**摘单分货**

**开发文档**

**京创智汇（上海）物流科技有限公司**

**2020年12月**

修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **作者** | **版本** | **修改原因** | **主要修改内容** |
| 2020.12.30 | 江俊峰、鲁铖锴 | 1.0 | 新建 | 框架设计，内容添加 |

**目录**

[1、概述 4](#_Toc60232470)

[2、需求背景 4](#_Toc60232471)

[2.1业务规则 4](#_Toc60232472)

[3、系统功能 5](#_Toc60232473)

[3.1 系统流程 5](#_Toc60232474)

[3.1.1人工流程 5](#_Toc60232475)

[3.1.2系统流程： 5](#_Toc60232476)

[3.2功能描述 5](#_Toc60232477)

[4、系统设计 6](#_Toc60232478)

[4.1总体架构 6](#_Toc60232479)

[4.2业务模型设计 7](#_Toc60232480)

[4.3业务逻辑设计 11](#_Toc60232481)

[4.4外部接口 22](#_Toc60232482)

[4.5代码管理 25](#_Toc60232483)

[附录 25](#_Toc60232484)

[A工具描述 25](#_Toc60232485)

[B 基类描述 25](#_Toc60232486)

# 1、概述

运营人员根据实时可发库存/明日排产计划，手动编制货源资源和摘单计划。现将运营人员手动编制货物资源部分改为由模型组自动分好，摘单计划部分尽可能减少运营人员操作。

# 2、需求背景

当前大宗商品产业链中存在物流资源（例如，运输车辆）短缺、仓库库存过多、货物超期严重等现象，这些现象导致了物流的严重阻塞以及物流成本的急剧增长。在物流供需不平衡的场景下，物资如何分配到车队、车辆是平衡物流资源的重要环节。分货策略不仅仅是决定单辆车辆的运载情况的配载策略，而且是对于长期整体物流效率与成本的把控。好的分货策略能够在车载重量、车辆数量限制的情况下，充分利用资源提高发运效率、降低库存。合理的分货策略能够在有限运输资源的条件下降低物流成本、提高服务质量，对整体物流带来经济效益的提升。

## 2.1业务规则

1、甩货最少：在保证处理优先发运的前提下进行合理搭配，做到同流向甩货最少。

2、装卸方便：优先一装一卸，其次是两装一卸、一装两卸，特殊情况会出现两装两卸，属于清理甩货的方式，调度人员与司机达成一致。

3、车次载重最大化：为保证车次不亏吨，尽可能配到满载上限。

# 3、系统功能

## 3.1 系统流程

### 3.1.1人工流程



图 1 人工流程图

### 3.1.2系统流程：



图 2 系统流程图

## 3.2功能描述

1、数据处理：获取库存json数据，对品名、重量等进行预处理；货物切分，货物筛选。

2、货物配载：将符合标载的数据直接生成车次；考虑品种搭配、重量上限进行组合。

3、生成车次，合并车次生成摘单计划

功能流程图如下说明。

库存json数据

品名的处理

货物切分

数据处理

品种搭配限制

货物配载

货物重量限制

生成车次，合并车次生成摘单计划

图 3 功能流程图

# 4、系统设计

## 4.1总体架构

该项目的总体架构如下图所示。

APP应用包

routes

main业务程序包

entity

service

dao

DataSet

返回结果，存储日志

摘单分货项目

config.py：项目配置信息

manage.py：项目入口，启动文件

util工具包

document文档包

rule

图 4 app应用解析

util工具类包：包含各种计算工具、动态规划算法、连接池、基类等。

main业务程序包：业务程序包内按不同的厂区业务划分成不同的业务包，如steel\_factory钢厂业务包。其内部结构是一致的，均含有entity业务实体包（存放数据对象类，如订单、车次等）、dao数据连接包（存放数据库访问逻辑）、service业务逻辑包（存放业务处理逻辑，如取库存预处理与分货等）、task定时任务包、rule规则包（存放重量规则、品类规则、拼货规则等，一类规则对应一个文件）、test测试包（存放测试代码。static包指静态数据，比如测试时读入、输出的excel表）、routes路由包（存放对外输入输出接口，负责把外部数据进行相应的提取操作并转化成项目里的对象数据，项目处理逻辑完结后再由该包负责对外输出）。后续的开发如果涉及到不同的厂，业务逻辑大体不变的情况下，编写不同的rule规则即可，别的类一般不需要修改。

