队伍编号	MC2203428
题号	B

基于启发式算法和规则设计的 AGV 动态接单调度模型

摘 要

随着物联网的高速发展，我国制造业面临着数字化、网络化、智能化的转型升级需求，而企业中的仓库管理升级改造的重要方面，就是借助 AGV(无人搬运车)实现仓库的高效自动化运作。而要实现无人仓库管理，就需要解决 AGV 的统筹调度问题。

本文基于 A*算法和遗传算法，旨于解决动态接单过程中货物运输的最优化调度 AGV 的问题，同时，本文通过合理的假设，并进行了调度仿真实验，最后，引入禁忌表和规则设计，以避免 AGV 的碰撞、死锁问题。

对于问题 1，在不考虑机器人碰撞的条件下，进行调度算法的建模。假设 AGV 运动速度固定，并以 AGV 运动一格的时长为时间片进行离散事件仿真。建立动态接单调度策略，即在接收到了一个订单后，首先尝试借助调度中、停靠位上货品资源进行订单预满足，当订单无法全部满足时，再调动 AGV 取货送拣。

在确定需要调度 AGV 以满足订单需求后，通过遗传算法对本轮决策中的调度方案进行快速搜索，其适应度计算方式则基于 A*算法，以预估满足当前订单下的最优解，从而进行 AGV 调度。

通过仿真实验，在工位的停靠位数量$\boldsymbol{b}$为 **4**，订单刷新时间间隔$\boldsymbol{T}$为 **4**，拣件时间t_0为 **5** 的时候，订单平均移动代价为 **19.09** 格。

对于问题 2，需要我们为库存商品动态分配一个拣选工位，实现拣选工位的负载均衡。可知对于货物分配到工位的过程中，AGV 前一段的取货路程是不随分派工位改变的，因此我们在 A*算法计算出货物到工位最短路基础上，基于信息熵计算方法定义边缘地带，以负荷均衡率最大作为分配目标，通过调整边缘货物的默认工位，实现指派优化和负载均衡。

将基于二元信息熵计算方法的“模糊地带”评价法的阈值定于大于等于 **0.995** 时，此时相较于初始划分情况，在各工位潜在负荷标准差降低 **110.85** 个货品的基础上，仅仅增加了全局 AGV 额外 **8** 格的送拣(区别于取托盘路程)成本。

对于问题 3，本文基于禁忌表和规则设计的方法来优化 AGV 路径规划，以解决 AGV 的死锁、碰撞问题。引入禁忌表，确保后规划路径的 AGV 不与已规划路径的 AGV 位置发生冲突；引入主动避让机制，确保送完托盘的 AGV 不在拣选工位、托盘回收处停留，确保畅通；通过停等机制，确保送拣 AGV 的有序等待。文中对改进系统的运作过程进行了规避过程的可视化展示。

关键词：AGV 调度，A*算法，遗传算法，负载平衡，禁忌表，仿真实验

目录

1. 问题重述

1.1 问题背景与文献综述

问题背景：

随着电子商务的高速发展，高效的无人仓管理系统逐步替代了传统了人工管理仓库的模式，并成为了制造业为实现数字化转型升级的发展方向。对于无人仓库中的(AGV)搬运机器人的调度方式，是在建设无人仓过程中需要解决的重点问题。

AGV 的调度问题本质上是组合优化问题，需要在大量可行解中寻找最优解，但随着问题规模的扩大，往往会产生“组合爆炸”的情况，以至于现代计算机无法对相应的指派顺序进行穷举确定。因此，该类问题也是 NP-hard 问题，具有不确定性。

如何在有限的时间内，优化 AGV 的调度过程，实现运输效益的最大化，提高效率，同时防止搬运机器人的死锁和碰撞的情况发生，是设计 AGV 调度系统的时候需要解决的问题。因此设计出一个能够实时根据订单和其他情况进行快速决策的 AGV 调度算法，具有重要意义。

文献综述：

对于 AGV 调度问题，有诸多前人进行了相关的研究，分别在调度优化、高度仿真建模等方面取得了较好的成果，如有以系统总作业时间最短为目标，使用改进的遗传算法进行寻优以取得更到调度方案的[2]，有基于偏好函数和使用改进 InceptionNet 卷积神经网络进行修正的方式，完成指派结果寻优的方法[3]。同时，也有以最小化加权总搬运完成时间为目标建立混合整数线性规划模型来完成 AGV 的任务指派和路径规划的方法[4]。

对于 AGV 调度的研究方式，常常采用诸多相似的假设，如有将 AGV 抽象为质点的研究方法[5]，有引入时间片的研究方法[6]，且大部分研究常常将 AGV 运动假设为匀速运动[2][7]，并且以上研究都获得了不错的实验成果。

以上文献，多基于 AGV 所调度托盘内容单一，且不考虑托盘回收的情况进行建模考虑的，或者是基于订单已知、任务量已知的情况下进行全局寻优，若要将以上方法应用于本题，还需要进行一定的改造。

1.2 问题描述

1.2.1 问题 1

本题要求我们在假设 AGV 在执行任务时不发生碰撞的条件下，设计最优的统筹调度策略。与此同时，题目还给定了订单数据、仓库库存分布情况、AGV 初始位置和数量。并将目标设定为 AGV 尽可能忙的条件下，AGV 调度系统的总行走路径最小。

1.2.2 问题 2

本题目需要我们为无人仓系统中的拣选工位进行负荷均衡的设计，根据仓库商品分布情况进行动态分区以实现两个目标：实现拣选机器人的工作量相对均衡和预防 AGV 的局部拥堵。最终形成一个仓储位——拣选工位的默认指派方案。

1.2.3 问题 3

对于本题目需要我们设计解决碰撞、拥堵和死锁的调度策略。本题目需要我们在应对不同场景的时候，使用预防、避免或者是检查以解除等方法来使多个 AGV 协调运行。为此需要设置一定的调度原则和规定。

2. 问题分析

对于以上问题，本文根据题目设定场景大小、货物、机器人规模等情况，使用 A* 算法进行调度的最短路计算，使用遗传算法缩小调度组合方式的搜索空间以完成 AGV 回收、回库、分配货物的三大任务。接下来，在总路径最短的原则下，建立“边缘地带”概念，对聚类边缘的仓储位按照负荷均衡的规则进行簇分配。最后考虑 AGV 个体之间的防死锁、碰撞规则。本文工作流程如下：

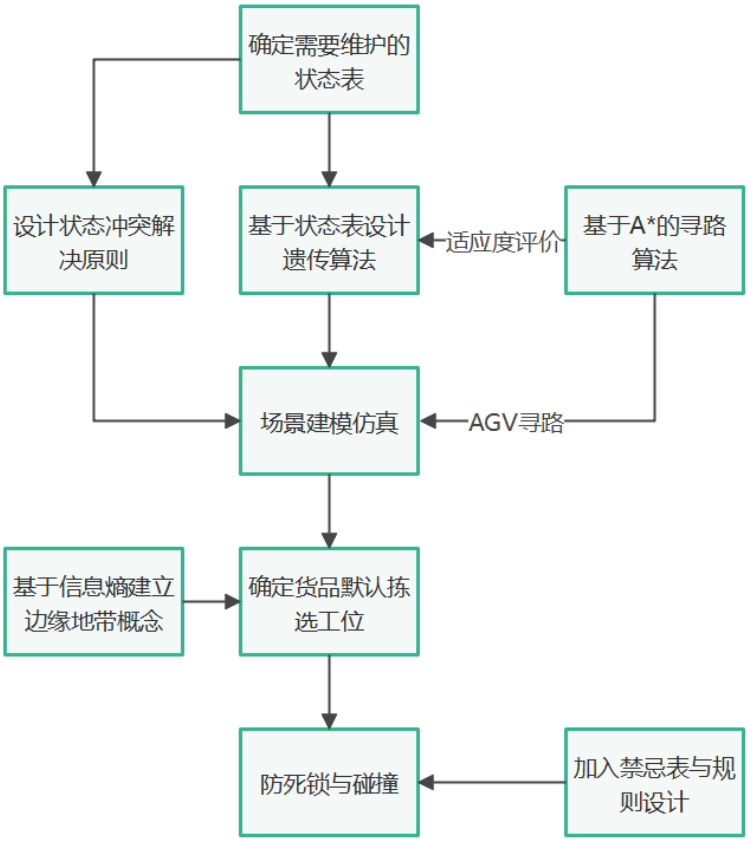


图 1 建模流程

以下对于题目中的 3 个问题进行分析：

2.1 问题 1

2.1.1 停靠位

通过对于题目描述 AGV 运作场景进行分析，可知在拣选工位处存在b个临时停靠位，对于这b个临时停靠位，拣选工位上的包装机器人可以直接停靠位上的货物进行拣选，并且题目对于一个托盘中货品的拣选时间进行了常数化假设。由题目描述可以

进一步推断出，只要在停靠位中剩余有足够的货物以应对某一次发货任务，那么，就无需 AGV 进行货物运输，对于该情况进行举例说明：

图 2 停靠位产品调取

可以存在如上图所示的情况，即在停靠位多余的未被取用的某类商品足以应付相关订单的情况。在这种情况下，无需调取 AGV 去获取相关货品。基于以上原因，本文对于**停靠位的机制进行了充分利用**。

此外，当某一工位的停靠位被托盘完全占满的时候，此时若有 AGV 来送货，则可通过 AGV 就地将停靠位中的空闲托盘取出，并放入新的带有需求货物的托盘，若此时没有空闲托盘，则需要 AGV 进行等待，关于**托盘交换**的情况如下图所示：

