

编号 \_\_\_\_\_

# 中国运筹学会科学技术奖 运筹应用奖申报表

申报项目 绿色计算下的资源分配和任务调度算法

申报人 张国川


所在单位 浙江大学

电子邮件地址 zgc@zju.edu.cn

中 国 运 筹 学 会 制

# 填 表 说 明

1. 本表需打印完成，可到中国运筹学会网站（[www.orsc.org.cn](http://www.orsc.org.cn)）下载。
2. 申报单位：委托或承担本项目的单位。
3. 封面编号由中国运筹学会应用奖评奖委员会统一填写。
4. 项目预期目标：项目合同规定的目标，如拟解决的问题，预期达到的经济或社会效益指标。
5. 技术方案概述：指项目内涵的运筹学问题，难度和挑战，采取的分析方法或建立的模型，技术创新点，取得的应用效果等。
6. 主要成果和贡献：如解决了什么实际问题，提高了系统效率，提高了应用单位的成本效益等。
7. 效益评价：指可以证明的经济效益或社会效益。
8. 应用单位意见：应用单位对本项目的评语，对成果的客观评价。
9. 表格中未包括的需说明的事项，可另文报送。

申报人		张国川	身份证号码	510102196605098473	
学历		博士研究生	学位	博士	
专业专长		运筹学	专业技术职务	教授	
单 位	名 称	浙江大学			
	通讯地址	浙江省杭州市西湖区浙大路 38 号浙江大学玉泉校区计算机科学与技术学院	邮码	310013	
	联系电话		传真		
	电子信箱	zgc@zju.edu.cn	手机	13082858709	
项目委托单位（甲方）		中国计算机学会/蚂蚁集团			
项目执行单位（乙方）		浙江大学			
<b>项目预期目标</b>					
<p>近年来，随着云计算与人工智能行业的飞速发展，各大互联网企业在全全球建立的数据中心规模越来越大，其碳排放量也迅速增加，目前已占全球温室气体排放量的 4%，超越了全球航空业的 2.8%。以我国为例，2018 年中国数据中心能耗总量约为 1608 亿度，超过了上海当年的用电量，约合二氧化碳排放量 9485 万吨。预计 2035 年全国数据中心能耗将达到 4,500 亿度。2020 年 9 月，国家提出于 2030 年前碳达峰、2060 年前碳中和的“双碳”目标，这对云计算的行业的绿色化发展提出了紧迫要求。</p> <p>蚂蚁集团积极响应国家号召，于 2019 年开始探索绿色计算，并于 2021 年提出了 2030 年实现碳中和的大目标。前蚂蚁集团首席技术官倪行军表示，2019 年起蚂蚁集团的所有业务开始陆续上云，全面进入云时代。作为服务几亿用户的平台，每分钟都在对海量的数据进行处理和计算，蚂蚁集团不仅要追求计算的效率，也要评估计算的能耗。如何在云计算的时代，做到“绿色计算”，成为一个重要的探索方向。</p> <p>在此背景下，本项目针对云计算中的资源分配和任务调度展开研究，建立运筹优化模型，提出高效（秒级）的求解算法，将计算资源的利用率提高 20%以上。</p>					

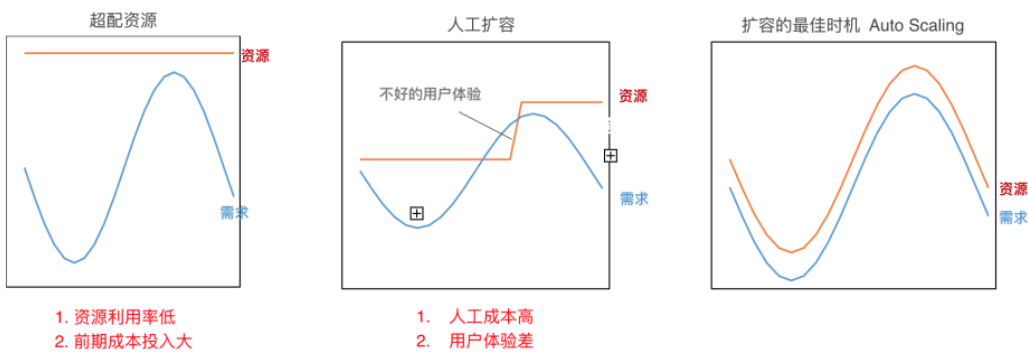
# 项目技术方案概述

## 1. 项目内涵的运筹学问题

云计算的核心思想是通过网络将计算资源、存储资源等以服务的形式提供给用户，实现资源共享和综合利用。对大规模的用户需求进行合理分配调度，可有效提高资源利用率、降低计算成本。本项目主要考虑计算资源配置，服务部署和任务迁移三个关键问题。

