arXiv:2305.04397v1 [cs.MA] 8 May 2023

基于 GPU-CPU混合加速的多目标任务分配与多智能体规划*

Thomas Robinson[0000−0002−6150−2587] 和 Guoxin Su[0000−0002−2087−4894]

澳大利亚新南威尔士州伍伦贡大学，邮编 2522
tmr463@uowmail.edu.au guoxin@uow.edu.au

摘要。 在多智能体系统中，为一组任务分配资源并进行规划是一项重要问题。一个常见的瓶颈是可扩展性，因为通常多智能体模型的规模会随着智能体数量的增加而呈指数级增长。我们研究多目标约束下随机任务分配与多智能体规划的结合，并表明该问题可以分散到各个独立的智能体-任务模型中。我们提出一种面向点的帕累托计算算法，用于检查与形式化问题中给定的成本阈值和概率阈值相对应的点是否可行。如果给定点不可行，我们的算法会找到距离该点最近的帕累托最优点。据我们所知，我们首次提出了同时利用 GPU 和多核加速的多目标模型检测框架。该框架将 CPU 和 GPU 设备的管理视为并行计算中的负载均衡问题。实验结果表明，与顺序计算相比，并行化能够显著缩短运行时间。

关键词： 多智能体系统 · 任务分配 · 规划 · 概率模型检测 · 图形处理器与多核加速

1 引言

马尔可夫决策过程（MDP） [26] 是随机环境中进行多智能体规划的基本模型，在这种环境下，智能体在某个状态下采取的行动可能产生不确定的结果。多智能体任务分配与规划旨在使一组智能体能够彼此分配任务，并开展规划与执行。可扩展性是这类问题普遍面临的瓶颈，因为通常情况下，随着系统中智能体数量线性增加，用于建模多智能体系统（MAS）的MDP规模会呈指数级增长 [5]。

概率模型检验（PMC）是一种验证技术，用于对现实世界中的随机系统的正确性建立严格保证 [1]。PMC

*本研究部分经费由澳大利亚国防科学与技术集团通过“2022年决策人工智能倡议”计划提供。

提供了计算马尔可夫决策过程的可达性奖励最优值，以及计算马尔可夫决策过程满足以线性时序逻辑（LTL）形式化的性质的最优概率的方法。LTL 的一个片段称为余安全线性时序逻辑，可用确定性有限状态自动机（DFA）表示 [19], ，因此适合用于描述必须在有限时间内完成的任务。在多智能体规划中，有限时间内执行任务十分重要，因为我们通常希望重复利用智能体来执行后续任务。

在实际应用中，智能体的协调通常会涉及相互冲突的解决方案，而多智能体系统必须满足多目标。例如，智能体可能需要在执行时间和能耗之间取得平衡。当需要同时验证多个目标时，我们需要采用多目标马尔可夫决策过程（MOMDP） [30], 其奖励结构指定的是奖励向量（而非标量）。MOMDP 的解空间是一个凸多胞体 [14,10],，因此 MOMDP 模型检测问题具有可处理性。目前，MOMDP 模型检测中考虑了三类查询 [12]：可达性查询是最基本的查询，它询问是否存在一个调度器能够满足所有目标阈值；数值查询是第一类查询的数值变体，用于计算在满足其他所有目标阈值的同时某一目标的最优值；帕累托查询是代价最高的查询，用于近似计算所有目标的帕累托曲线。

经典指派问题旨在求得一种分配，即从任务到智能体的一一映射，使指派奖励最大化。多目标指派问题则是要确定一种分配，使向量化的指派奖励达到帕累托最优。经典指派问题可以高效求解（例如，使用匈牙利算法 [18]），但多目标指派问题要困难得多 [33]。多目标随机指派（MORA）问题旨在求得一种在各个分配上的随机化分布，使期望指派奖励达到帕累托最优。

针对非随机智能体模型（即转移系统） [32] 和随机智能体模型（即马尔可夫决策过程） [11], 已有研究探讨了（单目标）任务分配与多智能体规划的结合。本文将多目标马尔可夫决策过程模型检测扩展到多智能体系统（MAS）中的多目标随机分配与规划（MORAP）场景，并提出一种采用 GPU-CPU 混合加速的新型实现。本文的主要贡献如下：

– 我们证明了形式化问题（MORAP）的凸性，并表明解决该问题的一种实用方法可以依赖于分散式模型，从而避免随着智能体和任务数量增加而导致的模型规模呈指数增长。– 我们的主要算法是一种新的面向点的帕累托计算方法，用于补充现有的可达性查询和帕累托查询 [14]。对于一个对应于成本和概率阈值的给定点，我们的算法能够找到一个对 MORAP问题可行的点，并根据一般向量范数找出与给定点距离最近的点。– 据我们所知，我们提供了首个同时利用 GPU 和多核加速的多目标模型检测框架。我们的框架将 CPU 和 GPU 设备作为负载均衡

并行计算问题进行管理。我们在一个智能仓库示例中评估了该实现的性能。

本文其余部分安排如下：第 2 节介绍问题的预备知识；第 3 节阐述问题的求解方法、模型构建和算法；第 4 节详述混合实现与并行架构；第 5 节分析我们的方法的性能；第 6 节介绍相关工作；最后，第 7 节总结全文。定理的形式化证明见本文长篇版本的附录 [28]。

2 预备知识

确定性有限自动机。确定性有限状态自动机（确定性有限状态自动机） $\mathcal{A}$由元组（Q, q_0, Q_F, Σ, δ）定义，其中（i） Q 是位置集合，（ii） $q_0 \in Q$是初始位置，（iii） $Q_F \subseteq Q$ 是接受位置集合，（iv） $\Sigma = 2^{AP}$（其中 AP 是非空的原子命题集合）是字母表，以及（v）δ： $S \times \Sigma \to S$是转移函数。如果对于某个 $W \subseteq AP$， δ（q， W） $= q'$，则称 q 为 q' 的前驱， q' 为 q的后继。令 pre（q）和 suc（q）分别表示 q 的前驱集合和后继集合。如果 suc（q） $= \{q\}$，则位置 q 是一个汇点。本文只需考虑接受位置为汇点的确定性有限状态自动机。 $\mathcal{A}$ 中的一次运行是位置序列 q_1，.....， q_m，满足对于所有 $1 \le i \le m-1$，$q_{i+1} \in$ suc（q_i）。如果从 q 出发不存在通往任何 $q' \in Q_F$ 的运行，则称 q 为陷阱。令 Q_R 表示 $\mathcal{A}$中的陷阱集合。

余安全线性时序逻辑。线性时序逻辑是一种紧凑表示线性时间性质的方式。线性时序逻辑的语法为 $\varphi ::= \top \mid$ a $\mid \neg\varphi \mid \varphi \wedge \varphi \mid$ X $\varphi \mid \varphi$ U φ，其中 a $\in AP$。算子 X 和 U 分别表示“下一个”和“直到”。令 F $\varphi := \top$U φ，且 G $\varphi := \neg$F $\neg\varphi$。语义关系 $\sigma \models \varphi$ where $\sigma \in \Sigma^\omega$ is standard where Σ^ω denotes the set of all infinite words over Σ。我们关注线性时序逻辑公式的余安全片段。非正式地说，若任意 σ（满足 $\sigma \models \varphi$）都包含某个良好前缀（它在某个确定性有限状态自动机中是可接受的） $pref_{good}$（φ），则 φ 是余安全的，记作 acc（$\mathcal{A}$）。从句法上说，任何仅包含时序算子 X（下一个）、U（直到）和 F（最终），且处于正范式（PNF）的线性时序逻辑公式都是余安全的。语义层面的形式化刻画见 [28] 的附录。

马尔可夫决策过程。一个（带标记的）马尔可夫决策过程由元组 $\mathcal{M} = (S, s_0, A, P, L)$ 给出，其中 (i) S 是有限的非空状态空间，(ii) $s_0 \in S$ 是初始状态，(iii) A 是动作集合，(iv) P: $S \times A \times S \to [0,1]$ 是转移概率函数，满足 $\sum_{s' \in S} P(s,a,s') \in \{0,1\}$，以及 (v) L: $S \to \Sigma$ 是标记函数。令 $A(s) = \{a \in A \mid \sum_{s' \in S} P(s,a,s') = 1\}$，即， $A(s)$ 是 s处启用的动作集合。 $\mathcal{M}$ 的大小为 $|\mathcal{M}| = |S| + |P|$，其中 $|P| = |\{(s,a,s') \in S \times A \times S \mid P(s,a,s') > 0\}|$。 $\mathcal{M}$ 的奖励函数或奖励结构是一个函数ρ： $\{(s,a) \in S \times A \mid a \in A(s)\} \to \mathbb{R}$。我们写作 $\mathcal{M}[\rho]$，以明确表示 $\mathcal{M}$的奖励结构 ρ。一个（无限）路径 π 是满足以下条件的序列 $s_1 a_1 s_2 a_2 \ldots$：对于所有 $i \ge 1$， $P(s_i, a_i, s_{i+1}) > 0$。令 $L(\pi)$ 表示单词 $L(s_1)$ $L(s_2)$ $\ldots \in$。

Σ^ω。令 IPath 为 $\mathcal{M}$ 中路径的集合，IPath(s) 为 IPath 中包含起源于 s 的路径的子集。A 上的概率分布集合记为 $Dist(A)$。对于所有 $s \in S$，$\mathcal{M}$ 的无记忆调度器（简称调度器）是一个映射 $\mu : s \mapsto Dist(A(s))$。若 μ 是一个简单（或纯）调度器，则对于每个 $s \in S$，存在某个 $a \in A(s)$，使得 $\mu(s)(a) = 1$。调度器（分别为简单调度器）的集合记为 $Sch(\mathcal{M})$（分别为 $Sch_{\mathrm{S}}(\mathcal{M})$）。

可达性奖励。给定任意线性时序逻辑公式 $\mathrm{F}B$，其中 B 为布尔公式，令 $\rho(\pi|\mathrm{F}B) = \sum_{i=1}^{n} \rho(s_i, a_i)$，其中 $\pi = s_1 a_1 s_2 a_2 \ldots \in \mathrm{IPath}(s_1)$，且 n 是满足 $L(s_n) \models B$ 且对所有 $i < n$ 均有 $L(s_i) \not\models B$ 的最小数；若此数不存在，则令 $\rho(\pi|\mathrm{F}B) = \infty$。令 $\Pr^{\mathcal{M},\mu}$ 为 IPath(s) 中路径上的概率测度。[1] 期望 $\mathrm{E}^{\mathcal{M}[\rho],\mu}(\mathrm{F}\, B) \doteq \int_\pi \rho(\pi|\mathrm{F}B)\mathrm{d}\Pr^{\mathcal{M},\mu}$，亦称可达性奖励 [21], ，是在 μ 下，沿 $\mathcal{M}$ 中一条路径直至到达满足 B 的状态时累积的期望奖励。若 $\sup^{\mu \in Sch(\mathcal{M})} \mathrm{E}^{\mathcal{M}[\rho],\mu}(\mathrm{F}\, B) < \infty$，则称 $\mathcal{M}[\rho]$ 关于 $\mathrm{F}B$ 奖励有限。

