、 David B. Knoester 、 Betty H. C. Cheng 和 Philip K. McKinley。 。将遗传算法应用于自主计算系统中的决策。载于第六届自主计算国际会议论文集，*ICAC 2009*，*2009*年*6*月*15*日至*19*日，西班牙巴塞罗那，Simon A. Dobson、John Strassner、Manish Parashar 和 Onn Shehory（编）。ACM，
97 106 10 1145 1555228 1555258 – 。https://doi.org/ 。 / 。Mohammed Sayagh、
[67] Noureddine Kerzazi、Bram Adams 和 Fábio Petrillo。2020。软件配置工程：实践访谈、综述与系统性文献综述。*IEEE*软件工程汇刊46, 6 (2020), 646–673。https://doi.org/ 10 1109 2018 2867847 。 /TSE。 。Arman
[68] Shahbazian、Suhrid Karthik、Yuriy Brun 和 Nenad Medvidovic。2020。eQual：为早期设计决策提供参考。载于*ESEC/FSE '20*：第*28*届*ACM*欧洲软件工程会议与软件工程基础研讨会联合会议，线上会议，美国，*2020*年*11*月*8*日至*13*日，Prem Devanbu、Myra B. Cohen 和 Thomas Zimmermann（编）。ACM，
1039 1051 10 1145 3368089 3409749 – 。https://doi.org/ 。 / 。Zain Shaukat Shaukat、Rashid
[69] Naseem 和 Muhammad Zubair。2018。软件需求风险预测数据集。载于*2018*年*IEEE*计算科学与工程国际会议，*CSE 2018*，罗马尼亚布加勒斯特，*2018*年*10*月*29*日至*31*日，Florin Pop、Catalin Negru、Horacio González-Vélez 和 Jacek Rak（编）。IEEE计算机学会，
112 118 10 1109 2018 00022 – 。https://doi.org/ 。 /CSE。 。Ravjot Singh、
[70] Cor-Paul Bezemer、Weiyi Shang 和 Ahmed E. Hassan。2016。使用多目标遗传算法优化对象关系映射框架的性能相关配置。载于第七届*ACM/SPEC*性能工程国际会议论文集，*ICPE 2016*，荷兰代尔夫特，*2016*年*3*月*12*日至*16*日，Alberto Avritzer、Alexandru Iosup、Xiaoyun Zhu 和 Steffen Becker（编）。ACM，309–320。https: //doi.org/
10 1145 2851553 2851576 。 / 。Urjoshi Sinha、Mikaela Cashman 和 Myra B. Cohen。2020。使用遗传算法优
[71] 化数据驱动应用中的配置。载于基于搜索的软件工程——第*12*届国际研讨会，*SSBSE 2020*，意大利巴里，*2020*年*10*月*7*日至*8*日，会议论文集（计算机科学讲义，第*12420*卷），Aldeida Aleti 和 Annibale Panichella（编）。施普林格，
137 152 10 1007 978 3 030 59762 7_10 – 。https://doi.org/ 。 / - - - - András Vargha 和
[72] Harold D. Delaney。 。对 McGraw 和 Wong 的 CL 通用语言效应量统计的批评与改进。[73] Nadarajen Veerapen、Gabriela Ochoa、Mark Harman 和 Edmund K Burke。2015。解决单目标与双目标下一版本发布问题的整数线性规划方法。《信息与软件技术》65 (2015), 1–13。[74] Handing Wang、Markus Olhofer 和 Yaochu Jin。2017。多目标优化中的偏好建模与表达小综述：现状与挑战。《复杂与智能系统》3, 4 (2017), 233–245。[75] Jon Whittle、Peter Sawyer、Nelly Bencomo、Betty H. C. Cheng 和 Jean-Michel Bruel。2009。RELAX：将不确定性纳入自适应系统的规约。载于*RE 2009*，第*17*届*IEEE*需求工程国际会议，美国佐治亚州亚特兰大，*2009*年*8*月*31*日至*9*月*4*日。IEEE计算机学会，79–88。https://doi.org/10. 1109/RE.2009.36[76] Frank Wilcoxon。1945。基于秩检验方法的个体比较。[77] Bowei Xi、Zhen Liu、Mukund Raghavachari、Cathy H. Xia 和 Li Zhang。2004。用于应用服务器配置的智能爬山算法。载于第*13*届万维网国际会议论文集，*WWW 2004*，美国纽约州纽约，*2004*年*5*月*17*日至*20*日，Stuart I. Feldman、Mike Uretsky、Marc Najork 和 Craig E. Wills（编）。ACM，
