doi: 10.19920/j.cnki.jmsc.2025.10.009

# 落地式分拣系统包裹分配与路径规划问题研究①

赫雪婷，镇 璐*，吴靓雯，高佳静
(上海大学管理学院，上海 200444)

**摘要：** 快递业务量的飞速增长导致分拣作业的复杂度大幅提高，推动了自动化分拣系统的迅速发展. 为了提升自动化分拣系统的灵活性、经济性以及可扩展性，落地式分拣系统应运而生. 本研究聚焦于落地式分拣系统中包裹分配以及自动引导车(Automated Guided Vehicle，AGV)无冲突路径规划问题，以最小化包裹在系统中的滞留时间为优化目标，建立了两阶段混合整数规划模型. 为了求解上述模型，设计了基于列生成的求解算法，并设计了变邻域搜索算法和 A* 算法加速模型的求解. 本研究通过大量数值实验验证了模型的有效性及算法的高效性，并基于敏感性分析实验提出一些管理启示.

**关键词：** 包裹分配；无冲突路径规划；时空网络模型；列生成；混合整数规划
**中图分类号：** TP18 **文献标识码：** A **文章编号：** 1007－9807(2025)10－0142－18

## 0 引 言

近年来，电子商务等行业的飞速发展促使快递业务量急剧增加. 2022 年，我国快递业务量高达 1 105 亿件，业务量连续 9 年位居世界第一[1]. 庞大的快递业务量、消费者对物流时效性的要求、传统分拣方式物流成本的不断增长、电子面单等信息技术的普及推动了自动化分拣系统的发展. 传统的手工分拣及半自动分拣人工成本高，分拣效率低，高峰期错分率高，已经难以满足当前快速攀升的快递业务量. 自动化分拣系统可实现 24 小时高效率、高准确率的分拣工作，极大的提升转运中心的包裹分拣能力. 邮政、顺丰、京东等多家头部快递公司均已开始部署并应用自动化分拣系统.

目前广泛应用的自动化分拣系统主要有两种：基于交叉带传送带分拣系统和基于 AGV 的分拣系统. 其中传送带式分拣系统主要适用于轻小件包裹的分拣，分拣效率高，但此类系统的主线轨道占地面积大，改造成本高，并且不易搬迁；此外，此类系统的分拣设备需要一次性成套投入，灵活性差，在业务量较小的场景中，经济性较低. 基于 AGV 的分拣系统，占地面积小，可根据业务量调整 AGV 数量，灵活性与经济性较高，但 AGV 集群工作会存在路径冲突的问题，从而降低分拣效率，并且随着 AGV 数量的增多，出现路径冲突的可能性增高，目前 AGV 调度系统难以处理分拣系统中大规模 AGV 的无冲突路径规划问题. 因此，综合来看，自动化交叉带传送带分拣系统适合于场地面积大，大规模包裹分拣的业务场景，AGV 分拣系统适合场地有限、中小规模包裹分拣的业务场景.

为解决当前分拣系统面临的难题，Geek + 研发了一种新型的自动化分拣系统—落地式分拣系统，该系统具有占地面积小，分拣效率高，可扩展性强，并且可灵活利用现有场地搭建自动化分拣系统，以满足不同规模的分拣需求等优势. 如图 1 所示，落地式分拣系统主要由传送带、分拣格口、AGV 以及笼车构成，进入系统的包裹首先由传送

① 收稿日期：2021－12－31；修订日期：2023－07－11.
基金项目：国家自然科学基金资助项目(72025103；72394360；72394362；72361137001).
通讯作者：镇 璐(1981—)，男，湖北宜都人，教授，博士生导师. Email：lzhen@shu.edu.cn

带运输到不同的分拣格口，到达分拣格口的包裹由 AGV 投递至对应的笼车中，整个系统不断运作，直至完成所有包裹的分拣任务. 在系统运作时，如何实现包裹与分拣格口、AGV 之间的合理匹配，规划 AGV 执行任务时的无冲突路径，成为亟需解决的问题.

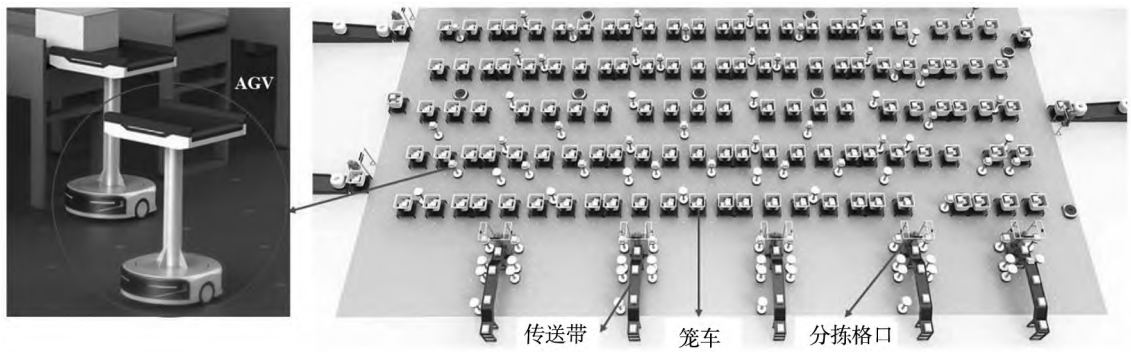


图 1　落地式分拣系统示意图(来源：Geek +)

Fig. 1 Schematic diagram of floor-based sorting system (Source: Geek +)

基于上述背景，本研究对落地式分拣系统中的运作优化问题进行了研究. 为了实现系统运作时资源的均衡利用，提高分拣效率，本研究建立了两阶段混合整数规划模型，分别求解包裹分配问题以及无冲突 AGV 路径规划问题；为了求解上述两个问题，设计并实现了基于列生成(column generation，CG)的求解算法；并根据两问题的结构特性，分别设计了变邻域搜索算法以及 $A^*$ 算法来加速子问题的求解；此外，本研究通过大量数值实验验证了两模型的有效性以及算法的高效性，并通过一些敏感性分析为系统的实际运作提供了一些管理启示.

## 1　文献综述

目前广泛应用的自动分拣系统有两类，基于传送带的分拣系统和基于 AGV 的分拣系统[2-4]. 基于传送带的分拣系统具有产能稳定，分拣效率高等优势，目前在自动化分拣系统领域占据主流地位. 现有学者对不同类型的分拣系统进行研究，对系统的分拣调度方案进行优化，以提升系统的分拣效率. 李珲等[5]介绍了一种基于传送带的新型包裹分拣系统，并根据系统的特点，构建了一个包裹定向分拣路径动态规划数学模型，随后设计改进的 Dijkstra 算法提升系统的分拣效率. 杨莹等[6]研究了基于传送带的自动分拣系统中的包裹分拣和路径规划问题，构建了包裹环境模型，设计了改进的快速搜索随机树，并通过仿真实验验证算法可有效提升包裹传输速度. Fedtke 和 Boysen[7]针对倾斜托盘分拣机系统进行了研究，并根据该系统的结构特性构建数学模型，分析了装载站数量、传送带运行方向、出入站目的地分配方案等因素对分拣系统吞吐量的影响. 镇璐等[8]对双层自动分拣系统进行研究，通过分析“成组分拣”方法和目的地指派问题对分拣效率的影响，建立了以最小化传送带的总分拣距离为目标的 0 - 1 规划模型，并设计变邻域禁忌搜索算法快速求解分拣调度方案.

基于 AGV 的分拣系统具有灵活性高、所需空间小，应用范围广等优势，正处于快速发展时期. 作为 AGV 分拣系统中的关键资源，AGV 的调度优化以及无冲突路径规划是系统运作过程中面临的难题. AGV 调度优化的核心科学问题本质上是车辆路径问题，Bae 和 Chung[9]研究了异质 AGV 的车辆路径问题，以最小化行驶距离为目标，为 AGV 规划任务序列，并设计了原始—对偶启发式方法求解该问题. Dang 等[10]研究了多负载异质 AGV 的调度优化问题，以最小化延迟服务成本和行驶成本的加权和为目标，在考虑 AGV 电池容量的基础上构建混合整数规划模型，并设计了混合自适应大邻域算法求解该问题. Polten 和 Emde[11]考虑了仓库狭窄巷道中的 AGV 调度问题，提出排他和并行两种访问策略，构建混合整数规划模型，并提出了一个大邻域搜索算法求解该问题. Boysen 等[12]对机器人分拣系统中订单分配以及机器人分配问题进行了深入研究，设计了可应

用于实时环境中的新型两步多场景优化方法，通过对现实案例的优化效果进行分析，提供了一系列决策建议.

在多 AGV 执行任务过程时，可能会出现路径冲突，从而降低系统分拣效率. 为解决 AGV 行驶过程中的冲突问题，Murakami[13] 考虑 AGV 运载能力、系统缓冲能力等因素，构建了一个基于时空网络的混合整数规划模型以求解 AGV 的无冲突路径，并提出一个有效不等式加速问题的求解，最后通过实例验证模型和有效不等式的有效性. 为提高 AGV 的运作效率，张新艳和邹亚圣[14] 以减少 AGV 行驶过程中的转弯次数为目标，提出一种引入时间因子的改进 $A^*$ 算法，并设计时间窗以及优先级策略以解决多 AGV 的路径冲突问题. 部分学者综合考虑了 AGV 的调度优化以及无冲突路径规划问题，李昆鹏等[15] 对“货到人”拣选系统进行了研究，为解决系统中多 AGV 任务分配以及路径规划问题，构建了两阶段数学模型，并基于货架优先级设计算法、设置 AGV 行驶规则及避撞机制分别求解 AGV 任务分配问题以及 AGV 无冲突路径问题. 余娜娜等[16] 对自动化分拣仓库中同时作业的 AGV 路径规划问题进行研究，以最小化最大搬运完成时间为目标建立数学模型，提出一种改进差分进化算法，并通过数值实验验证了算法的有效性.

本研究与现有文献主要区别如下：在研究对象方面，目前国内外的研究主要集中于传统的传送带分拣系统以及 AGV 分拣系统，对落地式分拣系统这一新兴系统的研究较少. 相较于传统分拣系统，在落地式分拣系统的运作优化过程中，需综合考虑包裹的分拣格口分配、AGV 调度优化以及无冲突路径规划问题. 在求解算法方面，目前求解分拣系统中包裹分配问题、AGV 调度问题的主流算法仍是元启发式算法，求解多 AGV 无冲突行驶路径的常用方法是设置避撞策略，本研究设计了基于列生成的算法求解包裹分配问题和 AGV 无冲突路径规划问题，设计不同的加速算法提升两问题的求解效率，实验结果验证了该算法的有效性和高效性.

