doi:10.3772/j.issn.1002-0470.2017.01.009

基于蚁群算法的双分区仓库拣货路径的优化①

刘建胜② 熊 峰 陈景坤 熊君星

(南昌大学机电工程学院 南昌 330031)

摘 要 为提高现代仓库作业中拣货这一核心环节的效率,研究了仓库拣货路径的优化,提出了根据双分区仓库中拣货路径的特点,采用蚁群算法优化求解的拣货路径算法,并通过仿真将该算法的性能与传统穿越策略、S 形启发式算法进行了比较。比较结果表明,以蚁群算法优化路径问题可以明显减少路径的距离,具有良好的实用性。

关键词 物流,拣货,车辆路径,蚁群算法

0 引 言

拣货作业在现代仓库作业中是一项十分重要的环节,即按照物料订单属性,包括物料编号、名称、数量及体积等,拣货人员把货物从货架取出拣选下来,并放在指定的位置。研究报告显示,拣货作业的人工费用和时间消耗占仓库全部劳动人工费用和时间消耗的比例高达 60%,无论从时间和人力的投入上看还是从对订单服务的影响上看,拣货作业都成为现代仓库作业的重要环节。拣货作业涉及物料拣取路径问题,这是一种组合优化问题。当规模较小时,该问题可以使用分枝定界求解,但当规模很大时,求解的复杂程度呈指数增长,必须使用启发式算法或智能算法等求满意解。

Goetschalckx[1] 等提出了 S-shape 方法优化路径,在问题规模不大的情况下能较好解决拣货路径问题。熊芳敏[2] 等通过蚁群算法与传统穿越策略对比,证明了蚁群算法解决此问题的可行性和优越性,但缺乏对一单多车场景研究。本文研究的是基于蚁群算法的现代仓库拣货路径的优化,将问题分为不考虑容积限制(一单一车)和考虑容积限制(一单多车)两种情况。在不考虑容积时将蚁群算法的求解结果与 S 形算法[3-6]、传统穿越策略求解结果进行比较,并用蚁群算法对一单多车的情况进行求解。

1 问题的描述与模型

1.1 问题描述

按结构和构造不同,仓库分为平面仓库、多层仓库、高层货架仓库、散装仓库。本文研究的仓库为平面仓库中的双分区仓库。双分区仓库由若干平行的巷道组成,货架分布在巷道的两侧,货架与巷道平行排列,集货点处于分拣区的左下角,该仓库有横向三条过道 a、b 和 c。根据相关文献[7,8]分析,对仓库进行分区,仓库中存在巷道对提高拣选效率有很大作用。

整个仓库平面如图 1 所示,I/O 为仓库的出入口,每个小方格代表一个货物储位,黑色方块代表订单上的货物,数组 $\{x-y-z\}$ 代表对应的储位编号,其中 x 代表所在的拣货巷道号;y 表示在巷道的左边还是右边(1 表示位于左边,2 表示位于右边);z 表示所在的行号。例如:{7 -1 -33} 表示储位位于第 7 号巷道左边第 33 行。

① 国家自然科学基金(51565036)资助项目。

② 男,1978 年生,博士,副教授;研究方向:数字化与智能制造,设施布局优化,物流管理与优化技术;联系人,E-mail: liujiansheng@ncu.edu.cn

(收稿日期:2016-04-06)

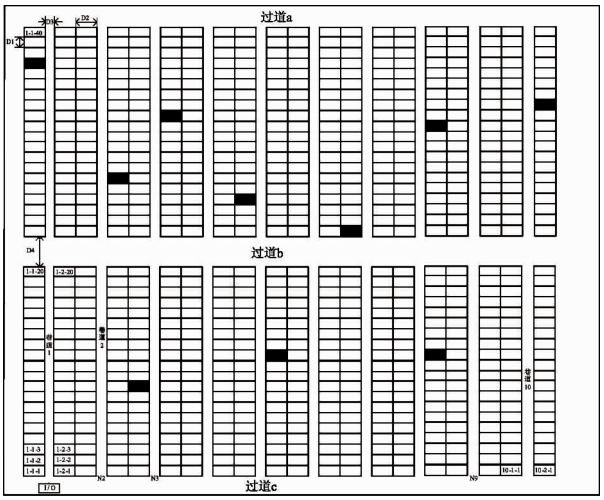

图1　双分区仓库平面

在各种算法解此问题时,都需要计算两点之间行走的路程。任意取仓库中的储位 i, j, 其编号分别为 $\{x_i - y_i - z_i\}$ 和 $\{x_j - y_j - z_j\}$, 规定过道 b 以上的区域为 $Area_1$ 区域,过道 b 以下的区域为 $Area_2$ 区域,复核台为 $\{1-1-0\}$。仓库中任意两点的路程计算如下所示:

