浙江大学

本 科 生 毕 业 论 文

文献综述和开题报告

学生姓名　　张志心

学生学号　　3210106357

指导教师　　王灿

年级与专业　　2021级计算机科学与技术

所在学院　　计算机科学与技术学院

一、题目：temp

二、指导教师对文献综述、开题报告、外文翻译的具体要求:

指导教师（签名）__________

年　月　日

目录

一、文献综述

1 背景介绍

1.1 多处理器调度问题

1.1.1 问题的背景与基本概念

多处理器调度问题（Multiprocessor Scheduling Problem）的研究始于计算机系统中对计算资源的有效利用。早期的计算机系统多为单核处理器，任务的执行通常由操作系统进行单核调度。随着多核处理器的普及，计算任务可以在多个处理器上并行执行，从而显著提高计算效率和处理能力。然而，任务之间可能存在依赖关系，或者每个任务的处理时间不同，如何合理地分配任务到各个处理器，最大限度地减少完成时间或优化其他目标，成为了多处理器调度问题的核心。

多处理器调度问题涉及在多个处理器之间合理地分配和调度一组作业或任务。任务之间可能是独立的，也可能存在依赖关系，例如某些任务必须在其他任务完成后才能开始。调度的目标通常包括最小化最大完成时间（Makespan）、最小化任务的平均完成时间、或者最大化系统的吞吐量等。

多处理器调度问题在许多实际应用中都有广泛的应用场景。如：高性能并行计算、实时系统、操作系统需要实时处理多个可并行进行的任务的场景。

我们可以建立这样一个简单的数学模型来描述多处理器调度问题。

定义 1.1 (朴素多处理器调度问题)**.** 一个实例 I 包含一组 n 个作业 $J=\{J_1,J_2,\cdots,J_n\}$，每个作业都要在 m 台并行相同的机器集 $M=\{M_1,M_2,\cdots,M_m\}$ 上非抢占执行。每个作业有一个确定的处理时间上界 p_j。目标是最小化完工时间 C_{max}，即所有任务中最后一个完成的任务的完成时间。

多处理器调度问题有很多变种。例如：最小化任务的完成时间之和、最小化任务的等待时间之和、在限制时间内最大化完成任务的数量等。还有一些变种

问题是任务之间存在一些依赖关系，某些任务必须在其他任务完成之后才能开始。而“非抢占执行”的含义是每个任务只要开始进行就不能中断，与之相对的是“抢占执行”，即任务可以中断并立即切换为另一个任务开始执行。

多处理器调度问题是一个 NPC 问题，即不存在多项式复杂度的确定性求解算法。目前研究者们已经设计出确定性算法如启发式搜索，非确定性算法如遗传算法、模拟退火算法等。

1.1.2 带测试的多处理器调度问题

带测试的多处理器调度问题（Testing Multiprocessor Scheduling Problem）是多处理器调度问题的一个扩展，其中除了通常的任务调度外，还需要进行额外的测试步骤。这些测试步骤用于评估任务的特性（如任务的处理时间、执行顺序等），从而帮助调度器更准确地做出调度决策。与传统的调度问题不同，带测试调度问题需要在优化任务调度的同时考虑测试操作的开销和影响。

带测试的多处理器调度问题一般包含两个重要元素：测试阶段和调度阶段。在测试阶段，系统会对任务进行某些形式的检测或预处理（如任务的处理时间、执行顺序等）。这些测试通常需要消耗时间和计算资源，因此测试的时间需要纳入整个调度优化过程中。

我们可以从上面的实际问题中抽象出一个数学模型：

定义 1.2 (带测试多处理器调度问题)**.** 一个实例 I 包含一组 n 个作业 $J=\{J_1,J_2,\cdots,J_n\}$，每个作业都要在 m 台并行相同的机器集 $M=\{M_1,M_2,\cdots,M_m\}$ 上非抢占执行。每个作业带有一个处理时间上界 u_j 和一个测试操作的时长 t_j，如果进行测试，作业将得到一个实际运行时长 p_j。每一个作业 J_j 都可以选择在某台机器上花 u_j 时间运行，或者先花 t_j 时间进行测试，之后在同一台机器上运行 p_j 时间。目标是最小化完工时间 C_{max}。

当 $t_j=0$ 时，问题就被归约为朴素多处理器调度问题。因此，带测试多处理器调度问题也至少是 NPC 问题，不存在多项式时间复杂度的算法。

1.2 在线、半在线问题和竞争比

1.2.1 在线、半在线带测试多处理器调度问题

在线和半在线多处理器调度问题是多处理器调度领域中的两个重要分支。它们在任务到达时机和调度决策上有所不同，主要涉及如何在信息不完全的情况下，合理调度多个处理器以优化系统性能。随着多核处理器和并行计算的普及，在线和半在线调度问题在实际应用中有着广泛的影响。

定义 1.3 (在线带测试多处理器调度问题)**.** 当一个任务到达时，调度器只能根据当前的信息进行调度，而无法对未来的任务做出预测。因此调度器需要在每个任务到达时就立即决定其处理器分配和执行顺序。

定义 1.4 (半在线带测试多处理器调度问题)**.** 介于在线和离线调度问题之间。在半在线调度中，调度器可以在任务到达之前得知全部的 t_i 和 u_i，但只有在花费 t_i 进行测试之后才可得知 p_i。

与离线算法不同，在线算法无法在一开始就拥有全部输入数据，而是必须随着输入的逐步到来，作出即时决策。由于无法预知未来的输入数据，在线算法通常面临一个重要问题：如何在信息不完全的情况下做出近似最优的决策。

1.2.2 竞争比

为了衡量在线算法的表现，竞争比（Competitive Ratio）是一个关键的概念。它用于比较在线算法的性能与最佳离线算法（或最优算法）在同样问题中的表现差异。

定义 1.5. 在线算法的竞争比是一个用于度量其性能的度量值。它定义为在线算法执行的最坏情况性能与离线算法执行的最优性能之间的比值。具体来说，假设有一个在线调度问题，其实例为 I，并记所有实例的集合为 $\mathscr{I}$，定义

- $C(I)$ 是某个离线算法的最优解，即假设我们能够提前知道所有任务的输入，可以通过全局优化来得到的最优解。

- $C^*(I)$ 是在线算法在最坏情况下的解。

则在线算法的竞争比 α 可以表示为

$$\alpha = \sup_{I \in \mathscr{I}} \frac{C(I)}{C^*(I)}$$

竞争比总是不小于 1 的，其实际意义在于：

1. 衡量在线算法的优劣：竞争比为评估在线算法的性能提供了一个标准。它告诉我们，在最坏的情况下，在线算法的表现相比于最优解有多差。例如，某些算法的竞争比是常数（如 3-竞争），这意味着无论输入数据如何变化，算法的执行时间或完成度始终不会超过最优解的 3 倍。

2. 适应性与实时性：在线算法的竞争比越低，说明它能在实时情况下做出更接近最优的决策。例如，在云计算和数据流处理等实时计算中，能够提供一个低竞争比的在线调度算法，对于高效利用计算资源、减少延迟至关重要。

3. 算法的鲁棒性：在面对不同类型的输入时，竞争比高的算法可能在某些场景下表现不佳。通过研究竞争比，我们能够了解在不同场景下算法的鲁棒性，进而选择更适合特定应用的算法。

1.2.3 竞争比下界

在线算法的竞争比下界是指在特定问题中，任何在线算法在最坏情况下的性能相对于最优解的比例所能达到的最小值。了解这一下界具有重要的理论意义，因为它为我们提供了一个不可逾越的性能界限，帮助我们理解在线算法的最佳表现。具体来说，如果一个问题的竞争比下界是 α_0，则意味着无论我们如何设计在线算法，算法在最差情况下的表现都至少是最优解的 α 倍。

研究一个问题的竞争比下界有时是比研究它的具体算法更有效的。具体来说，它可以

1. 揭示一个问题在最坏情况下的不可逾越的性能界限。

2. 帮助评估算法的实际应用性能：竞争比下界表示了在线算法在最坏情况中的最大劣化程度，而通过与实际算法的对比，我们可以判断一个在线算法是否能在实际应用中达到一个“可接受”的水平。

3. 为算法设计提供指导：了解竞争比下界有助于为在线算法的设计提供理论依据。如果某个问题的竞争比下界较高，那么即使在设计过程中对算法进行大量优化，期望算法的性能会接近最优解也是不现实的。相反，如果下界较低，那么设计一个接近最优的在线算法是可能的。

2 国内外研究现状

我们在本文中研究完全在线的多处理器调度与测试问题[1-4]，并从随机化算法的角度进行研究。多处理器调度[5]是最早的已知 NP 难组合优化问题之一，在过去几十年中得到了广泛研究。多处理器调度的一个实例 I 包含一组 n 个作业 $J = \{J_1, J_2, \ldots, J_n\}$，每个作业都要在一组 m 台并行的相同机器 $M = \{M_1, M_2, \ldots, M_m\}$ 上非抢占地执行，目标是最小化最大完工时间 $C_{\max}$，即最大作业完成时间。与经典设置不同的是，在调度与测试中，每个作业 J_j 都有一个处理时间上界 u_j，并且有一个长度为 t_j 的测试操作，但其处理时间 p_j 直到作业被测试后才会揭示。作业 J_j 可以在其中一台机器上执行 u_j 时间，或者选择先进行测试，测试时间为 t_j，然后立即执行 p_j 时间。若所有作业都在时间零到达，则多处理器调度与测试是一个半在线问题，表示为 $P|t_j, 0 \le p_j \le u_j|C_{\max}$。本文研究的是完全在线问题，其中作业按顺序到达，在作业到达时需要做出测试决策，并且指定测试和/或执行的机器，表示为 $P|\text{online}, t_j, 0 \le p_j \le u_j|C_{\max}$。显然，半在线是完全在线的特例，在这两种情况下，调度器都应该利用已知的作业信息，在作业到达时决定是否进行测试，以便最好地平衡由于未知处理时间所消耗的总时间。

给定一个多项式时间的确定性算法，针对半在线或完全在线问题，令 $C(I)$ 为该算法在实例 I 上产生的最大完工时间，而 $C^*(I)$ 为最优离线调度的最大完工时间。算法的性能通过竞争比来衡量，定义为 $\sup_I\{C(I)/C^*(I)\}$，其中 I 遍历所有问题实例，算法称为 $\sup_I\{C(I)/C^*(I)\}$-竞争的。切换到随机化算法时，我

们相应地收集其在实例 I 上的期望最大完工时间 $E[C(I)]$，该随机化算法称为 $\sup_I\{E[C(I)]/C^*(I)\}$-竞争的。对于在线问题，随机化算法有时可以更好地处理不确定性，从而导致比最好的确定性算法更低的期望竞争比。我们为 $P|\text{online},t_j,0 \leq p_j \leq u_j|C_{\max}$ 提出了这样的一个随机化算法，并且进一步证明，当只有两台机器时，其期望竞争比严格小于任何确定性算法已证明的竞争比下界。在我们的问题中，作业处理是非抢占的。

在文献中，研究人员还考虑了抢占式作业处理[1-4]，其中任何测试或执行操作都可以被中断并稍后恢复，或者考虑了更为受限的测试抢占式变体[4]，在该变体中，已测试作业的测试和执行操作是非抢占的，但执行操作不必立即跟随测试操作，且不一定发生在同一台机器上。此外，我们的目标是最小化最大完工时间 $C_{\max}$，即最小最大目标；而另一个重要目标是最小化总作业完成时间，或者最小和目标，这也是受到了许多研究的关注[1-2,4]。

2.1 全在线问题研究进展

现有的针对完全在线问题 $P|\text{online},t_j,0 \leq p_j \leq u_j|C_{\max}$ 的近似算法，无论是确定性算法还是随机化算法，都不多。我们首先区分一个特殊的情况，其中所有的测试操作时间均为单位时间，即对于每个作业 J_j，有 $t_j = 1$，我们称之为均匀测试情况[1-2,4]，表示为 $P|\text{online},t_j = 1,0 \leq p_j \leq u_j|C_{\max}$。注意，在一般的测试情况下，测试时间可以是任何非负值。当只有一台机器时，机器上的作业处理顺序与最大完工时间无关。这表明完全在线问题和半在线问题是相同的。第一组结果是关于半在线均匀测试问题 $P_1|t_j = 1,0 \leq p_j \leq u_j|C_{\max}$，由 D¨urr 等人[1-2]提出。他们提出，当 $u_j \geq \varphi = \frac{\sqrt{5}+1}{2}$ 时测试作业 J_j，或者以概率 $f(u_j) = \max\left(0, \frac{u_j(u_j-1)}{u_j(u_j-1)+1}\right)$ 测试它，从而得到了一个确定性的 φ-竞争算法和一个期望的 4/3 竞争算法。令 $r_j = \frac{u_j}{t_j}$；Albers 和 Eckl[3]将上述两种算法扩展到一般测试情况下 $P_1|t_j,0 \leq p_j \leq u_j|C_{\max}$，即当 $r_j \geq \varphi$ 时测试作业 J_j，或者以概率 $f(r_j)$ 测试它，分别达到了相同的竞争比和期望竞争比。作者们[1-3]证明了这两种算法是最优的，即 φ 是任何确定性算法的竞争比下界，而 4/3 是任何随机化算法的期望竞争比下界，这一结果由 Yao 原

理[6]得出。

当至少有两台机器时,完全在线问题比半在线问题更为一般。Albers 和 Eckl[4]提出了在列表调度规则[7]中测试作业 J_j，即当 $r_j \geq \varphi$ 时，按照该规则将每个作业分配到最空闲的机器上进行处理（可能是测试后执行)。他们证明了这样的算法是 $\varphi(2-\frac{1}{m})$-竞争的，且该分析是紧的，其中 m 是机器的数量。他们还证明了，即使在均匀测试情况下 $P|\text{online},t_j=1,0\leq p_j\leq u_j|C_{\max}$ 中，任何确定性算法的竞争比下界为 2[4]，并且对于两台机器的一般测试情况 $P_2|\text{online},t_j,0\leq p_j\leq u_j|C_{\max}$，即当只有两台机器时，其下界为 2.0953，这一结果也稍微更好[4]。

2.2 半在线问题研究进展

当只有一个机器时，半在线问题与全在线问题等价。

当至少有两台机器时，因为所有作业都在时间零到达，因此调度器可以利用所有已知的 u_j 和 t_j 值，不仅用于作业测试决策，还可以形成某些作业处理顺序，以更好地减少最大完工时间。事实上，这正是 Albers 和 Eckl[4] 在一般测试情况下提出的开创性所谓 SBS 算法的情况，该算法在机器数量趋向无穷大时为 3.1016-竞争算法。对于均匀测试情况，Albers 和 Eckl[4] 还提出了一个 3-竞争算法，并给出了对于任何确定性算法的竞争比下界 $\max\{\varphi,2-\frac{1}{m}\}$。

上述两个竞争比在一般和均匀测试情况下分别由 Gong 和 Lin[8] 改进为 2.9513 和 2.8081，最新的竞争比是 Gong 等人[9] 分别提出的 2.8019 和 2.5276。

为了证明竞争比下界 $2-\frac{1}{m}$，Albers 和 Eckl[4] 提出了一个 $P|t_j=1,0\leq p_j\leq u_j|C_{\max}$ 的实例，迫使任何确定性算法对所有作业进行测试。这意味着该下界不仅适用于任何确定性算法的竞争比，也适用于任何随机化算法的期望竞争比，依据 Yao 原理[6]。此外，$2-\frac{1}{m}$ 的下界因此适用于完全在线均匀测试问题 $P|\text{online},t_j=1,0\leq p_j\leq u_j|C_{\max}$ 中任何随机化算法的期望竞争比。

2.3 其他相关工作

Dürr 等人[1-2] 为单机均匀测试问题 $P_1|t_j=1,0\le p_j\le u_j|\sum_j C_j$ 提出了一个期望竞争比为 1.7453 的随机化算法，该算法按均匀随机顺序测试待测作业。他们还证明了期望竞争比的下界为 1.6257。Albers 和 Eckl[3] 为一般测试情况 $P_1|t_j,0\le p_j\le u_j|\sum_j C_j$ 设计了一个随机化算法，其中作业 J_j 根据比率 r_j 使用一个相当复杂的概率函数进行测试；该算法的期望竞争比为 3.3794。

回顾经典在线多处理器调度问题 $P|\text{online}|C_{\max}$ 的最先进的随机化算法，其中作业依次到达，处理时间 p_j 在作业 J_j 到达时揭示，并且在到达时必须无条件地分配作业到某台机器上执行，以最小化最大完工时间。众所周知，对于这个问题，从第一个确定性算法列表调度[7] 的竞争比 2 到当前最好的 1.9201[10]，经历了很长时间的改进。对于每个在线近似算法，性能分析中最具挑战性的一部分是精确估计最优离线最大完工时间。三个最典型的下界是 $\frac{1}{m}\sum_{j=1}^{n}p_j$（平均机器负载）、$\max_{j=1}^{n}p_j$ （最大的作业处理时间）和 $p[m]+p[m+1]$，其中 $p[j]$ 是所有作业中的第 j 大处理时间。Albers[11] 证明，如果只使用上述三个下界，则任何确定性算法的竞争比在 m 趋于无穷大时都不能小于 1.919。

Albers[11] 提出了一个期望竞争比为 1.916 的随机化算法，并且在性能分析中仅使用了上述三个下界来估计最优离线最大完工时间。更详细地说，Albers 提出了两个确定性算法 A_0 和 A_1，并提议以 0.5 的概率执行每个算法。作者表明，如果期望最大完工时间与这三个下界相比过大，即 A_0 和 A_1 都产生了较大的最大完工时间，那么在实例中会有许多大作业，使得最优离线最大完工时间无法太小。

这种由常数数量的确定性算法组成的随机化算法被称为“几乎随机化算法”[12]，它利用其优秀的组件算法，在至少一个组件算法表现良好时，它能够表现良好，而如果没有任何组件算法表现良好，那么最优离线最大完工时间也会远离下界。这个设计思想已被用于逼近许多其他优化问题，并取得了突破性的成果，例如其他调度问题[9,13] 和 k-服务器问题[14]。事实上，对于在线多处理器调度问题 $P|\text{online}|C_{\max}$，在[11] 之前，Seiden[13] 修改了 Bartal 等人[15] 和 Seiden[16] 的非几乎随机化算法，这些算法将当前作业分配给两台负载最小的机器之一，并

以一定的概率运行，变成了几乎随机化算法，这是一个 k 个确定性算法的均匀分布，其中 k 是算法内要创建的调度的数量，并且选择足够大以保证期望竞争比。尽管这种方法对小 m（特别是 $m \leq 7$）有效，Albers[11] 观察到[13,15-16] 中算法的分析方法在一般的大 m 情况下不起作用，随后提出了基于仅两个确定性算法的新几乎随机化算法。

2.4 存在问题

现有的针对全在线问题的竞争比下界的研究主要还停留在特殊情况（$m = 1$）和小范围情况（$m \leq 2, n \leq 3$）。研究方法也局限在人工构造博弈决策树，当决策树的层数更深（对应 n 更大）或点数更多（对应 m 更大）时，人工构造就已经很难奏效。事实上，对于 $m \geq 3, n \geq 4$ 的情形，现有的研究都只能仿照 $m = 2, n = 3$ 的构造方式尝试构造较坏情况的数据，这导致目前的竞争比下界和竞争比上界（即目前已经存在的在线算法的竞争比）仍有较大差距。

3 研究展望

由上可见，竞争比下界仍可能有较大的提升空间。因此设计一个通用的建立博弈搜索树的算法是有必要的。借助计算机的强大算力，我们可以扩大搜索范围，发现人力无法观察到的极端数据，进一步提升竞争比下界。

4 参考文献

[1] DÜRR C, ERLEBACH T, MEGOW N, et al. Scheduling with explorable uncertainty[C]// 9th Innovations in Theoretical Computer Science Conference (ITCS 2018). 2018.