document文档包：包含项目的文档资料

config.py项目配置信息：包含项目运行所需要的程序参数，如数据库连接的端口信息等。

model\_config.py 模型配置信息：包含模型处理时会用到的参数，比如重量上下限，可合并的品类等参数

manage.py 项目启动入口

## 4.2业务模型设计

该项目中使用的业务实体类如下类图所示。



图 5 业务实体类图

该项目中的数据结构设计与现实业务对象映射关系如下图，所有数据结构都存储在entity业务实体包中。



图 6 类与业务实体映射关系

LoadTask类对应实体业务中的车次，两者为一对一关系，一个LoadTask对象对应一个车次实体。业务实体库存在数据库中以发货通知单为元数据（单位）进行存储，Stock类对应业务实体中的库存，与发货通知单是多对一的关系，一个或多个Stock对象对应一条发货通知单记录。

（1）LoadTask类

继承BaseEntity类（详见附录）

车次类，一个对象表示一个车次，一个车次绑定一辆车，只能存放一个发货通知单中的内容

成员变量

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 含义 |
| load\_task\_id | string | 所属车次号 |
| priority | int | 优先级， |
| priority\_grade | string | 优先级对应的ABCD等级 |
| load\_task\_type | string | 装卸类型 |
| total\_weight | float | 总重量，当前车次的货物清单总重量 |
| notice\_num | string | 发货通知单号 |
| oritem\_num | string | 订单项次号， |
| weight | float | 发货通知单重量 |
| count | int | 发货通知单件数 |
| city | string | 城市 |
| end\_point | string | 区县 |
| big\_commodity | string | 大品种，货物品种的大分类 |
| commodity | string | 品种，货物品种的小分类 |
| standard | string | 规格 |
| sgsign | string | 材质 |
| outstock\_code | string | 出库仓库，取货的仓库 |
| instock\_code | string | 入库仓库，若货物运输的目的地需要其他仓库作为中转站，则存在入库仓库，表示该中转站 |
| receive\_address | string | 收货地址，货物运输目的地的详细地址 |
| price\_per\_ton | float | 当前车次的吨单价 |
| total\_price | flaot | 当前车次的总价 |
| latest\_order\_time | string | 最新挂单时间，该发货通知单中的货物生产结束进入仓库的时间 |
| remark | string | 注释 |

（2）Stock库存类

继承BaseEntity类

一个Stock对象对应一个库存子项（发货通知单子项），多个库存子项构成一个完整的库存数据（一条完整发货通知单记录），关系结构如下图所示。



图7 Stock拆分结构示意图

成员变量

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 含义 |
| Stock\_id | string | 库存对象编号 |
| Parent\_stock\_id | string | 父ID |
| Delivery | string | 发货通知单号 |
| Order | string | 订单号 |
| Priority | string | 发运的优先等级 |
| Consumer | string | 收货用户 |
| Small\_product\_name | string | 品种，货物品种的小分类 |
| Big\_product\_name | string | 大品种，货物品种的大分类 |
| Actual\_weight | int | 实际可发重量 |
| Actual\_number | int | 实际可发件数 |
| Piece\_weight | int | 件重 |
| mark | string | 牌号 |
| specs | string | 规格 |
| Pack\_form | string | 包装形式 |
| Warehouse\_out | string | 出库仓库，取货的仓库 |
| Warehouse\_in | string | 入库仓库，若货物运输的目的地需要其他仓库作为中转站，则存在入库仓库，表示该中转站 |
| Province | string | 省份，货物目的地的省份 |
| City | string | 城市，货物目的地的地级市 |
| Actual\_end\_point | string | 区县（存在部分数据为入库仓库） |
| End\_point | string | 区县 |
| Address | string | 卸货地址，详细地址 |
| Standard\_address | string | 合并卸货地址 |
| Delivery\_date | string | 合同约定交货日期 |
| Latest\_order\_time | string | 最新挂单时间，该发货通知单中的货物生产结束进入仓库的时间 |
| Logistics | string | 物流公司类型 |

