Advances in Applied Mathematics 应用数学进展, 2020, 9(8), 1239-1245
Published Online August 2020 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2020.98145

Hans汉斯

Study on the Problem of Picking Goods in Warehouse

Hao Li[1], Hangyu Meng[2], Chunyu Chen[2]

[1]Nanjing Tech University of Physical and Mathematical Sciences, Nanjing Jiangsu
[2]Nanjing Tech University of Overseas Education, Nanjing Jiangsu
Email: 952656623@qq.com

Received: Jul. 26th, 2020; accepted: Aug. 13th, 2020; published: Aug. 20th, 2020

Abstract

Nowadays, the domestic B2C e-commerce warehouse is mostly a human-to-material picking mode. Picking operations have become one of its core operations, occupying a large amount of time and capital costs in the warehouse. The optimization of picking methods has become an urgent problem for enterprises to solve. This paper aims at the optimization problem of picking in the warehouse, through the shortest path based on the picker, the shortest time, gives the operating rules and operating methods, and gives a set of actual data, and then uses the computer to simulate the actual picking situation, and gives the optimization results and the corresponding solution.

Keywords

Picking in Warehouse, Greedy Principle, Optimization Problem, Computer Simulation

仓内拣货问题研究

李　浩[1]，孟航宇[2]，陈春宇[2]

[1]南京工业大学数理科学学院，江苏　南京
[2]南京工业大学海外教育学院，江苏　南京
Email: 952656623@qq.com

收稿日期：2020年7月26日；录用日期：2020年8月13日；发布日期：2020年8月20日

摘　要

当前国内B2C电子商务仓库多为人至物的拣货模式，拣货作业成为其核心作业之一，占据仓库大量时间成本和资金成本，拣货方式的优化成为企业亟待解决的问题。本文针对仓内拣货的优化问题，通过基于

文章引用: 李浩, 孟航宇, 陈春宇. 仓内拣货问题研究[J]. 应用数学进展, 2020, 9(8): 1239-1245.
DOI: 10.12677/aam.2020.98145

拣货人的最短路径，最短时间，给出运行规则，运行方法，并给出一组实际数据，使用计算机模拟实际拣货情况，给出优化结果，以及结果所对应的方案。

关键词

仓内拣货，贪心原则，优化问题，计算机模拟

1. 引言

随着经济的迅速发展，使得社会需求日益增加，企业专业化生产也越来越深入，物流行业在中国也得到了快速发展，其中也包括供应链物流。现在的物流市场日趋成熟，竞争越来越激烈，提高客户的满意程度成为物流市场的主要竞争目标。影响客户满意程度的因素很多，其中主要的一个因素就是配送速度。对于供应链物流来说，可以通过优化其配送环节来提高配送速度与效率，其中拣货就是配送过程中的关键步骤[1]。

2. 仓内拣货问题介绍

电商公司客户订单下达仓库后，商品开始下架出库，出库主要包含 5 个流程如下图所示：

1) 定位：仓库有多个货架，每个货架有多个货格，商品摆放在货格中，且每个货格最多摆放一种商品，商品可以摆放在多个货格。订单下达仓库后，定位操作，确定商品下架的货格和每个货格下架的商品数量。

2) 组单：单个客户订单商品数量少，对于中小件商品仓库，需要将多个客户的订单合并，构成任务单，这就是组单操作。

3) 拣货：拣货开始，拣货员在某个复核台领拣货车及任务单，领取时间不计，然后根据推荐顺序依次访问任务单中商品所在货格，并下架商品，将商品放在拣货车上。下架完毕，拣货员将拣货车送往某个复核台，到达复核台后拣货员无需等待，继续领取拣货车和任务单，开始下一个任务单拣货流程。

4) 复核台先对任务中商品复核，然后将商品按照订单打包。

为方便问题的研究，我们给出一组信息，进行实际计算：

假设 4 个复核台(FH01，FH03，FH10，FH12)正常工作，49 个任务单(T0001~T0049)等待拣货，9 个拣货员(P1~P9)负责拣货，要求给每个拣货员分配任务单、起始拣货复核台，并分别规划理想的拣货路线，使得 49 个任务单尽快完成出库，并计算完成出库需要花费的时间和每个复核台利用率。

3. 模型的假设

1) 假设仓内货物充足，不会出现短缺。

2) 假设每个任务单的完成时间只与起始和终止复核台的选择有关。

3) 假设拣货员将货物放到复核台可直接再去取货，忽略中转时间。

4) 多人同时在一个货格拣货，不考虑等待的时间。

5) 拣货员每次均可完全按照任务单的要求进行拣货，不考虑因为漏拣，错拣所耽误的时间。

4. 模型的建立与求解

由于有 9 个拣货员，而只有 4 个复核台，必定会造成复核台的拥堵。因此，复核台成为影响总体出库时间的关键。同时，为了每个使任务单都尽快出库，减少任务单在仓库的平均“逗留时间”(即从任务单出现至出仓的时间)，我们以“需要拣货时间短的任务单先执行”的第一原则进行处理。由于总的出仓时间取决于最后一个工作完毕的复核台的时间，我们以“对 4 个复核台和 9 个拣货员的工作时间的平均分配”为第二原则进行，这样可以减少最后一个完成工作的复核台的时间，同时可以减少复核台的堵塞。