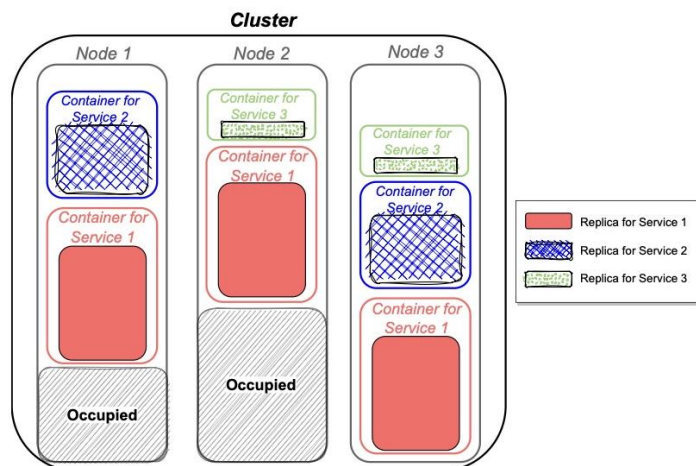
### 1.1 资源配置

云计算集群中的工作负载（资源需求）是不断变化的，而资源的供给往往有一定的滞后性，因此我们需要提前制定资源配置方案以适应需求的变化。初代的资源配置方案以超配资源（按最大需求来配置）或人工扩容为主，但是这样很容易产生资源浪费或损害用户体验。人工智能技术的进步使得需求预测成为可能。我们很自然地有以下问题：如何利用这些预测值来制定一个鲁棒和高效的资源配置方案？如何在保证用户体验和满足系统约束的前提下尽可能提升资源利用率？



### 1.2 服务部署

在多个服务（Service）混合部署的大规模集群中，例如支持蚂蚁营销推荐业务的集群组，资源管理以“集群（Cluster）→ 物理机（Node）→ 容器（Container）”的梯级层次形式进行（如下图所示）。在得到每个服务在不同时段资源配置之后，我们需要若干一定规格的容器来支持这些服务。在这里，我们需要决定各个服务所使用的容器规格（每种色块的大小）和数量、容器在物理机上的部署方式，最终达到节约物理机数量、实现降低成本、提升资源利用率的目的。



### 1.3 任务迁移

在蚂蚁自研的分布式数据库 OceanBase 中，每个集群存在若干个服务器，每个服务器在多个资源维度（CPU、内存、硬盘等）上是不同质的。这些服务器上支持着若干任务，每个任务只能属于一个服务器，且任务在资源维度上也是不同质的。初始状态下，每个集群有一个已知的任务分布方式；随着时间的推移，新的任务（组）会被分配到集群，旧的任务会被释放，导致集群各服务器的资源利用率发生变化。任务迁移研究的主要问题即为，给定某集群的一种初始任务分布，通过任务在不同服务器上的迁移，叠加上服务器的扩缩容，使得服务器整体上达到较高的资源利用率，降低服务器成本。

## 2. 难度和挑战

- (1) 服务部署问题最多可以涉及数千个服务和数百台机器，实际的决策变量数量达到数十万级别，这使得高效求解变得困难。而在任务迁移问题中，我们还需要面对 7 个资源维度，这使得问题的难度进一步上升。
- (2) 在服务部署和任务迁移问题中，除了考虑成本和资源利用效率，还需要关注负载均衡度和用户体验等次要目标。在保证主要目标的同时兼顾这些次要目标也是本项目的挑战之一。
- (3) 出于业务的要求，我们需要在数秒内求解问题。考虑到问题规模，这样的要求基本排除了直接使用整数规划求解器的可能。另一方面，求解的质量直接关系到本项目的实际效益。因此，我们必须深入分析问题，利用问题性质结合问题的实际特点，设计有针对性的高效算法。