当 i、j 位于同一巷道并且 i、j 位于过道 b 以上或过道 b 以下,此时路程计算式为

$$c_{ij} = |z_i - z_j| \times D_1$$

当 i、j 位于同一巷道并且 i、j 位于过道 b 的两则,此时路程计算式为

$$c_{ij} = |z_i - z_j| \times D_1 + D_4$$

当 i、j 位于不同巷道并且 i、j 位于过道 b 以上,此时路程计算式为

$$c_{ij} = \min\begin{cases} |x_i - x_j| \times (D_3 + 2D_2) + (82 - z_i - z_j) \times D_1 \\ |x_i - x_j| \times (D_3 + 2D_2) + (z_i + z_j - 40) \times D_1 \end{cases}$$

当 i、j 位于不同巷道并且 i、j 位于过道 b 以下,此时路程计算式为

$$c_{ij} = \min\begin{cases} |x_i - x_j| \times (D_3 + 2D_2) + (42 - z_i - z_j) \times D_1 \\ |x_i - x_j| \times (D_3 + 2D_2) + (z_i + z_j) \times D_1 \end{cases}$$

当 i、j 位于不同巷道并且 i、j 位于过道 b 两则,此时路程计算式为

$$c_{ij} = |z_i - z_j| \times D_1 + |x_i - x_j| \times (D_3 + 2D_2) + D_4$$

1.2　问题分类

本文将拣货订单的拣货方式分成一单一车和一单多车两种类型。一单一车不需要考虑推车的容积和载重,一单多车需考虑推车的载重约束,这种情况下推车需多次往返才能将订单中全部货物从仓库取出。

1.3　建立模型

将所拣选物品位置、拣货车的出发和返回地抽象为点,而点与点之间的路程计算可得到距离矩阵,因此拣货路径优化问题可映射为旅行商问题(travelling salesman problem, TSP)。

1.3.1　一单一车问题描述及模型

给定一组网络 $G = (N, A, C)$, 其中 N 为点的集合, A 为边的集合, $C = [c_{ij}]$ 为距离矩阵,表示从储位 v_i 到储位 v_j 的距离。拣货路径问题就是求解一条通过集合 N 所有的点且仅一次,并回到原来的点的最短距离路线。

一单一车拣货路径问题的数学模型为

$$\min \sum_{i=1}^{n+1} \sum_{j=1}^{n+1} c_{v_iv_j} x_{v_iv_j} + c_{v_{n+1}v_1} \tag{1}$$

s. t.

$$\sum_{i=1}^{n} x_{v_iv_j} = 1 \quad (j = 1,2,\cdots,n+1) \tag{2}$$

$$\sum_{j=1}^{n} x_{v_iv_j} = 1 \quad (i = 1,2,\cdots,n+1) \tag{3}$$

$$\sum_{i,j \in R} x_{v_iv_j} \leqslant R - 1 \tag{4}$$

$$x_{v_iv_j} \in \{0,1\} \quad (i, j = 1,2,\cdots,n+1) \tag{5}$$

其中 v_1 表示源点(I/O 口), $c_{v_iv_j}$ 为储位 v_i 到储位 v_j 的最短距离, $c_{v_{n+1}v_1}$ 为最后一个捡货点返回源点 v_1 的最短距离; $x_{v_iv_j}$ 为决策变量,当 $x_{v_iv_j} = 1$ 表示经过 (v_i, v_j) 这条边,反之, $x_{v_iv_j} = 0$ 表示没有经过 (v_i, v_j), 式 (1) 为目标函数求解总路程最短,约束式(2)、(3)表示每个点只能通过一次,式(4)表示路径不存在小回路, R 表示集合中点的个数,式(5) $x_{v_iv_j}$ 表示可能的取值。

1.3.2　一单多车问题描述及模型

(1)模型假设:仓库有 y 次小车从源点有序出发捡取货物,每辆小车的载重量为 W, 假设储位为 $v_1, v_2, v_3, \cdots, v_{n+1}$。每个储位存放的待拣货量为 Q_{v_j}, 且每个储位的待拣货量均小于小车的额定载重量

($Q_{v_j} \leqslant W$)，当捡取一定数量的货物后小车返回源点。源点的坐标为[1,1,0]，并且其他捡货点的位置已知，每个捡货点只能由一辆小车捡货。

(2)模型目标：每次小车取货路径都构成一个回路，要求所有取货小车走过的回路长度之和$f(x)$最小。

(3)符号说明：

i：每次车所对应的编号；

Q_{v_j}：表示储位 v_j 上待捡取的货量；

$Q^i_{v_k}$：表示第 i 次车捡货回路上第 k 个储位的取货量；

g_i：表示第 i 次车捡货回路上的储位数；

n：订单中所有货物对应的储位总数；

c_i：第 i 次车对应的捡货路径；

c^i_k：第 i 次车对应的捡货回路中顺序为 k 的储位；

V：订单上所有货物的总量；

W：单辆小车的最大载重量；

$c^i_{v_k v_{k+1}}$：第 i 次车捡货回路上第顺序为 k 和 $k+1$ 个储位之间最短的距离；

$c^i_{v_1 v_{g_{i+1}}}$：第 i 次车拣货回路上最后一个储位与 I/O 口之间的最短距离；

(4)模型建立：

对于任意一个路径序列 $x = \{v_1, v_2, \cdots, v_n, v_{n+1}\}$，其中 v_1 表示复核台。它的期望距离表示如下：

$$\min f(x) = \sum_{i=1}^{y} \left(\sum_{k=1}^{g_i} c^i_{v_k v_{k+1}} + c^i_{v_{g_{i+1}} v_1} \right) \tag{6}$$

