doi:10.3772/j.issn.1002-0470.2020.06.010

大批量订单整体分拣问题建模及其分布式并行方法①

赵云波② 李天舒 汪钰皓

(浙江工业大学信息工程学院 杭州 310000)

摘 要 对大批量订单中的货物进行高效整体分拣是影响大型电商整体物流效率的关键因素,但这一问题尚缺少严格模型和有效解决方法。本文通过对订单分拣系统的静态状态和动态演化进行数学刻画,并对订单分拣算法、系统限制条件和分拣效率等进行严格定义和区分,建立了大批量订单整体分拣问题的数学模型。在此基础上,利用智能货架思想提出了一种大批量订单的分布式并行整体分拣方法,有效解决了在订单整体分拣问题中的打包点数量瓶颈问题,解决了所提出方法的关键技术难点,并在数值仿真下证明了该方法的有效性。

关键词 大批量订单;整体分拣;模型化;分布式并行分拣

0 引 言

大型生产和销售需要对大批量订单进行拣取,典型场景可见于大型电商的仓库系统。在较短时间内仓库收到从电商平台产生的大量订单,仓库系统需按照每份订单中包含的商品种类和数量从仓库中拣选货物并打包,随后交付物流系统。这一订单分拣过程的准确性和快速性影响着整体电商流程的效率,已经引起如亚马逊、京东等各大电商的广泛注意,应用了各种尖端技术[1-2]。

常用的订单分拣方式将所有待处理订单中的所有货物归为统一的待拣货集合进行统一处理。系统从仓库货架上依照待拣货集合逐一拣选商品,然后运送至特定地点按照订单进行打包。订单分拣过程分为3个步骤,即货物拣选、货物运送和订单打包[3]。货物拣选大多手工进行,可通过优化物品和货架摆放[4-7]、划分批次等方式提升效率[8-11]。货物运送过程通常可以借由传送带或无人车自动化进行[12],传送带路线设置、无人车路径规划等是这一步骤效率提升所需考虑的问题[13-16]。订单打包的物理过程较为简单,但其执行依赖于订单中的货物是否可以准确快速到达,这是衡量整个分拣系统效率的标志。

现有订单分拣方式存在的主要问题是大批量订单"整体分拣"能力低。整体分拣是指对任意订单中的货物完整地分拣出来并打包,而不将订单分拆为子订单,每一子订单仅包含所考虑订单货物的真子集。整体分拣是电商系统的天然要求,因为电商的每一个订单都对应唯一一个运送目的地,将一个订单分拆为多个子订单将成倍增加后续物流包裹的数量,增加物流成本。但是,现有订单分拣方式本质上不适应这一整体分拣的要求,其核心原因是对订单打包点的限制。大多数方法的订单打包点都设置为较少的特定地点,比如使用传送带进行已拣选货物的运送,打包点应在传送带的出口(可设置多个出口);若使用无人车方式进行已拣选货物的运送,作为打包点的运送目的地通常也较为集中,容易导致拥堵和效率低下[17]。现有的对整体分拣的研究,未从根本上解决这一难题[18-19]。在订单分拣问题的研究中,大多关注于具体情境下的设置和算法,缺

① 国家自然科学基金(61673350),中组部青年千人计划和浙江省千人计划资助项目。

② 男,1981年生,博士,教授;研究方向:网络化控制系统,人工智能驱动控制,人机融合智能和系统生物学;联系人,E-mail:ybzhao@ieee.org

(收稿日期:2019-07-10)

少描述问题的统一数学框架,导致对这一问题研究不足。

面向上述的大批量订单整体分拣问题缺少严格模型和有效方法的现实,本文首先对该过程进行严格的数学模型化,进而提出一种基于智能货架的分布式并行解决方案。该模型定量描述了整体分拣的静态状态和动态演化,给出了整体分拣问题的明确定义。依托这一模型,提出了相应的解决方案,其核心思想是使用智能货架作为订单分拣点,将订单分拣过程从原先的单一分拣点改造为分布式并行分拣点,从本质上满足了整体分拣的需求。这一改造方式引出了若干技术上的困难,如无人车在不碰撞前提下的高效运行方式,订单的分配方案等。本文在预测控制和优化等框架下对问题进行了有效解决,仿真实验证明了所提出方法的有效性。

本文其他部分组织如下。第 1 节通过对订单分拣系统的静态状态、动态演化和系统目标等描述,给出了大批量订单整体分拣问题的数学模型。第 2 节描述所提出的分布式并行整体分拣方法,包括其基本思路、改进模型、关键技术问题和完整算法等方面。第 3 节利用仿真对比实验证明了所提出的分布式并行方法的有效性。第 4 节总结全文。

1 大批量订单整体分拣问题的模型化

考虑某一封闭仓库区域 H 内使用无人车进行货物运送的订单分拣系统。该系统具有如下特点:在任一时刻有大量的订单待处理;每一订单需将其包含的所有货物全部分拣出来并整体打包,不允许进行分拆。本节对这一订单整体分拣问题模型化,首先描述订单系统在任一固定时刻 t 的静态状态,然后刻画订单系统在时间 $[t, t+1)$ 内的动态演化过程,最后给出问题的明确定义。

本文中所用符号较多,故将本文所用符号及其含义列为表 1。

1.1 订单分拣系统在 t 时刻的静态状态描述

首先对订单分拣系统在 t 时刻的静态状态给出符号化描述。这包含货物和货架的状态、待处理订单情况和无人车的状态。

表 1 符号列表

符号	描述
N_s	仓库中货架个数
S_i	第 i 个货架
N_g	所有货物种类总数
G_g	标号为 g 的货物
I_g	货物 G_g 的处理状态
O_o	标识号为 o 的订单
$\mathbb{I}_o^g$	订单 O_o 中所有货物的标识号集合
$\mathbb{I}_o(t)$	t 时刻所有待处理订单的标识号集合
$N_o(t)$	t 时刻所有待处理订单的个数
$\Theta(t)$	t 时刻待处理订单集合
N_v	无人车数量
V_i	第 i 辆无人车
$V_i(t)$	无人车 V_i 在 t 时刻的状态,其中 $\boldsymbol{l}_i(t)$、$\boldsymbol{v}_i(t)$ 和 $\boldsymbol{a}_i(t)$ 分别为其位置、速度和加速度
$\mathbb{V}$	所有无人车的集合,$\mathbb{V}_I(t)$ 和 $\mathbb{V}_O(t)$ 分别表示 t 时刻空闲的和忙碌的无人车集合
$\lvert\cdot\rvert$	集合中元素的个数
$\Theta_C(t+1)$	在 $[t, t+1)$ 时间内处理完的订单集合
$\mathbb{G}_g$	需分配无人车进行分拣和运送的货物的集合
f_{gv}	将 $\mathbb{G}_g(t)$ 中的货物分配给空闲无人车 $\mathbb{V}_I(t)$ 的算法
$A(t)$	计算每一辆无人车在 $[t, t+1)$ 内的运行规则的算法
PRI_o	待处理订单 O_o 的处理优先级
$\boldsymbol{D}_g^s$	带运送货物 G_g 与空闲无人车的距离向量

