车辆路径问题的混合蚁群算法设计与实现①

刘志硕，申金升，关 伟
（北京交通大学交通运输学院系统工程与控制研究所，北京 100044）

摘要：蚁群算法是一种新型的模拟进化算法，具有许多优良的性质，可以很好地解决TSP问题.在分析车辆路径问题(VRP)与TSP区别的基础上，论文将蚁群算法应用于VRP的求解，针对VRP的具体特点，构造了具有自适应功能的混合蚁群算法.该算法对基本规则作了进一步改进，并有机结合了爬山法、节约法等方法，以减少计算时间，避免算法停滞.指出可行解问题是蚁群算法的关键问题，提出了大蚂蚁数、近似解可行化等四个解决策略.计算机仿真结果表明，自适应混合蚁群算法性能优良，能够有效地求解VRP.

关键词：车辆路径问题；旅行商问题；蚁群算法；爬山法；近似解可行化
中图分类号：TP391.9 **文献标识码**：A **文章编号**：1007-9807(2007)03-0015-08

0 引 言

车辆路径问题(vehicle routing problem, 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.该问题最早是由Dantzig和Ramser于1959年首次提出的[1].国外已对车辆路径问题作了大量而深入的研究[2,3]，例如早在1983年Bodin、Golden等人在他们的综述文章中就列举了700余篇文献[4].在国内，市场经济发展对国内物流行业发展的强大推动作用给车辆路径问题的研究带来了极大的推动力，越来越多的专家学者参与到该领域的研究中来[5~7].

求解车辆路径问题的方法非常丰富，基本上可以分为精确算法和启发式算法两大类.精确算法主要有分枝定界法、割平面法、网络流算法、动态规划法等.由于VRP是强NP难题，高效的精确算法存在的可能性不大，所以寻找近似算法是必要和现实的，为此专家们主要把精力花在构造高质量的启发式算法上.目前已经提出的启发式算法很多，比较典型的算法有节约法、两阶段法、扫描法、并行算法、禁忌搜索算法、模拟退火算法、遗传算法、神经网络算法等.

虽然国内外已经就车辆路径问题进行了很多的探讨和研究，并取得了许多成果，但是如何针对车辆路径问题的特点，构造一种运算简单、寻优性能优良的启发式算法，仍是一个值得深入研究的课题.

蚁群算法(Ant Colony Algorithm)[8]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，它吸收了昆虫王国中蚂蚁的行为特性，通过其内在的搜索机制，在一系列困难的组合优化问题求解中取得了成效.现有研究成果表明，蚁群算法可以很好的解决对称性和非对称性TSP[9,10].鉴于VRP与TSP的相似性，本文尝试采用该算法来对VRP进行求解.

1 蚁群算法研究现状

蚁群算法是由意大利学者Dorigo等人于1991年首先提出的.在提出蚁群算法后，Dorigo等人又应用该算法求解了TSP、分配问题[11]、Job-Shop调度问题[12]，取得了较好的结果.受其影响，蚁群算

① 收稿日期：2004-06-07；修订日期：2006-12-25.
基金项目：博士后科学基金资助项目(023209031).
作者简介：刘志硕(1977—)，男，湖南安仁人，博士，讲师.Email:lzs@sina.com.cn.

法逐渐引起了其它研究者的注意，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．如：Costa 在 Dorigo 等人研究成果的基础上，提出了一种求解分配类型问题的一般模型，并用来研究图着色问题[13]．Bilihev 研究了求解连续空间优化问题的蚁群算法模型[14]；Botte 等人对蚁群算法的控制参数的选择进行了深入的研究，用遗传算法求得这些参数最优组合[15]．为了克服基本蚁群算法的不足，人们对它进行了各种改进，以期提高搜索效率和避免过早停滞，其中主要有：Dorigo 等人提出的 Ant-Q System[16]；Stutzle 提出 的 MMAS (Max-Min Ant System)[17]；Gambardella 等人提出的混合型蚁群算法 HAS[18]等等．我国对蚁群算法的研究尚处于起步阶段．国内对蚁群算法的介绍和研究起始于 1999 年初[19,20]．由于该算法在 TSP、job-shop 问题以及组合优化等方面所显示的优越性能，几年来，得到了越来越多的学者关注，目前已经取得了一些成果[21,22]．