287 296 10 1145 988672 988711 – 。https://doi.org/ 。 / 。Tianyin Xu、
[78] Long Jin、Xuepeng Fan、Yuanyuan Zhou、Shankar Pasupathy 和 Rukma Talwadker。2015。嘿，你给了我太多旋钮！：理解并处理系统软件中过度设计的配置。载于*2015*年第*10*届软件工程基础联合会议，*ESEC/FSE 2015*，意大利贝加莫，*2015*年*8*月*30*日至*9*月*4*日，Elisabetta Di Nitto、Mark Harman 和 Patrick Heymans（编）。ACM，307–319。https: //doi.org/ 10 1145 2786805 2786852 。 / 。Tao Ye 和 Shivkumar Kalyanaraman。2003。用于大规模网络参数配置
[79] 的递归随机搜索算法。载于计算机系统测量与建模国际会议论文集，*SIGMETRICS 2003*，*2003*年*6*月*9*日至*14*日，美国加利福尼亚州圣迭戈，Bill Cheng、Satish K. Tripathi、Jennifer Rexford 和 William H. Sanders（编）。ACM，
196 205 10 1145 781027 781052 – 。https://doi.org/ 。 / 。Guo Yu、Yaochu Jin 和
[80] Markus Olhofer。 。参考点还是偏好——重新思考多目标进化优化。载于*IEEE*进化计算大会，*CEC 2019*，新西兰惠灵顿，*2019*年*6*月*10*日至*13*日。IEEE，
2019
2410 2417 10 1109 2019 8790106 – 。https://doi.org/ 。 /CEC。 。
[81] 2016
Guo Yu、Jinhua Zheng、Ruimin Shen 和 Miqing Li。 。分解多目标优化中的用户偏好。《软计算》20, 10 (2016), 4005–4021。[82] Qingfu Zhang 和 Hui Li。2007。MOEA/D：基于分解的多目标进化算法。*IEEE*进化计算汇刊11, 6 (2007), 712–731。

[83] Yuqing Zhu、Jianxun Liu、Mengying Guo、Yungang Bao、Wenlong Ma、Zhuoyue Liu、Kunpeng Song 和 Yingchun Yang。2017。BestConfig：通过自动配置调优挖掘系统的性能潜力。载于*2017*年云计算研讨会论文集，*SoCC 2017*，美国加利福尼亚州圣克拉拉，*2017*年*9*月*24*日至*27*日。ACM，338–350。 https://doi.org/ 10 1145 3127479 3128605 。 / 。E. Zitzler、D. Brockhoff 和 L. Thiele。2007。[84] 重新审视超体积指标：通过加权积分设计帕累托相容指标。载于进化多准则优化国际会议。施普林格，862–876。[85] Eckart Zitzler 和 Simon Künzli。2004。多目标搜索中的基于指标的选择。载于自然界并行问题求解——*PPSN VIII*，第八届国际会议，英国伯明翰，*2004*年*9*月*18*日至*22*日，会议论文集（计算机科学讲义，第*3242*卷），Xin Yao、Edmund K. Burke、José Antonio Lozano、Jim Smith、Juan Julián Merelo Guervós、John A. Bullinaria、Jonathan E. Rowe、Peter Tiňo、Ata Kabán 和 Hans-Paul Schwefel（编）。施普林格，832–842。[86] E. Zitzler 和 L. Thiele。1998。使用进化算法进行多目标优化——一项比较案例研究。载于自然界并行问题求解国际会议（*PPSN*）论文集。292–301。[87] E. Zitzler、L. Thiele、M. Laumanns、C. M. Fonseca 和 V. G. Da Fonseca。2003。多目标优化器的性能评估：分析与综述。*IEEE*进化计算汇刊7, 2 (2003), 117–132。