### 3. 采取的分析方法或建立的模型

#### 3.1 资源配置问题

首先考虑单一场景（云服务）的资源配置问题。每个时段的需求预测值以及由于预测值的不确定性所带来的鲁棒性要求，最终都可以转换每个时段的资源需求。因此，假定已知该场景在任意时段 $t$ 的资源需求 $d_t$ 。我们需要决定在每个时段分配给该场景的资源数量 $h_t$ ，既要保证能够满足每个时段的资源需求（ $h_t \geq d_t$ ），还要求资源数量的波动不可过大，即任意相邻时段的资源数量变化不能超过某个给定值 $\Delta$ 。优化目标有两个：（1）使用尽可能少的资源（ $\min \sum_t h_t$ ）；（2）资源波动尽可能平缓（ $\min \sum_t |h_{t+1} - h_t|$ ）。该多目标优化问题可用加权形式表示为下面的整数规划模型，其中 $\alpha$ 是平衡两个目标的参数。

$$\begin{aligned} \min \quad & \sum_{t=1}^T h_t + \alpha \sum_{t=1}^{T-1} |h_{t+1} - h_t| \\ \text{s.t.} \quad & h_t \geq d_t, \quad \forall t \in \{1, \dots, T\} \\ & |h_t - h_{t+1}| \leq \Delta, \quad \forall t \in \{1, \dots, T-1\} \\ & h_t \in \mathbb{Z} \quad \forall t \in \{1, \dots, T\} \end{aligned}$$

在单一场景的基础上，我们考虑多场景的资源配置问题。已知该任意场景 $i$ 在任意时段 $t$ 的资源需求 $d_t^i$ ，我们需要决定在每个时段配置给该场景的资源数量 $h_t^i$ 。对于每个场景，我们仍然需要满足前述两个约束（资源需求约束和资源波动约束）。同样的，我们希望使用尽可能少的资源。但是，由于不同场景的重要性不同，对于第二个目标，我们希望优先使重要场景的资源波动尽可能平缓。因此，每个场景都有一个权重 $w_i$ ，权重越大表示该场景越重要。第二个优化目标是 $\min \sum_i w_i \sum_t |h_{t+1}^i - h_t^i|$ 。多场景的资源配置问题仍然可以用整数规划表示。

$$\begin{aligned} \min \quad & \sum_{i=1}^n \sum_{t=1}^T h_t^i + \alpha \sum_{i=1}^n w_i \sum_{t=1}^{T-1} |h_{t+1}^i - h_t^i| \\ \text{s.t.} \quad & h_t^i \geq d_t^i, \quad \forall i \in \{1, \dots, n\}, \forall t \in \{1, \dots, T\} \\ & |h_t^i - h_{t+1}^i| \leq \Delta^i, \quad \forall i \in \{1, \dots, n\}, \forall t \in \{1, \dots, T-1\} \\ & h_t^i \in \mathbb{Z} \quad \forall i \in \{1, \dots, n\}, \forall t \in \{1, \dots, T\} \end{aligned}$$

#### 3.2 服务部署问题

已知 $n$ 个（云）服务和 $m$ 台已经被使用机器。每个服务 $i$ 的资源需求为 $d_i$ ，每台机器 $j$ 剩余的资源数为 $c_j$ 。除此之外，还有若干台未被使用的新机器，每台新机器可提供的资源数为 $c$ 。现在我们需要把服务部署到机器上。每个服务可以被均等地切割成若干个副本。为了保证容灾性，要求每个服务要被切割成至少两个副本且每个服务在一台机器上只能部署一个副本。我们希望在使用尽可能少的新机器的情况下来完成部署。在此基础上，我们还希望使副本总数尽可能少。该问题可使用以下规划描述。其中 $z_j$ 用于指示新机器 $j$ 是否被使用， $m_i$ 和 $y_i$ 代表服务 $i$ 被切割后的