## 2 问题描述

本研究对落地式分拣系统中的分拣作业流程进行研究. 落地式分拣系统中包裹的分拣流程如下：首先，在包裹进入系统前，将由入口处的信息识别仪扫描包裹上的电子面单，获取包裹信息；然后，包裹由交叉带传送带运送到不同的分拣格口，由分拣格口处的工人或机械臂将包裹搬运到 AGV 的托盘上；最后，AGV 将包裹投递到其对应的笼车中，完成该包裹的分拣任务. 上述分拣作业流程中，对包裹进行了两次分拣作业，第一次作业是交叉带传送带将包裹分配给不同的分拣格口，第二次作业是包裹由 AGV 从所在的分拣格口运输至对应的笼车处. 落地式分拣系统通过增加“中间节点”的方式降低了分拣作业的复杂度，同时也对系统中相关资源的调度优化提出了更高的要求. 大量包裹进入系统后，作为系统“中间节点”的分拣格口极易出现包裹滞留阻塞格口，造成系统的局部瘫痪，降低分拣效率；同时负责投递包裹的 AGV 在行驶过程时可能出现冲突，造成路线拥堵，降低分拣效率.

在落地式分拣系统中，包裹分配和 AGV 路径规划均会影响系统的运作效率. 在包裹分配中包含两个决策，其一为包裹与分拣格口的分配关系，其二是为包裹与 AGV 的分配关系，包裹与分拣格口分配方案的不合理将造成分拣格口作业量不均衡，导致部分分拣格口处出现阻塞现象，包裹与 AGV 分配不合理将延迟包裹完成分拣的时间，增加 AGV 行驶距离，并且增大 AGV 行驶过程中出现冲突的可能性. 为解决上述问题，本研究以最小化包裹在分拣格口处的滞留时间为目标构建了一个混合整数规划模型，该模型决策了包裹与分拣格口、AGV 的分配关系，以实现分拣格口的任务均衡.

完成包裹分配后，到达分拣格口的包裹将被分配的 AGV 投递至笼车，投递行驶过程中，多 AGV 作业可能会出现冲突，如相向冲突和节点冲突，其中相向冲突是指多辆 AGV 在同一时间同一路段相向行驶产生的冲突，如图 2(a) 所示，节点冲突是指多辆 AGV 在同一时间到达同一节点，如图 2(b) 所示. 为避免多 AGV 作业时产生冲突，本研究在给定的包裹与分拣格口、AGV 的分配关系的基础上，以最小化 AGV 服务包裹的延迟时间为目标构建了一个混合整数规划模型，为 AGV 规划无冲突行驶路线.

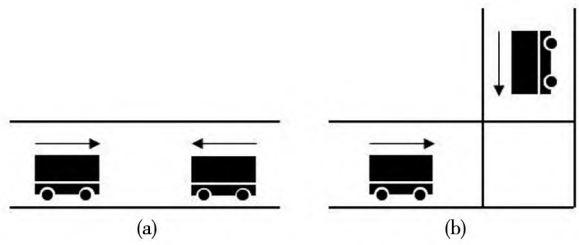


**图 2**　AGV 冲突类型

Fig. 2 AGV conflict types

为均衡落地式分拣系统中资源利用率，避免造成系统局部阻塞，本研究以最小化包裹在系统中滞留时间为目标，构建了两阶段混合整数规划模型，第一阶段决策包裹与分拣格口、AGV 之间的分配关系，第二阶段为 AGV 规划无冲突行驶路线，从而优化整个分拣系统运作的效率. 为了实现实时调度优化，本研究将包裹分成不同批次进行处理，通过设置包裹批次将实时动态优化问题转化为一系列的静态优化问题[17]，对进入系统的包裹进行批处理.

## 3　数学模型

本节详细介绍了包裹分配模型以及 AGV 无冲突路径规划模型，其中包裹分配模型主要用于求解包裹与分拣格口的分配关系以及 AGV 运输包裹序列的问题; AGV 无冲突路径规划模型运用时空网络建模方法，为 AGV 规划无冲突的行驶路径.

### 3.1　包裹分配模型

包裹分配模型以最小化包裹在分拣格口处的滞留时间为目标，构建了一个混合整数规划模型，同时决策包裹与分拣格口的分配关系以及 AGV 对应的包裹序列. 为简化问题，提出如下假设:

1) 所有 AGV 均以相同速度匀速行驶;

2) 不考虑 AGV 行驶时的冲突;

3) 忽略包裹在分拣格口与 AGV 之间、AGV 与笼车之间的搬运时间.

#### 3.1.1　符号说明

1) 集合与下标

$S$——包裹编号的集合，下标为 $s$;

$N$——分拣格口编号的集合，下标为 $n$;

$R$——*AGV* 编号的集合，下标为 $r$;

$L$——笼车编号的集合，下标为 $l$.

2) 参数

$e_s$——包裹进入系统的时间;

$l_s$——包裹 $s$ 对应的笼车编号;

$g_{s,n}$——包裹 $s$ 经传送带到达分拣格口 $n$ 的所需时间;

$t_{n,l}$——AGV 从分拣格口 $n$ 到达笼车 $l$ 的行驶时间;

$\dot{o}_r$——*AGV* $r$ 的路线起点;

$\dot{d}_r$——*AGV* $r$ 的路线终点;

$M$——较大的数.

3) 决策变量

$\alpha_{s,n}$——0 - 1 决策变量，若包裹 $s$ 被分配给分拣格口 $n$ 则为 1，否则为 0;

$\beta_{s,s',r}$——0 - 1 决策变量，若 *AGV* $r$ 在运输包裹 $s$ 后再运送包裹 $s'$ 为 1，否则为 0;

$\varepsilon_{s,r}$——*AGV* $r$ 到达包裹 $s$ 对应的分拣格口的时间;

$\delta_s$——包裹 $s$ 开始被 AGV 运输的时间.

#### 3.1.2　数学模型

$$\text{Min} \sum_{s\in S}\left[\delta_s - \sum_{n\in N}(e_s + g_{s,n})\ \alpha_{s,n}\right] \quad (1)$$

s. t.

$$\sum_{n\in N}\alpha_{s,n} = 1 \quad \forall s \in S \quad (2)$$

$$\delta_s \geqslant (e_s + g_{s,n})\ \alpha_{s,n} \quad \forall s \in S, n \in N \quad (3)$$

$$\sum_{r\in R}\sum_{s'\in S\cup\{\dot{d}_r\}}\beta_{s,s',r} = 1 \quad \forall s \in S \quad (4)$$

$$\sum_{s\in S\cup\{\dot{d}_r\}}\beta_{\dot{o},s,r} = \sum_{s\in S\cup\{\dot{o}_r\}}\beta_{s,\dot{d}_r,r} = 1 \quad \forall r \in R \quad (5)$$

$$\sum_{s\in S\cup\{\dot{o}_r\}}\beta_{s,s',r} = \sum_{s\in S\cup\{\dot{d}_r\}}\beta_{s',s,r} \leqslant 1 \quad \forall s' \in S, r \in R \quad (6)$$

$$\varepsilon_{s',r} \leqslant \delta_s + \sum_{n\in N} t_{n,l_s}\alpha_{n,s} + \sum_{n\in N} t_{n,l_{s'}}\alpha_{n,s'} + M(1 - \beta_{s,s',r})$$

$$\forall s \in S\cup\{\dot{o}_r\}, s' \in S\cup\{\dot{d}_r\}, r \in R \quad (7)$$

$$\varepsilon_{s',r} \geqslant \delta_s + \sum_{n\in N} t_{n,l_s}\alpha_{n,s} + \sum_{n\in N} t_{n,l_{s'}}\alpha_{n,s'} - M(1 - \beta_{s,s',r})$$

$$\forall s \in S\cup\{\dot{o}_r\}, s' \in S\cup\{\dot{d}_r\}, r \in R \quad (8)$$

$$\varepsilon_{s',r} \geqslant \varepsilon_{s,r} - M(1 - \beta_{s,s',r}) \quad \forall s \in S\cup\{\dot{o}_r\}, s' \in S\cup\{\dot{d}_r\}, r \in R \quad (9)$$

$$\delta_s \geqslant \varepsilon_{s,r} \quad \forall s \in S \cup \{\dot{o}_r, \dot{d}_r\}, r \in R \tag{10}$$

$$\alpha_{s,n}, \beta_{s,s',r} \in \{0,1\} \quad \forall s, s' \in S \cup \{\dot{o}_r, \dot{d}_r\}, r \in R, n \in N \tag{11}$$

$$\varepsilon_{s,r}, \delta_s \geqslant 0 \forall s, s' \in S \cup \{\dot{o}_r, \dot{d}_r\}, r \in R \tag{12}$$

目标函数(1)是最小化包裹在分拣格口处的滞留时间. 式(2)确保每一件包裹必须被分配给一个分拣格口;式(3)确保包裹被 AGV 服务的时间晚于包裹到达分拣格口的时间;式(4)确保每一件包裹必须被一辆 AGV 运输;式(5)确保每一个 AGV 的行驶路线中有一个起点和一个终点;式(6)确保 AGV 运输包裹的连贯性;式(7)和式(8)表示若 AGV 先后运输包裹 $s$ 与 $s'$,则 AGV 到达服务包裹 $s'$ 的分拣格口的时间等于 AGV 完成包裹 $s$ 运输的时间加上从包裹 $s$ 对应的笼车行驶至服务包裹 $s'$ 的分拣格口的时间之和. 式(9)表示若 AGV 先后运输包裹 $s$ 与 $s'$,则 AGV 到达服务 $s'$ 的分拣格口的时间晚于到达服务 $s$ 的分拣格口的时间;式(10)表示包裹开始被运输的时间不早于 AGV 到达服务包裹的分拣格口的时间. 式(11)和式(12)定义了决策变量.