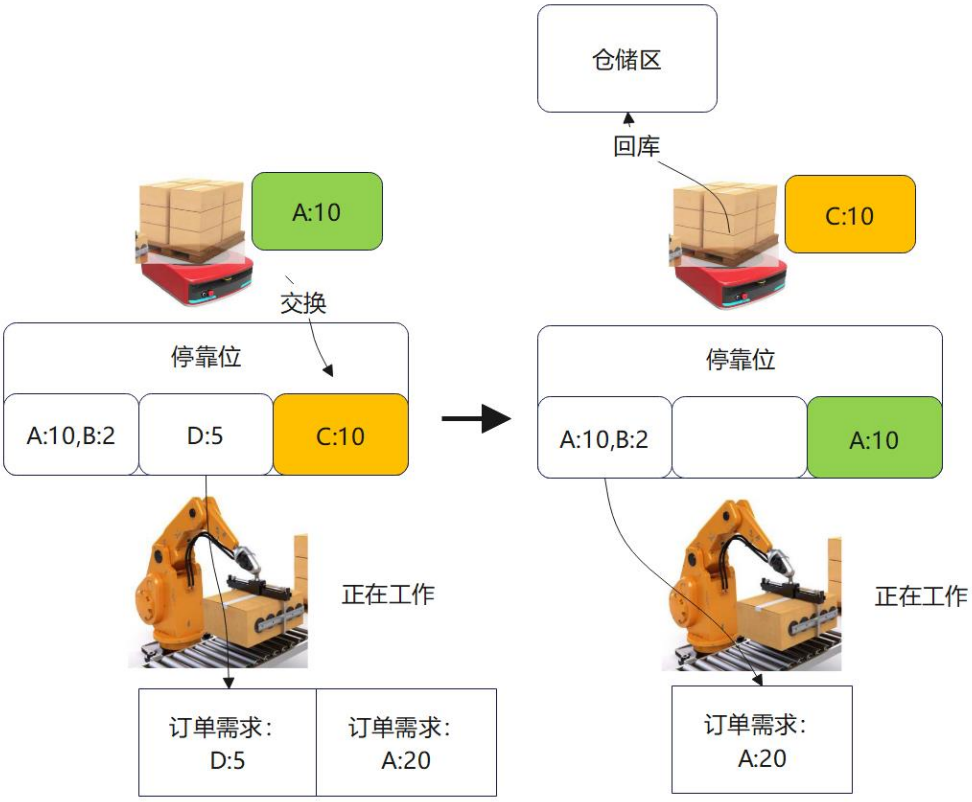


图 3 对停靠位中货物进行

通过一定的额外空间和时间交换并取出托盘，而后，AGV 需要根据情况将托盘回收或者回库，以避免托盘在路上停滞时间过长而造成交通堵塞。

2.1.2 订单出库

通过题目的描述，可以知道，对于同一订单，在实际发货的时候，无论是否为同批次，最终是以单个包裹的形式发出，基于此可知，题目并不硬性规定一下情形：

1. 同一订单由同一拣选工位完成
2. 同一订单同时打包发货

基于以上订单完成方式的特性，那么对于一个订单需求，可以采用分布式完成的方法，即有如下拣选方法：

1. 由多个拣选工位协同完成
2. 分批拣选

即可存在如下图示的订单满足方案：

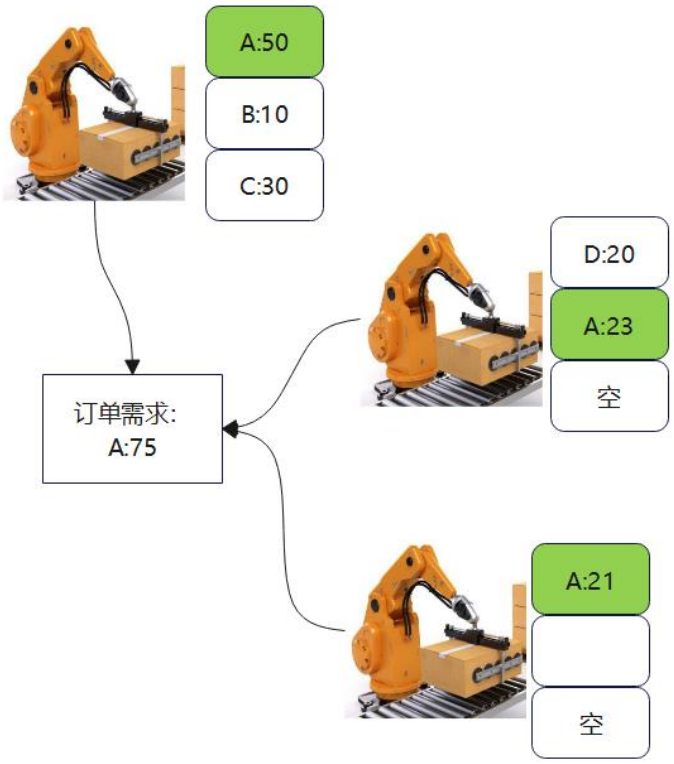


图 4 分布式满足订单需求

当一个订单被分配下来的时候，根据多个拣选工位的停靠位置中余留物品的情况，直接进行拣件，而不必调取 AGV 运输相关物品或减少 AGV 调取物品的数量，从而降低 AGV 所走的总路程。

同理，AGV 也不必将托盘送去某一个特定的工位进行统一拣选，而是就近选取送拣工位即可，进而也能够减小 AGV 的调度成本。

2.1.3 回库与回收

基于本文对于停靠位的充分利用原则，回库与回收的时机会且仅会出现于以下情况中：

1. 停靠位已满，需要换下占用停靠位的闲置托盘
2. 所有订单完成，AGV 将托盘回收

在本文中，假设托盘中货品对于仓储位置没有特殊要求，可放置于任意仓储位置，基于以上假设，AGV 在接收到回库任务的时候，会将托盘回库于最近的仓储位节点。而对于空托盘，本文设定 AGV 在换取到空托盘的时候，必须送回至托盘回收处。

2.1.4 回库托盘中带有订单需求的商品

考虑以下情况，由于停靠位被占满，相关托盘不得不被执行回库任务，但在回库过程中，出现了一个新订单，并且该订单需要的就是回库托盘中的货品。

此时 AGV 面临两种情况，即：

- 立刻重新规划路线，将手中的托盘交付到最近的拣选工位中。
- 将托盘继续送回原规划地点，在送回后，再做决策调度。

以上情况示例如下图所示：

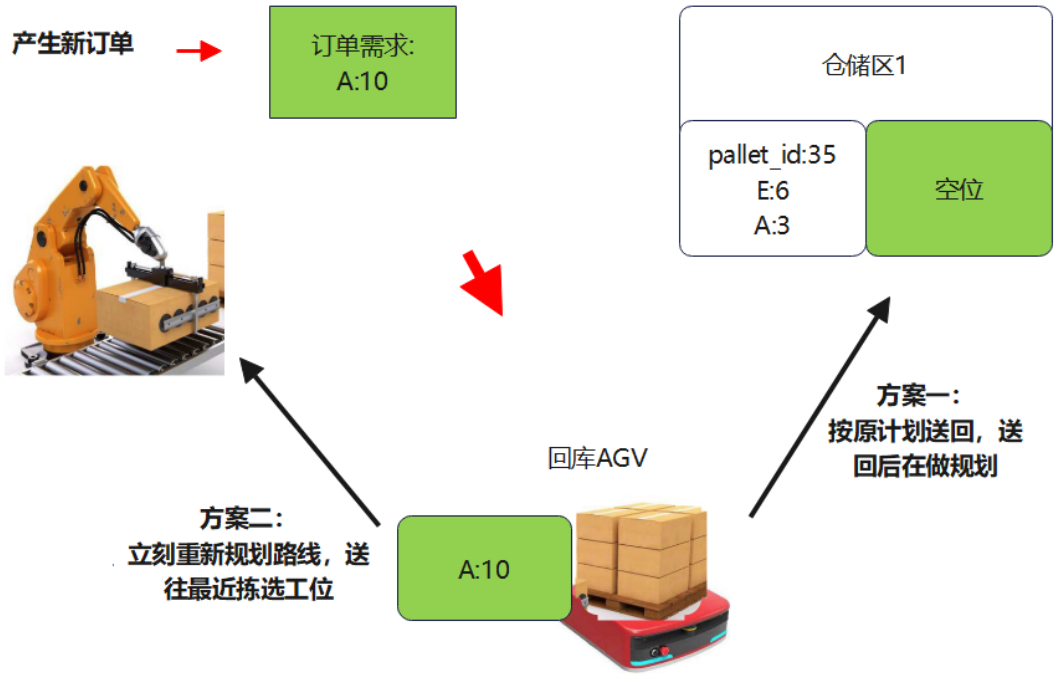


图 5 回库 AGV 应对新订单中相关货品需求的决策方法

在本文中，选择了**方案一**进行调度规划，使用方案一可以有效减少建模过程中的算法复杂度，有效简化问题。因为当一个 AGV 的调度方式发生更改的时候，会涉及全局中诸多信息的更改，更新起来过于繁琐。尽管相比较于方案二，方案一会增加 AGV 的运动路程，但本文中的托盘回库方式是就近回库的，因此在回库过程中多浪费的路程在可接受范围内。

基于以上特点，当订单派发下来时，根据相应货物、AGV 的分布情况，存在多种满足相应订单需求的调度方案，一一穷举来获取最优方案将有损时间效益，甚至会导致指令的发布速度无法跟上系统的调度速度。因此需要合适的算法对搜索空间进行剪枝，同时我们也需要维护多张表来辅助调度计算以提高效率。

2.2 问题 2

就指派某一 AGV 进行某一货物的**取货**并**送货（送拣）**这整个过程来说，**取货**路程是 AGV 完成任一分配任务的必经途径，当同一 AGV 对同一个货物进行分配的时候，其路程差异体现于 AGV 与送货目的地（拣选工位）之间的距离的差异。图示如下：

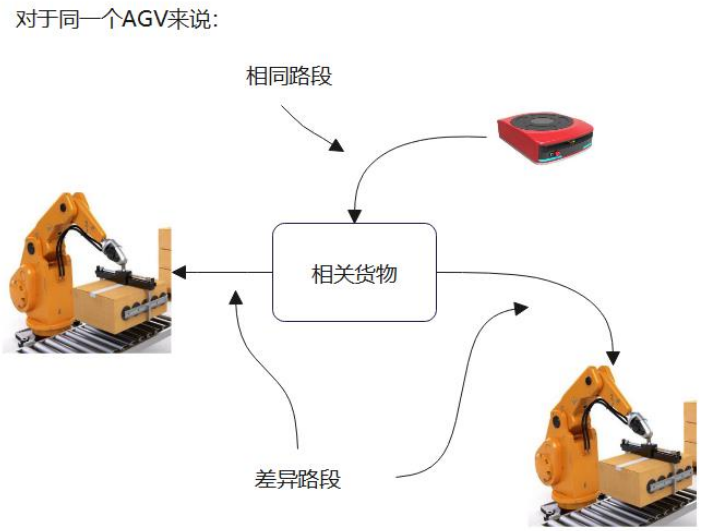


图 6 AGV 路程计算过程中涉及的路段

在问题二中，需要我们预先计算某一货品的默认拣选工位，同时兼顾如下情况：

1. 拣选机器人的负荷情况
2. 局部拥塞情况
3. 动态分区

要解决问题二，需要考虑如下问题：

1. 如何权衡一个存储位置的距离和货量的关系
2. 根据什么标准来评判一个拣选工位的工作量

2.3 问题 3

问题 3 需要在上述两个问题的基础上，防止 AGV 发生碰撞和死锁，需要我们设置合理的算法和防碰撞策略，使 AGV 能够智能的避免碰撞，特别是要对于某些特殊节点予以特殊关注。本文基于以上两问成果，在调度系统中加入了禁忌表，用于记录某一时刻中的不可占用位置，并在该表之上，设立了一系列规则来解决 AGV 的路径规划问题。

3. 模型假设

1. 拣选机器人的对于单个托盘中内容的拣选时间固定
2. 同一订单不必同批次同一拣选工位打包发货
3. 订单动态刷新，未来订单难以准确预测
4. 停靠位节点与拣选工位为同一格
5. AGV 运动速度为常量，以 AGV 运动一格的时间为时间粒度，拣选时间、订单刷新时间为该最小时间粒度的整数倍
6. 货品对于仓储位置无特殊要求
7. 忽略 AGV 在停靠位中交换托盘的时间

