文章编号: 1001-0920(2005)05-0562-05

基于自适应蚁群算法的车辆路径问题研究

刘志硕[1], 申金升[2], 柴跃廷[1]
(1. 清华大学 自动化系, 北京 100084; 2. 北京交通大学 系统工程研究所, 北京 100044)

摘 要: 车辆路径问题(VRP)是物流研究领域中一个具有重要理论和现实意义的问题. 蚁群算法是一种新型的模拟进化算法, 可以很好地解决旅行商问题(TSP). 在分析VRP与TSP区别的基础上, 构造了求解VRP的自适应蚁群算法. 指出可行解问题是蚁群算法的关键问题, 并重点对该问题进行了研究, 提出了近似解可行化等解决策略. 实验结果表明, 自适应蚁群算法性能优良, 能够有效地求解VRP问题.

关键词: 车辆路径问题; 旅行商问题; 自适应蚁群算法; 近似解可行化; 吸引力
中图分类号: TP301 **文献标识码**: A

Vehicle routing problem based on an adaptive ant colony algorithm

LIU Zhi-shuo[1], *SHEN Jin-sheng*[2], *CHAI Yue-ting*[1]
(1. Department of Automation of Tsinghua University, Beijing 100084, China; 2. Insititute of System Engineering of Beijing Jiaotong University, Beijing 100044, China. Correspondent: LIU Zhi-shuo, E-mail: l-z-s@sina.com.cn)

Abstract: On the basis of analyzing the differences between vehicle routing problem (VRP) and traveling salesman problem (TSP), an adaptive ant colony algorithm (AACA) is proposed to solve VRP, which is improved from basic ACA by means of integrating C-W algorithm and introducing the adaptive ant attraction of arc in order to decrease computing time and avoid stagnation behavior. Moreover, how to acquire feasible solution is a key problem in this algorithm, and three relative resolutions such as the feasibility process of approximate solution are presented. The computational experiments show that the AACA is feasible and valid for VRP.

Key words: vehicle routing problem; traveling salesman problem; adaptive ant colony algorithm; feasibility process of approximate solution; ant attraction of arc

1 引 言

车辆路径问题(VRP)是物流研究领域中一个具有十分重要理论和现实意义的问题[1]. VRP问题是强NP难题, 寻找到高效的精确算法的可能性不大, 所以寻找近似算法是必要和现实的. 蚁群算法[2]是一种源于自然界中生物世界的新的仿生类随机型搜索算法, 通过其内在的搜索机制, 已在一系列困难的组合优化问题求解中取得了成效, 本文尝试采用该算法对VRP问题进行求解.

蚁群算法是由Dorigo等于1991年首先提出的. 此后, Dorigo等又应用该算法求解了旅行商问题(TSP), 分配问题[3], Job-Shop调度问题[4], 取得了较好的结果. 受其影响, 蚁群算法逐渐引起了其他研究者的注意, 将蚁群算法的思想应用到各自研究领域, 取得了大量的研究成果. 比如Costa提出了一种求解分配类型问题的一般模型, 并用于研究图着色问题[5]. 为了克服基本蚁群算法存在的不足, 人们对其进行了各种改进, 以期提高搜索效率、避免过早停滞. 其中主要有Dorigo提出的Ant-Q System[6], Stutzle提出的MMAS[7]和Gambardella提出的混合型蚁群算法HAS[8]等. 我国对蚁群算法的研究尚处于起步阶段, 目前已经取得了一些研究成果[9].

收稿日期: 2004-06-24; **修回日期**: 2004-10-12
基金项目: 国家"十五"科技攻关项目(2001BA205A08-04).
作者简介: 刘志硕(1977—), 男, 湖南安仁人, 博士后, 从事人工智能、智能交通系统和智能物流系统的研究;
申金升(1966—), 男, 河北保定人, 教授, 博士生导师, 从事系统工程、环境工程等研究.