副本数量和副本大小， $x_{i,j}$ 用于指示是否有服务 $i$ 的副本被放入机器 $j$ 中。

$$\begin{aligned}
 \min \quad & \sum_{j \in \mathcal{M}_{new}} z_j + \alpha \sum_{i \in \mathcal{I}} m_i \\
 \text{s.t.} \quad & m_i = \sum_{j \in \mathcal{M}_{new} \cup \mathcal{M}_{old}} x_{i,j} & \forall i \in \mathcal{I} \\
 & y_i \cdot m_i = d_i, & \forall i \in \mathcal{I} \\
 & m_i \geq 2, & \forall i \in \mathcal{I} \\
 & \sum_{i \in \mathcal{I}} y_i \cdot x_{i,j} \leq c_j, & \forall j \in \mathcal{M}_{old} \\
 & \sum_{i \in \mathcal{I}} y_i \cdot x_{i,j} \leq c, & \forall j \in \mathcal{M}_{new} \\
 & x_{ij} \leq z_j, & \forall j \in \mathcal{M}_{new}, i \in \mathcal{I} \\
 & z_j \in \{0, 1\}, & \forall j \in \mathcal{M}_{new} \\
 & x_{ij} \in \{0, 1\}, & \forall j \in \mathcal{M}_{new} \cup \mathcal{M}_{old}, i \in \mathcal{I}
 \end{aligned}$$

### 3.3 任务迁移问题

给定一个任务的集合 $I$ 和机器的集合 $M$ 。机器 $j$ 的费用为 $p_j$ 。我们用 $R$ 表示资源种类的集合，用 $w_{ir}$ 表示任务 $i$ 所消耗的资源 $r$ 的数量，用 $c_{jr}$ 表示机器 $j$ 所能提供的资源 $r$ 的数量。已知当前任务到机器的分配为 $\{x_{ij}\}_{i \in I, j \in M}$ ，我们需要找到一个新的任务到机器的分配 $\{y_{ij}\}_{i \in I, j \in M}$ 满足

- (1) 新分配所使用的机器的总成本尽可能小；
- (2) 存在一个从当前分配到新分配的转换路径，且路径上的所有分配都是可行的（即每个机器上任务所需的资源之和不会超过机器可提供的资源数量）。

我们使用两阶段法：首先只考虑（1），找出一个成本最小的新分配，然后再通过路径搜索算法找一条从当前分配到新分配的路径。这样极大程度地简化了求解过程，但是问题也是显而易见的：不能保证一定存在从当前分配到新分配的路径。因此，在求解新分配的过程中，我们加入了减少迁移步数的目标。这在一定程度上缓解了上述问题，提高了路径的寻得率。找新分配的问题可以用以下规划描述。其中 $z_j$ 用于指示机器 $j$ 是否被使用， $\{y_{ij}\}_{i \in I, j \in M}$ 我们希望求得的新分配。 $\{x_{ij}\}_{i \in I, j \in M}$ 代表已知的当前分配。

$$\begin{aligned}
 \min \quad & \sum_{j \in M} p_j z_j + \alpha \sum_{i \in I, j \in M} |y_{ij} - x_{ij}| \\
 \text{s.t.} \quad & \sum_{i \in I} w_{ir} y_{ij} \leq c_{jr} \quad \forall j \in M, \forall r \in R \\
 & \sum_{j \in M} y_{ij} = 1 \quad \forall i \in I \\
 & y_{ij} \leq z_j \quad \forall i \in I, \forall j \in M \\
 & y_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in M
 \end{aligned}$$