[2] DÜRR C, ERLEBACH T, MEGOW N, et al. An adversarial model for scheduling with testing[J]. Algorithmica, 2020, 82(12): 3630-3675.

[3] ALBERS S, ECKL A. Explorable uncertainty in scheduling with non-uniform testing times[C] //Approximation and Online Algorithms: 18th International Workshop, WAOA 2020, Virtual Event, September 9–10, 2020, Revised Selected Papers 18. 2021: 127-142.

[4] ALBERS S, ECKL A. Scheduling with testing on multiple identical parallel machines[C]// Algorithms and Data Structures: 17th International Symposium, WADS 2021, Virtual Event, August 9–11, 2021, Proceedings 17. 2021: 29-42.

[5] GAREY MICHAEL R, JOHNSON DAVID S. Computers and Intractability: A guide to the theory of NP-completeness[Z]. 1979.

[6] YAO A C C. Probabilistic computations: Toward a unified measure of complexity[C]//18th Annual Symposium on Foundations of Computer Science (sfcs 1977). 1977: 222-227.

[7] GRAHAM R L. Bounds for certain multiprocessing anomalies[J]. Bell system technical journal, 1966, 45(9): 1563-1581.

[8] GONG M, LIN G. Improved approximation algorithms for multiprocessor scheduling with testing[C]//International Workshop on Frontiers in Algorithmics. 2021: 65-77.

[9] GONG M, CHEN Z Z, LIN G, et al. Randomized algorithms for fully online multiprocessor scheduling with testing[J]. ArXiv preprint arXiv:2305.01605, 2023.

[10] FLEISCHER R, WAHL M. On-line scheduling revisited[J]. Journal of Scheduling, 2000, 3(6): 343-353.

[11] ALBERS S. On randomized online scheduling[C]//Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. 2002: 134-143.

[12] REINGOLD N, WESTBROOK J, SLEATOR D D. Randomized competitive algorithms for the list update problem[J]. Algorithmica, 1994, 11(1): 15-32.

[13] SEIDEN S. Barely random algorithms for multiprocessor scheduling[J]. Journal of Scheduling, 2003, 6(3): 309-334.

[14] BARTAL Y, CHROBAK M, LARMORE L L. A randomized algorithm for two servers on the line[J]. Information and Computation, 2000, 158(1): 53-69.

[15] BARTAL Y, FIAT A, KARLOFF H, et al. New algorithms for an ancient scheduling problem[C]//Proceedings of the twenty-fourth annual ACM symposium on Theory of computing. 1992: 51-58.

[16] SEIDEN S S. Online randomized multiprocessor scheduling[J]. Algorithmica, 2000, 28(2): 173-216.

二、开题报告

1 问题提出的背景

1.1 背景介绍

1.1.1 多处理器调度问题

多处理器调度问题（Parallel Machine Scheduling Problem）[1] 是生产调度领域中的一个经典问题，它主要关注如何在 m 台相同的机器上合理安排 n 个任务的执行顺序，以达到某些特定的目标，比如最小化所有任务完成的时间总和、最小化最后一个任务的完成时间等。形式化地，这个问题可以描述为：

定义 1.1. 朴素多处理器调度问题：一个实例 I 包含一组 n 个作业 $J = \{J_1, J_2, \cdots, J_n\}$，每个作业都要在 m 台并行相同的机器集 $M = \{M_1, M_2, \cdots, M_m\}$ 上非抢占执行。每个作业有一个确定的处理时间上界 p_j。目标是最小化完工时间 C_{max}，即所有任务中最后一个完成的任务的完成时间。

然而在实际问题中，作业的实际完工时间可能需要测试才能得出。由此可以抽象出多处理器调度问题的一个变体[2][3][4]：

定义 1.2. 带测试多处理器调度问题：一个实例 I 包含一组 n 个作业 $J = \{J_1, J_2, \cdots, J_n\}$，每个作业都要在 m 台并行相同的机器集 $M = \{M_1, M_2, \cdots, M_m\}$ 上非抢占执行。每个作业带有一个处理时间上界 u_j 和一个测试操作的时长 t_j，如果进行测试，作业将得到一个实际运行时长 p_j。每一个作业 J_j 都可以选择在某台机器上花 u_j 时间运行，或者先花 t_j 时间进行测试，之后在同一台机器上运行 p_j 时间。目标是最小化完工时间 C_{max}。

这类问题广泛存在于制造业、云计算资源分配、物流运输等多个行业背景中。例如，在制造业中，工厂需要决定将哪些工件分配给哪几台机器进行加工以

及加工的先后顺序；在云计算环境中，需要决定如何高效地为虚拟机分配计算资源。解决多处理器调度问题对于提高生产效率、降低成本具有重要意义。针对不同场景下的具体需求，研究者们提出了多种算法来寻找最优解或近似解，有确定性算法例如启发式算法；非确定性算法（随机算法）例如遗传算法、模拟退火算法等。

1.1.2 在线问题和竞争比

在实际问题中，我们往往不能在一开始就得到所有数据，而只能随着决策的过程逐步获取数据。这就引出了离线问题、半在线问题和在线问题的概念。

定义 1.3. 离线带测试多处理器调度问题: 所有作业在时间零时刻到达，且 t_j, p_j, u_j 全部事先已知。此时问题等价于处理时间上界为 $\min(t_j + p_j, u_j)$ 的朴素多处理器调度问题。

定义 1.4. 半在线带测试多处理器调度问题： 所有作业在时间零时刻到达，记作 $P \mid t_j, 0 \le p_j \le u_j \mid C_{max}$。

定义 1.5. 在线带测试多处理器调度问题： 调度器需要在作业到达时做出测试决策，并指定机器进行测试或直接执行，每个作业在上一个作业的调度决策完成后到达，记作 $P \mid online, t_j, 0 \le p_j \le u_j \mid C_{max}$。

由于信息的缺失，在一般情况下我们无法设计出能够确定地作出最优决策的算法。因此只能退而寻求近似算法。我们用竞争比的概念来描述算法的性能:

定义 1.6. 确定性算法的竞争比： 假设存在一个确定性算法，令 $C(I)$ 表示该算法在实例 I 上产生的完工时间，$C^*(I)$ 表示最优离线调度的完工时间。定义该算法的竞争比为

$$\alpha = \sup_I \frac{C(I)}{C^*(I)}, \quad \text{其中}I\text{遍历所有问题实例},$$

该算法被称为 α-竞争算法。

提升竞争比下界对于设计在线和半在线算法具有重要的研究意义：它可以为在线算法提供一个性能保障，即使在最不利的情况下，也能保证算法的表现不会太差；它还有助于指导算法的设计和改进，如果一个在线算法的竞争比已经接近或达到已知的下界，那么进一步优化的空间可能非常有限。

1.1.3 竞争比下界的现有成果

定理 1.7. *对于问题 $P2 \mid online, t_j, 0 \le p_j \le u_j \mid C_{max}$，任何确定性算法的竞争比大于 2.2117。*[4]

证明. 一个可行的构造方案如下：定义 $\varphi = \frac{\sqrt{5}+1}{2}$（事实上，可以定义 φ 为 $(1.5,2)$ 的任意实数。）令 $t_i = 1, \forall i$。有两台机器，令其为 M_1, M_2，三个工件为 J_1, J_2, J_3。令 $u_1 = \varphi$，将 J_1 在 M_1 上加工。

Case1: 若对 J_1 测试，此时 $p_1 = \varphi$，令 $u_2 = \varphi$。

Case1.1: 若对 J_2 测试，此时 $p_2 = \varphi$。如果将 J_2 放在 M_1 上加工，此时 $C = 2+2\varphi$，而 $C^* = \varphi$，$C/C^* = 2\varphi+2/\varphi > 2$。如果将 J_2 放在 M_2 上加工，令 $u_3 = 2\varphi$，若 J_3 测试，此时 $p_3 = 2\varphi$，则此时 $C \ge 2+3\varphi$，而 $C^* = 2\varphi$，$C/C^* = 2+3\varphi/2\varphi > 2$。若 J_3 不测试，此时 $p_3 = 0$，则此时 $C \ge 1+3\varphi$，而 $C^* = 1+\varphi$，$C/C^* = 1+3\varphi/1+\varphi > 2$。

Case1.2: 若对 J_2 不测试，此时 $p_2 = 0$。如果将 J_2 放在 M_1 上加工，此时 $C = 2\varphi+1$，而 $C^* = \varphi$，$C/C^* = 2\varphi+1/\varphi > 2$。如果将 J_2 放在 M_2 上加工，令 $u_3 = 1+\varphi$，若 J_3 测试，此时 $p_3 = 1+\varphi$，则此时 $C \ge 2+2\varphi$，而 $C^* = 1+\varphi$，$C/C^* = 2+2\varphi/1+\varphi = 2$。若 J_3 不测试，此时 $p_3 = 0$，则此时 $C \ge 1+2\varphi$，而 $C^* = 2$，$C/C^* = 1+2\varphi/2 > 2$。

Case2: 若不对 J_1 测试，此时 $p_1 = 0$，令 $u_2 = \varphi$。

Case2.1: 若对 J_2 测试，此时 $p_2 = \varphi$。如果将 J_2 放在 M_1 上加工，此时 $C = 1+2\varphi$，而 $C^* = \varphi$，$C/C^* = 2\varphi+1/\varphi > 2$。如果将 J_2 放在 M_2 上加工，令 $u_3 = 1+\varphi$，若 J_3 测试，此时 $p_3 = 1+\varphi$，则此时 $C \ge 2+2\varphi$，而 $C^* = 1+\varphi$，$C/C^* = 2+2\varphi/1+\varphi = 2$。若 J_3 不测试，此时 $p_3 = 0$，则此时 $C \ge 1+2\varphi$，而 $C^* = 2$，$C/C^* = 1+2\varphi/2 > 2$。

Case2.2: 若对 J_2 不测试，此时 $p_2 = 0$。如果将 J_2 放在 M_1 上加工，此时 $C = 2\varphi$，而 $C^* = 1$，$C/C^* = 2\varphi/1 > 2$。如果将 J_2 放在 M_2 上加工，令 $u_3 = 1+\varphi$，若 J_3 测

试，此时 $p_3 = 1+\varphi$，则此时 $C \geq 2+2\varphi$，而 $C^* = 1+\varphi$，$C/C^* = 2+2\varphi/1+\varphi = 2$。若 J_3 不测试，此时 $p_3 = 0$，则此时 $C \geq 1+2\varphi$，而 $C^* = 2$，$C/C^* = 1+2\varphi/2 > 2$。

$\square$

对于所有测试操作时间都为单位时间的问题 $P \mid \text{online}, t_j = 1, 0 \leq p_j \leq u_j \mid C_{\max}$：当只有一台机器时，Dühr 等[2][3]提出，当 $u_j \leq \varphi := \frac{\sqrt{5}+1}{2}$ 时，对作业 j 进行测试，可以得到 φ-竞争算法；或者以概率 $f(u_j) = \max\left(0, \frac{u_j(u_j-1)}{u_j(u_j-1)+1}\right)$ 概率对作业进行测试，可以得到 $\frac{4}{3}$-竞争的随机化算法。论文[5]，这两种算法是最优的，即 φ 是确定性算法的竞争比下界，$\frac{4}{3}$ 是任何随机化算法的期望竞争比下界。当有 $m \geq 2$ 台机器时，存在 $\varphi(2-\frac{1}{m})$-竞争算法[6]。论文[4] 中提到了，对于任何确定性算法，竞争比下界为 2。该下界的证明使用了两台机器、三个工件来构造最坏实例，具体构造方法可以参考定理1.7 的证明。另外，在 Gong 的论文[7]中，进一步证明了任何确定性算法的竞争比下界为 2.2117；同时还提出了一种确定性算法集成的随机化算法，其期望竞争比为 $\frac{3\varphi+3\sqrt{13-7\varphi}}{4} \approx 2.1839$。

1.2 本研究的意义和目的

多处理器调度问题作为一个重要的组合优化问题，其在实际生产环境中具有广泛的应用场景，如工业生产和资源分配等领域。带测试的多处理器调度问题引入了相同的工作的不同处理方式（一种未知效果的新处理手段），并以提高整体系统的效率为目标，同样有着重要的研究意义。

目前在线带测试多处理器调度问题竞争比下界还有较大的提升空间。现有的研究主要集中在机器数量、工作数量较少的情况下，使用手动构造最坏情况来分析竞争比下界，但对于更一般的情形，特别是当 $m \geq 3, n \geq 4$ 时，现有的竞争比下界仍有改进余地。

通过引入计算机辅助方法，利用程序设计方法更高效地构建和分析决策树，可以扩大搜索范围，有希望进一步提高竞争比下界。

2　项目的主要内容和技术路线

2.1　主要研究内容

本研究旨在探索并证明在线带测试的多处理器调度问题的确定性算法竞争比下界。即：证明不存在竞争比总是低于某个下界的确定性算法。

我将对问题进行程序建模，通过分析特定情况下的竞争比表现，识别出影响竞争比的关键因素。在此基础上，我还将对现有算例进行分析，特别关注那些导致竞争比达到最坏的特殊情况。最后，我将运用计算机辅助的方法在有限的时间内搜索出更广泛的实例，以验证该问题的竞争比下界，并通过严谨的分析来确认最终结果。

2.2　技术路线

我将尝试沿用 Gormley 等设计的线性在线问题的竞争比下界求解模型[8]，将其应用到在线带测试的多处理器调度问题的竞争比下界的研究中。依据该论文的思路，我将先建立传统的博弈搜索模型，将调度的过程视为调度算法与对手的 n 轮博弈过程。每一轮博弈分为以下三步：第一步对手指定当前任务的 t_i 和 u_i；第二步算法为当前输入的任务分配一个处理器并决定是否测试; 第三步若算法决定测试则对手还要指定当前任务的 p_i。算法的最终目的是达到预设的竞争比 α，而对手的最终目的则是阻止算法达到该竞争比。如果最终的博弈结果是对手获胜，则证明无论如何决策，总存在一个实例使竞争比大于 α，即竞争比下界大于 α。

传统的博弈搜索模型将使用搜索剪枝的方法，暴力搜索所有可能的数据并剪去目前已不可能达到最优解的数据。Gormley 的模型[8]则考虑到了调度问题的“线性性”，即博弈搜索树的构建可视为一个带 max、min 运算的线性规划模型，进而通过拆分 max 和 min 运算得到多个传统线性规划模型。从而一方面可以直接利用线性规划算法求解，另一方面也可以通过线性规划的解的性质，将解的范围限制在有限的空间内，对对手的决策进行优化来辅助剪枝。

最后，我将用 C++ 语言实现上述最优化模型和搜索框架，先对 $m = 1$ 和 $m = 2$ 的具有较为成熟的情形验证框架的准确性，再探索 $m = 3, n \geq 4$ 更精确的竞争比下界。

2.3 可行性分析

现有的对竞争比下界的研究大多基于手工构造特殊数据。这对 $m \geq 3, n \geq 4$ 的情况已经很难奏效。因此借助计算机的强大算力构造人力难以搜索到的数据来提升竞争比下界是大有可为的。

此外，Gormley[8] 的论文中引用了一个定理:

定理 2.1. 若在线问题是线性的，给定任意实数 $\alpha > 1$ 和整数 n，存在一个算法，可在有限时间内构造出 n 层的博弈树证明在线问题的竞争比下界不小于 α，或证明竞争比下界大于 α。

这个定理保证了算法的可停机性。然而，由于直接拆解最优化问题会导致子问题的指数级增加，我们仍需设计一些搜索剪枝的操作以提高程序上运行效率，使其能在有限的时间内寻找到具有更大竞争比下界的实例。

3 研究计划进度安排及预期目标

3.1 进度安排

1. 2025.2.28 以前，完成问题建模和文献调研，归纳现有方法。完成三合一等准备工作。
2. 2025.3.15 以前，设计计算机搜索的代码并初步验证现有研究成果。
3. 2025.3.31 以前，通过进一步的理论分析，确定搜索剪枝策略。
4. 2025.4.15 以前对剪枝策略和现有运行结果，进行深入理论分析和证明，进一步改进竞争比下界。

5. 2025.5 以前，撰写毕业论文，持续尝试改进搜索策略，以得到更好的结果和获得更快的程序验证效率。

3.2 预期目标

我希望能够深入分析多处理器调度问题的数学模型，基于对于构建决策者和问题之间的博弈形成的决策树模型，探索在调度问题中形成的算法策略，发掘出竞争比低的在线算法。同时，通过使用数学工具和计算机辅助的方式，进行广泛的实例模拟，以验证现有研究成果中的竞争比下界。同时，我希望在原有的研究成果基础上，通过计算机进行高效的剪枝搜索，进一步提高机器数量 ≥ 3，工件数量 ≥ 4 的期望竞争比的下界。

4 参考文献

[1] GARY M R, JOHNSON D S. Computers and Intractability: A Guide to the Theory of NP-completeness[Z]. 1979.

[2] DÜRR C, ERLEBACH T, MEGOW N, et al. Scheduling with explorable uncertainty[C]// 9th Innovations in Theoretical Computer Science Conference (ITCS 2018). 2018.

[3] DÜRR C, ERLEBACH T, MEGOW N, et al. An adversarial model for scheduling with testing[J]. Algorithmica, 2020, 82(12): 3630-3675.

[4] ALBERS S, ECKL A. Scheduling with testing on multiple identical parallel machines[C]// Algorithms and Data Structures: 17th International Symposium, WADS 2021, Virtual Event, August 9–11, 2021, Proceedings 17. 2021: 29-42.

[5] ALBERS S, ECKL A. Explorable uncertainty in scheduling with non-uniform testing times[C] //Approximation and Online Algorithms: 18th International Workshop, WAOA 2020, Virtual Event, September 9–10, 2020, Revised Selected Papers 18. 2021: 127-142.

[6] GRAHAM R L. Bounds for certain multiprocessing anomalies[J]. Bell system technical journal, 1966, 45(9): 1563-1581.