乘积马尔可夫决策过程。给定 $\mathcal{M} = (S,\ s_0,\ A,\ P,\ L)$ 和 $\mathcal{A} = (Q,\ q_0,\ Q_F,\ \Sigma,\ \delta)$，乘积马尔可夫决策过程是一个元组 $\mathcal{M} \otimes \mathcal{A} = (S \times Q,\ (s_0,\ q_0)\ A,\ P',\ L')$，其中 (i) $P' \colon S \times Q \times A \times S \times Q \to [0,1]$ 是一个转移概率函数，满足

$$P'(s,q,a,s',q') = \begin{cases} P(s,a,s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise} \end{cases}$$

且 (ii) $L' \colon S \times Q \to 2^\Sigma$ 是一个标记函数，使得 $L'(s,\ q) = L(s)$。令 $\mathcal{M}[\rho] \otimes \mathcal{A}$ 表示 $(\mathcal{M} \otimes \mathcal{A})[\rho]$，其中对于所有 $(s,\ q) \in S \times Q,\ \rho(s,\ q,\ a) = \rho(s,\ a)$，且 $a \in A(s)$。

几何。对于某个 n 的一个点（即向量）$\boldsymbol{v} \in \mathbb{R}^n$，令 v_i 表示 $\boldsymbol{v}$ 的 i^{th} 元素。权重向量 $\boldsymbol{w}$ 是满足 $w_i \geq 0$ 和 $\sum_{i=1}^{n} w_i = 1$ 的向量。$\boldsymbol{v}$ 与 $\boldsymbol{u}$ 的点积记作 $\boldsymbol{v} \cdot \boldsymbol{u}$，其值为总和 $\sum_{i=1}^{n} v_i u_i$。对于集合 $\Phi = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\} \subseteq \mathbb{R}^n$，$\Phi$ 中的凸组合为某个权重向量 $\boldsymbol{w} \in \mathbb{R}^m$ 对应的 $\sum_{i=1}^{m} w_i \boldsymbol{v}_i$。$\Phi$ 的凸包的向下闭包记作 $down(\Phi)$，它是满足以下条件的向量集合：对于任意 $\boldsymbol{u} \in down(\Phi)$，都存在一个凸组合 $\boldsymbol{v} = w_1\boldsymbol{v}_1 + \ldots + w_m\boldsymbol{v}_m$，使得 $u_i \leq v_i$。令 $\Psi \subseteq \mathbb{R}^n$ 为点的任意向下闭包。若对于所有 $1 \leq i \leq n$，$\boldsymbol{u}$ 都从上方支配 $\boldsymbol{v}$，则记作 $\boldsymbol{v} \leq \boldsymbol{u}$。如果 Ψ 中不存在从上方支配向量 $\boldsymbol{u} \in \Psi$ 的点，则称向量 $\boldsymbol{u} \in \Psi$ 为帕累托最优。Ψ 中的帕累托曲线是 Ψ 中帕累托最优向量的集合。以下引理可由分离超平面定理和支撑超平面定理推出。

引理 1（[6]）。设 $\Psi \subseteq \mathbb{R}^n$ 是由点集 $\boldsymbol{x} \in \mathbb{R}^n$ 构造的凸包的任意向下闭包。对于任意 $\boldsymbol{v} \notin \Psi$，存在一个权重向量 $\boldsymbol{w}$，使得对于所有 $\boldsymbol{x} \in \Psi$ 都有 $\boldsymbol{w} \cdot \boldsymbol{v} > \boldsymbol{w} \cdot \boldsymbol{x}$。我们称 $\boldsymbol{w}$ 将 $\boldsymbol{v}$ 与 Ψ 分离。此外，对于 Ψ 的帕累托曲线上的任意 $\boldsymbol{u}$，存在一个权重向量 $\boldsymbol{w}'$，使得对于所有 $\boldsymbol{x} \in \Psi$ 都有 $\boldsymbol{w}' \cdot \boldsymbol{u} \geq \boldsymbol{w}' \cdot \boldsymbol{x}$。我们称 $\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{w}' \cdot \boldsymbol{x} = \boldsymbol{w}' \cdot \boldsymbol{u}\}$ 是 Ψ 的支撑超平面。

[1] 该概率测度定义在由调度器 μ 诱导的离散时间马尔可夫链上，该调度器属于 $\mathcal{M}$（参见 [2]中的定义 10.92）。

双随机矩阵。对于矩阵 $\boldsymbol{U} \in \mathbb{R}^{n\times n}$，若存在某个 n，使得 $u_{i,\ j}$ 表示 $\boldsymbol{U}$ 中第 i^{th} 行、第 j^{th} 列的元素，则称 $\boldsymbol{U}$ 为双随机矩阵，如果 $u_{i,\ j} \geq 0$ 且 $\sum_{i'=1}^{n} u_{i',\ j} = \sum_{j'=1}^{n} u_{i,\ j'} = 1$ 对所有 $1 \leq i,\ j \leq n$ 均成立。若双随机矩阵 $\boldsymbol{U}$ 的每一行和每一列中恰好有一个元素的值为 1，则称其为置换矩阵。我们回顾以下 Birkhoff–von Neumann 定理：

引理2（[3]）。阶数为 n 的双随机矩阵 $\boldsymbol{U}$ 等价于若干置换矩阵 $\boldsymbol{U}_1$、……、$\boldsymbol{U}_k$ 的凸组合，其中 $k \leq n^2 - 2n + 2$。

随机分配。给定由智能体（分别为任务）组成的集合 I（分别为 J），且 $|I| = |J|$，一个（平衡）分配是一个双射函数 $f:\ J \to I$。将 J 到 I 的分配集合记为 $\mathcal{F}$。随机分配 ν 是定义在 $\mathcal{F}$ 上的随机化分布（或者等价地，是$\mathcal{F}$中的分配的凸组合）。为方便起见，令 $I = J = \{1, \ldots, n\}$。令 $\nu_{j\to i} = \nu(\{f \in \mathcal{F} \mid f(j) = i\})$，即根据 ν将任务 j 分配给智能体 i 的边际概率。显然，任何分配都等价于一个置换矩阵。此外，根据引理2，任何双随机矩阵 $\boldsymbol{U}$ 都等价于一个随机分配 ν，使得$u_{i,\ j} = \nu_{j\to i}$。

3 问题与方法

3.1 问题陈述

在我们的多智能体系统设定中，每个智能体都是一个马尔可夫决策过程（MDP）（具有奖励结构），每项任务都是一个确定性有限状态自动机（DFA）（等价于一个余安全线性时序逻辑公式），奖励则是完成任务的概率，以及智能体执行任务的成本（作为负奖励）。因此，我们旨在计算一个随机任务分配，以及所有智能体和任务的调度器，使其满足多项概率和成本要求。直观而言，我们考虑满足以下两个条件的任务分配与智能体规划场景 [32]：

C1. 各项任务相互独立。**C2.** 智能体的行为不会相互影响。

对于每个 $(i, j) \in I \times J$,[2]，我们定义一个智能体—任务（乘积）MDP $\mathcal{M}_{i\otimes j}[\rho_i] \doteq \mathcal{M}_i[\rho_i] \otimes \mathcal{A}_j$，并加入一个原子命题 done^j，使得

$$L_{i,j}(s, q) \models \mathtt{done}_j \text{ iff } q \in Q_{j,F} \cup Q_{j,R}$$

这表示“任务 j 已结束（无论是已完成还是失败）”。对于每个 $j \in J$，我们定义一个指定奖励函数 $\rho_{j+|I|}:\ \bigcup_{i\in I}(S_i \times Q_j \times A_i) \to \{0, 1\}$，使得 $\rho_{j+|I|}(s,\ q,\ a) = 1$ 当且仅当 $q \notin Q_{j,\ F}$ 且 $\mathsf{suc}(q) \subseteq Q_{j,\ F}$。如果不存在这样的前汇点$q$，我们可以修改 $\mathcal{A}_j$，将 q 纳入其中，而不改变 $acc(\mathcal{A}_j)$。换言之，每当一个接受位置即将被

[2] 在本文中，除非另有明确说明，我们假设 $I = J = \{1, \ldots, n\}$ 对某个 n 成立，但仍使用 I, J 表示智能体或任务的引用。

$$
\begin{aligned}
&\text{最大化} \quad \sum_{(s,q)\in S_i\times Q_j} \sum_{a\in A_i(s)} \rho_i(s,q,a) x_{s,q,a} \\
&\begin{cases} \sum_{j\in J} \sum_{i\in I} \sum_{(s,q)\in S_i\times Q_j} \sum_{a\in A_i(s)} \rho_{j+|I|}(s,q,a) x_{s,q,a} & \forall i \in I \\ & \forall j \in J \end{cases} \\
&\text{约束条件} \quad \forall i \in I, j \in J, (s,q) \in S_i \times Q_j\text{:} \\
&\begin{cases} \sum_{a\in A_i(s)} x_{s,q,a} - \mathrm{I}_{(s,q)=(s_{i,0},q_{j,0})} x_{i,j} \\ \quad = \sum_{(s',q')\in S_i\times Q_j} \sum_{a'\in A_i(s')} P_{i,j}(s',q',a',s,q) x_{s',q',a'} \\ x_{s,q,a} \geq 0;\ x_{i,j} \geq 0;\ \sum_{i'\in I} x_{i',j} = 1;\ \sum_{j'\in I} x_{i,j'} = 1 \end{cases}
\end{aligned}
$$

图 1：MORAP 的多目标线性规划

首次经过时，$\rho_{j+|I|}$ 就会提供一次性单位奖励。通俗地说，$\rho_{j+|I|}$ 表示“完成任务 j 的概率”。由于原子命题 done$_j$ 对每个 $\mathcal{M}_{i\otimes j}$都是固定的，我们将 $E^{\mathcal{M}_{i\otimes j}[\rho_k],\ \mu_{i,\ j}}$(F done$_j$) 简写为 $\mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_k],\ \mu_{i,\ j}}$，其中 $k = j$或 $k = j+|I|$。由于智能体的可达性奖励可能是无穷的，并会导致计算不稳定，与多目标验证文献 [14,15],类似，我们要求对于所有 $(i,\ j) \in I \times J$，$\mathcal{M}_{i\otimes j}[\rho_i]$ 均为奖励有限（相对于 F done$_j$）。

定义 1（MORAP）。多目标随机分配与规划（*MORAP*）问题是求解一个双随机矩阵$(x_{i,\ j})_{i\in I,\ j\in J}$ 以及一组调度器 $\{\mu_{i,j} \in Sch(\mathcal{M}_{i\otimes j}) \mid i \in I,\ j \in J\}$ 使其满足以下两类要求，即 *R1* 和 *R2*：

（R1. 概率）$\sum_{i\in I} x_{i,\ j} \mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_{j+|I|}],\ \mu_{i,\ j}} \geq p_j$ 对于所有 $j \in J$，
（R2. 成本）$\sum_{j\in J} x_{i,\ j} \mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_i],\ \mu_{i,\ j}} \geq c_i$ 对于所有 $i \in I$，

其中给定了概率阈值 (p_j $c_{i\ i\in I} \in$ $)_{j\in J} \in [0,1]^{|J|}$ 和成本阈值()。如果满足上述要求，我们就称给定阈值下的 *MORAP*问题可行，或者简称这些阈值可行。