2　自适应蚁群算法设计

2.1　算法总体思路

TSP 是VRP 的基本问题, 二者既相似又存在诸多差别, 因此在设计VRP 的蚁群算法时, 既要注意吸收前人设计TSP 蚁群算法的经验, 又要充分考虑VPR 的具体要求. 此外, 由于VRP 的复杂程度远高于TSP, 在设计蚁群算法时, 研究如何减少算法计算时间和提高搜索效率的重要性就显得尤为突出. 鉴于此, 本文蚁群算法的设计思路是: 以TSP 蚁群算法为基础, 充分考虑VRP 的具体要求, 对算法的选择机制、更新机制以及协调机制作进一步改进, 引入自适应的转移策略和信息素更新策略, 并融合节约法, 以克服蚁群算法计算时间长、易出现停滞的缺陷.

2.2　VRP 与TSP 蚁群算法的区别

应用蚁群算法求解VRP 与TSP 的不同之处主要体现在以下3 方面:

(1) 子路径构造过程的区别

在TSP 中, 每只蚂蚁均要经过所有结点, 而在VRP 中, 每只蚂蚁并不需要遍历所有结点. 因此, 在VRP 中的每次迭代中, 每只蚂蚁移动次数是不确定的, 只能将是否已回到原点作为路径构造完成的标志, 即已走点集 $tabu_k$ 中包括两个相同的结点. 并且在VRP 中将路径构造过程分为如下两个阶段:

1) 对于初始位置不是配送中心(0 点)的蚂蚁而言, 第1 个阶段就是由初始结点出发寻找配送中心的过程, 称为"过程Ⅰ". 在此过程中, 可行点集 $allowed_k$ 中应包含0 点, 而不能包含初始结点. 在到达0 点后, 接下来的任务就是从0 点出发, 找寻一条返回初始结点的路径, 称该过程为"过程Ⅱ". 此过程中, $allowed_k$ 中不能包含0 点, 而应包含初始结点.

2) 对于初始结点为0 点的蚂蚁而言, 其"过程Ⅰ"就是由0 点移到任意其他结点的过程, "过程Ⅱ"就是由其他结点回到0 点的过程, 因此在"过程Ⅰ"中 $allowed_k$ 不能包含0 点, 而在"过程Ⅱ"中必须包含0 点.

(2) $allowed_k$ 的区别

$allowed_k$ 的确定是蚂蚁构造路径的一个十分关键的问题, 是蚂蚁进行选择和转移的前提和基础. 在TSP 中, 蚂蚁转移时只需考虑路径的距离和信息浓度即可, 但在VRP 中, 蚂蚁转移时不但要考虑上述因素, 还需要考虑车辆容量的限制. 这一差异在算法中的具体体现就是 $allowed_k$ 的确定问题. 在VRP 中, 首先 $allowed_k$ 中的结点必须满足车辆容量的硬约束, 即

$$allowed_k = \{v_j \mid v_j \in V, d_j + W_k < q\},$$

其中: v_j 表示结点 j, d_j 表示 j 点的配送量, W_k 表示蚂蚁 k 已负担的重量; 其次, 对应蚂蚁所处的不同过程, 需按前述方式对0 点和初始点进行特殊处理.

(3) 可行解结构的区别

在求解TSP 问题中, 每只蚂蚁所构造出来的路径均是一个可行解, 但在VRP 问题中, 每只蚂蚁所构造的回路仅是可行解的"零部件", 各蚂蚁所构造的回路可能能够组合成一些可行解, 也可能一个可行解都得不到. 因此, 研究如何设计算法以尽量避免无可行解的出现以及如何获取可行解是蚁群算法的关键问题.

2.3　算法基本规则设计

(1) 转移规则

借鉴Dorigo 的Ant-Q 算法思想, 采用确定性选择和随机选择相结合的选择策略, 其具体形式可参见文献[6].

(2) 信息素更新规则

定义1(吸引力)　设经过结点 i 的蚂蚁数量为 R, 经过有向边 (i, j) 的蚂蚁数量为 r, 则值 r/R 称为边 (i, j) 的蚂蚁吸引力.