（3）PickTask类

继承BaseEntity类（详见附录）

摘单类，一个对象表示一条摘单记录，一条摘单记录可以表示多个相同类型的车次信息。

成员变量

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 含义 |
| pick\_id | string | 摘单记录号 |
| source\_name | string | 货源名称 |
| total\_weight | number | 总重量 |
| truck\_num | number | 总车次数 |
| province | string | 省份 |
| city | string | 城市 |
| end\_point | string | 区县 |
| big\_commodity | string | 大品名 |
| commodity | string | 品名 |
| remark | string | 备注 |
| group\_flag | string | 跨厂区的标记：  1 跨厂区，0 不跨厂区 |
| template\_no | string | 模板号 |
| items | list | 摘单子类 |
| deliware\_district | list | 装点库区列表 |
| commodity\_count | dict | 各品种的件数 |
| commodity\_weight | dict | 各品种的重量 |
| bz\_rdx | string | 滨州市热镀锌钢卷的标记 |
| hot\_j | string | 热卷的标记 |

## 4.3业务逻辑设计

业务工程实现的逻辑结构如下图所示，所有工程逻辑都存储在service业务逻辑包和rules分货规则包中。

dispatch

json数据

get\_pick\_stock

库存 数据

库存预处理

dispatch\_filter

货物配载

返回结果

图 8 工程实现逻辑结构

业务流程如下图所示。

传入库存数据

库存预处理

1. 库存中不同客户名称，相同客户的处理
2. 品名的预处理
3. 筛除不需要模型分货的货物
4. 可发小于待发，待发在标载范围内，不参与配载
5. 如果某个区县的货物总重量<=40吨，在货物可拼的情况下，直接一车打包带走，不考虑重量限制
6. 如果某区县某客户某品种的总重量>200吨，则以最大重量上限来切分货物，不需要考虑尾货；否则以尾货最少为目标来动态切分货物
7. 将库存按照厂区分3类，每个厂区又按照卷类和非卷类分2类，以及不在所配置仓库中的货物
8. 老区的卷类先自己拼：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
9. 西区、岚北港件重24-31t的卷类先内部组合：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
10. 老区卷类配西区、岚北的卷类：两装一卸(非同区)、两装两卸(非同区)
11. 西区、岚北港的卷类内部自己组合：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
12. 将3个厂区的卷类与非卷类组合在一起走正常流程拼货：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
13. 不在当前仓库范围内的货物，则仓库内部自己组合(一装一卸，一装两卸)
14. 处理24吨及以上的卷(件重为1且重量大于24吨的卷在无货可拼的情况下也生成车次)

货物配载

生成摘单计划

1. 将相同属性的车次记录合并生成摘单计划
2. 合并时考虑的属性：城市、区县、厂区、是否必须铁架子（热卷）、品种（单品种货物：卷类看品种+件数，非卷类只看品种；多品种货物：看品种+件数）
3. 生成备注信息（品名+件数+重量+厂区）
4. 特殊备注：必须铁架子、木托窄带、窄带卷、热镀锌

返回结果

图 9 业务流程图

以下给出各模块详细描述。

**（1）pick\_goods\_dispatch\_service模块**

摘单分货主入口

* dispatch()方法

输入：json格式的库存数据

功能与流程：

1. 获取经过预处理的、切分完的、按厂区以及卷类与非卷类分类的库存列表
2. 调用dispatch\_filter方法进行配载，得到配载车次列表、无法配载的剩余尾货列表
3. 调用create\_pick\_task方法将车次合并生成摘单记录
4. 将生成的摘单记录转换为指定格式
5. 保存生成的摘单记录、尾货记录、被扣除的已调度未开单的摘单记录到数据库