[7] GONG M, CHEN Z Z, LIN G, et al. Randomized algorithms for fully online multiprocessor scheduling with testing[J]. ArXiv preprint arXiv:2305.01605, 2023.

[8] GORMLEY T, REINGOLD N, TORNG E, et al. Generating adversaries for request-answer games[C]//Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms. 2000: 564-565.

三、外文翻译: 在线带测试多处理器调度问题的随机算法

论文标题：Randomized algorithms for fully online multiprocessor scheduling with testing

作者：Mingyang Gong 等

摘要

我们贡献了第一个随机化算法，该算法是任意多个确定性算法在完全在线多处理器调度与测试问题中的集成。当只有两个机器时，我们展示了通过两个组件算法，其期望竞争比已经严格小于已证明的最佳确定性竞争比下界。这种算法结果在文献中极为罕见。多处理器调度是最早得到大量研究的组合优化问题之一。最近，多个研究小组研究了其测试变体，在该变体中，每个作业 J_j 具有一个处理时间的上界 u_j 和一个长度为 t_j 的测试操作；可以选择执行作业 J_j 的最大处理时间 u_j，或者选择对 J_j 进行测试，测试时间为 t_j，以获得确切的处理时间 p_j，然后立即执行该作业的 p_j 时间。我们的目标问题是完全在线多处理器调度与测试问题，其中作业依次到达，因此需要在作业到达时做出测试决策以及分配指定的机器。我们首先利用 Yao 原理证明了至少三台机器情况下和仅两台机器情况下随机算法的期望竞争比下界分别为 1.6682 和 1.6522，并提出了一个期望竞争比为 $(\sqrt{\varphi'+3}+1)(\approx 3.1490)$-竞争的随机算法，该算法是通过一个非均匀概率分布集成了任意多个确定性算法，其中 $\varphi' = \frac{\sqrt{5}+1}{2}$ 是黄金比例。当只有两台机器时，我们展示了基于两个确定性算法的随机化算法已经期望地 $\frac{3\varphi+3\sqrt{13-7\varphi}}{4}(\approx 2.1839)$-竞争，同时证明了任何确定性算法的竞争比下界为 2.2117。

关键词： 调度问题；多处理器；带测试调度；完工时间；随机算法

1 介绍

我们在本文中研究完全在线的多处理器调度与测试问题[1-4]，并从随机化算法的角度进行研究。多处理器调度[5]是最早的已知 NP 难组合优化问题之一，在过去几十年中得到了广泛研究。多处理器调度的一个实例 I 包含一组 n 个作业 $J = \{J_1, J_2, \ldots, J_n\}$，每个作业都要在一组 m 台并行的相同机器 $M = \{M_1, M_2, \ldots, M_m\}$ 上非抢占地执行，目标是最小化最大完工时间 $C_{\max}$，即最大作业完成时间。与经典设置不同的是，在调度与测试中，每个作业 J_j 都有一个处理时间上界 u_j，并且有一个长度为 t_j 的测试操作，但其处理时间 p_j 直到作业被测试后才会揭示。作业 J_j 可以在其中一台机器上执行 u_j 时间，或者选择先进行测试，测试时间为

t_j，然后立即执行 p_j 时间。若所有作业都在时间零到达，则多处理器调度与测试是一个半在线问题，表示为 $P|t_j, 0 \le p_j \le u_j|C_{\max}$。本文研究的是完全在线问题，其中作业按顺序到达，在作业到达时需要做出测试决策，并且指定测试和/或执行的机器，表示为 $P|\text{online}, t_j, 0 \le p_j \le u_j|C_{\max}$。显然，半在线是完全在线的特例，在这两种情况下，调度器都应该利用已知的作业信息，在作业到达时决定是否进行测试，以便最好地平衡由于未知处理时间所消耗的总时间。

给定一个多项式时间的确定性算法，针对半在线或完全在线问题，令 $C(I)$ 为该算法在实例 I 上产生的最大完工时间，而 $C^*(I)$ 为最优离线调度的最大完工时间。算法的性能通过竞争比来衡量，定义为 $\sup_I\{C(I)/C^*(I)\}$，其中 I 遍历所有问题实例，算法称为 $\sup_I\{C(I)/C^*(I)\}$-竞争的。切换到随机化算法时，我们相应地收集其在实例 I 上的期望最大完工时间 $E[C(I)]$，该随机化算法称为 $\sup_I\{E[C(I)]/C^*(I)\}$-竞争的。对于在线问题，随机化算法有时可以更好地处理不确定性，从而导致比最好的确定性算法更低的期望竞争比。我们为 $P|\text{online}, t_j, 0 \le p_j \le u_j|C_{\max}$ 提出了这样的一个随机化算法，并且进一步证明，当只有两台机器时，其期望竞争比严格小于任何确定性算法已证明的竞争比下界。在我们的问题中，作业处理是非抢占的。

在文献中，研究人员还考虑了抢占式作业处理[1-4]，其中任何测试或执行操作都可以被中断并稍后恢复，或者考虑了更为受限的测试抢占式变体[4]，在该变体中，已测试作业的测试和执行操作是非抢占的，但执行操作不必立即跟随测试操作，且不一定发生在同一台机器上。此外，我们的目标是最小化最大完工时间 $C_{\max}$，即最小最大目标；而另一个重要目标是最小化总作业完成时间，或者最小和目标，这也是受到了许多研究的关注[1-2,4]。

1.1 关于全在线问题的现有研究

现有的针对完全在线问题 $P|\text{online}, t_j, 0 \le p_j \le u_j|C_{\max}$ 的近似算法，无论是确定性算法还是随机化算法，都不多。我们首先区分一个特殊的情况，其中所有的测试操作时间均为单位时间，即对于每个作业 J_j，有 $t_j = 1$，我们称之为均匀测试

情况[1-2,4]，表示为 $P|\text{online}, t_j = 1, 0 \le p_j \le u_j|C_{\max}$。注意，在一般的测试情况下，测试时间可以是任何非负值。当只有一台机器时，机器上的作业处理顺序与最大完工时间无关。这表明完全在线问题和半在线问题是相同的。第一组结果是关于半在线均匀测试问题 $P_1|t_j = 1, 0 \le p_j \le u_j|C_{\max}$，由 D¨urr 等人[1-2]提出。他们提出，当 $u_j \ge \varphi = \frac{\sqrt{5}+1}{2}$ 时测试作业 J_j，或者以概率 $f(u_j) = \max\left(0, \frac{u_j(u_j-1)}{u_j(u_j-1)+1}\right)$ 测试它，从而得到了一个确定性的 φ-竞争算法和一个期望的 4/3 竞争算法。令 $r_j = \frac{u_j}{t_j}$；Albers 和 Eckl[3]将上述两种算法扩展到一般测试情况下 $P_1|t_j, 0 \le p_j \le u_j|C_{\max}$，即当 $r_j \ge \varphi$ 时测试作业 J_j，或者以概率 $f(r_j)$ 测试它，分别达到了相同的竞争比和期望竞争比。作者们[1-3]证明了这两种算法是最优的，即 φ 是任何确定性算法的竞争比下界，而 4/3 是任何随机化算法的期望竞争比下界，这一结果由 Yao 原理[6]得出。

当至少有两台机器时，完全在线问题比半在线问题更为一般。Albers 和 Eckl[4]提出了在列表调度规则[7]中测试作业 J_j，即当 $r_j \ge \varphi$ 时，按照该规则将每个作业分配到最空闲的机器上进行处理（可能是测试后执行）。他们证明了这样的算法是 $\varphi(2-\frac{1}{m})$-竞争的，且该分析是紧的，其中 m 是机器的数量。他们还证明了，即使在均匀测试情况下 $P|\text{online}, t_j = 1, 0 \le p_j \le u_j|C_{\max}$ 中，任何确定性算法的竞争比下界为 2[4]，并且对于两台机器的一般测试情况 $P_2|\text{online}, t_j, 0 \le p_j \le u_j|C_{\max}$，即当只有两台机器时，其下界为 2.0953，这一结果也稍微更好[4]。

1.2 我们的成果

对于完全在线和半在线的多处理器调度与测试问题，目标是最小化最大完工时间，我们已经看到，现有的随机化算法仅为单机器情况下的最优期望 $\frac{4}{3}$ 竞争算法[1-3]。它们的期望竞争比严格小于任何确定性算法竞争比下界 φ。对于经典的在线多处理器调度问题，只有当 $2 \le m \le 5$ 时，Bartal 等人[8] 和 Seiden[9] 提出的算法的期望竞争比严格小于任何确定性算法的竞争比下界；Albers[10] 提出的算法期望竞争比 1.916 也严格小于确定性下界 1.919，但条件是只使用了三种离线最大完工时间下界。

在本文中，我们始终假设完全在线问题 $P|\text{online},t_j,0 \le p_j \le u_j|C_{\max}$ 中至少有两台机器。我们提出了一个几乎随机化的算法，它是任意多个组件确定性算法的集成,每个算法以一定的概率运行,并证明其期望竞争比最多为 $\sqrt{\left(1-\frac{1}{m}\right)^2 \varphi^2 + 2\left(1-\frac{1}{m}\right)+1}$，其中 m 是机器的数量。这个期望竞争比随着 m 的增大而严格递增，严格小于 Albers 和 Eckl[4]提出的最佳已知确定性算法的竞争比 $\varphi(2-\frac{1}{m})$，并且当 m 趋于无穷大时，它逼近 $(\sqrt{\varphi+3}+1) \approx 3.1490$。据我们所知，文献中的几乎随机化算法大多数是其组件确定性算法的均匀分布，而我们的算法则是第一个非均匀分布的任意多个组件算法。

当只有两台机器时，即对于 $P_2|\text{online},t_j,0 \le p_j \le u_j|C_{\max}$，我们展示了在随机化算法中采用两个组件确定性算法导致的期望竞争比为 $\frac{3\varphi+3\sqrt{13-7\varphi}}{4} \approx$ 2.1839

对于这个两台机器的情况，Albers 和 Eckl[4] 证明了任何确定性算法的竞争比下界为 2.0953。我们将其改进为 2.2117，从而证明我们的随机化算法在期望竞争比方面优于任何确定性算法。

关于不可近似性，我们证明了至少三台机器时，任何随机化算法的期望竞争比下界为 1.6682，而在仅有两台机器时，对于 $P_2|\text{online},t_j,0 \le p_j \le u_j|C_{\max}$，期望竞争比的下界为 1.6522，这两个下界都是使用稍微修改过的 Yao 原理[6] 得出的。这两个下界在三台和两台机器情况下严格优于 Albers 和 Eckl[4] 提出的下界 $2-\frac{1}{m}$。我们的所有结果总结如图 3.1所示。

本文其余部分的组织结构如下: 第 2 节介绍一些基本的符号和定义; 接下来，在第 3 节中，我们证明了至少三台机器时完全在线问题 $P|t_j,0 \le p_j \le u_j|C_{\max}$ 的期望竞争比下界 1.6682 和仅两台机器时的下界 1.6522；第 4 节包含了该问题的随机化算法及其性能分析、两台机器情况下稍微修改的随机化算法及其性能分析，以及对于 $P_2|\text{online},t_j,0 \le p_j \le u_j|C_{\max}$ 的任何确定性算法的竞争比下界 2.2117; 最后，第 5 节对论文进行了总结。

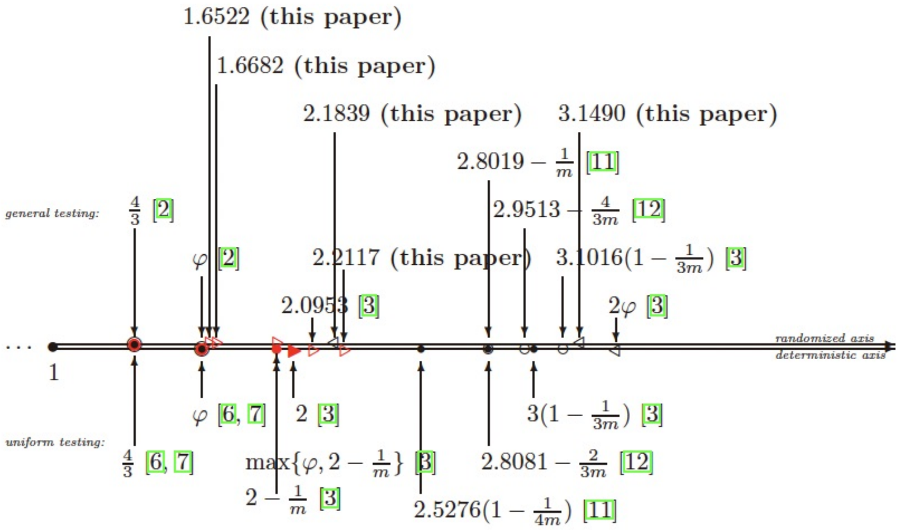


图 3.1 现有的确定性和随机化算法在完全在线和半在线多处理器调度与测试问题中的竞争比。在横轴上，红色符号表示下界；三角形（◁ 和 ▷ 表示一般测试，否则表示均匀测试）表示完全在线问题的结果，而圆形（○ 表示一般测试，· 表示均匀测试）表示半在线问题的结果。在两个横轴之间，底部的轴表示确定性算法，顶部的轴表示随机化算法；轴上方的结果是针对一般测试情况的，而轴下方的结果是针对均匀测试情况的。

.

2 预备内容

我们研究完全在线的多处理器调度与测试问题，以最小化最大完工时间，记作 $P|\text{online},t_j,0 \le p_j \le u_j|C_{\max}$，其中作业依次到达。我们的目标是设计期望竞争比优于现有最先进的确定性算法的随机化算法，或者更进一步，优于已证明的最佳确定性竞争比下界。所有作业都在时间零到达的特殊情况称为半在线问题，记作 $P|t_j,0 \le p_j \le u_j|C_{\max}$。在适用的情况下，我们将证明半在线变体的下界。

多处理器调度与测试的一个实例 I 包含一组作业集 $J = \{J_1,J_2,\dots,J_n\}$，每个作业都要在一组 m 台并行的相同机器 $M = \{M_1,M_2,\dots,M_m\}$ 上执行，其中 n 和 $m \ge 2$ 是输入的一部分。每个作业 J_j 都有一个已知的处理时间上界 u_j 和一个长度为 t_j 的测试操作，处理时间 p_j 在测试操作执行之前是未知的。也就是说，调度器可以选择不测试作业 J_j，而是执行它 u_j 时间，或者选择测试它 t_j 时间后，

在同一台机器上立即执行 p_j 时间。

确定性算法 A 对是否测试作业做出二元决策。令 p_{A_j} 和 ρ_j 分别表示作业 J_j 在算法 A 和最优离线调度中的总时间。可以看到，当作业被测试时，$p_{A_j}=t_j+p_j$，否则 $p_{A_j}=u_j$，而 $\rho_j=\min\{u_j,t_j+p_j\}$。令 C_j 表示作业 J_j 在算法 A 生成的调度中的完成时间。问题的目标是最小化最大完工时间 $C_{\max}=\max_j C_j$。我们使用 $C_A(I)$（当实例 I 在上下文中明确时，简称为 C_A）和 $C^*(I)$（或相应的 C^*）分别表示算法 A 和最优离线调度在实例 I 上的最大完工时间。

切换到更一般的随机化算法 A，其在实例 I 上的期望最大完工时间表示为 $E[C_A(I)]$。

定义 2.1. 竞争比：对于确定性算法 A，竞争比是算法 A 生成的调度的最大完工时间与最优离线调度的最大完工时间之间的最坏情况比值，即 $\sup_I\left\{\frac{C_A(I)}{C^*(I)}\right\}$ 其中 I 遍历所有问题实例。

对于更一般的随机化算法A，其期望竞争比为$\sup_I\left\{\frac{E[C_A(I)]}{C^*(I)}\right\}$

由于作业 J_j 的处理时间 p_j 可能达到 0 和 u_j 两个极端值，确定性算法通常会设置一个阈值函数 $r_j=\frac{u_j}{t_j}$ 用于二元测试决策。直观上，随机化算法应该以概率 $f(r_j)$ 测试作业 J_j，并且该概率函数应该随着比率 r_j 的增大而增大，且当 $r_j\le 1$ 时，$f(r_j)=0$。

实际上，D¨urr 等人[1-2] 和 Albers 与 Eckl[3] 在他们的最优期望 4/3 竞争随机化算法中使用了如下的概率函数：

$$f(r)=\begin{cases}0, & \text{如果} r\le 1,\\ \frac{r(r-1)}{r(r-1)+1}, & \text{如果} r>1.\end{cases} \tag{3-1}$$

他们分别应用于单机器问题 $P_1|\text{online},t_j=1,0\le p_j\le u_j|C_{\max}$ 和 $P_1|\text{online},t_j,0\le p_j\le u_j|C_{\max}$。遗憾的是，他们的成功不能轻易扩展到多机器情况，因为他们性能分析中使用的一个关键事实是，对于单台机器，期望最大完工时间等于所有作业期望处理时间的总和。我们在下面展示，对于多台机器，如果一个随机化算法

使用式 3-1 中的概率函数来测试作业，那么当机器数量趋近于无穷大时，它的期望竞争比将是无界的。

引理 2.2. 对于 $P|online,t_j=1,0\leq p_j\leq u_j|C_{max}$，如果一个随机化算法使用式 *3-1* 中的概率函数来测试作业，那么当机器数量趋近于无穷大时，它的期望竞争比将是无界的。

引理 2.2 的证明在某种程度上表明，当作业 J_j 的比率 r_j 足够大时，应该测试作业，而不是仅仅为不测试留下一点概率。事实上，稍后我们将看到，在我们提出的随机化算法中，对于绝对测试作业，有一个关于 r_j 的阈值。另一方面，在至少有两台机器的情况下，由于随机作业测试决策的存在，我们不再能确定每个作业分配给哪台机器。如果使用如式 3-1 中的作业测试概率函数，调度决策变得随机。因此，我们摒弃了[1-3] 中的随机化算法设计思想，而是扩展了 Albers[10] 的思想，设计了多个相互补充的组件确定性算法 $A_0,A_1,\ldots,A_\ell$，其中 ℓ 可以任意大，然后以一定的概率 α_i 运行每个 A_i。令 $p^j_{A_i}$ 表示作业 J_j 在算法 A_i 中消耗的总时间，对于任意 $i=0,1,\ldots,\ell$。随机化算法 A 中作业 J_j 的期望处理时间为

$$E[p^j_A]=\sum_{i=0}^{\ell}\alpha_i p^j_{A_i}. \tag{3-2}$$

类似地，令 C_{A_i} 为算法 A_i 生成的调度中的最大完工时间，对于任意 $i=0,1,\ldots,\ell$。期望最大完工时间为

$$E[C_A]=\sum_{i=0}^{\ell}\alpha_i C_{A_i}. \tag{3-3}$$

以下四个引理对于任何以概率分布 $(\alpha_0,\alpha_1,\ldots,\alpha_\ell)$ 运行的组件确定性算法 $A_0,A_1,\ldots,A_\ell$ 均成立，同时假设在不失一般性的情况下，它们都不会对任何作业 J_j 在 $r_j\leq 1$ 时进行测试。