在条件 C1 和 C2 下，定义 1 是一种恰当的表述。首先，由于任务相互独立（C1），概率要求只需考虑每项任务成功的概率。其次，由于每个智能体执行任何任务都不会影响其他智能体（C2），成本要求只需考虑每个智能体的成本。在实践中，我们可以将条件 $|I| = |J|$ 放宽为 $|I| \geq |J|$（例如，添加概率阈值为 0 的虚拟任务）。

3.2 凸性刻画与集中式模型

我们的 MORAP 问题的一个重要特征是凸性，即可行概率和成本阈值的向下闭包是一个凸多胞体（也就是某些有限点集的向下凸包）。这是因为 MORAP 问题可以通过一种类似于多目标马尔可夫决策过程 [14,24,10]验证所依据的技术，表示为多目标线性规划（线性规划）。图 1 展示了 MORAP 的多目标线性规划。

直观地，对于每个 (i,j)，$\in I\times J$, $x_{i,j}$ 表示将 j 分配给i 的概率（参见引理2）；对于每个 (s,q)，$\in S_i\times Q_i$, $x_{s,q,a}$ 是访问 (s,q) 并采取动作 a的期望频率。无记忆调度器可定义如下：$\mu_{i,j}(s,q)(a)=x_{s,q,a}/x_{s,q}$ 其中$x_{s,q}=\sum_{a\in A_i(s)}x_{s,q,a}$。因此，MORAP问题的时间复杂度如下：

定理1。 *MORAP*问题的可行性可在关于 $\sum_{i\in I,\ j\in J}|\mathcal{M}_{i\otimes j}|$ 的多项式时间内判定。

对于大规模问题，线性规划的效率不高，而在实践中，值迭代和策略迭代是更具可扩展性的方法。为此，我们定义一个新的马尔可夫决策过程，将所有智能体-任务马尔可夫决策过程结合起来，并加入一个额外变量，用于指示哪些智能体已被分配任务。该马尔可夫决策过程旨在直接以集中式方式求解随机分配问题

定义2（集中式马尔可夫决策过程）。 集中式马尔可夫决策过程为 $\mathcal{M}^{\mathrm{ct}}=(S^{\mathrm{ct}},s_0^{\mathrm{ct}},A^{\mathrm{ct}},P_{\mathrm{ct}}L^{\mathrm{ct}})$ 其中 *(i)* $S^{\mathrm{ct}}=\biguplus_{i\in I}\biguplus_{j\in J}S_i\times Q_j\times 2^I$，*(ii)* $s_0^{\mathrm{ct}}=(s_{1,\,0},\ q_{1,\,0},\ \emptyset)$，*(iii)* $A^{\mathrm{ct}}=\bigcup_{i\in I}A_i\cup\{b_1,b_2,b_3\}$，*(iv)* $P_{\mathrm{ct}}=S^{\mathrm{ct}}\times A^{\mathrm{ct}}\times S^{\mathrm{ct}}\to[0,1]$，满足：

– $P^{\mathrm{ct}}(s,q,\sharp,a,s',q',\sharp)=P_{i,j}(s,q,a,s',q')$ 当 $s,s'\in S_i,\ q,q'\in Q_j,\ a\in A_i(s)$ 且 $i\in\sharp$ 对某些 i,j 成立时，– $P^{\mathrm{ct}}(s,q,\sharp,b_1,s,q,\sharp\cup\{i\})=1$ 当 $s=s_{i,0},\ q=q_{j,0}$ 且 $i\notin\sharp$ 对某些 i,j 成立时，– $P^{\mathrm{ct}}(s,q,\sharp,b_2,s',q,\sharp)=1$ 当 $s=s_{i,0},\ q=q_{j,0}$ 且 $s'=s_{i',0}$，并且 $i'=\min\{i''\in I\mid i''>i,i''\notin\sharp\}$ 对某些 i,j 成立时，– $P^{\mathrm{ct}}(s,q,\sharp,b_3,s',q',\sharp)=1$ 当 $s\in S_i,\ q\in Q_{j,F}\cup Q_{j,R},\ i\in\sharp,\ s'=s_{i',0}$ 且 $i'=\min\{i''\in I\mid i''\notin\sharp\}$，并且 $q'=q_{j+1,0}$ 对某些 $i,\ j<|J|$ 成立时，

(v) $L^{\mathrm{ct}}:\ S^{\mathrm{ct}}\to 2^{\{\mathrm{done}\}}$ 当且仅当 $L^{\mathrm{ct}}(s,q)\models$ 完成 *iff* $q\in Q_{j,\ F}\cup Q_{j,\ R}$。

直观而言，$\sharp\subseteq I$ 表示曾处理过某些任务的智能体；b_1 表示“某个任务被分配给当前智能体”；b_2 表示“某个任务被转交给下一个智能体”；而 b_3 表示“正在考虑下一个任务”。处理已分配的任务时，该模型的行为如同一个独立的乘积马尔可夫决策过程。

给定适用于 $\mathcal{M}_{i\otimes j}$的任意奖励结构 ρ，我们通过令 $\rho(s,q,\sharp,a)=\rho(s,q,a)$（若 $a\in A_i(s)$且 $\rho(s,q,\sharp,a)$；否则为 $=0$），将 ρ 视为适用于$\mathcal{M}^{\mathrm{ct}}$的奖励结构，对所有$(s,q,\sharp,a)$均成立。类似地，给定适用于 $\mathcal{M}^{\mathrm{ct}}$的任意奖励结构 ρ，将ρ限制在 $S_i\times Q_j\times A_i$ 上，即得到适用于 $\mathcal{M}_{i\otimes j}$的奖励结构。与智能体-任务马尔可夫决策过程类似，对于给定的 ρ，我们将 $\mathrm{E}^{\mathcal{M}^{\mathrm{ct}}[\rho],\mu}(\mathrm{F\ done})$ 简写为 $\mathrm{E}^{\mathcal{M}^{\mathrm{ct}}[\rho],\mu}$。

定理 2. 定义 *1* 中的 *MORAP* 问题关于$(p_j)_{j\in J}$ 和 $(c_i)_{i\in I}$ 是可行的，当且仅当存在 $\mu\in Sch(\mathcal{M}^{\mathrm{ct}})$ 使得 $\mathbf{E}^{\mathcal{M}^{\mathrm{ct}}[\rho_{j+|J|}],\mu}\geq p_j$ 且 $\mathbf{E}^{\mathcal{M}^{\mathrm{ct}}[\rho_i],\mu}\geq c_i$ 对所有 $i\in I,j\in J$ 均成立。

根据上述定理，可以针对集中式马尔可夫决策过程 $\mathcal{M}^{\mathrm{ct}}$（例如，使用值迭代）来求解 MORAP问题。因此，可以使用现有的多目标马尔可夫决策过程验证概率模型检测工具（例如 Prism[20] 和 Storm [17]）然而，$\mathcal{M}^{\mathrm{ct}}$ 的状态空间随智能体团队规模 $|I|$呈指数增长。因此，这种方法难以扩展到规模相对较大的 $|I|$。

算法1： 面向点的帕累托计算

输入： $\{\mathcal{M}_{i\otimes j}\}_{(i,j)\in I\times J}$, $\boldsymbol{\rho}=\{\rho_k\}_{k=1}^{|I|+|J|}$, $\boldsymbol{t}$ （ $\boldsymbol{c}$ 和 $\boldsymbol{p}$的拼接）， $\varepsilon\geq 0$

1 $\boldsymbol{t}_{\uparrow}:=-\infty$; $\boldsymbol{t}_{\downarrow}:=\boldsymbol{t}$; $\Phi:=\emptyset$; $\Lambda:=\emptyset$; $\boldsymbol{w}:=(1,\ 0,\ \ldots\ldots,\ 0)$;
2 **当** $\|\boldsymbol{t}_{\downarrow}-\boldsymbol{t}_{\uparrow}\|>\varepsilon$ **时循环**
3 **如果** $\Phi\neq\emptyset$ **则**
4 求使 $\|\boldsymbol{t}-\boldsymbol{x}\|$最小的 $\boldsymbol{x}\in down(\Phi)$;
5 <escape>$\boldsymbol{t}_{\uparrow}:=\boldsymbol{x}$;</escape>
6 $\boldsymbol{w}:=\boldsymbol{M}(\boldsymbol{t}-\boldsymbol{t}_{\uparrow})/\|\boldsymbol{M}(\boldsymbol{t}-\boldsymbol{t}_{\uparrow})\|_1$;
7 求解 $\boldsymbol{r}$，使得 $\{\boldsymbol{y}\mid\boldsymbol{w}\cdot\boldsymbol{y}=\boldsymbol{w}\cdot\boldsymbol{r}\}$ 是 $\mathscr{C}$ 的支撑超平面;
8 $\Phi:=\Phi\cup\{\boldsymbol{r}\}$; $\Lambda:=\Lambda\cup\{(\boldsymbol{w},\ \boldsymbol{r})\}$;
9 **如果** $\boldsymbol{w}\cdot\boldsymbol{r}<\boldsymbol{w}\cdot\boldsymbol{t}_{\downarrow}$ **则**
10 求使 $\boldsymbol{z}$ 最小化的 $\|\boldsymbol{t}-\boldsymbol{z}\|$，满足 $\boldsymbol{w}'\cdot\boldsymbol{r}'\geq\boldsymbol{w}'\cdot\boldsymbol{z}$ 对所有 $(\boldsymbol{w}',\ \boldsymbol{r}')\in\Lambda$;
11 $\boldsymbol{t}_{\downarrow}:=\boldsymbol{z}$;

3.3 基于去中心化模型的面向点的帕累托计算

我们提出一种分散式方法来求解给定的 MORAP问题，尤其适用于智能体数量（即任务数量）较大的情况。除了判断问题是否可行之外，对于不可行的问题，我们的方法还会计算该问题帕累托曲线上的一个新的可行阈值向量，并使其尽可能接近原始阈值向量，误差不超过一定的数值容差。

令 $\mathscr{C}_0=\{(\mathbf{E}^{\mathcal{M}^{\mathrm{ct}}[\rho_k],\ \mu})_{1\leq k\leq|I|+|J|}\mid\mu\in Sch(\mathcal{M}^{\mathrm{ct}})\}$。奖励有限性意味着 $\mathscr{C}_0$ 非空且有界。令 $\mathscr{C}$ 为 $\mathscr{C}_0$ 的向下闭包，即 $\mathscr{C}$ 是定义 1 中的可行阈值向量集合。我们方法的主要算法见算法1，支撑超平面的计算（即第7行）详见算法2。算法1通过迭代细化 $\mathscr{C}$ 的下近似（编码为 Φ）和上近似（编码为 Λ）来运行。它计算向量 $\boldsymbol{t}_{\uparrow}$（分别为 $\boldsymbol{t}_{\downarrow}$），该向量是从原点阈值向量 $\boldsymbol{t}$ 到下近似（分别到上近似）的最近点，使得 $\boldsymbol{t}_{\uparrow}$ 和 $\boldsymbol{t}_{\downarrow}$ 最终收敛。