返回：指定格式的摘单记录。

* data\_format()方法

将生成的摘单记录和尾货数据按指定的格式进行转换。

* save\_result()方法

保存生成的摘单记录和尾货数据。

* save\_deduct\_pick()方法

保存被扣除的已调度未开单的摘单记录。

* weight\_delete()方法

已调度未开单的扣除操作。

功能与流程：

1. 根据已调度未开单表中的数据，先按照区县、品名、件数精准查询相应的摘单记录并扣除
2. 若没有查询到则按照区县、品名询相应的摘单记录并扣除

**（2）pick\_stock\_service模块**

摘单库存服务

* get\_stock\_id()方法

根据库存信息生成每条库存的唯一id

* get\_pick\_stock()方法

库存预处理，根据车辆目的地和可运货物返回库存列表

输入:

json格式的列表

功能与流程：

1）对获取的库存数据进行数据预处理，其中包括：

1.根据公式计算实际可发重量（\*1000），实际可发件数。过滤掉实际可发件数小于等于0的库存数据。

2.根据公式计算件重：实际可发重量/实际可发件数

3.将待发重量转换成kg（\*1000）

4.过滤掉实际可发重量/件数小于等于0和最新挂单时间为空（及当前没有货物可分）的数据。

5.对品种重命名，改成例如“老区-卷板”的格式。

6.将终点统一赋值到实际终点，方便后续处理联运。

2）初始化对象列表，转换成 List[Stock] 格式数据。

3）调用pick\_early\_dispatch\_filter模块的early\_dispatch\_filter()方法，对数据进一步处理。

4）调用split\_pick\_stock()方法，对数据做进一步尾货的筛选。

5）对货物列表做关于卷类、装点的筛选，规则如下：

1.筛选出西区的'新产品-卷板', '新产品-白卷'

2.筛选出西区中除'新产品-卷板', '新产品-白卷'外的其他货物

3.筛选出老区的'老区-卷板'

4.筛选出老区中除'老区-卷板'外的其他货物

5.筛选出岚北港的卷类

6.筛选出岚北港中除卷类外的其他货物

7.筛选出不在ModelConfig.RG\_WAREHOUSE\_LIST仓库中的货物

* split\_pick\_stock()方法

货物切分方法，将件重大于重量上限的货物保存到尾货

输入:

init\_stock\_list：以Stock对象为元数据的列表

功能与流程：

1）对库存进行平衡分组，保证组数不变的情况下，每组件数尽可能多。

2）处理平衡分组后的尾货，保证每组不超载的情况下，尾货尽可能少。

3）将单件件重超限的货物，加入尾货。

4）将可装件数大于实际可发件数的货物加入待发库存列表。

5）对剩下的库存进行拆分

**（3）pick\_early\_dispatch\_filter模块**

库存进一步处理筛选服务

* early\_dispatch\_filter()方法

早期处理，处理大于200的切分和流向剩余量少的情况

输入:

init\_stock\_list：以Stock对象为元数据的列表

功能与流程：

1）移除等货的数据，规则如下：

1.筛选出滨州市不需要分配的货物

2.如果可发小于待发，并且待发在标载范围内，就不参与配载

2）按区县分组，存在两种情况：

1.处理流向剩余量少的情况（不包括济南市），规则如下：

* 移除未完全生产完的货物，求新的总重
* 去重：考虑品种搭配、客户数量、仓库数量,包括两装两卸

2.处理整个区县大于200吨的货物，规则如下：

* 按客户和品名分组
* 调用early\_split\_pick\_stock()方法，寻找并切分某客户某品种总量大于200吨的情况。
* judge\_if\_across\_factory()方法

