# 基于分层序列模型学习混合整数线性规划的割选择

**Zhihai Wang*[1]，Xijun Li*[1, 2]，Jie Wang†[1, 3]，Yufei Kuang[1],Mingxuan Yuan[2], JiaZeng[2], YongdongZhang[1,3], Feng Wu[1,3]** 中国科学院 GIPAS 技术重点实验室，中国科学技术大学，[2] 诺亚方舟实验室，华为技术有限公司，[3] 人工智能研究院，合肥综合性国家科学中心

arXiv:2302.00244v1 [cs.LG] 1 Feb 2023

摘要

割平面（割）对于求解混合整数线性规划问题（MILPs）至关重要，这些问题刻画了广泛的重要现实世界应用。割选择旨在从候选割中选取适当的子集，以提高混合整数线性规划问题的求解效率；它在很大程度上取决于**（P1）**应优先选择哪些割，以及**（P2）**应选取多少个割。尽管许多现代混合整数线性规划求解器通过人工设计的启发式方法处理**（P1）-（P2）**，机器学习为从特定应用中收集的混合整数线性规划问题中学习更有效的启发式方法提供了有前景的途径。然而，许多现有的基于学习的方法侧重于学习哪些割应优先选择，却忽视了学习应选取割数量的重要性。此外，我们从大量实证结果中观察到，**（P3）**应优先选择何种选定割顺序也会显著影响混合整数线性规划问题的求解效率。为应对这一挑战，我们提出一种新颖的**分**层序**列**模**型**（HEM），通过强化学习来学习割选择策略。具体而言，HEM由一个双层模型构成：（1）高层模型用于学习应选取的割数量；（2）低层模型将割选择任务构建为序列到序列学习问题，从而学习选择一个有序子集的策略，其大小由高层模型确定。据我们所知，HEM是首个能够从数据驱动视角同时处理割选择中**（P1）-（P3）**的方法。实验表明，与人工设计和基于学习的基线方法相比，HEM显著提高了合成数据及大规模真实世界混合整数线性规划问题（包括 MIPLIB 2017）的求解效率。此外，实验还表明，HEM能够很好地泛化到规模远大于训练数据的问题。

## 1 引言

混合整数线性规划（MILP）是一种通用的优化建模形式，适用于众多重要的现实应用，例如供应链管理（Paschos，2014年）、生产计划（J¨unger 等，2009）、调度（Chen，2010）、设施选址（Farahani 和 Hekmatfar，2009）、装箱问题（Nair 等，2020）等。标准的 MILP 形式如下：

$$z^* \triangleq \min_{\mathbf{x}} \{\mathbf{c}^\top \mathbf{x} | \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^n, x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\}, \tag{1}$$

其中，$\mathbf{c} \in \mathbb{R}^n$、$\mathbf{A} \in \mathbb{R}^{m \times n}$、$\mathbf{b} \in \mathbb{R}^m$，$x_j$表示向量$\mathbf{x}$的第$j$个元素，$\mathcal{I} \subseteq \{1, \ldots, n\}$表示整数变量的索引集合，$z^*$表示问题（1）的最优目标值。然而，混合整数线性规划问题是$\mathcal{NP}$困难问题，因此可能极难求解（Bixby 等，2004年）。为求解混合整数线性规划问题，许多现代混合整数线性规划求解器（Gurobi，2021年；Bestuzheva 等人，2021年；FICO Xpress，2020年）采用分支定界树搜索算法（Land & Doig，2010年），在

*共同贡献。

†通讯作者：jiewangx@ustc.edu.cn

该算法中，每个节点都会求解混合整数线性规划问题（问题（1）或其子问题）的线性规划（LP）松弛。为进一步提升树搜索算法的性能，人们引入了割平面（割）（Gomory，1960年），以收紧线性规划松弛（Achterberg，2007年；Bengio 等人，2021年）。现有关于割的研究分为两类：割生成和割选择（Turner 等人，2022年）。割生成旨在生成割，即能够收紧线性规划松弛的有效线性不等式（Achterberg，2007年）。然而，将所有生成的割都添加到线性规划松弛中可能会带来计算问题（Wesselmann & Stuhl，2012年）。为进一步提高求解混合整数线性规划问题的效率，人们提出了割选择，以从生成的割中选取适当的子集（Wesselmann & Stuhl，2012年）。本文聚焦于**割选择问题**，该问题对整体求解器性能有重要影响（Achterberg，2007年；Tang 等，2020；Paulus 等人，2022年）。

割选择很大程度上取决于（**P1**）应优先选择哪些割，以及（**P2**）应选择多少个割（Achterberg，2007；Dey & Molinaro，2018b）。许多现代混合整数线性规划求解器（Gurobi，2021；Bestuzheva 等人，2021；FICO Xpress，2020）通过专家设计的硬编码启发式方法来处理（**P1**）-（**P2**）。然而，硬编码启发式方法并未考虑从某些类型的现实世界应用中收集的混合整数线性规划问题所蕴含的模式，例如日常生产计划、装箱问题和车辆路径问题（Pochet & Wolsey，2006；Laporte，2009；Nair 等人，2020）。为了进一步提高混合整数线性规划求解器的效率，近期的一些方法（Tang 等人，2020；Paulus 等人，2022；Huang 等人，2022）提出通过机器学习，尤其是强化学习，来学习割选择策略。这些方法为通过捕捉特定应用中混合整数线性规划问题的潜在模式，学习更有效的启发式方法提供了有前景的途径（Bengio 等人，2021）。然而，许多现有的基于学习的方法（Tang 等人，2020；Paulus 等人，2022；Huang 等人，2022）通过学习评分函数来衡量割质量，并选择得分较高的固定比例/数量的割，因此存在两个局限。首先，它们通过学习评分函数来判断应优先选择哪些割，却忽视了学习应选择多少个割的重要性（Dey & Molinaro，2018b）。此外，我们从大量实证结果中观察到，（**P3**）优先采用选定割的何种顺序也会显著影响混合整数线性规划问题的求解效率（见第3节）。其次，这些方法在学习应优先选择哪些割时，没有考虑割之间的交互，因为它们对每个割都进行独立评分。因此，它们难以选出彼此互补良好的割，这可能会严重降低混合整数线性规划问题的求解效率（Dey & Molinaro，2018b）。事实上，我们通过实证表明，这些方法往往会选出许多得分较高但彼此相似的割（见第5节中的实验 4）。

为应对上述挑战，我们提出了一种新颖的**层次序列模型**（HEM），通过强化学习来学习割选择策略。据我们所知，HEM 是首个能够通过提出一种双层模型，同时解决（**P1**）—（**P3**）的基于学习的方法。具体而言，HEM 由以下两部分组成：(1) 用于学习应选择割数量的高层模型；(2) 用于学习选择有序子集的策略的低层模型，其中子集大小由高层模型决定。低层模型将割选择任务表述为序列到序列学习问题，从而带来两项主要优势。首先，序列模型擅长捕捉潜在的顺序信息（Vinyals 等，2016），这对于解决（**P3**）至关重要。其次，序列模型能够很好地捕捉割之间的交互，因为它对给定候选割输入序列时选定割的联合条件概率进行建模。因此，实验表明，在三个合成混合整数线性规划问题和七个具有挑战性的混合整数线性规划问题上，就求解效率而言，HEM 显著优于人工设计的基线方法和基于学习的基线方法。这些具有挑战性的混合整数线性规划问题包括来自 MIPLIB 2017（Gleixner 等，2021）的一些基准测试，以及大规模现实世界生产计划问题。我们的结果表明，在现实世界应用中，利用所提出的 HEM 增强现代混合整数线性规划求解器的能力很强。此外，实验表明，HEM 能够很好地泛化到规模显著大于训练期间所见规模的混合整数线性规划问题。

我们将主要贡献总结如下。(1) 我们从大量实证结果中观察到，选定割的顺序会显著影响求解混合整数线性规划问题的效率（见第 3 节）。(2) 据我们所知，我们提出的 HEM 是首个能够从数据驱动视角同时处理**(P1)-(P3)** 割选择问题的方法。(3) 我们提出将割选择任务表述为序列到序列学习问题，这不仅能够捕获潜在的顺序信息，还能充分捕获割之间的交互，从而选出彼此良好互补的割。(4) 实验表明，在具有挑战性的混合整数线性规划问题上，HEM 相较于竞争性基线方法取得了显著改进，其中包括来自 MIPLIB 2017 的一些基准测试，以及大规模现实世界生产计划问题。

## 2 背景

**割平面**。给定式（1）中的混合整数线性规划问题，我们去除其中的所有整数约束，以获得其线性规划（*LP*）松弛，其形式为

$$z^*_{\text{LP}} \triangleq \min_{\mathbf{x}} \{\mathbf{c}^\top \mathbf{x} | \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^n\}. \tag{2}$$

由于问题 (2) 扩大了问题 (1) 的可行域，因此我们有 $z^*_{\text{LP}} \leq z^*$。我们将通过线性规划松弛得到的任何下界称为对偶界。给定问题 (2) 的线性规划松弛，割平面（割）是添加到线性规划松弛中的线性不等式，旨在收紧松弛，同时不移除问题 (1) 的任何整数可行解。混合整数线性规划求解器会分轮添加生成的割。具体而言，每一轮 $k$ 包括：(i) 求解当前的线性规划松弛；(ii) 生成一组候选割 $\mathcal{C}^k$；(iii) 选择一个子集 $\mathcal{S}^k \subseteq \mathcal{C}^k$；(iv) 将 $\mathcal{S}^k$ 添加到当前的线性规划松弛中，以得到下一轮的线性规划松弛；(v) 进入下一轮。将所有生成的割都添加到线性规划松弛中，可以最大程度地加强该松弛，并在每一轮改善下界。然而，添加过多的割可能会导致模型规模过大，从而增加计算负担并引发数值不稳定性（Wesselmann & Stuhl，2012年；Paulus 等人，2022年）。因此，提出了割选择方法，以从候选割中选出适当的子集，这对提高求解混合整数线性规划问题的效率具有重要意义（Dey & Molinaro，2018b；Tang 等，2020）。

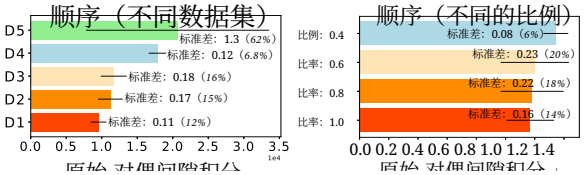
**分支割平面法**。在现代混合整数线性规划求解器中，割平面通常与分支定界算法（Land和Doig，2010年）结合使用，这种算法称为分支割平面算法（Mitchell，2002年）。分支定界技术通过构建搜索树进行隐式枚举，树中的每个节点都代表原问题 (1) 的一个子问题。求解过程从选择树中的一个叶节点并求解其线性规划松弛开始。令 $\mathbf{x}^*$ 为线性规划松弛的最优解。若 $\mathbf{x}^*$ 违反原问题的整数性约束，则通过分支为该叶节点创建两个子问题（子节点）。具体而言，分别为该叶节点添加约束 $x_i \leq \lfloor x^*_i \rfloor$ 和 $x_i \geq \lceil x^*_i \rceil$，其中 $x_i$ 表示第 $i$ 个变量，$x^*_i$ 表示向量 $\mathbf{x}^*$的第 $i$ 个分量，$\lfloor\rfloor$ 和 $\lceil\rceil$ 分别表示向下取整和向上取整函数。相反，如果 $\mathbf{x}^*$ 是 (1) 的（混合）整数解，那么我们就得到 (1) 的最优目标值的一个上界，记为

原始界。在现代混合整数线性规划求解器中，添加割平面与分支阶段交替进行。也就是说，在搜索树节点处先添加割，以收紧其线性规划松弛，然后再进行分支。由于在开始分支之前强化松弛对于确保高效的树搜索至关重要（Wesselmann & Stuhl，2012年；Bengio 等人，2021年），我们仅关注在根节点添加割，这一做法遵循 Gasse 等人（2019年）和 Paulus 等人（2022年）的研究。

**原始-对偶间隙积分**。在运行分支割平面法时，我们会跟踪两个重要界，即全局原始界和全局对偶界，它们分别是式（1）最优目标值的最佳上界和下界。我们将原始-对偶间隙积分（PD 积分）定义为求解器的全局原始界曲线与全局对偶界曲线之间的面积。PD 积分是评估求解器性能的常用指标（Bowly 等人，2021年；Cao 等人，2022年）。更多详情见附录 C.1。

## 3 动机结果

我们通过实验证明，选定割的顺序，即按照这一顺序将选定割添加到线性规划松弛中，会显著影响求解混合整数线性规划问题的效率。此外，我们通过实验证明，求解混合整数线性规划问题时，选定割的比例 也有显著影响（见附录 G.1）。本节所用数据集的详情请见附录 D.2。




(a) 在五个不同的数据集上评估 RandomAll。(b) 在 MIPLIBmixed neos 上评估 RandomNv。

图 1：我们设计了两种割选择启发式方法，即 RandomAll 和 RandomNV（详情见第3节）。对于给定的混合整数线性规划问题，两种方法都会以随机顺序添加相同的部分割。(a) 和 (b) 中的结果表明，以不同顺序添加相同的选定割会导致求解器整体性能出现差异。

**顺序很重要**。先前的研究（Bixby，1992；Maros，2002年；Li 等，2022）表明，给定线性规划（LP）的约束顺序会显著影响这会影响其构造出的初始基，而初始基对于求解线性规划至关重要。由于割是一种线性约束，向线性规划松弛中添加割，等价于向线性规划松弛中添加约束。

因此，添加割的顺序也可能显著影响线性规划松弛的求解，进而对求解混合整数线性规划问题至关重要。事实上，我们的实证结果表明确实如此。(1) 我们设计了一种**RandomAll** 割选择规则：将所有候选割随机打乱顺序，并按随机顺序将所有割添加到线性规划松弛中。我们在五个具有挑战性的数据集 D1 、 D2 、 D3 、 D4 和 D5 上评估 RandomAll。我们使用 SCIP 8.0.0 （ Bestuzheva 等，2021 ）作为后端求解器，并以时间限制内的平均 PD 积分评估求解器性能。我们使用十个随机种子，在每个数据集上评估 RandomAll；图 1a 中的每个柱形表示其在相应数据集上的性能均值和标准差。如图 1a 所示，RandomAll 在每个数据集上的性能会随选定割的顺序而大幅变化。(2) 我们进一步设计了一种**RandomNV** 割选择规则。RandomNV 与 RandomAll 的不同之处在于，它选取给定比例的候选割，而不是选取全部候选割。RandomNV 首先使用归一化违反度（Huang 等，2022）为每个割评分，再选取评分较高的一定比例的割，然后随机打乱选定割的顺序。图 1b 中的每个柱形表示在相同数据集上，RandomNV 在给定比例下的性能均值和标准差。图 1a 和图 1b 表明，以不同顺序添加相同的选定割会导致求解器性能出现差异，这说明选定割的顺序对求解混合整数线性规划问题至关重要。

## 4 基于分层序列模型的割选择学习

在割选择任务中，应该选择的最优子集无法直接获取，但可以使用求解器评估所选子集的质量，并向学习算法提供反馈。因此，我们利用强化学习（RL）来学习割选择策略。本节将详细介绍我们提出的用于学习割选择的 RL 框架。首先，我们将割选择建模为马尔可夫决策过程（MDP）（Sutton 和 Barto，2018年）。然后，我们详细介绍所提出的**分层序列模型**（HEM）。最后，我们推导出一种分层策略梯度，用于高效训练 HEM。

### 强化学习建模

如图 2 所示，我们将混合整数线性规划求解器建模为环境，并将我们提出的 HEM 作为智能体。我们考虑由元组 $(\mathcal{S},\ \mathcal{A},\ r,\ f)$ 定义的 MDP。具体而言，我们将依次说明状态空间 $\mathcal{S}$、动作空间 $\mathcal{A}$、奖励函数 $r:\ \mathcal{S}\times\mathcal{A}\to\mathbb{R}$、转移函数 $f$ 和终止状态。**（1）状态空间** $\mathcal{S}$。由于当前线性规划松弛和生成的割包含割选择的核心信息，我们将状态 $s$ 定义为 $(M_{\text{LP}},\ \mathcal{C},\ \mathbf{x}^*_{\text{LP}})$。其中，$M_{\text{LP}}$ 表示当前线性规划松弛的数学模型，$\mathcal{C}$ 表示候选割集合，而$\mathbf{x}^*_{\text{LP}}$ 表示线性规划松弛的最优解。为了编码状态信息，我们参考 Achterberg（2007年）和 Huang 等人（2022 年），为每个候选割设计了十三个特征

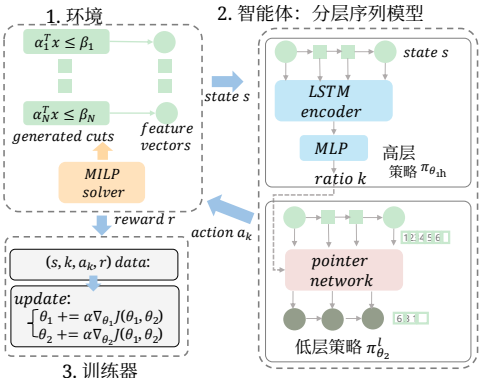



图 2：我们提出的用于学习割选择策略的 RL 框架示意图。我们将混合整数线性规划求解器建模为环境，将 HEM 建模为智能体。此外，我们通过分层策略梯度算法训练 HEM。