2 自适应混合蚁群算法设计

2.1 算法总体思路

TSP 是 VRP 的基本问题，二者相似又存在诸多差别．因此，在设计 VRP 的蚁群算法时，一方面要注意吸收前人设计 TSP 蚁群算法的经验，另一方面，又要充分考虑 VRP 的具体要求．此外，由于 VRP 的复杂程度远高于 TSP，因此在设计蚁群算法时，研究如何减少算法的计算时间，提高搜索效率就显得尤为重要．有鉴于此，本文蚁群算法的总体设计思路就是，以 TSP 的蚁群算法为基础，充分考虑 VRP 的具体要求，并在此基础上，对算法的选择机制、更新机制以及协调机制进一步改进，引入自适应的转移策略和信息素更新策略，并融合爬山法、节约法等求解 VRP 的传统方法，充分发挥其易与其它方法结合的优点，以克服蚁群算法计算时间长、易出现停滞的缺陷．

2.2 VRP 和 TSP 的蚁群算法区别

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，每只蚂蚁移动次数是不确定的，只能将是否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两个阶段．① 对于初始位置不是配送中心（0 点）的蚂蚁来说，第一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 $allowed_k$ 中要包括 0 点，而不能有初始结点．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”．在此过程中，$allowed_k$ 中不能包含 0 点，而应该包含初始结点．② 对于初始结点为 0 点的蚂蚁来说，其“过程 Ⅰ”就是由 0 点移到任意其它结点的过程，“过程 Ⅱ”就是由其它结点回到 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）可行解结构的区别　在 TSP 中，每只蚂蚁所构造的路径均是一个可行解，但在 VRP 中，每只蚂蚁所构造的回路仅是可行解的“零部件”，这些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．因此研究如何尽量避免无可行解的出现是蚁群算法设计中的一个无法回避的问题．

3）可行点集 $allowed_k$ 的区别　在 TSP 中，蚂蚁转移时，只须考虑路径距离和信息素浓度即可，但在 VRP 中，不但要考虑上述因素，还需考虑车辆的容量限制．这一差异在算法中的具体体现就是 $allowed_k$ 的确定问题．在 TSP 中，每只蚂蚁已走过的结点放在 $tabu_k$ 中，其他未走的结点均是其可选结点，放在 $allowed_k$ 中．而在 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 点和初始点作特殊处理．

2.3 算法基本规则的设计

2.3.1 转移规则

借鉴 Dorigo 的 Ant-Q 算法思想[20]，采用确定性选择和随机选择相结合的转移策略，如下

$$j=\begin{cases}\text{依概率 } \max\{p_{ij}\} \text{ 选择 } j,\text{其中 } p_{ij}=\dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}[\mu_{ij}]^{\gamma}}{\sum\limits_{h\in allowed_k}[\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}[\mu_{ih}]^{\gamma}},j\in allowed_k & \text{当 } p\leqslant r_0\\ \text{依 } \max_{j\in allowed_k}\{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}[\mu_{ij}]^{\gamma}\} \text{ 选择 } j & \text{当 } p>r_0\end{cases}\tag{1}$$

τ_{ij} 和 η_{ij} 的含义与 TSP 蚁群算法中的相同，分别表示信息素浓度和能见度（取边长倒数），$\mu_{ij}=d_{i0}+d_{0j}-d_{ij}$ 是考虑了任务点与配送中心间的距离而引入的变量，称为节约值. 这吸收了节约法[23]的思想. μ_{ij} 反映了将两点直接连接比将两点分别与配送中心连接所能获得的路径长度节约量，显然，μ_{ij} 越大，收益越大，而选择结点 v_j 的概率也就越大. p 是个随机数，服从(0,1) 区间上的均匀分布. r_0 随算法的进化过程在 0 ~ 1 范围内动态调整. 在初始阶段，蚂蚁选择的范围较大，因此 r_0 也应该比较大. 在中间阶段，若算法收敛速度较慢，则减小 r_0，而若较快，则适当地放大 r_0. 若算法陷入局部解，则加大 r_0，让蚂蚁更多地进行随机选择.