货物和货架 设该仓储区域内中有 N_s 个货架,记第 i 个货架为 S_i,设货物种类总数为 N_g,每一货物以其标识号 g 为唯一识别,并均存放于某一货架 S_g 上。记标号为 g 的货物为 G_g,则 G_g 可表示为

$$G_g := \{g, S_g, I_g\} \tag{1}$$

其中,I_g 为货物 G_g 在当前时刻的处理状态,定义如下:

$$I_g := \begin{cases} 1 & \text{货物已分拣} \\ 0 & \text{货物已分拣中} \\ -1 & \text{货物尚未分拣} \end{cases} \tag{2}$$

注意其中 $I_g = 0$,即“货物在分拣中”的含义是该货物已经交由某无人车运送处理,但尚未取回。

待处理订单 每一个订单包含其唯一的订单标识号 o,该订单进行整体分拣处理的打包点 p_o,订单到达订单分拣系统的时刻 t_o,和它所包含的所有货

物,记该订单为 O_o,则:

$$O_o := \{o, p_o, t_o; G_g \mid g \in \mathbb{I}_o^g\} \tag{3}$$

其中,$\mathbb{I}_o^g$ 为订单 O_o 中所有货物的标识号集合。

记 $\mathbb{I}_o(t)$ 为 t 时刻所有待处理订单的标识号集合,其个数记为 $N_o(t) := |\mathbb{I}_o(t)|$。则 t 时刻待处理订单的集合可写为

$$\Theta(t) := \{O_o \mid o \in \mathbb{I}_o(t)\} \tag{4}$$

无人车 设该仓库区域内有 N_v 辆无人车用于货物运送。记第 i 辆无人车为 V_i,所有无人车的集合记为 $\mathbb{V}$,即:

$$\mathbb{V} := \{V_i \mid i = 1, \cdots, N_v\} \tag{5}$$

无人车接受货物运送的指令,从相应货架取得货物,并将其运送到目的地。在时刻 t 的无人车状态由其当前位置、运动情况、承担货物运输情况等信息描述。记无人车 V_i 在 t 时刻的状态为 $V_i(t)$,则:

$$V_i(t) := \{\boldsymbol{l}_i(t), \boldsymbol{v}_i(t), \boldsymbol{a}_i(t), o_i^v(t), g_i^v(t)\} \tag{6}$$

其中,$\boldsymbol{l}_i(t)$、$\boldsymbol{v}_i(t)$ 和 $\boldsymbol{a}_i(t)$ 分别为无人车 V_i 的位置、速度和加速度。在笛卡尔坐标系下(假设已经建有原点和 x,y 轴),有:

$$\boldsymbol{l}_i(t) = [l_i^x(t), l_i^y(t)] \tag{7}$$

$$\boldsymbol{v}_i(t) = [v_i^x(t), v_i^y(t)] \tag{8}$$

$$\boldsymbol{a}_i(t) = [a_i^x(t), a_i^y(t)] \tag{9}$$

其中,上标 x 和 y 分别表示相应向量在 x 轴和 y 轴的分量。

$o_i^v(t)$ 和 $g_i^v(t)$ 是无人车 V_i 运送的货物所属订单标识号和货物标识号。若当前无人车为空闲状态,则标记为 $o_i^v = g_i^v = 0$。注意货物所在货架和运送目的地可分别由式(1)和式(3)得到。

由以上定义可知,t 时刻空闲的无人车集合 $\mathbb{V}_I(t)$ 可计算如下:

$$\mathbb{V}_I(t) = \{V_i \mid o_i^v(t) = g_i^v(t) = 0 \quad i = 1, \cdots, N_v\} \tag{10}$$

从而,正在执行某项任务的无人车集合 $\mathbb{V}_o(t)$ 可计算如下:

$$\mathbb{V}_o(t) = \{V_i \mid o_i^v(t) \neq 0, g_i^v(t) \neq 0 \quad i = 1, \cdots, N_v\} \tag{11}$$

可知,在 t 时刻订单分拣系统的静态状态可由式(4)中的待处理订单集合 $\Theta(t)$ 和式(6)中的所有无人车的状态 $V_i(t)$, $i = 1, \cdots, N_v$ 完全描述。

1.2 订单分拣系统在 $[t, t+1)$ 时间间隔内的动态演化

本节定量描述订单分拣系统在 $[t, t+1)$ 时间间隔内的动态演化。包含新订单的产生过程、订单的处理过程和无人车的运动。注意,$[t, t+1)$ 时间间隔的大小主要由无人车的运动情况确定,在该时间间隔内,无人车的运动不会过大而影响算法设计,具体数值需由具体应用确定。

新订单的产生 考虑大型电商订单的复杂性,假定订单的到来是随机的。类似情形的随机分布一般建模为指数分布。指数分布假设在很小的时间间隔 Δt 内,一个新订单到来的概率是 $\lambda\Delta t$, 其中 λ 是指数分布的强度,即:

$$\Pr\{\text{一个新订单在}[t, t+\Delta t)\text{ 到达}\} = \lambda \tag{12}$$

由上式可知,$[t, t+1)$ 时间间隔内新到订单数的期望为 $\lambda/\Delta t$。

根据式(3),新到的标号为 N 的订单可记为

$$O_N = \{o_N, p_N, t; G_g \mid g \in \mathbb{I}_{o_N}^g\} \tag{13}$$

新订单 O_N 中的货物假设为随机从所有货物集合中取出。这一"随机性"由 2 个过程构成,即首先从 N_g 种不同货物种类中随机选择订单 O_N 中的货物种类多少,$\mathbb{I}_{o_N}^g$ 中元素的个数,记为 $|\mathbb{I}_{o_N}^g|$;然后就每一个货物种类确定其具体货物标识号。以上过程写为

$$\Pr\{|\mathbb{I}_{o_N}^g| = i\} = \kappa_i \quad i = 1, \cdots, N_s \tag{14}$$

$$\Pr\{S_g = S_j\} = \eta_j \quad j \in \mathbb{I}_{o_N}^g \tag{15}$$

在具体的订单系统中,可通过调节上述概率 κ_i 和 η_j 描述订单包含货物的数量变化和货物在货架的摆放情况。

同时也需注意,上述新订单产生过程是对真实订单分拣系统的模拟,是为了在数学模型中对该过程进行描述,有利于后续订单分拣方法的设计。而在实际的订单分拣系统运行中并不需要上述模拟过程,新订单是系统实际产生的。

订单的处理 在 $[t, t+1)$ 时间内,待处理订单中的货物 G_g 若被订单分拣系统处理,则其状态会相应发生变化。将货物的状态变化标记为 ΔI_g, 令:

$$\Delta I_g = \begin{cases} 1 & \text{在}[t, t+1)\text{内}\ I_g\ \text{有变化} \\ 0 & \text{在}[t, t+1)\text{内}\ I_g\ \text{无变化} \end{cases} \tag{16}$$