$$\sum_{k=2}^{g_{i+1}} Q^i_{v_k} \leqslant W \tag{7}$$

$$\sum_{i=1}^{y} \sum_{k=2}^{g_{i+1}} Q^i_{v_j} = V \tag{8}$$

$$\sum_{i=1}^{y} g_i = n \tag{9}$$

$$c_i = \{c^i_k / c^i_k \in \{v_1, v_2, v_3, \cdots, v_{n+1}\}, \quad k = 1,2,3,\cdots,g_{i+1}\} \tag{10}$$

$$c_i \cap c_j = v_1, \ \forall i \neq j \tag{11}$$

式(6)为目标函数，最短的行走距离。式(7)为推车载重限制约束，即每一次车从储位捡取货物的总量不超过小车的载重量。式(8)表示 y 次车将订单所有货物全部捡取。式(9)，(10)，(11)表示从每个待捡储位有且仅能由一辆小车捡货。

可以看出，当小车的载重量 W 为无穷大时，即为一单一车的情况。

2 蚁群算法

蚁群算法是一种新型的模拟进化算法，该方法已经用于求解旅行商、指派问题、调度问题等，取得了一系列较好的实验结果。

2.1 参数初始化

n 为节点总数，$NC\_\max$ 为最大迭代次数，α 为信息启发式因子，β 为期望启发式因子，ρ 为信息素挥发因子，m 是蚂蚁的个数，对第 k 只蚂蚁，allowed_k 为该只蚂蚁为访问的点的集合，tabu_k 为该只蚂蚁已访问的点的集合，η_{ij} 为启发值，τ_{ik} 为信息素。

2.2 概率转换规则

迭代开始时，各条路径上的信息素相等 ($\tau_{ik}(t) = C$(C 为常数))，将 m 只蚂蚁随机分配到 n 个节点上，蚂蚁 k($k = 1,2,3,\cdots,m$)在运动过程中，根据各条路径上的信息量决定转移方向，$p^k_{ij}(t)$ 表示在 t 时刻蚂蚁 k 有位置 i 转移到位置 j 的概率为

$$p^k_{ij} = \begin{cases} \dfrac{\tau^{\alpha}_{ij}(t) \cdot \eta^{\beta}_{ij}(t)}{\sum\limits_{s \in \text{allowed}_k} \tau^{\partial}_{is}(t) \cdot \eta^{\beta}_{is}(t)}, & j \in \text{allowed}_k \\ 0, \text{其他} \end{cases} \tag{12}$$

2.3 信息素更新

经过 n 个时刻，蚂蚁完成一次循环，各路径上的信息量要根据以下公式作调整：

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij} \tag{13}$$

$$\Delta\tau_{ij}(t+n) = \sum_{k=1}^{m} \Delta\tau^k_{ij} \tag{14}$$

本文中使用 Ant-cycle system 模型：该模型中，

$$\Delta\tau^k_{ij} = \begin{cases} \dfrac{Q}{L_k} & \text{若第 } k \text{ 只蚂蚁在时刻 } t \text{ 和时刻 } t+1 \text{ 经过 } ij \\ 0 & \text{其他} \end{cases} \tag{15}$$

$\Delta\tau_{ij}$ 表示在本次循环中路径 ij 上的信息素的增

量,L_k 表示第 k 只蚂蚁环游一周的路径长度,Q 为常数。

3 蚁群算法参数的设置

蚁群算法的关键在于参数的确定,即研究 α,β,ρ,m,Q 的最佳配置[9-11]。目前尚无理论依据,针对具体的问题可以用实验方法确定其最优组合。

3.1 信息素挥发率 ρ 的设置

为了防止残留信息过多引起的残留信息淹没启发信息的问题,在每一只蚂蚁完成对所有 n 个城市的访问后必须对残留信息进行更新处理,模仿人类记忆的特点,对旧的信息进行削弱。ρ 的大小直接关系到算法的全局搜索能力和收敛速度。为了得到满意的 ρ,随机选择 20 个储位,将参数 $m = 20$,$\alpha = 1$,$\beta = 2$,$Q = 100$ 保持不变,将信息素保留率分别设置为 $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$,以此研究信息素保留率对收敛速度和优化结果的影响。表 1 给出了信息素挥发率对收敛速度与搜索能力的影响。从表 1 可知,ρ 对算法收敛速度影响很大,在处理规模较大的问题时,信息挥发度 $1-\rho$ 的存在会使未被搜索到的路径上的信息量减小,因而降低了算法的全局搜索能力。当信息挥发度 $1-\rho$ 过大时之前搜索过路径被再次选择的机会增大,从而影响了算法的随机性和全局搜索能力;当信息挥发度 $1-\rho$ 过小时,虽然可以提高算法的随机性和全局搜索能力,但是会使算法的收敛速度降低。因此要从多方面综合考虑来选取信息素保留率 ρ 的取值,经比较,参数 ρ 取值范围在 $\rho \in [0.5, 0.9]$ 间较合理,本文中采用 $\rho = 0.7$。