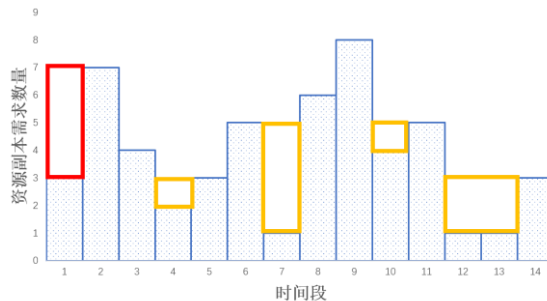
## 4. 方法和技术的创新点

本项目的成果兼顾了实际应用效果和算法理论价值。

- (1) 针对单场景资源配置问题，我们给出了一个易于实现的 $O(T)$ 时间的最优算法。这里 $T$ 是时段的数量。
- (2) 针对多场景资源配置问题，我们给出了一个易于实现的 $O(nT)$ 时间的最优算法。这里 $n$ 是场景数量， $T$ 是时段的数量。我们还在该问题求解过程中涉及的多重背包问题上取得了重要的理论进展。我们提出了针对(多重)背包问题的 $\tilde{O}(n + s^{2.4})$ 时间的最优算法和 $\tilde{O}(n + (1/\epsilon)^2)$ 的完全多项式时间近似方案(FPTAS)。这里 $n$ 是物品的数量， $s$ 为最大物品的 size。关于多重背包问题的两个理论结果分别发表在 SODA2024 和 STOC2024。
- (3) 针对服务部署问题，我们首先给出了一个只考虑所有使用机器数量的渐近近似比为 $e/(e - 1)$ 的高效算法。以这个算法为基础，我们设计了兼顾了容器数量等其他目标的启发式算法。
- (4) 针对任务迁移问题，我们通过分类和降维等手段，减小了整数规划的规模，在略微牺牲解的质量情况下，极大程度地提高了求解速度。

### 4.1 资源配置问题

对于单场景的资源配置问题我们给出了一个 $O(T)$ 时间的算法。我们首先求出一个使用资源数量最少的可行解(如下图蓝色部分)。这个可行解只要对所有时段的资源需求进行两次扫描就可以求出，因此只需要 $O(T)$ 时间。



然后将这个可行解扩充成最优解，这个扩充的过程实际上是用更多的资源去填充图中黄色和红色的方框，每填充一层， $\sum_t |h_t - h_{t+1}|$ 就会减少 1 (黄色框) 或者 2 (红色框)，而代价是所有资源总数 $\sum h_t$ 会增加且增加的数量等于方框的宽度。因此我们只要填充所有性价比超过 $\alpha$ 的方框就可以了。同时，我们还证明了最多只会有 $O(T)$ 个物品。因此，单场景的资源配置问题可以在 $O(T)$ 时间内求解。类似地，多场景的资源配置问题可以在 $O(nT)$ 时间内求解。

我们还考虑了以下问题：给定各个时段的资源需求以及有限的资源，找到一个可行且波动尽可能小的资源配置方案。我们仍然先找到一个使用资源数量最少的可行配置方案，然后我们



使用剩余的资源去填充方框，使整个配置方案变得平滑。填充方框的问题可以转换成多重背包问题。背包的容量等于剩余的资源数量。每个方框相当于一个物品，物品的 size 为方框的宽度，物品可以被选取的次数为方框的高度。每个物品的收益取决于场景的权重和方框的位置。虽然多重背包问题是 NP 困难的，但是我们注意到，这里每个物品的 size（即每个方框的宽度）不会超过  $T$ 。对于这一类多重背包问题，我们给出了一个易于实现的  $O(ns^2)$  时间的算法，其中  $n$  为物品的数量， $s$  为最大的物品 size。对应到我们的问题，我们有  $nT$  个物品，最大的 size 为  $T$ ，因此可以在  $O(nT^3)$  给出最优解。

随着对该问题研究的进一步深入，我们在算法理论方向也取得了重要进展。通过利用卷积技术和加性组合中的工具，我们提出了针对（多重）背包问题的  $\tilde{O}(n + s^{2.4})$  时间的精确算法和  $\tilde{O}(n + (1/\epsilon)^2)$  的完全多项式时间近似方案 (FPTAS)。特别地，我们的近似结果在 (min, +)-卷积假设下是最优的。

## 4.2 服务部署问题

该问题实际上是装箱问题的变形，我们称为等分装箱问题 (Equally-Split Bin Packing)：给定  $n$  个物品，每个物品有大小  $w_1, \dots, w_n$ ， $m$  个部分使用的箱子，每个箱子的剩余容量为  $c_1, \dots, c_m$ ，新箱子的容量为  $c$ 。我们需要将物品均匀地切割至少两块放入不同的箱子中，每个物品在每个箱子中最多放一块。目标是使用尽可能少的箱子并在此基础上尽可能减少物品的总切割数。注意到每个箱子的剩余容量各异，不存在所有物品平均分到每个箱子的平凡解。