### 3.2 AGV 无冲突路径规划模型

根据包裹分配模型 M1 的求解结果,可获得 AGV 的任务序列. 假设包裹分配模型求得 *AGV r* 的包裹序列为 $\dot{o}_r \rightarrow s1 \rightarrow s2 \rightarrow \dot{d}_r$,且包裹 $s1$ 与 $s2$ 对应的分拣格口分别为 $n1$ 与 $n2$;则可得到 AGV 的任务序列为 $\dot{o}_r \rightarrow n1 \rightarrow l_{s1} \rightarrow n2 \rightarrow l_{s2} \rightarrow \dot{d}_r$,其中包含了五个任务,依次表示为 $\dot{o}_r \rightarrow n1, n1 \rightarrow l_{s1}, l_{s1} \rightarrow n2, n2 \rightarrow l_{s2}, l_{s2} \rightarrow \dot{d}_r$. 根据已知的 AGV 任务序列,AGV 无冲突路径规划模型需决策 AGV 执行全部任务的无冲突行驶路线.

时空网络综合考虑时间维度与空间维度信息,可实时监测路网信息以及 AGV 的状态信息,从而为 AGV 规划无冲突行驶路径,因此本研究采用时空网络建模方法构建 AGV 路径规划模型. 自动化分拣仓库环境属于结构化环境,因此本研究采用如图 3 所示的栅格图表示仓库路网. 应用时空网络建模方法构建 AGV 路径规划模型时,首先将栅格图中每个栅格抽象成节点,任意两相邻节点抽象成边;然后在给定的 AGV 任务序列的基础上,以最小化任务的延误时间为目标,构建模型为 AGV 规划无冲突的行驶路径.

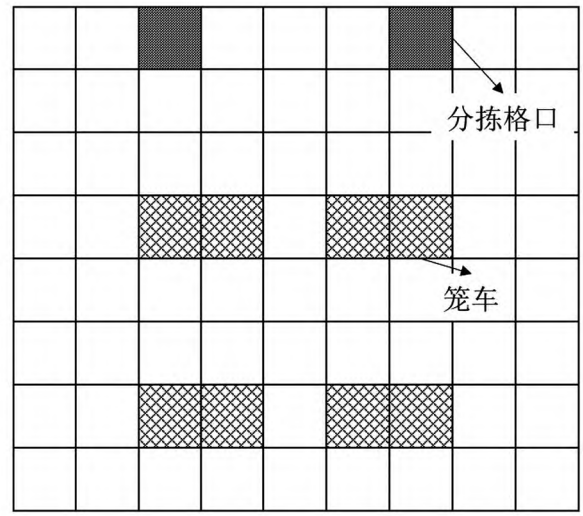


图 3 仓库路网示意图

Fig. 3 Schematic diagram of the warehouse road network

#### 3.2.1 符号说明

1) 集合与下标

$\bar{R}$——M1 的调度方案中被分配包裹的 AGV 的编号集合,下标为 $r$;

$T$——时间段集合,下标为 $t$;

$V$——路网中节点集合,下标为 $i, j$;

$V_i$——路网中节点 $i$ 的相邻节点集合;

$A$——路网中可达边集合,$A = \{(i,j) \mid i \in V, j \in V_i \cup i\}$;

$V^T$——时空网络中可用节点(时空节点)集合,$V^T = \{(t,i) \mid t \in T, i \in V\}$;

$V^T_{t,i}$——时空网络中节点 $(t,i)$ 的可用相邻节点集合,$V^T_{t,i} = \{(t+1,j) \mid t \in T, j \in V_i \cup i\}$;

$A^T$——时空网络中可达边(时空边)集合,$A^T = \{(t,i,j) \mid t \in T, i \in V, j \in V_i \cup i\}$;

$L_r$——$M1$ 的调度方案中 *AGV r* 的任务集合,$V_r = \{l_1, l_2, \cdots, l_k, \cdots l_{|V_r|}\}$,其中 $l_k$ 表示 *AGV r* 的第 $k$ 个任务.

2) 参数

$x_{r,l}$——任务 $l$ 被 *AGV r* 服务的期望时间,可根据 $M1$ 中 $\varepsilon_{s,r}$ 的值得到;

$y_{r,i}$——节点 $i$ 被 *AGV r* 访问的最少次数;

$u_l$——任务 $l$ 对应的节点编号;

$\dot{o}_r$——*AGV r* 的路线起点;

$\dot{d}_r$——*AGV r* 的路线终点.

3) 决策变量

$\theta_{r,t,i,j}$ ——0 – 1 决策变量，若 $AGV\ r$ 在时刻 $t$ 开始经过弧 $(i,j)$ 则为 1，否则为 0；

$\mu_{r,t,i}$ —— 0 – 1 决策变量，若 $AGV\ r$ 在时刻 $t$ 位于点 $i$ 则为 1，否则为 0；

$\gamma_{r,l,t}$ —— 0 – 1 决策变量，若任务 $l$ 在时刻 $t$ 被服务为 1，否则为 0.

### 3.2.2 数学模型

[M2] $$\text{Min} \sum_{r\in\bar{R}}\sum_{i\in V_r}\left(\sum_{t\in T} t\,\gamma_{t,l} - x_{r,l}\right) \tag{13}$$

s. t.

$$\sum_{r\in\bar{R}}(\theta_{r,t,i,j} + \theta_{r,t,j,i}) \leqslant 1 \quad \forall(t,i,j),\ (t,j,i) \in A^T, i \neq j \tag{14}$$

$$\sum_{r\in\bar{R}}\sum_{(t,i,j)\in A^T}\theta_{r,t,i,j} \leqslant 1 \quad \forall(t,i) \in V^T \tag{15}$$

$$\sum_{(0,\dot{o},j)\in A^T}\theta_{r,0,\dot{o},j} = 1 \quad \forall r \in \bar{R} \tag{16}$$

$$\sum_{(t,i,\dot{d})\in A^T}\theta_{r,t,i,\dot{d}} = 1 \quad \forall r \in \bar{R} \tag{17}$$

$$\sum_{(t,i,j)\in A^T}\theta_{r,t,i,j} = \sum_{(t-1,j,i)\in A^T}\theta_{r,t-1,j,i} \quad \forall(t,i) \in V^T,\ t \neq 0, r \in \bar{R} \tag{18}$$

$$\sum_{(t,i,j)\in A^T}\theta_{r,t,i,j} \leqslant 1 \quad \forall t \in T, r \in \bar{R} \tag{19}$$

$$\sum_{(t,u_l,j)\in A^T}\theta_{r,t,u_l,j} \geqslant y_{r,u_l} \quad \forall l \in L_r, r \in \bar{R} \tag{20}$$

$$\mu_{r,t,i} = \sum_{(t,i,j)\in A^T}\theta_{r,t,i,j} \quad \forall(t,i) \in V^T, r \in \bar{R} \tag{21}$$

$$\gamma_{r,l,t} \leqslant \mu_{r,t,u_l} \quad \forall l \in L_r, t \in T, r \in \bar{R} \tag{22}$$

$$\sum_{t'<t}\gamma_{t',l_k} \geqslant \gamma_{t,l_{k+1}} \quad \forall l \in L_r, t \in T, r \in \bar{R} \tag{23}$$

$$\sum_{t\in T}\gamma_{r,l,t} = 1 \quad \forall l \in L_r, r \in \bar{R} \tag{24}$$

$$\sum_{t\in T} t\,\gamma_{r,l,t} \geqslant x_{r,l} \quad \forall l \in L_r, r \in \bar{R} \tag{25}$$

$$\theta_{r,t,i,j} \in \{0,1\} \quad \forall(t,i,j) \in A^T, r \in \bar{R} \tag{26}$$

$$\mu_{r,t,i} \in \{0,1\} \quad \forall(t,i) \in V^T, r \in \bar{R} \tag{27}$$

$$\gamma_{r,l,t} \in \{0,1\} \quad \forall l \in L_r, t \in T, r \in \bar{R} \tag{28}$$

目标函数式(13)最小化任务的延迟时间. 式(14)为避免 AGV 行驶时出现相向冲突. 式(15)确保同一时刻同一节点仅有一辆 AGV，即避免 AGV 间出现节点冲突. 式(16)和式(17)确定 AGV 的起止点位置. 式(18)为流量守恒式. 式(19)确保每个时刻 AGV 最多位于一个节点. 式(20)确保任务 $l \in L_r$ 对应节点的访问次数不少于需要被访问的最小次数. 式(21)连接决策变量 $\mu_{r,t,i}$ 与 $\theta_{r,t,i,j}$ . 式(22)连接决策变量 $\mu_{r,t,i}$ 与 $\gamma_{t,l}$ . 式(23)确保每个 AGV 对应的任务必须按任务序列中的先后顺序被服务. 式(24)确保每个 AGV 对应的任务必须在一个时间段被服务. 式(25)确保 AGV 服务任务的时间必须大于其期望时间. 式(26)至式(28)定义了决策变量.

## 4 模型求解

本研究构建了包裹分配模型以及 AGV 路径规划模型来协同调度落地式分拣系统中的分拣格口、AGV 等资源. 其中包裹分配模型决策分拣格口的包裹分配以及 AGV 服务包裹的序列，该模型需在包裹进入分拣系统后，迅速求解包裹的分配方案，从而使包裹经传送带到达相应的分拣格口. 随后，根据包裹分配模型的调度结果获得 AGV 的任务序列，AGV 路径规划模型需迅速反应，为 AGV 规划无冲突路径. 在落地式分拣系统的运作过程中，对两模型的求解速度都提出了极高的要求；本研究构建的两个模型均为 NP-hard 问题，采用商业求解器难以在短时间内求解；因此，本节通过分析两模型的结构特性，设计了基于列生成的算法进行求解，并分别设计了变邻域算法以及 $A^*$ 算法用于加速两模型对应的子问题的求解.

### 4.1 包裹分配模型求解

为提升包裹分配问题的求解速度，本节根据包裹分配模型 M1 的结构特性，设计了基于列生成的算法求解该模型，并应用变邻域算法提升子问题的求解速度.