基于（$M_{\text{LP}}$、$\mathcal{C}$、$\mathbf{x}^*_{\text{LP}}$）的信息。也就是说，我们实际上用一系列十三维特征向量来表示状态 $s$。我们在附录 F.1 中详细介绍所设计的特征。**（2）动作空间** $\mathcal{A}$。为考虑选定割的比例和顺序，我们将候选割的所有有序子集定义为动作空间 $\mathcal{C}$。由于其组合结构，动作空间的基数可能极大，因此高效探索动作空间颇具挑战。**（3）奖励函数** $r$。为了评估添加的割对求解混合整数线性规划问题的影响，我们设计了奖励函数，其依据包括：（i）在线性规划松弛求解结束时收集的指标，例如对偶界改进；（ii）运行结束统计数据，例如求解时间和原始-对偶间隙积分。对于第一种情况，每一步的奖励 $r$（$s$、$a$）可定义为对偶界改进的相反数。对于第二种情况，除轨迹中的最后一步（$s_T$、$a_T$）外，奖励 $r$（$s$、$a$）均可定义为零；也就是说，$r$（$s_T$、$a_T$）定义为求解时间或原始-对偶间隙积分的相反数。**（4）状态转移函数** $f$。状态转移函数将当前状态 $s$ 和动作 $a$ 映射到下一状态 $s'$，其中 $s'$ 表示在当前线性规划松弛中添加选定割后生成的下一个线性规划松弛。**（5）终止状态**。有

目前没有确定何时终止割平面分离过程的标准统一准则（Paulus 等人，2022年）。假设我们将割分离轮数设为 $T$，那么求解器环境会在 $T$ 轮后终止割平面分离。在多轮设置下（即 $T > 1$），我们将割选择建模为马尔可夫决策过程。在单轮设置下（即 $T = 1$），该建模可简化为上下文老虎机。

**分层序列模型**

**动机**。令 $\pi$ 表示割选择策略 $\pi$： $\mathcal{S} \to \mathcal{P}(\mathcal{A})$，其中 $\mathcal{P}(\mathcal{A})$ 表示动作空间上的概率分布，而 $\pi(\cdot|s)$ 表示给定状态 $s$时动作空间上的概率分布。我们强调，学习此类策略可以同时解决割选择中的 **(P1)-(P3)** 问题。然而，直接学习此类策略面临以下挑战。首先，由于动作空间具有组合结构，其基数可能极大，因此难以有效探索动作空间。其次，不同混合整数线性规划问题中的动作长度及其最大长度（即有序子集）各不相同。然而，传统强化学习通常处理动作长度固定的问题。许多现有的基于学习的方法（Tang 等，2020；Huang 等，2022；Paulus 等人，2022）并不直接学习上述策略，而是学习一个评分函数，为每个割输出一个分数，并选择分数较高且比例/数量固定的割。然而，正如第 1 节所述，这些方法存在两项局限。

策略网络架构。为解决上述问题，我们提出了一种新颖的层次序列模型（HEM），用于学习割选择策略。为促进高效探索，HEM 利用割选择任务的层次结构，将策略分解为两个子策略，即高层策略 $\pi^h$ 和低层策略 $\pi^l$。图 2 还展示了 HEM 的策略网络架构。首先，高层策略通过预测一个合适的比例，学习应选择的割的数量。假设状态的长度为 $N$，预测的比例为 $k$，那么应选择的割的预测数量为 $\lfloor N * k \rfloor$，其中 $\lfloor \cdot \rfloor$ 表示向下取整函数。我们将高层策略定义为 $\pi^h$： $\mathcal{S} \to \mathcal{P}([0,1])$，其中 $\pi^h(\cdot|s)$表示给定状态 $s$时， $[0,1]$ 上的概率分布。其次，低层策略学习选择一个有序子集，其大小由高层策略确定，为[1] 。我们将低层策略定义为 $\pi^l$: $\mathcal{S} \times [0,1] \to \mathcal{P}(\mathcal{A})$，其中 $\pi^l(\cdot|s,k)$ 表示给定状态 $s$ 和比例 $k$ 时，动作空间上的概率分布。具体而言，我们将低层策略建模为序列模型，以捕捉割之间的交互。最后，我们利用全概率公式推导出割选择策略，即 $\pi(a_k|s) = \mathbb{E}_{k\sim\pi^h(\cdot|s)}[\pi^l(a_k|s,k)]$, ，其中 $k$表示给定的比例，$a_k$ 表示动作。该策略通过取期望计算，因为 $a_k$无法确定比例 $k$。例如，假设 $N = 100$ 且 $a_k$ 的长度为 10，那么比例 $k$ 可以是区间 $[0.1, 0.11)$ 内的任意数值。实际上，我们先从 $\pi^h$ 中采样一个比例 $k$，再根据该比例从 $\pi^l$ 中采样一个动作，以此从策略 $\pi$ 中采样动作。

对于高层策略，我们首先将高层策略建模为双曲正切高斯分布，即带有可逆压缩函数（tanh）的高斯分布，这种分布在深度强化学习中得到了广泛应用（Schulman 等人，2017年；Haarnoja 等人，2018年）。高斯分布的均值和方差由神经网络给出。双曲正切高斯分布的支撑度为 $[-1,1]$，但选定割的比例应属于 $[0,1]$。因此，我们进一步对双曲正切高斯分布执行线性变换。具体而言，我们通过 $\pi^h_{\theta_1}(\cdot|s) = 0.5 * \tanh(K) + 0.5$ 定义参数化的高层策略，其中 $K \sim \mathcal{N}(\mu_{\theta_1}(s)\ \sigma_{\theta_1}(s))$ 。由于不同实例（混合整数线性规划问题）中的状态序列长度各不相同，我们使用长短期记忆网络（LSTM）（Hochreiter & Schmidhuber，1997年）对候选割序列进行嵌入。随后，我们使用多层感知机（MLP）（Goodfellow 等人，2016年）根据 LSTM 的最后一个隐藏状态预测均值和方差。

对于低层策略，我们将其建模为序列模型。也就是说，其输入是候选割序列，输出则是候选割有序子集的概率分布，其大小由高层策略决定。具体而言，给定一个状态—动作对（$s$， $k$， $a_k$），序列模型使用参数化模型估计概率链式法则中的各项，从而计算条件概率 $\pi^l_{\theta_2}(a_k|s,\ k)$，即 $\pi^l_{\theta_2}(a_k|s,\ k) = \prod_{i=1}^{m} p_{\theta_2}(a^i_k|a^1_k,\ \ldots,\ a^{i-1}_k,\ s,\ k)$。其中， $s = \{s^1, \ldots, s^N\}$ 是输入序列， $m = \lfloor N * k \rfloor$是输出序列的长度， $a_k = \{a^1_k, \ldots, a^m_k\}$ 是由 $m$ 个索引组成的序列，每个索引对应输入序列中的一个位置 $s_o$。此类策略可以通过机器翻译中常用的原始序列模型进行参数化（Sutskever 等人，2014年；Vaswani 等人，2017年）。

[1]我们交替使用以下术语：选定子集的大小以及选定割的数量/比例 .

然而，原始序列模型只能用于单个实例上的学习，因为不同实例中的候选割数量各不相同。为了在不同实例之间进行泛化，我们使用指针网络（Vinyals 等人，2015年；Bello* 等人，2017年）对 $\pi^l_{\theta_2}$ 进行参数化（详情参见附录F.4.1）。指针网络使用注意力机制作为指针，在每个解码器步骤中选择输入序列的一个成员作为输出。据我们所知，我们是首个将割选择任务建模为序列到序列学习问题，并将指针网络应用于割选择的研究。这带来了两项主要优势：（1）捕捉底层的顺序信息；（2）捕捉割之间的交互。附录E中的示例也对此进行了说明。

**训练：分层策略梯度**

For 在割选择任务中，我们旨在找到 $\theta$，使其最大化所有轨迹上的期望奖励 s

$$J(\theta) = \mathbb{E}_{s\sim\mu, a_k\sim\pi_\theta(\cdot|s)}[r(s,a_k)], \tag{3}$$

其中，$\theta = [\theta_1,\theta_2]$，$\mathbb{E}_{k\sim\pi^h_{\theta_1}(\cdot|s)}[\pi^l_{\theta_2}(a_k|s,k)]$，$[\theta_1,\theta_2]$ 表示两个向量的拼接，$\mu$ $\pi_\theta$ = $(a_k|s)$ $()(,)$ 表示初始状态分布。

为了训练具有层次结构的策略，我们遵循广为人知的策略梯度定理（Sutton 等人，1999a；Sutton 和 Barto，2018）推导出分层策略梯度。

**命题 1**。 给定割选择策略 $\pi_\theta(a_k|s) = \mathbb{E}_{k\sim\pi^h_{\theta_1}(\cdot|s)}[\pi^l_{\theta_2}(a_k|s,k)]$ 以及训练目标（*3*），分层策略梯度的形式为

$$\nabla_{\theta_1} J([\theta_1,\theta_2]) = \mathbb{E}_{s\sim\mu,k\sim\pi^h_{\theta_1}(\cdot|s)}[\nabla_{\theta_1}\log(\pi^h_{\theta_1}(k|s))\mathbb{E}_{a_k\sim\pi^l_{\theta 2}(\cdot|s,k)}[r(s,a_k)]],$$
$$\nabla_{\theta_2} J([\theta_1,\theta_2]) = \mathbb{E}_{s\sim\mu,k\sim\pi^h_{\theta_1}(\cdot|s),a_k\sim\pi^l_{\theta_2}(\cdot|s,k)}[\nabla_{\theta_2}\log\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)].$$

详细证明见附录 A。我们使用推导得到的分层策略梯度来更新高层和低层策略的参数。我们以并行方式实现训练算法，该算法与异步优势演员-评论家（A3C）（Mnih 等人，2016）密切相关。由于篇幅有限，我们在附录 F.3.6 中总结了训练算法的流程。此外，我们还讨论了 HEM 的其他一些优势（详情见附录 F.4.3）。(1) HEM 利用了割选择任务的层次结构，这对于在复杂决策任务中进行高效探索十分重要（Sutton 等人，1999b）。(2) 我们通过基于梯度的算法训练 HEM，因而具有较高的样本效率（Sutton 和 Barto，2018年）。

5 实验

我们的实验主要分为五个部分：**实验1**。 在三个经典混合整数线性规划问题和六个来自不同应用领域、具有挑战性的混合整数线性规划问题基准测试上评估我们的方法。**实验 2**。 开展精心设计的消融研究，以进一步深入了解 HEM。**实验 3**。 测试 HEM 能否泛化到显著大于训练期间所见规模的实例。 **实验 4**。 将我们的方法选择的割与基线方法进行可视化比较。**实验 5**。 将我们的方法应用于现实世界的生产计划问题。

**基准测试**。 我们在九个困难的混合整数线性规划问题基准测试上评估我们的方法，其中包括三个经典的合成混合整数线性规划问题和六个来自不同应用领域的具有挑战性的混合整数线性规划问题。我们根据使用 SCIP 8.0.0 求解器（Bestuzheva 等，2021）求解这些问题的难度，将九个问题基准测试分为三类，分别称为简单、中等难度和困难数据集。(1) 简单数据集 包括三个广泛使用的合成混合整数线性规划问题基准测试：集合覆盖（Balas 和 Ho，1980）、最大独立集（Bergman 等人，2016）和多重背包（Scavuzzo 等人，2022）。我们按照 Gasse 等（2019）和 Sun 等（2020）的方法人工生成实例。(2)中等难度数据集 包括 MIK （ Atamt¨urk，2003 ）和 CORLAT （ Gomes 等人，2008 ），它们是广泛用于评估混合整数线性规划求解器的基准测试（He 等人，2014；Nair 等，2020）。(3) 困难数据集 包括负载均衡问题（灵感来自大规模系统的现实应用）和 Anonymous 问题（灵感来自一项大规模工业应用）（Bowly等人，2021年）。此外，困难数据集还包含来自 MIPLIB 2017（MIPLIB）的基准测试（Gleixner 等，2021）。尽管 Turner 等，2022 已表明，直接在完整的 MIPLIB 上进行学习极具挑战性，但我们提出在 MIPLIB 的子集上进行学习。我们构建了两个子集，分别称为 MIPLIB mixed neos 和 MIPLIB mixed supportcase。由于篇幅有限，关于这些数据集的详情请参见附录 D.1。

表1：在简单、中等和困难数据集上的策略评估。最佳性能以粗体标出。令 $m$ 表示约束的平均数量， $n$ 表示变量的平均数量。我们报告时间和 PD 积分的算术平均值（标准差）。

| 方法 | 简单：集合覆盖（$n = 1000,\ m = 500$）时间（秒） ↓ | 改进（时间，%） ↑ | PD 积分↓ | 简单：最大独立集（$n = 500,\ m = 1953$）时间（秒） ↓ | 改进（时间，%） ↑ | PD 积分↓ | 简单：多重背包（$n = 720,\ m = 72$）时间（秒） ↓ | 改进（时间，%） ↑ | PD 积分 ↓ |
|---|---|---|---|---|---|---|---|---|---|
| NoCuts | 6.31（4.61） | NA | 56.99（38.89） | 8.78 (6.66) | NA | 71.31 (51.74) | 9.88 (22.24) | NA | 16.41 (14.16) |
| 默认 | 4.41 (5.12) | 29.90 | 55.63 (42.21) | 3.88 (5.04) | 55.80 | 29.44 (35.27) | 9.90 (22.24) | -0.20 | 16.46 (14.25) |
| 随机 | 5.74 (5.19) | 8.90 | 67.08 (46.58) | 6.50 (7.09) | 26.00 | 52.46 (53.10) | 13.10 (35.51) | -32.60 | 20.00 (25.14) |
| NV | 9.86 (5.43) | -56.50 | 99.77 (53.12) | 7.84 (5.54) | 10.70 | 61.60 (43.95) | 13.04 (36.91) | -32.00 | 21.75 (24.71) |
| Eff | 9.65 (5.45) | -53.20 | 95.66 (51.71) | 7.80 (5.11) | 11.10 | 61.04 (41.88) | 9.99 (19.02) | -1.10 | 20.49 (22.11) |
| SBP | 1.91 (0.36) | 69.60 | 38.96 (8.66) | 2.43 (5.55) | 72.30 | 21.99 (40.86) | 7.74 (12.36) | 21.60 | 16.45 (16.62) |
| HEM（本文方法） | **1.85（0.31）** | **70.60** | **37.92（8.46）** | **1.76（3.69）** | **80.00** | **16.01（26.21）** | **6.13（9.61）** | **38.00** | **13.63 (9.63)** |

| 方法 | 中等：MIK ($n = 413,\ m = 346$) 时间（秒） ↓ | PD 积分↓ | 改进 ↑（PD积分，%） | 中等：CORLAT ($n = 466,\ m = 486$) 时间（秒） ↓ | PD 积分↓ | 改进 ↑（PD积分，%） | 困难：负载均衡 ($n = 61000,\ m = 64304$) 时间（秒） ↓ | PD 积分↓ | 改进 ↑（PD积分，%） |
|---|---|---|---|---|---|---|---|---|---|
| NoCuts | 300.01 (0.009) | 2355.87 (996.08) | NA | 103.30 (128.14) | 2818.40 (5908.31) | NA | 300.00 (0.12) | 14853.77 (951.42) | NA |
| 默认 | 179.62 (122.36) | 844.40 (924.30) | 64.10 | 75.20 (120.30) | 2412.09 (5892.88) | 14.40 | 300.00 (0.06) | 9589.19 (1012.95) | 35.40 |
| 随机 | 289.86 (28.90) | 2036.80 (933.17) | 13.50 | 84.18 (124.34) | 2501.98 (6031.43) | 11.20 | 300.00 (0.09) | 13621.20 (1162.02) | 8.30 |
| NV | 299.76 (1.32) | 2542.67 ( 529.49) | -7.90 | 90.26 (128.33) | 3075.70 (7029.55) | -9.10 | 300.00 (0.05) | 13933.88 (971.10) | 6.20 |
| Eff | 298.48 (5.84) | 2416.57 (642.41) | -2.60 | 104.38 (131.61) | 3155.03 (7039.99) | -11.90 | 300.00 (0.07) | 13913.07 (969.95) | 6.30 |
| SBP | 286.07 (41.81) | 2053.30 (740.11) | 12.80 | 70.41 (122.17) | 2023.87 (5085.96) | 28.20 | 300.00 (0.10) | 12535.30 (741.43) | 15.60 |
| HEM（本文方法） | **176.12 (125.18)** | **785.04 (790.38)** | **66.70** | **58.31 (110.51)** | **1079.99 (2653.14)** | **61.68** | **300.00 (0.04)** | **9496.42 (1018.35)** | **36.10** |

| 方法 | 困难：匿名 ($n = 37881,\ m = 49603$) 时间（秒） ↓ | PD 积分↓ | 改进 ↑（PD积分，%） | 困难：MIPLIB 混合 neos ($n = 6958,\ m = 5660$) 时间（秒） ↓ | PD 积分↓ | 改进 ↑（PD积分，%） | 困难：MIPLIB 混合支持案例 ($n = 19766,\ m = 19910$) 时间（秒） ↓ | PD 积分↓ | 改进 ↑（PD积分，%） |
|---|---|---|---|---|---|---|---|---|---|
| NoCuts | 246.22 (94.90) | 18297.30 (9769.42) | NA | 253.65 (80.29) | 14652.29 (12523.37) | NA | 170.00 (131.60) | 9927.96 (11334.07) | NA |
| 默认 | 244.02 (97.72) | 17407.01 (9736.19) | 4.90 | 256.58 (76.05) | 14444.05 (12347.09) | 1.42 | 164.61 (135.82) | 9672.34 (10668.24) | 2.57 |
| 随机 | 243.49 (98.21) | 16850.89 (10227.87) | 7.80 | 255.88 (76.65) | 14006.48 (12698.76) | 4.41 | 165.88 (134.40) | 10034.70 (11052.73) | -1.07 |
| NV | 242.01 (98.68) | 16873.66 (9711.16) | 7.80 | 263.81 (64.10) | 14379.05 (12306.35) | 1.86 | **161.67 (131.43)** | 8967.00 (9690.30) | 9.68 |
| Eff | 244.94 (93.47) | 17137.87 (9456.34) | 6.30 | 260.53 (68.54) | 14021.74 (12859.41) | 4.30 | 167.35 (134.99) | 9941.55 (10943.48) | -0.14 |
| SBP | 245.71 (92.46) | 18188.63 (9651.85) | 0.59 | 256.48 (78.59) | 13531.00 (12898.22) | 7.65 | 165.61 (135.25) | 7408.65 (7903.47) | 25.37 |
| HEM（本文方法） | **241.68 (97.23)** | **16077.15 (9108.21)** | **12.10** | **248.66 (89.46)** | **8678.76 (12337.00)** | **40.77** | 162.96 (138.21) | **6874.80 (6729.97)** | **30.75** |