4. 符号与概念说明

符号说明：

表 1 符号说明

符号	说明
$\boldsymbol{\gamma i}$	突变率
$\boldsymbol{\varepsilon i}$	交叉互换率
$\boldsymbol{f_i}$	适应度函数的值
$\boldsymbol{f_{max}}$	最大适应度函数值
$\boldsymbol{f_{arv}}$	平均适应度函数的值
$\boldsymbol{N}$	托盘总数
$\boldsymbol{S_{总}}$	一次调度中 SVG 总运动距离
$\boldsymbol{P_{ij}}$	第 **i** 个托盘中第 **j** 个货物的个数
$\boldsymbol{A_1A_2\ldots A_N}$	遗传序列
$\boldsymbol{Set_i}$	第 **i** 个工位中潜在的拣件数
$\boldsymbol{T}$	产生新订单时间间隔
$\boldsymbol{t_0}$	拣件时长
$\boldsymbol{b}$	停靠位数量

概念说明：

送拣：指某个有货托盘正在被送往拣选工位拣收。

模糊地带：指某个仓储位置的相对于全部的拣选工位的距离中，第一近距离和第二近距离大致相等，尽管根据最短距离来说，理论上应当划分于最近的拣选工位，但仍具有划分在距离第二远的拣选工位中的潜力。这样的仓储位置称为“模糊地带”。

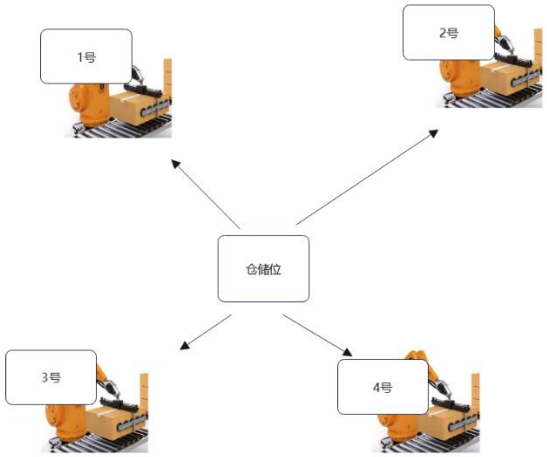


图 7 模糊地带示例

如上图所示，图中的仓储位理论上距离 3 号拣选工位最近，根据距离最近的规则来说归属于 3 号拣选工位，但其仍然存在归属于 4 号仓储位的潜力，因此该仓储位置属于模糊地带。

潜在拣选货品：对于被归入到某个拣选工位下的货品，称为潜在拣选货品，因为尽管该货品并未被该拣选工位实际拣取，但由于被归到了该工位下，因此本质上最终仍要该拣选工位进行拣选，因此称为这样的货品为潜在拣选货品。

订单分期：用于解决托盘在回库过程中，接到需要托盘中同款货物的订单的问题。当遇到此类情况的时候，将在订单中需求回库托盘中的相应部分的数量先行减去，延后到回库完成的时候进行订单调度决策，如下图所示：

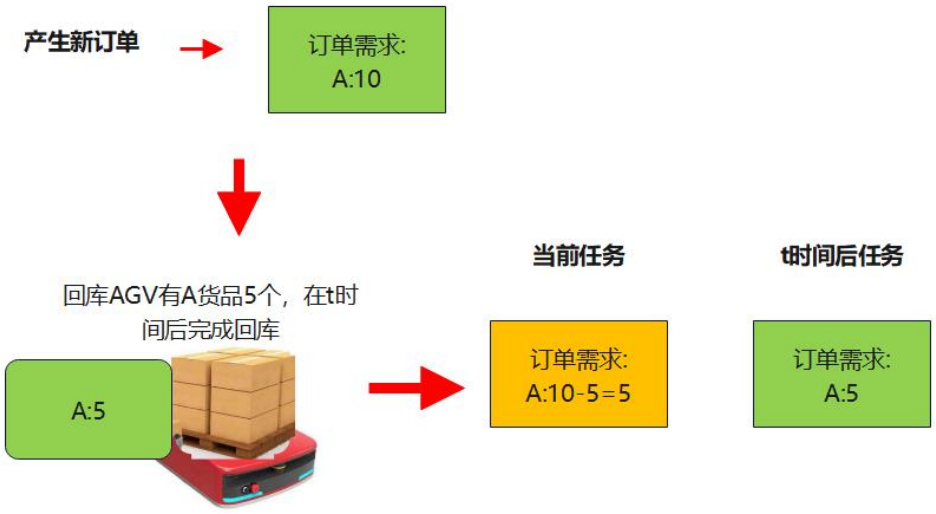


图 8 订单分期图示

在上图情况中，AGV 中携带了新订单中的相关货品的托盘进行回库，将新该订单拆分为两部分，第一部分先满足，第二部分后满足。以此解决在调度过程中出现的现有闲置资源无法满足新订单的问题。

5. 模型的建立

5.1 问题 1 模型的建立

5.1.1 调度系统模型

基于题目一的假设，我们首先建立六张需要动态维护的表，对于程序中使用到表，给出表头以及两行样例数据以供参考，具体如下：

停靠位余量统计表样例如下：

表 2 停靠位货品余量统计表

SKU	AMOUNT
298104	13:8,0
2195219	6:20,0

通过停靠位的特性可知，当停靠位中余量充足的时候，无需调度 AGV，因而能够提升调度效率，通过维护一张停靠位的余量表来提高查找效率。

其中 SKU 为货品编号，AMOUNT 记录相应坐标下的货品数量。

托盘表样例如下：

表 3 托盘表

PALLET_ID	X	Y	Sit	AGV_ID	T_remaing	SKU
10000	8	0	2	-1	10	104151:9,840211:35
10010	20	1	3	-1	20	746439:7,1302641:23

托盘表用于记录当前时刻下，托盘的状态，托盘有 3 种状态，其状态码和托盘状态如下：

1) **空闲**：可以随时供 AGV 调动；
2) **锁定**：在停靠位中将要或正在进行拣选的托盘，记录在该表中的托盘无法被交换；
3) **送拣**：正在被 AGV 送去拣选工位的路上；
4) **回收**：正在被 AGV 送去托盘回收点；
5) **回库**：正在被 AGV 送回仓储位置存储。

对于上表元素，解释如下：

表 4 托盘表属性注释

元素名	含义
X，Y	托盘所在的拣选工位坐标
PALLET_ID	托盘 ID
Sit	托盘状态

元素名	含义
AVG_ID	正在进行运输该托盘的 AGV
T_remaining	状态剩余时间
SKU	托盘上的货物余量

例如，上表中第一行的含义为在拣选工位(8，0)处的 ID 号为 10000 的托盘将要或正在拣件，在 10 个单位时间后完成拣选，该托盘上剩余 104151 号货物 9 件，840211 号货物 35 件。

AGV 状态表样例如下：

表 5 AGV 状态表

AGV_ID	X	Y	Is_active	Toward	T_remaining	PALLET_ID	FX	FY
1	31	14	1	4	10	10000	8	0
13	11	16	0	0	0	-1	-1	-1

AGV 有如下 2 种状态：

1) 移动；
2) 空闲

对于上表元素，解释如下：

表 6 AGV 状态表属性注释

元素名	含义
AVG_ID	正在进行运输该托盘的 AGV
X,Y	AGV 当前坐标
Is_active	是否处于移动
Toward	运动方向（0，1，2，3，4 分别对应静止、右下左上）
T_remaining	运动时间剩余
PALLET_ID	附带的托盘 ID
FX,FY	终点坐标

基于以上几个表，便能够解决一部分订单的需求。当一个新订单到来的时候，首先会比对停靠位余留货品表，当不能够满足订单需求的时候，接着比对送拣货品表，当送拣表仍然不能够满足需求的时候，比对回库表，若在回库表中存在相应货品，则进行订单分期，在回库中的部分订单会在回库完成再做调度决策，在扣除回库部分的货品数量余下订单则先行满足。