引理 2.3. 对于任意 $i=0,1,\ldots,\ell$，如果作业 J_j 在 A_i 中被测试且 $r_j>1$，则

$$p^j_{A_i}\leq\left(1+\frac{1}{r_j}\right)\rho_j.$$

引理 2.4. 对于任意 $i=0,1,\ldots,\ell$，如果作业 J_j 在 A_i 中未被测试且 $r_j>1$，则

$$p^j_{A_i}\leq r_j\rho_j.$$

引理 2.5. 如果作业 J_j 在一组算法 T 中被测试，但在任何 $\{A_0, A_1, \ldots, A_\ell\} \setminus T$ 中未被测试，则

$$E[p_A^j] \leq \max\left(\theta + (1-\theta) r_j, 1 + \frac{\theta}{r_j}\right) \rho_j,$$

其中 $\theta = \sum_{A_i \in T} \alpha_i$。

引理 2.6. 以下两个最优离线最大完工时间 C^* 的下界成立：

$$C^* \geq \max\left\{\frac{1}{m}\sum_{j=1}^{n} \rho_j, \quad \max_{j=1}^{n} \rho_j\right\}. \tag{3-4}$$

在完全在线问题 $P|\text{online}, t_j, 0 \leq p_j \leq u_j|C_{\max}$ 中，作业依次到达，假设作业的顺序为 $hJ_1, J_2, \ldots, J_n$。在半在线问题 $P|t_j, 0 \leq p_j \leq u_j|C_{\max}$ 中，所有作业都在时间零到达，算法可以利用所有已知的 u_j 和 t_j 来按任意顺序处理它们。

3 期望竞争下界

定理 3.1. 对于任何随机算法 A_r 和任何随机实例 I_r，A_r 的期望竞争比满足

$$\sup_{I \in I} \frac{E[C_{A_r}(I)]}{C^*(I)} \geq \inf_{A \in A} \frac{E[C_A(I_r)]}{E[C^*(I_r)]},$$

其中 $E[C_A(I_r)]$ 和 $E[C^*(I_r)]$ 分别是确定性算法 A 在实例 I_r 上的期望目标值和最优期望目标值。

对于半在线单机均匀测试问题 $P1|t_j = 1, 0 \leq p_j \leq u_j|C_{\max}$，以下随机实例展示了期望竞争比的紧致下界 $\frac{4}{3}$[1][2][3]。

例子 3.2. 在这个 $P1|t_j = 1, 0 \leq p_j \leq u_j|C_{\max}$ 的随机实例中，只有一个任务 J_1，其参数为 $u_1 = 2$ 和 $t_1 = 1$，且 $p_1 = 0$ 或 2 的概率各为 *0.5*。

在任何确定性算法 A 中，J_1 要么被测试，要么未被测试。如果 J_1 未被测试，则 $p_1^A = u_1 = 2$；否则 J_1 被测试，因此 $p_1^A = t_1 + p_1$，这可能是 *1* 或 *3*，每种情况的概率各为 *0.5*。由此得出期望总执行时间为 $E[p_1^A] = 2$，无论 J_1 是否被测试。

另一方面，最优离线执行时间是 $\rho_1 = 1$ 如果 $p_1 = 0$，或者 $\rho_1 = 2$ 如果 $p_1 = 2$；即，期望最优执行时间为 $E[\rho_1] = 1.5$。根据定理 *3.1*，$\frac{E[p_1^A]}{E[\rho_1]} = \frac{4}{3}$ 是半在线单机均匀测试问题 $P1|t_j = 1, 0 \leq p_j \leq u_j|C_{\max}$ 的期望竞争比的下界。

当有多台机器时，以下确定性实例展示了期望竞争比的下界 $2-\frac{1}{m}$[4]。

例子 3.3. 在这个 $P|t_j=1,0\le p_j\le u_j|C_{\max}$ 的确定性实例中，有 m 台机器和 m 个任务 $J_1,J_2,\ldots,J_m$，每个任务的参数为 $u_j=2$ 和 $t_j=1$，且 $p_j=0$ 或 2 的概率各为 *0.5*。

在任何确定性算法 A 中，每个任务 J_j 要么被测试，要么未被测试。如果 J_j 未被测试，则 $p_j^A=u_j=2$；否则 J_j 被测试，因此 $p_j^A=t_j+p_j$，这可能是 *1* 或 *3*，每种情况的概率各为 *0.5*。由此得出每个任务的期望执行时间为 $E[p_j^A]=2$。

考虑最优离线调度。如果所有任务 J_j 都未被测试，则每个任务的实际处理时间为 *2*，总完成时间为 $2m$。如果所有任务都被测试，则每个任务的实际处理时间是 *1* 或 *3*，每种情况的概率各为 *0.5*，总完成时间为 $m\times1.5=1.5m$。最优离线调度可以通过将任务分配到不同的机器上来实现最小的完成时间。最优离线完成时间为 $\rho=\frac{2m}{m}=2$ 如果所有任务都未被测试，或者 $\rho=\frac{1.5m}{m}=1.5$ 如果所有任务都被测试。因此，期望最优完成时间为 $E[\rho]=1.5$。

根据定理 *3.1*，对于多台机器的情况，$\frac{E[C_A]}{E[\rho]}=\frac{2m}{1.5m}=\frac{4}{3}$ 是一个下界。然而，通过更精细的分析，可以得到更紧致的下界 $2-\frac{1}{m}$。

因此，对于多台机器的情况，期望竞争比的下界为 $2-\frac{1}{m}$。

例子 3.4. 在这个 $P|online,t_j,0\le p_j\le u_j|C_{\max}$ 的确定性实例 I 中，有 m 台机器和 $m+1$ 个任务按顺序到达 $\langle J_1,J_2,\ldots,J_{m+1}\rangle$，每个任务的参数为

$$u_j=2,\quad t_j=1,\quad 对于任何 j=1,2,\ldots,m+1.$$

也就是说，这 $m+1$ 个任务在到达时是不可区分的。对于一个确定性算法，对手将前两个未被测试的任务（如果存在）或序列中的最后两个任务的实际执行时间设置为 *0*；其他 $m-1$ 个任务的实际执行时间为 *2*。

例如，当 $m=4$ 且算法不测试任何任务时，实际执行时间序列为 $\langle0,0,2,2,2\rangle$；如果算法不测试 J_3,J_4 和 J_5 中的任何一个，则实际执行时间序列为 $\langle2,2,0,0,2\rangle$；如果算法仅不测试 J_1，则实际执行时间序列为 $\langle0,2,2,2,0\rangle$；如果算法测试所有五个任务，则实际执行时间序列为 $\langle2,2,2,0,0\rangle$。

给定一个确定性算法 A，记 $C^A_{\min}(I)$ 为算法 A 在实例 I 上产生的调度中的最小机器负载。

引理 3.5. 当 $m \geq 3$ 时，在实例 *3.4* 中的实例 I 上，对于任何确定性算法 A，要么 $C_A(I) \geq 4$，要么 $C_A(I) = 3$ 且 $C_{\min}(I) \geq 2$。

证明. 如果算法 A 不测试两个或更多任务，则每个任务的总处理时间至少为 2。根据鸽巢原理，这会导致完成时间 $C_A(I) \geq 4$。如果算法 A 测试了所有任务中的至多一个任务，则在前 m 个任务 $\langle J_1, J_2, \ldots, J_m \rangle$ 中，除了一个例外任务外，每个任务的总处理时间为 3。这个例外任务的总处理时间如果是被测试的则为 1，未被测试的则为 2。可以看到，当一台机器被分配了这些 m 个任务中的任意两个时，根据鸽巢原理，完成时间 $C_A(I) \geq 4$。在另一种情况下，每台机器恰好被分配了一个这些 m 个任务中的任务，将最后一个任务 J_{m+1} 分配给任何一台机器会导致要么 $C_A(I) \geq 4$，要么 $C_A(I) = 3$ 且 $C_{\min}(I) \geq 2$。 □

引理 3.6. 当 $m = 2$ 时，在实例 *3.4* 中的实例 I 上，对于任何确定性算法 A，要么 $C_A(I) \geq 5$，要么 $C_A(I) \geq 4$ 且 $C_{\min}(I) \geq 1$，要么 $C_A(I) \geq 3$ 且 $C_{\min}(I) \geq 2$。

证明. 设 I 是一个三任务实例。我们区分两种情况：算法 A 是否测试第一个任务 J_1。

1. J_1 被测试。在这种情况下，$p_1 = 2$，从而 $p^A_1 = 3$。注意到 J_2 和 J_3 的总处理时间至少为 1。这三个值 $\{3, 1, 1\}$ 保证了其中一个完成时间场景。

2. J_1 未被测试。在这种情况下，$p_1 = 0$，从而 $p^A_1 = 2$。注意到 J_2 和 J_3 的总处理时间至少为 2。这三个值 $\{2, 2, 2\}$ 也保证了其中一个完成时间场景。

因此，无论哪种情况，都可以保证完成时间 $C_A(I) \geq 4$ 或 $C_A(I) = 3$ 且 $C_{\min}(I) \geq 2$。

如果 J_1 未被测试，则 $p^A_1 = 2$。注意到 J_2 和 J_3 中的一个任务的总处理时间至少为 2（要么未被测试，要么被测试且实际执行时间为 2），另一个任务的总处理时间至少为 1。这三个值 $\{2, 2, 1\}$ 保证了其中一个完成时间场景。 □

定理 3.7. 对于存在三台或更多机器的情况 $P|online,t_j,0 \le p_j \le u_j|C_{\max}$，任何随机算法的期望竞争比至少为 $\frac{21}{2}-\sqrt{78} \approx 1.6682$。

定理 3.8. 对于问题 $P2|online,t_j,0 \le p_j \le u_j|C_{\max}$，任何随机算法的期望竞争比至少为 $\frac{21+4\sqrt{51}}{30} \approx 1.6522$。

4 随机算法

在本节中，我们首先提出了一种针对至少两台机器存在的完全在线多处理器调度与测试问题 $P|\text{online},t_j,0 \le p_j \le u_j|C_{\max}$ 的随机算法，并证明其期望竞争比为 $(\sqrt{\varphi+3}+1) \approx 3.1490$。当只有两台机器时，我们稍微调整了算法中的参数，从而得到了改进的期望竞争比 $\frac{3\varphi+3\sqrt{13-7\varphi}}{4} \approx 2.1839$。最后，我们证明了对于两台机器的情况，任何确定性算法的竞争比的改进下界为 2.2117，这意味着我们的随机算法在期望竞争比方面优于任何确定性算法。

对于完全在线多处理器调度与测试问题 $P|\text{online},t_j,0 \le p_j \le u_j|C_{\max}$，Albers 和 Eckl[3] 提出如果 $r_j \ge \varphi$ 则对任务 J_j 进行测试，在列表调度规则[7] 中将每个任务分配给负载最小的机器进行处理（可能先测试后执行）。他们证明了这种确定性算法是紧致的 $\varphi(2-\frac{1}{m})$-竞争比，其中 m 是机器的数量。我们将它作为我们的第一个组合算法 A_0，并以概率 $\alpha_0=\alpha(m,\ell)$ 运行该算法，如公式 3-5 所示，其中 $\ell \ge 1$ 是由我们的随机算法选择的一个固定常数。

$$\alpha_0=\alpha(m,l)=\sqrt{\frac{(1-\frac{1}{m})(l+1)\varphi^2}{(1-\frac{1}{m})(l+1)\varphi^2+2l}}, a_i=\beta(m,l)=\frac{1-\alpha(m,l)}{l}, i=1,2,\ldots,l. \tag{3-5}$$

以下 $\alpha(m,\ell)$ 和 $\beta(m,\ell)$ 分别简写为 α 和 β。

可以看到，当 $r_j \le 1$ 时，任务 J_j 不应该被测试，这在算法 A_0 中也是如此。另一方面，当 r_j 很大时，即使 p_j 非常接近 u_j，测试也不会浪费太多时间，这也与算法 A_0 的情况一致。“做出错误测试决策的风险”发生在 r_j 接近 φ 时，因为在这种情况下，如果 J_j 未被测试，则 p_j 可能非常接近 0；而如果被测试，则 p_j

可能非常接近 u_j。然而，从引理 2.2 的证明中我们也可以看到，即使对具有较大比率 r_j 的任务进行不测试的概率很小，也可能导致期望竞争比无界，特别是在机器数量 m 趋于无穷大时。

因此，在其他每个组合算法 A_i（$i=1,2,\dots,\ell$）中，我们设置一个阈值 $y_i(m,\ell)>\varphi$ 如公式 3-6 所示，使得比率 $r_j \geq y_i(m,\ell)$ 的任务 J_j 在 A_i 中被测试；然后我们设置另一个阈值 $x_i(m,\ell)=1+\frac{1}{y_i(m,\ell)}$，使得比率 $r_j \leq x_i(m,\ell)$ 的任务 J_j 在 A_i 中不被测试。

$$y_i(m,l)=\frac{\varphi(\alpha+i\beta)}{\alpha}, x_i(m,l)=1+\frac{1}{y_i(m,l)}, i=0,1,\dots,l. \tag{3-6}$$

类似地，下面我们将 $y_i(m,\ell)$ 和 $x_i(m,\ell)$ 简化为 y_i 和 x_i，分别表示。当 $r_j\in(x_i,y_i)$ 时，我们在 A_i 中反转测试决策，即如果 $r_j\leq\varphi$ 则测试 J_j，如果 $r_j>\varphi$ 则不测试 J_j。我们的随机算法，记作 GCL，以概率 α_i 选择运行 A_i，其中 $i=0,1,\dots,\ell$。

从公式 3-5 和 3-6 可以看出，$\alpha,\beta>0$，并且

$$1<x_\ell<\dots<x_1<x_0=\varphi=y_0<y_1<\dots<y_\ell=\frac{\varphi}{\alpha}. \tag{3-7}$$

算法 GCL 的简要描述如图 2 所示。

输入: m 台机器，以及按顺序到达的 n 个任务 $\langle J_1, J_2, \ldots, J_n \rangle$;

输出: 一个调度方案，其中每个任务被测试或不被测试，并指定处理该任务的机器。

步骤: 1. 选择 ℓ，并根据公式 3-5 和 3-6 设置参数 α_i, y_i 和 x_i，对于 $i = 0, 1, \ldots, \ell$;

2. 组成算法 A_i，$i = 0, 1, \ldots, \ell$: 对于每个任务 J_j，

(a) 如果 $r_j \leq x_i$ 或 $\varphi < r_j \leq y_i$，则将 J_j 未测试地分配到负载最小的机器上进行处理;

(b) 否则，将 J_j 分配到负载最小的机器上进行测试和处理。

3. 以概率 α_i 选择运行 A_i，对于 $i = 0, 1, \ldots, \ell$。

引理 4.1.

$$\frac{1-\alpha}{x_i} < \frac{\alpha}{\varphi}, (l-i)\beta(y_{i+1}-1) < \frac{\alpha}{\varphi}, \forall i = 0, 1, \ldots, \ell-1.$$

引理 4.2. 在算法 A_i 中，$p_j^{A_i} \leq y_i \rho_j, \forall J_j, i = 0, 1, \ldots, \ell$。

引理 4.3. 在算法 *GCL* 中，$E[p_j^A] \leq x_l \rho_j, \forall J_j$。

定理 4.4. 对于问题 $P|online, t_j, 0 \leq p_j \leq u_j|C_{\max}$，若设置 $\alpha(m,\ell)$ 和 $\beta(m,\ell)$ 如公式 *3-5* 所示，并且对于每一个 $i = 0, 1, \ldots, \ell$ 设置 $y_i(m,\ell)$ 和 $x_i(m,\ell)$ 如公式 *3-6* 所示，其中 $m \geq 2$ 是机器的数量，$\ell \geq 1$ 是一个固定的常数，则算法 *GCL* 的期望竞争比最多为

$$\sqrt{(1-\frac{1}{m})^2(1+\frac{1}{\ell})^2\varphi^2 + 2(1-\frac{1}{m})(1+\frac{1}{\ell})} + 1 - (1-\frac{1}{m})\frac{\varphi}{\ell}$$

通过选择一个足够大的 ℓ,算法 GCL 的期望竞争比接近 $\sqrt{(1-\frac{1}{m})^2\varphi^2 + 2(1-\frac{1}{m})} + 1$ 该值随着 m 的增加而增加, 并且当 m 趋于无穷大时, 接近 $\sqrt{\varphi+3}+1 \approx 3.1490$。相应地，可以验证 α, y_ℓ 和 x_ℓ 分别接近 $\frac{\sqrt{\varphi+1}}{\sqrt{\varphi+3}} \approx 0.7529$, $\sqrt{\varphi+3} \approx 2.1490$, $1 + \frac{1}{\sqrt{\varphi+3}} \approx 1.4653$.