**实验设置**。 在所有实验中，我们均使用 SCIP 8.0.0（Bestuzheva 等，2021）作为后端求解器。它是最先进的开源求解器，广泛应用于组合优化机器学习研究（Gasse 等，2019；Huang 等，2022；Turner 等，2022；Nair 等，2020）。按照 Gasse 等（2019）、Huang 等（2022）和 Paulus 等人（2022）的做法，我们仅允许在根节点生成和选择割平面，并将割分离轮数设为一。为尽可能确保比较公平且可复现，我们将其他所有 SCIP 参数保留为默认值。需要强调的是，SCIP 求解器的所有高级功能（如预求解和启发式方法）均保持启用，以确保我们的设置符合实际应用场景。在所有实验中，我们将求解时间限制设为 300秒。为完整起见，我们还在更长的三小时求解时间限制下评估了 HEM。结果见附录 G.6。我们使用 PyTorch（Paszke 等，2019）和 ADAM（Kingma 和 Ba，2014）训练 HEM。此外，我们还提供了另一种基于 MindSpore（Chen，2021）的实现。为简单起见，我们将每个数据集划分为训练集和测试集，分别包含 80% 和 20% 的实例。若要进一步提升 HEM，可以构建有效的超参数调优集。我们在训练集上训练模型，并从中选出最佳模型，再在测试集上进行评估。实现细节、超参数和硬件规格请参见附录 F.3。

**基线方法**。 我们的基线方法包括五种广泛使用的人工设计的割选择规则，以及一种最先进的基于学习的方法。割选择规则包括 NoCuts、随机、归一化违反度（NV）、效能（Eff）和默认。NoCuts 不添加任何割。默认表示 SCIP 8.0.0 中使用的默认割选择规则。对于基于学习的方法，我们实现了最先进的基于学习的方法（Tang 等，2020；Huang 等，2022）的一种轻微变体，即基于评分的策略（SBP）。有关这些基线方法的实现细节，请参见附录 F.2。

**评估指标**。我们采用两种广泛使用的评估指标，即平均求解时间（时间，越低越好）和平均原始-对偶间隙积分（PD 积分，越低越好）。此外，我们还在附录 G.2 中提供了另外两项指标的更多结果，即平均节点数和平均原始-对偶间隙。为了评估不同割选择方法相较于不进行割平面分离的纯分支定界法的表现，我们提出了一项改进指标。具体而言，该指标定义为 $\mathrm{Im}_M(\cdot) = {}^{M_{(\mathrm{NoCuts})-M(\cdot)}}\, M(\mathrm{NoCuts})$，其中，$M$(NoCuts) 表示 NoCuts 的表现，而 $M(\cdot)$ 表示从方法到其表现的映射。改进指标表示给定方法相较于 NoCuts 的改进。我们主要关注简单数据集上的时间指标，因为求解器能够在给定时间限制内将所有实例求解至最优。然而，在中等和困难数据集上，HEM 和基线方法无法在时间限制内将所有实例求解至最优。因此，这些未求解实例的平均求解时间相同，难以使用时间指标区分不同割选择方法的表现。因此，我们

表2：HEM 与不使用高层模型的 HEM 的比较。

| | 简单：最大独立集 ($n = 500,\ \ m = 1953$) | | | 中等：CORLAT ($n = 466,\ \ m = 486$) | | | 困难：MIPLIB 混合 neos ($n = 6958,\ \ m = 5660$) | | |
|---|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒）↓ | 改进 ↑ (时间，%) | PD 积分↓ | 时间（秒）↓ | PD 积分 ↓ | 改进 ↑ （PD积分，%） | 时间（秒）↓ | PD 积分↓ | 改进 ↑ （PD积分，%） |
| NoCuts | 8.78 (6.66) | NA | 71.31 (51.74) | 103.30 (128.14) | 2818.40 (5908.31) | NA | 253.65 (80.29) | 14652.29 (12523.37) | NA |
| 默认 | 3.88 (5.04) | 55.81 | 29.44 (35.27) | 75.20 (120.30) | 2412.09 (5892.88) | 14.42 | 256.58 (76.05) | 14444.05 (12347.09) | 1.42 |
| SBP | 2.43 (5.55) | 72.32 | 21.99 (40.86) | 70.41 (122.17) | 2023.87 (5085.96) | 28.19 | 256.48 (78.59) | 13531.00 (12898.22) | 7.65 |
| HEM w/o H | 1.88 (4.20) | 78.59 | 16.70 (28.15) | 63.14 (115.26) | 1939.08 (5484.83) | 31.20 | 249.21 (88.09) | 13614.29 (12914.76) | 7.08 |
| HEM（本文方法） | **1.76 (3.69)** | **79.95** | **16.01 (26.21)** | **58.31 (110.51)** | **1079.99 (2653.14)** | **61.68** | **248.66 (89.46)** | **8678.76 (12337.00)** | **40.77** |

表 3：HEM、HEM-ratio 和 HEM-ratio-order 之间的比较。

| | 简单：最大独立集 ($n = 500,\ m = 1953$) | | | 中等：CORLAT ($n = 466,\ m = 486$) | | | 困难：MIPLIB mixed neos ($n = 6958,\ m = 5660$) | | |
|---|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒）↓ | 改进 ↑ (时间, %) | PD 积分 ↓ | 时间（秒）↓ | PD 积分 ↓ | 改进 ↑ （PD积分，%） | 时间（秒）↓ | PD 积分 ↓ | 改进 ↑ （PD积分，%） |
| NoCuts | 8.78 (6.66) | NA | 71.31 (51.74) | 103.30 (128.14) | 2818.40 (5908.31) | NA | 253.65 (80.29) | 14652.29 (12523.37) | NA |
| 默认 | 3.88 (5.04) | 55.81 | 29.44 (35.27) | 75.20 (120.30) | 2412.09 (5892.88) | 14.42 | 256.58 (76.05) | 14444.05 (12347.09) | 1.42 |
| SBP | 2.43 (5.55) | 72.32 | 21.99 (40.86) | 70.41 (122.17) | 2023.87 (5085.96) | 28.19 | 256.48 (78.59) | 13531.00 (12898.22) | 7.65 |
| HEM-ratio-order | 2.30 (5.18) | 73.80 | 21.19 (38.52) | 70.94 (122.93) | 1416.66 (3380.10) | 49.74 | 245.99 (93.67) | 14026.75 (12683.60) | 4.27 |
| HEM-ratio | 2.26 (5.06) | 74.26 | 20.82 (37.81) | 67.27 (117.01) | 1251.60 (2869.87) | 55.59 | **244.87 (95.56)** | 13659.93 (12900.59) | 6.77 |
| HEM（本文方法） | **1.76 (3.69)** | **79.95** | **16.01 (26.21)** | **58.31 (110.51)** | **1079.99 (2653.14)** | **61.68** | 248.66 (89.46) | **8678.76 (12337.00)** | **40.77** |

主要关注中等和困难数据集上的 *PD* 积分指标。PD 积分也是广受认可的求解器性能评估指标（Bowly 等，2021年；Cao 等，2022年）。

**实验1. 比较评估** 表1中的结果表明如下几点。(1) **简单数据集**。 在简单数据集上，HEM 的表现显著优于所有基线方法，尤其是在最大独立集和多重背包问题上。SBP 的表现远胜于所有基于规则的基线方法，说明我们实现的 SBP 是一种强有力的基线方法。与 SBP 相比，HEM 在三个数据集上的时间最多缩短了 16.4%，体现了我们的方法优于最先进的基于学习的方法。(2) **中等难度数据集**。 在 MIK 和 CORLAT 上，HEM 仍优于所有基线方法。尤其是在 CORLAT 上，与基线方法相比，HEM 在 PD 积分方面至少提升了 33.48%。(3) **困难数据集**。 在困难数据集中的若干问题上，HEM 在 PD 积分方面显著优于基线方法。HEM 在来自 MIPLIB 2017 和现实世界问题（负载均衡和匿名）的两个具有挑战性的数据集上表现出色，证明了在大规模现实世界应用中，HEM 能够显著增强混合整数线性规划求解器的能力。此外，SBP 在若干中等和困难数据集上的表现极差，这意味着在具有挑战性的混合整数线性规划问题上，学习良好的割选择策略可能颇具难度。

**实验2。消融研究** 我们展示了最大独立集（MIS）、CORLAT 和 MIPLIB mixed neos 上的消融研究。这些数据集分别代表简单、中等和困难数据集。关于其他数据集的更多结果，请参见附录G.3。

**各组件的贡献**。 我们进行消融研究，以了解 HEM 中各组件的贡献。我们在表2中报告了 HEM 和不含高层模型的 HEM（HEM w/o H）的性能。HEM w/o H 本质上是一个指针网络。请注意，它仍可通过预测一个结束标记，隐式预测应选择的割的数量，这种做法与语言任务中使用的方法相同（Sutskever等人，2014年）。详情请参见附录 F.4.2。 **首先**，表2中的结果显示，HEM w/o H 在 MIS 和 CORLAT 上优于所有基线方法，证明了低层模型的优势。尽管 HEM w/o H 在 MIPLIB mixed neos 上优于默认方法，但其表现与 SBP 相当。一个可能的原因是，HEM w/o H 难以高效探索动作空间，因此容易陷入局部最优。 **其次**，表2中的结果显示，HEM 在这三个数据集上显著优于 HEM w/o H 和各基线方法。这些结果表明，高层模型对于复杂任务中的高效探索至关重要，因此能够显著提升求解效率。

**解决 (P1)-(P3) 的重要性**。我们通过消融研究来理解在割选择中解决 **(P1)-(P3)** 的重要性。(1) **HEM**。HEM 在割选择中同时解决 **(P1)-(P3)** 。(2) **HEM-ratio**。为了不学习应选择多少个割，我们移除 HEM 的高层模型，并强制低层模型选择固定比例的割。我们将其称为 HEM-ratio。注意，HEM-ratio 与 HEM w/o H 不同（见附录 F.4.2）。HEM-ratio 在割选择中解决 **(P1)** 和 **(P3)** 。(3) **HEM-ratio-order**。为进一步减弱选定割的顺序所产生的影响，我们根据生成割的原始索引，对 HEM-ratio 给出的选定割重新排序，并将其称为 HEM-ratio-order。HEM-ratio-order 主要解决割选择中的**(P1)** 。表 3 中的结果表明：HEM-ratio-order 显著优于默认方法和 NoCuts，这说明通过数据驱动方法解决 **(P1)** 至关重要。就 PD 积分而言，HEM 显著优于 HEM-ratio，这说明解决 **(P2)**具有重要意义。就时间和 PD 积分而言，HEM-ratio 优于 HEM-ratio-order，

表 4：**左侧**：HEM 的泛化能力。**右侧**：在生产计划问题上的测试。

| 方法 | 最大独立集($n=1000,\ m=3946,\ 4\times$) 时间（秒）↓ | 改进↑（时间，%） | PDintegral↓ | 最大独立集($n=5040,\ m=1500,\ 9\times$) 时间（秒）↓ | 改进↑（时间，%） | PDintegral↓ |
|---|---|---|---|---|---|---|
| NoCuts | 170.06(100.61) | NA | 874.45(522.29) | 300.00(0) | NA | 2019.93(353.27) |
| 默认 | 42.40(76.00) | 48.72 | 198.61(331.20) | 111.18(144.13) | 60.91 | 616.46(798.94) |
| 随机 | 118.25(109.05) | -43.00 | 574.33(516.11) | 245.13(115.80) | 13.82 | 1562.20(793.09) |
| NV | 160.30(101.41) | -93.86 | 784.98(493.24) | 299.97(0.49) | -5.46 | 1922.52(349.67) |
| Eff | 158.75(100.40) | -91.98 | 779.63(493.05) | 299.45(3.77) | -5.28 | 1921.61(361.26) |
| SBP | 50.55(89.14) | 38.87 | 253.81(426.94) | 108.42(143.68) | 61.88 | 680.41(903.88) |
| HEM（本文方法） | **35.34(67.91)** | **57.26** | **160.56(282.03)** | **108.02(143.02)** | **62.02** | **570.48(760.65)** |

| 方法 | 生产计划 ($n=3582.25,\ m=5040.42$) 时间（秒）↓ | 改进↑（时间，%） | PD 积分↓ | 改进↑（PD积分，%） |
|---|---|---|---|---|
| NoCuts | 278.79 (231.02) | NA | 17866.01 (21309.85) | NA |
| 默认 | 296.12 (246.25) | -6.22 | 17703.39 (21330.40) | 0.91 |
| 随机 | 280.18 (237.09) | -0.50 | 18120.21 (21660.01) | -1.42 |
| NV | 259.48 (227.81) | 6.93 | 17295.18 (21860.07) | 3.20 |
| Eff | 263.60 (229.24) | 5.45 | 16636.52 (21322.89) | 6.88 |
| SBP | 276.61 (235.84) | 0.78 | 16952.85 (21386.07) | 5.11 |
| HEM（本文方法） | **241.77 (229.97)** | **13.28** | **15751.08 (20683.53)** | **11.84** |

这说明解决 **(P3)**的重要性。此外，在 MIS 和 CORLAT 上，HEM-ratio 和 HEM-ratio-order 的表现优于 SBP，这表明使用序列模型学习割选择优于使用 SBP。在 MIPLIB mixed neos 上，HEM-ratio 和 HEM-ratio-order 的表现与 SBP 相当。可能的原因见附录 G.3.1。

**实验 3. 泛化** 我们评估 HEM 在不同规模的混合整数线性规划问题之间进行泛化的能力。令 $n \times m$ 表示混合整数线性规划实例的规模。遵循 Gasse 等人（2019）；Sun 等人（2020）的做法，我们在集合覆盖和最大独立集（MIS）问题上测试 HEM 的泛化能力，因为我们可以为合成混合整数线性规划问题人工生成任意规模的实例。在 MIS 问题上，我们使用比训练期间见过的实例大四倍和九倍的实例测试 HEM。表 4（**左侧**）中的结果表明，在 $4\times$ 和 $9\times$ MIS 上，HEM 在时间和 PD 积分方面显著优于基线方法，证明了 HEM 在

泛化能力。有趣的是，SBP 对大型实例也具有良好的泛化能力，这表明 SBP 是一个强有力的基线。关于集合覆盖的更多结果见附录 G.4。

**实验 4：选定割的可视化**我们将选定割的多样性可视化。这是一项重要指标，可用于评估这些选定割是否能很好地互为补充（Dey 和 Molinaro, 2018b）。我们分别在从最大独立集和 CORLAT 中随机抽取的实例上，对 HEM-ratio 和 SBP 选定的割进行可视化。我们评估 HEM-ratio 而非 HEM，因为 HEM-ratio 选出的

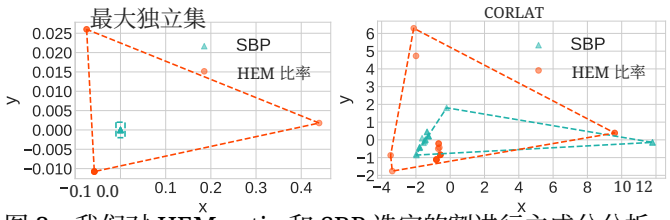



图 3：我们对 HEM-ratio 和 SBP 选定的割进行主成分分析。彩色点表示降维后的割特征。虚线所覆盖的区域代表选定割的多样性。结果表明，与 SBP 相比，HEM-ratio 选出的割多样性高得多。

割数量与 SBP 相同。此外，我们对选定割进行主成分分析，将割特征降至二维空间。彩色点表示降维后的割特征。为了展示选定割的多样性，我们用虚线连接 x、y 坐标最小和最大的点。也就是说，虚线所覆盖的区域代表多样性。图 3 显示，SBP 往往会选出许多相似、可能存在冗余的割，尤其是在最大独立集上。相比之下，HEM-ratio 选出的割多样性高得多，能够很好地互为补充。更多数据集上的结果请参见附录 G.5。