#### 4.1.1 集合划分模型(主问题)

本节应用 Dantzig-Wolfe 分解算法将模型重构为集合划分模型. 为构建集合划分主问题(set-partitioning master problem，MP)，设置以下集合参

数以及决策变量: 集合 $P_r$ 表示 $AGV\ r$ 所有可行的包裹分配方案集合,下标为 $p_r$; 参数$A_{p_r,s}$ 表示方案 $p_r$ 中 $AGV\ r$ 是否服务包裹 $s$ ,如果服务包裹 $s$ 其值为 1,否则为 0; 参数 $C_{p_r}$ 表示方案 $p_r$ 导致的滞留成本; 决策变量 $\eta_{p_r}$ 为 0 - 1 型整数变量,若方案 $p_r$ 被选择为 1,否则为 0. 由此构建集合划分模型如下

$$\text{M3}\quad \text{Min}\sum_{r\in R}\sum_{p_r\in P_r}C_{p_r}\eta_{p_r} \tag{29}$$

s. t.

$$\sum_{p_r\in P_r}\eta_{p_r}\leqslant 1\quad \forall r\in R \tag{30}$$

$$\sum_{r\in R}\sum_{p_r\in P_r}A_{p_r,s}\eta_{p_r}=1\quad \forall s\in S \tag{31}$$

$$\eta_{p_r}\in\{0,1\}\quad \forall p_r\in P_r, r\in R. \tag{32}$$

目标函数式( 29) 最小化分配方案的总成本. 式( 30) 确保每辆 AGV 至多选择一个分配方案; 式( 31) 确保所有包裹必须被服务; 式( 32) 定义了决策变量.

4.1.2 限制性松弛主问题

主问题 M3 中包含了所有可行的包裹分配方案 $\bigcup_{r\in R}P_r$,问题规模的增长将导致可行方案的数量呈指数级增长,从而增大主问题求解难度. 为此,定义 $P_r'\subseteq P_r$ 为 $AGV\ r$ 的当前可行分配方案,并将 0 - 1 决策变量 $\eta_{p_r}$ 松弛为连续变量,构建线性松弛限制性主问题( linear relaxation of restricted master problem, LR-RMP) 如下

$$\text{M4}\quad \text{Min}\sum_{r\in R}\sum_{p_r\in P_r'}C_{p_r}\eta_{p_r} \tag{33}$$

s. t.

$$\sum_{p_r\in P_r'}\eta_{p_r}\leqslant 1\quad \forall r\in R \tag{34}$$

$$\sum_{r\in R}\sum_{p_r\in P_r'}A_{p_r,s}\eta_{p_r}=1\quad \forall s\in S \tag{35}$$

$$\eta_{p_r}\geqslant 0\quad \forall p_r\in P_r', r\in R \tag{36}$$

在列生成算法的迭代过程中,LR-RMP 的对偶变量将被用于构建子问题,定义 $\rho_r$ 和 $\pi_s$ 分别表示式( 34) 和式( 35) 的对偶变量.

4.1.3 子问题

列生成算法中,子问题用于产生检验数为负的列,并将其加入 RMP 中求解. 此外,由于 AGV 是异质的,列生成迭代过程中每次需要求解 $|R|$ 个子问题,其中每个定价子问题可用 $\text{PP}_r$( M5) 表示,用于生成一辆 $AGV\ r$ 对应的包裹序列方案.

$$\text{M5}\quad \text{Min}\{C_{p_r}-\rho_r-\sum_{s\in S}\sum_{n\in N}\alpha_{s,n}\pi_s\} \tag{37}$$

s. t.

去掉维度 $r$ 后的式( 5) 至式( 12) ;

$$C_{p_r}=\sum_{s\in S}\left(\delta_s-\sum_{n\in N}g_{s,n}\alpha_{s,n}\right) \tag{38}$$

4.1.4 求解规则生成初始解

列生成算法运行时,首先需要为 LR-RMP 生成初始解. 本节基于贪心的思想为生成初始解,即根据包裹进入系统的先后顺序,按照包裹被服务的时间尽可能早的原则,依次为每件包裹分配 AGV 以及分拣格口,具体算法流程如 **Algorithm 1** 所示.

**Algorithm 1: M1 求解规则**

**输入:** 包裹集合 $\bar{S}\leftarrow S$ , $P=\{p_r, r\in R\}$ ,初始化 $p_r\leftarrow\emptyset, r\in R$ .

**定义:** $EN_{p_r}, EL_{p_r}, ET_{p_r}$ 分别为路线 $p_r$ 中最后一件包裹对应的分拣格口、笼车和分拣时间.

```
1  将 S̄ 中的包裹按照 e_s 值升序排序
2  For each s ∈ S̄
3      (n*, r*) ← argmin_{n∈N, r∈R}{ max{ ET_{p_r} +
       t_{EN_{p_r},EL_{p_r}} + t_{n,EL_{p_r}}, e_s + g_{s,n} } } ;
4      p_{r*} ← p_{r*} ∪ {s} ;
5      更新 EN_{p_r*}, EL_{p_r*}, ET_{p_r*} ;
6  End for
```

4.1.5 子问题求解

为了加速子问题的求解,本节设计了变邻域搜索算法( variable neighborhood search,VNS) 求解 PP. VNS 算法通过设计不同邻域动作构成的邻域结构间交替搜索获得满意解.

1) 初始解生成

基于贪心的思想为 VNS 算法生成初始解,设置包裹集合 $\bar{S}\leftarrow\{s,\pi_s>0\}$ ,调用 Algorithm 1,得到子问题的初始解.

2) 邻域结构

由于子问题需同时决策分拣格口的包裹分配以及 AGV 的包裹序列,VNS 算法设计时将 AGV 的包裹序列作为外层决策,分拣格口的包裹分配

作为内层决策. 外层通过不同邻域结构的交替搜索获得 AGV 的包裹序列; 内层根据获得的 AGV 包裹序列,通过更新解策略为包裹分配分拣格口. 下面将对本研究设计的扰动策略以及交换,删除与插入三种邻域结构进行描述,然后对内层决策时调用的更新解策略进行说明.

①扰动: 为扩展解的搜索范围,避免算法陷入局部最优,本算法设计了一个扰动策略. 当 VNS 算法获得局部最优解时,则执行扰动操作生成一组新解,用于下一次迭代寻优.

**Algorithm 2: 扰动策略**

**输入:** 当前 AGV 路线 $p_{cur}$

1 生成两个随机数 $rnd_1$,$rnd_2 \in [0, |p_{cur}|]$,且 $rnd_1 < rnd_2$

2 将路线 $p_{cur}$ 中位于区间 $[rnd_1, rnd_2]$ 包裹倒序,得到新路线 $p_{new}$

3 $UpdateSol(p_{new})$ //Algorithm 6

②交换: 由于包裹进入系统时存在一定的先后顺序,此顺序会一定程度上影响包裹被 AGV 服务的顺序,基于此问题特性,本研究设计了点交换操作,使 AGV 路线小幅度变动. 点交换操作是指寻找当前路线 $p_{cur}$ 中对目标值改善最大的两个点进行交换,并更新路线 $p_{cur}$,具体算法流程参见 Algorithm 3.

**Algorithm 3: 交换操作**

**输入:** 当前路线 $p_{cur}$;

**定义:** 路线 $p \leftarrow p_{cur}$

```
For i from 1 to |p|
    Forj from i + 1 to |p|
        交换路线 p 中第 i 个和第 j 个包裹;
        UpdateSol (p); //Algorithm 6
        If Obj(p) < Obj(p_cur)
                p_cur ← p
        End if
    End for
End for
```

③ 删除: 删除是指从当前路线 $p_{cur}$ 中删除一个对目标值改善最大的点,并更新路线 $p_{cur}$,具体算法流程参见 Algorithm 4.

**Algorithm 4: 删除操作**

**输入:** 当前 AGV 路线 $p_{cur}$,路线 $p \leftarrow p_{cur}$

```
For i from 1 to |p|
    删除路线 p 的位置 i 处的包裹
    UpdateSol (p); //Algorithm 6
    If Obj(p) < Obj(p_cur)
            p_cur ← p
    End if
End for
```

④插入: 插入操作是指从当前路线 $p_{cur}$ 中插入对目标值改善的点,并更新路线 $p_{cur}$,具体算法流程参见 Algorithm 5.

**Algorithm 5: 插入操作**

**输入:** 当前 AGV 路线 $p_{cur}$; 路线 $p \leftarrow p_{cur}$

```
For s from 1 to |S|
    If 当前路线 p 包含包裹 s
        continue;
    End if
    For i from 1 to |p|
        路线 p_tmp ← p
        将包裹 s 插入到 p_tmp 的位置 i 处
        UpdateSol (p_tmp); //Algorithm 6
        If Obj(p_tmp) < Obj(p_cur)
            p_cur ← p_tmp;
        End if
    End for
End for
```

⑤更新解策略: 上述邻域结构搜索过程中,可获得不同的 AGV 包裹序列,更新解策略将决策分拣格口的包裹分配,即包裹与分拣格口的对应关系,并根据决策结果计算当前包裹分配方案的费用值. 对于一组给定的 AGV 包裹序列(路线),更新解策略依次将每个包裹分配给对方案目标值影响较小的分拣格口,具体算法流程参见 Algorithm 6.

**Algorithm 6: 更新解策略**

**输入:** 当前 AGV 路线 $p_{cur}$

**定义:** 路线 $p \leftarrow p_{cur}$, $Obj^* \leftarrow M$, $Obj \leftarrow -\rho_r$; $\tilde{n}_s$, $\hat{t}_s$ 表示路线 $p$ 中包裹 $s$ 对应的分拣格口与分拣时间

**Algorithm 6: 更新解策略**

1 **For** $i$ from 1 to $|p|$

2 $s_1 \leftarrow p_{new}[i]$, $Obj \leftarrow Obj - \pi_{s_1}$, $n^* \leftarrow 0$ // $p_{new}[i]$ 表示路线 $p_{new}$ 中第 $i$ 个包裹

3 **For** each $n$ in $N$

4 **For** $j$ from $i+1$ to $|p|$

5 $s_2 \leftarrow p[j]$, $Obj \leftarrow Obj + \min_{n' \in N}\{\max\{0, \overset{\wedge}{t}_{s_1} + t_{\overset{\sim}{n}_{s_1}, l_{s_1}} + t_{n', l_{s_1}} - (e_{s_2} + g_{s_2, n'})\}\} - \pi_{s_2}$; 更新 $\overset{\sim}{n}_{s_2}$, $\overset{\wedge}{t}_{s_2}$;

6 **End for**

7 **If** $Obj^* > Obj$

8 $Obj^* \leftarrow Obj$, $n^* \leftarrow n$;

9 **End if**

10 **End for**

11 更新 $\overset{\sim}{n}_s \leftarrow n^*$, $\overset{\wedge}{t}_s$;

13 **End for**

#### 4.1.6 可行化策略

列生成算法用于求解线性松弛受限主问题(LR-RMP),并不能保证找到一组可行整数解. 因此,本研究设计了一种可行化策略,得到一组近似最优整数解,即外层通过给定的规则获取整数解,内层调用列生成算法求解方案. 具体可行化方法如下: 在列生成算法结束时,得到一组浮点解,首先选出决策变量 $\eta_{p_r}$ 最大的方案,若存在多组方案,则选择 $C_{p_r}$ 值最小的方案,然后令该方案的 $\eta_{p_r}$ 值为 1,将该方案加入可行方案集合中,再次调用列生成算法寻找下一组方案,重复上述步骤,直至得到一组完整可行方案集合. 具体算法流程参见 Algorithm 7.