表 1 信息素挥发率 ρ 对收敛速度与搜索能力的影响

ρ	0.1	0.3	0.5	0.7	0.9
最短路径(m)	292	290	286	284	284
搜索循环次数	31	41	56	80	93

3.2 蚂蚁数量 m 的设置

蚂蚁数量对算法全局搜索能力有一定的影响,蚂蚁数量越多,算法全局求解的可靠性更高,但是算法收敛速度也越来越慢。经研究,蚂蚁数量对算法的循环次数的影响基本呈线性规律变化,蚂蚁数量过大,算法的收敛速度变慢。在保持其他参数不变的情况下,设置不同数量的蚂蚁数量,得到蚂蚁数量对算法收敛速度和搜索能力的影响如下。本文假设每个订单中待拣货的储位数目为 20,对全局搜索能力和收敛速度两个方面综合考虑,m 的取值范围在 $m \in [n, n/2]$,本文中选用 $m = \frac{3}{4}n$,即蚂蚁总数为 15。表 2 给出了蚂蚁数量对收敛速度与搜索能力的影响。

表 2 蚂蚁数量 m 对收敛速度与搜索能力的影响

m	5	10	15	20	30
最短路径(m)	300	286	284	287	284
搜索循环次数	37	69	56	77	98

3.3 启发因子 α 及能见度启发因子 β 的设置

α,β 两个启发因子可以综合影响蚂蚁对两点间路程信息素浓度的敏感度。α 是信息素启发因子,当 α 取较大值时,就意味着蚂蚁在选择下一个节点时路径上的信息量起主要作用,蚂蚁过分依赖路径上的信息量引导搜索,会使算法过早收敛,在问题规模较大时易使最终结果只是局部最优解。β 是自启发量因子,当 β 过大时,算法会过早收敛于局部最优解;当 β 过小时,蚂蚁群体会陷入随机搜索状态,此时算法收敛速度慢且很难找到最优解。当 α,β 均过大时,算法容易出现过早停滞且搜索的结果不理想;当 α,β 均过小时,算法虽然不会出现过早停滞但依旧得不到理想的结果。因此 α 和 β 的选择对算法的影响是非常大的。在保持其他参数不变的情况下,选取了几组常用的 α 与 β 的组合,仿真结果如下,综合考虑其影响,本文选择 $\alpha = 1.5$,$\beta = 2$ 的组合。表 3 给出了启发因子和能见度启发因子对收敛速度与搜索能力的影响。

3.4 信息素总量 Q 和信息素初始浓度的设置

Q 是蚂蚁遍历所有的点所留下的信息素总量。Q 越大,在路径上留下的信息素越多,当路线接近真解时,信息素浓度更大,反过来影响其它的蚂蚁,进而加快算法收敛速度。

表 3 α, β 对收敛速度与搜索能力的影响

α	0.1	0.5	1	1	1.5
β	0.1	5	1	2	2
最短路径(m)	477	284	286	284	284
搜索循环次数	35	70	47	82	24

在其他参数保持不变的情况下,分别选取 3 个不同的 Q 值($Q \in \{10,50,100\}$)。下面给出运行得到的结果。可以看出,总信息量 Q 对 antcycle 模型蚁群算法的性能没有明显影响,本文选择 $Q = 50$,信息素初始浓度为 1。信息素初始浓度也即是各路径上信息素量最少时的浓度,为了防止算法停滞,将各条寻优路径上残留的信息素限制在 $[\tau_{\min}, \tau_{\max}]$[12] 之内。表 4 给出了信息素总量对收敛速度和搜索能力的影响。

表 4 信息素总量 Q 对收敛速度和搜索能力的影响

Q	10	50	100
最短路径(m)	291	286	288
搜索循环次数	37	33	33

4 蚁群算法求解拣货路径问题

4.1 一单一车的情况

随机产生 10 个订单,每个订单有 20 个待捡货点(见附录一)。分别用蚁群算法、S 形启发算法、传统穿越策略和遗传算法计算拣货路线,再将结果与传统穿越策略进行比较。

为了计算方便,设定了以下仓库参数:

(1) 货架的长度 D_1 =1m;

(2) 货架的宽度 D_2 =1m;

(3) 巷道的宽度 D_3 =1m;

(4) 过道的宽度 D_4 =2m。

由于该仓库中巷道宽度只有 1m,拣货人员进行左右两则拣取的行走的距离较短,在计算总路径时忽略不计。

4.1.1 算法设计

在仓库拣货过程中,基于蚁群算法的具体实现步骤如下。

步骤 1:初始化参数,本文中订单中包含 20 个储位,即 $n = 20$,所以根据前文中对参数选择的讨论,本文选用蚂蚁数量 $m = 15$,启发因子 $\alpha = 1.5$,能见度启发因子 $\beta = 2$,信息素总量 $Q = 50$,信息素初始浓度 $C = 1$。$NC = 0$ (NC 为迭代步数或搜索次数);各 τ_{ij} 和 $\Delta\tau_{ij}$ 的初始化;将 m 只蚂蚁置于 n 个顶点上。