**实验 5：在现实世界生产计划问题中的部署**为进一步评估我们提出的 HEM 的有效性，我们将 HEM 部署于[2] 一家匿名企业的大规模现实世界生产计划问题中。该企业是全球规模最大的商业科技企业之一。有关这些问题的更多详情，请参阅附录 D.3。表 4（**右**）中的结果显示，在时间和 PD 积分方面，HEM 显著优于所有基线方法。这些结果表明，我们提出的 HEM 能够在现实世界应用中显著增强现代混合整数线性规划求解器的能力。有趣的是，默认方法的表现不如 NoCuts，这意味着不恰当的割选择策略可能会显著降低混合整数线性规划求解器的性能。

## 6 结论

在本文中，我们从大量实证结果中观察到，选定割的顺序会显著影响求解混合整数线性规划问题的效率。我们提出了一种新颖的**分**层式序**列**模**型**（HEM），通过强化学习来学习割选择策略。具体而言，HEM由一个双层模型构成：（1）高层模型用于学习应选取的割的数量；（2）低层模型将割选择任务建模为序列到序列学习问题，

[2]论文录用发表后，我们将公开数据集和代码。

并学习选择一个有序子集的策略，其大小由高层模型确定。实验表明，与人工设计的基线方法和基于学习的基线方法相比，HEM在合成数据和大规模真实世界混合整数线性规划问题上均显著提高了求解效率。我们相信，所提出的方法为学习割选择带来了新的见解。

## 7 伦理声明

由于我们的工作属于求解混合整数线性规划问题（MILP）的基础研究，因此不存在直接的伦理风险。采用专家设计的割选择启发式方法的现代混合整数线性规划求解器旨在求解一般的混合整数线性规划问题，而采用我们提出的 HEM 的现代混合整数线性规划求解器，则可以从某些类型的现实世界应用中收集的混合整数线性规划问题中，学习更有效的割选择启发式方法。

## 8 可复现性声明

为确保可复现性，我们在第 4 节介绍所提出的 HEM，并在附录F中提供更多实现细节。此外，我们在附录 F.3.6 中提供所提出的 HEM 的伪代码。论文被接收后，我们将公开发布所有源代码。

除实验中使用的现实世界生产计划问题数据集外，所有数据集均已公开。有关这些数据集的详细信息，请参阅附录 D。论文被接收后，我们也将公开发布来自现实世界生产计划问题的数据集。

## 参考文献

Tobias Achterberg。约束整数规划。2007年。

Alper Atamt¨urk。关于混合整数背包多面体的面。数学规划 -
*m*学，98(1)：145–175，2003年。

Egon Balas 和 Andrew Ho。使用割平面、启发式方法和次梯度优化的集合覆盖算法：一项计算研究。载于组合优化，第37–60页。Springer，1980年。

Maria-Florina Balcan、Travis Dick、Tuomas Sandholm 和 Ellen Vitercik。学习分支。载于 Jennifer Dy 和 Andreas Krause（编），第*35*届国际机器学习会议论文集，《机器学习研究论文集》第80卷，第344–353页。PMLR，2018年7月10–15日。

Maria-Florina F Balcan、Siddharth Prasad、Tuomas Sandholm 和 Ellen Vitercik。《树搜索配置的样本复杂度：割平面及其他方法》。神经信息处理系统进展，34：4015–4027，2021年。

Radu Baltean-Lugojan、Pierre Bonami、Ruth Misener 和 Andrea Tramontani。通过训练神经网络为二次优化中的半正定割平面评分。预印本：*http://www. optimization-online. org/DB HTML/2018/11/6943. html*，2019。 -

Irwan Bello*、Hieu Pham*、Quoc V. Le、Mohammad Norouzi 和 Samy Bengio。利用强化学习进行神经组合优化，2017年。

Yoshua Bengio、Andrea Lodi 和 Antoine Prouvost。组合优化中的机器学习：方法论概览。《欧洲运筹学杂志》，290(2)：405–421，2021年。

David Bergman、Andre A Cire、Willem-Jan Van Hoeve 和 John Hooker。《用于优化的决策图》，第 1 卷。Springer，2016 年。

Ksenia Bestuzheva、Mathieu Besanc¸on、Wei-Kun Chen、Antonia Chmiela、Tim Donkiewicz、Jasper van Doornmalen、Leon Eifler、Oliver Gaul、Gerald Gamrath、Ambros Gleixner等。SCIP 优化套件 8.0。*arXiv* 预印本 *arXiv:2112.08872*，2021年。

Robert E Bixby。实现单纯形法：初始基。*ORSA Journal on Computing*，4(3):267–284，1992。

Robert E Bixby、Mary Fenelon、Zonghao Gu、Ed Rothberg 和 Roland Wunderling。混合整数规划：进展报告。载于 《最锐利的割：*Manfred Padberg* 及其工作的影响》，第 309–325 页。SIAM，2004。

Simon Bowly、Quentin Cappart、Jonas Charfreitag、Laurent Charlin、Didier Ch´etelat、Antonia Chmiela、JustinDumouchelle、Maxime Gasse、Ambros Gleixner、Aleksandr M、Kazachkov、Elias B、Khalil、Pawel Lichocki、Andrea Lodi、Miles Lubin、Chris J、Maddison、Christopher Morris、Dimitri J、Papageorgiou、Augustin Parjadis、Sebastian Pokutta、Antoine Prouvost、Lara Scav- uzzo 和 Giulia Zarpellon。组合优化中的机器学习，2021年。网址：https://www.ecole.ai/2021/ml4co-competition/。

Zixuan Cao、Yang Xu、Zhewei Huang 和 Shuchang Zhou。Ml4co-kida：数据集聚合中的知识继承。*arXiv* 预印本 *arXiv:2201.10328*，2022年。

Lei Chen。*Deep Learning and Practice with MindSpore*。Springer Nature，2021年。

Zhi-Long Chen。集成生产与配送调度：综述与扩展。运筹学，58(1):130–148，2010。

Santanu S Dey 和 Marco Molinaro。割平面选择面临的理论挑战。*Mathematical Progra mming*，170(1):237–266，2018a。

Santanu S Dey 和 Marco Molinaro。割平面选择面临的理论挑战。*Mathematical Progra mming*，170(1):237–266，2018b。

Reza Zanjirani Farahani 和 Masoud Hekmatfar。设施选址：概念、模型、算法与案例研究。Springer Science & Business Media，2009年。

FICO Xpress。Xpress优化套件，https://www.fico.com/en/products/ fico-xpress-opti mization，2020年。

Scott Fujimoto、Herke van Hoof 和 David Meger。解决演员-评论家方法中的函数逼近误差。载于 Jennifer Dy 和 Andreas Krause（编），第*35*届机器学习国际会议论文集，机器学习研究论文集第80卷，第1587–1596页。PMLR，2018年7月10–15日。

Maxime Gasse、Didier Chetelat、Nicola Ferroni、Laurent Charlin 和 Andrea Lodi。利用图卷积神经网络进行精确组合优化。载于 H. Wallach、H. Larochelle、A. Beygelzimer、F. d'Alch´e-Buc、E. Fox 和 R. Garnett（编），神经信息处理系统进展，第32卷。Curran Associates, Inc.，2019。

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, PhilippChristophel, KatiJarck, ThorstenKoch, Jeff Linderoth 等。Miplib2017：基于数据的第六版混合整数规划库编纂。数学规划计算，13(3):443–490，2021年。

Carla P Gomes、Willem-Jan van Hoeve 和 Ashish Sabharwal。网络连接：一种混合方法。载于 人工智能（*AI*）与运筹学（*OR*）技术在约束规划中的集成国际会议，第 303–307 页。Springer，2008年。

RalphGomory。混合整数问题的一种算法。技术报告，RAND公司，圣莫尼卡，加利福尼亚州，1960年。

Ian Goodfellow、Yoshua Bengio 和 Aaron Courville。深度学习。麻省理工学院出版社，2016年。

Prateek Gupta、Elias B Khalil、Didier Chet´elat、Maxime Gasse、Yoshua Bengio、Andrea Lodi 和 M Pawan Kumar。用于学习分支的回溯。*arXiv* 预印本 *arXiv:2206.14987*，2022年。

Gurobi。Gurobi求解器，https://www.gurobi.com/，2021年。

Tuomas Haarnoja、Aurick Zhou、Pieter Abbeel 和 Sergey Levine。软演员-评论家：采用随机演员的离策略最大熵深度强化学习。载于 Jennifer Dy 和 An- dreas Krause（编），第*35*届国际机器学习会议论文集，《机器学习研究论文集》第80卷，第1861–1870页。PMLR，2018年7月10–15日。

He He、Hal Daume III 和 Jason M Eisner。学习在分支定界算法中进行搜索。载于 Z. Ghahramani、M. Welling、C. Cortes、N. Lawrence 和 K.Q. Weinberger（编），神经信息处理系统进展，第 27 卷。Curran Associates, Inc., 2014年。

Gregor Hendel、Matthias Miltenberger 和 Jakob Witzig。使用老虎机算法解决混合整数规划的自适应算法行为。载于 运筹学研究进展 *2018*年，第 513–519 页。Springer，2019。

Sepp Hochreiter 和 J¨urgen Schmidhuber。长短期记忆。神经计算，9(8)：1735–1780，1997。

Zeren Huang、Kerong Wang、Furui Liu、Hui-Ling Zhen、Weinan Zhang、Mingxuan Yuan、Jianye Hao、Yong Yu 和 Jun Wang。学习为高效混合整数规划选择割。模式识别，123：108353，2022年。ISSN 0031-3203。doi：https://doi.org/10.1016/j.patcog. 2021.108353。

Frank Hutter、Holger H Hoos 和 Kevin Leyton-Brown。《混合整数规划求解器的自动配置》。发表于人工智能（*AI*）与运筹学（*OR*）技术在约束规划中的集成国际会议，第186–202 页。Springer，2010。

Michael J¨unger、Thomas M Liebling、Denis Naddef、George L Nemhauser、William R Pulleyblank、Gerhard Reinelt、Giovanni Rinaldi 和 Laurence A Wolsey。《整数规划五十年（*1958—2008*）：从早期发展到最先进水平》。Springer Science & Business Media，2009。

Elias Khalil、Pierre Le Bodic、Le Song、George Nemhauser 和 Bistra Dilkina。《在混合整数规划中学习分支》。*AAAI*人工智能会议论文集，30(1)，2016年2月。doi: 10.1609/aaai.v30i1.10080。URL https://ojs.aaai.org/index. php/AAAI/article/view/10080。

Elias B Khalil、Bistra Dilkina、George L Nemhauser、Shabbir Ahmed 和 Yufen Shao。学习在树搜索中运行启发式方法。载于*IJCAI*，第 659–666 页，2017年。

Diederik P Kingma 和 Jimmy Ba。Adam：一种随机优化方法。*arXiv* 预印本 *arXiv:1412.6980*，2014年。

VR Konda 和 JN Tsitsiklis。论演员-评论家算法。*SIAM*控制与优化杂志，42(4)：1143–1166，2003年。

Ailsa H Land 和 Alison G Doig。一种求解离散规划问题的自动方法。载于整数规划五十年（*1958—2008*），第 105–132 页。Springer，2010。

Gilbert Laporte. 车辆路径规划五十年。《交通科学》，43(4):408–416，2009。Jan Karel Lenstra 和 David Shmoys。旅行商问题：一项计算研究，2009。Xijun Li、Qingyu Qu、Fangzhou Zhu、Jia Zeng、Mingxuan Yuan、Kun Mao 和 Jie Wang。学习如何重新表述线性规划。*arXiv* 预印本 *arXiv:2201.06216*，2022年。Andrea Lodi 和 Giulia Zarpellon。关于学习与分支：一篇综述。*Top*，25(2):207–236，2017年。Istv´an Maros。单纯形法的计算技术，第61卷。Springer Science & Business Media，2002年。John E Mitchell。组合优化问题的分支切割算法。《应用优化手册》，1(1):65–77，2002年。

Volodymyr Mnih、Adria Puigdomenech Badia、Mehdi Mirza、Alex Graves、Timothy Lillicrap、Tim Harley、David Silver 和 Koray Kavukcuoglu。用于深度强化学习的异步方法。载于 Maria Florina Balcan 和 Kilian Q. Weinberger（编），《第*33*届国际机器学习大会论文集》，《机器学习研究论文集》第48卷，第1928–1937页，纽约，纽约，美国，2016年6月20日至22日。PMLR。

Mehryar Mohri、Afshin Rostamizadeh 和 Ameet Talwalkar。《机器学习基础》。麻省理工学院出版社，2018年。

Mouad Morabit、Guy Desaulniers 和 Andrea Lodi。用于列生成的基于机器学习的列选择。《交通科学》，55(4):815–831，2021年。

Ofir Nachum、Shixiang Shane Gu、Honglak Lee 和 Sergey Levine。数据高效的分层强化学习。《神经信息处理系统进展》，31，2018年。

Vinod Nair、Sergey Bartunov、Felix Gimeno、Ingrid von Glehn、Pawel Lichocki、Ivan Lobov、Bren- dan O'Donoghue、Nicolas Sonnerat、Christian Tjandraatmadja、Pengming Wang 等。使用神经网络求解混合整数规划。《arXiv 预印本 arXiv:2012.13349》，2020年。

Vangelis Th Paschos。《组合优化的应用》，第3卷。约翰·威立父子公司，2014年。

Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga、Alban Desmaison、Andreas Kopf、Edward Yang、Zachary DeVito、Martin Raison、Alykhan Tejani、Sasank Chilamkurthy、Benoit Steiner、Lu Fang、Junjie Bai 和 Soumith Chintala。PyTorch：一种命令式风格、高性能的深度学习库。载于 H. Wallach、H. Larochelle、A. Beygelzimer、F. d'Alch´e-Buc、E. Fox 和 R. Garnett（编），神经信息处理系统进展，第32卷。Curran Associates, Inc.，2019。

Max B Paulus、Giulia Zarpellon、Andreas Krause、Laurent Charlin 和 Chris Maddison。通过前瞻学会生成割：基于模仿学习的割平面选择。载于 Kamalika Chaudhuri、Stefanie Jegelka、Le Song、Csaba Szepesvari、Gang Niu 和 Sivan Sabato（编），第*39*届国际机器学习大会论文集，机器学习研究论文集第162卷，第17584–17600页。PMLR，2022年7月17日至23日。

Yves Pochet 和 Laurence A Wolsey。《混合整数规划生产计划》，第 149 卷。Springer，2006年。

Ashish Sabharwal、Horst Samulowitz 和 Chandra Reddy。使用 UCT 指导组合优化。载于国际会议论文集：约束规划中的人工智能（*AI*）与运筹学（*OR*）技术集成，第 356–361 页。Springer，2012年。

Tim Salimans、Jonathan Ho、Xi Chen、Szymon Sidor 和 Ilya Sutskever。进化策略：强化学习的一种可扩展替代方案。*arXiv* 预印本 *arXiv:1703.03864*，2017年。

Lara Scavuzzo、Feng Yang Chen、Didier Ch´etelat、Maxime Gasse、Andrea Lodi、Neil Yorke-Smith 和 Karen Aardal。使用树形马尔可夫决策过程学习分支。*arXiv* 预印本 *arXiv:2205.11107*，2022年。

John Schulman、Sergey Levine、Pieter Abbeel、Michael Jordan 和 Philipp Moritz。信赖域策略优化。载于 Francis Bach 和 David Blei（编），第 *32* 届国际机器学习会议论文集，机器学习研究论文集第 37 卷，第 1889–1897 页，法国里尔，2015年7月7–9日。PMLR。

John Schulman、Filip Wolski、Prafulla Dhariwal、Alec Radford 和 Oleg Klimov。近端策略优化算法。*arXiv* 预印本 *arXiv:1707.06347*，2017年。

Haoran Sun、Wenbo Chen、Hui Li 和 Le Song。通过强化学习改进学习分支。载于*NeurIPS 2020* 学习与组合算法交汇专题，2020年。

Ilya Sutskever、Oriol Vinyals 和 QuocV Le。利用神经网络进行序列到序列学习。载于 Z. Ghahramani、M. Welling、C. Cortes、N. Lawrence 和 K.Q. Weinberger（编），神经信息处理系统进展，第 27 卷。Curran Associates, Inc.，2014年。

Richard S Sutton 和 Andrew G Barto。强化学习：导论。麻省理工学院出版社，2018年。

Richard S Sutton、David McAllester、Satinder Singh 和 Yishay Mansour。结合函数逼近的强化学习策略梯度方法。神经信息处理系统进展，12，1999a。

Richard S Sutton、Doina Precup 和 Satinder Singh。介于马尔可夫决策过程与半马尔可夫决策过程之间：强化学习中时间抽象的框架。人工智能，112(1-2)：181–211，1999b。

Yunhao Tang、Shipra Agrawal 和 Yuri Faenza。整数规划的强化学习：学习生成割。收录于 Hal Daum´e III 和 Aarti Singh（编），第*37*届国际机器学习会议论文集，第119卷，机器学习研究论文集，第9367–9376页。PMLR，2020年7月13–18日。