接下来，我们通过计算机模拟的方式按照上述原则仿照实际情况，对任务单进行处理，得出具体的实施方案，已经相应的时间信息。

4.1. 初始化阶段

1) 初始的时候形成一个 49 列，16 行的矩阵 C，列对应 49 个货单，行对应 16 个不同的开始和终止复核台的方案，C_{ij} 表示其第 i 行第 j 列的元素，其值表示货单 j 按第 i 行的方案运行所需的时间。矩阵 C 的值如表 1 所示。

Table 1. Time required for a task single scenario (unit: s)
表 1. 任务单个方案所需时间(单位：s)

	T0001	T0002	T0003	……	T00049
FH1-FH1	536.27	574	533.73	……	408
FH1-FH3	530.27	568	527.73	……	402
……	……	……	……	……	……
FH12-FH12	524.13	537.4	487.8	……	403.93

2) C_d 为 C 动态矩阵，用来存储 C 的加权信息；

3) F 为 4 维行向量，每一个元素代表一个复核台，用来动态存储复核台的工作时间，初始值为 0。

4) h 为 49 维向量，则 h_j 为其第表示第 j 个元素，含义是任务单 j 所需的复核时间；h 的值如表 2 所示。

Table 2. Task order review time (unit: s)
表 2. 任务单所需复核时间(单位：s)

任务单	T0001	T0002	T0003	……	T0049
时间	300	450	330	……	300

5) P 为 9 列 10 行的矩阵，则每一列代表一个人所执行的任务单 j；

6) Q 与 P 同规模，用来存放 P 对应元素的任务单所对应的方案 i；

7) P_t 为 9 维列向量，每一列对应一个拣货员，用来动态储存拣货员的积累行动时间，初始值 $P_t = 0$。

4.2. 任务执行阶段

Step 1：对 C_d 中的元素进行加权。将 F 中每个复核台对应的元素值，加入到 C_d 中以该复核台为终止复核台的方案中，从而使得选到相对忙碌的复核台为终止复核台的概率降低。

Step 2：选取 C_d 中最小权值的元素的位置 (i,j)，将 C_{ij} 对应的任务作为第 k 个人的任务，然后 j 将存入 P 中第 k 个人对应的位置处，i 存入 Q 的相应位置，P_t 中第 k 个人对应元素增加 C_{ij}。

Step 3：判断 F 中方案 i 的终止复核台对应元素是否为 0，若为 0，F 中该元素的值加 h_j，否则加 $h_j + C_{ij}$。

Step 4：将第 j 列的值都置为极大 inf。

Step 5：最后恢复 C_d，重新令 $C_d = C$。

Step 6：重复执行 Step 1 至 Step 5 共 9 次，来确定 9 个人的初始任务。

Step 7：搜索 P_t 中的最小值所存的位置 R，找到 Q 中第 R 个人最后一次任务的方案，保证任务先完成的先开始下一任务。

Step 8：确定第 R 个人当前最后一次任务方案的终止复核台 FH。执行 Step1，在矩阵 C_d 中寻找以该终止复核台为起点的方案中的最小值元素的位置 (i,j)，选定之后把 j，i 分别放入 P，Q 中，C_{ij} 的值加入到 P_t 中该人的位置 R 处。然后把 C_{ij} 对应的 h_j 加入到 F 中方案 i 的终止复核台对应元素中。

Step 9：恢复 C_d 重新令 $C_d = C$。

Step 10：重复执行 Step 7 到 Step 9，直至所有任务单均已被选定。

Step 11：计算最终结果，在得到的结果之中，P_t 为 9 个人分别行动的时间，Q 和 P 为相应的每个人的任务信息，为复核台的工作时间信息，并且 F 中的最大值即为最终的出仓时间。

4.3. 模型求解

通过 MATLAB 编程进行求解，得出拣货方案结果如表 3、表 4 所示，复核台与拣货人员的工作时间如表 5、表 6 所示，复核台的效率如表 7 所示。

Table 3. Operation scheme number
表 3. 运行方案编号

1	FH1-FH1	9	FH10-FH1
2	FH1-FH3	10	FH10-FH3
3	FH1-FH10	11	FH10-FH10
4	FH1-FH12	12	FH10-FH12
5	FH3-FH1	13	FH12-FH1
6	FH3-FH3	14	FH12-FH3
7	FH3-FH10	15	FH12-FH10
8	FH3-FH12	16	FH12-FH12