首先仅考虑最小化总箱子数量的优化目标，从理论角度对问题进行了分析并设计了算法。通过将划分问题归约到等分装箱问题，我们证明了这个问题是 NP-困难的，且不存在近似比好于  $4/3$  的多项式时间近似算法。我们继而提出了基于贪心的快速算法：Merge-and-Split，将所有物品看作一个物品，计算将这个物品等分切割需要箱子数最少的切法。证明了这个算法的渐近近似比为  $e/(e-1)$ 。从理论上讲，该近似比在平凡下界的对比上是几乎最好的。

基于上述近似算法，提出了可以兼顾物品的总切割数的两个启发式算法框架：Pack-and-Adapt 和 Guess-and-Pack。这两个算法框架结合常用的装箱规则：First Fit, Worst Fit, Best Fit 等，可以适应不同数据分布下的问题。我们在公开数据集上对这几个算法进行实验。在这些数据集上，相较于求解整数规划，我们的启发式算法在求解速度上有 2-3 个数量级的提升，同时解的质量接近于最优解。

## 4.3 任务迁移问题

即便不考虑路径这一要求，寻找新分配已经是一个七维向量装箱问题。出于求解质量的考虑，我们无法完全脱离整数规划求解器来解决这一问题。但是，直接使用求解器又存在求解速度慢等问题。通过分析实际中问题的特点，我们利用物品分类和降维等手段，有效降低了规划

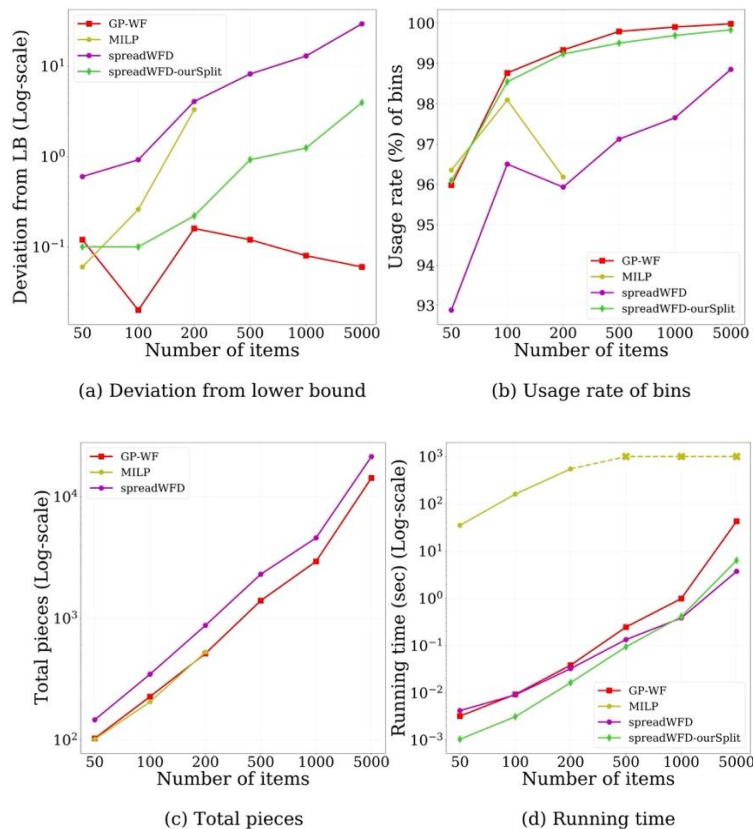
的规模，从而加快了求解速度。同时，得益于对迁移数量的限制，第二阶段在保证路径寻得率的前提下实现了更快的寻路速度。

## 5. 取得的主要应用效果

### 5.1 实验结果

资源配置问题的算法已经有很强的理论保证，因而我们主要对服务部署问题和任务迁移问题的算法进行了实验验证。

对于服务部署问题，实验数据来自阿里巴巴天池平台提供的公开数据集。我们分别从箱子数与下界的差、箱子利用率、物品切割数和运行时间这四个方面对算法进行了验证。首先对比了我们的两个算法框架（Pack-and-Adapt 和 Guess-and-Pack）与三种装箱规则（First Fit, Worst Fit, Best Fit）的组合。实验表明，Guess-and-Pack 和 Worst Fit 的组合（以下简称 GP-WF）拥有最好的实际表现。然后我们将 GP-WF、Gurobi 中混合整数规划求解器（MILP）以及 SpreadWFD 算法进行了对比，其中 SpradWFD 由 Mommessin 等人 (Affinity-aware resource provisioning for long-running applications in shared clusters. *Journal of Parallel and Distributed Computing* 177, 1–16 (2023)) 提出。由于 SpreadWFD 算法要求已知每个物品的切割数，我们分别将数据集中提供的切割数和我们算法输出的切割数作为算法输入进行了比较。实验结果如下图所示。