在算法 GCL 中，我们可以将任务的测试概率 $f(r)$ 作为其比率 r 的函数来确定。例如，如果 $r \geq y_\ell$，则 $f(r)=1$；如果 $r \leq x_\ell$，则 $f(r)=0$。对于一个固定的常数 ℓ，可以看到这种概率函数 $f(r)$ 是阶梯状且递增的。当 ℓ 趋于无穷大时，这种概率函数的极限（仍然记作 $f(r)$）是有趣的。首先，注意到 $\alpha(m,\ell)$ 接近 $\alpha(m)=\sqrt{\frac{(1-\frac{1}{m})\varphi^2}{(1-\frac{1}{m})\varphi^2+2}}$，$y_\ell(m,\ell)$ 和 $x_\ell(m,\ell)$ 分别接近 $y(m)=\frac{\varphi}{\alpha(m)}$ 和 $x(m)=1+\frac{1}{y(m)}$。然后，根据公式 3-6，我们有

$$f(r)=\begin{cases} 0, & \text{当} r \leq x(m), \\ 1-\frac{\alpha(m)}{\varphi(r-1)}, & \text{当} x(m) < r \leq \varphi, \\ \frac{\alpha(m)r}{\varphi}, & \text{当} \varphi < r \leq y(m), \\ 1, & \text{当} r > y(m), \end{cases}$$

该函数是递增的，并且除了在 φ 处不连续外，在其他地方都是连续的。

当只有两台机器时，即 $m=2$，通过选择一个足够大的 ℓ，算法 GCL 对于问题 $P2|\text{online},t_j,0 \leq p_j \leq u_j|C_{\max}$ 已经是期望 $\frac{1}{2}\sqrt{\varphi+5}+1 \approx 2.2863$-竞争比。注意，在算法内部，选择算法 A_0 的概率接近 $\alpha(2)=\frac{\varphi}{\sqrt{\varphi+5}} \approx 0.6290$。

接下来我们展示，通过选择另一个运行 A_0 的概率，基于两个组成算法的修订后的 GCL 算法对于问题 $P2|\text{online},t_j,0 \leq p_j \leq u_j|C_{\max}$ 是期望 $\frac{3\varphi+3\sqrt{13-7\varphi}}{4} \approx 2.1839$-竞争比。进一步地，这个期望竞争比的界是紧的。

我们注意到，对于问题 $P|\text{online},t_j,0 \leq p_j \leq u_j|C_{\max}$，我们的 GCL 算法的期望竞争比优于已知的最佳确定性竞争比 $\varphi(2-\frac{1}{m})$，但远高于确定性竞争比的下界 2[4]。它更远离定理 3.8 中的 1.6522、定理 3.7 中的 1.6682 以及在两台、三台和 $m \geq 4$ 台机器存在的情况下期望竞争比的下界 $2-\frac{1}{m}$[4]。

回顾一下，除了对均匀测试情况 $P|\text{online},t_j=1,0 \leq p_j \leq u_j|C_{\max}$ 的确定性竞争比下界 2 之外，Albers 和 Eckl[4] 还展示了两台机器一般测试情况 $P2|\text{online},t_j,0 \leq p_j \leq u_j|C_{\max}$ 的稍好一些的下界 2.0953。我们在下一个定理中将该下界从 2.0953 改进到 2.2117。因此，对于 $P2|\text{online},t_j,0 \leq p_j \leq u_j|C_{\max}$，GCL 的期望竞争比不仅优于半在线变体 $P2|t_j,0 \leq p_j \leq u_j|C_{\max}$ 的已知最佳确定性竞争比 2.3019，而且优于完全在线问题的确定性竞争比下界 2.2117。

定理 4.5. 对于问题 $P2|online,t_j,0 \leq p_j \leq u_j|C_{\max}$，任何确定性算法的竞争比大于 *2.2117*。

证明. 使用对抗性论证来构造一个三任务实例，任务顺序为 $\langle J_1,J_2,J3\rangle$。考虑一个确定性算法 A。

第一个任务 J_1 的参数为 $u_1=\varphi$ 和 $t_1=1$。如果 A 测试 J_1，则对手设置 $p_1=\varphi$。因此 $p_1^A=\varphi+1$ 且 $\rho_1=\varphi$。如果 J_1 未测试，则对手设置 $p_1=0$，从而 $p_1^A=\varphi$ 且 $\rho_1=1$。无论哪种情况，都有 $\frac{p_1^A}{\rho_1}=\varphi$，因此我们假设 $p_1^A=\varphi$ 且 $\rho_1=1$。（如果 $p_1^A=\varphi+1$ 且 $\rho_1=\varphi$，则下面与 J_2 和 J_3 相关的 u_j 和 t_j 值将乘以一个因子 φ 。）

对于任何即将到来的任务 J_j（即 $j=2,3$，对手总是设置 $p_j=u_j$ 如果 J_j 被算法测试，否则设置 $p_j=0$。

设 $x_0=\frac{3\varphi+1-\sqrt{11\varphi+6}}{2}\approx 0.4878$，这是二次方程 $x^2-(3\varphi+1)x+\varphi^2=0$ 的一个根。第二个任务 J_2 的参数为 $u_2=\varphi-x_0$ 和 $t_2=x_0$。我们区分两种情况：J_2 是否被算法测试。

情况 1: J_2 被算法测试。在这种情况下，$p_2=u_2$ 且 $p_2^A=\varphi$ 且 $\rho_2=\varphi-x_0$。如果 J_2 被安排在与 J_1 同一台机器上，则第三个任务 J_3 被取消（通过设置 $u_j=t_j=0$），导致 $C_A=p_1^A+p_2^A=2\varphi$。注意到 $\rho_2>1$ 且 $C^*=\rho_2=\varphi-x_0$。因此竞争比为 $\frac{2\varphi}{\varphi-x_0}$。

下面假设 J_j 被分配到机器 M_j 上，对于 $j=1,2$，每台机器的负载为 φ。第三个任务 J_3 的参数为 $u_3=y_0$ 和 $t_3=\varphi+1-x_0$，其中 $y_0=\frac{1-x_0+\sqrt{(3\varphi-5)x_0+15\varphi+8}}{2}\approx 3.0933$ 是二次方程 $y^2-(1-x_0)y-(\varphi+1-x_0)(2\varphi+1-x_0)=0$ 的一个根。

如果 J_3 未被测试，则 $p_3=0$，从而 $p_3^A=y_0$ 且 $\rho_3=\varphi+1-x_0$，导致 $C_A=\varphi+y_0$。在最优离线调度中，J_1 和 J_2 被安排在一台机器上，而 J_3 被安排在另一台机器上，导致最优离线完成时间为 $C^*=\varphi+1-x_0$。因此，竞争比为 $\frac{C_A}{C^*}=\frac{y_0+\varphi}{\varphi+1-x_0}$。

如果 J_3 被测试，则 $p_3=u_3$，从而 $p_3^A=t_3+u_3=\varphi+1-x_0+y_0$ 且 $\rho_3=y_0$，导致 $C_A=2\varphi+1-x_0+y_0$。在最优离线调度中，J_1 和 J_2 被安排在一台机器上，而 J_3 被安排在另一台机器上，导致 $C^*=y_0$，因此，竞争比为 $\frac{C_A}{C^*}=\frac{2\varphi+1-x_0+y_0}{y_0}$。

综上所述，在这种情况下，我们有

$$\frac{C_A}{C^*}\geq \min\left\{\frac{2\varphi}{\varphi-x_0},\frac{y_0+\varphi}{\varphi+1-x_0},\frac{2\varphi+1-x_0+y_0}{y_0}\right\}.$$

由于 y_0 是二次方程 $y^2-(1-x_0)y-(\varphi+1-x_0)(2\varphi+1-x_0)=0$ 的一个根，我们有

$$y_0(y_0+\varphi)=(\varphi+1-x_0)(y_0+2\varphi+1-x_0),$$

即上述最后两个量相等且大约为 2.21172。第一个量约为 2.8634。因此，$\frac{C_A}{C^*}>2.2117$。

情况 2: J_2 未被算法测试。在这种情况下，$p_2=0$，从而 $p_2^A=\varphi-x_0$ 且 $\rho_2=x_0$。如果 J_2 被安排在与 J_1 同一台机器上，则第三个任务 J_3 被取消（通过设置 $u_j=t_j=0$），导致 $C_A=p_1^A+p_2^A=2\varphi-x_0$。注意到 $\rho_2=x_0<1$ 且 $C^*=1$。因此，竞争比为 $\frac{C_A}{C^*}=2\varphi-x_0$。

下面假设 J_j 被分配到机器 M_j 上，对于 $j=1,2$，两台机器的负载分别为 φ 和 $\varphi-x_0$。第三个任务 J_3 的参数为 $u_3=\frac{(1+\varphi)y_0}{2\varphi+1-x_0}\approx 2.1606$ 和 $t_3=1+x_0$，其中 y_0 与情况 1 相同。

如果 J_3 未被测试，则 $p_3=0$，从而 $p_3^A=u_3=\frac{(1+\varphi)y_0}{2\varphi+1-x_0}$ 且 $\rho_3=1+x_0$，无论 J_3 被分配到哪台机器上，都有 $C_A\geq\varphi-x_0+\frac{(1+\varphi)y_0}{2\varphi+1-x_0}$。在最优离线调度中，$J_1$ 和 J_2 被安排在一台机器上，而 J_3 被安排在另一台机器上，导致最优离线完成时间为 $C^*=1+x_0$。因此，竞争比为

$$\frac{C_A}{C^*}\geq\frac{\varphi-x_0+\frac{(1+\varphi)y_0}{2\varphi+1-x_0}}{1+x_0}=\frac{(\varphi-x_0)(2\varphi+1-x_0)+(1+\varphi)y_0}{(1+x_0)(2\varphi+1-x_0)}.$$

由于 x_0 是二次方程 $x^2-(3\varphi+1)x+\varphi^2=0$ 的一个根，我们有

$$(\varphi-x_0)(2\varphi+1-x_0)=x_0^2-(3\varphi+1)x_0+\varphi(1+2\varphi)=\varphi(1+\varphi)$$

和

$$(1+x_0)(2\varphi+1-x_0)=-x_0^2+2\varphi x_0+2\varphi+1=(\varphi+1)(1+\varphi-x_0).$$

因此，$\frac{C_A}{C^*}\geq\frac{y_0+\varphi}{1+\varphi-x_0}$。

如果 J_3 被测试，则 $p_3=u_3$，从而 $p_3^A=t_3+u_3=1+x_0+\frac{(1+\varphi)y_0}{2\varphi+1-x_0}$ 且 $\rho_3=\frac{(1+\varphi)y_0}{2\varphi+1-x_0}$，无论 J_3 被分配到哪台机器上，都有 $C_A\geq 1+\varphi+\frac{(1+\varphi)y_0}{2\varphi+1-x_0}$。由于 $\rho_3>\rho_1+\rho_2$，在最优离线调度中，J_1 和 J_2 被安排在一台机器上，而 J_3 被安排在另一台机器上，导

致 $C^* = \rho_3 = \frac{(1+\varphi)y_0}{2\varphi+1-x_0}$，进而竞争比为

$$\frac{C_A}{C^*} \geq \frac{1+\varphi+\frac{(1+\varphi)y_0}{2\varphi+1-x_0}}{\frac{(1+\varphi)y_0}{2\varphi+1-x_0}} = \frac{y_0+2\varphi+1-x_0}{y_0}.$$

综上所述，在这种情况下，我们有

$$\frac{C_A}{C^*} \geq \min\left\{2\varphi - x_0, \frac{y_0+\varphi}{1+\varphi-x_0}, \frac{2\varphi+1-x_0+y_0}{y_0}\right\}.$$

与情况 1 相同，上述最后两个量相等且大约为 2.21172。第一个量约为 2.7482。因此，$\frac{C_A}{C^*} > 2.2117$。

以上两种情况证明了对于问题 $P2|\text{online}, t_j, 0 \leq p_j \leq u_j|C_{\max}$，任何确定性算法的竞争比大于 2.2117。

□

5 结论

我们研究了完全在线多处理器调度与测试问题 $P|\text{online}, t_j, 0 \leq p_j \leq u_j|C_{\max}$，并提出了一种随机算法 GCL，该算法是非均匀分布的任意多个确定性算法。据我们所知，文献中的随机算法作为元算法大多是其组成确定性算法的均匀分布，而我们的 GCL 偏向于其中一个组成算法；此外，当有多个组成算法时，以前的随机算法通常需要记录所有解，而我们的 GCL 不需要这样做，因为组成算法彼此独立。我们证明了 GCL 的期望竞争比约为 3.1490。当只有两台机器时，即对于 $P2|\text{online}, t_j, 0 \leq p_j \leq u_j|C_{\max}$，使用修订后的 GCL 算法中的两个组成算法可以得到期望竞争比为 2.1839。

我们还证明了三个不可近似结果，包括在至少三台机器存在的情况下，任何随机算法的期望竞争比的下界为 1.6682，对于 $P2|\text{online}, t_j, 0 \leq p_j \leq u_j|C_{\max}$ 的任何随机算法的期望竞争比的下界为 1.6522，以及对于 $P2|\text{online}, t_j, 0 \leq p_j \leq u_j|C_{\max}$ 的任何确定性算法的竞争比的下界为 2.2117。根据最后一个下界，我们得出结论，算法 GCL 在期望竞争比方面优于任何确定性算法。这种算法结果在文献中很少见。

算法 GCL 的期望竞争比远高于两台、三台和 $m \geq 4$ 台机器情况下的下界 1.6522、1.6682 和 $2-\frac{1}{m}$，这表明未来的研究可以缩小这些差距，即使是在两台机器的情况下也是如此。可以看到，算法 GCL 中的任务比率测试概率函数在 φ 处不连续，这暗示可能存在一个更好的概率分布函数来选择组成算法。

6 外文翻译参考文献

[1] DÜRR C, ERLEBACH T, MEGOW N, et al. Scheduling with explorable uncertainty[C]//9th Innovations in Theoretical Computer Science Conference (ITCS 2018). 2018.

[2] DÜRR C, ERLEBACH T, MEGOW N, et al. An adversarial model for scheduling with testing[J]. Algorithmica, 2020, 82(12): 3630-3675.

[3] ALBERS S, ECKL A. Explorable uncertainty in scheduling with non-uniform testing times[C]//Approximation and Online Algorithms: 18th International Workshop, WAOA 2020, Virtual Event, September 9–10, 2020, Revised Selected Papers 18. 2021: 127-142.

[4] ALBERS S, ECKL A. Scheduling with testing on multiple identical parallel machines[C]//Algorithms and Data Structures: 17th International Symposium, WADS 2021, Virtual Event, August 9–11, 2021, Proceedings 17. 2021: 29-42.

[5] GAREY MICHAEL R, JOHNSON DAVID S. Computers and Intractability: A guide to the theory of NP-completeness[Z]. 1979.

[6] YAO A C C. Probabilistic computations: Toward a unified measure of complexity[C]//18th Annual Symposium on Foundations of Computer Science (sfcs 1977). 1977: 222-227.

[7] GRAHAM R L. Bounds for certain multiprocessing anomalies[J]. Bell system technical journal, 1966, 45(9): 1563-1581.

[8] BARTAL Y, FIAT A, KARLOFF H, et al. New algorithms for an ancient scheduling problem[C]//Proceedings of the twenty-fourth annual ACM symposium on Theory of computing. 1992: 51-58.

[9] SEIDEN S S. Online randomized multiprocessor scheduling[J]. Algorithmica, 2000, 28(2): 173-216.

[10] ALBERS S. On randomized online scheduling[C]//Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. 2002: 134-143.

四、外文原文

Published as a conference paper at ICLR 2025

SPREAD PREFERENCE ANNOTATION: DIRECT PREFERENCE JUDGMENT FOR EFFICIENT LLM ALIGNMENT

Dongyoung Kim[1], **Kimin Lee**[1], **Jinwoo Shin**[1], **Jaehyung Kim**[2]
[1]Korea Advanced Institute of Science and Technology , [2]Yonsei University
kingdy2002@kaist.ac.kr, jaehyungk@yonsei.ac.kr

ABSTRACT

Aligning large language models (LLMs) with human preferences becomes a key component to obtaining state-of-the-art performance, but it yields a huge cost to construct a large human-annotated preference dataset. To tackle this problem, we propose a new framework, **S**pread **P**reference **A**nnotation with direct preference judgment (SPA), that boosts the alignment of LLMs using only a very small amount of human-annotated preference data. Our key idea is leveraging the human prior knowledge within the small (seed) data and progressively improving the alignment of LLM, by iteratively generating the responses and learning from them with the self-annotated preference data. To be specific, we propose to derive the preference label from the logits of LLM to explicitly extract the model's inherent preference. Compared to the previous approaches using external reward models or implicit in-context learning, we observe that the proposed approach is significantly more effective. In addition, we introduce a noise-aware preference learning algorithm to mitigate the risk of low quality within generated preference data. Our experimental results demonstrate that the proposed framework significantly boosts the alignment of LLMs. For example, we achieve superior alignment performance on AlpacaEval 2.0 with only 3.3% of the ground-truth preference labels in the Ultrafeedback data compared to the cases using the entire data or state-of-the-art baselines.[1]

1 INTRODUCTION

Recently, large language models (LLMs) have made huge progress in various NLP tasks, leading to real-world applications that are used by millions of users, such as coding assistants and chatbots (Anthropic, 2024; OpenAI, 2022; Team et al., 2023). Aligning LLMs with human feedback, particularly through learning from human preferences, is widely considered a crucial technique for their success (Christiano et al., 2017; Lee et al., 2021; Ziegler et al., 2019). To enhance this alignment, various preference learning algorithms have been extensively explored (Ouyang et al., 2022; Rafailov et al., 2023). Despite these advancements, one of the remaining challenges is the reliance on large-scale human-annotated preference data. As the quality and quantity of the preference data are critical for the successful alignment of LLMs (Bai et al., 2022a; Cui et al., 2023), the huge cost to acquire such data inevitably presents significant obstacles.

To mitigate this challenge, engaging LLMs in constructing preference data and improving their alignment using these data has recently gained attention. For example, a representative way on this line is generating multiple responses for the input prompts, and then approximating human preference between them through LLM's predictions, often referred to as *LLM-as-judge* (Bai et al., 2022b; Yuan et al., 2024). However, these approaches are only effective when the given LLM is sufficiently large and well-aligned to mimic human preference via in-context learning. On the other hand, using an external reward model is considerable to substitute human preference annotation efficiently (Jiang et al., 2023b; Snorkel, 2024), but it is built on the availability of large human preference data and could also be ineffective if there is a distribution mismatch. Lastly, these approaches have a risk of potential labeling noise from LLMs, but this aspect has not been explored yet. Therefore, in this work, we aim to develop a method to effectively improve the alignment of LLM by overcoming these limitations but only relying on small human annotation.

[1]https://github.com/kingdy2002/SPA

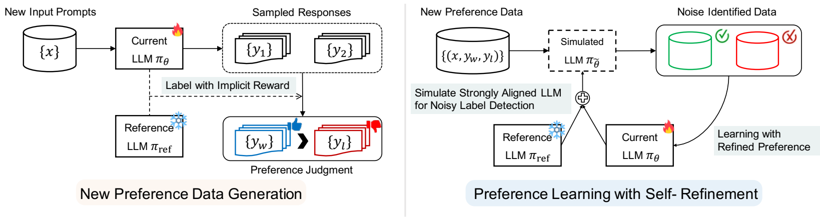


Figure 1: **Illustration of the proposed SPA framework.** SPA progressively improves the alignment of LLMs by iterating (1) the generation of new preference data and (2) the preference learning on the constructed data with self-refinement. Technical details are presented in Section 4.

Contribution. We introduce a simple yet effective framework, coined SPA, to improve the alignment of LLMs with only a small amount of human-labeled preference data, by **S**preading **P**reference **A**nnotation via direct preference judgment. Our key idea is to progressively expand the knowledge of human preference within the small (seed) data, by iteratively generating the responses and learning from them through the self-annotated preference labels. Specifically, our technical contributions are three-fold as described in what follows. First, we judge the preference labels directly using the logits of LLM to explicitly extract the model's inherent preference. This approach is more effective than previous methods that rely on external reward models or implicit in-context learning. Second, we introduce a confidence-based refinement of preference labels to reduce the risk of noise in preference learning with generated data. Third, to further enhance the effectiveness of this refinement, we propose using a linearly extrapolated prediction between current and reference models; it approximates predictions of a more strongly aligned model, leading to better noise identification.