Table 4. Programs and tasks performed by pickers (left: Program No., right: Task Order No.)
表 4. 拣货员执行的方案与任务(左边：方案编号，右边：任务单编号)

	P1	P2	P3	P4	P5	P6	P7	P8	P9
Q|P	14|23	16|49	5|5	7|15	16|9	7|13	1|30	6|37	16|42
Q|P	5|19	14|33	2|39	12|36	15|24	9|4	2|34	7|27	13|47
Q|P	12|14	7|22	8|43	13|3	10|46	1|1	8|6	11|29	2|32
Q|P	15|17	10|18	14|31	4|38	5|21	2|8	16|35	11|7	5|48
Q|P	9|28	7|10	8|20	13|2	3|41	8|14	15|11	10|25	1|45
Q|P	2|16	9|40	0|0	4|44	11|26	0|0	0|0	0|0	0|0

Table 5. Cumulative working time of review Desk (unit: s)
表 5. 复核台累计工作时间(单位：s)

复核台	时间
FH01	5602.67
FH03	5328.23
FH10	5666.47
FH12	5623.93

Table 6. Cumulative working hours of the picker (unit: s)
表 6. 拣货员的累计工作时间(单位：s)

人的编号	时间
P1	3006.97
P2	3021.97
P3	2526.77
P4	3083.17
P5	3074.77
P6	2590.1
P7	2560
P8	2558.53
P9	2585.77

Table 7. Operating efficiency of the review table and total warehousing time (time unit: s)
表 7. 复核台的运行效率及出仓总时间(时间单位：s)

复核台	FH01	FH03	FH10	FH12
效率	0.9107	0.8736	0.9212	0.9212
出仓总时间	5666.467			

5. 总结

对于此问题，需要解决 9 个拣货员，4 个复核台，49 个任务单的任务分配，任务单的执行顺序等问题。我们给出拣货员领取任务单的原则，以及方案选择的选择，构建了一套计算机模拟体系，运用了 MATLAB 模拟，得到总的出库时间为 5666.47 s，复核台的利用率分别为 0.9107、0.8736、0.9212、0.9192。

参考文献

[1] 王雄志. 配送中心拣货计划方法研究[D]: [博士学位论文]. 广州: 暨南大学, 2007.

附 录

MATLAB 程序

```
load('zuobiao_H.mat')
load('zuobiao_C.mat')
load('zuobiao_fht.mat')
P=zeros(10,9);
P1=zeros(size(P));
Pt=zeros(1,9);
F=[1,3,10,12;0,0,0,0];
      C=zuobiao_C;
Cd=C;
      for i1=1:9
            for i3=1:4    %加权
          for i2=i3:4:i3+12
                Cd(i2,:)= Cd(i2,:)+F(2,i3);
          end
            end
      a1=min(min(Cd));
      [b1,c1]=find(Cd==a1);
            P(1,i1)=c1;
      P1(1,i1)=b1;
      a2=C(b1,c1);
      Pt(1,i1)=Pt(1,i1)+a2;
      h=zuobiao_H(1,c1);         %为求 F
      z1=zuobiao_fht(b1,2); %终止复核台编号
      [~,d1]=find(F(1,:)==z1);
     if F(2,d1)==0
          F(2,d1)=F(2,d1)+a1+h;
     else
      F(2,d1)=F(2,d1)+h;
     end
      C(:,c1)=99999999999999;
      Cd=C;       %重置
      end
      %前 9 次循环结束
      for i4=10:49
            for i3=1:4      %加权
          for i2=i3:4:i3+12
                Cd(i2,:)= Cd(i2,:)+F(2,i3);
          end
```

```
        end
        e1=min(Pt);
        [~,k1]=find((Pt)==e1);
        zero_index=find(P1(:,k1)==0);   %找一列中最后一个任务方案
          first_zero_index=zero_index(1);
          f1=first_zero_index-1;
          P1_end=P1(f1,k1);
          z2=zuobiao_fht(P1_end,2);%终止复核台编号
          g1=find(zuobiao_fht(:,1)==z2);
            n1=Cd(g1,:);
          m1=min(min(n1));
        [p1,q1]=find(Cd==m1);
          P(f1+1,k1)=q1;
          P1(f1+1,k1)=p1;
          m2=C(p1,q1);
          Pt(1,k1)=Pt(1,k1)+m2;
          h2=zuobiao_H(1,q1);         %为求 F
  z2=zuobiao_fht(p1,2); %终止复核台编号
  [~,d2]=find(F(1,:)==z2);
if F(2,d2)==0
      F(2,d2)=F(2,d2)+a1+h2;
else
  F(2,d2)=F(2,d2)+h2;
end
            C(:,q1)=99999999999999;
                Cd=C;       %重置
  end
```