Mark Turner、Thorsten Koch、Felipe Serrano 和 Michael Winkler。混合整数规划中的自适应割选择整数线性规划。*arXiv* 预印本 *arXiv:2202.10962*，2022年。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Ł ukasz Kaiser 和 Illia Polosukhin。Attention Is All You Need。载于 I. Guyon、U. Von Luxburg、S. Bengio、H. Wallach、R. Fergus、S. Vishwanathan 和 R. Garnett（编），神经信息处理系统进展，第 30 卷。Curran Associates, Inc.，2017年。

Oriol Vinyals、Meire Fortunato 和 Navdeep Jaitly。指针网络。载于 C. Cortes、N. Lawrence、D. Lee、M. Sugiyama 和 R. Garnett（编），神经信息处理系统进展，第 28 卷。Curran Associates, Inc.，2015年。

Oriol Vinyals、Samy Bengio 和 Manjunath Kudlur。顺序很重要：集合的序列到序列。载于 Yoshua Bengio 和 Yann LeCun（编），第 *4* 届国际学习表征会议（*ICLR 2016*），波多黎各圣胡安，*2016* 年 *5* 月 *2* 日至 *4* 日，会议论文集，2016 年。网址：http://arxiv.org/abs/1511.06391。

Franz Wesselmann 和 U Stuhl。割平面管理与选择技术的实现技术报告。帕德博恩大学，2012年。

Giulia Zarpellon、Jason Jo、Andrea Lodi 和 Yoshua Bengio。通过参数化分支定界搜索树来学习分支策略。载于*AAAI*人工智能会议论文集，第35卷，第3931–3939页，2021年。

# A 证明

## A.1 命题 1 的证明

证明。 优化目标的形式为

$$\begin{aligned} J(\theta) &= \mathbb{E}_{s\sim\mu, a_k\sim\pi_\theta(\cdot|s)}[r(s,a_k)] \\ &= \mathbb{E}_{s\sim\mu}[\sum_{a_k}\int_{k=0}^{1}\pi^h_{\theta_1}(k|s)\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)dk] \\ &= \mathbb{E}_{s\sim\mu}[\int_{k=0}^{1}\sum_{a_k}\pi^h_{\theta_1}(k|s)\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)dk], \end{aligned}$$

其中 $\theta = [\theta_1, \theta_2]$ 与 $[\theta_1, \theta_2]$ 表示两个向量的拼接，$\mathbb{E}_{k\sim\pi^h_{\theta_1}(\cdot|s)}[\pi^h_{\theta_2}(a_k|s,k)]$，$\pi_\theta = \mu(a_k|s)\pi_\theta(k|s)$，并表示初始状态分布。

我们首先计算 $\theta_1$的策略梯度:

$$\begin{aligned} &\nabla_{\theta_1} J([\theta_1,\theta_2]) \\ =&\nabla_{\theta_1}(\mathbb{E}_{s\sim\mu}[\int_{k=0}^{1}\sum_{a_k}\pi^h_{\theta_1}(k|s)\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)dk]) \\ =&\mathbb{E}_{s\sim\mu}[\nabla_{\theta_1}[\int_{k=0}^{1}\pi^h_{\theta_1}(k|s)\sum_{a_k}\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)dk]]. \end{aligned}$$

Let

$$\begin{aligned} r^h(s,k,\theta_2) &= \sum_{a_k}\pi^l_{\theta_2}(a_k|s,k)r(s,a_k) \\ &= \mathbb{E}_{a_k\sim\pi^l_{\theta_2}(\cdot|s,k)}[r(s,a_k)], \end{aligned}$$

于是我们有

$$\begin{aligned} \nabla_{\theta_1} J([\theta_1,\theta_2]) &= \mathbb{E}_{s\sim\mu}[\nabla_{\theta_1}[\int_{k=0}^{1}\pi^h_{\theta_1}(k|s)r(s,k,\theta_2)dk]] \\ &= \mathbb{E}_{s\sim\mu,k\sim\pi^h_{\theta_1}(\cdot|s)}[\nabla_{\theta_1}\log\pi^h_{\theta_1}(k|s)r(s,k,\theta_2)]. \end{aligned}$$

因此，我们有

$$\begin{aligned} &\nabla_{\theta_1} J([\theta_1,\theta_2]) \\ &= \mathbb{E}_{s\sim\mu,k\sim\pi^h_{\theta_1}(\cdot|s)}[\nabla_{\theta_1}\log(\pi^h_{\theta_1}(k|s))\mathbb{E}_{a_k\sim\pi^l_{\theta_2}(\cdot|s,k)}[r(s,a_k)]]. \end{aligned}$$

然后，我们计算 $\theta_2$的策略梯度:

$$\begin{aligned} &\nabla_{\theta_2} J([\theta_1,\theta_2]) \\ =&\nabla_{\theta_2}(\mathbb{E}_{s\sim\mu}[\int_{k=0}^{1}\sum_{a_k}\pi^h_{\theta_1}(k|s)\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)dk]) \\ =&\mathbb{E}_{s\sim\mu,k\sim\pi^h_{\theta_1}(\cdot|s)}[\nabla_{\theta_2}[\sum_{a_k}\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)]] \\ =&\mathbb{E}_{s\sim\mu,k\sim\pi^h_{\theta_1}(\cdot|s),a_k\sim\pi^l_{\theta_2}(\cdot|s,k)}[\nabla_{\theta_2}\log\pi^l_{\theta_2}(a_k|s,k)r(s,a_k)], \end{aligned}$$

至此，证明完成。 □

# B 相关工作

**混合整数线性规划的机器学习**。近年来，利用机器学习方法提升混合整数线性规划求解器性能，一直是备受关注的研究课题（Bengio 等人，

2021年；Lodi 和 Zarpellon，2017年；Bowly 等人，2021年；Gasse 等人，2019年）。在求解器的求解过程中，许多对求解器性能有重大影响的关键决策都基于启发式方法（Achterberg，2007年）。近期研究提出用机器学习模型取代这些人工设计的启发式方法（Bengio 等人，2021年）。这一研究方向已显著提升求解器性能，涉及割选择（Tang 等人，2020年；Paulus 等人，2022年；Turner 等人，2022年；Baltean-Lugojan 等人，2019年）、变量选择（Khalil 等人，2016年；Gasse 等人，2019年；Balcan 等人，2018年；Zarpellon 等人，2021年）、节点选择（He 等人，2014年；Sabharwal 等人，2012年）、列生成（Morabit 等人，2021年）以及原始启发式选择（Khalil 等人，2017年；Hendel 等人，2019年）。本文聚焦于割选择，它在现代混合整数线性规划求解器中发挥着重要作用（Dey 和 Molinaro，2018a；Tang 等人，2020年）。

对于割选择，许多现有的基于学习的方法（Tang 等，2020；Paulus 等人，2022；Huang 等，2022）着重于学习应优先选择哪些割，方法是学习一个用于衡量割质量的评分函数。具体而言，Tang 等（2020）提出了一种强化学习方法，用于学习如何为 Gomory 割（Gomory，1960）评分，并选择得分最高的 Gomory 割。此外，Paulus 等人（2022）设计了一种前瞻选择规则，选择能够带来最大对偶界改进的割，并提出通过模仿学习来学习该专家规则。Huang 等（2022）没有选择得分最高的单个割，而是将割选择建模为多实例学习问题，以学习评分函数，并选取得分较高的固定比例的割。然而，他们忽视了学习应选取多少个割的重要性。此外，我们通过实证表明，选定割的顺序会对求解混合整数线性规划问题的效率产生重大影响（见第3节）。此外，Turner 等，2022提出学习由专家设计的四种现有评分规则的权重。对于理论分析，Balcan 等人，2021年为学习割选择策略提供了一些可证明的保证。

**序列模型**。 诸如长短期记忆和 Transformer 之类的序列模型在机器翻译等语言任务中取得了卓越的性能（Hochreiter & Schmidhuber，1997；Sutskever等人，2014年；Vaswani 等人，2017年）。对于组合优化，近期工作（Vinyals 等人，2015年；Bello* 等人，2017年）提出了传统序列模型的一种变体，即指针网络，并将其应用于直接寻找特定组合优化问题的解，例如旅行商问题（Lenstra & Shmoys，2009）。我们不直接寻找解，而是提出在现代混合整数线性规划求解器中使用指针网络进行割选择。据我们所知，我们是首个将指针网络应用于割选择的研究，这种方法不仅能够捕捉选定割的顺序，还能很好地捕捉割之间的交互，从而选择彼此互补的割。

## C 背景详情

### C.1 原始-对偶间隙积分详情

运行分支割平面法时，我们会跟踪两个重要的界，包括全局原始界和全局对偶界。全局原始界对应于当前找到的最佳可行解的值，即问题（1）的最佳上界。全局对偶界对应于搜索树所有叶节点中的最小对偶界，即问题（1）的最佳下界。我们将原始-对偶间隙积分定义为求解器的全局原始界曲线与全局对偶界曲线之间的面积。在时间限制 $T$ 下，我们将原始-对偶间隙积分定义为

$$\int_{t=0}^{T} (\mathbf{c}^T \mathbf{x}_t^* - \mathbf{z}_t^*) \mathrm{d}t,$$

其中，$\mathbf{c}$ 是（1）式中的目标系数向量，$\mathbf{x}_t^*$ 是在时间 $t$找到的最佳可行解，$\mathbf{z}_t^*$是在时间 $t$的最佳对偶界。我们将全局原始界与全局对偶界之差定义为原始-对偶间隙 。在 SCIP 8.0.0（Bestuzheva 等，2021）中，原始-对偶间隙的初始值设为常数 100。原始-对偶间隙积分是评估求解器性能的公认指标。例如，原始-对偶间隙积分是 NeurIPS 2021 ML4CO竞赛（Bowly等人，2021年）中的主要评估指标。

**算法 1 构建 MIPLIB 数据集的伪代码 1：输入初始实例 $I_0$、完整 MIPLIB $\mathcal{M}$ 集合、空集 $\mathcal{M}'$、空队列 $Q$。2：用实例 $I_0$ 初始化 $Q$，$I_0 \to Q$3：当 Q 不为空时执行 4：n= Q.size() 5：对于 $i = 1$，...，$n$ 执行 6：从 $Q$中取出一个元素，即 $I'$7：计算 $\mathcal{M}$ 中除 $I'$ 外每个实例与 $I'$ 之间的相似度分数 8：选择相似度分数最高的五个实例 $\mathcal{M}_i$9：对于 $\mathcal{M}_i$ 中的 $I$ 执行 10：如果 $I$ 不在 $\mathcal{M}'$ 中 11：将 $I$ 推入 $\mathcal{M}'$；将 $I$ 推入 $Q$12：结束如果 13：结束循环 14：结束循环 15：结束当 16：返回 $\mathcal{M}'$**

## D 本文所用数据集的详细信息

### D.1 主要评估中使用的数据集

**简单数据集**。SCIP 8.0.0 求解器需要一分钟才能将简单数据集中的混合整数线性规划实例求解至最优。简单数据集由三类合成混合整数线性规划问题组成：集合覆盖（Balas 和 Ho，1980）、最大独立集（Bergman 等人，2016）和多重背包（Scavuzzo 等人，2022）。我们选择这三类问题有以下原因。首先，它们被广泛用作评估混合整数线性规划求解器的基准测试（Gasse 等人，2019；Huang 等，2022；Sun 等人，2020；Gupta 等人，2022）。其次，它们代表了实践中遇到的各类混合整数线性规划问题。第三，对于每一类问题，生成割的平均数量至少为 20，这确保了恰当的割选择策略对于提升求解器性能具有重要意义。与（Gasse 等人，2019；Scavuzzo 等人，2022；Sun 等人，2020；Gupta 等人，2022）类似，我们生成包含 500 行和 1000 列的集合覆盖实例、图含 500 个节点且亲和度设为 4 的最大独立集实例，以及包含 60 个物品和 12 个背包的多重背包实例。对于每个基准测试，我们生成包含 10,000 个实例的训练集，以及包含 100 个训练期间从未见过的实例的测试集。具体而言，读者可以参考 https://github.com/ds4dm/learn2branch 或 https://github.com/lascavana/rl2branch，获取生成简单数据集的代码。论文被接收发表后，我们也将发布代码。

**中等难度数据集**。SCIP 8.0.0 求解器至少需要五分钟，才能将中等难度数据集中的实例求解至最优。遵循 He 等人（2014年）；Hutter 等人（2010）；Nair 等人（2020）的做法，中等难度数据集包括 MIK（Atamt¨urk，2003），这是一个包含背包约束的混合整数线性规划问题集合；以及 CORLAT（Gomes 等人，2008），这是一个用于在北落基山脉地区为灰熊修建野生动物廊道的真实数据集。每个问题集都按实例数量的 80% 和 20% 划分为训练集和测试集。有关 MIK 的信息，请参阅 https://atamturk.ieor.berkeley.edu/data/mixed.integer.knapsack/。有关 CORLAT 的信息，请参阅 https://bitbucket.org/mlindauer/aclib2/src/master/。

**困难数据集**。SCIP 8.0.0 求解器至少需要一小时才能将困难数据集中的实例求解至最优。

表 5：从 MIPLIB 2017 中移除实例的标准。

| 标准 | 实例移除比例 |
|---|---|
| 标签：可行性，数值问题，不可行，无解 | 4.5%，17.4%，2.8%，0.9% |
| 在默认条件下，预求解时间超过300秒 | 4.8% |
| 在根节点求解至最优 | 9.9% |

**(1) 来自 MIPLIB 2017 的基准测试**。请注意，MIPLIB 2017（MIPLIB）（Gleixner 等人，2021年）包含来自许多不同应用领域的混合整数线性规划问题实例，并长期以来一直被用作混合整数线性规划求解器的标准基准（Nair 等人，2020年；Turner 等，2022；Gleixner 等，2021）。先前的研究（Turner 等，2022）表明，直接在完整的 MIPLIB 上进行学习极具挑战性，因为这些实例具有异构性，而机器学习

表 6：所用数据集的统计描述。在所有数据集中，$m$ 表示约束的平均数量，$n$ 表示变量的平均数量。推理时间表示所提出的 HEM 在候选割平均数量下的推理时间。

| 数据集 | 集合覆盖 | 最大独立集 | 多重背包 | MIK | CORLAT | 负载均衡 | 匿名 | MIPLIB mixed neos | MIPLIB mixed supportcase |
|---|---|---|---|---|---|---|---|---|---|
| $m$ | 500 | 1953 | 72 | 346 | 486 | 64304 | 49603 | 5660 | 19910 |
| $n$ | 1000 | 500 | 720 | 413 | 466 | 61000 | 37881 | 6958 | 19766 |
| 平均候选割 | 780.51 ± 289.92 | 57.04 ± 15.53 | 45.00 ± 12.71 | 62.00 ± 13.1 | 60.00 ± 33.29 | 392.53 ± 32.92 | 79.40 ± 72.64 | 239.00 ± 154 | 173.25 ± 267.27 |
| 推理时间（秒） | 1.58 | 0.11 | 0.09 | 0.12 | 0.12 | 0.77 | 0.15 | 0.47 | 0.34 |

难以从异构数据集中学习。尽管面临这一挑战，我们仍迈出了在 MIPLIB 子集上进行学习的第一步。具体而言，我们从 MIPLIB 中选取相似实例，构建了两个子集。我们通过 100 个人工设计的实例特征来衡量实例之间的相似性（Gleixner 等，2021）。按照 Turner 等（2022）的做法，我们首先剔除 MILLIB 中符合表 5 任一条件的实例。这样可以确保在剩余实例上，良好的割选择策略能够显著改善对偶界。请注意，为尽可能保留更多实例，我们只使用了 Turner 等（2022）所用七项标准中的三项。

为了从 MIPLIB 2017 中选择相似实例，我们首先选择一个具有背包约束的代表性实例（neos-1456979），以及一个具有集合覆盖约束的代表性实例（supportcase40）。然后，以 neos-1456979 为初始实例，按照算法 1 中的流程构建数据集 MIPLIB mixed neos。以 supportcase40 为初始实例，按照算法 1 中的流程构建数据集 MIPLIB mixed supportcase。请注意，我们使用 100 个人工设计的实例特征来衡量实例之间的相似性（Gleixner 等，2021）。每个数据集按实例数量的 80% 和 20% 划分为训练集和测试集。

具体而言，MIPLIB mixed neos 包含 20 个实例：neos-1456979、ic97 tension、icir97 tension、l2p12、lectsched-4-obj、lectsched-5-obj、loopha13、neos-686190、neos-2294525-abba、neos-3009394-lami、neos-3046601-motu、neos-3046615-murg、neos-3610173-itata、neos-4338804-snowy、neos-5221106-oparau、neos-5260764-orauea、neos-5261882-treska、neos-5266653-tugela、neos16 和 timtab1CUTS。

此外，MIPLIB 混合支持案例包含 40 个实例：supportcase40、30 70 45 05 100、30 70 45 095 100、acc-tight2、acc-tight4、acc-tight5、comp07-2idx、comp08-2idx、comp12- 2idx、comp21-2idx、decomp1、decomp2、gus-sch、istanbul-no-cutoff、mkc、mkc1、neos-555343、neos-555424、neos-738098、neos-872648、neos-933562、neos-933638、neos-933966、neos- 935234、neos-935769、neos-983171、neos-1330346、neos-1337307、neos-1396125、neos-3209462- rhin、neos-3755335-nizao、neos-3759587-noosa、neos-4300652-rahue、neos18、physicianshed6-1、physicianshed6-2、piperout-d27、qiu、reblock354 和 supportcase37。