步骤 2:将各个蚂蚁的初始位置置于当前解集当中;对每只蚂蚁 $k(k = 1,2,3,\cdots,m)$,按照概率 p_{ij}^k 转移到下一个节点 j;将顶点 j 置于当前解集中。

步骤 3:计算各蚂蚁的路径长度 $L_k(k = 1,2,3,\cdots,m)$;记录当前最好的解。

步骤 4:按更新方程修改轨迹强度。

步骤 5:对各弧边 (i,j),置 $\Delta\tau_{ij} = 0$,$NC = NC + 1$。

步骤 6:若 NC 小于 NC _max(最大迭代次数)且无退化行为(即解都是相同的),则转步骤 2。

步骤 7:输出最优解。

4.1.2 算法流程图

算法流程如图 2 所示。

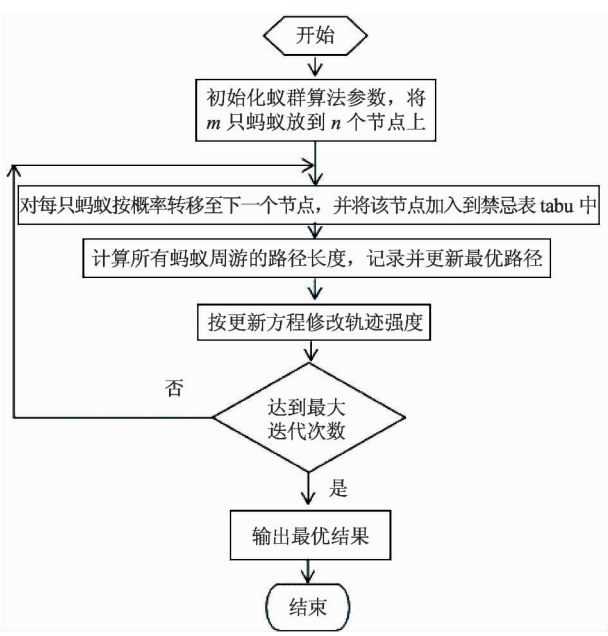

图 2 蚁群算法流程图

4.1.3 优化结果

以表 5 中订单 1 为例的三种方法求解结果如图 3至图 5 所示。

表 5　优化结果比较

订单	穿越策略距离(m)	S 行启发算法			蚁群算法		
		距离(m)	减少距离(m)	减少比例(%)	距离(m)	减少距离(m)	减少比例(%)
1	484	358	126	26.03	284	200	41.32
2	484	352	132	27.27	308	169	36.36
3	484	338	146	30.17	312	207	44.44
4	484	324	160	33.06	290	175	35.54
5	484	344	140	28.93	292	183	40.08
6	484	340	144	29.75	314	116	35.12
7	482	360	122	25.31	302	90	37.34
8	482	376	106	21.99	308	157	36.10
9	398	324	74	18.59	276	200	30.65
10	398	316	82	20.60	302	171	24.12