判断deliware\_house\_set中的仓库是否跨厂区，跨厂区返回True，否则返回False

* early\_split\_pick\_stock()方法

早期的切分方法，拿最大重量切

输入:

consumer\_big\_commodity\_list 某客户某品种，以Stock对象为元数据的列表

early\_stock\_list 以Stock对象为元数据的列表

early\_wait\_list 尾货，以Stock对象为元数据的列表

功能与流程：

1）去除 件重大于35500的货物，保存到尾货

2）如果可装的件数大于实际可发件数，不用拆分，直接添加到stock\_list列表中

3）如果可装的件数小于等于实际可发件数，则拆分后添加到stock\_list列表中

**（4）pick\_goods\_dispatch\_filter模块**

* dispatch\_filter()方法

输入：前面分类好的扣除列表

功能与流程：

1. 老区的卷类先自己拼：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
2. 西区、岚北港的卷类 件重24-31t的先内部组合：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
3. 老区配西区、岚北：两装一卸(非同区)、两装两卸(非同区)
4. 西区、岚北港的卷类内部自己组合：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
5. 将卷类与非卷类组合在一起走正常流程拼货：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)
6. 不在当前仓库范围内的货物，则仓库内部自己组合
7. 处理24吨及以上的卷

返回：车次列表，剩余的尾货列表

* stock\_compose()方法

标准货物处理流程

功能与流程：

1. 将库存分为标载huge\_stock\_list和尾货tail\_stock\_list，标载直接生成车次
2. 将尾货按照区县客户分组
3. 将同区县同客户的尾货组合配载，得到配载列表、无法配载的尾货列表，一装一卸、两装一卸(同区)
4. 将剩余的尾货按照区县仓库分组
5. 将同区县同仓库的尾货组合配载，一装两卸
6. 最后进行两装两卸(同区)

返回：返回最后剩余的尾货列表

**（5）pick\_compose\_j模块**

* compose\_1\_to\_1()方法

一装一卸

* compose\_2\_to\_1()方法

两装一卸：如果传入的one\_stock\_list和other\_stock\_list是同区的则是同区两装一卸，否则是非同区两装一卸

* compose\_1\_to\_2()方法

一装一卸

* compose\_2\_to\_1()方法

两装一卸

* compose\_2\_to\_2()方法

两装两卸：如果传入的one\_stock\_list和other\_stock\_list是同区的则是同区两装两卸，否则是非同区两装两卸

* compose\_method()方法

一装一卸，两装一卸，一装两卸，两装两卸(其中两装一卸、两装两卸是否同区，取决于one\_stock\_list与other\_stock\_list是否是同区的)

输入：2个库存列表，是否是同区的标记（默认同区）

功能与流程：

分别调用上面的一装一卸，两装一卸，一装两卸，两装两卸方法进行组合

返回：车次列表、传入的两个库存组合配载后剩余的库存

* inner\_compose()方法

将当前库存中件数为1重量为24-31t的库存先和自己内部的货物组合：一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)

输入： 库存列表

功能与流程：

1. 将库存分为件数为1重量为24-31t的和其他的
2. 调用compose\_method方法进行一装一卸，两装一卸(同区)，一装两卸，两装两卸(同区)

返回：车次列表、剩余的库存

**（6）pick\_group\_by\_huge\_and\_tail\_rule模块**

* stock\_filter()方法

输入： 库存列表

功能与流程：

将库存分为标载和尾货

返回： 标载库存列表、尾货库存列表

**（7）pick\_split\_group\_rule模块**

* split\_group()方法

按指定的属性分组

返回：分组后的字典

**（8）pick\_compose\_in\_same\_district\_and\_consumer模块**

* tail\_compose()方法

将同区县同客户的尾货组合配载，一装一卸、两装一卸(同区)

输入： 分组的库存字典

功能与流程：

1. 调用compose\_in\_same\_deliware\_house进行一装一卸
2. 调用compose\_not\_in\_same\_deliware\_house进行同区两装一卸