We demonstrate the effectiveness of the proposed SPA by aligning recent LLMs with small human-annotated preference data and evaluating their alignment on the commonly used benchmarks. For example, using only 3.3% of ground-truth preference in Ultrafeedback data (Cui et al., 2023) with the mistral-7b-0.1v SFT model (Jiang et al., 2023a), our framework achieves over 16.4% increase in AlpacaEval2.0 (Li et al., 2023a) win rate compared to the initial SFT model (see Figure 2). Additionally, the AlpacaEval 2.0 length-controlled win rate is improved from 7.58% to 15.39%, and MT-bench score (Zheng et al., 2023) increased from 6.38 to 6.94. Compared to preference judgment methods like LLM-as-judge (Zheng et al., 2023), and even strong reward models such as PairRM (Jiang et al., 2023b), which have recently shown state-of-art performance in AlpacaEval2.0 benchmark, our approach consistently outperforms them across all metrics. More interestingly, the proposed SPA successfully improves the alignment of various LLMs, even without the initial human preference data. These results demonstrate that our framework is highly competitive and practical for real-world applications.

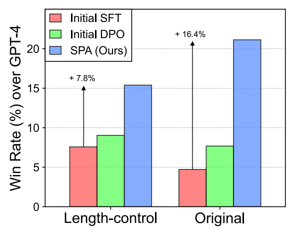


Figure 2: **Summary of main result.** Evaluation results on AlpacaEval 2.0 (Li et al., 2023a). Our framework significantly improves the alignment of LLMs, without additional human preference data. See detailed results in Section 5.

2 Related Work

Alignment of LLMs with human preference. Learning from human preferences now serves as a core component for the state-of-the-art LLMs (Anthropic, 2024; OpenAI, 2023; Team et al., 2023; Touvron et al., 2023) for aligning their responses with users' intent and values (Ouyang et al., 2022; Ziegler et al., 2019). Arguably, one of the most popular frameworks is reinforcement learning with human preference (RLHF) (Christiano et al., 2017; Lee et al., 2021), which first trains the reward model, and then fine-tunes LLM to maximize that reward with KL divergence regularization to prevent the reward over-optimization of LLM. On the other hand, various preference learning algorithms have recently been proposed to fine-tune LLMs with human preference more efficiently (Ethayarajh et al., 2024; Hong et al., 2024; Liu et al., 2023; Rafailov et al., 2023; Xu et al., 2023; Zhao et al., 2023; Meng et al., 2024). For example, Rafailov et al. (2023) proposes Direct Preference Optimization (DPO) which allows one to fine-tune LLMs without a separate reward modeling stage, by deriving the training objective mathematically equivalent to RLHF. Ethayarajh et al. (2024) further removes the reliance on pair-wise preference labels by formulating the objective based on a human utility model. However, these methods assume that large human-annotated preference data is available, which requires a huge data acquisition cost.

Engagement of LLMs for constructing preference data. For an efficient and scalable alignment procedure, engaging LLMs for preference dataset construction has recently received attention. One common approach involves generating multiple responses to input prompts from LLM, and using an LLM's predictions to approximate human preferences between them, a technique often referred to as *LLM-as-judge* (Bai et al., 2022a; Yuan et al., 2024). However, this method is effective only when the LLM is sufficiently large and well-aligned to mimic human preferences through in-context learning. Alternatively, employing an external reward model can efficiently replace human preference judgment (Jiang et al., 2023b; Snorkel, 2024), but this approach relies on the availability of extensive human preference data to pre-train reward model and may be ineffective if there is a distribution mismatch. Some concurrent works (Rosset et al., 2024; Snorkel, 2024; Wu et al., 2024; Xiong et al., 2024) have proposed the alignment procedure with iterative data expansion and preference learning. However, they use the external reward model or stronger LLM for the preference judgment. In contrast, we only utilize the intrinsic knowledge of training LLM for new data expansion and preference learning.

3 Preliminaries

Let us denote LLM as π_θ, which generates an output sequence (*e.g.*, response) y for a given input sequence (*e.g.*, prompt) x, *i.e.*, $y \sim \pi_\theta(\cdot|x)$. Then, our goal is to make π_θ provide human-aligned responses to various input prompts. To this end, we consider the popular framework of preference learning, which optimizes π_θ to learn human preferences between two different responses (Christiano et al., 2017; Lee et al., 2021; Ouyang et al., 2022). Specifically, we assume that the preference dataset $\mathcal{D} = \{(x, y_l, y_w)\}$ is available which consists of the triplets of input prompt x, preferred response y_w, and dispreferred response y_l. Here, the preference labels were annotated by a ground truth annotator, that is usually a human expert.

Reward modeling and RL fine-tuning. Since a pairwise preference between y_w and y_l is hard to model directly, one of the common practices is introducing reward function $r(x, y)$ and modeling the preference based on this using the Bradley-Terry model (Bradley & Terry, 1952):

$$p(y_w \succ y_l \mid x) = \frac{\exp\left(r(x, y_w)\right)}{\exp\left(r(x, y_w)\right) + \exp\left(r(x, y_l)\right)}. \tag{1}$$

From this formulation, one can introduce a parametrized reward model $r_\phi(x, y)$ by estimating its parameters with the maximum-likelihood objective:

$$\mathcal{L}_R(r_\phi) = -\mathbb{E}_{(x, y_w, y_l)\sim\mathcal{D}}\left[\log \sigma\left(r_\phi(x, y_w) - r_\phi(x, y_l)\right)\right]. \tag{2}$$

where σ is a sigmoid function. After this reward modeling procedure, one could improve the alignment of LLM π_θ by optimizing it to maximize the reward captured by r_ϕ. Here, KL-distance from the reference model π_{ref} is usually incorporated as a regularization to prevent the reward over-optimization of π_θ, with a hyper-parameter $\beta > 0$ (Ouyang et al., 2022; Ziegler et al., 2019):[2]

$$\mathcal{L}_{\text{RLHF}}(\pi_\theta) = -\mathbb{E}_{y\sim\pi_\theta, x\sim\rho}\left[r_\phi(x, y)\right] + \beta \mathrm{D}_{\mathrm{KL}}\left(\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x)\right). \tag{3}$$

Direct preference modeling and optimization. Rafailov et al. (2023) propose an alternative approach to align LLM π_θ with the preference dataset $\mathcal{D}$, which is called Direct Preference Optimization (DPO). DPO integrates a two-step alignment procedure with reward modeling and RL fine-tuning into a single unified fine-tuning procedure. Specifically, the optimal reward function is derived from the

[2] π_{ref} is usually initialized with supervised fine-tuned (SFT) LLM (Chung et al., 2024; Wei et al., 2022a). Also, π_θ is initialized with π_{ref}.

RLHF objective (Eq. 3), with the target LLM π_θ and the reference model π_{ref} (Go et al., 2023; Peng et al., 2019; Peters & Schaal, 2007).

$$r(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x), \text{ where } Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right). \quad (4)$$

Then, the preference between two responses could be measured using this reward derivation, and π_θ is optimized to maximize this preference of y_w over y_l using the preference dataset $\mathcal{D}$.

$$p_\theta(y_w \succ y_l | x) = \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right). \quad (5)$$

$$\mathcal{L}_{\text{DPO}}(\pi_\theta) = \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}}\left[-\log p_\theta(y_w \succ y_l | x)\right]. \quad (6)$$

4 SPA: Spread Preference Annotation to boost Alignment of LLMs

Overview. In this section, we present SPA: **S**pread **P**reference **A**nnoation via direct preference judgment to align LLMs while mitigating the huge cost for preference dataset construction. Our main idea is to fully exploit the human prior knowledge within the small (seed) data, and progressively update LLM to improve the alignment. To be specific, SPA iterates two steps: (1) data expansion with self-generated preference (Section 4.1) and (2) fine-tuning LLM with self-refined preference learning (Section 4.2). See Figure 1 for the overview.

Initial stage. We assume that a small (seed) preference dataset D_0 and an initial LLM π_{init} are given. Here, following the common practice (Ouyang et al., 2022; Rafailov et al., 2023; Ziegler et al., 2019), we use π_{init} which has been supervised fine-tuned (SFT) LLM on the instruction dataset (Chung et al., 2024; Wei et al., 2022a), but not aligned with human preference yet. Then, we first obtain weakly aligned LLM π_0 by fine-tuning π_{init} on D_0 using DPO (Rafailov et al., 2023) (Eq. 6). We adopt DPO among various preference learning methods due to its simplicity and effectiveness.

4.1 Direct preference judgment to align LLMs with self-generated data

For the i-th iteration ($i = 1, \dots$), we assume that the new prompt set $X_i = \{x\}$ is available, *i.e.*, $X_i \cap X_j = \emptyset$ for all $j = 0, \dots, i-1$.[3] From X_i, we construct i-th artificial preference dataset $\mathcal{D}_i = \{(x, y_l, y_w) | x \in X_i\}$, by using LLM's intrinsic generation and reward modeling capabilities. Specifically, for each input prompt $x \in X_i$, we sample two responses y_1 and y_2 from π_{i-1}, *i.e.*, $y_1, y_2 \sim \pi_{i-1}(x)$ where π_{i-1} is the resulting model from the previous iteration. Then, using the reward captured with π_{i-1} and π_{init} (Eq. 4), we measure the preference of π_{i-1} between y_1 and y_2:

$$p_{i-1}(y_1 \succ y_2 | x) = \sigma\left(\beta \log \frac{\pi_{i-1}(y_1|x)}{\pi_{\text{init}}(y_1|x)} - \beta \log \frac{\pi_{i-1}(y_2|x)}{\pi_{\text{init}}(y_2|x)}\right). \quad (7)$$

Then, we directly judge the preference label as below and construct $\mathcal{D}_i$ through this:

$$(y_w, y_l) = (y_1, y_2) \text{ if } p_{i-1}(y_1 \succ y_2 | x) > 0.5 \text{ else } (y_w, y_l) = (y_2, y_1). \quad (8)$$

4.2 Self-refinement of generated preference data for effective learning

After the construction of $\mathcal{D}_i$, we conduct i-th preference learning by fine-tuning π_θ, which is initialized by π_{i-1}, using DPO (here, we also use π_{i-1} as π_{ref} in Eq. 6). Learning the self-generated preference data $\mathcal{D}_i$ could improve the alignment by effectively spreading the human preference prior from $\mathcal{D}_0$ using the power of LLM. However, it also has a risk of the potential labeling noise which could occur from the distribution shift with X_i or insufficient reward modeling with π_{i-1}. Therefore, we further propose an improved preference learning method by introducing a novel denoising technique: *self-refinement* of preference labels with *de-coupled noise detection*.

[3] $X_0 = \{x | (x, y_l, y_w) \in \mathcal{D}_0\}$

Algorithm 1 SPA algorithm

Input: initial LLM π_{init}, seed preference dataset $\mathcal{D}_0$, number of improving iterations T, new prompt sets $\{X_i\}_{i=1}^{T}$,

Obtaining an initial weakly aligned model π_0 using DPO with π_{init} and $\mathcal{D}_0$ (Eq. 6)
for $t = 1$ **to** T **do**
  Synthesizing preference data $\mathcal{D}_t$ with π_{t-1} and X_t (Eq. 7 and 8)
  Initialization of training and reference models $\pi_\theta \leftarrow \pi_{t-1}, \pi_{\text{ref}} \leftarrow \pi_{t-1}$
  for mini-batch $B \sim \mathcal{D}_t$ **do**
    $z_{\widetilde{\theta}} \leftarrow$ De-coupled noise detection for B from $\pi_\theta, \pi_{\text{ref}}, X_t$ (Eq. 11 and 12)
    Calculate training loss $\mathcal{L}_{\text{rf}}$ with refined preference labels using $z_{\widetilde{\theta}}$ and π_θ (Eq. 10)
    Update model parameter: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{rf}}$
  end for
  Initializing next iteration model π_t with the updated parameters θ
end for
return π_T

Self-refinement of preference label: Our key intuition is that one can view the derived preference (Eq. 5) can be viewed as the confidence of the currently training LLM π_θ for the labels assigned by π_{i-1}. Then, π_θ would exhibit lower confidence if the given pair of responses is uncertain to answer, indicating a higher probability of labeling noise. Notably, we also remark that confidence is one of the most popular metrics in the noisy label learning literature (Han et al., 2018; Reed et al., 2014; Sohn et al., 2020). Under this intuition, we first identify the $K\%$ least confident samples:

$$z_\theta = 1 \text{ if } p_\theta(y_w \succ y_l|x) < \tau \text{ else } z_\theta = 0, \tag{9}$$

where τ is the confidence of K percentile sample of $\mathcal{D}_i$. Then, with this (potentially) noise identification label z_θ, we refine the assigned preference label using label smoothing (Müller et al., 2019), to train π_θ less confidently when the risk of label noise is high (*i.e.*, $z_\theta = 1$):

$$\mathcal{L}_{\text{rf}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_i}\left[-\big((1-\alpha * z_\theta)\log p_\theta(y_w \succ y_l|x) + \alpha * z_\theta \log p_\theta(y_l \succ y_w|x)\big)\right], \tag{10}$$

where α is a hyper-parameter. Then, we train π_θ using $\mathcal{L}_{\text{rf}}(\pi_\theta)$ instead of naive DPO (Eq. 6).

De-coupled noise preference detection: While learning with the refined preference label reduces the risk of learning π_θ the noisy preference, its effectiveness could be limited as the model π_θ for noise detection originated from the label generation model π_{i-1}. Therefore, to further improve the effectiveness of our preference label refinement framework, we introduce the de-coupled noise detection (Han et al., 2018; Li et al., 2020) technique for LLM alignment. Specifically, we identify the preference noise by mimicking the preference prediction of a more strongly aligned LLM $\pi_{\widetilde{\theta}}$: [4]

$$z_{\widetilde{\theta}} = 1 \text{ if } p_{\widetilde{\theta}}(y_w \succ y_l|x) < \tau \text{ else } z_{\widetilde{\theta}} = 0. \tag{11}$$

With this de-coupled identification, π_θ is trained with refined preference labels via Eq. 10 , *i.e.*, $z_{\widetilde{\theta}}$ is used to substitute z_θ in Eq. 10. Here, we obtain the prediction of $\pi_{\widetilde{\theta}}$ by approximating its logit $h_{\widetilde{\theta}}$ through the linear combination of the logits of π_θ and π_{ref}. [5] It is motivated by the recent work (Liu et al., 2024) that shows the aligned models via RLHF with varying β are geometric mixtures of a reference model and a single aligned model:

$$h_{\widetilde{\theta}}(x, y_{1:t-1}) = (1+\lambda) * h_\theta(x, y_{1:t-1}) - \lambda * h_{\text{ref}}(x, y_{1:t-1}), \tag{12}$$

where $\lambda > 0$ is a hyper-parameter and $y_{1:t-1}$ indicates the output sequence before t-th output.

We remark that this de-coupled noise identification by approximating $p_{\widetilde{\theta}}(y_w \succ y_l|x)$ *does not require additional computations* compared to DPO, since the required measurements h_θ and h_{ref} are obtained during the calculation of the original DPO objective (Eq. 6). Therefore, SPA only requires a few lines of additional code to the original DPO codebase. We present full procedure of SPA in Algorithm 1.

[4] With λ in Eq. 12, $\pi_{\widetilde{\theta}}$ is equivalent to model trained with $(1 + \lambda)$ times smaller KL term than π_θ via Eq. 3.
[5] When $\pi_\theta(\cdot|x) := \text{Softmax}\big(h_\theta(x)\big)$, we refer $h_\theta(x)$ as the logit of LLM π_θ for the given input x.

5 EXPERIMENTS

In this section, we present our experimental results to answer the following question:

- Does SPA improve the alignment of LLMs only using a small amount of human-labeled preference data? (Table 1, Figure 4)
- Does the proposed method outperform other preference labeling methods? (Table 2, Figure 3)
- Is SPA generalizable across various choices of seed data and types of LLMs? (Tables 3,4,5)
- What is the effect of each component in SPA? (Tables 6,7)

5.1 EXPERIMENTAL SETUPS

Models. When there are no specific mentions, our experiments were conducted using the supervised fine-tuned Mistral-7b-0.1 model (Jiang et al., 2023a), as the initial model π_{init} in Section 4. Specifically, we use the open-sourced model[6] that follows the recipe of Zephyr (Tunstall et al., 2023) and fine-tuned on the instructions of Ultrachat (Ding et al., 2023). More details are in Appendix B.

Baselines. To evaluate the effectiveness of the proposed preference judgment method (Eq. 7), we compare it with other preference judgment methods. Specifically, we consider the baselines that train the model via Iterative DPO (Snorkel, 2024; Xu et al., 2023), which iteratively generate preference data and update the model, using LLM-as-judge (Bai et al., 2022b; Zheng et al., 2023) (*i.e.*, in-context learning) or an external powerful reward model (PairRM (Jiang et al., 2023b)) for the preference judgment. Notably, these approaches are the same in the case of changing the judgment method and removing self-refinement in SPA. Details are presented in Appendix B.

Datasets. For the preference learning dataset, we utilized UltraFeedback (Cui et al., 2023), following the previous works (Snorkel, 2024; Rosset et al., 2024).[7] To be specific, from this dataset, we first construct the seed data, consisting of 2K samples (3.3% of 60K) with prompts, responses, and ground truth preference labels. We refer the ground-truth preference label provided by the UltraFeedback as *gold label* in Tables 1 and 5. Then, the remaining samples are divided into subsets of 8K, 20K, and 30K samples, leaving only the prompts. These subsets were used as the prompt sets for the iteration stages 1, 2, and 3, respectively. Only for the experiments in Table 3, the size of seed data is changed.

Evaluations. Following the common practice in LLM alignment, we mainly evaluate each model our evaluations using (1) AlpacaEval 2.0 (Dubois et al., 2023; 2024; Li et al., 2023a). AlpacaEval 2.0 approximately evaluates human preference for instruction following. Using 805 instructions from various datasets, the evaluation is conducted by comparing the response of GPT-4 (OpenAI, 2023) and the testing model to measure win rates. To mitigate the length bias of LLM's preference (Wang et al., 2023b; Zheng et al., 2023), both original and length-controlled (LC) win rates are simultaneously measured. LC win rate is an adjusted win rate by neutralizing the effect of response length to focus on quality, using a separately trained regression model (Dubois et al., 2024). We also evaluate trained LLMs using (2) MT-Bench (Zheng et al., 2023) to assess different aspects of LLMs. Namely, MT-Bench evaluates a chatbot's overall abilities across multiple categories related to key LLM capabilities such as math, coding, roleplay, writing, etc. The evaluation is conducted by scoring responses to multi-turn questions using GPT-4. These benchmarks also provide a thorough evaluation of LLMs' alignment with human preferences and their overall effectiveness in practical applications.

Implementation details. After the initialization stage, we conduct three rounds of data expansion with self-generated preference data. For data expansion, we sampled 2 responses independently per each prompt with a temperature of 0.7. Then, using the SFT model as the reference model, we assign the preference label (Eq. 7). The initial DPO training to obtain π_0 was conducted for 3 epochs on the seed dataset. Training on each subsequent iteration was carried out for 1 epoch. For the hyper-parameter β of DPO, we used a fixed value of $\beta = 0.1$. The batch size was set to 32, and the learning rate was 5×10^{-7}. We employed AdamW optimizer and a cosine learning rate scheduler with a warm-up phase corresponding to 10% of the total training steps. For the hyper-parameters α and $K\%$ for SPA, we used fixed values of $\alpha = 0.1$ and $K = 10$. Additionally, a warm-up phase was included in the denoising stage, with denoising activated after 20% of the total training steps had been completed. Regarding the hyper-parameters λ for de-coupled noise detection, we utilized the progressively reduced values of 1/2, 1/4, and 1/8 for iterations 1, 2, and 3, respectively.