2.3.2　信息素更新规则

信息素更新采用局部更新和全局更新相结合的策略.

1) 局部更新规则

定义 1　吸引力. 设经过结点 i 的蚂蚁数为 R，经过有向边 (i,j) 的蚂蚁数为 r，则称 r/R 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量 Q 为常量，则 (i,j) 的蚂蚁吸引力越大，经过边 (i,j) 的蚂蚁相对其它的边来说数量越多，从而局部更新的次数就越多，久而久之，会导致边之间的信息素量差距过大，限制算法搜索的全局性. 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. 随着算法搜索状态的变化，Q 值也应该不断调整，调整的原则是，边的吸引力越大，$Q(t)$ 越小，不妨设 $Q(t)=Q(1-r/R)$.

假设第 k 只蚂蚁在第 nc 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). 则该蚂蚁在 (i,j) 上的信息素局部更新量（采用 Ant-Q system 模型[19]）为

$$\Delta\tau_{ij}^{k}=\begin{cases}Q_1(1-r_k/R_k) & \text{蚂蚁 } k \text{ 从 } i \text{ 转移到 } j\\ 0 & \text{否则}\end{cases}\tag{2}$$

因此得到算法的局部更新规则如下

$$\tau_{ij}^{\text{new}}=\rho_1\cdot\tau_{ij}^{\text{old}}+\Delta\tau_{ij}\quad\forall i,j\quad i\neq j\tag{3}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}\tag{4}$$

τ_{ij}^{old} 和 τ_{ij}^{new} 分别表示边 (i,j) 上的原有信息素浓度以及更新后的信息素浓度.

2) 全局更新规则

在所有蚂蚁均构造完路径后，对所有可行解以及岂今为止最优解的边进行全局信息素更新. 采取的规则如下

$$\tau_{ij}^{\text{new}}=\rho_2\tau_{ij}^{\text{old}}+\sum_{p=1}^{P}\Delta\tau_{ij}^{p}+\Delta\tau_{ij}^{*}\quad\forall i,j,i\neq j\tag{5}$$

其中，当 $(i,j)\in A_p$ 时，$\Delta\tau_{ij}^{p}=\dfrac{Q_2}{D(A_p)}$，否则 $\Delta\tau_{ij}^{p}=0$.

当 $(i,j)\in L^{*}$ 时，$\Delta\tau_{ij}^{*}=\dfrac{Q_3}{D(L^{*})}$，否则 $\Delta\tau_{ij}^{*}=0$.

$D(A_p)$ 表示第 p 个可行解的路径长度，$D(L^{*})$ 表示迄今为止最优解的路径长度.

2.3.3　负反馈机制

蚁群算法的选择机制就是使好的路径会以较大的概率选中，而正反馈机制的存在又必然会促使这些路径在以后的选择中更具竞争优势. 当搜索陷入局部最优解时，按传统的方式，算法无法从局部极小值点跳出来. 为此引入负反馈机制，包括：1) 借鉴 MIN - MAX 的思想[21]，将各个路径上的信息量限定在某个固定的范围内. 这在一定程度上可以减少路径之间信息素浓度的差距，促使算法从局部极小值点跳出来；2) 若出现停滞现象，则在全局更新时，加入负反馈信息量，即让 Q_2、$Q_3<0$，以减少该局部解所对应路径上的信息素量.