该算法使用一般范数 $\|\cdot\|$ 来衡量向量之间的距离，因为在实践中，人们可能更倾向于区分概率和成本阈值的重要性。 $\boldsymbol{v}$的内积 $\boldsymbol{u}\in\mathbb{R}^m$ （m 为正整数），记作 $\langle\boldsymbol{v},\boldsymbol{u}\rangle$，是矩阵与向量的乘法 $\boldsymbol{v}^T\boldsymbol{M}\boldsymbol{u}$，其中 $\boldsymbol{M}$是对称正定矩阵。注意，如果 $\boldsymbol{M}$ 是单位矩阵，那么 $\langle\boldsymbol{v},\boldsymbol{u}\rangle$就是 $\boldsymbol{v}\cdot\boldsymbol{u}$。于是， $\|\boldsymbol{v}|=\langle\boldsymbol{v},\boldsymbol{v}\rangle$。令 $\|\cdot\|_1$ 表示向量1-范数。第 6 行计算得到的权重向量$\boldsymbol{w}$ 是将 $\boldsymbol{t}$ 投影到 $down$上的（反向）方向（Φ）。此外， $\boldsymbol{w}\cdot\boldsymbol{\rho}$ 表示 $\boldsymbol{\rho}$中奖励函数的加权组合。

定理 3. 算法*1* 对任意 $\varepsilon\geq 0$ 均会终止。在算法*1* 的整个执行过程中，以下性质均成立：*(i)* $\boldsymbol{t}_{\uparrow}\in\mathscr{C}$。*(ii)* 如果 $\boldsymbol{t}\in\mathscr{C}$，则 $\boldsymbol{t}_{\downarrow}=\boldsymbol{t}$。*(iii)* $\|\boldsymbol{t}-\boldsymbol{t}_{\downarrow}\|\leq$最小值 $_{\boldsymbol{u}\in\mathscr{C}}\|\boldsymbol{t}-\boldsymbol{u}\|\leq\|\boldsymbol{t}-\boldsymbol{t}_{\uparrow}\|$。

算法2： 算法1第7行中的支撑超平面计算

输入： $\{\mathcal{M}_{i\otimes j}\}_{(i,j)\in I\times J}$, $\boldsymbol{\rho}=\{\rho_k\}_{k=1}^{|I|+|J|}$, $\boldsymbol{w}$

对每个$(i,j)\in I\times J$ **执行**

  /* 第2行通过策略迭代计算。 */

  $c_{i,j}:=\mathbf{E}_{\mathcal{M}_{i\otimes j}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\mu_{i,j}}$ 其中 $\mu_{i,j}:=\arg\max_{\mu}\mathbf{E}_{\mathcal{M}_{i\otimes j}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\mu}$;

寻找一个使 $\sum_{j\in J}c_{f(j),j}$ 最大化的分配 $f\in\mathcal{F}$;

对 $j\in J$ **执行 foreach**

  /*第 5–6 行通过值迭代计算。*/ */

  $r_{j+|I|}:=\mathbf{E}^{\mathcal{M}_{f(j)\otimes j}[\rho_{j+|I|}],\ \mu_{f(j)^j}}$;

  $r_{f(j)}:=\mathbf{E}^{\mathcal{M}_{f(j)\otimes j}[\rho_{f(j)}],\mu_{f(j)^j}}$;

返回$(r_k)_{k=1}^{|I|+|J|}$;

推论 1。 假设 $\varepsilon=0$。算法*1* 终止后，以下性质成立：*(i)* $\boldsymbol{t}_{\uparrow}=\boldsymbol{t}_{\downarrow}$。*(ii)* $\boldsymbol{t}\in\mathscr{C}$ 当且仅当 $\boldsymbol{t}_{\downarrow}=\boldsymbol{t}$。*(iii)* 如果 $\boldsymbol{t}\notin\mathscr{C}$，那么 $\boldsymbol{t}_{\downarrow}$ 位于 $\mathscr{C}$的帕累托曲线上。

算法2针对给定方向 $\boldsymbol{w}$，求出 $\mathscr{C}$ 的一个支撑超平面。由于两个内循环中采用了概率模型检验，这通常是一项计算开销很大的任务。为了说明算法2的重要性，我们指出，$\mathscr{C}$ 是定义在集中式模型 $\mathcal{M}^{\mathrm{ct}}$ 上的一个凸集，其规模为$O(2^{|I|})$。但算法2并不处理 $\mathcal{M}^{\mathrm{ct}}$，而是在由 $\{\mathcal{M}_{i\otimes j}\}_{(i,j)\in I\times J}$组成的去中心化模型上运行。第一个内循环包含 $|I|\times|J|$（即$|I|^2$）个策略迭代过程，用于计算最优调度器和可达性奖励。第二个内循环则在固定调度器下使用 $2|I|$ 值迭代过程。[3] 模型选择通过匈牙利算法[18]（第3行）计算，其运行时间为 $O(|I|^3)$。使用去中心化模型的另一个重要意义是，两个内循环可以并行执行，我们将在第4节对此作进一步阐述。还请注意，若对于某个$(i,\ j)$，$\mathcal{M}_{i\otimes j}=\mathcal{M}_{i'\otimes j'}\neq(i',\ j')$，则两个内循环中可以跳过某些模型。

在实现中，我们应选择某个正的 ε，原因有以下三点：首先，用于计算两个内循环的策略迭代和值迭代都是近似的。其次，在算法中求解优化问题时，通常会出现轻微的数值不精确性（例如舍入误差）。第三，由于算法1的最坏情况下迭代次数随模型规模和智能体数量 $[14]$，呈指数增长，因此，合适的 ε 可以提前终止算法，并得到精度在实践中可接受的近似阈值向量。

为便于综合，我们可以提取一个随机分配和一个调度器集合。假设 while 循环总共迭代 ℓ 次。令 $\{\mu_{i,j}^{\iota}\mid i\in I、\ j\in J、\ \}$和 f_{ι}分别为算法2第 2—3 行在第 ι^{th} 次迭代中生成的结果。令 $v_1\boldsymbol{r}_1+\ldots+v_{\ell}\boldsymbol{r}_{\ell}\geq\boldsymbol{t}_{\uparrow}$对于某个权重向量 $\boldsymbol{v}$成立（该 $\boldsymbol{v}$存在，因为 $\boldsymbol{t}_{\uparrow}\in down(\Phi)$）。分配 $v_1f_1+\ldots+v_{\ell}f_{\ell}$的凸组合定义了一个随机分配（即双随机矩阵）。在按照概率 f_{ι}

[3] 计算这两个内循环的方法详见 [28] 的附录。

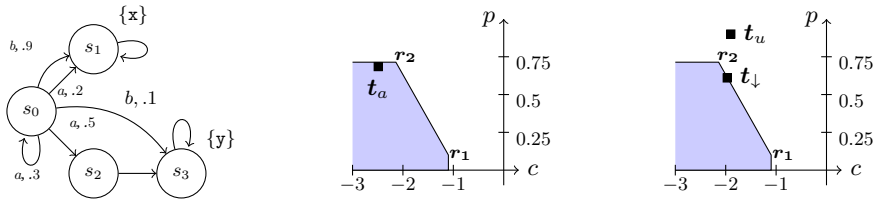

图2：MOMDP 智能体示例及算法1的相应执行过程。

随机选择后，用于规划的调度器来自 $\{\mu^{\iota}_{i,j} \mid j \in J, f_{\iota}(j) = i\}$。

示例。图2是一个简单示例，由一个智能体和一个任务组成，用于演示算法1的执行过程。图2a展示了智能体的马尔可夫决策过程，其中对于每个 $a \in A(s)$和 $s \in S$，$\rho(s,a) = -1$，且任务为 $\varphi := \neg \mathrm{x}\ \mathrm{U}\ \mathrm{y}$。设 $\varepsilon = 0.001$。图2b展示了使用可行阈值向量$\boldsymbol{t}_a = (-2.5, 0.7)$进行计算的过程。初始时，$\boldsymbol{w} = (1, 0)$，由此得到 $\boldsymbol{r}_1 = (-1.1, 0.1)$和超平面 $\boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w} \cdot \boldsymbol{r}_1 = -1.1$。此处，$\|\boldsymbol{t}_\downarrow - \boldsymbol{t}_\uparrow\| = 0.6$，因此还需进行一次迭代。算法找到 $\boldsymbol{w} = (0.4, 0.6)$以及相应的$\boldsymbol{r}_2 = (-2.1, 0.71)$。由于 $\boldsymbol{t}_a$包含于 $down(\{\boldsymbol{r}_1, \boldsymbol{r}_2\})$中，算法终止。图2c展示了使用不可行的 $\boldsymbol{t}_u = (-1.8, 0.9)$的情况。与前一种情况类似，算法找到 $\boldsymbol{w} = (1, 0)$和 $\boldsymbol{r}_1$，随后找到 $\boldsymbol{w} = (0.4, 0.6)$和 $\boldsymbol{r}_2$。由于 $\boldsymbol{w} \cdot \boldsymbol{r}_2 < \boldsymbol{w} \cdot \boldsymbol{t}_u$，算法在第10行找到新的阈值向量 $\boldsymbol{t}_\downarrow = (-1.97, 0.61)$。此时，由于 $\|\boldsymbol{t}_\downarrow - \boldsymbol{t}_\uparrow\| < \varepsilon$，算法终止。

4 GPU-CPU 混合实现

在现代系统中，图形处理器和多核中央处理器硬件已广泛可用。我们为 MORAP 框架开发了一种实现，利用异构的图形处理器和多核中央处理器资源来加速计算。加速基于算法2中两个概率模型检验循环内彼此独立的数据；在实际应用中，这些循环占据了算法1的大部分运行时间。通过将模型分配给每个可用的图形处理器设备和 CPU 核心，即可在图形处理器和 CPU 上并行执行。对于图形处理器，还可以在概率模型检验所需的底层矩阵运算中实现更大规模的并行化。

实现目标。 我们的框架的主要目标是最大化吞吐量和并行性。组合使用多个设备是一个负载均衡问题，我们可以通过有效地调度模型检测问题，让所有设备都保持最佳利用状态，并缩短运行时间。我们将运行在 GPU 上的计算称为设备操作。多核处理器可以在计算前利用共享内存，且延迟可忽略不计。使用多核处理器实现并行性时，主要需要避免线程阻塞 和上下文切换 开销。此外，由于底层计算是顺序执行的，处理器的执行时间将取决于模型的

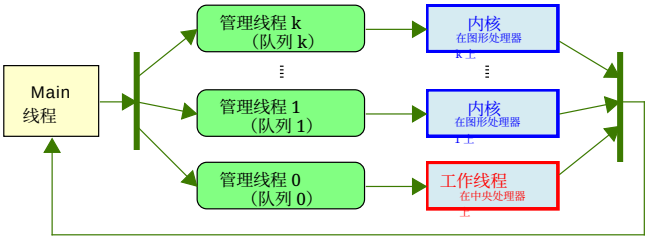

图 3：MORAP框架的并行架构。

表 1：MORAP实现中的线程角色

组件	**功能**
主线程	将模型加载到主存并运行所有计算】【。 除算法 2 中两个（内部）循环之外的计算；生成和 为工作线程和内核将模型分配到队列中
管理线程	管理有界先进先出队列；（一个线程）生成 CPU 工作线程；（其他线程）调用 GPU 内核， 包括在主机内存与图形处理器之间复制数据 的设备内存；通过消息传递通道相互通信， 以实现负载均衡
工作线程	计算算法 2 中的循环；每个线程绑定到一个 CPU 核心，并且每次处理一个模型
GPU 内核	在算法 2 中计算循环；每个内核运行在一个 图形处理器设备，每次处理一个模型。