[6] `alignment-handbook/zephyr-7b-sft-full`
[7] `"argilla/ultrafeedback-binarized-preferences-cleaned"`

Table 1: **Main results.** Evaluation results on AlpacaEval 2.0 and MT-Bench with different variants of Mistral-7B-v0.1. The best scores are highlighted with **bold**.

Models	Gold Label (%)	AlpacaEval 2.0 Len-control. Win Rate (%)	AlpacaEval 2.0 Win Rate vs. GPT-4 (%)	MT-Bench Avg. Score (0-10)
Mistral-7B-v0.1	-	0.17	0.50	3.25
Zephyr-7b-β	100	11.75	10.03	6.87
SFT	-	7.58	4.72	6.34
DPO	3.3	9.03	7.68	6.81
SPA (Ours)	3.3	**15.39**	**21.13**	**6.94**

Table 2: **Comparison with baselines for preference judgment.** Evaluation results on AlpacaEval 2.0 and MT-Bench with iteratively trained models (from SFT model) under different preference judgment methods. The best scores are highlighted with **bold**.

Methods	External Model	AlpacaEval 2.0 Len-control. Win Rate (%)	AlpacaEval 2.0 Win Rate vs. GPT-4 (%)	MT-Bench Avg. Score (0-10)
Iterative DPO (PairRM)	✓	11.87	9.46	**6.98**
Iterative DPO (LLM-as-judge)	✗	9.28	9.18	6.67
SPA (Ours)	✗	**15.39**	**21.13**	6.94

5.2 Main results

After completing 3 iterations of data expansion and fine-tuning via SPA, the trained model achieved a 21.13% win rate against GPT-4 on the AlpacaEval 2.0 benchmark, as presented in Table 1. This represents a significant improvement compared to the 7.68% (7.68% $\rightarrow$ 21.13%) win rate achieved when using only 3.3% of labeled data with the standard DPO training, while the length-control win rate is also improved. (9.03% $\rightarrow$ 15.39%). In addition, SPA achieved a score of 6.94 on the MT-Bench, clearly outperforming the model trained with DPO (6.81) on the same amount of 3.3% gold labeling data. More interestingly, our framework achieved superior performance in both win rate (10.03% vs 21.13%) and length-control win rate (11.75% vs 15.39%), compared to Zephyr-7b-β which uses same base model (Mistral-7B-0.1v) and SFT dataset but uses significantly larger labeled preference data, *i.e.*, 100% of UltraFeedback dataset (v.s. 3.3% for SPA). These significant improvements in both win rates clearly affirm the overall enhancement in performance from SPA.

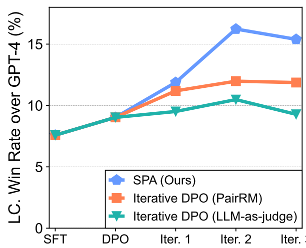


Figure 3: **Improvements during iterations.** Length control (LC.) win rate (%) measured by AlpacaEval 2.0 is consistently improved by SPA and it outperforms other baselines.

Next, in Table 2, we present additional experimental results to validate the proposed preference judgment method. Namely, three experiments in Table 2 can be viewed as the Iterative DPO variants with different preference judgment methods. One can observe that SPA showed significantly better performance compared to other methods. Specifically, SPA achieved a win rate of 21.13% against GPT-4 on AlpacaEval 2.0, compared to 9.46% for the baseline with an external reward model, PairRM. In terms of length control win rate, SPA achieved 15.39%, surpassing the reward model's 11.84%. Here, we conjecture that the reason why the Iterative DPO training with the proposed direct preference judgment method (using training LLM) outperforms the case with inferred labels from the external reward model is related to the distribution shift. As the iteration is increased, the distribution of the generated data with LLM is more shifted from the distribution of the seed preference data. Then, the effectiveness of the external reward model inevitably decreases, as the

Table 3: **Different number of seed data.** Evaluation results on AlpacaEval 2.0 with Mistral-7B-v0.1 trained with DPO and SPA under the different number of seed ground-truth preference labels.

	Used Ground-truth Preference Data			
Methods	0.8%	1.7%	3.3%	10%
DPO: LC Win Rate (%)	7.85	7.68	9.03	11.37
DPO: Win Rate (%)	5.53	5.49	7.68	9.32
SPA: LC Win Rate (%)	10.36	12.36	16.23	18.52
SPA: Win Rate (%)	11.34	13.72	19.94	23.79

reward model is fixed while the generated data is increasingly distant from its training distribution. In contrast, SPA generates the preference label using the intrinsic reward model that is continuously updated for each iteration. Therefore, it less suffers from the distribution shift during the iterations, and hence could be more effective for iterative training. Regarding this, we remark on the results in Figure 3; at iteration 1, the effectiveness of both approaches is not much different. However, the gap is significantly widened at iteration 2, and it empirically supports the above rationale.

On the other hand, the in-context learning approach (LLM-as-judge) shows a similar win rate compared to PairRM, but falls short in length control win rate (11.87% vs 9.28%), showing the limitations of the LLM-as-judge approach. Overall, the results reveal the superiority of our direct preference judgment over other judgment methods. Also, this superiority is consistently observed through the iterations, as shown in Figure 3.

5.3 More analyses

In this section, we conduct additional analyses of SPA by comparing the results on AlpacaEval 2.0. More comparisons on the MT-Bench and the additional experiments are presented in the Appendix.

Generalization across different numbers of seed data. Previously, we conducted the experiment by assuming that only a limited number of human preference data is initially given, *e.g.*, 3.3% of UltraFeedback dataset. However, the effectiveness of SPA does not depend on the size of the seed preference dataset and we validate this with the additional experiments. First, we conduct the experiments by varying the portion of the seed ground-truth preference data. Specifically, to use the fixed input prompt datasets for each iteration, we consider the following portions for the experiments: [0.8%, 1.7%, 10%]. Table 3 shows the results on AlpacaEval 2.0 with Mistral-7B-v0.1 after 2 iterations of training with SPA, including the original experiments with 3.3% seed preference data. Here, one can observe that the alignment performance under DPO and SPA is improved with the increased seed data, and SPA consistently outperforms DPO which demonstrates the robustness of SPA regarding the size of seed preference data.

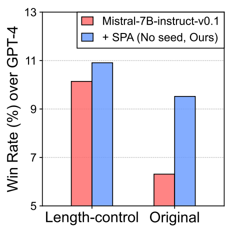


Figure 4: **Improvements without seed data.** Evaluation results on AlpacaEval 2.0 with Mistral-7B-instruct-v0.1 and SPA with no seed preference data.

We further evaluated the feasibility of using SPA even *without seed preference data*. Namely, we want to answer whether LLM can derive explicit human preference between responses, by leveraging their intrinsic knowledge learned about humans, during the previous training, such as pre-training or supervised instruction tuning (SFT). For this experiment, we used the Mistral-7b-instruct-0.1v (Jiang et al., 2023a) as the initial model (*i.e.*, π_0) and the Mistral-7b-0.1v-base as the reference model (*i.e.*, π_{init}) (see the initial setup in Section 4). This setup allows us to demonstrate that our framework can function effectively even in the absence of seed preference data, when the model is sufficiently fine-tuned with iterative data expansion and learning through self-refinement. As shown in Figure 4, the win rate increased from 6.31% to 9.79%, and the length-control win rate improved from 10.14% to 11.59%. This result indicates that SPA can leverage the internal information of LLMs to be aligned with human preference even without seed data.

Table 4: **Different initial seeds.** Evaluation results on AlpacaEval 2.0 with different variants of Mistral-7B-v0.1 under the different sampling of the initial seed preference data.

Methods	1st Seed Data	2nd Seed Data	3rd Seed Data	Average	Variance
DPO: LC Win Rate (%)	9.03	8.74	9.54	9.10	0.16
DPO: Win Rate (%)	7.68	7.17	7.59	7.48	0.07
SPA (Ours): LC Win Rate (%)	16.23	13.77	16.38	15.46	2.10
SPA (Ours): Win Rate (%)	19.94	20.06	19.74	19.91	0.03

Table 5: **Compatibility across various LLMs.** Evaluation results on AlpacaEval 2.0 with different training methods (SFT, DPO, and SPA) across various types of LLMs (Phi-2-2.7B, LLaMA-3-8B, and Phi-3-14B). The best scores are highlighted with **bold**.

Methods	Gold Label (%)	Phi-2-2.7B		LLaMA-3-8B-Instruct		Phi-3-14B-Instruct	
		Len-control. Win Rate (%)	Win Rate vs. GPT-4 (%)	Len-control. Win Rate (%)	Win Rate vs. GPT-4 (%)	Len-control. Win Rate (%)	Win Rate vs. GPT-4 (%)
SFT	-	5.88	3.78	21.40	21.71	26.51	21.41
DPO	3.3	7.02	5.67	24.17	25.39	27.70	22.12
SPA (Ours)	3.3	**9.10**	**9.43**	**25.03**	**34.84**	**28.77**	**24.14**

Variance with different initial seed dataset. In addition, we conduct experiments to check the sensitivity of SPA with the initial seed preference dataset by varying them with different random sampling. The results after 2 iterations of training with SPA are presented in Table 4. Here, one can observe that the proposed SPA consistently improves the alignment performance regardless of the given seed data, and the variance between them is not significant, especially in the case of a normal win rate. While ours exhibits a relatively high variance for length-controlled (LC) win rate, its lowest confidence interval value (13.36 %) is certainly higher than the value of the strongest baseline (11.98 %) which confirms the effectiveness of our method.

Compatibility with different models. Next, to verify the compatibility of our framework across various LLMs, we conducted experiments using three different LLMs: Phi-2-2.7B (Li et al., 2023b), LLaMA3-8B (Dubey et al., 2024), and Phi-3-14B. Specifically, we conducted experiments based on their supervised fine-tuned versions; for Phi-2, we used the model that has been fine-tuned on the UltraChat dataset like Mistral.[8] For LLaMA-3[9] and Phi-3[10], we used the generally fine-tuned models as there are no models that have been fine-tuned on the UltraChat dataset. Here, most of the experimental setups for these experiments are maintained, and the slightly adjusted setups are detailed in Appendix B.3. As shown in Table 5, the experimental results showed that applying SPA to various LLMs yields consistent improvements in the performance. For example, the win rate improved from 5.67% to 9.43%, and the length control win rate increased from 7.02% to 9.1%, in the case of Phi-2 after being trained with SPA compared to DPO. These results demonstrate that the effectiveness of SPA is not limited to the specific LLMs and is generalized across various LLMs.

Ablation study. To evaluate the impact of the self-refinement components, we conducted ablation experiments by excluding both self-refinement (SR) and decoupled noise detection (DND) from the existing framework. The results are presented in Table 6. With self-refinement without decoupled noise detection (Eq. 10), we observed a slight performance improvement, with the win rate against GPT-4 marginally increasing from 19.91% to 19.94%, and the length control win rate rising from 14.41% to 14.7%. But, when the decoupled noise detection is incorporated into the self-refinement (Eq. 11), we observed significant improvements, with the win rate increasing from 19.91% to 21.13% and the length control win rate improving from 14.41% to 15.39%. Also, these results confirm that the self-refinement component is a crucial factor in enhancing performance, contributing to both higher win rates and better length control.

Additional analysis with judgment methods. In Table 7, we further analyzed the impact of the reference model in the preference judgment process in Eq. 7. This analysis was conducted during

[8] lole25/phi-2-sft-ultrachat-full
[9] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[10] https://huggingface.co/microsoft/Phi-3-medium-4k-instruct

Table 6: **Ablation study.** Evaluation results on AlpacaEval 2.0 with iteratively trained models (from SFT) under different methodological configurations of SPA. DE, SR, DND are abbreviations of data expansion, self-refinement, and de-coupled noise detection, respectively. The best scores are highlighted with **bold**.

Methods	DE	SR	DND	AlpacaEval 2.0 Len-control. Win Rate (%)	AlpacaEval 2.0 Win Rate vs. GPT-4 (%)
SFT	-	-	-	7.58	4.72
DPO	-	-	-	9.03	7.68
SPA (Ours)	✓	✗	✗	14.41	19.91
	✓	✓	✗	14.7	19.94
	✓	✓	✓	**15.39**	**21.13**

Table 7: **Additional analyses.** Evaluation results on AlpacaEval 2.0 with models that fine-tuned with different judgment methods, from the resulting model of 1st iteration of SPA.

Models	AlpacaEval 2.0 Len-control. Win Rate (%)	AlpacaEval 2.0 Win Rate vs. GPT-4 (%)
SPA after iteration 1	10.57	11.89
Eq. 7 with initial SFT model (Ours)	15.08	19.56
Eq. 7 with previous model	13.73	17.66
Judgment with PairRM	13.57	13.72
Judgment without reference model	12.83	12.35

the transition from iteration 1 to iteration 2, where the most significant performance changes were observed (*i.e.*, we fine-tune from the resulting model of iteration 1). To isolate and compare the effect of judgment methods, we followed the setup in Table 2 and so excluded the influence of the self-refinement component. Then, we experimented with three setups by varying the judgment method using (1) the current policy from the previous iteration as the reference model, (2) performing judgment without any reference model, and (3) using the PairRM for judgment.

The results are presented in Table 7. Here, the experimental results demonstrated that the method used in SPA, where the SFT model was utilized as the reference model for preference judgment, achieved the highest performance increase. Specifically, using the model from the previous iteration as the reference model showed lower performance, with a relatively larger decrease in the length control win rate (15.08% vs 13.73%) compared to the win rate (19.56% vs 17.66%). Despite these decreases, it still outperforms using PairRM. These results may imply the importance of judging the preference through the training LLM rather than the external model, as it is less suffering from the distribution mismatch. However, without reference model (*i.e.*, only using the likelihood of the current model), the performance increase was the lowest compared to all other cases. These findings underscore the substantial impact of the choice of proper judgment method and reference model.

6 Conclusion

In this paper, we proposed SPA, a method that can efficiently improve the alignment of LLMs using minimal human-labeled preference data. Our main contributions include the development of an effective data expansion method with the direct preference judgment method and a preference learning algorithm with the self-refinement of (potentially) noise preference. We demonstrate the effectiveness of SPA by fine-tuning the recent LLMs with the various setups, and observing the significant improvements when evaluating them on the commonly used benchmarks, AlpacaEval 2.0 and MT-Bench. We expect SPA to make significant contributions to future research and practical applications, especially when the human-labeled preference is hard to collect. Limitations and societal impacts are further discussed in Appendix A.

Reproducibility Statement

For the reproducibility of our results, we have provided a detailed description of our methods and experimental setups in Section 5.1 and Appendix B. We also confirmed the robustness of our results through the experiment (Table 4). In addition, to further facilitate the reproduction, we will release our codes and the checkpoints for the trained models.

Acknowledgments

This work was mainly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub, 50%; RS-2022-II220959, Few-shot Learning of Casual Inference in Vision and Language for Decision Making, 50%).

References

Anthropic. Introducing the next generation of claude. https://www.anthropic.com/news/claude-3-family, 2024.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems*, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. Aligning language models with preferences through f-divergence minimization. In *International Conference on Machine Learning*, 2023.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, 2018.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023a.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Annual Conference of the Association for Computational Linguistics*, 2023b.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, 2021.

Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023a.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023b.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base llms: Rethinking alignment via in-context learning. In *International Conference on Learning Representations*, 2024.

Tianlin Liu, Shangmin Guo, Leonardo Bianco, Daniele Calandriello, Quentin Berthet, Felipe Llinares, Jessica Hoffmann, Lucas Dixon, Michal Valko, and Mathieu Blondel. Decoding-time realignment of language models. In *International Conference on Machine Learning*, 2024.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems*, 2024.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 2019.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. In *arXiv*, 2021.

OpenAI. Introducing chatgpt. *https://openai.com/blog/chatgpt*, 2022.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning*, 2007.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacroce, Ahmed Awadallah, and Tengyang Xie. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*, 2024.

Snorkel. New benchmark results demonstrate value of snorkel ai approach to llm alignment. `https://snorkel.ai/new-benchmark-results-demonstrate-value-of-snorkel-ai-approach-to-llm-alignment`, 2024.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, 2020.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. In *NeurIPS*, 2020.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR, abs/2312.08935*, 2023a.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? exploring the state of instruction tuning on open resources. In *Advances in Neural Information Processing Systems*, 2023b.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022b.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*, 2024.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A LIMITATION AND SOCIETAL IMPACT

A.1 LIMITATION AND FUTURE WORK

In the experiments, SPA has shown the tendency to increase the responses' length (please see Appendix D for the relevant results and discussions). We demonstrated that the improvement by SPA is not a simple result of such length increase, by observing the increase of win rate under a length length-controlled setup or MT-bench. However, depending on the user, this behavior could be dispreferred. In this sense, focusing on mitigating this bias during the self-improving alignment will be an interesting future direction, and can enhance the robustness and generalizability of SPA across more diverse scenarios.

A.2 SOCIETAL IMPACT

SPA enables efficient human preference learning, allowing for cost-effective training of models in data-scarce or domain-specific areas. Our framework supports alignment learning in various fields, including multilingual language learning and preferences beyond human helpfulness. Consequently, it could contribute to facilitating the widespread adoption of LLM technology across diverse sectors. By lowering the barriers to alignment learning, SPA makes it more accessible to a broader audience. However, the widespread availability of this technology also brings potential risks. The reduced cost of training models could enable malicious actors to misuse the technology, leading to societal issues. Therefore, it is crucial to implement ethical considerations and safety measures when deploying SPA technology to mitigate these risks.

B MORE DETAILS OF EXPERIMENTAL SETUPS

B.1 SFT MODEL SETUP

Mistral. For supervised fine-tuning, Ultrachat dataset (Ding et al., 2023) is used[11], batch size was set 128, total epoch was 1, and the learning rate was 2×10^{-5}. It employed Adam optimizer (Kingma & Ba, 2015) and a cosine learning rate scheduler with a warm-up phase corresponding to 10% of the total training steps.

Phi-2. For supervised fine-tuning, Ultrachat dataset is used, batch size was set 64, total epoch was 3, and the learning rate was 2×10^{-5}. It employed Adam optimizer and a cosine learning rate scheduler with a warm-up phase corresponding to 10% of the total training steps.

LLaMA-3 and Phi-3. As described in Section 5.3, we use the generically instruct-tuned versions for both LLaMA-3-8B and Phi-3-14B, as there are no SFT models tuned on Ultrachat dataset.