2.3.4 爬山法

将爬山法混入蚁群算法求解过程,对每代最优解进行改进.设最优解由 p 个子回路组成,记为 $\{L_1, L_2, \cdots, L_p\}$,改进的过程实际上就是对各个子回路分别应用爬山法.不失一般性,设 W_k 是路径 L_k 经过结点的有序排列,$W_k = \{v_0, v_1, v_2, \cdots, v_n, v_0\}$, v_i、v_j 是从 W_k 中随机地选择的2个顶点,交换这两点的位置得到 L_k', C_N 为没有任何改进的最大循环次数,则爬山法的步骤如下:

步骤 1 初始化循环次数变量 $t = 1$,当前最优解 $S^* = S$,其长度为 $L(S^*)$;

步骤 2 在 W_k 中随机选择 2 个顶点,v_i、v_j,$i < j$,v_i 和 v_j 不相邻;

步骤 3 计算距离节约 $\Delta C_d = [d(x_{i+1}, x_j) + d(x_i, x_{j+1})] - [d(x_{i+1}, x_i) + d(x_j, x_{j+1})]$, 若 $\Delta C_d > 0$,则不进行交换,t^{++},转步骤4,否则实施交换,更新 W_k,相应的解为 S',则最优解 $S^* = S'$, $t = 1$, 转步骤 2;

步骤 4 若 $L(S^*)$ 在最后 C_N 个循环中没有减少,则本算法结束;否则转步骤 2.

2.4 可行解问题的研究

在 VRP 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.如果得不到可行解,则该次迭代除施加了局部信息素影响外对最优解的搜索未有任何影响,若经常出现这种情况,则显然会浪费大量的计算时间.因此研究如何有效避免无可行解现象是算法设计的一个关键问题.可以采取以下策略: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) 多周期蚂蚁组合策略.蚂蚁数过大会导致计算时间长,而若蚂蚁数较少,即使用初始分布均匀策略,其效果往往也不会很理想.因此,在蚂蚁数较少情况下,可以将本次迭代中各蚂蚁所产生的回路和往次未发现可行解的迭代(一次或多次)所产生的子回路进行组合.这样就充分利用了往次迭代中各蚂蚁所构造的子回路,既增加了组合的蚂蚁数量(按倍数增加),提高发现可行解的可能性,又不太影响算法的计算时间,并且由于各子回路更加多样化,还能够有效提高算法的全局搜索能力.4) 近似解可行化策略.当问题规模较大时,会有大量蚂蚁找到的回路是相同的,从而使得事实上真正起作用的蚂蚁数大大减少,严重影响可行解产生的可能性.针对这个问题,提出近似解可行化策略.所谓近似解可行化,就是指在找不到可行解的情况下,选择一些和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(如节约法等)将近似解转变成可行解.当算法不存在可行解时,采取该策略显然可以保证获取至少一个可行解.

2.5 算法步骤

蚁群算法求解 VRP 的步骤如下:

步骤 1 初始化各参数,输入系统基础数据,设找到的最优解 $L^* = \text{MAX\_ RLT}$,迭代计数器 $nc = 0$;

步骤 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,初始化蚂蚁 k 的已走点集 $tabu_k$ 以及候选点集 $allowed_k$;l 表示已完成任务的蚂蚁数,$l = 0$;初始点为0的蚂蚁的过程变量 $Pro[k] = 1$,初始点为非 0 点的蚂蚁的过程变量 $Pro[k] = 2$;

步骤 3 在结点 i 取蚂蚁 k,判断其初始结点.若为 0 点,按转移规则确定结点 j,更新 $tabu_k$、S_i、b_i.并且若 $Pro[k] = 1$,则 $Pro[k] = 2$,转步骤 3;而当 $Pro[k] = 2$ 时,若 j 点为初始结点,则 $l++$,转步骤 3,否则转步骤 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 $Pro[k] = 1$ 时,若 j 点为 0,则 $Pro[k] = 2$,转步骤 3;当 $Pro[k] = 2$ 时,若 j 点为初始结点,则 $l = l + 1$, 转步骤 3,否则转步骤 4;