上述的提到的送拣货品表、回库表为视图，是基于基本表 AGV 状态表及托盘表动态生成的，上述表的运作流程如下：

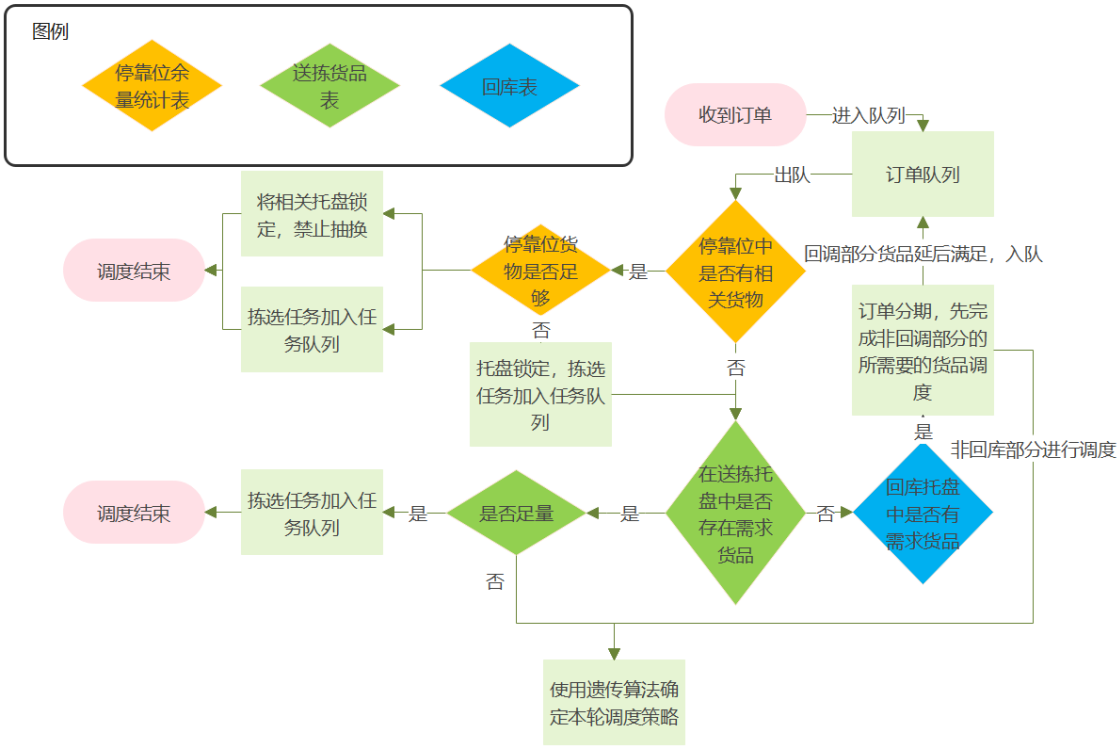


图 9 订单预处理策略

通过上述的订单预处理后，如果仍然无法满足订单需求，则需要调度 AGV 挑选托盘送货以满足需求。

基于以上订单预处理，建立以下调度系统的决策模型：

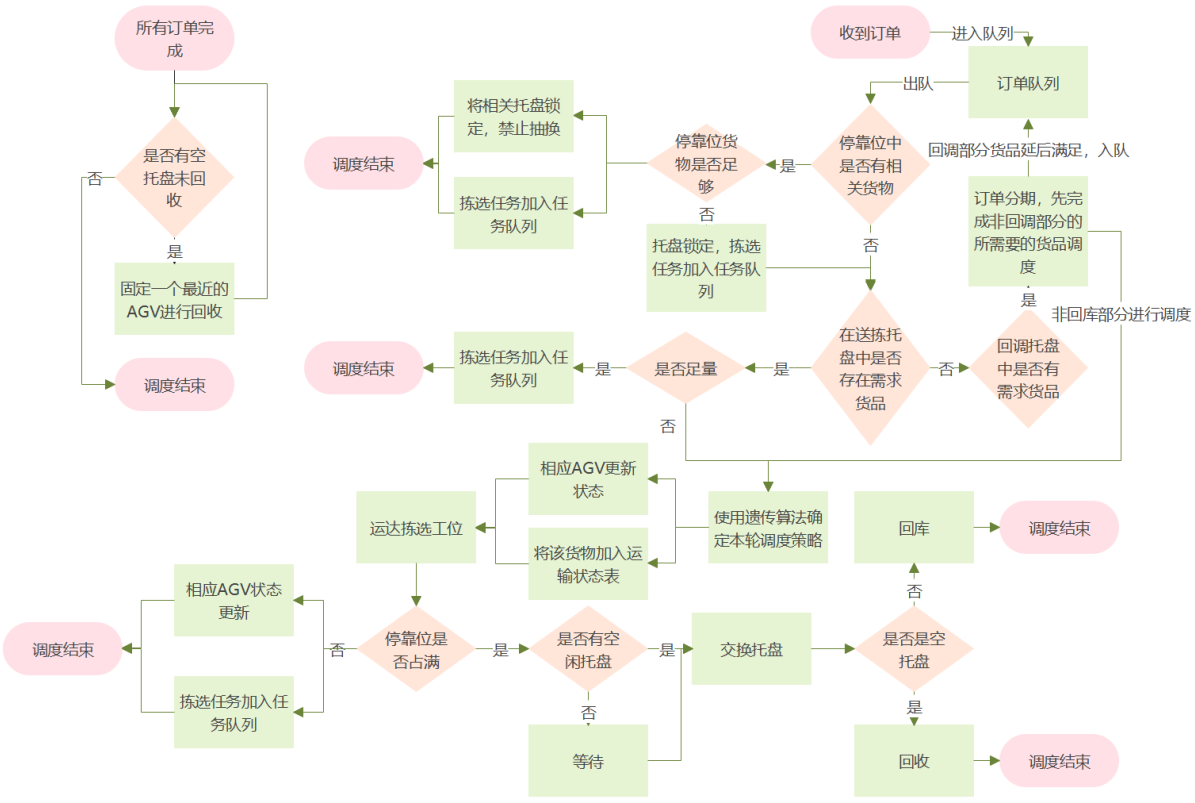


图 10 调度程序运作方式

上图为本文设计的调度模型，当系统收到订单信息后，仓库系统首先经过上述三个表的订单预处理，先行对符合需求的货品加入拣选机器人的工作队列中，若停靠位中资源不足以应对本次订单需求，则与**调度货品表单**进行比对，当正在运输的货物足以满足需求，则仍然无需派单。

基于以上过程，能够省去一部分的多余调度。在此基础上，构建遗传算法模型来对调度方式进行选择。考虑情况如下的一次决策过程：

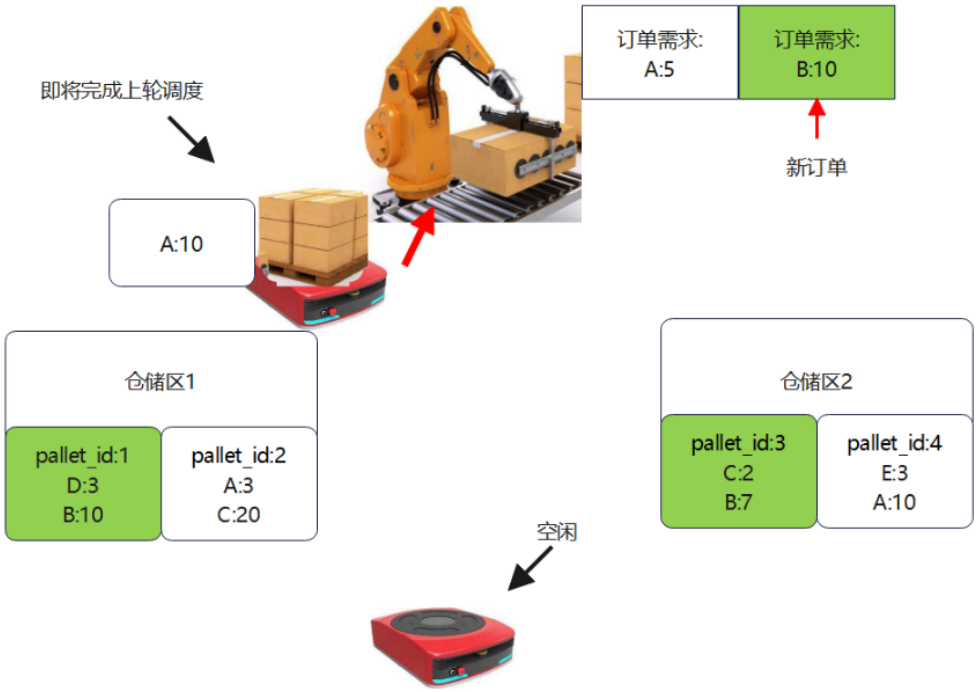


图 11 决策情况

如上图所示，在一个新订单产生的时候，存在一个可以马上接受任务的 AGV，但其距离较远；同时存在一个即将完成工作的 AGV，对于完成该新订单，存在以下取托盘的方法：

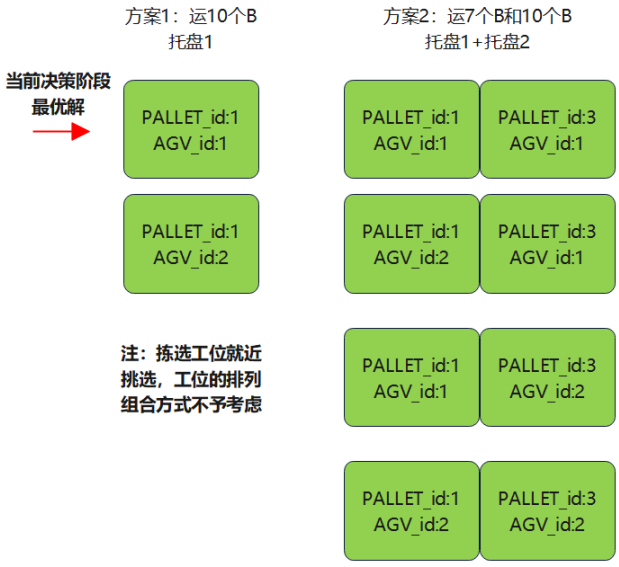


图 12 解决本轮决策的运输方案和最优解

完成以上订单，首先会有不同数量的满足方案的运输托盘的方案，例如上例中，为了满足一个需求为 10 个 B 货品的订单，既可以只运输 1 号托盘，也可以同时运输 1 号和 2 号托盘。对于方案 1，存在2^1种可行的运输方案，对于方案 2，存在2^2种可行的方案，其中底数为 AGV 总数，幂为托盘数量。

在以上可行解中，通过计算相应的路程代价，以最小路程和为目标，得出本轮决策的最优调度方案。

根据以上示例演示，引出以下仿真模型和数学抽象模型。

5.1.2 仿真实验模型

根据题意和上述调度系统的确立，建立栅格无人仓场景，模拟无人仓在接受订单的时候，货物的调动情况，基于 Python 进行仿真场景搭建，使用 Matlab 进行 AGV 流调可视化。定义相关对象如下：

表 7 仿真对象

对象名	对应实体
Pallets	托盘
Storage	储位
AGV	AGV
SortMachine	拣选工位

```python
class Pallets:  # 托盘
    def __init__(self, x, y, id_, goods):
        self.x = x
        self.y = y
        self.id = id_
        self.goods = goods

class Storage:  # 储位
    def __init__(self, x, y, id_, PlateId=-1):
        self.x = x
        self.y = y
        self.id = id_
        self.PlateId = PlateId

class AGV:  # AGV
    def __init__(self, id_, x, y, pid=-1, toX=-1, toY=-1):
        self.id = id_
        self.x = x
        self.y = y
        self.pid = pid  # AGV上存放的托盘
        self.toX = toX if toX+1 else x
        self.toY = toY if toY+1 else y

class SortMachine:  # 拣选工位
    def __init__(self, x, y, id_, work_list=-1):
        self.x = x
        self.y = y
        self.id = id_
        self.work_list = [] if work_list == -1 else work_list #拣选工位的任务单
```

图 13 仿真实验对象代码定义情况

对于题目中提供的数据进行统计，对于本次任务的数据规模进行评估，以选择合适复杂度的算法进行编程实现，统计题目数据形成如下表格：

表 8 数据统计

数据项	大小
货品总个数	8865
货品种类数	181
托盘数/仓储位数	148
订单数	675
AGV 数	20
拣选工位数	6
无人仓面积	32*22

仿真方式：

依照 AGV 运动一格的时间作为时间切片，进行基于**离散事件**的形式进行仿真模拟。

在一个单位时间片中，程序将完成以下行为：

订单接收：判断当前时刻是否有新的订单下发，如果有，则更新当前**货品累积需求**状况。

需求预满足：基于订单信息整合该决策轮次的货品总需求，按照当前需求货品在停靠位、调度过程中的分布情况，进行需求预满足，若不能满足，进入调度决策。

调度决策：依据托盘、AGV 情况，依照最短路径原则，进行分配任务，同时更新被分配到任务的 AGV 的状态。

全局状态更新：首先对 AGV 进行位置更新，让其在被分配的任务中前进一个单位长度。如果任务完成，则更新 AGV 的状态和对应的储位或者工位的状态。

拣选工位工作：最后处理每一个工位，令每一个工位处理当前正在处理的任务，依据工位处理任务的时间来更新工位任务状态。

场景建模：

根据题目提供的信息，基于 python 进行无人仓场景初始化，并借助 Matlab 进行可视化，仓库建模结果以及初始 AGV 布局如图：

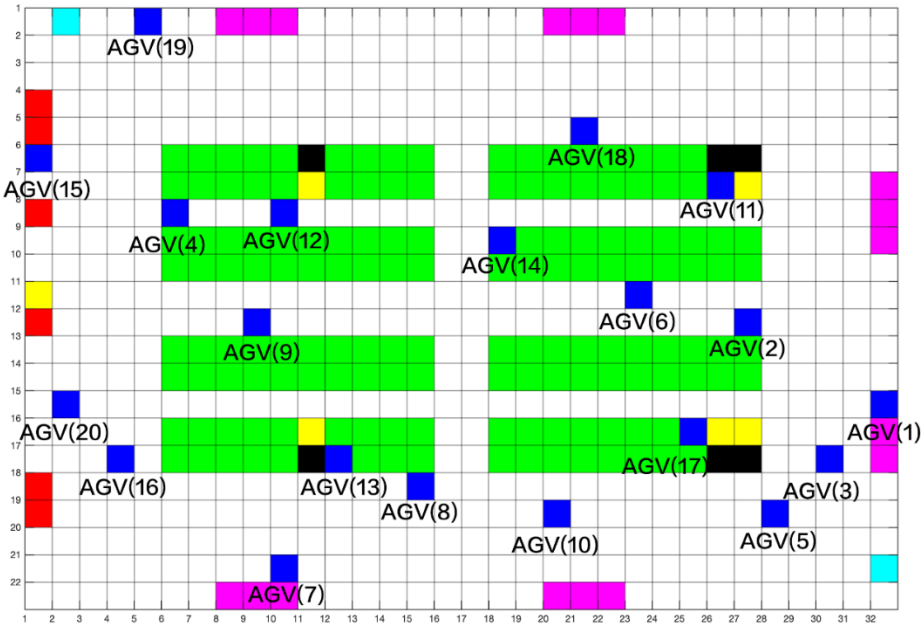


图 14 AGV 初始布局

其中，蓝色方格表示 AGV，相应编号如上图所示。限于 Matlab 的功能限制，本文所使用的节点编色与题目有所出入，但其代表内涵与拓扑结构与题目一致。

5.1.3 遗传算法模型

在 AGV 调度的研究中，常常结合遗传算法来进行调度策略的优化。如有基于改进 NSGA-II算法的防死锁任务调度方法[1]等。在本此建模任务中，则将遗传算法用于计算某一决策环节的最优解。