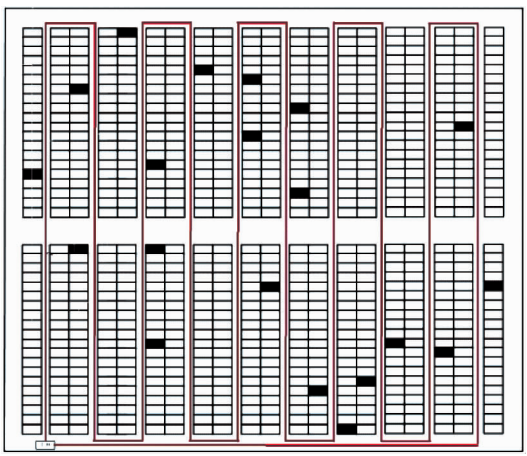

图 3　传统穿越策略路径示意图

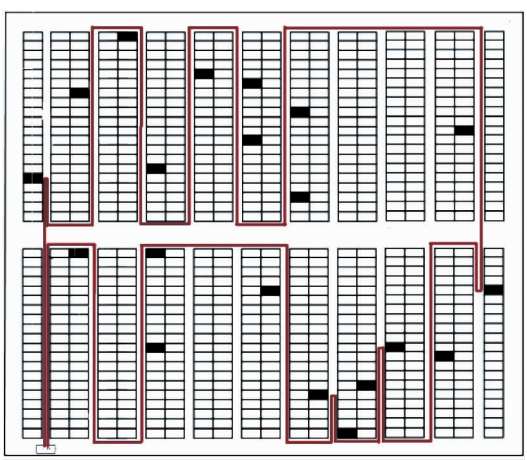

图 4　S 形穿越路径示意图

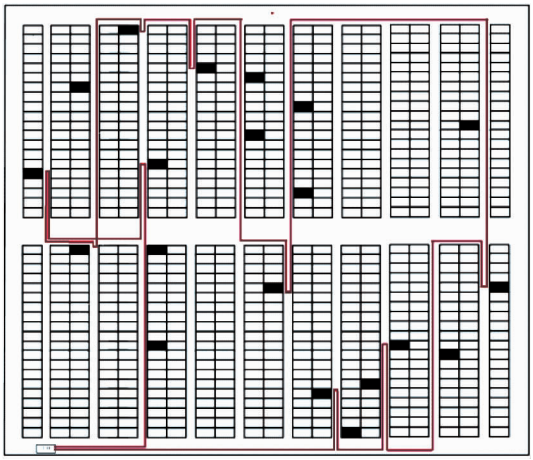

图 5　蚁群算法路径示意图

4.2　一单多车的情况

由于 S 形算法难以实现不同车次之间的优化分组,本文将用蚁群算法进行求解。随机生成 100 个储位的订单进行仿真试验。考虑载重的限制,在原有的储位表示方法加上一个数据表示该储位上单个货物的重量,单位是 kg。例如,{3-1-18-3.5}表示第三巷道左边第 18 行的待捡货物重量是3.5kg。

4.2.1　算法设计

一单多车增加了小车载重量的限制条件,当小车所载货物达到小车上限时,小车即需要返回出发点,每辆小车的取货路径构成一个小回路,模型目标即要求所有小回路的路径之和最短。算法设计如下:

步骤 1:初始化参数,参数 α,β,ρ,Q,C 的设置同一单一车的情况,因为随机生成了 100 个储位的订单,此时仍设置 $m = \frac{3}{4}n$,即此时蚂蚁的数量为 75。$t = 0, NC = 0$。

步骤 2:将 m 只蚂蚁分配到 n 个节点上,并将各自的位置 v_j 加入到各自的禁忌表当中。

步骤 3:根据概率选择公式计算蚂蚁 k 的转移概率 $p^i_{v_j v_{j+1}}$,根据转移概率选择下一个节点 v_{j+1},并将 v_{j+1} 加入到蚂蚁 k 的禁忌表中;$t = t + 1$。

步骤 4:判断小车的载货量是否达到上限,如果 $\sum_{j=1}^{g_i} G^i_{v_j} > W$,则小车立即返回源点,并记录蚂蚁 k 第 i 次回路的路径长度;否则继续步骤 3。

步骤 5：蚂蚁 k 走完所有储位后，按更新方程更新修改轨迹强度，记录蚂蚁 k 行走的总长度，并更新当前的最优路径。

步骤 6：如果 $NC \leqslant NC\_\max$，则 $NC = NC + 1$，清空禁忌表，继续步骤 2；否则输出最优结果。

4.2.2 算法流程图

算法流程见图 6。

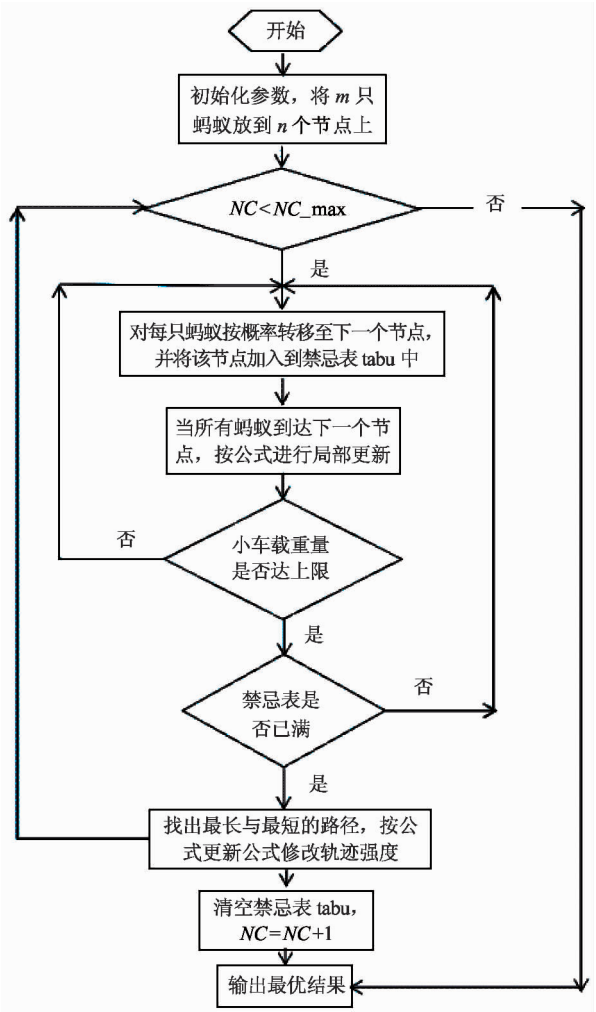

图 6 一单多车算法流程图

4.2.3 优化结果

随机选取了一个含有 100 个待捡货点的订单，订单待捡货物总量为253.9kg，限制推车的载重为 30kg。用蚁群算法搜索最优解，采用 Matlab 实现算法，共需要 9 次车才能捡完订单中的全部货物，总路径长度为 859m，优化路径如下：