本文选择的时间间隔足够小,保证了在每一步货物的3种状态最多只能发生一步变化。因此,

$$I_g(t+1) = I_g(t) + \Delta I_g \tag{17}$$

令 c_o 为订单 O_o 中所有货物的处理状态的最小值,即:

$$c_o := \min_{I_g \in C_g \in O_o} I_g \tag{18}$$

注意订单 O_o 仍待处理的充要条件是该订单中仍有货物未分拣完,即:

$$O_o\ \text{处理状态}: \begin{cases} c_o = 1 & \text{订单已处理完} \\ c_o < 1 & \text{订单未处理完} \end{cases} \tag{19}$$

注意 $c_o < 1$ 包含了 $c_o = 0$ 和 $c_o = -1$ 两种可能。若 $c_o = 0$,则 O_o 内所有货物都已经安排分拣,但尚有货物还未分拣完成;若 $c_o = -1$,则 O_o 内尚有货物未安排分拣。不管是哪种情况,订单 O_o 都尚未完成分拣。

由此,在 $[t, t+1)$ 时间内处理完的订单集合为

$$\Theta_C(t+1) = \{O_o \in \Theta(t) \mid c_o(t+1) = 1\} \tag{20}$$

从而,$t+1$ 时刻的待处理订单集合可由下式计算:

$$\Theta(t+1) = \begin{cases} \Theta(t) \cup \Theta\{O_N\} \setminus \Theta_C(t+1) & \text{有新订单产生} \\ \Theta(t) \setminus \Theta_C(t+1) & \text{无新订单产生} \end{cases} \tag{21}$$

其中,符号"\"表示集合的差集。

无人车的运动 无人车 V_i 在 $[t, t+1)$ 内的运动取决于无人车的初始位置 $\boldsymbol{l}_i(t)$、初速度 $\boldsymbol{v}_i(t)$ 和此时间间隔内的加速度 $\boldsymbol{a}_i(t)$。由于 $[t, t+1)$ 很小,可假设在此时间内加速度为常值,即在 $[t, t+1)$ 内有:

$$\boldsymbol{a}_i(t) = \boldsymbol{a}_i^t \tag{22}$$

其中,$\boldsymbol{a}_i^t$ 为某一常值。

由式(6)可知,无人车在 $t+1$ 时刻的状态可写为

$$V_i(t+1) = \{\boldsymbol{l}_i(t+1), \boldsymbol{v}_i(t+1), \boldsymbol{a}_i(t+1), V_i^o(t+1), V_i^g(t+1)\} \tag{23}$$

其中,$\boldsymbol{a}_i(t+1)$ 在 $t+1$ 时刻由算法确定,且:

$$\boldsymbol{l}_i(t+1) = \boldsymbol{l}_i(t) + \boldsymbol{v}_i(t) + \frac{1}{2}\boldsymbol{a}_i^t \tag{24}$$

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + \boldsymbol{a}_i^t \tag{25}$$

若无人车 V_i 在 $[t, t+1)$ 内有货物运送状态改变,货物标识号 $g_i^v(t+1)$ 需按实际情况改变。

$$g_i^v(t+1) = \begin{cases} g' & g_i^v(t) = 0\ \text{且有新货物}\ g'\ \text{运送} \\ 0 & g_i^v(t) = 1\ \text{且货物}\ g\ \text{运送完成} \end{cases} \tag{26}$$

订单标识号 $o_i^v(t+1)$ 可类似得到,不再赘述。

1.3 大批量订单整体分拣问题

综合1.1和1.2两小节给出的订单分拣系统运行的基本描述。希望设计算法尽可能提升系统的运行效率。在此之前,需要给出算法及其限制条件和运行效率的定量描述。

算法的定义 所设计的算法应处理2方面的工作:一方面是将待处理订单中的货物分配给无人车进行分拣和运送;另一方面是协调指挥无人车的运动。

注意到 t 时刻所有尚需分配无人车进行分拣和运送的货物的集合为

$$\mathbb{G}_g(t) = \{G_g \mid G_g \in O_o \in \Theta(t), I_g(t) = -1\} \tag{27}$$

首先,算法将 $\mathbb{G}_g(t)$ 中的货物分配给空闲无人车 $\mathbb{V}_I(t)$ 进行运输,即设计如下函数:

$$f_{gv}: \mathbb{G}_g(t) \to \mathbb{V}_I(t) \tag{28}$$

而假设函数 f_{gv} 与时刻 t 无关。这是因为,上述映射的确定仅决定当前的待处理货物集合 $\mathbb{G}_g(t)$ 和空闲无人车集合 $\mathbb{V}_I(t)$,而与时刻 t 无明显依赖关系。

另一方面,算法也需要给出每一辆无人车在 $[t, t+1)$ 内的运行规则。在已知 t 时刻无人车 V_i 的动力学状态前提下(即 $\boldsymbol{l}_i(t)$ 和 $\boldsymbol{v}_i(t)$),等价于给出在此时间内每一辆运行中无人车的加速度 $\boldsymbol{a}_i(t)$,$V_i \in \mathbb{V}_O(t)$。从而,所设计的算法具有如下形式:

$$A(t) := \{f_{gv}; \boldsymbol{a}_i^t, V_i \in \mathbb{V}_O(t)\} \tag{29}$$

限制条件 在本文中,限制条件主要指无人车的物理性质及其动力学特性。

无人车所运行的范围、速度和加速度都有其限制,总结如下:

$$\forall t,\ \forall i=1,\cdots,N_v\begin{cases}\boldsymbol{l}_i(t)\in H\\ 0\leqslant \boldsymbol{v}_i(t)\leqslant \boldsymbol{v}_{\max}\\ 0\leqslant \boldsymbol{a}_i(t)\leqslant \boldsymbol{a}_{\max}\end{cases}\tag{30}$$

其中，$\boldsymbol{v}_{\max}$ 和 $\boldsymbol{a}_{\max}$ 分别是无人车的速度和加速度上界。本文假定所有无人车都有相同的动力学特性，因此 $\boldsymbol{v}_{\max}$ 和 $\boldsymbol{a}_{\max}$ 对所有无人车都相同。这在大多数实际系统中是成立的。

无人车在运行过程中不可相互碰撞。定义 2 辆无人车 V_i，V_j 间的欧氏距离为

$$d_{ij}^{v}(t)=d(V_i(t),V_j(t))\tag{31}$$

$$d_{ij}^{v}(t)=\sqrt{(\boldsymbol{l}_i^x(t)-\boldsymbol{l}_j^x(t))^2+\boldsymbol{l}_i^y(t)-\boldsymbol{l}_j^y(t))^2}\tag{32}$$