在进行信息素局部更新时, 若每次施放的信息素数量 Q 为常量, 则 (i, j) 的蚂蚁吸引力越大, 经过边 (i, j) 的蚂蚁数量就越多, 从而局部更新的次数越多. 久而久之, 会导致边之间的信息素数量差距过大, 限制了算法搜索的全局性. Q 值的大小也会影响算法的搜索效率, Q 值过大会使算法易收敛于局部极小值, 过小又会影响算法的收敛速度. 因此, 随着算法搜索状态的变化, Q 值应不断调整. 调整的原则是, 路径的吸引力越大, $Q(t)$ 越小. 不妨设 $Q(t) = Q(1 - r/R)$, 并且若 $R = 0, Q(t) = Q$.

假设第 k 只蚂蚁在第 q 次迭代周期中的第 f 次转移时经过边 (i, j), 在此之前共有 R_k 只蚂蚁经过 i 点, 其中有 r_k 只蚂蚁选择了边 (i, j). 算法的全局更新规则与TSP 的类似, 而局部更新规则设计如下:

$$\tau_{ij}^{new} = \rho_1 \tau_{ij}^{old} + \Delta\tau_{ij}, \quad \forall i, j, \ i \neq j; \tag{1}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}. \tag{2}$$

其中 τ_{ij}^{old} 和 τ_{ij}^{new} 分别表示 (i, j) 上的原信息素浓度以及更新后的浓度. (i, j) 上信息素局部更新量为

$$\Delta\tau_{ij}^{k} = \begin{cases} Q_1(1 - r_k/R_k), \\ \quad \text{ant } k \text{ move to } i \text{ from } j; \\ 0, \quad \text{else.} \end{cases} \tag{3}$$

2.4　可行解问题的研究

在VRP 问题中, 每只蚂蚁所构造的回路仅是可行解的一个组成部分, 各蚂蚁所构造的回路可能能够组成一些可行解, 但也可能一个可行解都得不到

若经常得不到可行解, 则会造成大量的计算时间浪费. 因此, 研究如何避免无可行解是算法设计的一个关键问题. 可采取以下策略: 1) 大蚂蚁数策略. 增加算法的蚂蚁数量M, 扩大组合范围, 从而增加可行解产生的可能性. 2) 蚂蚁初始分布均匀策略. 在每次迭代前, 将蚂蚁随机均匀分布于各个结点, 从而增加发现可行解的机会. 3) 近似解可行化策略. 前两种策略的目的都是为了提高各蚂蚁所构造的子回路组合成可行解的可能性, 是一些针对无可行解的"事前"预防性措施. 然而, 当问题规模较大时, 会有大量的蚂蚁所找到的回路是相同的, 使得事实上真正起作用的蚂蚁数大大减少, 严重影响产生可行解的可能性. 针对这个问题, 提出近似解可行化策略. 所谓近似解可行化, 是指在找不到可行解的情况下, 选择一些与可行解接近的非可行解作为问题的近似解, 再按照某些规则将近似解转变成可行解的过程. 由定义可以看出, 近似解可行化策略实质上是一种"事后"性质的治理措施. 显然, 采取这种措施后, 一般可确保得到问题的至少一个可行解. 下面对近似解可行化策略进行深入研究.

2.4.1 近似解的获取

记$\mathrm{CNT}(\mathrm{tabu}_k)$为$\mathrm{tabu}_k$中的结点数, 如$\mathrm{tabu}_1 = \{0, 1, 3, 0\}$, 则$\mathrm{CNT}(\mathrm{tabu}_1) = 4$.

定义 2(子回路互异) 若$\forall v_i \neq \forall v_j, v_i \in \mathrm{tabu}_k, v_j \in \mathrm{tabu}_{k'}, i \neq 0, j \neq 0, k \neq k'$, 则称子回路$\mathrm{tabu}_k$和$\mathrm{tabu}_{k'}$互异, 记为$\mathrm{tabu}_k \cap \mathrm{tabu}_{k'} = \phi$, 意即两个子回路除 0 点外, 无其他相同结点.