第 1 次车：I/O-[1 2 4 1.02]-[1 2 4 3.19]-[1 2 5 3.91]-[1 1 17 3.24]-[1 2 19 2.86]-[1 2 22 2.11]-[1 2 25 2.78]-[1 1 27 1.95]-[2 1 20 2.35]-[2 2 9 1.38]-[2 2 18 3.05]-[2 1 16 1.2]-I/O

第 2 次车：I/O-[2 2 14 1.53]-[2 2 24 3.28]-[2 1 26 3.18]-[2 2 28 3.06]-[2 2 30 1.00]-[3 2 24 2.58]-[3 1 19 1.21]-[3 1 18 1.83]-[3 2 19 2.82]-[3 1 16 2.95]-[3 2 14 2.41]-[3 1 11 3.77]-I/O

第 3 次车：I/O-[3 2 9 3.75]-[3 1 7 1.51]-[4 2 1 2.7]-[4 2 1 3.11]-[4 1 10 2.93]-[4 2 13 2.64]-[4 2 13 3.96]-[4 2 15 3.07]-[4 1 16 2.14]-[4 1 16 1.63]-[4 1 24 1.2]-I/O

第 4 次车：I/O-[5 2 22 2.41]-[5 2 23 2.78]-[5 1 24 3.36]-[5 2 25 3.81]-[5 1 27 2.07]-[5 1 27 3.46]-[5 2 20 2.39]-[5 1 18 3.59]-[5 2 17 3.29]-[5 1 12 2.41]-I/O

第 5 次车：I/O-[5 2 3 3.61]-[6 1 1 1.16]-[6 1 6 2.71]-[6 2 14 1.07]-[6 1 16 1.46]-[6 1 17 1.04]-[6 1 20 3.86]-[6 1 23 2.69]-[6 1 23 3.31]-[6 1 26 3.67]-[6 1 28 2.89]-[6 2 30 1.67]-I/O

第 6 次车：I/O-[5 1 20 3.68]-[5 1 20 2.69]-[5 1 20 1.38]-[5 2 19 1.82]-[5 2 19 2.01]-[6 1 30 1.41]-[7 1 27 3.03]-[8 2 22 2.81]-[8 2 28 2.69]-[8 2 29 3.06]-[8 2 30 2.16]-[8 1 29 2.40]-I/O

第 7 次车：I/O-[8 1 18 3.85]-[9 1 20 3.21]-[9 1 19 2.15]-[9 2 15 2.15]-[9 2 14 2.37]-[9 1 12 3.08]-[9 2 11 2.19]-[9 2 1 1.16]-[8 2 1 2.83]-[8 2 4 2.43]-[7 2 2 3.05]-I/O

第 8 次车：I/O-[7 1 6 2.75]-[7 2 8 3.47]-[7 2 10 1.76]-[7 2 15 3.41]-[10 1 20 2.83]-[10 1 21 3.59]-[10 2 22 2.71]-[10 2 29 2.23]-[9 1 24 1.53]-[9 1 26 1.33]-[9 1 30 3.1]-I/O

第 9 次车：I/O-[10 2 8 2.91]-[10 1 7 2.13]-[10 1 8 3.30]-[10 1 8 1.90]-[10 1 5 1.2]-[10 2 11 2.27]-[10 2 12 1.83]-[10 1 13 2.31]-[10 1 18 3.72]-I/O

5 结 论

合理地选择拣货路径对于降低物流配送成本有

重大作用。针对问题的特点,建立了数学模型,设计了蚁群算法对其求解,并与多种方法对比了优化结果,结果显示蚁群算法能令人满意地解决问题。针对一单多车的情况,本文修改了一单一车时算法部分编码,对一单多车的仓库捡货情况进行了优化求解,两种情况表明蚁群算法能够很好地解决仓库拣货路径优化问题。

参考文献

[1] Goetschalckx M, Ratliff H D. Order picking in an aisle. *IIE Transactions*,1988,20(1):53-62

[2] 熊芳敏,岑宇森,曾碧卿. 运用蚁群算法解决物流中心拣货路径问题. 华南师范大学学报(自然科学版),2010,5: 50-54

[3] Ratliff H D, Rosenthal A S. Order picking in a rectangular warehouse: a solvable case of traveling salesman problem. *Oprations Research*,1983,31(3):507-521

[4] Hwang H, Oh Y H, Lee Y K. An evaluation of routing policies fororder-picking operations in low-level picker-to-part system. *International Journal of Production Research*,2004,42(18):3873-3889

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

[6] Daniels R L, Rummel J L, Schantz R. A model for warehouse order picking. *European Journal of Operational Research*,1998,105(1):1-17