其中，$d(\cdot,\cdot)$ 表示 2 点之间的欧氏距离。

任意 2 辆无人车的距离都不可超过某一事先预定的距离 d_v，即：

$$d_{ij}^{v}(t)>d_v,\ \forall t,\ \forall i,j=1,\cdots,N_v\tag{33}$$

注意到，由于订单是大批量的，其含有的货物量很大，同时$\mathbb{G}_g(t)$无人车的数量 N_v 也较大。因此，设计函数$f_{gv}(t)$较为困难；进一步地，在有限的空间内如何调度所有无人车尽量高速运行而不碰撞也变得异常复杂。

分拣效率的定义 订单分拣系统的运行效率可由在统计时域$[t,\ t+N_t)$内处理订单或货物的多少来衡量，其中 N_t 为某正整数，其值的大小表示了统计时域的大小。注意到使用货物或订单的完成量描述分拣效率并不等价，后者更多地与分拣的整体性相关。因此，本文使用在该统计时域内订单分拣系统所完成的订单分拣数量来衡量分拣效率 $J_{N_t}(t)$，由式(20)的定义，有：

$$J_{N_t}(t):=\sum_{\tau=t}^{t+N_t-1}|\Theta_C(t+1)|\tag{34}$$

综上所述，大批量订单整体分拣问题定义如下。

问题 1(大批量订单整体分拣) 给定封闭仓库区域 H，式(1)中定义的摆放于 N_s 个货架上的 N_g 种货物，式(5)～(11)中定义的无人车集合$\mathbb{V}$及其状态和式(22)～(26)描述的无人车运动方式，式(12)～(15)中描述的新订单产生方式和式(16)～(21)中描述的订单处理过程。在限制条件式(30)～(33)下，设计式(29)中定义的算法$A(t)$以优化在式(34)中定义分拣效率 $J_{N_t}(t)$。

2 大批量订单的分布式并行分拣

面向第 1 节定义的大批量订单整体分拣问题，本节提出一种基于智能货架的分布式并行解决方法。首先描述该方法的基本思路和硬件要求，进而给出其模型化描述，然后解决其中的关键技术问题，最后给出详细的算法描述。

2.1 分布式并行整体分拣策略及硬件要求

注意到大多数现有整体分拣系统的效率瓶颈在于打包处的稀缺性，本文提出如图 1 所示的一种新型大批量订单的分布式并行整体分拣模型。该模型的核心要点是允许所有的货架都作为潜在的订单打包处，从而使得订单的打包成为分布式并行的。

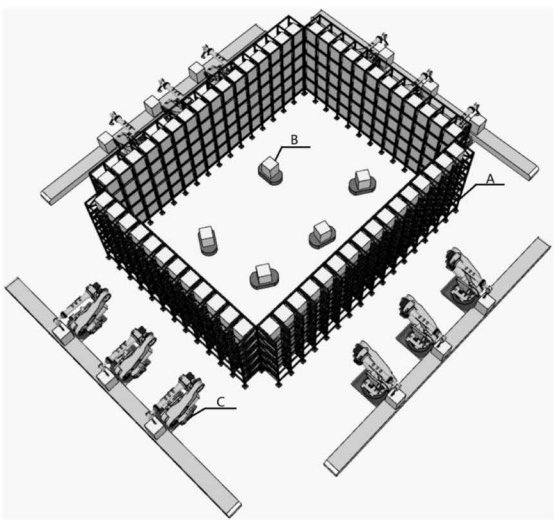


图 1 分布式并行分拣模型示意图

在该模型中，将所有 N_s 个货架首尾相连，其封闭的内部区域作为无人车的运行区域，即 H。要求任意 2 个货架在内 H 直线可达，无人车在货架内侧取货，并沿直线运送至目的货架。

该模型对现有订单分拣系统的硬件改造主要涉及货架，即原有单纯陈列货物的货架，现其内侧陈列货物，并可将相应货物交付无人车(实现方式可为推取式或机械手[20,21]等)，外侧则改造为订单的打包处。可接收分配的订单。利用无人车从其他涉及的货架上分拣货物，并在订单的所有货物到齐后打包送出。所使用的无人车可以是较为简单的，只需

要具有无线通信和运输能力,并能在货架配合下取卸货物即可。

采用该分布式并行分拣模型的分拣流程如下所述:新订单到达后被分配至某一货架打包处处理,利用无人车将订单的所有货物从相关货架中取出并运至该打包货架,待所有货物齐备后完成订单打包任务交给外部物流系统。

2.2 分布式并行整体分拣的模型化描述

本节把在第 1 节中提出的通用模型针对所提出的分布式并行策略进行具体化(图 2)。该策略对一般模型描述的订单和无人车均有影响,详述如下。

2.2.1 分布式并行整体分拣下的订单及其预处理

首先,在分布式并行策略下,订单的打包点是某一货架,因此式(4)中对订单的一般静态定义转化为

$$O'_o := \{o,\ S_o,\ t_o;\ G_g \mid g \in \mathbb{I}_o^g\} \tag{35}$$

其次,在分布式并行整体分拣下,每一个货架都可以独立接受订单的到来,这意味着式(12)~(15)中新订单产生的描述应理解为对每一个货架适用。从而,式(12)中的新订单产生过程变为

$$\Pr\{\text{一个新订单在}[t,\ t+\Delta t)\ \text{到达货架}\ S_k\} = \lambda_k \tag{36}$$

式(14)可相应定义。

最后,与式(13)中的一般性表述不同,新到来的订单自动分配至某一货架作为打包点,这就引发了一个自然的预处理过程,即所有已经在该货架上的货物都应该直接可以用来打包,无需再使用无人车运送。也就是说,在分布式并行整体打包框架下,所有 $[t,\ t+1)$ 新到来订单都要进行预处理以得到如下新的订单形式。

$$O'_N = \{o_N,\ S_{o_N},\ t;\ G_g \mid g \in \mathbb{I}_{o_N}^g \setminus \mathbb{I}_{o_N}^{g'}\} \tag{37}$$

其中,$\mathbb{I}_{o_N}^{g'} := \{g' : G_{g'} \in O_N,\ S_{g'} = S_{o_N}\}$ 是新来的订单 O'_N 中存放于该订单打包的货架 S_{o_N} 上的货物的集合。

2.2.2 分布式并行整体分拣下的无人车

首先,在分布式并行整体分拣框架下,运行中的无人车在 t 时刻的位置与一般描述并无不同,但 t 时刻空闲的无人车 $V_i \in \mathbb{V}_I(t)$ 的位置 $\boldsymbol{l}_i(t)$ 一定与某一货架相同。在系统运行之后,这一货架即为该无人车上次执行货物运送时所运送货物所属订单的打包点,即:

$$\boldsymbol{l}_i(t) = \begin{cases} S_i^0 & \max_{\tau<t} o_i^v(\tau) = 0 \\ S_{o_i^v(t')} & \max_{\tau<t} o_i^v(\tau) > 0 \end{cases} \tag{38}$$