命题 1 任意数量的子回路所构成的集合(记为S)必属于可行解、未满非可行解、溢出非可行解以及混合非可行解等 4 种情形之一.

可行解等概念的定义如下:

定义 3(可行解) 所谓可行解指的是这样一种集合, 集合的元素是两两互异的子回路, 并包括所有结点, 即

$$S_1 = \{\mathrm{tabu}_r \mid r = 1, 2, \cdots, R, \forall \mathrm{tabu}_r \cap \forall \mathrm{tabu}_{r'} = \phi, \sum_{r=1}^{R} \mathrm{CNT}(\mathrm{tabu}_r) = N + 2R - 1\}$$

定义 4(未满非可行解)

$$S_2 = \{\mathrm{tabu}_r \mid r = 1, 2, \cdots, R, \forall \mathrm{tabu}_r \cap \forall \mathrm{tabu}_{r'} = \phi, r \neq r', \sum_{r=1}^{R} \mathrm{CNT}(\mathrm{tabu}_r) < N + 2R - 1\}$$

未满非可行解由两两互异的子回路组成, 并存在一个或多个结点未在该解的线路中. 记$\mathrm{Lg}(S_2)$为S_2中已包含的结点数, 显然$\mathrm{Lg}(S_2)$越大, 未满非可行解越接近可行解. 设$\overline{S_2}$为未包含的结点集, 则显然有$\mathrm{Lg}(S_2) + \mathrm{Lg}(\overline{S_2}) = N$. 本文将那些在所有未满非可行解中的具有最大$\mathrm{Lg}(S_2)$的未满非可行解称为最大未满非可行解.

定义 5(溢出非可行解)

$$S_3 = \{\mathrm{tabu}_r \mid r = 1, 2, \cdots, R, \exists \mathrm{tabu}_r \cap \mathrm{tabu}_{r'} = \phi, r \neq r', \sum_{r=1}^{R} \mathrm{CNT}(\mathrm{tabu}_r) > N + 2R - 1\}$$

溢出非可行解虽然包括了所有结点, 但它存在某些子回路不互异, 即存在除 0 点外的公共结点. 记$L_p(S_3)$为溢出非可行解S_3中公共结点数. 公共结点数的计算方法为: 比较任意两个不互异的子回路, 计算其除 0 点外的相同结点数量, 公共结点数为所有这些结点数量的累加. 有

$$\sum_{r=1}^{R} \mathrm{CNT}(\mathrm{tabu}_r) = N + 2R - 1 + L_p(S_3),$$

因此$L_p(S_3)$越小, 溢出非可行解越接近可行解. 显然, 当$L_p(S_3) = 0$时, 溢出非可行解就变为可行解了. 将那些在所有溢出非可行解中具有最小$L_p(S_3)$的溢出非可行解称为最小溢出非可行解.

定义 6(混合非可行解) 混合非可行解是指未包括所有结点, 并存在某些子回路不互异的解, 即

$$S_4 = \{\mathrm{tabu}_r \mid r = 1, 2, \cdots, R, \exists \mathrm{tabu}_r \cap \mathrm{tabu}_{r''} = \phi, \sum_{r=1}^{R} \mathrm{CNT}(\mathrm{tabu}_r) < N + 2R - 1\}$$

定义 7(近似解) 将所有最大未满非可行解和最小溢出非可行解称为近似解. 按该定义, 虽然当算法不存在可行解时, 从理论上并不能保证一定存在近似解, 但实践中同时不存在溢出非可行解和未满非可行解的情况可能性极小, 因此几乎不对近似解的获取产生影响.