状态空间大小。另一方面，GPU 计算的主要问题是主机与设备之间的数据传输。

设计。 图 3 展示了我们框架的并行架构，表 1 说明了不同线程类型的作用。具体而言，有 $k+1$ 个管理线程负责控制 $k+1$ 个智能体-任务模型的先进先出队列 $\mathcal{M}_{i\otimes j}$，其中 k 是可用 GPU 设备的数量。一个特定的管理线程负责创建绑定到每个可用 CPU 核心的工作线程，而其他管理线程则负责在图形处理器上调用内核函数。由于每个工作线程专门负责计算一个 $\mathcal{M}_{i\otimes j}$，因此响应时间和上下文切换开销均得以最小化。管理线程无需绑定到任何 CPU 核心，因为程序管理对计算资源的需求并不高。图形处理器设备与 CPU 核心之间的计算工作负载通过工作窃取 方法 [4]，进行控制，即，如果某个处理器或设备处于空闲状态且其队列为空，那么相应的管理线程就会从其他队列中请求（即窃取）一个模型。通过这种方式，硬件资源能够得到最优的任务负载分配，所有线程均以异步方式运行。

编程与数据结构。我们使用多种语言实现了该框架，包括 Rust（框架应用程序接口）、CUDA C（图形处理器设备控制）以及 Python 用户界面，其中 Rust 应用程序接口调用 C，而 Python 则通过外部函数接口（FFI）进行调用。我们的实现使用了 仿射 属性

[25] 在 Rust [22] 中采用类型系统，以确保在应用程序中，使用仅可移动类型的已拥有变量至多只能使用一次。这项特性在并行架构中特别有用，因为 $\mathcal{M}_{i\otimes j}$ 在同一时间至多只能由一个线程拥有，而且线程计算的副作用不会影响其他线程。将数据访问隔离到各个 $\mathcal{M}_{i\otimes j}$，可以减少对共享内存访问的需求，使框架免受数据竞争和数据饥饿问题的影响。因此，这个问题极易 并行化。以这种方式构建架构，可确保我们的实现接近并行度的上限。我们的实现使用显式数据结构（即稀疏矩阵），为每个 $\mathcal{M}_{i\otimes j}$存储转移概率函数和奖励结构。GPU 上的低级并行矩阵运算通过 CUDA cuSPARSE 应用程序接口实现，从而确保线程安全。第 2 行中用于寻找最优策略的状态-动作值归约操作，也通过一次内核启动并行计算。GPU 内核的最佳占用率由内核启动器和对 CUDA cudaOccupancyMaxPotentialBlockSize 的调用进行管理。

5 实验

我们的 MORAP问题的一个现实示例是智能仓储或机器人移动履行系统（RMFS）。该系统通常集中控制任务，机器人之间的通信有限 [35]。如图4所示，环境是一个 $W \times H$ 二维网格，通常由可移动货架（搁板）、存储位置和工作站组成，可在其中进行拣货和补货 [23]。机器人在仓

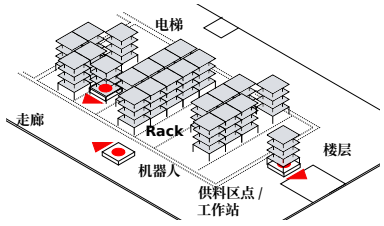

图4：智能仓库布局

库中移动，以执行订单拣货和补货等任务。

机器人的状态由机器人位置、内部机器人状态（例如是否携带货架）以及环境参数（例如货架位置）描述，并且是离散的。机器人可以执行向左/右旋转、向前移动、装载/卸下载货架等动作。本示例中的 MORAP问题是在多目标要求——运行成本和任务完成概率——下，将（随机）任务 n 分配给 n 机器人，并为机器人进行任务规划。我们考虑了面向智能体的补货任务，其非正式描述如下：“在未携带任何物品时，前往仓库中的某个货架位置，取出货架并将其运送到供料区进行补货，然后搬运该货架并将其放回指定的货架位置。”形式上，每个补货任务都可以表示为一个协安全线性时序逻辑公式或一个确定性有限状态自动机。其他任务（例如拣货任务）也可以用类似方式表示。

我们进行了两项实验，在智能仓储示例中使用算法1评估我们的 MORAP 实现，考察不同仓库尺寸 $W \times H$ 和不同智能体（任务）数量 n。请注意，我们评估的是主算法（即算法1）每次迭代的运行时间，而非端到端运行时间。

表2：算法1中集中式和分散式模型每次迭代的平均运行时间（秒）。其中，states 和 transitions 分别指可达状态和可达转移。

W.H. size $W \times H$	智能体（任务）num. n	分散式							集中式			
		Dec.		每次迭代时间					集中式每次迭代时间			
		模型规模	混合状态转换。	多重	图形处理器	CPU		CPU	模型规模	状态转换。	CPU	GPU
6×6	2	17k	104k	0.016	**0.01**	0.037	0.025		21.2K	136K	0.059	0.017
	5	106k	652k	0.023	**0.02**	0.2	0.36		3.5M	22.5M	6.1	2.35
	6	152K	940K	0.03	**0.022**	0.96	0.38		24.9M	162M	timeout	15.2
	50	10.6M	65.3M	1.36	**1.0**	27.2	11.1		memerr	memerr	-	-
	100	42.4M	261M	4.8	**3.9**	90.8	31.9		-	-	-	-
12×12	2	254k	1.5M	0.18	0.14	0.13	**0.09**		190k	1.2M	1.08	1.5
	4	1.0M	6.1M	**0.36**	0.38	0.33	1.78		635k	4.2M	9.8	2.1
	6	2.2M	13.8M	**0.7**	0.9	0.7	4.0		memerr	memerr	-	-
	8	4.1M	24.5M	**1.1**	1.6	1.2	7.2		-	-	-	-
	10	6.4M	38.3M	**1.8**	4.23	2.46	11.7			-	-	-
	20	25.4M	153M	**6.5**	17.3	9.8	45.7		-	-	-	-
	30	57.2M	345M	**15.3**	38.8	22.1	timeout		-	-	-	-

实验 1 进行了两项比较：首先，比较集中式模型和分散式模型的模型规模；其次，比较 GPU-CPU 混合、（纯）图形处理器、多核 CPU 和单核 CPU 的计算运行时间。请注意，GPU-CPU 混合和多核 CPU 计算仅适用于分散式模型。由于概率模型检验工具 Prism 和 Storm 不支持任务指派问题，为了对我们的实现进行性能基准测试，**实验 2** 比较了我们的实现、Prism 和 Storm 对集中式模型进行验证时的平均运行时间（仅验证）。Prism 和 Storm 的工作方式与算法1类似，即通过迭代生成权重向量 $\boldsymbol{w}$ 并计算新的帕累托点。在我们进行的所有实验中，迭代次数介于 2 到 16 次之间。

所有实验均在 Debian 系统上进行，所用设备包括 AMD 2970WX 24 核 3.0GHz 处理器、PCIe 3.0 32Gbps 带宽、配备 6144 个 CUDA Cores 的 3070Ti 1.77GHz 8Gb 图形处理器，以及 32Gb 内存。用于复现实验的工件可在线获取[4]。实验使用单个图形处理器，因此 $k = 1$ 图形处理器管理线程的数量。Prism 的配置包括使用显式数据结构，并将 Java 堆大小和混合 maxmem 设为 32Gb，以避免内存异常。Storm 的默认配置已足够。值迭代的停止阈值设为 10^{-6}。运行时间上限设为 180 秒；若运行时间超过该上限，则记录为超时错误。若图形处理器设备内存不足，则记录为 memerr。帕累托曲线阈值 ε（见算法 1 第 2 行）设为 0.01。

实验1的结果列于表2。可以观察到，总体而言，去中心化模型的运行时间性能显著

[4] https://github.com/tmrob2/hybrid-motap

表 3：集中式模型在一个智能体和一个任务条件下，每次迭代的仅验证平均运行时间（秒）对比

仓库尺寸 $W \times H$	模型规模 状态	trans	CPU	每次迭代时间 图形处理器	Prism	Storm
3×3	334	2.17k	**1e-4**	0.03	0.005	0.038
6×6	4.2k	18.8k	**0.004**	0.038	0.025	0.058
8×8	12.9k	78.5k	**0.017**	0.041	0.081	0.114
10×10	30.9k	187k	0.048	**0.046**	0.17	0.33

优于集中式模型。正如预期，随着智能体和任务数量的增加，集中式模型的运行时间呈指数级增长，而去中心化模型的增长则呈线性。表2还显示，某种形式的并行实现能够提升运行时间性能。对于 6×6 仓库尺寸，多 CPU 相较于单 CPU 几乎实现了 10 倍的性能提升。对于 12×12 仓库尺寸，GPU-CPU 混合方式实现了类似的性能提升。在进行这项实验时，我们观察到一个性能指标，即模型检测时间与模型（数据）复制时间之比：该比值越高（或越低），意味着 GPU 加速越（或越不）有效。与6×6 仓库相比， 12×12 仓库的这一比值更高，因为前者的尺寸会产生比后者更大的单个智能体-任务模型。我们尤其观察到，较高的比值对 GPU-CPU 混合方法至关重要。对于较大的单个智能体-任务模型，混合方法相较于纯 GPU 加速和多核CPU加速都取得了显著提升。

实验2比较了我们的实现与 Prism 和 Storm 中多目标模型检测功能的性能。本实验针对一个智能体和一个任务的集中式模型进行，该模型本质上是这两种工具均可接受的标准多目标马尔可夫决策过程模型。对于我们的实现，我们将 MORAP 问题限制在仅验证设置，通过将算法1中的第10–11行替换为 $break$ 语句来终止算法。（因此，当且仅当验证返回 $false$ 时，$break$ 语句才会执行。）对于 Prism 和 Storm，我们将同一问题指定为可达性查询。比较仅包括模型检测时间，不包括模型构建时间。如表3所示，结果表明，与现有工具相比，我们的实现仍能达到有竞争力的性能。可以看出，即使没有利用去中心化模型的并行性，我们的实现仍是一个高效的多目标概率模型检测框架。

6 相关工作

多目标优化研究目标可能相互冲突的规划领域，并关注帕累托最优解。这些问题通常是多目标模型检测的重点 [30]。高效合成一组最优调度器，以最大化期望总奖励。

多目标模型检测的相关内容见 [7,9,10,14,29,13]。而步数和奖励有界的可达概率见 [14,16]。最近，[27]，提出了一种计算效率较高的多目标模型检测方法，用于处理长期平均奖励和总混合奖励，该方法是对 [14] 的推广。我们的面向点的帕累托计算是一种新方法，可补充 [14] 中现有的多目标查询，并专门针对多智能体系统的可扩展性进行优化。与现有方法不同，如果给定的阈值点不可行，我们的算法会计算出一个距离该给定点最近的帕累托最优点。