其中,S_i^0 为无人车 V_i 在系统尚未运行时的初始位置(也为某一货架),$t' = \max_{\tau \leqslant t,\ o_i^v(\tau)=0} \tau$ 是无人车 V_i 在 t 时刻之前最后将某一货物运送到某货架打包进而空闲的时刻。

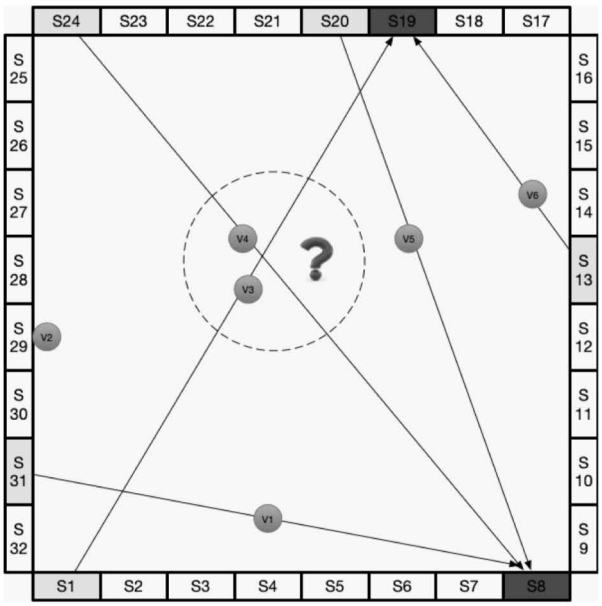


图 2 分布式并行分拣方法的模型化示意图

其次,在分布式并行策略的货架摆放下,无人车的运行轨迹可以尽可能地靠近从货物摆放货架到目的打包货架的直线。为了简化问题,本文假定任意无人车的运行轨迹一定是上述直线,而不允许有所偏离,这样的处理是对系统的整体运行效率有益的。在此假设下,任一无人车的速度方向与加速度方向应保持一致。从而,$\forall i = 1,\cdots,N_v$,$\forall t$ 若无人车 V_i 在 t 时刻的速度和加速度都至少在某一方向分量不为 0,即 $\max\{\boldsymbol{v}_i^x(t),\ \boldsymbol{v}_i^y(t)\} > 0$ 且 $\max\{\boldsymbol{a}_i^x(t),\ \boldsymbol{a}_i^y(t)\} > 0$,则不为 0 的方向对速度和加速度是一致的(不失一般性的设为 x 方向),且:

$$\frac{\boldsymbol{v}_i^y(t)}{\boldsymbol{v}_i^x(t)} = \frac{\boldsymbol{a}_i^y(t)}{\boldsymbol{a}_i^x(t)} \tag{39}$$

按照式(35),待处理订单 O'_o 的打包处在货架

S_o 上。设其中某货物的存放货架为 S_g，则无人车需要在货架 S_o 和 S_g 之间往返一次。设 S_o 和 S_g 货架在选定的笛卡尔坐标系下的坐标分别为

$$S_o = [x_o, y_o], S_g = [x_g, y_g] \tag{40}$$

因为 S_o 和 S_g 为不同货架，则 $x_g - x_o$ 和 $y_g - y_o$ 至少有一个不为 0。若，记货架 S_o 和 S_g 之间连线与 x 轴的夹角为 θ_{og}，则

$$\theta_{og} = \arctan\frac{y_g - y_o}{x_g - x_o} \tag{41}$$

进而，如果 $x_g - x_o = 0$ 且 $y_g - y_o \neq 0$，则若 $y_g - y_o > 0$ 则 $\theta_{og} = 0$，若 $y_g - y_o < 0$ 则 $\theta_{og} = \pi$。

由于 $\boldsymbol{v}_i(t)$ 和 $\boldsymbol{a}_i(t)$ 方向相同（或相反），无人车在任一时刻从打包货架 S_o 到目的货架 S_g（未到达目的地前）的行进距离为

$$\boldsymbol{l}_v^{og} = \sum_{\tau=0}^{t}\left(\boldsymbol{v}_i(\tau) + \frac{1}{2}\boldsymbol{a}_i^{\tau}\right) \tag{42}$$

从而，在 t 时刻无人车 V_i 的坐标为

$$\boldsymbol{l}_i^x(t) = x_o + \boldsymbol{l}_v^{og}\cos\theta_{\text{og}} \tag{43}$$

$$\boldsymbol{l}_i^y(t) = y_o + \boldsymbol{l}_v^{og}\sin\theta_{\text{og}} \tag{44}$$

若无人车是从 S_g 到 S_o 行进，其运行轨迹可类似得到，此时 $\theta_{og} = \pi - \theta_{og}$。

2.3 分布式并行分拣的关键技术问题

本小节针对大批量订单整体分拣问题在上述分布式并行框架下对式（29）中的算法 $A(t)$ 进行具体化。其中包含了对式（28）中函数 f_{gv} 的构建和无人车在 $[t, t+1)$ 内的加速度 $\boldsymbol{a}_i^t$，$i = 1, \cdots, N_v$ 的确定。

2.3.1 f_{gv} 的构建：优先级优先分配算法

由式（28）定义，f_{gv} 将当前待处理货物集合 $\mathbb{G}_g(t)$ 映射到空闲无人车集合 $\mathbb{V}_I(t)$。在分布式并行整体分拣框架下，f_{gv} 的构建应考虑 2 个重要因素：待处理订单的完成情况，越接近完成分拣（剩余分拣货物较少）的订单应给予较高处理优先级，这有利于优化式（34）中定义的整体分拣的分拣效率；空闲无人车与打包点距离，货物运送应安排给较近的无人车执行以减小运输距离。

首先，从式（4）定义的待处理订单集合 $\Theta(t)$ 中，计算每个待处理订单 O_o 中尚需处理的货物数量。

$$N_g^r(O_o, t) = |\{G_g \in O_o, I_g = -1\}| \tag{45}$$

待处理订单 O_o 的处理优先级 PRI_o 以如下方法确定。

若订单 O_{o1} 中剩余待处理货物较少，即 $N_g^r(O_{o1}, t) < N_g^r(O_{o2}, t)$，则剩余待处理货物较少的订单 O_{o1} 的处理优先级较高，即 $\text{PRI}_{o1} > \text{PRI}_{o2}$。

若订单 O_{o1} 和 O_{o2} 中剩余待处理货物相同且订单 O_{o1} 中货物总数大于订单 O_{o2}，即 $N_g^r(O_{o1}, t) = N_g^r(O_{o2}, t)$ 且 $|\mathbb{I}_{o1}^g| > |\mathbb{I}_{o2}^g|$，则货物总数较多的订单 O_{o1} 优先级较高，即 $\text{PRI}_{o1} > \text{PRI}_{o2}$。