2.4.2 近似解的可行化

在获得近似解以后, 接下来的问题就是如何将近似解转变为可行解. 对于最大未满非可行解, 可行化策略就是采用某种方法将未在线路中的点插入到现有线路中. 本文采用节约算法[10], 该算法的基本思想是首先将各点单独与源点 0 相连, 构成 1 条仅含一个点的线路; 然后计算将点i与j在满足车辆容量约束的基础上连接在同一条线路上的费用节约值

$$s(i, j) = s(j, i) = c_{i0} + c_{j0} - c_{ij}. \quad (4)$$

$s(i, j)$越大, 说明将i与j连接在一起时总路程减少越多. 设计近似解可行化的节约法步骤如下:

Step 1: 将 $\overline{S_2}$ 集中的结点单独与 0 点相连, 构成一些仅含一个点的线路, 并将它们添加到 S_2 中; 然后计算 $s(i,j)$, 令

$$M = \{s(i,j) \mid v_i \in \overline{S_2}, \forall j, s(i,j) > 0\};$$

Step 2: 在 M 内按 $s(i,j)$ 从大到小的顺序排列;

Step 3: 若 $M = \phi$, 则算法终止, 此时有 $\overline{S_2} = \phi$, 否则对第 1 项 $s(i,j)$, 考察对应的 (i,j), 若点 j 不在已构成的线路上, 即 $v_j \in \overline{S_2}$, 或点 j 在已构成的线路上, 即 $v_j \in S_2$, 但不是线路的内点, 转 Step 4, 否则转 Step 7;

Step 4: 考察点 i 与 j 连接后的线路上总货物量 Q, 若 $Q \leq q$, 转 Step 5, 否则转 Step 7;

Step 5: 连接点 i 与 j, 修改相应的 tabu_k 以及 S_2, 转 Step 6;

Step 6: 从 M 中去掉所有以 i 为起始点的 $s(i,j)$, 且若 $v_j \in \overline{S_2}$, 则从 M 中去掉所有以 j 为起始点的 $s(j,k)$, 转 Step 3;

Step 7: 令 $M := M - s(i,j)$, 转 Step 3.

对于最小溢出非可行解, 也可采用节约法将其转变为可行解, 只是在应用该方法之前, 需对其公共结点进行"擦除"处理, 即先去掉所有线路中的公共结点, 将溢出非可行解转换成未满非可行解, 再应用节约法将公共结点插入到线路中.

2.5　算法步骤

Step 1: 初始化各参数, 输入基础数据, 最优解 $L^* = \text{MAX\_RLT}$, 计数器 $q = 0$.

Step 2: 将 M 个蚂蚁随机均匀地放到 N 个结点上, 得到结点 i 的蚂蚁集 S_i 和蚂蚁数 b_i, 初始化 tabu_k 以及 allowed_k, $l = 0$ (已完成任务蚂蚁数); 初始点为 0 的蚂蚁的过程变量 $\text{Pro}[k] = 1$, 初始点为非 0 点的蚂蚁的过程变量 $\text{Pro}[k] = 2$.

Step 3: 在结点 i 取蚂蚁 k, 判断其初始结点. 若为 0 点, 则按转移规则确定结点 j, 更新 tabu_k, S_i, b_i. 且若 $\text{Pro}[k] = 1$, 则 $\text{Pro}[k] = 2$, 转 Step 4; 而当 $\text{Pro}[k] = 2$ 时, 若 j 点为初始结点, 则 $l++$, 转 Step 3, 否则转 Step 4. 若为非 0 点, 则按转移规则确定转移结点 j, 更新 tabu_k, S_i, b_i. 并且当 $\text{Pro}[k] = 1$ 时, 若 j 点为 0, 则 $\text{Pro}[k] = 2$, 转 Step 3; 当 $\text{Pro}[k] = 2$ 时, 若 j 点为初始结点, 则 $l++$, 转 Step 3, 否则转 Step 4;

Step 4: 更新 S_j, 重复 Step 3 和 Step 4, 直到 $b_i = 0$.

Step 5: 在所有蚂蚁都移动一次后, 按局部更新规则进行信息素的更新.