Algorithm 7: M1 基于列生成的算法

定义: $AGV_r$ 和 $Parcel_s$ 分别为式(34)和式(35)的右手边系数, $SolList$ 为可行解集;

1 初始化 $Parcel_s \leftarrow 1, \forall s \in S, AGV_r \leftarrow 1, \forall r \in R$

2 **While**(当前解集 $SolList$ 不可行)

3 将当前获得的所有方案 $p_r \in \bigcup_{r \in R} P_r'$,首先按照 $\eta_{p_r}$ 值的大小降序排序,再按照 $C_{p_r}$ 值的大小升序排序,获得方案集合 $P'$;

4 选择集合 $P'$ 中第一个方案 $p_r^1$, $SolList \leftarrow SolList \cup \{p_r^1\}$;

$Parcel_s \leftarrow 0, \forall s in p_r^1$; $AGV_r \leftarrow 0$;

5 //列生成算法

6 **While**(子问题求得检验数为负的方案)

7 调用 Algorithm 1 生成初始解;

8 Cplex 求解 LR-RMP,获得对偶变量 $\rho_r$ 与 $\pi_s$ 的值;

9 调用 4.1.5 中 VNS 算法求解 $PP_r$, $\forall r \in R$;

10 **End while**

11 **End while**

**Algorithm 7: M1 基于列生成的算法**

### 4.2 AGV 路径规划模型求解

随着路网节点数量的增大,AGV 路径规划模型中变量与约束的数量迅速增加,造成求解难度急剧增大. 不同 AGV 执行任务时的路线之间存在耦合关系,因此可将原模型分解成主问题与子问题并应用列生成算法求解,提升模型求解效率.

在列生成算法中,子问题用于规划每辆 AGV 执行任务的行驶路线,主问题用于选择多辆 AGV 的无冲突路线集合. 通过分析可以发现,子问题与静态路网中的最短路径问题相似,因此本节采用 $A^*$ 算法进一步加速子问题的求解.

#### 4.2.1 集合划分模型

本节首先应用 Dantzig-Wolfe 分解算法将模型重构为集合划分模型. 为构建集合划分主问题,设置以下集合参数以及决策变量: 集合 $H_r$ 表示 $AGV\ r$ 所有可行的行驶路线集合,下标为 $h_r$; 参数 $A_{h_r,t,i}$ 表示路线 $h_r$ 中 $AGV\ r$ 是否在 $t$ 时刻访问节点 $i$,若是值为 1,否则为 0; 参数 $B_{h_r,t,(i,j)}$ 表示路线 $h_r$ 中 $AGV\ r$ 是否在 $t$ 时刻开始访问弧 $(i,j)$,若是值为 1,否则为 0; 参数 $C_{h_r}$ 表示路线 $h_r$ 导致的延误成本; 决策变量 $\zeta_{h_r}$ 为 0-1 型整数变量,若路线 $\zeta_{h_r}$ 被选择为 1,否则为 0; 由此构建集合划分模型如下

[M6] $$\text{Min} \sum_{r \in R} \sum_{h_r \in H_r} C_{h_r} \zeta_{h_r} \tag{39}$$

s.t.

$$\sum_{h_r \in H_r} \zeta_{h_r} = 1 \quad \forall r \in \bar{R} \tag{40}$$

$$\sum_{r\in\bar{R}}\sum_{h_r\in H_r}\left(B_{h_r,t,(i,j)}+B_{h_r,t,(j,i)}\right)\zeta_{h_r}\leqslant 1$$
$$\forall(i,j)\in A,t\in T \tag{41}$$

$$\sum_{r\in\bar{R}}\sum_{h_r\in H_r}A_{h_r,t,i}\,\zeta_{h_r}\leqslant 1\quad\forall i\in A,t\in T \tag{42}$$

$$\zeta_{h_r}\in\{0,1\}\quad\forall h_r\in H_r,r\in\bar{R} \tag{43}$$

目标函数式(39)最小化行驶路线的总成本. 式(40)已被分配包裹的 AGV 一定对应一条路线;式(41)避免 AGV 行驶时产生相向冲突;式(42)避免 AGV 行驶时产生相向冲突;式(43)定义了决策变量.

4.2.2　线性松弛限制性主问题

定义 $H_r'\subseteq H_r$ 为 $AGV\ r$ 的当前可行行驶路线集合,将 0 - 1 决策变量 $\zeta_{h_r}$ 松弛为连续变量,构建线性松弛限制性主问题如下

[M7] $$\text{Min}\sum_{r\in\bar{R}}\sum_{h_r\in H_r'}C_{h_r}\zeta_{h_r} \tag{44}$$

s. t.

$$\sum_{h_r\in H_r'}\zeta_{h_r}=1\quad\forall r\in\bar{R} \tag{45}$$

$$\sum_{r\in\bar{R}}\sum_{h_r\in H_r'}\left(B_{h_r,t,(i,j)}+B_{h_r,t,(j,i)}\right)\zeta_{h_r}\leqslant 1$$
$$\forall(i,j)\in A,t\in T \tag{46}$$

$$\sum_{r\in\bar{R}}\sum_{h_r\in H_r}A_{h_r,t,i}\,\zeta_{h_r}\leqslant 1\quad\forall i\in A,t\in T \tag{47}$$

$$\zeta_{h_r}\geqslant 0\quad\forall h_r\in H_r',r\in\bar{R} \tag{48}$$

定义 $\tau_r$,$\varphi_{tij}$ 和 $\omega_{ti}$ 分别表示式(45)、式(46)和式(47)的对偶变量,在列生成算法的迭代过程中,LR - RMP 的对偶变量将被用于构建子问题,寻找检验数为负的列.

4.2.3　子问题

由于 AGV 是异质的,因此每次需要求解 $|R|$ 个子问题,其中每个定价子问题可用 $\text{PP}_r$M8 表示,用于生成一辆 AGV $r$ 的行驶路线.

[M8] $$\text{Min}\ C_{h_r}-\tau_r-\sum_{t\in T}\sum_{(i,j)\in A}\left(\theta_{t,i,j}+\theta_{t,j,i}\right)\varphi_{tij}-\sum_{t\in T}\sum_{(i,j)\in A}\theta_{t,i,j}\,\omega_{ti} \tag{49}$$

s. t.

去掉维度后的式(16)至式(28);

$$C_{h_r}=\sum_{i\in V_r}\left(\sum_{t\in T}t\,\gamma_{t,l}-x_{r,l}\right) \tag{50}$$

4.2.4　生成初始解

运行列生成算法时,需为 RMP 生成一组可行解. 本节基于贪心算法思想生成一组 AGV 行驶路线,即根据 AGV 开始执行任务的时间的先后顺序,依次为 AGV 规划行驶路线. 若无法得到一组可行行驶路径,即路线之间存在冲突,则设置路线的延误成本 $C_{h_r}$ 为一个极大值 $M$.

4.2.5　子问题求解($\text{A}^*$ 算法)

为了加速子问题的求解,本节设计了 $\text{A}^*$ 算法求解 PP. $\text{A}^*$ 算法具有对环境反应迅速,计算速度快等优点,被广泛用于求解 AGV 路径规划问题中. 子问题 M8 用于求解一个已知任务序列的 AGV 的行驶路线,因此,可使用 $\text{A}^*$ 算法依次为任务序列中的每个任务规划行驶路线. 具体算法流程参见 Algorithm 8.

**Algorithm 8:** $\text{A}^*$ 算法

输入: $AGV r$ 的任务集合 $L_r$;集合 $\bar{V}_{t,i}^T\leftarrow V_{t,i}^T$;

定义: $\dot{o}_l$ 与 $\dot{d}_l$ 为任务 $l$ 的起止点; $\dot{t}_l$ 为任务 $l$ 的最早开始时间, $\dot{t}_l\leftarrow 0$; $SL$ 为节点搜索列表; $f_{t,i}$ 为列表 $SL$ 中节点 $(t,i)$ 的代价值; $Pre_{t,i}$ 为节点 $(t,i)$ 的来源, $Pre_{t,i}\leftarrow\varnothing$;

```
For k from 1 to |L_r|
    l ← L_r[k];
    SL ← SL ∪ (ṫ_l, ȯ_l); f_{ṫ_l,ȯ_l} ← -ω_{ṫ_l,ȯ_l}; Δ_{ṫ_l,ȯ_l} ← -ω_{ṫ_l,ȯ_l}; Pre_{ṫ_l,ȯ_l} ← ∅;
    While (SL ≠ ∅)
        (t*, i*) ← SL 中 f_{t,i} 值最小的 (t, i);
        If (i* is ḋ_l and t* ⩾ x_{r,l})
            et ← t*;        // et 记录当前任务的完成时间
```

**Algorithm 8:** $A^*$ 算法

```
        break;
      End if
      For each (t^* + 1, j) in V^T_{t^*, i^*}
        Δ^{new}_{t^*+1, j} = Δ_{t^*, i^*} − ω_{t^* i^*} − φ_{t^* i^* j} − φ_{t^* j i^*}
        If Pre_{t^*+1, j} is Ø or Δ^{new}_{t^*+1, j} < Δ_{t^*+1, j}
        Δ_{t^*+1, j} ← Δ^{new}_{t^*+1, j};
        f_{t^*, i} ← Δ^{new}_{t^*+1, j} + max{0, t^* + 1 + dis_{j, d_l}; − x_{r, l}};
        Pre_{t^*+1, j} ← (t^*, i^*);
      End if
    End for
  End while
  根据 Pre_{et, d_l}; 可确定 AGV 执行任务 l 的行驶路线; 更新集合 V̄^T_{t^*, i^*};
  ṫ_{L_r[k+1]} ← et;
End for
```

4.2.6 可行化策略

列生成算法不能保证找到一组可行整数解，因此需要对列生成的求解结果进行可行化处理，本节采用的可行化策略与 4.1.6 中可行化方法类似，因此不再赘述.