若订单 O_{o1} 和 O_{o2} 中剩余待处理货物和货物总数均相同，即 $N_g^r(O_{o1}, t) = N_g^r(O_{o2}, t)$ 且 $|\mathbb{I}_{o1}^g| = |\mathbb{I}_{o2}^g|$，则二者处理先后顺序随机确定。

将优先级以自然数从 1 到 $N_o(t)$ 表示，其中数字越小优先级越大。将带有处理优先级的订单 O_o 记为 O_o^{PRI}，而每一待处理货物的优先级则与其所属订单的优先级相同，并将货物 G_g 在赋予了优先级后记为 G_g^{PRI}。则原先的待处理订单集合 $\Theta(t)$ 变为

$$\Theta^{\text{PRI}}(t) = \{O_o^{\text{PRI}} \mid o \in \mathbb{I}_o(t)\} \tag{46}$$

原先的待处理货物集合 $\mathbb{G}_g(t)$ 变为

$$\mathbb{G}_g^{\text{PRI}}(t) = \{G_g^{\text{PRI}} \mid G_g^{\text{PRI}} \in O_o^{\text{PRI}} \in \Theta^{\text{PRI}}(t), I_g(t) = -1\} \tag{47}$$

其次，空闲无人车的集合及其所在位置可由式（10）和式（38）确定。对任一待处理货物，可计算其与所有空闲无人车距离，该距离可类似式（42）进行计算。例如，待处理货物 G_g^{PRI} 与空闲无人车 $V_i \in \mathbb{V}_I(t)$ 的距离，记为 d_{gi}^s，计算如下：

$$d_{gi}^s = \sqrt{(\boldsymbol{l}_i^x(t) - S_o^x(t))^2 + (\boldsymbol{l}_i^y(t) - S_o^y(t))^2}$$

进而，对每一待处理货物 G_g^{PRI}，可计算其与所有空闲无人车的距离，并按升序排列，将该向量记为 $\boldsymbol{D}_g^s$。

$$\boldsymbol{D}_g^s = \text{asc}([d_{g1}^s \cdots d_{gi}^s \cdots d_{gN_v}^s]) \tag{48}$$

其中，asc(·)表示将向量按升序排列的函数。

主要从优化订单整体分拣的角度提出如算法 1 中描述的解决方案，其核心是以货物的运送优先级为先。注意该方法有利于在短期内完成更多的订单，但这个过程可能会使用更多的远距离无人车，从而造成对长期目标的优化不利。如何做到全局优化仍是一个待研究的复杂问题。

算法 1 f_{gv}的优先级优先分配算法

输入:t 时刻的待处理货物集合 $\mathbb{G}_g(t)$ 和空闲无人车集合$\mathbb{V}_I(t)$。

输出:f_{gv} 函数。

(1)由式(45)~(47)计算带有优先级的待处理货物集合 $\mathbb{G}_g^{\mathrm{PRI}}(t)$;由式(48)计算每件待处理货物与空闲无人车的距离向量 $\boldsymbol{D}_g^s$;

(2)将优先级最高的待处理货物分配给其与无人车的距离向量中的第一个无人车(距离最短);

(3)更新空闲无人车信息,将刚分配任务的无人车从$\mathbb{V}_I(t)$中删除;更新所有待处理货物与无人车的距离向量,将刚分配任务的无人车的距离从向量中删除;

(4)从剩余待处理货物中选择优先级最高的,重复步骤(2)和(3),直至任一条件满足:1)所有待处理货物全部分配完毕;2)空闲无人车集合为空集。

2.3.2 a_i^t 的确定:模型预测优化算法

在本文的分布式并行整体分拣框架下,无人车的所有运动决策都由某中心控制器给出。这一决策的核心是给出 $[t, t+1)$ 内每辆运行中无人车的加速度。

从单独一辆无人车的角度看,为了使其运行效率最高,应该使其一直在最高速度运行。但多辆无人车在限定区域 H 中交叉运行,如果不进行限制不可避免会遭遇碰撞。按式(33)中的要求,V_i、V_j 发生碰撞即二者之间距离小于某预定 d_v。注意到为了整体的效率优化,考虑在多步运行后的无人车状态:如果让某无人车在 $[t, t+1)$ 内保持高速运行会极大提高在此后与其他无人车碰撞的可能,限制其在 $[t, t+1)$ 内的运行速度,而这是只考虑在 $[t, t+1)$ 内的运动无法得到的。为了系统运行的整体最优,需要预测系统很多步后的可能运动状态,并从中选取合适的运动序列,但这种对未来轨迹的预测对计算能力有很高的要求,这一原因促使本文中做了 2 个决定:一方面,系统架构中采用中央控制器进行每辆无人车的路径规划,不允许每辆无人车自主规划;另一方面,在路径规划上,采用如下的模型预测控制策略,以有限步长的滚动优化来决定无人车的运动。

记无人车预测步长为 T,在 $[t, t+T)$ 内所有执行任务的无人车集合为$\mathbb{V}_o(t:t+T)$,其中包含了在此时间内一直执行任务的无人车,在 t 时刻执行任务但在 $[t, t+T)$ 完成了该项任务的无人车,和在 t 时刻未执行任务但在 $[t, t+T)$ 开始执行任务的无人车。需确定 $\boldsymbol{a}_i^t,\ \forall V_i \in \mathbb{V}_o(t, t+T)$。

由式(7)可得:

$$\boldsymbol{v}_i(t+\tau) = \boldsymbol{v}_i(t) + \sum_{\tau'=0}^{\tau-1} \boldsymbol{a}_i^{t+\tau'}$$

由上式并注意到无人车不会倒向行驶且一直保持直线行驶,无人车 V_i 在 $[t, t+T)$ 内的运动总长度 $L_i[t, t+T)$ 为

$$\begin{aligned}
&\max_{\boldsymbol{a}_i^{t+\tau},\ \tau=0,\cdots,T-1} \sum_{V_i \in \mathbb{V}_o(t:t+T)} L_i(t, t+T) \\
L_i(t, t+T) :&= \sum_{\tau=0}^{T-1} \left| \boldsymbol{l}_i(t+\tau+1) - \boldsymbol{l}_i(t+\tau) \right| \\
&= \sum_{\tau=0}^{T-1} \left(\boldsymbol{l}_i(t+\tau+1) - \boldsymbol{l}_i(t+\tau) \right) \\
&= \sum_{\tau=0}^{T-1} \left(\boldsymbol{v}_i(t+\tau) + \frac{1}{2}\boldsymbol{a}_i^{t+\tau} \right) \\
&= \sum_{\tau=0}^{T-1} \left(\boldsymbol{v}_i(t) + \sum_{\tau'=0}^{\tau-1} \boldsymbol{a}_i^{t+\tau'} + \frac{1}{2}\boldsymbol{a}_i^{t+\tau} \right) \\
&= T\boldsymbol{v}_i(t) + \sum_{\tau=0}^{T-1} \left(T - \frac{1}{2} - \tau \right) \boldsymbol{a}_i^{t+\tau} \qquad (49)
\end{aligned}$$