Step 6: 更新所有结点的 b_i, 若所有结点上蚂蚁数量 $b_i = 0$ 或 $l = M$, 转 Step 7, 否则转 Step 3.

Step 7: 由 tabu_k 生成路径集 $L = \{L_1, L_2, \cdots, L_M\}$, 寻找可行解, 得到可行解集 $A = \{A_1, A_2, \cdots, A_p\}$; 若未发现可行解, 则采取近似解可行化策略, 转 Step 8.

Step 8: 计算本次搜索到的最优路径 $L^*(q)$, 并得到岂今为止的最优路径, $L^* = \min\{L^*(q), L^*\}$, 按全局更新规则进行信息素更新.

Step 9: 判断 q 是否等于最大迭代次数 n_c, 若是则算法终止, 输出 L^*; 否则, q^{++}, 转 Step 2.

3　实验及分析

设有 19 个客户随机分布于边长为 10 km 的正方形区域内. 配送中心位于区域正中央, 其坐标为 (0, 0). 各客户的需求由计算机随机产生, 车辆载重量为 9 t. 基础数据如表 1 所示.

表 1　实验基础数据

客户编号	0	1	2	3	4	5	6	7	8	9
横坐标/km	0	0	0	-2	-3	3	-4	-4	1	1
纵坐标/km	0	-1	3	-2	-3	-1	0	-1	-2	-1
配送量/t	0	1.5	1.8	2.0	0.8	1.5	1.0	2.5	3.0	1.7
客户编号	10	11	12	13	14	15	16	17	18	19
横坐标/km	1	3	-3	2	1	2	2	1	-3	-1
纵坐标/km	3	4	0	0	-3	-1	1	-4	2	-1
配送量/t	0.6	0.2	2.4	1.9	2.0	0.7	0.5	2.2	3.1	0.1

运行参数为 $M = 60$, $n_c = 50$, $\tau_{ij} = 10$, $\alpha = 1$, $\beta = 1$, $\rho_1 = 0.85$, $\rho_2 = 0.95$, $Q_1 = 10$, $Q_2 = 50$, $Q_3 = 100$. 运行 10 次, 结果如表 2 所示.

本文以第 2 次运行过程为例, 得到算法进化过程如图 1 所示.

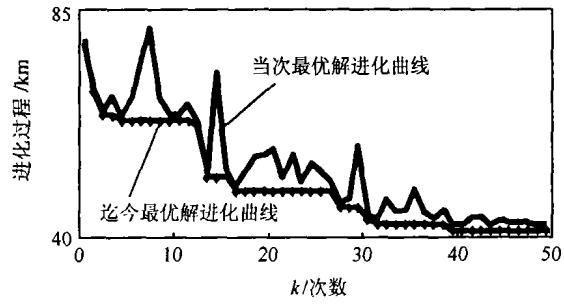

图 1　蚁群算法进化曲线图

从表 2 可以看出, 采用本文所设计的蚁群算法的 10 次求解, 都得到了质量很高的解, 所找到的最好解为 41.86 km, 对应的 4 条配送路径分别为路线 1: 0-18-0, 路线 2: 0-1-17-14-8-0, 路线 3: 0-12-6-7-4-3-19-0, 路线 4: 0-2-10-11-16-7-5-15-9-0. 蚁群算法的计算结果也很稳定, 10 次求解中, 最差解的配送里程仅比最好解多 5.8%. 从计算效率看, 10 次求解的平均计算时间为 2.14 s, 计算效率较高, 有效地解决了计算时间长的问题. 可见, 对于 VRP 问题, 利用

表2 实验计算结果

计算次序	1	2	3	4	5	6	7	8	9	10	平均
最小配送距离	43.37	41.86	43.1	41.99	41.86	43.21	41.94	43.1	44.32	42.58	42.73
使用车辆数	4	4	4	4	4	4	4	4	4	4	4
首次搜索终解代数	26	39	26	32	35	28	41	37	19	39	32.3
计算时间	1.99	2.2	1.99	2.14	2.15	2.07	2.23	2.16	1.3	2.19	2.14
与最小的差值	1.51	0	1.24	0.13	0	1.35	0.08	1.24	2.46	0.72	0.87