4.2.7 基于时空网络的 Dijkstra 算法

为了进一步加速 AGV 路径规划问题的求解，本研究基于可实时更新的时空网络[18]，运用 Dijkstra 算法求解 M2. 时空网络可以通过时空节点以及时空边，实时监测路网节点的状态信息以及 AGV 的行驶状态信息，本研究设计基于时空网络的 Dijkstra 算法，可通过实时监测路网信息，避免同一时空节点或同一时空边被不同 AGV 访问，从而为 AGV 规划无冲突行驶路径.

本节提出的算法由内外两层算法构成，外层算法用于提供任务以及更新当前路网信息，即集合 $V^T$ 和 $A^T$，内层算法根据外层指定的任务，根据当前路网信息，实时规避冲突，求解执行该任务的 AGV 的具体行驶路径. 具体算法流程参见 Algorithm 9.

**Algorithm 9:** M2 基于时空网络的 Dijkstra 算法

**输入:** AGV$r$ 的任务集合 $L_r$

**定义:** $\dot{o}_l$ 与 $\dot{d}_l$ 为任务 $l$ 的起止点; $\dot{t}_l$ 为任务 $l$ 的最早开始时间，$\dot{t}_l \leftarrow 0$; $SL$ 为节点搜索列表; $f_{t,i}$ 为列表 $SL$ 中节点 $(t,i)$ 的代价值; $Pre_{t,i}$ 为节点 $(t,i)$ 的来源，$Pre_{t,i} \leftarrow \emptyset$;

```
For each r in |R̄|
  l ← L_r[1], ṫ_l ← 0 // L_r[1] 表示路线 L_r 中第 1 个任务
        taskList ← taskList ∪ {l}
End for
While(taskList ≠ Ø)
  将 taskList 中的任务 l 按照 ṫ_l 值的大小升序排序，l ← taskList[1]
  //Dijkstra 算法求解 AGV 执行任务 l 的行驶路线 ph
  SL ← Ø; SL ← SL ∪ (ṫ_l, ȯ_l); f_{t,i} ← 0; Pre_{t,i} ← Ø;
```

**Algorithm 9：** M2 基于时空网络的 Dijkstra 算法

```
While( SL ≠ Ø )
    ( t* ,i* ) ← SL 中 f_{t,i} 值最小的 ( t,i) ;
    If ( i* is d_l and t* ≥ x_{r,l} )
    et ← t* ;          // et 记录当前任务的完成时间
      break;
    End if
    For each ( t* + 1,j) in V^T_{t*,i*}
      f^{new}_{t*+1,j} = f_{t*,i*} + 1 ;
      If Pre_{t*+1,j} is Ø or  f^{new}_{t*+1,j} < f_{t*+1,j}
        f_{t*+1,j} ← f^{new}_{t*+1,j} ; Pre_{t*+1,j} ← ( t* ,i* ) ;
      End if
    End for
End while
根据 Pre_{et,d_l} 可确定 AGV 执行任务 l 的行驶路线; 更新集合 V^T_{t*,i*} ,V^T ,A^T ;
t_{L_r[k+1]} ← et ;
taskList ← taskList ∪ { l}
End while
```

## 5　数值实验

为验证本研究所提模型的有效性以及设计算法的高效性，本节进行了大量的数值实验. 本研究实验中所用平台的 CPU 为 Intel Xeon E5 – 2680 v4 @2.4Ghz，内存 256GB，采用 Windows10 64 位处理系统. 代码在 Visual Studio 2019 环境中使用 C#实现，实验调用了 IBM ILOG CPLEX 12.6.1 求解器. 所有算例的最大求解时间限制为 1 h (3 600 s) .

### 5.1　算例设置

对于包裹分配模型 M1，其问题规模大小主要受包裹数量、AGV 数量以及分拣格口数量的影响; 对于 AGV 路径规划问题 M2，其问题规模大小主要受包裹数量以及栅格布局大小的影响; 因此，本研究设置了四个算例组( instance group，ISGs) 对两个模型进行实验验证. 表 1 中列出了算例组的参数设置，其中小规模算例为 ISG1、ISG2，大规模算例为 ISG3、ISG4.

**表 1　算例的问题规模**

Table 1 Problem scales of instance groups

| Group ID | $\|S\|$ | $\|R\|$ | $\|N\|$ | 栅格布局 |
|---|---|---|---|---|
| ISG1 | 9 | 3 | 3 | 10 × 10 |
| ISG2 | 12 | 4 | 4 | 15 × 15 |
| ISG3 | 15 | 5 | 5 | 20 × 20 |
| ISG4 | 20 | 6 | 6 | 25 × 25 |

### 5.2　算法效率测试

#### 5.2.1　包裹分配模型 M1 测试

为了验证包裹分配模型的有效性以及设计的基于列生成的算法的高效性，本研究分别使用 Cplex、基于列生成的求解算法、以及求解规则求解，并对比三种方法求解结果之间的差距.

表 2 展示了三种方法的求解结果，对于 ISG1，Cplex 可以在合理的时间内求得最优解，基于列生成的算法以及求解规则均可以在 1 s 内求得问题的解，基于列生成的算法与求解规则的求解结果与最优解的平均差距值分别为 0.00% 和 58.57%. 对于 ISG2 中部分算例，Cplex 无法在一小时内求得最优解，因此表 2 中记录

了 Cplex 求解一小时的获得的上界值，可以看出 Cplex 的求解结果明显优于求解规则. 本研究提出的基于列生成算法的求解结果明显优于求解规则，且求解速度明显优于 Cplex，这验证了本研究提出的模型的有效性以及基于列生成算法的高效性.

表 2 M1 小规模算例结果

Table 2 Results of M1 insmall-scale instances

| 算例 | Cplex | | CG | | | 规则 | |
|---|---|---|---|---|---|---|---|
| | $F_{Cplex}$ | $t_{Cplex}(s)$ | $F_{CG}$ | $t_{CG}(s)$ | $\Delta_{CG}(\%)$ | $F_{Rule}$ | $\Delta_{Rule}(\%)$ |
| ISG1 – 1 | 8 | 3.14 | 8 | 0.20 | 0.00 | 10 | 25.00 |
| ISG1 – 2 | 24 | 22.24 | 24 | 0.16 | 0.00 | 38 | 58.33 |
| ISG1 – 3 | 12 | 4.52 | 12 | 0.20 | 0.00 | 16 | 33.33 |
| ISG1 – 4 | 28 | 18.74 | 28 | 0.35 | 0.00 | 54 | 92.86 |
| ISG1 – 5 | 36 | 31.55 | 36 | 0.27 | 0.00 | 66 | 83.33 |
| Ave. | | | | | | | 58.57 |
| ISG2 – 1 | 34 | 3 600.00 | 34 | 0.83 | 0.00 | 56 | 64.71 |
| ISG2 – 2 | 68 | 3 600.00 | 68 | 0.76 | 0.00 | 84 | 23.53 |
| ISG2 – 3 | 28 | 699.37 | 28 | 0.81 | 0.00 | 50 | 78.57 |
| ISG2 – 4 | 32 | 985.69 | 32 | 0.66 | 0.00 | 60 | 87.50 |
| ISG2 – 5 | 28 | 1 287.35 | 28 | 1.17 | 0.00 | 42 | 50.00 |
| *Ave.* | | | | | 0.00 | | 60.86 |

注：$F_{Cplex}$、$F_{CG}$、$F_{Rule}$ 表示 M1 分别由 Cplex、CG 和规则得到的解；$t_{Cplex}$、$t_{CG}$ 表示 M1 分别由 Cplex 和 CG 求解的耗时；$\Delta_{CG}$、$\Delta_{Rule}$ 分别表示 CG、规则所得计算结果与 Cplex 结果的相对误差，$\Delta_{CG}=\frac{F_{CG}-F_{Cplex}}{F_{Cplex}}$、$\Delta_{Rule}=\frac{F_{Rule}-F_{Cplex}}{F_{Cplex}}$.

为了进一步评估所提出的基于列生成的算法的高效性，本研究应用 ISG3 和 ISG4 进行大规模实验. 与表 2 类似，表 3 也对比了三种方法的求解结果与求解速度，比较结果表明，基于列生成的算法的求解结果优于 Cplex 一小时内获得上界值以及求解规则的解值，平均 gap 为 –23.29% 和 164.46%，并且本研究提出的算法可在短时间内求解包裹分配问题，进一步验证本研究设计算法的高效性.

表 3 M1 大规模算例结果

Table 3 Results of M1 in large-scale instances

| 算例 | Cplex | | CG | | | 规则 | |
|---|---|---|---|---|---|---|---|
| | $F_{Cplex}$ | $t_{Cplex}(s)$ | $F_{CG}$ | $t_{CG}(s)$ | $\Delta_{CG}(\%)$ | $F_{Rule}$ | $\Delta_{Rule}(\%)$ |
| ISG3 – 1 | 56 | 3 600 | 56 | 3.76 | 0.00 | 124 | 121.43 |
| ISG3 – 2 | 74 | 3 600 | 72 | 3.35 | –2.70 | 94 | 27.03 |
| ISG3 – 3 | 46 | 3 600 | 38 | 1.07 | –17.39 | 110 | 139.13 |
| ISG3 – 4 | 68 | 3 600 | 68 | 2.81 | 0.00 | 264 | 288.24 |
| ISG3 – 5 | 50 | 3 600 | 34 | 1.66 | –32.00 | 176 | 252.00 |
| ISG4 – 1 | 56 | 3 600 | 36 | 10.29 | –35.71 | 164 | 192.86 |
| ISG4 – 2 | 82 | 3 600 | 38 | 15.77 | –53.66 | 128 | 56.10 |
| ISG4 – 3 | 54 | 3 600 | 38 | 9.50 | –29.63 | 198 | 266.67 |
| ISG4 – 4 | 106 | 3 600 | 74 | 20.26 | –30.19 | 216 | 103.77 |
| ISG4 – 5 | 76 | 3 600 | 52 | 13.11 | –31.58 | 226 | 197.37 |
| *Ave.* | | | | | –23.29 | | 164.46 |

5.2.2 AGV 路径规划模型 M2 测试

为了验证 AGV 路径规划模型的有效性以及设计的基于列生成的算法的高效性，本节对比了 Cplex、基于列生成的求解算法、以及基于时空网络的 Dijkstra 算法的求解结果与求解速度. 表 4 结果表明在小规模算例下，三种方法均可以为 AGV

规划无冲突路径，其中 Cplex 可以在一定时间内求得问题最优解，但其求解时间过长，无法直接应用于现实问题的求解. 基于列生成的算法可在 30s 内求解，且求解结果与最优解的差值为 0；基于时空网络的 Dijkstra 算法可在 1s 内求解，求解结果与最优解的差值为 17. 这验证了 AGV 路径规划模型的有效性以及基于列生成算法以及基于时空网络的 Dijkstra 算法的高效性.