B.2 BASELINES EXPERIMENT SETUP

Zephyr-7b-β. We implemented Zephyr-7b-β (Tunstall et al., 2023), which is compared in Table 1, according to recipes. Our Zephyr-7b-β was trained using the same pre-trained model (mistral-7b-0.1v (Jiang et al., 2023a)) and the same SFT data (Ultrachat (Ding et al., 2023)), but there are marginal differences compared with recipes. We use SFT [12] models which trained with different recipes. Specifically, Zephyr-7b-β's SFT used the batch size of 512, but 128 was used for the ours SFT model. In addition, regarding the preference dataset, Zephyr-7b-β was trained using the original Ultrafeedback (Cui et al., 2023) [13] but we use cleaned version[14]. These changes in training data and the SFT model were aligned with SPA to ensure a fair comparison.

LLM-as-Judgement. For LLM-as-judge, we used an SFT model to employ Consitual AI's pairwise comparison prompt for judging preferences (Bai et al., 2022a). Preference is measured by comparing the logprob value of the token output as input to the following prompt (Listing 1). To ensure fair

[11] https://huggingface.co/datasets/HuggingFaceH4/ultrachat_200k
[12] https://huggingface.co/alignment-handbook/zephyr-7b-sft-full
[13] https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized
[14] https://huggingface.co/datasets/argilla/ultrafeedback-binarized-preferences-cleaned

comparison and prevent low judgment performance, evaluation instructions were created using seed preference data which is the same form as Consitual AI's pairwise comparison. (Listing 2) Using these, additional SFT learning is performed to obtain an independent LLM-as-judge model. For this supervised fine-tuning, we set the batch size 32, total epoch is 3, and the learning rate was 2×10^{-5}. We employed Adam optimizer and a cosine learning rate scheduler with a warm-up phase corresponding to 10% of the total training steps.

Reward model judgment. For the reward model baseline, we selected PairRM (Jiang et al., 2023b) due to its high performance on AlpacaEval 2.0 (Snorkel, 2024; Wu et al., 2024). Unlike SPA, which was trained on only 2K gold label data, PairRM was trained on a large-scale dataset. The training data for PairRM includes the following:

- openai/summarize_from_feedback (Stiennon et al., 2020)
- openai/webgpt_comparisons (Nakano et al., 2021)
- Dahoas/synthetic-instruct-gptj-pairwise[15]
- Anthropic/hh-rlhf (Bai et al., 2022a)
- lmsys/chatbot_arena_conversations (Zheng et al., 2023)
- openbmb/UltraFeedback (Cui et al., 2023)

The total number of pairwise samples in this training data is approximately 500K, compared to 2K for SPA. Specifically, the summarize_from_feedback dataset contributes 179K samples, and the hh-rlhf dataset contributes 161K samples, making up a significant portion of the total.

B.3 Adjusted experimental setups for different LLMs

In Table 5, we conduct the experiments with different LLMs. As they exhibit different characteristics from the difference in backbone and sizes, we slightly adjusted the experimental setups while keeping most identical to the setups in Section 5.1.

Phi-2. We slightly adjust the learning rate to accommodate the different characteristics of the Phi-2 (5×10^{-6}). In addition, due to the smaller size of the Phi-2, we observe that performance improvements were not evident beyond iteration 2. Therefore, we present the results of iteration 1.

LLaMA-3 and Phi-3. We slightly adjust the learning rate to accommodate the different characteristics of models (1×10^{-5}). We conduct 1 epoch for the initial DPO training and maintain $\beta = 0.01$ throughout the entire training process. Since performance improvement has been only observed up to iteration 2 in Section 5.2. we conduct the experiments up to iteration 2 for these models.

B.4 Implementation details

Resources and computation cost. For all experiments, we utilized 4 A6000 GPUs. Under this computational resource, generating responses for 10K prompts takes approximately 1 to 2 hour, and preference judging for generated responses also takes about 1 to 2 hour. For training of model with Eq. 10, it takes about 1 to 2 hours per epoch. Therefore, the total time required to complete response generation, preference judgment, and one epoch of training was between 5 to 6 hours per 10K prompt.

Response generation. To mitigate the length bias from training with Direct Policy Optimization (DPO), we restricted the maximum token length for self-generated responses to 300 tokens. This limit corresponds to approximately 900 characters.

Refinement. Using the simple moving average (SMA) method, we measured the confidence over the most recent 50 samples. Based on this, we calculated the boundary corresponding to the lower K% of the confidence distribution (Eq. 11).

[15] https://huggingface.co/datasets/Dahoas/synthetic-instruct-gptj-pairwise

Table 8: **MT-Bench.** Evaluation results on MT-bench with different models. $\mathrm{SPA}_{\texttt{inst}}$ and $\mathrm{SPA}_{\texttt{phi}}$ denotes the fine-tuned Mistral-7B-Instruct-v0.1 and Phi-2 SFT with SPA, respectively.

Models	Gold Label (%)	Avg Score
Mistral-7B-Instruct-v0.1	-	6.78
$\mathrm{SPA}_{\texttt{inst}}$ (Ours)	-	**7.12**
Phi-2 SFT	-	5.35
Phi-2 DPO	3.3	6.16
$\mathrm{SPA}_{\texttt{phi}}$ (Ours)	3.3	**6.33**

```
f'''
Consider the following conversation between a human and an assistant:
[prompt]
Please choose the response that is the most helpful, honest, and
↪  harmless.
Options:
(A) [response 1]
(B) [response 2]
The answer is:
'''
```

Listing 1: Prompt used for judging preferences.

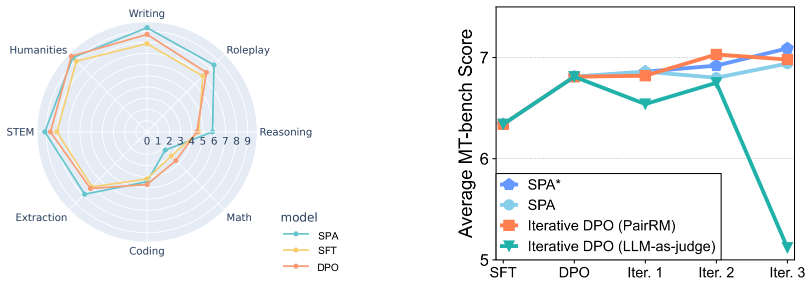
```
f'''
Consider the following conversation between a human and an assistant:
[Horses are hybrids of which two animals?]
Please choose the response that is the most helpful, honest, and
↪  harmless.
Options:
(A) [Horses, scientifically known as Equus caballus, are not the result
↪  of hybridization between two other animals. They are considered a
↪  species of their own within the Equidae family, which also includes
↪  other equines such as donkeys, mules, and zebras. Horses are
↪  believed to have evolved from a small, multi-toed forest-dwelling
↪  animal known as Eohippus, which lived around 60 million years ago.
↪  Over millions of years, these horses evolved into the large,
↪  single-toed animals we know today.]
(B) [Horses are not hybrids of any other animals. They are a unique
↪  species and are not a product of cross-breeding between different
↪  animals. Horses are domesticated animals and have been bred and
↪  developed over thousands of years by humans for various purposes
↪  such as transportation, riding, and agriculture.]
The answer is: (A)

'''
```

Listing 2: Evaluation instruction.

C More Evaluation Results in MT-Bench

In this section, we further provide the additional evaluation results on MT-Bench (Zheng et al., 2023), which couldn't be presented in the main text, due to the space issue.

We first present (a) task-wise evaluation results and (b) iteration-wise average improvement in Figure 5. As shown in Figure 5a, SPA consistently improves the performance in various tasks. Notably, there is almost no gain in Coding and degradation in Math. We remark that this phenomenon is



(a) MT-bench task-wise evaluation results

(b) Iteration-wise improvement with MT-bench

Figure 5: **MT-bench Evaluation.** More evaluation results with MT-bench.

Table 9: **Ablation study including MT-Bench.** Evaluation results on AlpacaEval 2.0 and MT-Bench with iteratively trained models (from SFT) under different methodological configurations of SPA. DE, SR, DND are abbreviations of data expansion, self-refinement, and de-coupled noise detection, respectively. The best scores are highlighted with **bold**.

				AlpacaEval 2.0		**MT-Bench**
Methods	DE	SR	DND	Len-control. Win Rate (%)	Win Rate vs. GPT-4 (%)	Avg. Score (0-10)
SFT	-	-	-	7.58	4.72	6.34
DPO	-	-	-	9.03	7.68	6.81
SPA (Ours)	✓	✗	✗	14.41	19.91	6.86
	✓	✓	✗	14.7	19.94	**7.09**
	✓	✓	✓	**15.39**	**21.13**	6.94

commonly observed in the relevant literature (Lin et al., 2024), which indicates that different training (Wang et al., 2023a) or inference (Wei et al., 2022b) schemes might be necessary to improve the performance in these tasks.

Next, in Figure 5b, one can observe that the average performance on the MT-bench is increased with more iterations. Specifically, while the Iterative DPO using PairRM shows the best performance until iteration 2, SPA* (without DND) outperforms it in iteration 3. It demonstrates the effectiveness of our framework for iteratively improving the alignment of LLM.

In addition, we measure the performances of Phi-2 variants and Mistral-7B-Instruct-v0.1 variants on MT-Bench in Table 8; these models are presented in Table 5 and Figure 4, respectively. As one can see, SPA consistently yields the improvement across different backbones of Mistral-7B-Instruct-v0.1 and Phi-2. Lastly, we present the full results of the ablation study (presented in Table 6) that includes the evaluation results on MT-Bench, in Table 9.

D More Quantitative Results

In this section, we present more quantitative results to demonstrate the effectiveness of SPA.

Mitigating length bias with SPA. Here, we provide a discussion of the relevant experimental results about the length bias present in SPA. During the experiments, we observe that LLMs trained with SPA tend to generate longer responses (see 10), which could be dispreferred depending on the user. Regarding this, we first emphasize that the improvement with SPA is not merely due to longer outputs, as shown by the significant gains in the length-controlled win rate in all experiments in Section 5.

Nevertheless, to further address the concerns regarding this issue, we further investigate whether previously researched length control techniques can be easily integrated into SPA. Specifically, we

Table 10: **SPA with length regularization.** Evaluation results on AlpacaEval2.0 with different variants of Mistral-7B-v0.1 from SPA and the additional length regularization term.

Models	Gold Label (%)	Len-control. Win Rate	Win Rate vs. GPT-4	Avg. len (# chars)
Mistral-7B-v0.1	-	0.17	0.50	5692
SFT	-	7.58	4.72	901
DPO	3.3	9.03	7.68	1802
Zephyr-7b-β	100	11.75	10.03	1552
SPA (Original, Iter. 1)	3.3	11.88	12.95	2150
SPA (Modified, Iter. 1)	3.3	11.39	12.31	**2013**
SPA (Original, Iter. 2)	3.3	16.23	19.94	2749
SPA (Modified, Iter. 2)	3.3	14.46	18.23	**2448**

Table 11: **LLM-as-Judgment with model from previous iteration.** Evaluation results on AlpacaEval 2.0 with different variants of Mistral-7B-v0.1.

Methods	Len-control. Win Rate (%)	Win Rate vs. GPT-4 (%)
LLM-as-judge (Iter. 1)	8.88	8.01
LLM-as-judge (Iter. 2, orig)	9.49	8.46
LLM-as-judge (Iter. 2, prev. init)	9.74	10.09
SPA (Iter. 2, ours)	**15.46**	**19.91**

apply the length penalty approach from RLHFlow Dong et al. (2024). This method heuristically reduces the reward model's reward based on the output length (Eq. 13) during preference labeling. We utlze hyperparamter α between 0.001 to 0.01 that minimize the length increase. The results, shown in Table 10, indicate that this modification successfully reduces the average length while largely preserving the performance improvements from SPA. These results demonstrate that SPA can be easily integrated with existing research related to length control.

$$r_{\texttt{penalty}}(x, y) = r(x, y) - \alpha|y| \tag{13}$$

LLM-as-Judge with the model from previous iteration. For the LLM-as-Judgement baseline, we used a fixed model fine-tuned specifically for evaluating preferences between responses, as described in Appendix B.2. On the other hand, at each iteration, the proposed SPA utilizes the previous iteration's model, π_{i-1}, for the direct preference judgment. Therefore, one can be curious how this difference affect to the performance.

To answer this, we conducted new experiments; at the 2nd iteration, the evaluation model was initialized with the resulting model from the 1st iteration and fine-tuned with the constructed preference evaluation dataset using the seed preference data (see details in Appendix B.2). The evaluation results (denoted as *LLM-as-Judge (Iter. 2, prev. init)*) on AlpacaEval 2.0 are presented in Table 11, along with other methods at the 2nd iteration. While this approach yielded improved alignment compared to the fixed model, SPA still significantly outperformed this baseline. This underscores that the superior performance of SPA arises from its novel preference evaluation techniques rather than the specific evaluation model used.

E Qualitative Examples

In this section, we present examples of the responses from the models trained under different methods (SFT, DPO, and SPA), for a qualitative comparison. The results are presented in Figures 6, 7, and 8. As one can see in the examples, the model trained under SPA provides more detailed and informative responses to various questions.

Prompt:
Who is Larry Page?

SFT response:
Larry Page is the co-founder and CEO of Google, a multinational technology company that provides various services such as search engine, online advertising, and cloud computing. He was born on March 26, 1973, in East Lansing, Michigan, and graduated from the University of Michigan with a degree in computer engineering. Page and Sergey Brin founded Google in 1998, and since then, it has become one of the most successful and influential companies in the world.

DPO response:
Larry Page is an American computer scientist and entrepreneur who co-founded Google, one of the world's largest technology companies, with Sergey Brin in 1998. He is the CEO of Alphabet Inc., the parent company of Google, and has been instrumental in the development and growth of the company. Page is known for his innovative ideas and his focus on developing new technologies that can improve people's lives. He has been recognized for his contributions to the field of computer science and has received numerous awards and honors, including the Presidential Medal of Freedom in 2015.

SPA response:
Larry Page is an American entrepreneur, computer scientist, and co-founder of Alphabet Inc., the parent company of Google, one of the world's largest technology conglomerates. Born on March 26, 1973, in East Lansing, Michigan, Page grew up in a family of academics and engineers. He earned his Bachelor's degree in Computer Engineering and Master's degree in Computer Science from Stanford University in 1995 and 1998, respectively.

In 1998, Page teamed up with Sergey Brin, his classmate at Stanford, to develop a new search engine called "Backrub," which later evolved into Google. Initially funded by venture capitalists, Google quickly gained popularity for its innovative search algorithm, user-friendly interface, and extensive collection of online content. Page served as the CEO of Google from its inception in 1999 until 2019, overseeing its rapid growth, expansion into new markets, and development of groundbreaking products and services, such as Android, Maps, YouTube, and Cloud Computing.

Under Page's leadership, Google became a dominant player in the tech industry, with a market capitalization of over $1 trillion and a global workforce of over 135,000 employees. In 2019, Page stepped down as CEO of Google and assumed the role of CEO of Alphabet, the holding company that reorganized Google's core businesses and diversified into new areas, including healthcare, artificial intelligence, and sustainability.

Aside from his entrepreneurial achievements, Page is known for his philanthropic efforts, particularly through the Larry Page Foundation, which supports research in renewable energy, education, and medical innovation. He has also been recognized for his contributions to the tech industry, receiving numerous awards, honors, and recognitions, including the Presidential Medal of Freedom, Time Person of the Year, and Forbes' Person of the Year.

Figure 6: **Example of responses in AlpacaEval 2.0.**

Prompt:
What causes the northern lights

SFT response:
The northern lights, also known as aurora borealis, are caused by the interaction between charged particles from the sun and the Earth's magnetic field. When the charged particles from the sun enter the Earth's atmosphere, they collide with the gases in the atmosphere, causing them to emit light. The colors of the northern lights are determined by the type of gas that is colliding with the charged particles. Green is the most common color, but other colors such as red, yellow, and blue can also be seen.

DPO response:
The northern lights, also known as aurora borealis, are caused by the interaction between charged particles from the sun and the Earth's magnetic field and atmosphere.

When the sun emits charged particles, such as electrons and protons, they travel through space towards the Earth. As they approach the Earth, they are deflected by the Earth's magnetic field, which creates a protective bubble around the planet.

When these charged particles enter the Earth's atmosphere, they collide with the gases in the atmosphere, such as oxygen and nitrogen, causing them to become excited and emit light. This light is what we see as the northern lights.

The colors of the northern lights depend on the type of gas that is being excited. For example, oxygen produces green and yellow lights, while nitrogen produces blue and red lights.

The intensity and frequency of the northern lights depend on the activity of the sun, which can vary over time. During periods of high solar activity, the northern lights can be seen more frequently and with greater intensity.

SPA response:
The northern lights, also known as aurora borealis, are a natural phenomenon caused by the interaction between charged particles from space and the Earth's magnetic field and atmosphere. The primary cause of the northern lights is the solar wind, a stream of charged particles (primarily electrons, protons, and alpha particles) emitted from the Sun's corona and solar atmosphere.

As the solar wind approaches the Earth, it encounters the magnetic field, which deflects and traps some of the charged particles in a region known as the magnetosphere. The magnetosphere extends outward from the Earth's surface to a distance of several hundred thousand kilometers and is shaped like a teardrop, with the magnetic field lines converging towards the polar regions.

When energetic particles from the solar wind enter the magnetosphere, they collide with the atoms and molecules in the Earth's upper atmosphere, primarily in the polar regions, where the magnetic field lines are strongest. These collisions ionize the atmospheric gases, creating a plasma region known as the ionosphere.

The ionization process releases energy in the form of electromagnetic radiation, including ultraviolet, visible, and radio waves. The visible auroral emissions occur when the energetic particles excite the electrons in the ionized gases, causing them to emit photons with specific wavelengths, resulting in the characteristic colors of the northern lights.

The colors of the aurora depend on the type of gas molecule involved in the collisions and the energy level of the excited states. Green and yellow auroras are typically associated with oxygen atoms, while red and blue auroras are caused by nitrogen molecules. Other gases, such as hydrogen, helium, and carbon dioxide, can also contribute to the auroral spectrum, resulting in additional colors like purple, pink, and white.

The intensity and frequency of auroral displays vary depending on solar activity levels, geomagnetic conditions, and seasonal factors. Solar maximum periods, when the Sun's activity is at its highest, tend to produce more frequent and intense auroral displays in both hemispheres. However, even during solar minimum periods, when solar activity is low, occasional auroral events can still occur in the polar regions.

Figure 7: **Example of responses in AlpacaEval 2.0.**

毕业论文（设计）文献综述和开题报告考核

导师对开题报告、外文翻译和文献综述的评语及成绩评定：

成绩比例	文献综述 占（10%）	开题报告 占（15%）	外文翻译 占（5%）
分值			

导师签名 ___________

年　月　日

学院盲审专家对开题报告、外文翻译和文献综述的评语及成绩评定：

成绩比例	文献综述 占（10%）	开题报告 占（15%）	外文翻译 占（5%）
分值			

开题报告审核负责人（签名/签章）___________

年　月　日