在本文中，基于遗传算法对一轮中的搜索空间进行剪枝，协助快速决策，过程如下：

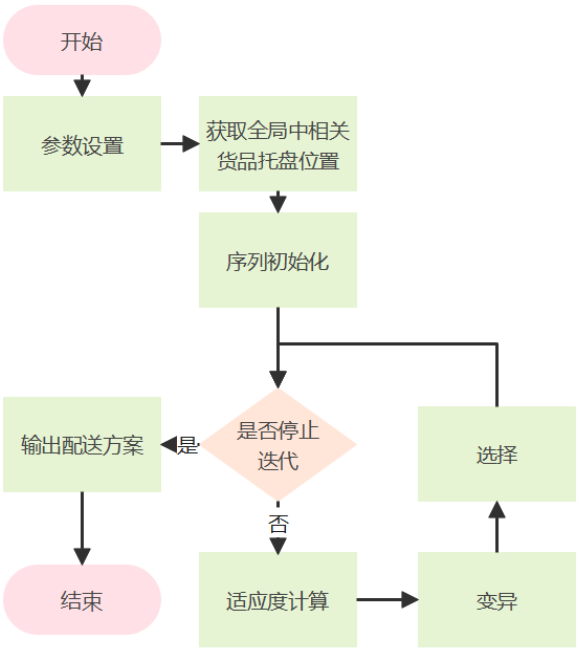


图 15 基于遗传算法进行调度决策

在本文中，决策环节的特点是面向货品需求量进行整体调度，即不只立足于一个个订单，而是对本轮所有订单中需求的货物整合后，进行统筹满足，并且已有任务的AGV 仍在新任务的范围中，目的在于总调度路程最小。

根据题意，基于现实场景中的全局优化出库效率的思想，题目希望能够使机器人尽可能的忙碌，并且最小化搬运机器人的行走总路径。即一轮决策中希望以下公式成立：

$$\min S_{总} = \min \sum_{i=1}^{20} s_i \tag{1}$$

其中，$S_{总}$为本轮决策中所有 SVG 将要行走的总路程，s_i为第i个机器人在本轮决策中需要行走的路程。在这里，由于对于订单的更新情况无法准确预测(假设 3)，则本轮决策环节全局最优解可由子路程最优叠加所得，即：

$$\min S_{总} = \min \sum_{i=1}^{20} s_i = \sum_{i=1}^{20} \min s_i \tag{2}$$

上述目标需要在选取到满足当前订单需求的货物数量为前提，即目标函数服从以下约束：

$$s.t. \begin{cases} V_j \le \sum_{i=1}^{N} \sum_{j=1}^{m} p_{ij} \\ p_{ij} \ge 0, V_j \ge 0, 1 \le j \le m \end{cases} \tag{3}$$

其中，V_j为本轮决策中，订单需要第$\boldsymbol{j}$号货品的数量，P_{ij}为第$\boldsymbol{i}$号托盘为订单中第$\boldsymbol{j}$号货品提供的数量。$\boldsymbol{N}$为托盘个数，$\boldsymbol{m}$为当前决策轮次中，需要满足的所有货品的种数。

基因编码：

本文使用整数序列进行编码，形式如下：

图 16 基因序列

如上图所示，定义序列$A_1A_2A_3A_4A_5...A_N$，A_j表示第j个$(1 \le j \le N)$托盘被A_j号 AGV 进行运输，其中A_j取值含义如下：

$$A_j=\begin{cases}SGV\_ID, \text{第j号托盘被编号为SGV\_ID的SGV进行运输}\\ 0, \text{该托盘不被运输}\end{cases} \tag{4}$$

定义以下$active$集合：

$$active=\{i|\text{第i号托盘中有订单中需求的货品}\} \tag{5}$$

则约束条件进一步变为：

$$s.t.\begin{cases}V_j \le \sum_{i\le N, i\in active, A_i\ne 0} p_{ij}\\ p_{ij}\ge 0, V_j \ge 0, 1\le j\le m\end{cases} \tag{6}$$

即选取有V_j中相关货物的托盘进行计算，简化运算步骤，以上方式为定长编码加上条件选择的编码方式，目的为方便于统一套用遗传变异模型。遗传算法的关键部分设计如下：

交叉变异与选择

该部分参数设计如下：

$$\begin{cases}inrate=0.2\\ round=300\end{cases} \tag{7}$$

$$\gamma_i=\begin{cases}c_1+(c_2-c_1)\dfrac{f_{\max}-f_i}{f_{\max}-f_{avg}}, f_i\ge f_{avg}\\ c_3, f_i<f_{avg}\end{cases} \tag{8}$$

$$\varepsilon_i=\begin{cases}c_4-(c_5-c_4)\dfrac{f_{\max}-f_i}{f_{\max}-f_{avg}}, f_i\ge f_{avg}\\ c_6, f_i<f_{avg}\end{cases} \tag{9}$$

其中γi为变异概率，εi为染色体交换的概率，$inrate$为优秀率，$round$为迭代次数，其中γi和εi使用自适应算子，c_1、c_2、c_3、c_4、c_5、c_6为常数。以上算子用于缓解遗传算法容易收敛于局部最优解的问题并提高搜索效率。

当种群开始迭代的时候，需要较低的交叉和变异几率来保存优良品种。当适应度较高的时候，通过提高变异、交叉的概率来走出局部收敛的问题。

染色体交叉为对两个序列$A_1A_2\ldots A_N$和$B_1B_2\ldots B_N$，当两个序列都发生交叉的时候，则随机抽取出两个不同的数字 i 和 j，进行位点交换，交换结果如下：

$$A_1\ldots A_{i-1}B_i\ldots B_jA_{j+1}\ldots A_N$$
$$B_1\ldots B_{i-1}A_i\ldots A_jB_{j+1}\ldots B_N$$

染色体变异则依据变异算子概率进行位点数据随机赋值，其中数值范围为0~AGV 总数。

在完成以上交叉变异后，取出前 20%的优秀序列进入下一轮迭代。

5.1.4 A*算法模型

评价函数依托于 A*算法进行代价计算。

本文基于 **A***算法进行 AGV 的路径规划，相较于 Floyd 的静态寻址，A*算法可以实现动态寻址，因而能够与接下来加入的预防碰撞、死锁规则实现较好兼容。

A*算法本质上是对 **dijkstra** 算法的优化，加入了**启发函数**。在扩展每一个节点的相邻节点时，选择相邻节点中代价最小的节点放入队列中进行扩展，最后得到的结果在大多数情况下是最短路，且算法效率明显高于 **dijkstra** 算法。

本文基于目标到 AGV 的曼哈顿距离来实现启发函数，A*算法伪代码如下：

表 9　A*算法伪代码

Algorithm 1: ShortestPath

```
创建并初始化地图棋盘;
openlist.append(currentposition);
while openlist! = [] do
    取出第一个（F 最小，判定最优）位置;
    currentposition = openlist[0];
    openlist.remove(currentposition);
    if 成功找到解 then
        逆溯路径;
        倒序输出;
        return
    end
    else
        将下一步可到达的位置加入 openlist, 并检查记录的最短路
         径 G 是否需要更新，记录最短路径经过的上一个点;
        维护当前已知最短 G, 如果未遍历或需更新;
    end
end
```

5.2 问题 2 模型的建立

5.2.1 模糊地带

模糊地带是本文用于描述位于依据距离进行划分后处于不同簇交界处的仓储位置的概念，处于该地带的存储位，在必要的时候，需要进行再划分以平摊各拣选工位

的工作负荷量。对于模糊地带借助信息熵公式进行判断，如下：

$$\begin{cases} rate_{old} = \dfrac{d_{old}}{d_{old} + d_{new}} \\ even\_rate = -rate_{old} \bullet \log_2 rate_{old} - (1 - rate_{old}) \bullet \log_2 (1 - rate_{old}) \end{cases} \tag{10}$$

其中，d_{old} 和 d_{new} 分别某个仓储位中的托盘距离工位的**调度距离**，而给曼哈顿距离。$even\_rate$ 描述了某个仓储位归属于某个拣选工位的确定程度，定为**模糊程度**，其中 $rate_{old}$ 服从伯努利分布，因此，$even\_rate$ 介于 0~1 之间，为二元信息熵，其随 $even\_rate$ 变化的分布如下：

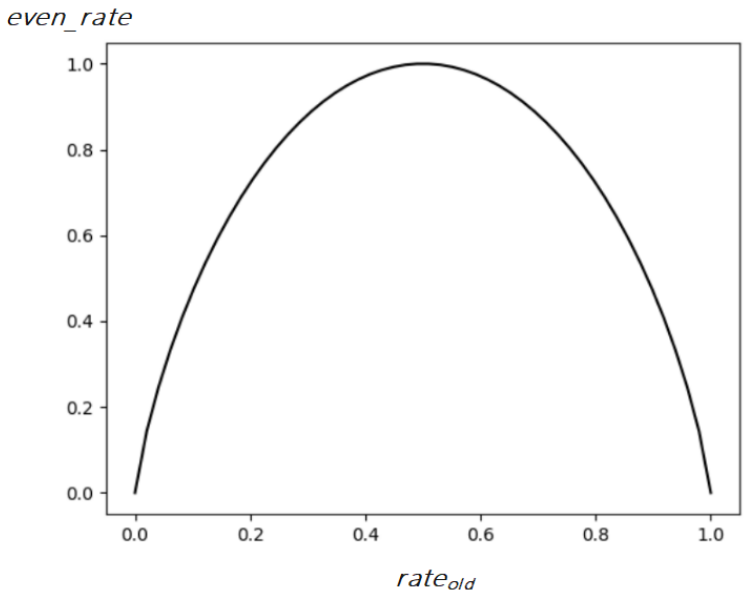


图 17 $even\_rate$ 与 $rate_{old}$ 的关系图像

由上图可知，当某仓储位应该分配于哪个工位下的进行拣选的程度越不确定，那么其信息熵越大，即**模糊程度**越高。

基于以上模糊地带定义，需要设立一个阈值，当基于第一近距离和第二近距离所得的模糊程度**高于个阈值**的时候，定为模糊地带。

5.2.2 模糊地带再分配

同时，基于信息熵公式，再定义描述集合的均匀程度的方法，给拣选工位进行编号，记Set_i为第i个拣选工位下分到的拣件总数，即**潜在拣选货品总数**，基于每个工位下的拣件总数，建立如下**分配均匀描述模型**：

$$\begin{cases} P_{ti} = \dfrac{Set_i}{\sum_{i=1}^{m} Set_i} \\ P_t = \dfrac{-\sum_{i=1}^{m} P_{ti} \log_2 P_{ti}}{\log_2 m} \end{cases} \tag{11}$$

$$m=6 \tag{12}$$

其中m为工位数量，P_{ti}为在t时刻，编号为i的拣选工位的潜在拣选货品总数在当前剩余拣选货品所占的比例，称为**潜在拣选工作贡献率**。P_t用于刻画无人仓在t时刻的**潜在拣选工作**均匀程度，称为**负荷均衡率**。当P_t接近于 1 的时候，系统的分配趋于负荷均衡；当P_t接近于 0 的时候，系统的分配趋于负载失衡。