**表 4　M2 小规模算例结果**

Table 4 Results of M2 in small-scale instances

| 算例 | Cplex | | CG | | | 基于时空网络 Dijkstra 算法 | | |
|---|---|---|---|---|---|---|---|---|
| | $F_{Cplex}$ | $t_{Cplex}(s)$ | $F_{CG}$ | $t_{CG}(s)$ | $\Phi_{CG}$ | $F_{Alg}$ | $t_{Alg}(s)$ | $\Phi_{Alg}$ |
| ISG1 – 1 | 1 | 52.41 | 1 | 6.19 | 0 | 4 | 0.03 | 3 |
| ISG1 – 2 | 2 | 85.87 | 2 | 9.84 | 0 | 3 | 0.02 | 1 |
| ISG1 – 3 | 0 | 61.31 | 0 | 0.39 | 0 | 0 | 0.04 | 0 |
| ISG1 – 4 | 0 | 46.56 | 0 | 0.37 | 0 | 0 | 0.03 | 0 |
| ISG1 – 5 | 0 | 45.81 | 0 | 1.27 | 0 | 2 | 0.04 | 2 |
| ISG2 – 1 | 2 | 368.01 | 2 | 28.31 | 0 | 6 | 0.10 | 4 |
| ISG2 – 2 | 2 | 330.70 | 2 | 10.10 | 0 | 3 | 0.14 | 1 |
| ISG2 – 3 | 0 | 258.11 | 0 | 8.54 | 0 | 3 | 0.11 | 3 |
| ISG2 – 4 | 2 | 420.86 | 2 | 8.17 | 0 | 2 | 0.20 | 0 |
| ISG2 – 5 | 0 | 259.78 | 0 | 1.59 | 0 | 0 | 0.30 | 0 |
| *Sum.* | | | | | 0 | | | 17 |

注：$F_{Cplex}$、$F_{CG}$、$F_{Alg}$ 表示 M2 分别由 Cplex、CG 和基于时空网络的 Dijkstra 算法得到的解；$t_{Cplex}$、$t_{CG}$、$t_{Alg}$ 表示 M2 分别由 Cplex、CG 和基于时空网络的 Dijkstra 算法求解的耗时；$\Phi_{CG}$、$\Phi_{Alg}$ 分别表示 CG、基于时空网络的 Dijkstra 算法所得计算结果与 Cplex 结果的绝对误差，$\Phi_{CG} = F_{CG} - F_{Cplex}$、$\Phi_{Alg} = F_{Alg} - F_{Cplex}$.

为进一步评估所提出的基于列生成的算法以及基于时空网络的 Dijkstra 算法的效果，本节应用 ISG3 和 ISG4 进行了大规模实验. 由于 Cplex 无法在一个小时内求得大规模算例的可行解，因此表 5 对比了两种算法的求解结果与求解速度. 结果表明，基于列生成的算法的求解结果优于基于时空网络的 Dijkstra 算法，总的差值为 39，但在 ISG4 中，基于列生成的算法的求解时间较长，难以应用到现实的路径规划问题；虽然基于时空网络的 Dijkstra 算法存在无法规划出最优行驶路径，但其求解速度极快，在大规模算例求解时，有较大的应用空间.

**表 5　M2 大规模算例结果**

Table 5 Results of M2 in large-scale instances

| 算例 | CG | | 基于时空网络的 Dijkstra 算法 | | |
|---|---|---|---|---|---|
| | $F_{CG}$ | $t_{CG}(s)$ | $F_{Alg}$ | $t_{Alg}(s)$ | $\Phi_{Alg}$ |
| ISG3 – 1 | 2 | 56.74 | 3 | 0.60 | 1 |
| ISG3 – 2 | 0 | 4.13 | 0 | 0.50 | 0 |
| ISG3 – 3 | 0 | 20.26 | 8 | 0.47 | 8 |
| ISG3 – 4 | 1 | 72.01 | 7 | 0.79 | 6 |
| ISG3 – 5 | 2 | 65.69 | 4 | 0.81 | 2 |
| ISG4 – 1 | 8 | 216.25 | 9 | 1.70 | 1 |
| ISG4 – 2 | 4 | 149.49 | 7 | 1.66 | 3 |
| ISG4 – 3 | 3 | 331.06 | 9 | 1.57 | 6 |
| ISG4 – 4 | 0 | 75.17 | 8 | 1.83 | 8 |
| ISG4 – 5 | 0 | 74.75 | 4 | 1.85 | 4 |
| *Sum.* | | | | | 39 |

### 5.4 敏感度分析

#### 5.4.1 案例实验

本节通过仿真生成了两组包裹数据集，每个数据集包含五个批次的包裹，其中第一组数据每批次 15 件包裹，第二组数据每批次 20 个包裹；本节应用第 4 节提出算法对包裹进行分批处理，为每批次包裹提供调度优化方案. 为了验证本研究提出算法在多批次问题的应用效果，本研究对比了基于列生成算法求解两阶段问题（CG&CG）的求解结果与求解规则（rule）求解一阶段问题，基于时空网络的 Dijkstra 算法求解二阶段问题（Rule & Dijkstra，R&D）的求解结果. 两阶段的目标值之和为包裹在系统中的总滞留时间. 从图 4 可以看出 CG&CG 算法的求解结果明显优于 R&D 算法. 随着批次的增加，包裹的总滞留时间也在显著增加，这是因为 AGV 数量较少，难以及时处理已经到达分拣格口的包裹. 实际应用中，可通过调整系统布局以及 AGV 数量减少包裹的总滞留时间.

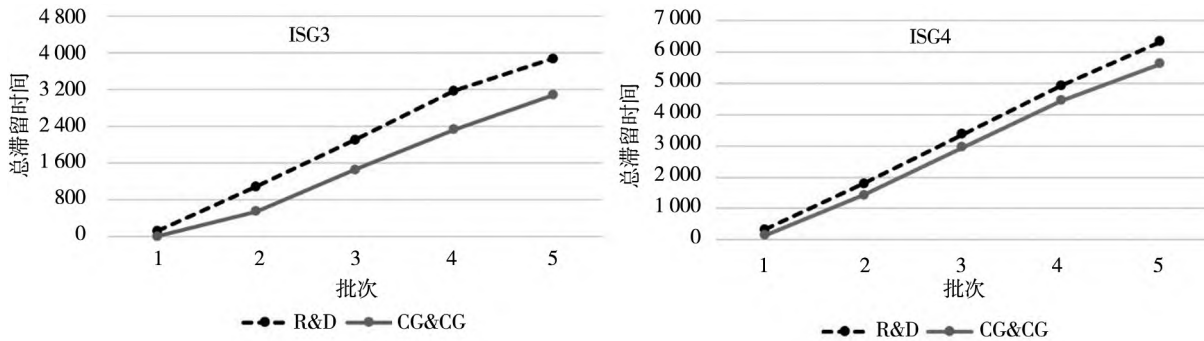


**图 4 多批次案例下两种算法的结果**

Fig. 4. Results of two algorithms in multi-batch cases

#### 5.4.2 AGV 数量的敏感度分析

本节分析了 AGV 数量对系统运作效率的影响. 本节应用基于列生成的算法求解 M1，获得 AGV 的任务序列；随后应用基于时空网络的 Dijkstra 算法求解 M2，规划 AGV 的详细行驶路径. 为了考虑路径拥堵对整个系统的运作影响，当基于时空网络的 Dijkstra 算法无法找到无冲突的行驶路径，则设置一个较大的惩罚值.

图 5 展示了不同 AGV 数量对系统延误的影响，其中横轴表示 AGV 数量，纵轴表示的系统延误时间是包裹的总滞留时间与 AGV 拥堵惩罚值之和. 结果表明，随着 AGV 数量的增多，总延误时间先下降后上升，这是由于随着 AGV 数量的增长，包裹处理速度加快，将降低系统延误时间，随着 AGV 数量的不断增多，AGV 拥堵几率也会增加，从而导致系统延误时间增加. 此外，分拣系统布局规模的不同，AGV 数量的最佳配置也不同，决策者需根据现实场景配置恰当数量的 AGV.

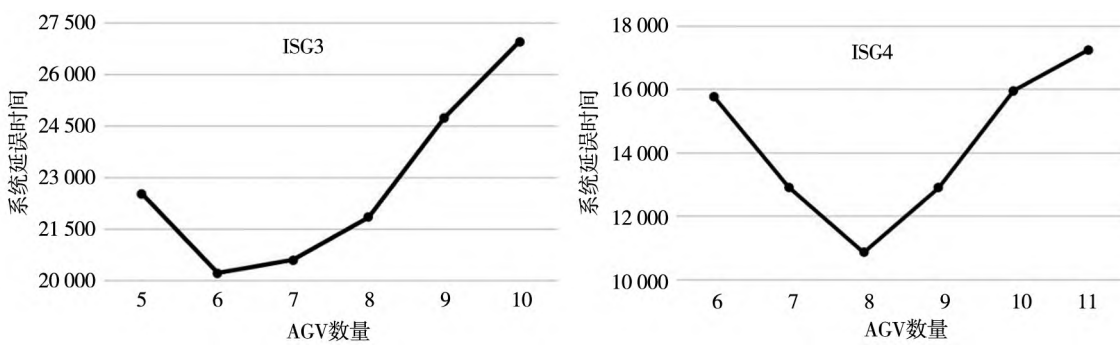


**图 5 AGV 数量与系统延误时间的关系**

Fig. 5 The relationship between the number of AGVs and system delays

#### 5.4.3 分拣格口布局的敏感度分析

本节对分拣格口布局对包裹的总滞留时间的影响进行分析，结果展示在图 6 中. 其中，横轴表示两个分拣格口之间的距离，该值可表示系统中

分拣格口布局的分散程度，纵轴表示包裹的总滞留时间. 结果表明，在 ISG3 中，总滞留时间随分拣格口之间距离的增加先降低后增大，在 ISG4 中，滞留时间随着分拣格口之间距离的增加先降低后趋于稳定，这说明分拣格口并非越分散越好，过于分散可能会造成 AGV 空载时间过长，降低系统的分拣效率；因此，仓库运营者需考虑系统规模的大小，对分拣格口进行合理布局，以提升系统分拣效率.