**(2) NeurIPS 2021 ML4CO竞赛中使用的基准测试**正文中使用的负载均衡和匿名问题来自 NeurIPS 2021 ML4CO竞赛（Bowly 等人，2021年）。读者可访问 https://www.ecole.ai/2021/ml4co-competition/ 了解竞赛详情。该竞赛发布了三个具有挑战性的数据集，但我们只使用其中两个。主要原因是，第三个数据集（物品放置）中的实例平均候选割数量不足五个，因此割选择对求解器整体性能影响甚微。

#### D.1.1 上述数据集的详细说明

本部分将详细介绍上述数据集。请注意，除 MIPLIB 2017 外，我们使用的所有数据集都针对特定应用，即每个数据集仅包含来自单一应用的实例。我们在表 6 中汇总了本文所用数据集的统计描述。令 $n$，$m$ 表示混合整数线性规划问题中的变量和约束的平均数量。令 $m \times n$ 表示混合整数线性规划问题的规模。我们强调，我们所用数据集的最大规模比以往基于学习的割选择方法所用数据集的最大规模大两个数量级（Tang 等，2020；Paulus 等人，2022），这体现了我们提出的 HEM 的优越性。此外，我们还测试了所提出的 HEM 在平均候选割数量下的推理时间。表 6 的结果显示，HEM 的计算开销非常低。

### D.2 正文第3节中使用的数据集

在正文的图 1a 中，我们分别使用了五个具有挑战性的数据集，即 D1、D2、D3、D4 和 D5。具体而言，D1 表示 MIPLIBmixed 支持案例，D2 表示单个实例

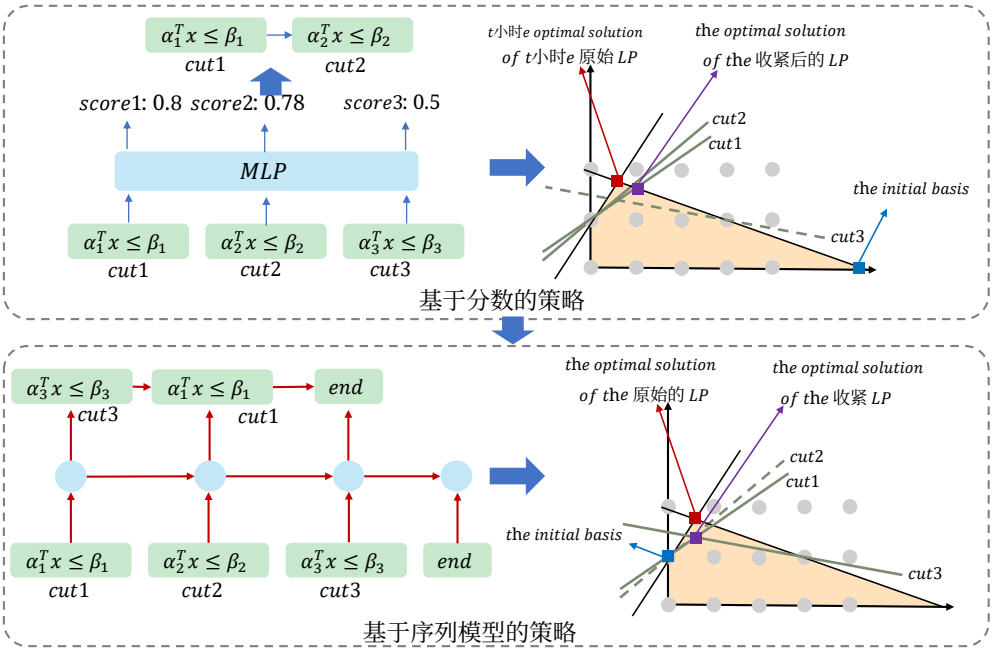



图 4：使用序列到序列模型选择割与使用评分函数选择割的示意图。序列模型有两个主要优点。首先，它通过逐个选择割来捕捉割之间的交互。因此，它会选择彼此互补良好的割3和割1，从而得到更收紧的线性规划松弛。其次，它能自然地捕捉选定割的顺序。更好的选定割顺序可能会带来更好的初始基，从而更快地求解线性规划松弛（Li 等，2022）（见第3节）。

neos-1456979（来自 MIPLIB 2017），D3 表示 MIPLIB mixed neos，D4 表示匿名，D5 表示来自 MIPLIB 2017 的单个实例 lectsched-5-obj。在正文的图 1b 中，我们使用了数据集 MIPLIBmixed neos。

### D.3 大规模现实世界生产计划问题

生产计划问题旨在根据每日订单需求，为数千家工厂制定最优生产计划。约束包括各工厂每条生产线的生产能力、运输限制、订单率等。优化目标是同时最大限度地降低生产成本和交货时间。我们将数据集中的实例按 80% 和 20% 的比例划分为训练集和测试集。生产计划问题的平均规模约为 $3500 \times 5000 = 1.75 \times 10^8$，属于大规模现实世界问题。为了推动混合整数线性规划领域的机器学习研究，我们将在论文被接收发表后发布该数据集。

## E 使用序列模型的优势示例

图 4 展示了使用序列模型学习割选择的两大优势。首先，序列模型将选定割建模为输出序列，从而考虑选定割的顺序。如图 4 所示，序列模型确定的割顺序优于基于评分的方法，因此能够为求解线性规划松弛提供更好的初始基，加快求解速度。其次，序列模型通过对给定候选割输入序列时选定割的联合条件概率进行建模，捕捉割之间的交互。如图 4 所示，序列模型选出的割彼此互补得很好，从而得到更收紧的线性规划松弛，并加快混合整数线性规划的求解速度。

## F 算法实现与实验设置

### F.1 设计的割特征

继 Huang 等人（2022年）；Wesselmann 和 Stuhl（2012年）；Dey 和 Molinaro（2018b）；Achterberg（2007年）之后，我们为割选择任务设计了 13 个割特征，例如某个割在多大程度上违反当前线性规划解，以及该割的非零系数所占比例。我们

表7：生成割 $\boldsymbol{\alpha}^T\mathbf{x} \leq \beta$ 的设计割特征。（假设$\mathbf{c}$ 表示目标系数。）

| 特征 | 描述 | 数量 |
|---|---|---|
| 割系数 | 割系数的均值、最大值、最小值和标准差 | 4 |
| 目标系数 | 目标系数的均值、最大值、最小值和标准差 | 4 |
| 平行度 | 目标函数与割$\boldsymbol{\alpha}$之间的平行度 $\frac{\mathbf{c}^T\boldsymbol{\alpha}}{\|\mathbf{c}\|\|\boldsymbol{\alpha}\|}$ | 1 |
| 效能 | 割超平面与当前线性规划解之间的欧几里得距离 | 1 |
| 支撑度 | 割的非零系数比例 | 1 |
| 整数支撑度 | 该割的非零系数相对于整数变量的比例 | 1 |
| 归一化违反度 | 该割对当前线性规划解的违反度 $\max\{0, \frac{\boldsymbol{\alpha}^T\mathbf{x}^* - \beta}{\|\boldsymbol{\alpha}\|}\}$ | 1 |

表7详细介绍了设计的割特征。需要强调的是，我们没有对割特征进行调优。因此，未来可以通过设计更好的割特征或使用图神经网络学习更好的特征，进一步改进我们的方法。

### F.2 基线方法的实现细节

本节将详细介绍本文中使用的所有基线方法。我们用 $\boldsymbol{\alpha}^T\mathbf{x} \leq \beta$ 表示一个割，用 $\mathbf{x}^*$表示当前线性规划松弛的最优解。在所有实验中，对于所有基于评分的规则和学习型基线方法，我们都将选定割的比例设为 0.2。

**随机**。 随机方法以随机方式选取固定比例的候选割。本文中该比例设为 0.2。

**归一化违反度（NV）**。NV 是一种基于评分的规则。它根据每个割相对于当前线性规划解的归一化违反度进行评分，并选取固定比例的高分割。归一化违反度定义为 $\max\{0, \frac{\boldsymbol{\alpha}^T\mathbf{x}^* - \beta}{\|\boldsymbol{\alpha}\|}\}$。本文中该比例设为 0.2。

**效能（Eff）**。 Eff 是一种基于评分的规则。它根据每个割超平面与当前线性规划解之间的欧几里得距离进行评分，并选取固定比例的高分割。本文中该比例设为 0.2。

**默认**。 默认是 SCIP 8.0.0 中使用的默认割选择规则（Bestuzheva 等，2021年）。有关 SCIP 默认割选择规则的详细说明，请参阅（Achterberg，2007年）。请注意，Default 通过人工设计的启发式方法处理割选择中的两个问题：（1）应优先选择哪些割；（2）应选择多少个割。也就是说，Default 选择的割的比例可变，而非固定比例。

**基于分数的策略（SBP）**。由于目前最先进的（SOTA）基于强化学习的割选择方法（Tang 等，2020年）是为每轮选择最佳割的场景设计的，因此我们对该 SOTA 方法进行了少量调整，使其适用于我们每轮选择一组割的场景，即 SBP。具体而言，SBP 的核心思想是学习一个评分函数来衡量割质量，这与 Tang 等人（2020年）、Huang 等人（2022年）和 Paulus 等人（2022年）的方法相同。为公平比较，SBP 使用与 HEM 相同的割特征，我们也通过强化学习训练 SBP。我们实现的 SBP 同样是对 Huang 等人（2022年）提出的方法稍作调整后的版本。我们强调，正文中的实验表明，我们实现的 SBP 是一个强大的基线。具体而言，我们使用多层感知机实现评分函数，以预测给定割的分数。也就是说，评分函数根据割的特征预测其分数。该多层感知机网络包含两个隐藏层，每层有 128 个单元。此外，我们像 Tang 等人（2020年）一样，通过进化策略训练评分函数。论文获准发表后，我们也会发布 SBP 的实现代码。

**算法2**用于训练 HEM 的伪代码

**1：初始化分层序列模型 $\pi_{[\theta_1,\theta_2]}$、混合整数线性规划实例 $\mathcal{D}$、训练数据集 $\mathcal{D}_{\text{train}}$、批量大小 $N_b$、训练轮次 $N_e$、策略学习率 $\alpha$ 2：对于 $N_e$ 轮训练，执行以下操作：3：清空训练数据集 $\mathcal{D}_{\text{train}}$ 4：对于 $N_b$ 步，执行以下操作：5：从 $\mathcal{D}$ 中随机抽取一个混合整数线性规划问题 $s_0$ 6：使用策略 $\pi$ 在状态 $s_0$ 下采取动作 $k$ 和 $a_k$ 7：获得奖励 $r$，并将 $(s_0, k, a_k, r)$ 添加到 $\mathcal{D}_{\text{train}}$ 8：结束循环 9：按照命题 1，使用 $\mathcal{D}_{\text{train}}$ 计算分层策略梯度 10：更新参数 $\theta_1 = \theta_1 + \alpha\nabla_{\theta_1} J\left([\theta_1, \theta_2]\right)$ 、 $\theta_2 = \theta_2 + \alpha\nabla_{\theta_2} J\left([\theta_1, \theta_2]\right)$ 11：结束循环**

### F.3 实现细节和超参数

#### F.3.1 硬件规格

在所有实验中，我们都使用一台配备八个图形处理器设备（NVidia GeForceGTX3090Ti）和两颗 Intel Gold 6246R CPU 的机器。

#### F.3.2 求解器设置

为确保结果可复现，我们强调，正文中的所有结果均通过对 SCIP 随机种子 $\{1, 2, 3\}$ 得到的结果取平均获得。

#### F.3.3 奖励函数

在简单数据集上，我们将奖励设为负求解时间。在中等和困难数据集上，我们将奖励设为 300秒时间限制内的负原始-对偶间隙积分。

对于现实世界的生产规划问题，我们将奖励设为 600 秒时间限制内的负原始-对偶间隙积分，或负对偶界改进。正文中报告的结果是由采用负对偶界改进作为奖励的 HEM 得出的。

表 8：在奖励为负 PD 积分的现实世界生产计划问题上进行评估。结果表明，HEM 仍显著优于所有基线方法。

| | 生产计划 | | | |
|---|---|---|---|---|
| 方法 | 时间（秒） | 时间改进率（%） | PD 积分 | 改进（PD积分，%） |
| NoCuts | 278.79 | NA | 17866.01 | NA |
| 默认 | 296.12 | -6.22 | 17703.39 | 0.91 |
| 随机 | 280.18 | -0.50 | 18120.21 | -1.42 |
| NV | 259.48 | 6.93 | 17295.18 | 3.20 |
| Eff | 263.60 | 5.45 | 16636.52 | 6.88 |
| SBP | 276.61 | 0.78 | 16952.85 | 5.11 |
| HEM（本文方法） | **251.64** | **9.74** | **16533.05** | **7.46** |

奖励。表 8 展示了采用负原始-对偶间隙积分作为奖励的 HEM 的性能。结果仍然表明，就时间和 PD 积分而言，HEM 显著优于所有基线方法。

我们强调，在现实世界问题中，可以根据目标设定奖励。例如，假设我们的目标是在时间限制内最小化原始-对偶间隙，那么可以将时间限制内的原始-对偶间隙设为奖励。

#### F.3.4 策略网络架构

高层模型包含一个LSTM编码器和一个多层感知机。LSTM网络将可变大小输入编码为维度为128的隐藏向量。多层感知机网络包含两个各有128个单元的隐藏层。低层模型本质上是一个指针网络。我们将指针网络的超参数设为与（Bello 等人，2017年）中相同。

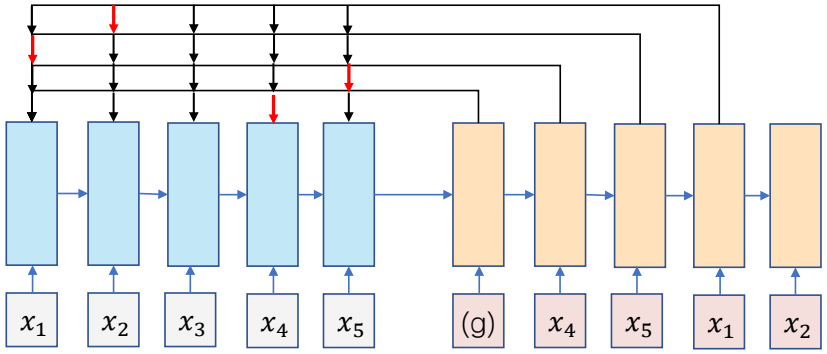



图5：Vinyals 等人（2015年）提出的指针网络架构示意图。

#### F.3.5 优化

在所有实验中，我们使用学习率为 $\alpha_1 = 1 \times 10^{-4}$ 的Adam 优化器来优化低层模型，并使用学习率为 $\alpha_2 = 5 \times 10^{-4}$ 的Adam 优化器来优化高层模型。每个时期，我们收集32个样本用于训练，并将总时期数设为100。正如Tang 等人（2020年）所示，学习出良好的割选择策略并不需要太多数据，这令人惊讶。为确保训练稳定，我们延迟更新高层策略。这构成了一种双时间尺度算法，而这通常是在线性设定下保证收敛所需的（Fujimoto 等人，2018年；Konda 与 Tsitsiklis，2003年）。我们将延迟更新频率设为两次。也就是说，我们先训练低层策略两次，再训练高层策略一次。此外，附录G.3.4 中的结果表明，HEM的收敛表现对超参数“延迟更新频率”并不敏感。

#### F.3.6 训练算法

我们在算法2中给出了 HEM 的训练算法流程。

### F.4 HEM 的更多细节

#### F.4.1 指针网络的详细信息

指针网络最初由（Vinyals 等人，2015年）提出，用于直接求解特定的组合优化问题，例如旅行商问题。指针网络架构如图5所示。指针网络由长短期记忆编码器、长短期记忆解码器以及注意力机制组成，其中注意力机制用作指针，从输入序列中选择一个元素作为输出（Vinyals 等人，2015年）。具体而言，我们参照 Bello* 等人（2017年）实现了指针网络。有关指针网络的实现细节，请参阅（Bello* 等人，2017年）。

我们使用的指针网络与（Bello* 等人，2017年）使用的指针网络之间的主要区别在于：我们使用指针网络从输入序列中选择有序子集，而（Bello* 等人，2017年）使用指针网络输出输入序列的排列。

#### F.4.2 HEM-RATIO 与 HEM w/o H 的区别

**HEM w/o H 的详细信息**为了实现 HEM w/o H，我们在每个输入序列中添加一个结束标记，即一个值全为 1 的十三维张量。结束标记位于输入序列的末尾。当 HEM w/o H 的解码器输出结束标记时，解码便结束。也就是说，HEM w/o H 可以通过预测当前解码步骤是否输出结束标记，隐式地预测应选择的割的数量。

HEM-ratio 和 HEM w/o H 的策略网络本质上都是指针网络（Vinyals 等人，2015年），属于序列模型的一种变体。下面介绍 HEM-ratio 与 HEM w/o H 之间的主要区别。HEM w/o H 预测一个语言任务中使用的结束标记（Sutskever等人，2014年；Vaswani 等人，2017年），以隐式确定应选择的割数量。相比之下，HEM-ratio 始终选择固定比例的割，即始终结束解码