动态分配方法如下：

在无人仓实际运作过程中，若 AGV 要选取的托盘处于模糊地带，则需要确定是运输去原拣选工位，还是新拣选工位。不妨设该托盘将要被分配到的两个拣选工位拣件工作集合为B_{old}和B_{new}，一个为归原有工位拣件对应的托盘集合，另一个为归第二近的拣选工位拣件的托盘集合，分类标准如下：

$$\begin{aligned} &\Delta P = P_{tbefore} - P_{tafter} \\ &A_j \in \begin{cases} B_{new}, \Delta P \geq 0 \\ B_{old}, \Delta P < 0 \end{cases} \end{aligned} \tag{13}$$

其中，$P_{tbefore}$为未归入新集合情况下，全体工位潜在工作量的**负荷均衡率**，P_{tafter}为归入新集合后的负荷均衡率。基于该次划分是否让全体工位的潜在工作量变得更加均匀（负荷均衡率变大）来决定该仓储位上的托盘是否被运往另外的一个拣选工位。

静态分配方法如下：

静态再分配考虑将全部位于模糊地带中的仓储位点全部都进行一次重新分配。即在某一时刻的仓储布局中，求出全部仓储位的默认工位。

5.3 问题 3 模型的建立

死锁和碰撞的本质原因是路径规划错误，基于此，引入禁忌表对寻路过程进行校正。禁忌表用于保护先进行路径规划的 AGV 的下一目标地不被路径规划，从而避免 AGV 的路径规划冲突，进而解决死锁、碰撞等问题。

5.3.1 基于禁忌表改进 A*算法

本文基于禁忌表的广度优先搜索算法伪代码如下：

表 10 基于问题 1，加入禁忌表的改进 A*算法

Algorithm 2: 结合禁忌表的 A*
1 def get_route(u,v):
2 init queue
3 向 queue 中放入初始值(u,0,0)
4 while not queue.isempty():
5 node = queue.pop(0)
6 if node[0] == v:
7 p = node[2]
8 cnt = 0
9 while p!=u:
10 令 p 节点在 node[1] - cnt 时刻被占用
11 在路径列表中记录 p 节点
12 cnt++
13 queue 删除其中第一个元素
14 for j in node[0]周围可到达且在 node[1]+1 时刻未被占用的节点:
15 计算该节点的估价函数值并放入 openlist 中
17 在 queue 的尾部按照估价函数升序依次插入 openlist 中的元素

5.3.2 主动退让机制

为了避免完成工作任务的 AGV 因为没有及时接受到任务而在拣选工位中等待而造成的死锁，为 AGV 设计规则，即 AGV 在完成任务后，并且没有接到新任务，则会自行前往拣选工位附近的一个开阔的空地处。

5.3.3 空闲 AGV 无体积化处理

对于没有接到任务的 AGV 来说，根据**主动退让机制**，AGV 会停留于一个开阔的空地处，在这种情况下，尽管该 AGV 可能位于另一个 AGV 所规划的路径上，但由于空闲 AGV 处于开阔地带，即能够**自行避让**，因此，本文将空闲的 AGV 仍然视为无体积。情景如下：

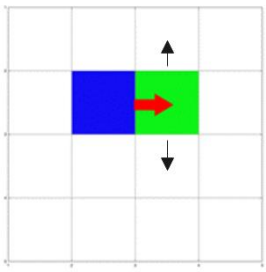


图 18 基于上个规则，空闲机器人一定可以自行避让

5.3.4 繁忙停等

在加入禁忌表后，当 AGV 运动起来后，如果目标工位不可达，则会随机游走。

因此，如果基于上述假设，尽管无需专门设置排队机制解决工口被繁忙的问题，但“无路可走”的 AGV 在对应工位繁忙的时候会随机移动，直到对应工口中开发可供调换的停靠位后，才重新规划路线成功送货，这样就会潜在增加移动成本。为解决工位繁忙的问题，在程序中加入如下规则：

表 11 空地停等

Algorithm 3: 停等机制
1 if 对应工位繁忙 and AGV 位于工位附近空地：
2 AGV 本轮停等

由于繁忙，AGV 首先在到达工位附近后，开始随机游走，当游走到工位结束繁忙则重新开始送货。当在游走的时候，遇到满足上述停等条件后，AGV 会原地等待，此时的 AGV 基于以上空闲无体积规则，并不会干扰其他 AGV 正常的路径规划。

基于以上的改进 A*算法、主动退让机制、空闲无体积化处理、工位繁忙停等后，本系统能够有效避免死锁、碰撞等问题。

6. 模型结果

6.1 问题 1

建立仿真实验场景，在场景中进行 orders.csv 中的订单情况进行调度模拟。
相关参数设定如下：

表 12 仿真场景相关参数

参数
停靠位数量b
拣选时长t_0
订单下派时间间隔T

表 13 实验参数设置与仿真结果

序号	b	t_0	T	AGV 忙碌率/%	订单平均移动量/格
1	4	5	4	22.4	19.09
2	4	5	5	28.3	22.38
3	4	6	4	23.9	22.60

实验数据解释：

以上仿真结果，为进行了 5 次实验后取均值的计算结果，AGV 忙碌率、订单平均移动量两个评价指标计算方法如下：

$$AGV\text{忙碌率} = \frac{\text{单位时间中实际运动量}}{\text{单位时间理论运动量}} \tag{14}$$

$$\text{订单平均移动量}=\frac{AGV\text{总运动路程}}{\text{订单总量}} \tag{15}$$

其中，由于单位时间中 AGV 运动一格，因此单位时间的理论运动量为 **20**(*AGV 数量* × *AGV 单位时间移动速度*)，其中订单总量为 **675**。

基于序号为 1 的参数设定，调度相关结果如下：

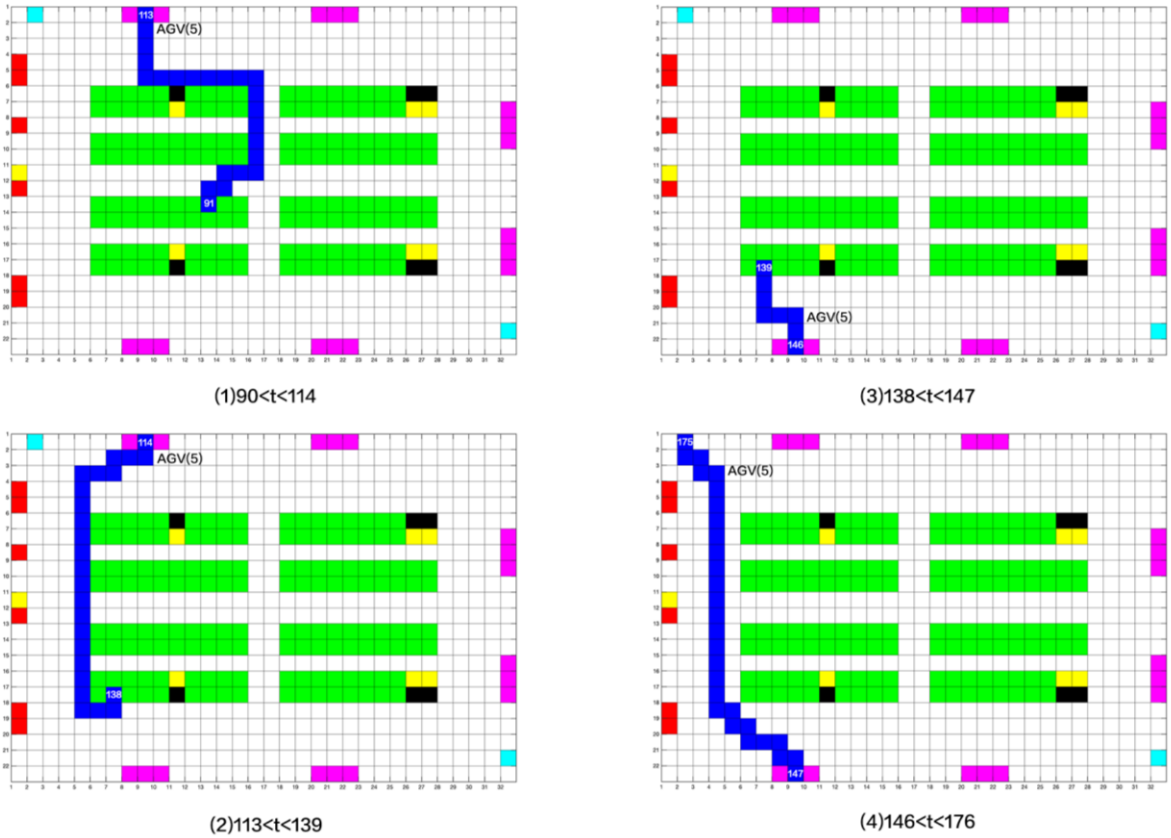


图 19 5 号 AGV 在一次仿真实验中 90~176 时间段内的活动轨迹

对于上图仿真情况进行相应解释：

1. 91-113：5 号 AGV 从停靠的仓储位置上直接取下相应货物，送去左上角的拣选工位；
2. 114-138：5 号 AGV 在拣选工位处完成工作后，马上接到了下一个调度任务，而后赶往下一个仓储位置处获取货品；
3. 139-146：5 号 AGV 将托盘送到左下角的工位中；
4. 147-176：5 号 AGV 换下空托盘，而后将空托盘运往托盘回收处。