蚁群算法可取得非常好的结果,且计算结果较为稳定

为了便于比较,分别用爬山法、遗传算法、模拟退火算法对该实例求解10次,在解的搜索次数都为15 000次的前提下,4种算法的计算结果如表3所示

表3 爬山法、遗传算法、模拟退火算法以及蚁群算法的比较

计算次序	爬山法	遗传算法	模拟退火算法	蚁群算法
平均配送总距离/km	50.64	56.62	46.15	42.73
解的标准差	7.48	3.25	2.67	0.87
平均使用车辆数	4.3	4.5	4	4
首次搜索到最终解的迭代次数	4 905	12 810	9 080	9 690

由表3不难看出,从寻优结果看,蚁群算法的计算结果明显优于其他3种算法;从计算效率看,蚁群算法的计算效率低于爬山算法和模拟退火算法,但高于遗传算法;从算法的稳定性看,蚁群算法计算结果的稳定性明显优于其他3种算法

4 结 论

蚁群算法是一种来自大自然的随机搜索寻优方法,是生物界的群体启发行为,现已陆续应用于组合优化、人工智能等多个领域.蚁群算法的正反馈性和协同性使其可用于分布式系统,隐含的并行性更使之具有极强的发展潜力.本文将蚁群算法引入VRP问题的求解,构造了适于求解该问题的自适应蚁群算法.实验结果表明,蚁群算法不但具有很强的发现较好解的能力,而且具有较高的计算效率,计算结果也较为稳定

参考文献(References)

[1] Laport G. The vehicle routing problem: An overview of exact and approximate algorithms[J]. *European J of Operational Research*, 1992, 59(1): 345-358.

[2] Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents[J]. *IEEE Trans on System, Man, and Cybernetics*, 1996, 26(1): 29-41.

[3] Maniezzo V, Colorni A. An ANTS heuristic for the frequency assignment problem[J]. *Future Generation Computer Systems*, 2000, 16(8): 927-935.

[4] Colorni A, Dorigo M. Ant system for job shop scheduling[J]. *Operation Research*, 1994, 34(1): 39-53.

[5] Costa D. Ant can color graphs[J]. *J of the Operations Research Society*, 1997, 48(3): 295-305.

[6] Dorigo M, Luca M. A study of some properties of ant-Q[A]. *Proc of 4th Int Conf on Parallel Problem Solving fom Nature (PPSN)*[C]. Berlin: Springer Verlag, 1996: 656-665.

[7] Stutzle T. MAX-MIN ant system[J]. *Future Generation Computer Systems J*, 2000, 16(8): 889-914.

[8] Gambardella L M, Dorigo M. An ant colony system hybridized with a new local search for the ordering problem[J]. *Informs J on Computing*, 2000, 12(3): 237-255.

[9] Zhang J H, Xu X H. A new evolutionary algorithm-ant conoly algorithm[J]. *System Engineering Theory and Application*, 1999, 36(3): 84-87.

[10] Clarck G, Wright J W. Scheduling of vehicles form a central depot to a number of delivery points[J]. *Operations Research*, 1964, 12(4): 568-581.

下 期 要 目

对队形控制的思考 …… 任德华,卢桂章

基于LMI方法的时滞分布参数控制系统的镇定 …… 罗毅平,等

量子力学系统的输出反馈控制 …… 张 清,等

不确定时滞广义系统的鲁棒非脆弱H_∞控制 …… 舒伟仁,张庆灵

基于形式化描述的逻辑分层延迟PSO算法及应用 …… 汪 镭,等

供应链中间产品动态价格振荡及其收敛性分析 …… 杜义飞,李仕明

资本市场系统结构模型应用研究 …… 蒋振声,等