返回： 车次列表，剩余的尾货列表

* compose\_in\_same\_deliware\_house()方法

一装一卸

* compose\_not\_in\_same\_deliware\_house()方法

同区两装一卸

* goods\_compose()方法

输入： 可组合的列表filter\_list, 需要组合的对象temp\_stock, 最大可组合重量surplus\_weight, 组合的最小重量new\_min\_weight, 装卸类型load\_task\_type

功能与流程：

拿temp\_stock与filter\_list中的货物进行组合，如果一单在达标重量之上并且无货可拼的情况也生成车次

返回： 生成的车次，被组合的列表

**（9）pick\_compose\_in\_same\_district\_and\_deliware\_house模块**

同区县同仓库的尾货组合配载服务

* surplus\_compose()方法

将同区县、同仓库，不同客户的尾货组合配载（一装两卸）

输入:

stock\_list 以Stock对象为元数据的列表

mark 标记，判断是否为滨州市的货物，用于get\_weight()方法

功能与流程：

1）按重量降序排序

2）调用get\_optimal\_group()方法，配载货物

3）若配载成功：

1.调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次

2.调用delete\_the\_stock\_be\_composed()方法，删除对应库存

4）若配载不成功，拆分货物后继续尝试配载：

1.调用split\_rule模块的split()方法，将货物拆分成单件

2.调用get\_optimal\_group()方法，配载货物

3.配载成功：

* 调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次
* 调用delete\_the\_stock\_be\_composed()方法，删除对应库存

4.配载失败，调入尾货

5）若拆分货物后配载也失败，调入尾货

**（10）pick\_compose\_2\_to\_2模块**

两装两卸服务

* same\_deliware\_compose()方法

两装(同区仓库)两卸

输入:

stock\_list 以Stock对象为元数据的列表

mark 标记，判断是否为滨州市的货物，用于get\_weight()方法

功能与流程：

1）筛选出两装的待匹配的库存列表

2）按重量降序排序

3）调用get\_optimal\_group()方法，配载货物

4）若配载成功：

1.调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次

2.调用delete\_the\_stock\_be\_composed()方法，删除对应库存

5）若配载不成功，拆分货物后继续尝试配载：

1.调用split\_rule模块的split()方法，将货物拆分成单件

2.调用get\_optimal\_group()方法，配载货物

3.配载成功：

* 调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次
* 调用delete\_the\_stock\_be\_composed()方法，删除对应库存

4.配载失败，调入尾货

6）若拆分货物后配载也失败，调入尾货

* not\_same\_deliware\_compose()方法

两装(非同区仓库)两卸

输入:

west\_stock\_list 以Stock对象为元数据的列表

old\_stock\_list 以Stock对象为元数据的列表

lbg\_stock\_list 以Stock对象为元数据的列表

功能与流程：

1）西区货物处理：

1.筛选出两装的待匹配的库存列表

* 筛选出符合条件的老区库存
* 筛选出符合条件的岚北港库存

2.按重量降序排序

3.调用get\_optimal\_group()方法，配载货物

4.若配载成功：

* 调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次
* 调用delete\_the\_stock\_be\_composed()方法，删除对应库存

5.若配载不成功，拆分货物后继续尝试配载：

* 调用split\_rule模块的split()方法，将货物拆分成单件
* 调用get\_optimal\_group()方法，配载货物
* 配载成功：调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次；调用delete\_the\_stock\_be\_composed()方法，删除对应库存
* 配载失败，调入尾货

2）老区货物处理：

1.筛选出两装的待匹配的库存列表

* 筛选出符合条件的岚北港库存

2.按重量降序排序

3.调用get\_optimal\_group()方法，配载货物

4.若配载成功：

* 调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次
* 调用delete\_the\_stock\_be\_composed()方法，删除对应库存