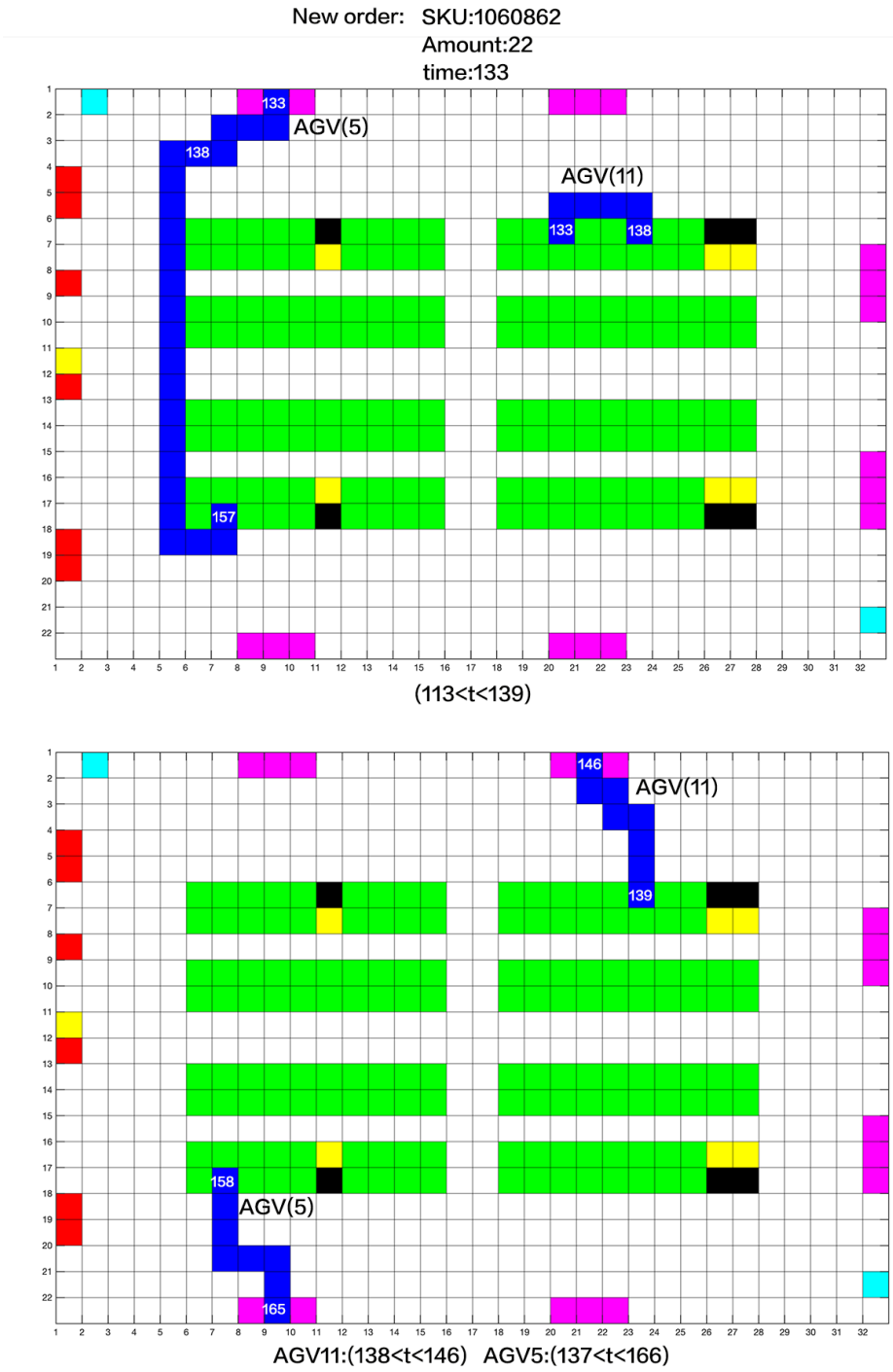


图 20 AGV 对于同一订单的协同运输

上图所示的为此次仿真过程中，对与 5 号 AGV 进行协同调度货物的 11 号 AGV 的活动情况进行观察。

对 11 号 AGV 的活动的说明如下：

1. 133-138：11 号 AGV 在刚刚完成回库任务后，立马去旁边的仓储位置中取满足订单需求为 1060862 号货品的托盘；
2. 139-146：11 号 AGV 在取到货物后，将托盘送往最近的拣选工位。

6.2 问题 2

基于模糊地带模型，对于初始仓储情况进行静态再分配，即在一个时刻中，对所有位于模糊地带的托盘进行再次分配，模拟实验结果如下：

表 14 不同阈值下的分配方案结果

阈值	负荷均衡率	工位潜在负荷量标准差	储位-工位距离和
1(默认情况)	0.926	731.54	1202
0.995	0.942	620.69	1210
0.99	0.949	614.35	1240
0.985	0.952	564.10	1306
0.975	0.965	493.75	1362

对于以上分配评价指标，详细可以参考 **5.2 问题 2 模型的建立**，对于以上指标简述如下：

负荷均衡率、工位潜在负荷量标准差：衡量一个分配方案的合理程度的指标；

储位-工位距离和：衡量一个分配方案的送拣成本。

其中对于一次分配，最优情况下是潜在工作量既均匀同时成本最小。将上述评价指标负荷均衡率和托盘-工位距离总和关系的可视化进行如下展示：

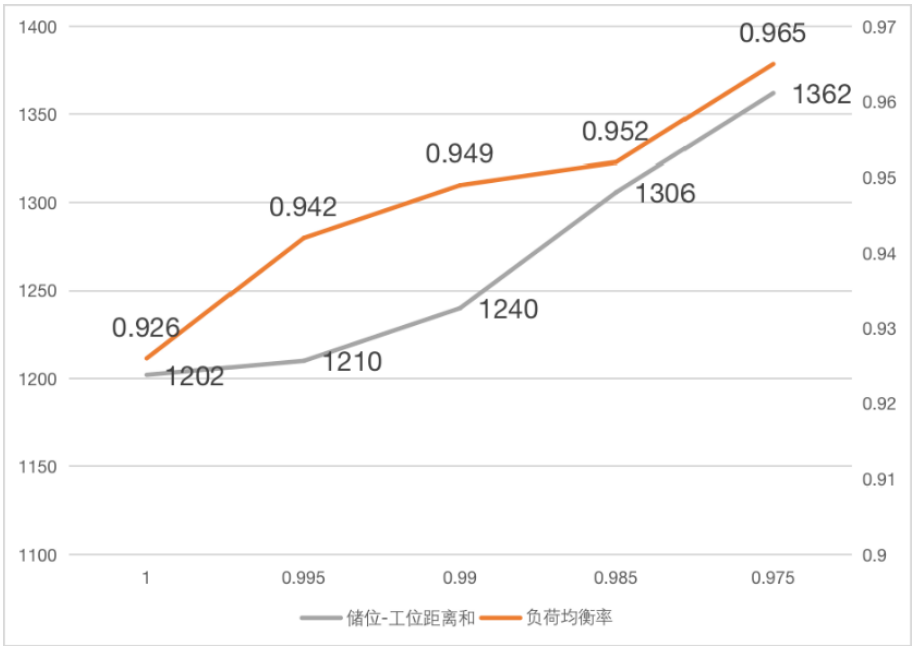


图 21 储位-工位距离 负载均衡率

对于上述方案中，较优分配方式为阈值为 **0.995** 的分配情况，在此情况下，在比原始分配方案多 8 格额外运动成本的情况下，均匀负荷率提升效果最明显。对于以上不同阈值的仓储位分配结果，对应可视化如下：