从而求解 $\boldsymbol{a}_i^t,\ \forall V_i \in \mathbb{V}_o(t: t+T)$ 的模型预测问题可描述如下。

$$\max_{\boldsymbol{a}_i^{t+\tau},\ \tau=0,\cdots,T-1} \sum_{V_i \in \mathbb{V}_o(t:t+T)} L_i(t, t+T) \qquad (50)$$

$$\text{s.t.} \quad \begin{cases} \boldsymbol{l}_i(t) \in H \\ 0 \leqslant \boldsymbol{v}_i(t) \leqslant \boldsymbol{v}_{\max} \\ 0 \leqslant \boldsymbol{a}_i(t) \leqslant \boldsymbol{a}_{\max} \\ d_{ij}(t) > d_v,\ \forall t,\ \forall i, j = 1, \cdots, N_v \end{cases} \qquad (51)$$

求解上述优化问题可以得到对 $V_i \in \mathbb{V}_o(t:t+T)$ 的加速度序列 $\boldsymbol{a}_i^{(t+\tau)*}$,$\tau = 0, \cdots, T-1$。进而,按照模型预测控制,选取第一个控制量作为 t 时刻的加速度,即:

$$\boldsymbol{a}_i^t = \boldsymbol{a}_i^{t*},\ V_i \in \mathbb{V}_o(t) \qquad (52)$$

2.4 分布式并行整体分拣框架和算法

将所提出的分布式并行整体分拣方法总结为算法 2,算法中所有货架都作为打包点并行进行订单处理,这是本方法效率提升的主要原因。而所造成的技术难题,即 f_{gv} 函数的确定和 $\boldsymbol{a}_i^t,\ i \in \mathbb{V}_o(t)$ 的确定,仍有很多可以提升的空间。

算法 2　分布式并行整体分拣算法

输入：$t = 0$，N_s 个货架的位置，待处理订单集合 $\Theta(0) = \varnothing$，无人车集合 $\mathbb{V} = \mathbb{V}_I(0)$，$\mathbb{V}_o(0) = \varnothing$，及其初始状态 $V_i(0)$，$V_i \in \mathbb{V}$。

输出：订单的整体分拣运行。

在 t 时刻，

(1) 按式(12)～(15)和式(35)～(37)产生新订单并更新待处理订单集合，按式(2)更新待处理货物集合；

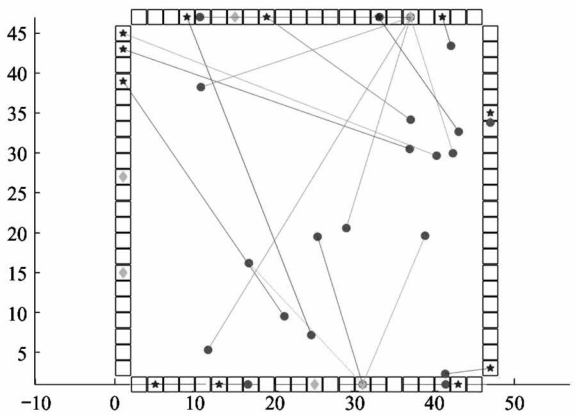
(2) 按式(45)～(48)和算法 1 确定 f_{gv} 函数，按(13)确定 $\boldsymbol{a}_i^t$，$V_i \in \mathbb{V}_o(t)$；

(3) 按式(22)～(26)更新无人车状态，按式(16)～(21)更新待处理订单集合，按式(27)更新待处理货物集合；

(4) 令 $t = t + 1$。

3　仿真算例

考虑封闭仓库区域 H 为长宽各为 50 m 和 45 m 的方形区域，将笛卡尔坐标系的坐标原点建于其中某顶点，x 轴和 y 轴各沿方形的一边，如图 3 所示。

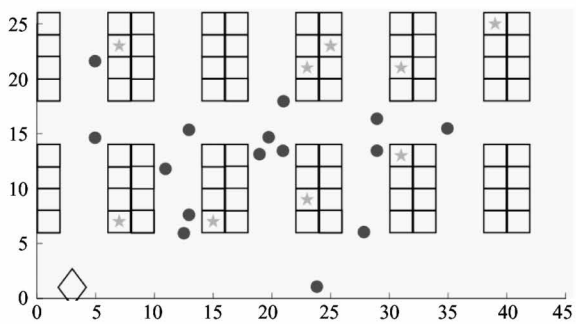


其中菱形表示某订单的打包所在货架，五角星表示某待取货的商品所在货架，圆形表示运输无人车

图 3　分布式并行整体分拣策略的仿真示意图

在仿真中，设货架数 $N_s = 88$，无人车数量 $N = 20$，在仿真开始的初始位置是随机确定的。式(36)中新订单的到达强度 $\lambda_k = 0.23$，$\forall k = 1, \cdots, N_s$，式(14)中 $\kappa = 0.1$，$\eta_j = 1/88$。无人车的速度和加速度约束为 $\boldsymbol{v}_{\max} = 3$ m/s，$\boldsymbol{a}_{\max} = 2$ m/s^2，任意 2 辆无人车的距离都不可超过某一事先预定的距离 $d_v = 3.15$ m。

本文考虑与传统订单分拣系统做比较以证明所提出方法的有效性[22]，系统架构见图 4。可以看出，该传统方法并未使用本文中提出的分布式整体分拣策略，无人车的行进路线也不限于直线，订单的分拣点在仓库一侧。常规的无人车路径分拣策略主要有返回法(return)、最大间隙法(largest gap)、S 型法(S-shape)。算法的执行步骤仍与算法 2 类似，只是在步骤(2)中，$f_{gv}(t)$ 函数由文献[23,24]中方法确定，并采取横向优先、直走先行、转弯即停的原则确定 $\boldsymbol{a}_i^t$，$V_i \in \mathbb{V}_o(t)$。



其中菱形表示订单的打包口，五角星表示某待取货的商品所在货架，圆形表示运输无人车

图 4　常规分拣方法仿真示意图

考虑 $N_t = 1$ 时的分拣效率 $J_1(t)$，即在单位时间 $[t, t+1)$ 内所完成的订单数。其演化规律在 4 种方法下的比较见图 5。从中可以看出，采用分布式整体分拣方式的分拣效率比常规分拣方法高出 3 倍。这一效率提升的原因可从分布式并行分拣允许多个订单打包点的本质特点解释。