[8]，研究了多目标马尔可夫决策过程的 GPU 加速，但该方法针对特定问题，且不包含任务验证。[31,34]提出了 GPU 加速的并行稀疏值迭代算法。[31] 中的实现与我们的方法类似，尤其是算法2第 2 行中的值迭代，包括用于动作比较的归约内核操作，但未考虑多目标或任务规范。[34]，所研究的 GPU 加速需要特定的强连通分量拓扑，才能实现最佳并行性能。相比之下，我们的并行实现利用了多智能体和任务因子分解，而这两者始终存在于我们的问题中。

[11] 中的方法旨在降低多智能体马尔可夫决策过程 [5] 中的冗余复杂性，适用于智能体不协作完成任务、而是由智能体各自最优完成分配给自己的任务的问题。我们考虑经典的随机分配问题，其中智能体只能独立地处理单个任务。[11] 中生成的模型不适合解决我们的问题，因为没有机制可以跟踪哪些智能体已被分配到特定任务。此外，通过将任务分配模型分散化，这项工作实现了相对于智能体数量和任务数量的线性可扩展性。

7 结论

本文提出了一种方法，用于解决多目标约束下多智能体系统（MAS）中的随机任务同时分配与规划问题。我们证明了该问题具有凸性，并且可以在多项式时间内求解；同时，最优随机分配和调度器可以以分散式方式计算。我们提供了一个 GPU-CPU 混合的多目标模型检测框架，能够最优地管理 GPU 设备和多个 CPU 核心上的计算负载。我们开展了两项实验，结果表明，将问题分散化可以实现并行实现，从而获得线性扩展能力并显著缩短运行时间。实验还表明，与概率模型检验器 Prism 和 Storm 相比，我们框架的多目标模型检测性能具有竞争力。未来的工作包括进一步优化实现，利用 CUDA 流缓解小型单智能体-任务模型的 PCI 瓶颈。我们还希望将 MORAP问题扩展为包含以 ω-正则时序性质和限制行为（例如平均收益）表示的任务。

参考文献

1. Baier, C., Hermanns, H., Katoen, J.P.: 《MDP模型检测的 10,000 个方面》。载于: 《计算与软件科学》, 第 420–451 页。Springer（2019） 2. Baier, C., Katoen, J.P.: 《模型检测原理》。麻省理工学院出版社, 剑桥, 马萨诸塞州（2008） 3. Birkhoff, G.: 《关于线性代数的三个观察》。塔库曼国立大学, 《Rev. Ser. A》**5**, 147–151（1946） 4. Blumofe, R.D., Leiserson, C.E.: 《通过工作窃取调度多线程计算》。美国计算机协会学报（JACM）**46**（5）, 720–748（1999） 5. Boutilier, C.: 《多智能体决策过程中的规划、学习与协调》。载于: 第 6 届理性与知识理论方面会议论文集, 第 195–210 页（1996） 6. Boyd, S., Boyd, S.P., Vandenberghe, L.: 《凸优化》。剑桥大学出版社（2004） 7. Chatterjee, K., Majumdar, R., Henzinger, T.A.: 《具有多目标的马尔可夫决策过程》。载于: 计算机科学理论方面年度研讨会, 第 325–336 页。Springer（2006） 8. Chowdhury, R., Navsalkar, A., Subramani, D.: 《随机动态环境中的 GPU加速多目标最优规划》。海洋科学与工程学报**10**（4）, 533（2022） 9. Delgrange, F., Katoen, J.P., Quatmann, T., Randour, M.: 《多目标 MDP 中的简单策略》。载于: 《系统构造与分析工具和算法: 第 26 届国际会议, TACAS 2020, 作为欧洲软件理论与实践联合会议 ETAPS 2020 的一部分举办, 爱尔兰都柏林, 2020 年 4 月 25–30 日, 论文集, 第 I 部分 26》, 第 346–364 页。Springer（2020） 10. Etessami, K., Kwiatkowska, M., Vardi, M.Y., Yannakakis, M.: 《马尔可夫决策过程的多目标模型检测》。载于: 系统构造与分析工具和算法国际会议, 第 50–65 页。Springer（2007） 11. Faruq, F., Parker, D., Laccrda, B., Hawes, N.: 《不确定性下的同步任务分配与规划》。载于: 2018 年 IEEE/RSJ 国际智能机器人与系统会议（IROS）, 第 3559–3564 页。IEEE, 马德里（2018 年 10 月） 12. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: 《概率系统的自动验证技术》。载于: 《永恒网络形式化方法》**6659**, 60–120（2011） 13. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D., Qu, H.: 《概率系统的定量多目标验证》。载于: 系统构造与分析工具和算法国际会议, 第 112–127 页。Springer（2011） 14. Forejt, V., Kwiatkowska, M., Parker, D.: 《概率模型检测的帕累托曲线》。载于: 自动化验证与分析技术国际研讨会, 第 317–332 页。Springer（2012） 15. Hahn, E.M., Hashemi, V., Hermanns, H., Lahijanian, M., Turrini, A.: 《区间马尔可夫决策过程的多目标鲁棒策略综合》。载于: 定量评估系统国际会议, 第 207–223 页。Springer（2017） 16. Hartmanns, A., Junges, S., Katoen, J.P., Quatmann, T.: 《MDP 中的多成本有界权衡分析》。自动推理学报**64**（7）, 1483–1522（2020）

17. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.：概率模型检验器Storm。软件技术转移国际期刊**24**(4)，589–610（2022） 18. Kuhn, H.W.：指派问题的匈牙利算法。海军研究后勤季刊**2**(1-2)，83–97（1955） 19. Kupferman, O., Vardi, M.Y.：安全属性的模型检测。系统设计形式化方法**19**(3)，291–314（2001） 20. Kwiatkowska, M., Norman, G., Parker, D.：PRISM 4.0：概率实时系统的验证。载于：Gopalakrishnan, G., Qadeer, S.（编）计算机辅助验证第23届国际会议（CAV'11）论文集。LNCS，第6806卷，第585–591页。Springer（2011） 21. Kwiatkowska, M., Norman, G., Parker, D.：概率模型检验与自主性。控制、机器人与自主系统年度评论**5**(1)，385–410（2022年5月 010947 22 10 1146 042820）。https://doi.org/ . / annurev-control- - 。Matsakis, N.D., Klock, F.S.：Rust语言。ACM SIGAda Ada通讯**34**(3)，103–104（2014） 23. Merschformann, M., Xie, L., Li, H.：Rawsim-o：机器人移动履行系统的仿真框架。物流研究**11(1)**（2018） 24. Papadimitriou, C.H., Yannakakis, M.：关于权衡的可近似性与网络资源的最优访问。载于：第41届计算机科学基础年度研讨会论文集。第86–92页。IEEE（2000） 25. Pierce, B.C.：类型与编程语言高级主题。MIT出版社（2004） 26. Puterman, M.L.：马尔可夫决策过程：离散随机动态规划。John Wiley & Sons（2014） 27. Quatmann, T., Katoen, J.P.：长期平均值与总奖励的多目标优化。载于：系统构造与分析工具和算法：第27届国际会议，TACAS 2021，作为欧洲软件理论与实践联合会议ETAPS 2021的一部分举行，卢森堡卢森堡市，2021年3月27日至4月1日，论文集，第一部分27。第230–249页。Springer（2021） 28. Robinson, T., Su, G.：基于GPU-CPU混合加速的多目标任务分配和多智能体规划，https://github.com/tmrob2/hybrid-motap/ blob/master/GPU_MOTAP_NFM23_LONG.pdf 29. Roijers, D., Scharpff, J., Spaan, M., Oliehoek, F., De Weerdt, M., Whiteson, S.：不确定性下线性多目标规划的有界近似。载于：国际自动化规划与调度会议论文集。第24卷，第262–270页（2014） 30. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.：多目标序贯决策综述。人工智能研究期刊**48**，67–113（2013） 31. Sapio, A., Bhattacharyya, S.S., Wolf, M.：使用并行化稀疏矩阵在图形处理器上高效求解马尔可夫决策过程。载于：2018年信号与图像处理设计与架构会议（DASIP）。第13–18页。IEEE（2018） 32. Schillinger, P., B¨urger, M., Dimarogonas, D.V.：异构多机器人系统中面向时序逻辑目标的任务分配与规划同步进行。机器人研究国际期刊**37**(7)，818–838（2018） 33. Ulungu, E.L., Teghem, J.：多目标组合优化问题：综述。多准则决策分析期刊**3**(2)，83–104（1994年8月）。https://doi.org/10.1002/mcda.4020030204

34. Wu, Z., Hahn, E.M., G¨unay, A., Zhang, L., Liu, Y.：用于计算马尔可夫决策过程（MDP）中可达概率的 GPU 加速值迭代。载于：ECAI 2016，第 1726–1727 页。IOS出版社（2016）35. Wurman, P.R., D'Andrea, R., Mountz, M.：在仓库中协调数百辆协作型自主车辆。《AI Magazine》**29**(1)，9–9（2008）

A 补充材料与证明

A.1 余安全线性时序逻辑的形式化定义

语义关系记为 $\sigma \models \varphi$，适用于任意 $\sigma \in \Sigma^\omega$ 和任意线性时序逻辑公式 φ，其定义是标准的。对于任意 $\varsigma \in \Sigma^*$ 和 $\sigma \in \Sigma^\omega$，令 $\varsigma \cdot \sigma$ 表示 ς 与 σ 的串接。若满足以下条件，则 φ 是安全的：存在 Σ^* 的一个子集，记为 $pref_{bad}(\varphi)$，使得若 $\varsigma \in pref_{bad}(\varphi)$，则对所有 $\sigma \in \Sigma^\omega$，都有 $\varsigma \cdot \sigma \in \Sigma^\omega \setminus \models \varphi$；若满足以下条件，则 φ 是余安全的：存在 Σ^* 的一个子集，记为 $pref_{good}(\varphi)$，使得若 $\varsigma \in pref_{good}(\varphi)$，则对所有 $\sigma \in \Sigma^\omega$，都有 $\varsigma \cdot \sigma \models \varphi$。

引理 3 ([19])。对于任意余安全线性时序逻辑公式 φ，存在一个确定性有限状态自动机 $\mathcal{A}$，其接受位置均为汇点，使得 $pref_{good}(\varphi) = acc(\mathcal{A})$。

A.2 几何与随机矩阵补充材料

回顾一下，对于某个 m，集合 $C \subseteq \mathbb{R}^m$ 是一个凸多胞体，当且仅当它是某个有限向量集的所有凸组合构成的集合。凸多胞体的面C是一个子集 $H \subseteq C$，满足存在一个向量 $\boldsymbol{v}$，使得对于所有$\boldsymbol{u} \in H$，$\boldsymbol{u'} \in C$都有 $\boldsymbol{u} \cdot \boldsymbol{v} \geq \boldsymbol{u'} \cdot \boldsymbol{v}$。我们回顾凸多胞体的以下性质（这也是面对面的另一种定义）。

引理 4（[14]）。设 C 是一个凸多胞体。对于任意向量 $\boldsymbol{v}$，存在一个面H，使得 $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u'} \cdot \boldsymbol{v}$，并且对于所有 $\boldsymbol{u}$、$\boldsymbol{u'} \in H$ 和 $\boldsymbol{u''} \in C \setminus H$，都有 $\boldsymbol{u} \cdot \boldsymbol{v} > \boldsymbol{u''} \cdot \boldsymbol{v}$。