5.若配载不成功，拆分货物后继续尝试配载：

* 调用split\_rule模块的split()方法，将货物拆分成单件
* 调用get\_optimal\_group()方法，配载货物
* 配载成功：调用pcik\_create\_load\_task\_rule模块的create\_load\_task()方法，生成具体车次；调用delete\_the\_stock\_be\_composed()方法，删除对应库存
* 配载失败，调入尾货

3）如果岚北港还有货物，则调入尾货

**（11）last\_j\_handler\_rule模块**

* j\_handler()方法

处理24吨及以上的卷：卷类1件，并且重量大于24t可发走

**（12）create\_pick\_task\_rule模块**

* create\_pick\_task()方法

输入： 车次列表

功能与流程：

1. 调用deal\_pick\_task方法进行预处理，将load\_task\_list转换为pick\_list
2. 调用split\_group方法将相同的记录合并
3. 调用merge\_and\_remark方法生成备注， 并且生成摘单计划

返回： 摘单记录列表

* deal\_pick\_task()方法

输入： 车次列表

功能与流程：

1. 找出货物所在的厂区
2. 如果是济南市Z1、Z2库24h内的热卷，则标记：必须铁架子
3. 调用get\_special\_remark生成特别的备注信息：(木托窄带)、(窄带卷)、(热镀锌)
4. 求出各品名的总件数、总重量
5. 拼货时，跨厂区和非跨厂区的处理
6. 生成模板号

返回： 返回处理后的摘单记录列表

* split\_group()方法

输入： 摘单记录列表

功能与流程：

将摘单记录按照条件分组

分组条件：①城市；②区县；③厂区；④是否是热卷；⑤品种（单品种货物，分组时非卷类只看品种，不看件数，卷类既看品种，又看件数；多品种货物，分组时看品种+件数）

返回： 分组后的字典

* merge\_and\_remark()方法

输入： 摘单记录分组后的字典

功能与流程：

将相同的车次记录合并，并且生成备注， 生成摘单计划

返回： 摘单计划列表

**（13）goods\_filter\_rule模块**

* goods\_filter()方法

输入： 库存列表，重量限制

功能与流程：

通过0-1背包算法求解最优化问题，调用pulp\_pack()方法进行求解

返回：被组合的列表，组合后的最优值

**（14）pulp\_solve模块**

* pulp\_pack()方法

调用pulp包解决最优化问题，使用线性规划的方法求解最优解，优化目标与限制条件如下：

其中w表示单件货物的重量，x表示该货物的选择情况，，c为阈值表示装载上限。优化目标为最大化选择的货物总重量，限制条件为选择的货物总重量小于阈值。

因此方法结果求解，得到实数列x。最终需要得到货物是否被选择（0表示不选择，1表示选择）的结果列表，因此程序使用0.5为阈值将x进行01离散化处理。

**（15）pick\_compose\_public\_method模块**

摘单配载共用方法服务

* get\_weight()方法

获取库存重量

滨州大于等于29吨小于31吨的卷，优先配货，最大载重调到39（只限滨州卷配货）；剩余流向载重照常。

* get\_optimal\_group()方法

获取最优组别

调用**goods\_filter()**方法（0-1背包），获取单背包的货物配载最优解（对型钢做特判）

* delete\_the\_stock\_be\_composed()方法

删除库存中已经被组合的货物

若扣减后实际件数为0，则删除库存

* merge\_result()方法

将load\_task中相同订单的子项合并

* merge\_split\_stock()方法

将stock\_list中parent\_stock\_id相同的订单合并为一条记录

## 4.4外部接口

（1）测试接口链接:[http://192.168.1.27:9238 /pickGoodsAllocation](http://192.168.1.44/#/system/goods-allocation/goods-allocation/billOfManage)

（2）功能说明：根据输入的库存json数据，按照分货配载规则分发到车次，将相同类型的车次合并为一条摘单记录，产生摘单列表

（3）接口说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求协议 | 请求方式 | 编码格式 | 返回格式 |
| HTTP | POST | UTF-8 | JSON |