表 9：在简单、中等和困难数据集上进行策略评估，评估指标为节点总数和原始-对偶间隙。最佳表现以粗体标出。

| | 集合覆盖 | | 最大独立集 | | 多重背包 | |
|---|---|---|---|---|---|---|
| 方法 | 节点 | PD间隙 | 节点 | PD间隙 | 节点 | PD间隙 |
| NoCuts | 189.44 (423.68) | 0.00 (0) | 2170.66 (4054.09) | 0.00 (0) | 16945.58 (41242.04) | 0.00 (0.000128) |
| 默认 | 116.77 (420.09) | 0.00 (0) | 588.23 (1916.99) | 0.00 (0) | 16949.88 (41297.50) | 0.00 (0.000128) |
| 随机 | 95.70 (285.05) | 0.00 (0) | 1416.96 (3820.01) | 0.00 (0) | 21463.16 (59411.07) | 0.00 (0.000361) |
| NV | 199.70 (436.25) | 0.00 (0) | 1618.42 (3089.12) | 0.00 (0) | 20673.32 (62526.13) | 0.00 (0.00022) |
| Eff | 194.17 (439.35) | 0.00 (0) | 1575.88 (2742.66) | 0.00 (0) | 14909.93 (28575.12) | 0.00 (0) |
| SBP | 1.16 (2.04) | 0.00 (0) | 698.41 (2869.65) | 0.00 (0) | 13537.59 (22527.45) | 0.00 (0) |
| HEM（本文方法） | **1.11 (1.46)** | **0.00 (0)** | **311.64 (1309.94)** | **0.00 (0)** | **10463.85 (18491.25)** | **0.00 (0)** |

| | MIK | | CORLAT | | 负载均衡 | |
|---|---|---|---|---|---|---|
| 方法 | 节点 | PD间隙 | 节点 | PD间隙 | 节点 | PD间隙 |
| NoCuts | 395290.17 (120526.21) | 0.09 (0.018) | 97098.72 (119287.56) | 2.000000e+18 (1.4E+19) | 108.20 (56.28) | 0.94 (0.12) |
| 默认 | 224746.60 (179450.80) | 0.02 (0.033) | 70215.01 (113515.62) | 2.666667e+18 (1.61E+19) | **65.22 (55.34)** | 0.43 (0.073) |
| 随机 | 403618.10 (115802.29) | 0.07 (0.035) | 74149.90 (109043.28) | 2.000000e+18 (1.4E+19) | 107.05 (68.05) | 0.79 (0.133) |
| NV | 397025.47 (106676.73) | 0.08 (0.028) | 77943.73 (111355.03) | 4.666667e+18 (2.11E+19) | 81.91 (38.55) | 0.82 (0.11) |
| Eff | 406832.83 (113363.27) | 0.07 (0.033) | 95956.53 (120858.92) | 5.333333e+18 (2.25E+19) | 86.22 (45.35) | 0.82 (0.11) |
| SBP | 417070.37 (133702.50) | 0.07 (0.030) | 62297.99 (109664.21) | 6.666667e+17 (8.14E+18) | 138.37 (53.79) | 0.65 (0.06) |
| HEM（我们的方法） | **220547.93 (172537.94)** | **0.02 (0.028)** | **51929.71 (100973.74)** | **0.02 (0.051)** | 74.47 (60.98) | **0.42 (0.072)** |

| | 匿名 | | MIPLIB mixed neos | | MIPLIB mixed supportcase | |
|---|---|---|---|---|---|---|
| 方法 | 节点 | PD间隙 | 节点 | PD间隙 | 节点 | PD间隙 |
| NoCuts | 20110.58 (14167.51) | 8.333333e+18 (2.76E+19) | 169348.83 (117745.23) | 2.500000e+19 (4.33E+19) | 5971.00 (8679.98) | 9.19 (19.29) |
| 默认 | 20025.78 (13056.50) | 8.333333e+18 (2.76E+19) | 161366.25 (107049.96) | 2.500000e+19 (4.33E+19) | **4055.04 (7404.61)** | 11.73 (31.52) |
| 随机 | 19824.48 (12366.55) | 1.000000e+19 (3E+19) | **143930.17 (95296.41)** | 2.500000e+19 (4.33E+19) | 4656.13 (7745.46) | 6.05 (14.69) |
| NV | **19313.33 (12391.94)** | 8.333333e+18 (2.76E+19) | 150046.50 (94801.62) | 2.500000e+19 (4.33E+19) | 4986.21 (9024.19) | 2.78 (10.08) |
| Eff | 19526.23 (12116.80) | 6.666667e+18 (2.49E+19) | 144128.83 (93404.76) | 2.500000e+19 (4.33E+19) | 4450.88 (7845.48) | 5.58 (15.40) |
| SBP | 19351.67 (12337.81) | 6.666667e+18 (2.49E+19) | 177736.25 (122005.81) | 2.500000e+19 (4.33E+19) | 5618.08 (9765.43) | 0.14 (0.26) |
| HEM（我们的方法） | 20191.28 (13219.21) | **1.666667e+18 (1.28E+19)** | 177735.83 (129020.42) | **2.500000e+19 (4.33E+19)** | 4844.88 (9996.16) | **0.14 (0.25)** |

在预先确定的位置。因此，HEM w/o H 可以学习应选择的割的数量，但 HEM-ratio 不行。

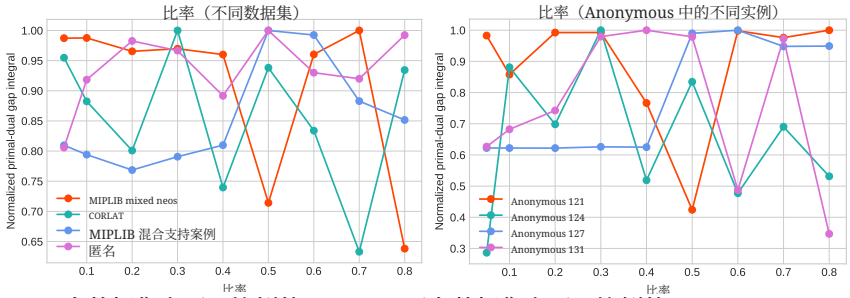
### F.4.3 关于 HEM 的更多讨论

在本节中，我们将详细介绍 HEM 的一些其他优势。(1) 受分层强化学习（Sutton 等人，1999b；Nachum 等人，2018年）的启发，HEM 利用了割选择任务的层次结构，这对于复杂决策任务中的高效探索非常重要。(2) 以往的方法（Tang 等，2020；Huang 等，2022）通常通过黑箱优化方法（如进化策略（Salimans 等人，2017年））训练割选择策略。相比之下，HEM 是可微的，我们通过基于梯度的算法训练 HEM，其样本效率高于黑箱优化方法（Sutton 和 Barto，2018年；Schulman 等人，2015年）。尽管我们可以利用混合整数线性规划求解器离线生成尽可能多的训练样本，但较高的样本效率仍然十分重要，因为在实践中生成样本可能极其耗时。

## G 更多结果

### G.1 更多动机结果

**比例很重要**。为评估选定割的比例对求解混合整数线性规划问题的影响，我们重点考察采用不同选定割比例的归一化违反度（NV）割选择方法。（1）首先，我们在四个数据集上评估分别选择 5%，10%，20%，30%，40%，50%，60%，70% 和 80% 候选割的 NV 方法。图 6a 的结果显示，在 CORLAT 和 MIPLIB mixed neos 上，随着比例增大，NV 的求解器性能有所提升。这些结果表明，不同数据集上带来更佳求解器性能的比例各不相同，这意味着学习依赖数据集的比例十分重要。（2）随后，我们在 Anonymous 数据集的四个实例上评估分别选择 5%，10%，20%，30%，40%，50%，60%，70% 和 80% 候选割的 NV 方法。图 6b 的结果显示，在 Anonymous 121 和 Anonymous 131 上，随着比例增大，NV 的求解器性能有所提升。这些结果表明，即使在同一数据集的不同实例上，带来更佳求解器性能的比例也各不相同，这意味着学习依赖实例的比例同样十分重要。




(a) 四个数据集上不同比例的 NV。(b) 匿名数据集上不同比例的 NV。

图 6：我们采用归一化违反度（NV）规则（Huang 等，2022）。(c) 中的结果表明，在四个数据集上，给定不同比例的 NV 会产生不同的归一化 PD 积分。(d) 中的结果表明，在同一数据集的不同实例上，不同比例的 NV 会产生不同的归一化 PD 积分。

### G.2 主要评估的更多结果

本节给出主要评估的更多结果。表 9 展示了 HEM 和基线方法在节点总数（节点数）和原始-对偶间隙（PD间隙）方面的表现。（1）**简单数据集**。在简单数据集上，HEM 和大多数基线方法都能在时间限制内找到最优解，因为 PD间隙收敛至零。此外，在简单数据集上，HEM 在节点数方面显著优于所有基线方法。（2）**中等和困难数据集**。就 PD间隙而言，HEM 在中等和困难数据集上优于所有基线方法，尤其是在 CORLAT 上。然而，对于以下两个原因，节点数指标无法很好地区分中等和困难数据集上不同割选择方法的表现。首先，每个节点上线性规划松弛的求解时间各不相同，因此节点数无法直接决定求解时间（Huang 等，2022）。其次，对于在时间限制内未能求解的实例，节点数并非恰当的指标，因为它无法评估求解过程的质量。

### G.3 消融研究的更多结果

本节将补充正文中消融研究的更多结果。

#### G.3.1 HEM-RATIO 与 SBP 的深入分析

我们对 HEM-ratio 在若干具有挑战性的混合整数线性规划问题基准测试上表现不如 SBP 的原因进行了探讨。从根本上说，HEM-ratio 将割选择任务建模为序列建模问题，这相比 SBP 有两大优势：序列模型不仅能够捕捉潜在的顺序信息，还能够捕捉割之间的交互。然而，训练序列模型比训练评分函数更困难，因为序列模型旨在学习复杂得多的任务。具体而言，评分函数旨在为每个割评分，而序列模型旨在对选定割的联合概率进行建模。后者是一项更具挑战性的学习任务。此外，我们采用强化学习范式而非监督学习来训练模型，这使训练过程更加不稳定。因此，序列模型可能会遭遇低效探索，并陷入局部最优。由此，HEM-ratio 的表现可能不如 SBP，尤其是在具有挑战性的混合整数线性规划问题基准测试上。

#### G.3.2 各组成部分的贡献

为了解 HEM 各组成部分的贡献，我们补充了 HEM 及不含高层模型的 HEM 在集合覆盖、多重背包、MIK、负载均衡、匿名和 MIPLIB mixed supportcase 上的更多结果。表10中的结果表明，在若干具有挑战性的数据集上，HEM 在求解时间、原始-对偶间隙和原始-对偶间隙积分方面优于 HEM w/o H，这证明了我们提出的高层模型的重要性。此外，HEM w/o H 在求解时间、原始-对偶

表10：HEM 与不含高层模型的 HEM 在更多数据集上的比较。

| | 集合覆盖 | | | | 多重背包 | | | |
|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | 节点 | PD间隙 | PD 积分 | 时间（秒） | 节点 | PD间隙 | PD 积分 |
| NoCuts | 6.31 (4.61) | 189.44 (423.68) | 0.00 (0) | 56.99 (38.89) | 9.88 (22.24) | 16945.58 (41242.04) | 0.00 (0.000128) | 16.41 (14.16) |
| 默认 | 4.41 (5.12) | 116.77 (420.08) | 0.00 (0) | 55.63 (42.21) | 9.90 (22.24) | 16949.87 (41297.49) | 0.00 (0.000128) | 16.46 (14.25) |
| SBP | 1.91 (0.36) | 1.16 (2.04) | 0.00 (0) | 38.96 (8.66) | 7.74 (12.36) | 13537.58 (22527.45) | 0.00 (0) | 16.45 (16.62) |
| HEM w/o H | **1.84 (0.31)** | 1.18 (2.03) | 0.00 (0) | **37.69 (8.33)** | 7.36 (12.90) | 12418.85 (23256.47) | 0.00 (0) | 14.01 (10.94) |
| HEM（本文方法） | 1.85 (0.31) | **1.09 (1.46)** | **0.00 (0)** | 37.92 (8.46) | **6.13 (9.61)** | **10463.84 (18491.25)** | **0.00 (0)** | **13.63 (9.63)** |

| | MIK | | | | 负载均衡 | | | |
|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | 节点 | PD间隙 | PD 积分 | 时间（秒） | 节点 | PD间隙 | PD 积分 |
| NoCuts | 300.01 (0.009) | 395290.16 (120526.21) | 0.09 (0.018) | 2355.87 (996.08) | 300.00 (0.12) | 108.20 (56.28) | 0.94 (0.12) | 14853.77 (951.42) |
| 默认 | 179.62 (122.36) | 224746.60 (179450.80) | 0.02 (0.033) | 844.40 (924.30) | 300.00 (0.06) | **65.22 (55.34)** | 0.43 (0.07) | 9589.19 (1012.95) |
| SBP | 286.07 (41.81) | 417070.36 (133702.49) | 0.07 (0.0295) | 2053.30 (740.11) | 300.00 (0.10) | 138.37 (53.79) | 0.65 (0.06) | 12535.30 (741.43) |
| HEM w/o H | 218.87 (115.97) | 297058.97 (189410.43) | 0.04 (0.039) | 1321.63 (1165.23) | 300.03 (0.04) | 70.93 (60.36) | **0.42 (0.07)** | **9475.16 (1005.81)** |
| HEM（本文方法） | **176.12 (125.18)** | **220547.93 (172537.94)** | **0.02 (0.028)** | **785.04 (790.38)** | **300.00 (0.04)** | 74.47 (60.98) | **0.42 (0.07)** | **9496.42 (1018.35)** |

| | 匿名 | | | | MIPLIB mixed supportcase | | | |
|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | 节点 | PD间隙 | PD 积分 | 时间（秒） | 节点 | PD间隙 | PD 积分 |
| NoCuts | 246.22 (94.90) | 20110.58 (14167.51) | 8.333333e+18 (2.76E+19) | 18297.30 (9769.42) | 170.00 (131.60) | 5971.00 (8679.98) | 9.19 (19.29) | 9927.96 (11334.07) |
| 默认 | 244.02 (97.72) | 20025.78 (13056.50) | 8.333333e+18 (2.76E+19) | 17407.01 (9736.19) | 164.61 (135.82) | 4055.04 (7404.61) | 11.73 (31.52) | 9672.34 (10668.24) |
| SBP | 245.71 (92.46) | **19351.66 (12337.81)** | 6.666667e+18 (2.49E+19) | 18188.63 (9651.85) | 165.61 (135.25) | 5618.08 (9765.43) | 0.14 (0.26) | 7408.65 (7903.47) |
| HEM w/o H | 251.00 (88.45) | 21192.40 (16436.32) | 5.000000e+18 (2.18E+19) | 17226.84 (9553.90) | **156.96 (133.35)** | **3779.83 (7465.75)** | 0.56 (1.43) | 7709.19 (8655.48) |
| HEM（本文方法） | **241.68 (97.23)** | 20191.28 (13219.21) | **1.666667e+18 (1.28E+19)** | **16077.15 (9108.21)** | 162.96 (138.21) | 4844.87 (9996.15) | **0.14 (0.25)** | **6874.80 (6729.97)** |