(1) 在求解速度上（图（d）所示），我们的算法 GP-WF 相较于 MILP 有 2-3 个数量级的提升。事实上，当物品数量超过 200 时，MILP 已经无法在合理时间内返回可行解，而我们的算法仍然可以在 1-20 秒之内返回。GP-WF 要略慢于 SpreadWFD，其主要原因在于 SpreadWFD 不需要决策每个物品的切割数。

(2) 在使用的箱子数量以及箱子利用率上（图（a）（b）所示），当物品数量小于 50 时，我们的算法 GP-WF 要略差于 MILP 的最优解。随着物品数量的增加，MILP 无法求得最优解，此时 GP-WF 要优于其他算法。特别地，GP-WF 要好于使用默认切割方案的 SpreadWFD。当 SpreadWFD 使用 GP-WF 所返回的切割方案时，其解的质量显著提升，但仍然要略微劣于 GP-WF。这也反映了 GP-WF 在物品切割和装箱两方面都有很好的决策效果。

(3) 在总切割数方面（图（c）所示），GP-WF 返回的解的物品总切割数，在小规模例子上接近于 MILP，在大规模例子上要优于默认的切割数。

对于服务迁移问题，我们在实际业务产生的数据集上选取了 1000 例规模较大的实例进行了验证，结果如下。可以看到，在几乎不增加机器成本且仅小幅牺牲其他指标的前提下，我们将求解时间降低了 72%。


	平均耗时（秒）	路径寻得率	迁移数指标	机器成本指标	均衡度指标
直接求解 IP	57.52	99%	907.5	4140.32	2520.64
我们的算法	9.93	97.5%	961.4	4140.83	2677.84

## 5.2 实际落地效果

(1) 资源配置算法：已集成到蚂蚁弹性扩缩容算法框架 OptScaler 中。相对于 AutoPilot 等业界基线技术方案，OptScaler 可减少超过 36% 的 SLO 违反度，节约了 19.5% 的机器资源

(2) 服务部署：研发的大规模 GP-WF 算法应用于支付宝内部的 30+ 应用上，部署优化的求解速度比商用求解器快 2~3 个量级，算法调度机器相对于基线可节约机器用量 30.4%。

(3) 任务迁移：研发的“一阶段线性规划求迁移终态、二阶段启发式求迁移路径”的二阶段运筹算法，求解的平均速度从基线的 10 分钟缩短至 20 秒以内；支持并发迁移负载，成倍提高了实际迁移效率；以此算法为基础的成本巡检功能，覆盖 4400+ 集群，累计指导节约 CPU 资源约 1 万核。

项目小组成员			
序号	姓名	单位	专业职称/职务
1	张国川	浙江大学	教授
2	毛宇尘	浙江大学	讲师
3	段亦超	浙江大学	硕士研究生
4	连佳宜	浙江大学	博士研究生
5	周义涵	浙江大学	硕士研究生
6	卢星宇	蚂蚁集团	高级算法专家
7	鲁炜	蚂蚁集团	高级算法工程师
8	邹丁	蚂蚁集团	高级算法工程师
应用单位意见：项目的成果、贡献、效益等			
<p>论文 1: A Nearly Quadratic-Time FPTAS for Knapsack, to appear in STOC '24</p> <p>论文 2: Resource Planning and Allocation, submitted to SoCC'24</p> <p>专利 1: 针对应用容器的资源调度方法及装置, 申请公布号: CN 117687794 A</p> <p>其它请见附录</p>			
声 明	<p>本人对申报表上述内容及全部附件材料的客观性和真实性负责。</p> <p style="text-align: right;">申报人签名: </p> <p style="text-align: right;">2024 年 5 月 28 日</p>		