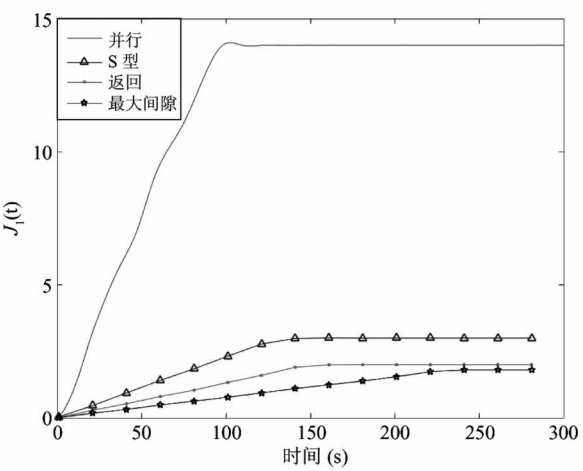


图 5　在统计时域 $[t, t+1)$ 内，4 种分拣方法的分拣效率 $J_1(t)$ 随时间的演化规律图

这4种分拣方式在系统初装和运行成本上的差别主要在无人车方面。常规方法的无人车一般需要有相当的自主能力,如避障等能力,而本文所提出的分布式整体策略的无人车则可以简化,只要可接受指令行进即可。分布式整体方法需要对货架进行一定改造,但原先货架若允许无人车分拣也应已经具有一定智能,添加打包点能力的改装并不额外增加太多成本。因此总体上来说,分布式整体分拣方法的初装和运行成本也可以有一定降低。

4 结 论

大批量订单的整体分拣是物流系统中的一个重要问题,其效率提升对整体物流效率的提升至关重要。本文提出的分布式并行模式的整体分拣策略和方法利用多打包点的想法,使用优先级订单分发和基于预测的无人车动态规划等技术,提升了订单分拣系统的整体分拣效率,具有较好的潜在应用价值。本文研究较为初步,在后续研究中,还需要在货架摆放、货物摆放、高效算法即f_{gv}和$\boldsymbol{a}_i(t)$的确定等方面做更为深入的研究。

参考文献

[1] Kulak O, Sahin Y, Taner M E. Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms[J]. *Flexible Services and Manufacturing Journal*, 2012, 24(1): 52-80

[2] Wang H, Baek W J, Lee M K. Clustering algorithms for order picking in an automated storage and retrieval system [J]. *International Journal of Production Research*, 1988, 26(2): 189-201

[3] Önüt S, Tuzkaya U R, Doğaç B. A particle swarm optimization algorithm for the multiple-level warehouse layout design problem[J]. *Computers and Industrial Engineering*, 2008, 54(4): 783-799

[4] 卢尧,任晓明,吴勇志, 等. 智能仓储控制系统的设计与实现[J]. 自动化仪表, 2017, 38(2): 28-30

[5] 赵金萍,熊君星,邹文强, 等. 基于flexsim的自动化立体仓库出入库仿真与优化[J]. 高技术通讯,2017, 27(1): 81-87

[6] 胡颖聪,刘建胜,张有功. 基于AGA与MPSO的非传统布局仓储货位分配优化[J]. 高技术通讯, 2018, 28(11-12): 980-990

[7] 柳赛男,柯映林. 基于CTPN的自动化仓库系统调度策略研究[J]. 高技术通讯, 2008, 18(8): 823-829

[8] Scholz A, Schubert D, Wäscher G. Order picking with multiple pickers and due dates-simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems[J]. *European Journal of Operational Research*, 2017, 263(2): 461-478

[9] Scholz A, Wäscher G. Order batching and picker routing in manual order picking systems: the benefits of integrated routing[J]. *Central European Journal of Operations Research*, 2017, 25(2): 491-520

[10] Lin C C, Kang J R, Hou C C, et al. Joint order batching and picker manhattan routing problem[J]. *Computers and Industrial Engineering*, 2016, 95: 164-174

[11] Valle C A, Beasley J E, Da Cunha A S. Optimally solving the joint order batching and picker routing problem [J]. *European Journal of Operational Research*, 2017, 262(3): 817-834

[12] Zhang B, Li L W, Zhao Y H, et al. The research on E-commerce logistics picking AGV path optimization method based on the improved A* algorithm[C]//The 2nd International Conference on Cybernetics, Robotics and Control (CRC)-IEEE, Hong Kong, China, 2016: 99-103

[13] Werners B, Thorn J, Freiwalds. Robust optimization of internal transports at a parcel sorting center[J]. *European Journal of Operational Research*, 2010, 201(2): 419-426

[14] 禹鑫燚, 陈浩, 郭永奎, 等. 基于线性时序逻辑理论的仓储机器人路径规划[J]. 高技术通讯, 2016, 26(1): 16-23

[15] Hwan D, Kim H J, Kim S B. Closed loop motion synchronous velocity control for AC motor drives: a solution for increasing speed of a cross-belt sorting conveyor system[C]//The 3rd International Conference on Advanced Engineering Theory and Applications, Busan, Korea, 2016: 577-586

[16] 刘建胜,熊峰,陈景坤, 等. 基于蚁群算法的双分区仓库拣货路径的优化[J]. 高技术通讯, 2017, 27(1): 72-80

[17] Yuan Z, Gong Y Y. Bot-in-time delivery for robotic mo-

bile fulfillment systems[J]. *IEEE Transactions on Engineering Management*, 2017, 64(1): 83-93

[18] Caputo A C, Pelagagge P M. Management criteria of automated order picking systems in high-rotation high-volume distribution centers[J]. *Industrial Management and Data Systems*, 2006,106(9): 1359-1383

[19] Lee S D, Kuo Y C. Exact and inexact solution procedures for the order picking in an automated carousal conveyor [J]. *International Journal of Production Research*, 2008, 46(16): 4619-4636

[20] Zeng A, Yu K T, Song S, et al. Multi-view self-supervised deep learning for 6D pose estimation in the Amazon picking challenge[C]//IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016: 1386-1383

[21] 伍锡如, 黄国明, 孙立宁. 基于深度学习的工业分拣机器人快速视觉识别与定位算法[J]. 机器人, 2016, 38(6): 711-719

[22] Roodbergen K J. Storage assignment for order picking in multiple-block warehouses[J]. *Warehousing in the Global Supply Chain*, 2012: 139-155

[23] Hong S, Kim Y. A route-selecting order batching model with the s shape routes in a parallel-aisle order picking system[J]. *European Journal of Operational Research*, 2017, 257(1): 185-196

[24] Henn S, Koch S, Wäscher G. Order Batching in Order Picking Warehouses: A Survey of Solution Approaches [M]. Berlin: Springer, 2012:105-137

Modeling and distributed parallel solution for the overall picking of large volume orders

Zhao Yunbo, Li Tianshu, Wang Yuhao

(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310000)

Abstract

The efficient overall picking of large volume orders is a key factor affecting the e-commerce logistic efficiency, however, which is lack of rigid modeling and effective solutions. By mathematically describing the static state and the dynamic evolution of order picking system, as well as defining the algorithms, constraints and efficiency, the mathematical model of order picking system is built. Following this model and by using smart shelves, a distributed parallel solution is proposed, which can effectively deal with the bottleneck of the lack of picking points in such order picking system. The technical issues are solved and the performance of the proposed solution is verified numerically by comparing with conventional solutions.

Key words: large volume order, overall sorting, modeling, distributed parallel solution