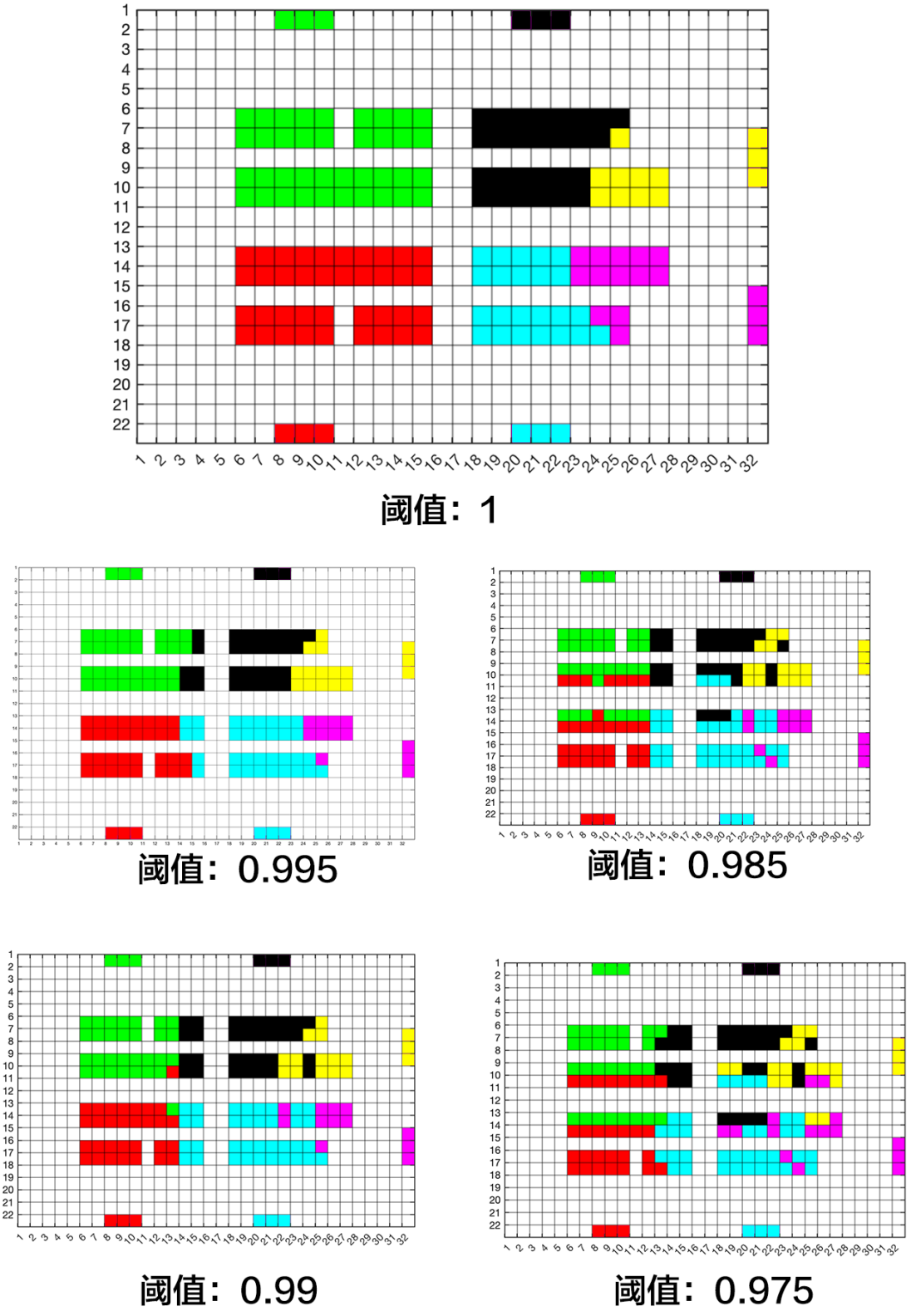


图 22 不同模糊地带大小的负载分配结果

注：以上距离基于 A*计算所得最短距离，而非曼哈顿距离

其中，阈值越小，模糊地带越大，需要进行再分配的区域越大。再分配所涉及到的区域越大，最终的交错混杂区域也就越大，当无人仓越大的时候，需要选取相对较

小的阈值，以提高负荷均衡的效果。模糊程度与距离比例关系为事件概率与二元信息熵关系，如图：

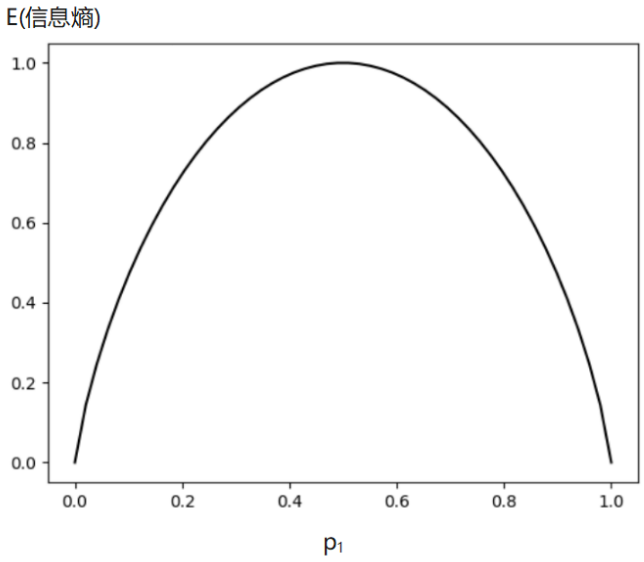
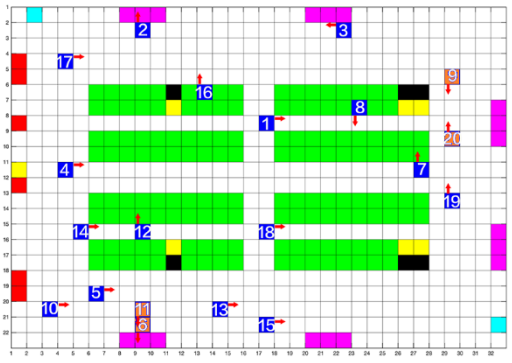
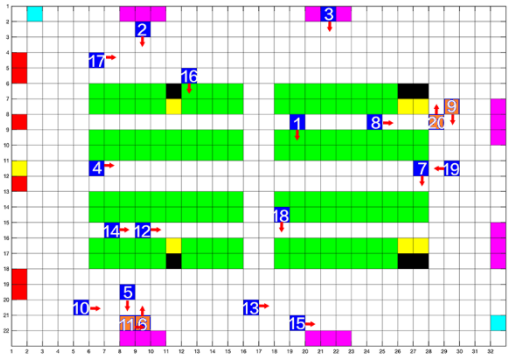


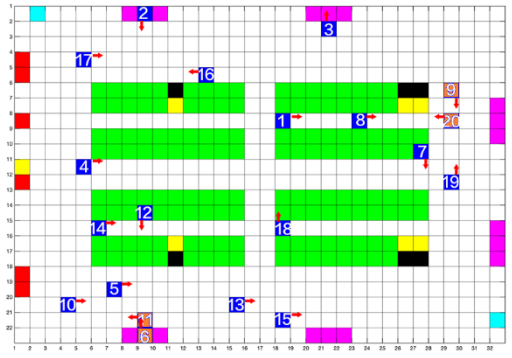
图 23 二元信息熵图像

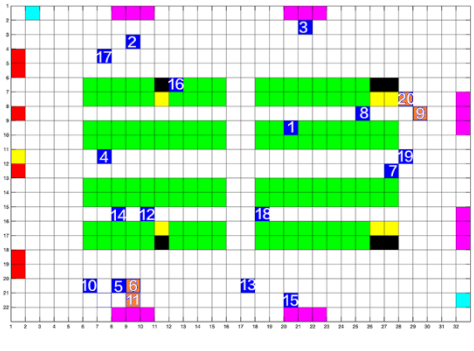
6.3 问题 3

进行默认参数环境下加入**禁忌表**机制后的仿真实验，截取在 1321~1324 时间段中仿真情况：


t=1321


t=1323


t=1322


t=1324

对于上图所示结果说明如下：

表 15 基于禁忌表的冲突解决方法演示说明

时间段 t	冲突描述
1321	**11 号和 6 号** AGV 同时想要将托盘送去左下角处的拣选工位； **9 号和 20 号** AGV 正在相向而行。
1322	**6 号** AGV 进入工位放置完货物，此时 **11 号** AGV 由于目的地被占领无法到达，进行随机游走，在图中，**11 号** AGV 选择左移。**6 号**由于没有活动空间，不进行移动。 根据禁忌表，由于 **9 号** AGV 先进行路径规划，因此会将 9 号的下一刻的位置加入禁忌表，**20 号** AGV 由于路径中不可行，重新寻路，而后向左运动
1323	基于本文设置的运动机制，上述两对路径冲突问题成功解决

7. 灵敏度分析

对于最终模型的相关参数进行灵敏度实验，结果如下：
对拣选工位的拣选时长t_0依次取 3，4，5，6 进行仿真实验，结果如下：

表 16 总路程代价对于拣选时长变化的敏感度

变化起止点t_0	3,4	4,5	5,6
Δt_{0rate}	0.33	0.25	0.2
$\Delta S_{总}$	0.02698	0.02713	0.02829

通过上表可以知道，在的最终模型中，总路程代价对于拣选时长的变化，总体在但不太敏感。因此，对于实现尽可能减小总调度路程的目标来说，拣选时长的波动对对于结果的影响不大，但在一定程度上会影响到任务的完成总时长，最终对总调度效率造成负面影响。

单就满足最短路径来说，重点的优化方向应为搜索调度方案的遗传算法的遗传变异因子的参数。

8. 模型的评价、改进与推广

8.1 模型评价

模型优点：

1. 模型同时结合了遗传算法和 A*两个启发式算法；
2. 将遗传算法进行改进，使用了变异率算子和交叉率算子，因此可以大大提升收敛的速率，同时也可以更好的达到最佳的位置；

3. 基于信息熵建立模糊地带的概念，实现保证距离短的同时，提高负载均衡率；
4. 充分利用了停靠位的货品暂存机制；
5. 建立了合理的调度系统；
6. 进行了仿真实验。

模型缺点：

1. 简化了其中的一些操作，默认小车的运动为匀速，没有考虑到在实际的情况中，由于两车相遇会导致的加速与减速的问题。
2. 忽略小车在转弯的时候会有的一些问题。我们的这个模型适用于那些对于加速度比较快的情况，对于那种加速比较缓慢的，且转弯性能不好的情况，本模型的拟合程度会比较差。

8.2 模型的改进与推广

模型改进：

模型可以进一步考虑到复杂情况下的 AGV 调度问题，还应当考虑小车的加速度和减速度对于最后结果的影响。此外，还应该考虑将强化学习、优化算法用于防止 AGV 的死锁当中。

模型推广：

本模型不仅可以运用于 AGV 的调度问题，还可以运用于其他的调度问题，比如外卖派送、警力分配等等。可以运用蚁群算法与遗传算法的结合来计算调度的最优指派问题，从而能够提升总体的工作效率。

9. 参考文献

[1]肖海宁,楼佩煌,武星,翟晶晶,胡亚.多载量自动导引车系统防死锁任务调度方[J/OL].计算机集成制造系统:1-28[2022-04-15].
[2]王晓军,王博,晋民杰,杨春霞,白新利.改进多种群遗传算法的 AutoStore 系统多 AGV 调度优化[J].工业工程,2021,24(04):112-118+167.
[3]卜人杰,朱瑾.基于多特征与改进卷积神经网络的 AGV 实时调度研究[J].制造业自动化,2021,43(08):58-63.
[4]余娜娜,李铁克,王柏琳.自动化分拣仓库中多 AGV 在线协同调度算法[J/OL].计算机集成制造系统:1-23[2022-04-15].
[5]李天童,宁平凡,牛萍娟. 基于改进遗传算法的工厂 AGV 安全路径规划[J]. 组合机床与自动化加工技术,2020(3):160-163. DOI:10.13462/j.cnki.mmtamt.2020.03.038.
[6]陶秋云. 基于改进粒子群算法的 AGV 调度问题研究[D].聊城大学,2021.
[7]郭鹏,汪世杰,周士祺,史海超.基于并行搜索遗传算法的 AGV 自适应集群调度[J/OL].华中科技大学学报(自然科学版):1-7[2022-04-15].

附录

附录 1
功能：仿真实验程序
程序共计 590 余行，放入影响美观，因此放入支撑材料

附录

附录 2
功能：基于“模糊地带”的分类方法
程序共计 200 余行，放入影响美观，因此放入支撑材料

附录 3
功能：画图

```
clc
clear
close all

%% 构建颜色 MAP 图
cmap = [1 1 1; ...            % 1-白色-空地
        0 1 0; ...            % 2-黑色-静态障碍
        0 0 0; ...            % 3-红色-动态障碍
        1 1 0;...             % 4-黄色-起始点
        1 0 1;...             % 5-品红-目标点
        1 0 0; ...            % 6-绿色-到目标点的规划路径
        0 1 1;
        0 0 1;];                % 7-青色-动态规划的路径
% 构建颜色 MAP 图
colormap(cmap);

%% 构建栅格地图场景
% 栅格界面大小:行数和列数
rows = 22;
cols = 32;

% 定义栅格地图全域，并初始化空白区域
field = [
1 7 1 1 1 1 1 5 5 5 1 1 1 1 1 1 1 1 1 5 5 5 1 1 1 1 1 1 1 1 1 ;
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 ;
1 1 1 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ;
6 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ;
```

```
6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ;
1 1 1 1 1 2 2 2 2 2 3 8 2 2 2 1 1 2 2 2 2 2 2 2 2 3 3 1 1 1 1 1 ;
1 1 1 1 1 2 2 2 2 2 4 2 2 2 2 1 1 2 2 2 2 2 2 2 2 4 4 8 1 1 1 5 ;
6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 8 1 1 1 8 1 1 5 ;
1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 1 1 2 2 8 2 2 2 2 2 2 2 1 1 1 1 5 ;
1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 ;
4 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 8 1 1 1 1 ;
6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 8 1 1 1 1 1 ;
1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 ;
1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 ;
1 1 1 1 1 1 1 8 1 8 1 1 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 1 1 1 5 ;
1 1 1 1 1 2 2 2 2 2 4 2 2 2 2 1 1 2 2 2 2 2 2 2 2 4 4 1 1 1 1 5 ;
1 1 1 1 1 2 2 2 2 2 3 2 2 2 2 1 1 2 2 2 2 2 2 2 2 3 3 1 1 1 1 5 ;
6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ;
6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ;
1 1 1 1 1 8 1 8 8 1 1 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ;
1 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 1 1 8 1 1 1 1 1 1 1 1 1 1 1 7 ;
1 1 1 1 1 1 1 5 5 5 1 1 1 1 1 1 1 1 1 5 5 5 1 1 1 1 1 1 1 1 1 1 ;
];


%% 画栅格图
image(1.5,1.5,field);
grid on;
set(gca,'gridline','-','gridcolor','k','linewidth',1,'GridAlpha',0.5);
set(gca,'xtick',1:cols,'ytick',1:rows);
axis image;
```