表11：在更多数据集上比较 HEM、HEM-ratio 和 HEM-ratio-order。

| | 集合覆盖 | | | | 多重背包 | | | |
|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | 节点 | PD间隙 | PD 积分 | 时间（秒） | 节点 | PD间隙 | PD 积分 |
| NoCuts | 6.31 (4.61) | 189.44 (423.68) | 0.00 (0) | 56.99 (38.89) | 9.88 (22.24) | 16945.58 (41242.04) | 0.00 (0.000128) | 16.41 (14.16) |
| 默认 | 4.41 (5.12) | 116.77 (420.08) | 0.00 (0) | 55.63 (42.21) | 9.90 (22.24) | 16949.87 (41297.49) | 0.00 (0.000128) | 16.46 (14.25) |
| SBP | 1.91 (0.36) | 1.16 (2.04) | 0.00 (0) | 38.96 (8.66) | 7.74 (12.36) | 13537.58 (22527.45) | 0.00 (0) | 16.45 (16.62) |
| HEM-ratio-order | 2.11 (0.38) | 1.11 (1.29) | 0.00 (0) | 42.01 (9.88) | 10.46 (29.74) | 18434.20 (59254.10) | 0.00 (0.000128) | 16.92 (18.19) |
| HEM-ratio | 2.10 (0.40) | 1.11 (1.18) | 0.00 (0) | 41.95 (9.82) | 7.63 (12.64) | 13307.73 (24339.3) | 0.00 (0) | 16.19 (15.16) |
| HEM（本文方法） | **1.85 (0.31)** | **1.09 (1.46)** | **0.00 (0)** | **37.92 (8.46)** | **6.13 (9.61)** | **10463.84 (18491.25)** | **0.00 (0)** | **13.63 (9.63)** |

| | MIK | | | | 负载均衡 | | | |
|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | 节点 | PD间隙 | PD 积分 | 时间（秒） | 节点 | PD间隙 | PD 积分 |
| NoCuts | 300.01 (0.009) | 395290.16 (120526.21) | 0.09 (0.018) | 2355.87 (996.08) | 300.00 (0.12) | 108.20 (56.28) | 0.94 (0.12) | 14853.77 (951.42) |
| 默认 | 179.62 (122.36) | 224746.60 (179450.80) | 0.02 (0.033) | 844.40 (924.30) | 300.00 (0.06) | **65.22 (55.34)** | 0.43 (0.07) | 9589.19 (1012.95) |
| SBP | 286.07 (41.81) | 417070.36 (133702.49) | 0.07 (0.0295) | 2053.30 (740.11) | 300.00 (0.10) | 138.37 (53.79) | 0.65 (0.06) | 12535.30 (741.43) |
| HEM-ratio-order | 282.92 (42.07) | 417397.10 (132077.3) | 0.06 (0.034) | 2072.69 (849.16) | 300.11 (0.11) | 145.18 (56.52) | 0.65 (0.061) | 12368.12 (726.77) |
| HEM-ratio | 283.75 (39.87) | 401540.97 (131295.8) | 0.07 (0.034) | 1869.66 (978.85) | 300.10 (0.098) | 148.92 (59.12) | 0.65 (0.058) | 12410.84 (715.44) |
| HEM（本文方法） | **176.12 (125.18)** | **220547.93 (172537.94)** | **0.02 (0.028)** | **785.04 (790.38)** | **300.00 (0.04)** | 74.47 (60.98) | **0.42 (0.07)** | **9496.42 (1018.35)** |

| | 匿名 | | | | MIPLIB 混合 supportcase | | | |
|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | 节点 | PD间隙 | PD 积分 | 时间（秒） | 节点 | PD间隙 | PD 积分 |
| NoCuts | 246.22 (94.90) | 20110.58 (14167.51) | 8.33e+18 (2.76E+19) | 18297.30 (9769.42) | 170.00 (131.60) | 5971.00 (8679.98) | 9.19 (19.29) | 9927.96 (11334.07) |
| 默认 | 244.02 (97.72) | 20025.78 (13056.50) | 8.33e+18 (2.76E+19) | 17407.01 (9736.19) | 164.61 (135.82) | **4055.04 (7404.61)** | 11.73 (31.52) | 9672.34 (10668.24) |
| SBP | 245.71 (92.46) | **19351.66 (12337.81)** | 6.67e+18 (2.49E+19) | 18188.63 (9651.85) | 165.61 (135.25) | 5618.08 (9765.43) | 0.14 (0.26) | 7408.65 (7903.47) |
| HEM-ratio-order | 245.45 (94.99) | 20495.80 (12472.44) | 5.00e+18 (2.18E+19) | 16496.06 (9282.15) | 169.45 (132.55) | 6252.63 (9827.98) | 4.25 (15.97) | 9226.95 (9995.94) |
| HEM-ratio | 245.17 (95.21) | 20942.07 (13379.46) | 5.00e+18 (2.18E+19) | 16148.82 (9247.48) | 163.03 (137.16) | 5551.29 (10708.46) | 7.54 (21.85) | 9979.35 (11048.11) |
| HEM（本文方法） | **241.68 (97.23)** | 20191.28 (13219.21) | **1.67e+18 (1.28E+19)** | **16077.15 (9108.21)** | **162.96 (138.21)** | 4844.87 (9996.15) | **0.14 (0.25)** | **6874.80 (6729.97)** |

gap, 以及在多个具有挑战性的数据集上的原始-对偶间隙积分，证明了.....的重要性 e of o我们提出的低层模型。

### G.3.3 处理割选择中的**P1-P3** 的重要性

为了解在割选择中处理 **P1-P3** 的重要性，我们补充了 HEM、HEM-ratio 和 HEM-ratio-order 在集合覆盖、多重背包、MIK、负载均衡、匿名和 MIPLIB mixed supportcase 上的更多结果。这里我们重新说明 HEM、HEM-ratio 和 HEM-ratio- order 的含义。(1) **HEM**。HEM 同时处理割选择中的 **P1-P3** 。(2) **HEM-ratio**。为了避免学习应选取多少个割，我们移除了 HEM 的高层模型，并强制低层模型选取固定比例的割。我们将其称为 HEM-ratio。请注意，HEM-ratio 与 HEM w/o H 不同（见附录F）。HEM-ratio 处理割选择中的 **P1** 和 **P3** 。(3) **HEM-ratio-order**。 为进一步减弱选定割的顺序所产生的影响，我们按照生成割的原始索引，对 HEM-ratio 给出的选定割重新排序，并将其称为 HEM-ratio-order。HEM-ratio-order 主要处理**P1**。

表11中的结果表明：HEM-ratio-order在若干数据集上的表现优于默认方法和NoCuts，说明采用数据驱动方法解决**P1** 至关重要。就原始-对偶间隙积分而言，HEM显著优于HEM-ratio，说明解决**P2**的重要性。HEM-ratio在若干数据集上的表现优于HEM-ratio-order，说明解决**P3**的重要性。此外，就

表1 2：HEM 对超参数延迟更新频率 $d$的敏感性分析。

| | 最大独立集 | | | CORLAT | | | MIPLIBS 混合 (neos) | | |
|---|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | PD 积分 | 改进（PD积分，%） | 时间（秒） | PD 积分 | 改进（PD积分，%） | 时间（秒） | PD 积分 | 改进（PD积分，%） |
| NoCuts | 8.78 (6.66) | 71.31 (51.74) | NA | 103.30 (128.14) | 2818.40 (5908.31) | NA | 253.65 (80.29) | 14652.29 (12523.37) | NA |
| HEM（延迟=1） | 1.79 (3.65) | 16.56 (26.6) | 76.78 | 57.39 (111.76) | 1260.83 (3518.07) | 55.26 | 254.48 (78.84) | 9273.01 (12031.44) | 36.71 |
| HEM（延迟=2） | 1.76 (3.69) | 16.01 (26.21) | 77.55 | 58.31 (110.51) | 1079.99 (2653.14) | 61.68 | 248.66 (89.46) | 8678.76 (12337.00) | 40.77 |
| HEM（延迟=3） | 1.80 (3.82) | 16.91 (28.49) | 76.29 | 58.50 (108.88) | 1003.52 (2264.91) | 64.39 | 247.62 (90.73) | 8408.16 (12467.02) | 42.62 |
| HEM（延迟=4） | 1.89 (3.90) | 17.34 (28.46) | 75.68 | 69.60 (115.42) | 1309.33 (3336.59) | 53.54 | 246.93 (91.93) | 8368.46 (12489.68) | 42.89 |

表13：评估 HEM 在集合覆盖问题上的泛化能力。

| | 集合覆盖（2×） | | | 集合覆盖（4×） | | |
|---|---|---|---|---|---|---|
| 方法 | 时间（秒）↓ | 改进 ↑（时间，%） | PD 积分 ↓ | 时间（秒）↓ | 改进 ↑（时间，%） | PD 积分↓ |
| NoCuts | 82.69 (78.27) | NA | 609.43 (524.92) | 284.44 (48.70) | NA | 3215.34 (1019.47) |
| 默认 | 61.01 (78.12) | 26.22 | 494.63 (545.76) | 149.69 ( 141.92) | 47.37 | 1776.22 (1651.15) |
| 随机 | 64.44 (73.98) | 22.07 | 520.84 (489.52) | 208.12 (131.52) | 26.53 | 2528.36 (1678.66) |
| NV | 92.05 (80.11) | -11.32 | 725.53 (541.68) | 286.10 (45.47) | -0.58 | 3422.46 (1024.19) |
| Eff | 92.32（79.33） | -11.64 | 733.72（538.60） | 286.20（45.04） | -0.62 | 3437.06（1043.44） |
| SBP | 3.52（1.36） | 95.74 | 92.89（25.83） | 7.62 (6.46) | 97.32 | 256.79 (145.92) |
| HEM（本文方法） | **3.33 (0.47)** | **95.97** | **89.24 (14.26)** | **7.40 (5.03)** | **97.40** | **250.83 (131.43)** |

求解时间和原始-对偶间隙积分而言，除集合覆盖和MIPLIB混合支持案例外，HEM-ratio在其余六个数据集上均优于SBP。这表明，与将割选择建模为评分任务相比，将其建模为序列到序列学习问题更具优势。然而，在集合覆盖和MIPLIB混合支持案例上，HEM-ratio和HEM-ratio-order的表现略逊于SBP。一个可能的原因是，相较于训练多层感知机，训练序列模型更加困难，因此序列模型可能会遭遇低效探索并陷入局部最优。详细分析请参见附录 G.3.1。

#### G.3.4 敏感性分析

此外，我们还进行了消融研究，以测试 HEM 对超参数延迟更新频率 $d$ 的敏感性。表12中的结果表明，对于 $d$，HEM 在最大独立集、CORLAT 和 MIPLIB mixed neos 上均能取得可比的性能，范围相当广泛。此外，我们强调并未调节超参数 $d$。如表12所示，$d = 3$ 和$d = 4$ 分别在 CORLAT 和 MIPLIB mixed neos 上表现最佳。然而，在正文的所有实验中，我们始终简单地将其设为 $d = 2$。

### G.4 泛化的更多结果

这里我们给出在集合覆盖上的更多泛化实验结果。在集合覆盖上，我们使用比训练期间见过的实例大两倍和四倍的实例对 HEM 进行测试。表13中的结果表明，HEM 能够很好地泛化到显著大于训练期间见过的实例。具体而言，与所有基于规则的基线方法相比，HEM 在时间方面至少实现了70%的改进。此外，SBP 也能很好地泛化到大型实例，表明 SBP 是一种强大的基线方法。

### G.5 更多可视化结果

在本部分中，我们提供了更多关于集合覆盖和MIK的可视化结果。对于MIK，我们在一个随机抽取的实例上可视化HEM-ratio和SBP选定的割。我们对选定割进行主成分分析（Mohri 等人，2018年），将割特征降至二维空间。彩色点表示降维后的割特征。为了展示选定割的多样性，我们用虚线连接x、y坐标最小和最大的点。图7中的结果仍然表明，在MIK上，HEM-ratio选出的割比SBP多样得多。然而，在集合覆盖问题上，HEM-ratio的表现不如SBP（详细分析见附录 G.3.1）。因此，我们在一个从集合覆盖数据集中随机抽取的实例上可视化HEM和SBP选定的割。尽管HEM能够学习应选定的割数，但我们发现HEM选出的割远少于SBP。具体而言，HEM选出25个割，而SBP选出158个割。有趣的是，图7中的结果表明，SBP选出的158个割彼此相似且得分较高，而HEM选出的割比SBP多样得多。这些结果表明，HEM倾向于选择能够相互良好补充的割。

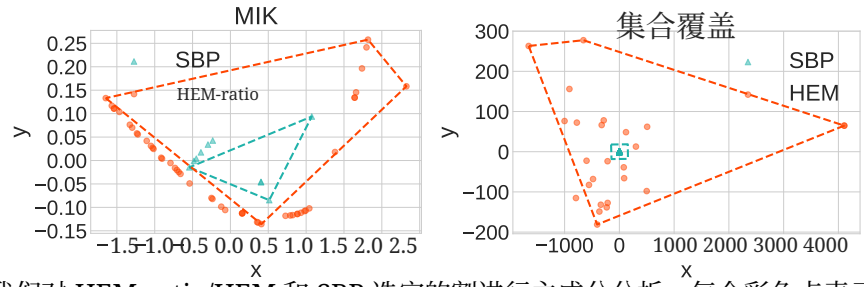



图 7：我们对 HEM-ratio/HEM 和 SBP 选定的割进行主成分分析。每个彩色点表示一个降维后的割特征。为了直观展示选定割的多样性，我们用虚线连接 x、y 坐标最小和最大的点。

表14：在 MIPLIB mixed neos 和 MIPLIB mixed supportcase 上进行策略评估，时间限制为三小时。

| | MIPLIB mixed neos | | | | | MIPLIB mixed supportcase | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 方法 | 时间（秒） | 节点 | PD间隙 | PD 积分 | 改进（PD积分，%） | 时间（秒） | 节点 | PD间隙 | PD 积分 | 改进（PD积分，%） |
| NoCuts | 8,126.15 (4631.32) | 5,058,282.42 (5067358) | 0.97 (1.14) | 369,180.71 (368274) | NA | 3,616.09 (4703.68) | 258,543.71 (692821) | 0.00 (0.0096) | 37,858.65 (59578.6) | NA |
| 默认 | 8,129.27（4625.92） | 4,800,372.67 (4815222) | 0.94 (1.07) | 392,324.09 (393266) | -6.27 | 3,997.69 (4720.2) | 188,844.71 (599665) | 0.01 (0.012) | 56,375.64 (106694) | -48.91 |
| 随机 | 8,125.44（4632.53） | 5,107,627.25 (5114152) | 0.95 (1.09) | 395,694.74 (393445) | -7.18 | 2,605.78 (3993.84) | **8,458.96 (14785.9)** | 0.01 (0.029) | 44,222.39（70937.6） | -16.81 |
| NV | 8,133.64 (4618.37) | 4,930,498.92 (4930971) | 0.91 (1.03) | 326,315.04 (391163) | 11.61 | **2,561.10 (3980.44)** | 34,851.08 (142069) | 0.00 (0.0095) | 27,908.00（37742.2） | 26.28 |
| Eff | 8,130.78（4623.31） | 4,788,568.75（4804443） | 0.94 (1.07) | 395,640.44 (393572) | -7.17 | 3,105.98（4308.57） | 182,187.63 (543167) | 0.01 (0.028) | 39,932.37（58307.3） | -5.48 |
| SBP | 8,130.63（4626.62） | 4,763,378.00 (4829037) | 0.91 (1.03) | 388,564.35 (398176.3) | -5.25 | 3,014.63 (4082.19) | 146,180.96（543662.45） | 0.00 (0.0084) | 24,447.37（55984.9） | 35.42 |
| HEM（随机种子） | **8,124.32 (4634.54)** | **4,748,599.50 (4816327)** | **0.78 (0.82)** | **194,557.87 (345035)** | **47.30** | 3,465.48（4485.11） | 52,750.08 (140410) | **0.00 (0.0087)** | **17,885.98 (24045.9)** | **52.76** |

G.6 三小时时间限制下的评估

在本节中，我们旨在评估HEM能否很好地泛化到求解时间限制长得多的问题。具体而言，我们在时间限制为三小时的条件下，在两个极具挑战性的MIPLIB数据集上评估HEM。请注意，我们仍以300秒的时间限制训练HEM，而测试HEM时的时间限制为三小时。表14中的结果表明，HEM仍显著优于所有基线方法，尤其是在MIPLIB mixed neos和MIPLIB mixed supportcase上的原始-对偶间隙积分方面。就原始-对偶间隙而言，HEM也优于基线方法。此外，在MIPLIB mixed neos上，HEM的求解时间优于基线方法；但在MIPLIB mixed supportcase上，HEM的求解时间表现较差。有趣的是，原始-对偶间隙积分与求解时间并不总是一致。我们强调，我们使用负原始-对偶间隙积分奖励进行训练。若要进一步提升HEM在求解时间方面的表现，可以将奖励设为负求解时间，而非负原始-对偶间隙积分。

G.7 训练曲线

本节展示 HEM 在全部九个数据集上的训练曲线。图8中的结果表明，随着训练轮次的增加，我们学习得到的策略在求解时间或原始-对偶间隙积分方面的表现不断提升，证明了学习过程的有效性。

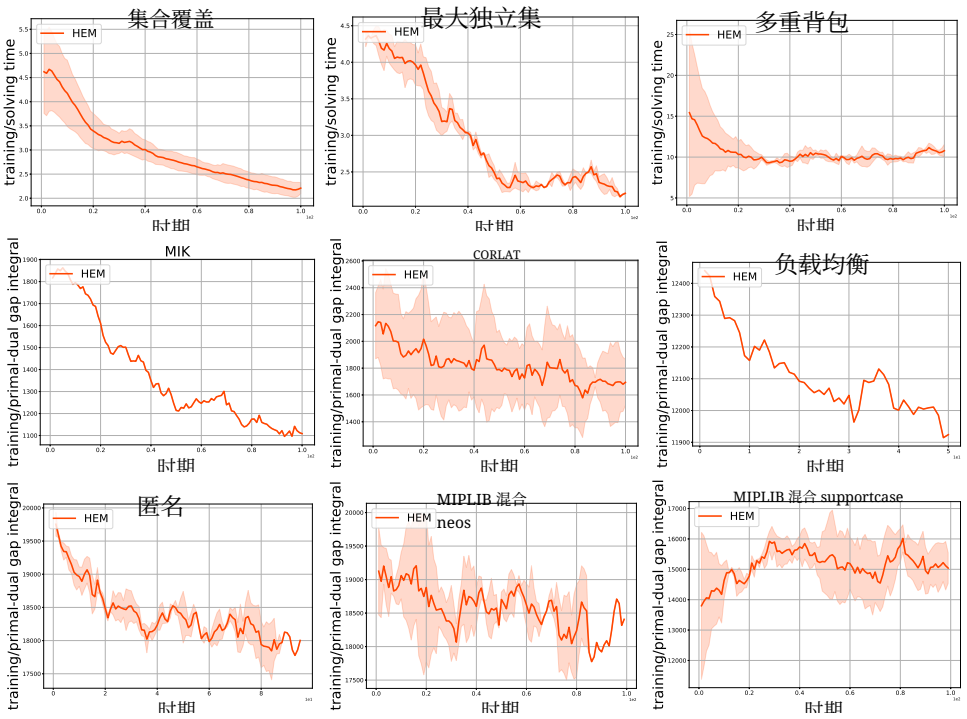



图8：HEM 在全部九个数据集上的训练曲线。横轴表示训练轮次。纵轴表示简单数据集上的平均求解时间，以及其他数据集上的原始-对偶间隙积分。实线表示三个随机种子的均值，阴影区域表示标准差。为便于观察，我们对曲线进行了平滑处理。