[7] 宛剑业,刘卫博,张飞超. 基于微遗传算法的仓储布局优化方法研究. 物流工程与管理, 2016,38(1): 39-40

[8] 康莹. 调整型 SLP 方法在某物流企业仓库布局优化中的应用. 见:2011 年信息技术、服务科学与工程管理国际学术会议. 北京:中国, 2011. 1396-1400

[9] 雷娟娟. 基于蚁群算法的仓库拣货路径优化研究:[硕士学位论文]. 合肥:合肥工业大学管理学院,2010. 6-10

[10] 张毅,梁艳春. 蚁群算法中求解参数最优选择分析. 计算机应用研究, 2007, 24(8): 70-71

[11] 王军. 蚁群算法求解 TSP 时参数设置的研究. 科学技术与工程, 2007, 7(17): 4501-4503

[12] 段海滨,王道波,朱家强等. 蚁群算法原理及其应用研究的进展. 控制与决策,2004,19(12): 1321-1326

Picking routing optimization for in 2-block warehouses based on ant colony algorithm

Liu Jiansheng, Xiong Feng, Chen Jingkun, Xiong Junxing

(School of Mechanical and Electronic Engineering, Nanchang University, Nanchang 330031)

Abstract

To improve the efficency of goods picking, the key link of the hole operation of a modern warehouse, a study of picking routing optimization was conducted. According to the characteristics of the picking routing problem of a 2-block warehouse, the ant colony algorithm was applied to optimization of the picking routing problem, and an effective ant colony algorithm for optimized picking routing was achieved. Its performance was compared with the traditional passing strategy, S-shape algorithm and genetic algorithm by simulation. The simulation results show that the ant colony algorithm can reduce the routing distance significantly, with the better practicality and effectiveness.

Key words: logistics, picking, vehicle routing, ant colony algorithm

附录一

订单一：

[1 1 25][2 1 20][2 1 34][3 1 40][3 2 10]
[3 2 20][3 2 26][4 2 36][5 2 35][6 1 16]
[6 2 23][6 2 32][7 2 1][8 1 6][8 2 10]
[10 1 30][9 2 9][10 2 16][5 2 29][7 1 5]

订单二：

[1 1 20][1 1 15][1 2 32][2 1 21][3 1 26]
[4 1 10][4 2 40][5 1 6][5 2 10][5 2 29]
[6 2 11][7 1 4][7 2 36][8 2 6][9 1 16]
[9 1 24][9 2 5][10 1 12][10 2 24][8 1 22]

订单三：

[1 1 20][1 2 15][1 2 32][2 1 11][3 1 16]
[4 1 26][4 2 30][5 1 6][5 2 10][5 2 29]
[6 2 11][7 1 4][7 2 36][8 1 22][8 2 6]
[9 1 7][9 1 24][9 2 26][10 1 5][10 2 24]

订单四：

[1 1 20][2 2 15][2 2 32][2 2 40][3 1 16]
[4 1 26][4 2 30][5 1 6][5 2 10][6 1 29]
[6 2 11][7 1 4][7 2 36][8 1 22][8 2 6]
[8 2 21][9 1 12][9 1 26][9 2 5][10 2 24]

订单五：

[1 1 12][1 1 32][1 2 21][2 2 5][3 1 16]
[3 2 26][4 2 30][5 1 6][6 1 10][6 1 29]
[6 2 11][7 1 4][7 2 36][8 1 22][8 2 6]
[8 2 21][9 1 12][9 1 26][9 2 5][10 2 24]

订单六：

[1 1 12][1 1 32][2 1 21][2 2 5][3 1 16]
[3 2 26][4 1 30][4 2 6][5 1 10][5 1 29]
[6 2 11][6 2 40][7 2 17][7 2 25][8 2 6]
[8 2 21][9 1 12][9 1 26][9 2 5][10 2 24]

订单七：

[1 1 12][1 1 32][1 2 5][1 2 27][3 1 16]
[3 2 26][4 1 30][4 2 6][5 1 10][5 1 29]
[6 2 11][6 2 40][7 2 17][7 2 25][8 2 6]
[8 2 21][9 1 12][9 1 26][9 2 5][10 2 24]

订单八：

[1 1 12][1 1 32][2 2 5][2 2 27][3 1 16]
[3 2 26][4 1 30][4 2 6][5 1 10][5 1 29]
[5 2 40][7 1 35][7 2 17][7 2 25][8 2 6]
[8 2 21][9 1 12][9 1 26][9 2 5][10 2 24]

订单九：

[1 1 12][1 1 32][1 2 5][1 2 27][4 1 16]
[4 1 30][4 2 6][4 2 26][5 1 10][5 1 29]
[5 2 40][6 1 35][7 1 17][7 2 25][8 2 6]
[8 2 21][9 1 12][9 1 26][9 2 5][10 2 24]

订单十：

[1 1 12][1 1 32][2 1 5][2 2 27][4 1 16]
[4 1 30][4 2 6][4 2 26][5 1 10][5 1 29]
[5 2 40][7 1 35][7 2 17][7 2 25][8 1 6]
[8 1 21][8 2 12][9 1 26][10 1 5][10 2 24]