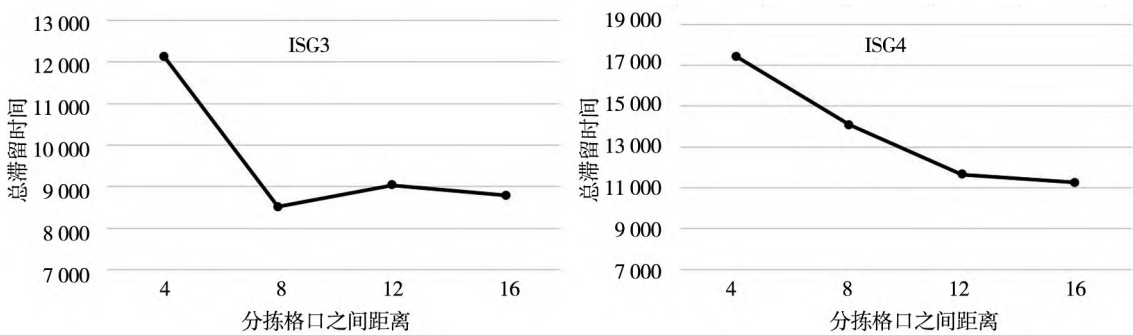


**图 6　分拣格口之间距离与总滞留时间的关系**

Fig. 6 The relationship between the distance of sorting slots and total dwell time

## 6　结束语

本研究对落地式分拣系统这一新型自动化分拣系统展开研究，已知一批次包裹的进入系统时间以及对应的笼车等信息，决策服务包裹与分拣格口以及包裹与 AGV 的分配关系，并为 AGV 规划无冲突行驶路线. 本研究提出的方法为落地式分拣系统的实际运作提供了优化调度决策方法. 本研究的贡献主要有以下几点：

1) 在模型层面，本研究综合考虑了系统运作过程中包裹、分拣格口、AGV 等主要资源之间的联系及相互影响，建立了两阶段混合整数规划模型，其中一阶段模型( M1) 决策包裹与分拣格口、AGV 之间的分配关系；二阶段模型( M2) 在一阶段模型基础上，为 AGV 规划详细的无冲突行驶路线.

2) 在算法层面，本研究提出了基于列生成算法求解两阶段模型，并设计了 VNS 算法和 $A^*$ 算法分别加速两模型中子问题的求解速度；此外，针对布局较大的分拣系统，本研究设计了基于时空网络的 Dijkstra 算法求解 AGV 的行驶路径，增加本研究的应用价值. 相比于 Cplex，本研究提出的算法可以更快速的求解更大规模的算例.

3) 在应用层面，本研究应用案例数据进行实验，验证了本研究设计算法的适用性，基于批次的滚动决策下，本研究设计的算法可有效降低包裹在系统中的滞留时间，并为 AGV 规划无冲突路径. 此外，本研究通过分析 AGV 数量以及分拣格口的布局方式对分拣效率的影响，为决策者提供了一些管理意见.

本研究仍有不足之处，提出的算法目前无法在极短的时间内求解更大规模的问题，未来研究中可进一步探索并设计更加高效的求解算法. 此外，该系统运作过程中，需将满载的笼车搬运至指定地点，如何协同调度包裹的分拣投递任务与笼车的搬运任务也是未来的研究方向之一.

**参 考 文 献：**

[1] 蔡　熙，刘登峰. 自动分拣行业 2022 年回顾与 2023 年展望 [J]. 物流技术与应用，2023，28(3)：59－61.
Cai Xi，Liu Dengfeng. Automated sorting industry：Review of 2022 and outlook for 2023 [J]. Logistics & Material Handling，2023，28(3)：59－61. (in Chinese)

[2] 刘雨浩，费叶琦，潘知瑶，等. 物流分拣设备的发展现状与展望 [J]. 机电工程技术，2023，52(1)：59－62＋171.

Liu Yuhao, Fei Yeqi, Pan Zhiyao, et al. Development status and prospect of logistics sorting equipment [J]. Mechanical & Electrical Engineering Technology, 2023, 52(1): 59 – 62 + 171. (in Chinese)

[3] Boysen N, de Koster R, Weidinger F. Warehousing in the e-commerce era: A survey [J]. European Journal of Operational Research, 2019, 277(2): 396 – 411.

[4] Azadeh K, de Koster R, Roy D. Robotized and automated warehouse systems: Review and recent developments [J]. Transportation Science, 2019, 53(4): 917 – 945.

[5] 李 珲, 肖海宁, 周临震, 等. 包裹智能分拣系统分拣路径动态规划方法 [J]. 机械制造与自动化, 2023, 52(3): 83 – 86 + 116.

Li Hui, Xiao Haining, Zhou Linzhen, et al. Dynamic sorting path planning method for parcel intelligent sorting system [J]. Machine Building & Automation, 2023, 52(3): 83 – 86 + 116. (in Chinese)

[6] 杨 莹, 张 莉, 郭瑞鸿, 等. 基于改进快速搜索随机树算法的包裹分拣路径规划算法 [J]. 计算机集成制造系统, 2022, 28(3): 951 – 958.

Yang Ying, Zhang Li, Guo Ruihong, et al. Improved RRT-based sorting path planning algorithm for parcel [J]. Computer Integrated Manufacturing Systems, 2022, 28(3), 951 – 958. (in Chinese)

[7] Fedtke S, Boysen N. Layout planning of sortation conveyors in parcel distribution centers [J]. Transportation Science, 2017, 51(1): 3 – 18.

[8] 镇 璐, 谭哲一, 萧理阳, 等. 面向双层自动分拣系统的包裹分拣优化模型与算法研究 [J]. 中国管理科学, 2021, 29(7): 171 – 180.

Zhen Lu, Tan Zheyi, Xiao Liyang, et al. Research on parcel optimization model and algorithm for double-layar automatic systems [J]. Chinese Journal of Management Science, 2021, 29(7): 171 – 180. (in Chinese)

[9] Bae J, Chung W. A heuristic for path planning of multiple heterogeneous automated guided vehicles [J]. International Journal of Precision Engineering and Manufacturing, 2018, 19(12): 1765 – 1771.

[10] Dang Q V, Singh N, Adan I, et al. Scheduling heterogeneous multi-load AGVs with battery constraints [J]. Computers & Operations Research, 2021, 136: 105517.

[11] Polten L, Emde S. Scheduling automated guided vehicles in very narrow aisle warehouses [J]. Omega-International Journal of Management Science, 2021, 99: 102204.

[12] Boysen N, Schwerdfeger S, Ulmer M W. Robotized sorting systems: Large-scale scheduling under real-time conditions with limited lookahead [J]. European Journal of Operational Research, 2023, 310(2): 582 – 596.

[13] Murakami K. Time-space network model and MILP formulation of the conflict-free routing problem of a capacitated AGV system [J]. Computers & Industrial Engineering, 2020, 141(C): 106270.

[14] 张新艳, 邹亚圣. 基于改进 A* 算法的自动导引车无碰撞路径规划 [J]. 系统工程理论与实践, 2021, 41(1): 240 – 246.

Zhang Xinyan, Zou Yasheng. Collision-free path planning for automated guided vehicles based on improved A* algorithm. [J]. Systems Engineering—Theory & Practice, 2021, 41(1): 240 – 246. (in Chinese)

[15] 李昆鹏, 刘腾博, 贺冰倩, 等. “货到人”拣选系统中 AGV 路径规划与调度研究 [J]. 中国管理科学, 2022, 30(4): 240 – 251.

Li Kunpeng, Liu Tengbo, He Bingqian, et al. A study on routing and scheduling of automated guided vehicle in “Cargo-to-picker” system [J]. Chinese Journal of Management Science, 2022, 30(4): 240 – 251. (in Chinese)

[16] 余娜娜, 李铁克, 王柏琳, 等. 自动化分拣仓库中多 AGV 调度与路径规划算法 [J]. 计算机集成制造系统, 2020, 26(1): 171 – 180.

Yu Nana, Li Tieke, Wang Bolin, et al. Multi-AGVs scheduling and path planning algorithm in automated sorting warehouse [J]. Computer Integrated Manufacturing Systems [J], 2020, 26(1): 171 – 180. (in Chinese).

[17] 徐小峰，姜明月，邓忆瑞. 整合逆向物流协同配送动态路径优化问题研究 [J]. 管理科学学报，2021，24(10)：106－126.
Xu Xiaofeng，Jiang Mingyue，Deng Yirui. Dynamic vehicle routing problem with simultaneous pickup and delivery in collaborative distribution under demand concurrent [J]. Journal of Management Sciences in China，2021，24(10)：106－126. (in Chinese)

[18] 高一鹭，胡志华. 基于时空网络的自动化集装箱码头自动化导引车路径规划 [J]. 计算机应用，2020，40(7)：2155－2163.
Gao Yilu，Hu Zhihua. Path planning for automated guided vehicles based on tempo-spatial network at automated container terminal [J]. Journal of Computer Applications，2020，40(7)：2155－2163. (in Chinese)

## Parcel assignment and AGV routing in floor-based sorting systems

*HE Xue-ting，ZHEN Lu*，WU Jing-wen，GAO Jia-jing*
School of Management，Shanghai University，Shanghai 200444，China

**Abstract:** The rapid growth of express business has led to a significant increase in the complexity of sorting，driving the rapid development of automated sorting systems. Therefore，a floor-based sorting system has been developed to enhance the flexibility，economy and scalability of automated sorting systems. In this paper，parcel assignment and conflict-free routing of automated guided vehicles in a floor-based sorting system are studied. Two mixed integer programming models are proposed to minimize the parcel dwell time in the system. To solve the models efficiently，two column generation-based solution algorithms are designed with embedded acceleration techniques such as the variable neighborhood search algorithm and the A* algorithm. Numerous numerical experiments are performed to verify the validity of the models and the efficiency of the algorithms. Sensitivity analysis experiments are also performed to provide some management insights.

**Key words:** parcel assignment；conflict-free routing optimization；space-time network model；column generation；mixed integer programming