（4）接口参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名字 | | 类型 | 说明 |
| data | | list |  |
|  | notice\_num | string | 发运通知单号 |
|  | consumer | string | 收货用户 |
|  | oritem\_num | string | 订单号 |
|  | deliware\_house | string | 出库仓库 |
|  | commodity\_name | string | 品名 |
|  | big\_commodity\_name | string | 大品名 |
|  | detail\_address | string | 详细地址 |
|  | city | string | 市 |
|  | waint\_fordel\_number | number | 待发件数 |
|  | waint\_fordel\_weight | number | 待发重量 |
|  | can\_send\_number | number | 可发件数 |
|  | can\_send\_weight | number | 可发重量 |
|  | dlv\_spot\_name\_end | string | 终点（区县） |
|  | pack\_number | number | 窄带捆包数 |
|  | need\_lading\_num | number | 需开单件数 |
|  | need\_lading\_wt | number | 需开单重量 |
|  | over\_flow\_wt | number | 需短溢重量 |
|  | latest\_order\_time | string | 最新挂单时间 |
|  | specs | string | 规格 |
|  | source\_number | string | 货源编号 |
|  | source\_name | string | 货源名称 |
|  | can\_split\_weight | number | 可分重量 |
|  | can\_split\_number | number | 可分数量 |
|  | pack | string | 包装形式 |

（5）返回结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名字 | |  | 类型 | 说明 |
| code | | | int | 状态码 |
| msg | | | string | 返回提示信息 |
| data | | |  | 结果数据 |
| pick\_list | sourceName |  | string | 货源名称 |
| totalWeight |  | number | 总重量 |
| truckNum |  | number | 总车次数 |
| province |  | string | 省份 |
| city |  | string | 城市 |
| endPoint |  | string | 区县 |
| bigCommodity |  | string | 大品名 |
| commodity |  | string | 品名 |
| remark |  | string | 备注 |
| groupFlag |  | boolean | 是否跨厂区 |
| templateNo |  | string | 模板号 |
| items | sourceName | string | 货源名称 |
| totalWeight | number | 总重量 |
| truckNum | number | 总车次数 |
| truckCount | number | 一车的件数 |
| province | string | 省份 |
| city | string | 城市 |
| endPoint | string | 区县 |
| bigCommodity | string | 大品名 |
| commodity | string | 品名 |
| remark | string | 备注 |
| tail\_list | sourceNumber |  | string | 货源名称 |
| noticeNum |  | string | 发货通知单 |
| oritemNum |  | string | 订单号 |
| deliwareHouse |  | string | 出库仓库 |
| sendNumber |  | number | 已发数量 |
| sendWeight |  | number | 已发重量 |

## 4.5代码管理

一、项目Git地址：<https://git.jczh56.com/models/gc-goods-allocation>

二、分支说明:

1.dev\_1.0为项目主发布分支，在该分支上构建开发分支

2.dev\_steel为钢厂开发分支

3.dev\_pipe为管厂开发分支

4.test为摘单分货开发分支

三、开发分支管理规范

1.开发新功能时选择钢厂或管厂分支拉到本地进行开发；

2.在开发分支进行功能开发和测试，同步提交相关文档；

3.测试完成后准备发布时将代码合并到dev\_1.0并打上对应的版本tag；

# 附录

## A工具描述

util包db\_pool.py数据库连接池

connection（pymsql中的数据库连接对象）统一存放在该工具模块中

## B 基类描述

BaseEntity类

位于app.main.entity.base\_entity.py，实体类的基类，子类继承可调用或重写以下通用方法，当前工程中类继承关系如下图。



包含方法as\_dict、get\_attr\_str、set\_attr

as\_dict将对象（成员变量）转换为dict数据类型返回

get\_attr\_str读取类成员变量，将其统一转换为string数据类型

set\_attr根据传入的对象设置成员变量，将传入的dict数据类型转换为成员变量