我们还回顾双随机矩阵的一项众所周知的性质。

引理 5. 给定 $\boldsymbol{U} \in \mathbb{R}^{n \times n}$，最大化 $\sum_{1 \leq i,\ j \leq n} u_{i,\ j} x_{i,\ j}$ 且满足 $\boldsymbol{X}$ 为双随机矩阵这一条件的问题，存在一个最优解 $\boldsymbol{X}^*$，它是一个置换矩阵。

A.3 集中式马尔可夫决策过程的证明

我们在第 3.2 节给出定理 2 的完整证明。

证明（定理 *2*）。假设对每个 $i \in I$，$j \in J$，均存在一个双随机矩阵 $(x_{i,\ j})_{i \in I,\ j \in J}$以及 $Sch(\mathcal{M}_{i \otimes j})$，使得满足定义 1 中的不等式。令 $y_{i,\ j,\ \sharp} = x_{i,\ j} / (x_{i,\ j} + \sum_{i' > i,\ i' \notin \sharp} x_{i',\ j})$。令 $\mu^{\mathrm{ct}} \in Sch(\mathcal{M}^{\mathrm{ct}})$，使得对于所有配对 $(i,\ j)$，$\in I \times J$ 且 $\sharp \subseteq I$:

– $\mu^{\mathrm{ct}}(s_i, q_j, \sharp)(a) = \mu_{i,j}(s_i, q_j)(a)$ 对所有 $a \in A_i(s_i)$，若 $i \in \sharp$ 和 $q_j \notin Q_{j,F} \cup Q_{j,R}$– $\mu^{\mathrm{ct}}(s_i, q_j, \sharp)(b_3)$，若 $i \in \sharp$ 和 $j \in Q_{j,F} \cup Q_{j,R}$– $\mu^{\mathrm{ct}}(s_i,0,q_j,0,\sharp)(b_1) = y_{i,j,\sharp}$ 且 $\mu^{\mathrm{ct}}(s_i,0,q_j,0,\sharp)(b_2) = 1 - y_{i,j,\sharp}$，若 $i \notin \sharp$

μ^{ct} 是一个定义良好的调度器，并满足定理 2 中的条件。

反过来，假设存在 $\mu^{\mathrm{ct}} \in Sch(\mathcal{M}^{\mathrm{ct}})$，使得定理 *2* 中的条件成立。令 $I^{-i} = \{i' \in I \mid i' \neq i\}$。对于每个 $(i, j) \in I \times J, \sharp \subseteq I^{-i}$，令$x_{i,j,\sharp} = \mu^{\mathrm{ct}}(s_{i,0}, q_{j,0}, \sharp)(b_1) z_{i,j,\sharp}$，其中 $z_{i,j,\sharp}$ 是在 μ^{ct} 下，$\mathcal{M}^{\mathrm{ct}}$ 中到达 $(s_{i,0}, q_{j,0}, \sharp)$ 这一元组的概率，并令 $x_{i,j} = \sum_{\sharp' \subseteq I^{-i}} x_{i,j,\sharp'}$。调度器 $\mu_{i,j} \in Sch(\mathcal{M}_{i \otimes j})$ 定义如下：$\mu_{i,j}(s, q)(a) = \sum_{\sharp \subseteq I^{-i}} \mu^{\mathrm{ct}}(s, q, \sharp)(a) x_{i,j,\sharp}$

对于所有 $(s,q) \in S_i \times Q_j$ 和 $a \in A_i(s)$。矩阵 $(x_{i,j})_{i\in I, j\in J}$ 是双随机的。对此，一个（非正式的）论证如下：令 π 为 $\mathcal{M}^{\text{ct}}$ 中任意一个从其初始状态 $(s_1,0,q_1,0,\emptyset)$ 出发的对象。对于每个 $i \in I$, π，都会恰好一次经过 $(s_i,0,q_j,0,\natural)b_1(s_i,0,q_j,0,\natural \cup \{i\})$，其中某个 $j \in J$。对于每个 $j \in J$, π，都会恰好一次经过 $(s_i,0,q_j,0,\natural)b_1(s_i,0,q_j,0,\natural \cup \{i\})$，其中某个 i，且其中某个 $i \in I$。因此，MORAP问题是可行的。□

我们给出 $\mathcal{M}^{\text{ct}}$ 的两个重要性质，这些性质将在后续证明中使用。令 $|I| = |J| = n$。回忆一下，$\mathscr{C}_0 \doteq \{(\mathbf{E}^{\mathcal{M}^{\text{ct}}[\rho_k],\mu})_{1\le k\le 2n} \mid \mu \in Sch(\mathcal{M}^{\text{ct}})\}$，并且 $\mathscr{C}$ 是 $\mathscr{C}_0$的向下闭包。奖励有限性假设意味着 $\mathscr{C}_0$ 非空且有界。第一个性质是多目标马尔可夫决策过程的一项基本性质。

引理 6（[10]）。$\mathscr{C}_0$ 是一个具有有限个面的凸多胞体。

在最坏情况下，$\mathscr{C}_0$ 中的面数相对于 $\mathcal{M}^{\text{ct}}$的大小以及目标数量（此处为$2n$ ）呈指数增长。

令 ρ （分别为$\rho_{i,\ j}$）为 $\mathcal{M}^{\text{ct}}$ （分别为 $\mathcal{M}_{i\otimes j}$）的奖励结构。若将 ρ 限制在$\rho_{i,\ j}$的定义域上，我们记$\rho \sim \rho_{i,\ j}$，当$\rho = \rho_{i,\ j}$ 时。

引理 7. 对于任意 ρ 、 $\mathcal{M}^{\text{ct}}$ 和 $\mu \in Sch(\mathcal{M}^{\text{ct}})$，存在一个双随机矩阵 $(x_{i,j})_{i\in I,j\in J}$ 和 $\mu_{i,j} \in Sch(\mathcal{M}_{i\otimes j})$ 使得

$$\mathbf{E}^{\mathcal{M}^{\text{ct}}[\rho],\mu} = \sum_{1\le i,j\le n} x_{i,j}\mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_{i,j}],\mu_{i,j}} \tag{1}$$

其中 $\rho \sim \rho_{i,j}$ 对所有 $1 \le i,j \le n$ 均成立。此外，存在一个置换矩阵$(x_{i,j})_{i\in I,j\in J}$ 使得

$$\max_{\mu\in Sch(\mathcal{M}^{\text{ct}})} \mathbf{E}^{\mathcal{M}^{\text{ct}}[\rho],\mu} = \sum_{1\le i,j\le n} x_{i,j}\mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_{i,j}],\mu^*_{i,j}} \tag{2}$$

其中 $\mu_{i,\ j} = \arg\max_{\mu'} \mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_{i,j}],\ \mu'}$ 且 $\rho \sim \rho_{i,\ j}$对所有 $1 \le i,j \le n$ 均成立。

证明。首先，我们按照定理 3 证明的第二部分，为所有 $1 \le i,j \le n$ 构造双随机矩阵 $(x_{i,j})_{i\in I,j\in J}$ 和 $\mu_{i,j} \in Sch(\mathcal{M}_{i\otimes j})$。式 (1) 可通过离散时间马尔可夫链的标准概率模型检测方法以及期望总奖励 [2] 推导得出。此外，由于每个 $x_{i,j}$ 均为非负数，要最大化 $\mathbf{E}^{\mathcal{M}^{\text{ct}}[\rho],\mu}$，我们需要对所有 i,j 最大化 $\mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_{i,j}],\mu_{i,j}}$。我们可以对所有 i,j 固定 $\mu^*_{i,j} = \arg\max_{\mu'} \mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_{i,j}],\mu'}$；根据引理 5，存在一个置换矩阵 $(x_{i,j})_{i\in I,j\in J}$，使 $\sum_{1\le i,j\le n} x_{i,j}\mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho_{i,j}],\mu^*_{i,j}}$ 最大化。□

A.4 算法1和算法2的证明

引理 8. $\boldsymbol{w}$ 在算法 *1* 第 *6* 行中计算得出，是一个权重向量，即 $\boldsymbol{w} \ge 0$ 和$\|\boldsymbol{w}\|_1 = 1$。

证明。回顾一下，$\|\boldsymbol{x}\|^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle = \boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}$。令 $\boldsymbol{w}_\perp = \boldsymbol{M}(\boldsymbol{t} - \boldsymbol{t}_\uparrow)$。由于 $\boldsymbol{t} \neq \boldsymbol{t}_\uparrow$，$\boldsymbol{w}_\perp \neq \boldsymbol{0}$（否则 $(\boldsymbol{t} - \boldsymbol{t}_\uparrow)^T \boldsymbol{M}(\boldsymbol{t} - \boldsymbol{t}_\uparrow) = 0$，这违反了 $\boldsymbol{M}$的正定性）。由于 $\|\boldsymbol{t} - \boldsymbol{t}_\uparrow\|$ 是 $\boldsymbol{t}$ 与 $down(\Phi)$ 之间的距离，集合$\{\boldsymbol{x} \mid \boldsymbol{w}_\perp \cdot (\boldsymbol{x} - \boldsymbol{t}_\uparrow) = 0\}$ 是 $\{\boldsymbol{t}\}$ 与 $down(\Phi)$ 之间的分离超平面，也就是说，对于所有 $\boldsymbol{y} \in down(\Phi)$，$\boldsymbol{w}_\perp \cdot (\boldsymbol{y} - \boldsymbol{t}_\uparrow) \leq 0$。由于 $down(\Phi)$ 无下界，$\boldsymbol{w}_\perp \geq 0$，并且 $\boldsymbol{w}_\perp$ 中至少存在一个元素 w_i 满足 $w_i > 0$。因此，$\boldsymbol{w} = \boldsymbol{w}_\perp / \|\boldsymbol{w}_\perp\|_1$ 是一个权重向量。

引理 9。*算法 2 是正确的，即，$\boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w} \cdot \boldsymbol{r}$ 定义了 $\mathscr{C}$ 的一个支撑超平面，其中 $\boldsymbol{w}$ 是在算法 1 的第 6 行计算得到的，而 $\boldsymbol{r}$ 由算法 2 返回。*

证明。令 $\boldsymbol{u} \in \mathscr{C}_0$ 为中的任意向量，并令 $\boldsymbol{u} = \mathbf{E}^{\mathcal{M}^{\mathrm{ct}}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\mu_0}$，$\mu_0 = (\mu_{0\ 1\leq k\leq 2n})$。根据引理 8，$\mathcal{M}^{\mathrm{ct}}$ 是一个权重向量。令 $\mu^*_{i,\ j} = \arg\max_{\mu_{i,\ j}} \mathbf{E}^{\mathcal{M}^{\mathrm{ct}}[\boldsymbol{w}\cdot\boldsymbol{\rho}]}{}_{\mu_{i\ j}}$。对于所有 $1 \leq i,\ j \leq n$，其中 $\boldsymbol{\rho} = \{\rho_i\}_{1\leq i\leq 2n}$。于是，

$$
\begin{aligned}
& \boldsymbol{w} \cdot \boldsymbol{u} \\
&= \mathbf{E}^{\mathcal{M}^{\mathrm{ct}}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\mu_0} \\
&\leq \max_{\mu} \mathbf{E}^{\mathcal{M}^{\mathrm{ct}}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\mu} \\
&= \textstyle\sum_{i,j} x_{i,j} \mathbf{E}^{\mathcal{M}_{i\otimes j}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\mu^*_{i,j}} && \text{(for some permutation matrix } (x_{i,j})_{1\leq i,j\leq n}\text{;} \\
& && \text{c.f. Eq. (2) in Lemma 7)} \\
&\leq \textstyle\sum_{i,j} \mathrm{I}_{i=f(j)} \mathbf{E}^{\mathcal{M}_{i\otimes j}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\mu^*_{i,j}} && \text{(according to def. of } f\text{)} \\
&= \boldsymbol{w} \cdot \boldsymbol{r}
\end{aligned}
$$

最后一个等式也证实了 $\boldsymbol{r} \in \mathscr{C}_0 \subset \mathscr{C}$（即，$\boldsymbol{r}$ 是一个可行的阈值）。 □

现在我们给出定理 3 的完整证明。

证明（定理 3）。我们首先证明算法1的终止性。假设算法1的 ℓ^{th} 次迭代已针对任意 $\ell > 1$完成。根据引理 8，$\boldsymbol{w} \geq 0$是一个权重向量。直观地说，该算法为 $\boldsymbol{t}_\uparrow$（分别为 $\boldsymbol{t}_\downarrow$）找到一系列数值，使其朝向（分别远离）$\boldsymbol{t}$ 移动，并最终以 $\|\boldsymbol{t}_\uparrow - \boldsymbol{t}_\downarrow\| \leq \varepsilon$终止。形式化证明依赖于以下两个论断。

断言。如果 $\boldsymbol{w} \cdot \boldsymbol{r} > \boldsymbol{w} \cdot \boldsymbol{t}_\uparrow$，那么 $\boldsymbol{r}$ 位于 $\mathscr{C}_0$ 的一个新面上。

实际上，$\boldsymbol{w} \cdot \boldsymbol{r} > \boldsymbol{w} \cdot \boldsymbol{t}_\uparrow$ 意味着对于所有 $\boldsymbol{u} \in down(\Phi \backslash \{\boldsymbol{r}\})$，都有 $\boldsymbol{w} \cdot \boldsymbol{r} > \boldsymbol{w} \cdot \boldsymbol{u}$（因为 $\{\boldsymbol{u} \in \mathbb{R}^{2n} \mid \boldsymbol{w} \cdot \boldsymbol{u} = \boldsymbol{w} \cdot \boldsymbol{t}_\uparrow\}$ 是 $down(\Phi \backslash \{\boldsymbol{r}\})$ 的一个支撑超平面）。因此，在此条件下，根据引理 9 和引理 4，存在 $\mathscr{C}_0$ 的一个面 H，使得 $\boldsymbol{r} \in H$ 且对于所有 $\boldsymbol{r}' \in \Phi \backslash \{\boldsymbol{r}\}$都有 $\boldsymbol{r}' \notin H$；换言之，$\boldsymbol{r}$ 位于 $\mathscr{C}_0$的一个新面上。

断言。若 $\boldsymbol{w} \cdot \boldsymbol{r} \leq \boldsymbol{w} \cdot \boldsymbol{t}_\uparrow$，则 $\boldsymbol{t}_\downarrow = \boldsymbol{t}_\uparrow$。

实际上，根据 $\boldsymbol{t}_{\uparrow} \in down(\Phi)$，由引理 9 可知，对于所有 $(\boldsymbol{w}', \boldsymbol{r}')$，$\boldsymbol{w}' \cdot \boldsymbol{r}' \geq \boldsymbol{w}' \cdot \boldsymbol{t}_{\uparrow} \in \Lambda$。因此，条件 $\boldsymbol{w} \cdot \boldsymbol{r} \leq \boldsymbol{w} \cdot \boldsymbol{t}_{\uparrow}$ 等于 $\boldsymbol{w} \cdot \boldsymbol{r} = \boldsymbol{w} \cdot \boldsymbol{t}_{\uparrow}$。在此条件下，$\boldsymbol{t}_{\uparrow}$ 和 $\boldsymbol{t}_{\downarrow}$ 是满足以下条件的唯一向量：

$$\begin{aligned}
\boldsymbol{t}_{\uparrow} &= \underset{\boldsymbol{z} \in \mathbb{R}^{2n}, \boldsymbol{w} \cdot \boldsymbol{t}_{\uparrow} = \boldsymbol{w} \cdot \boldsymbol{z}}{\arg\min} \|\boldsymbol{t} - \boldsymbol{z}\| \\
&= \underset{\boldsymbol{z} \in \mathbb{R}^{2n}, \boldsymbol{w} \cdot \boldsymbol{r} \geq \boldsymbol{w} \cdot \boldsymbol{z}}{\arg\min} \|\boldsymbol{t} - \boldsymbol{z}\| \\
&= \underset{\boldsymbol{z} \in \mathbb{R}^{2n}, \boldsymbol{w}' \cdot \boldsymbol{r}' \geq \boldsymbol{w}' \cdot \boldsymbol{z}, \forall (\boldsymbol{w}', \boldsymbol{r}') \in \Lambda}{\arg\min} \|\boldsymbol{t} - \boldsymbol{z}\| \\
&= \boldsymbol{t}_{\downarrow}
\end{aligned}$$

第二个等式成立，是因为 $\{\boldsymbol{x} \mid \boldsymbol{w} \cdot \boldsymbol{r} = \boldsymbol{w} \cdot \boldsymbol{x}\}$ 是 $\boldsymbol{t}$ 与 $\{\boldsymbol{z} \mid \boldsymbol{w} \cdot \boldsymbol{r} \geq \boldsymbol{w} \cdot \boldsymbol{z}\}$ 之间的分离超平面。

由上述第一个断言可直接得出，不等式 $\boldsymbol{w} \cdot \boldsymbol{r} > \boldsymbol{w} \cdot \boldsymbol{t}_{\uparrow}$ 不可能在无限多次迭代中成立。因此，该算法要么终止，要么在某次迭代中 $\boldsymbol{w} \cdot \boldsymbol{r} \leq \boldsymbol{w} \cdot \boldsymbol{t}_{\uparrow}$。如果 $\boldsymbol{w} \cdot \boldsymbol{r} \leq \boldsymbol{w} \cdot \boldsymbol{t}_{\uparrow}$，则上述第二个断言保证，对于任意 $\varepsilon \geq 0$，算法 1 都会终止。

性质 *(i)* 显然，因为 $\boldsymbol{t}_{\uparrow} \in down(\Phi) \subseteq \mathscr{C}$。观察可知，性质 *(ii)* 成立：如果 $\boldsymbol{t} \in \mathscr{C}$，那么算法1第 9 行的条件始终为假。对于性质 *(iii)*，算法1经过第一次迭代后，不等式 $\min_{\boldsymbol{u} \in \mathscr{C}} \|\boldsymbol{t} - \boldsymbol{u}\| \leq \min_{\boldsymbol{u} \in down(\Phi)} \|\boldsymbol{t} - \boldsymbol{u}\| = \|\boldsymbol{t} - \boldsymbol{t}_{\uparrow}\|$ 成立。另一方面，对所有 $\boldsymbol{u} \in \mathscr{C}$，$\|\boldsymbol{t} - \boldsymbol{t}_{\downarrow}\| \leq \|\boldsymbol{t} - \boldsymbol{u}\|$。该不等式成立，是因为任何 $\boldsymbol{u} \in \mathscr{C}$ 都满足第 10 行中的约束。因此，$\|\boldsymbol{t} - \boldsymbol{t}_{\downarrow}\| = \min_{\boldsymbol{u} \in \mathscr{C}} \|\boldsymbol{t} - \boldsymbol{u}\|$。□

证明（推论 *1*）。性质 (i)，即 $\boldsymbol{t}_{\uparrow} = \boldsymbol{t}_{\downarrow}$，可由终止条件直接得出。性质 (ii) 的一个方向即为定理 3 中的性质 (ii)。对于另一个方向，假设 $\boldsymbol{t}_{\downarrow} = \boldsymbol{t}$。那么，$\boldsymbol{t} = \boldsymbol{t}_{\uparrow} \in down(\Phi) \subseteq \mathscr{C}$。对于性质 (iii)，若 $\boldsymbol{t} \notin \mathscr{C}$，则 $\boldsymbol{t}_{\uparrow} = \boldsymbol{t}_{\downarrow}$，而定理 3 中的性质 (iii) 表明$\boldsymbol{t}_{\downarrow}$ 位于 $\mathscr{C}$的帕累托曲线上。

算法2中的策略迭代和值迭代

计算第 2 行的方法以及计算算法2中第 5 行和第 6 行的方法分别见算法 3 和算法 4。

算法 3： 通过策略迭代计算算法 2 中的第 2 行

输入： $\mathcal{M}_{i\otimes j}$, $\boldsymbol{w}$, $\boldsymbol{\rho}$, $\varepsilon_1 > 0$

输出： $\mu = \operatorname{argmax}_{\mu} \mathbf{E}_{\mathcal{M}_{i\otimes j}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\ \mu}$, $\mathbf{E}_{\mathcal{M}_{i\otimes j}[\boldsymbol{w}\cdot\boldsymbol{\rho}],\ \mu}$

初始化 μ;
$\boldsymbol{x} := \mathbf{0}$; $\boldsymbol{y} := \mathbf{0}$;
策略稳定：= true;
当 *not* 策略-稳定**时**
  对于每个 $(s,q) \in S_{i,j}$ **执行】【**。
    $y_{s,q} :=$ 最大值$_{a\in A_i(s)}[(\boldsymbol{w}\cdot\boldsymbol{\rho})(s,q,a) + \sum_{(s',\ q')\in S_{i,\ j}} P_{i,\ j}(s,\ q,\ a,\ s',\ q')\cdot x_{s',\ q'}]$;
    如果 $|y_{s,\ q} - x_{s,\ q}| > \varepsilon_1$ **则**
      策略稳定：= false;
      μ（s，q）：$= a$ 来自第 6 行;
    $x_{s,q} := y_{s,q}$;
返回 μ，$y_{s_{j,\ 0},\ q_{j,\ 0}}$

算法 4： 通过值迭代计算算法 2 中的第 5 行和第 6 行

输入： $\mathcal{M}_{i\otimes j}$, $\mu_{i,\ j}$, ρ, $\varepsilon_2 > 0$

输出： $\mathbf{E}^{\mathcal{M}_{i\otimes j}[\rho]}_{,\ \mu_{i,j}}$

$\boldsymbol{x}$：$= \mathbf{0}$；$\boldsymbol{y}$：$= \mathbf{0}$;
值稳定 := 真;
当 *not* 值-稳定**执行**
  对每个 $(s,q) \in S_{i,j}$ **执行**
    $y_{s,q}$：$= \rho_{(s,\ q,\ \mu_{i,\ j}(s,\ q)\ )} + \sum_{(s',\ q')\in S_{i,\ j}} P_{i,j}(s,q,\mu_{i,j}(s,q)\ s',q')\ \cdot x_{s',q'}$;
    如果 $|y_{s,q} - x_{s,q}| > \varepsilon_2$ **则**
      值稳定：= 假;
    $x_{s,q} := y_{s,q}$;
返回$y_{s_{i,0},q_{j,0}}$