步骤 4 更新 S_j,重复步骤 3 ~ 4 直到该结点的蚂蚁数量 $b_i = 0$ 为止;

步骤 5 重复步骤 3 ~ 5,直到所有结点的 $b_i = 0$;

步骤 6 在所有蚂蚁都移动一次后,按局部

更新规则对所有路径的信息素进行更新；

步骤7　更新所有结点的 b_i，若所有结点上蚂蚁数量 $b_i = 0$ 或 $l = M$，转下一步，否则转步骤3；

步骤8　由各蚂蚁的 $tabu_k$ 得到路径集 $L = \{L_1, L_2, \cdots, L_M\}$，在 L 中寻找可行解，得到可行解集 $A = \{A_1, A_2, \cdots, A_p\}$；若未发现可行解，实施多周期蚂蚁组合策略生成 L_ch 子回路集，再在 L_ch 中寻找可行解. 若仍未发现可行解，则采取近似解可行化策略，转下一步.

步骤9　采用爬山法对可行解集 A 中各个可行解进行局部优化，得到 A'，计算本次搜索到的最优路径 $L^*(nc)$，并得到迄今为止的最优路径，$L^* = \min\{L^*(nc), L^*\}$；

步骤10　若算法陷入停滞，采用负反馈机制调整路径上的信息素，并调大 r_0，否则按全局更新规则进行信息素的全局更新；

步骤11　判断 nc 是否等于最大迭代次数 $nc_{\max}$，若是，则算法终止，输出 L^*；否则，nc^{++}，转步骤2.

3　实验及分析

3.1　实例仿真

设有20个客户随机分布于边长为10千米的正方形区域内. 配送中心位于区域正中央，其坐标为(0,0). 各客户的需求也由计算机随机产生，车辆容量为9吨. 基础数据如表1所示.

表1　实验基础数据

Table 1 Description of a VRP

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

采用的运行参数为 $M = 60$，$nc_{\max} = 50$，$\tau_{ij} = 10$，MAX = 100，MIN = 2，$\alpha = 1$，$\beta = 1$，$\gamma = 0.5$，$\rho_1 = 0.85$，$\rho_2 = 0.95$，$Q_1 = 10$，$Q_2 = 50$，$Q_3 = 100$，$r_0 = 1$. 运行10次，结果如表2所示.

表2　实验计算结果

Table 2 Computing results of ACA

计算次序	1	2	3	4	5	6	7	8	9	10	平均
最小配送距离/km	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/s	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/km	1.51	0	1.24	0.13	0	1.35	0.08	1.24	2.46	0.72	0.87

从表2中可以看出，用蚁群算法的10次求解中，都得到了质量很高的解. 其中最好解为41.86 km，对应的配送路径为：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较稳定，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，利用蚁群算法可取得比较理想的结果，且计算结果也较稳定.

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了10次，在解的搜索次数都为15 000次的前提下，4种算法的计算结果见表3.

表3　爬山法、遗传算法、模拟退火算法以及蚁群算法的比较

Table 3 Performance comparison between the ACA, GA, SA and 2－opt

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,蚁群算法明显优于其它3种算法;从计算效率看,蚁群算法低于爬山算法和模拟退火算法,但高于遗传算法;从算法的稳健性看,蚁群算法计算结果的稳定性明显优于其它3种算法.

3.2 运行参数和算法策略对蚁群算法性能的影响分析

1) 运行参数对蚁群算法搜索性能的影响

蚁群算法的运行参数主要有蚂蚁数 M、基本信息素量 τ_{ij}、转移规则各相对重要性参数(α、β、γ、λ)、信息素的持久度(ρ_1、ρ_2)以及施放量(Q_1、Q_2、Q_3)等.各参数不同的取值会对算法的搜索性能产生一定的影响,限于篇幅,本文仅分析蚂蚁数 M 对算法搜索性能的影响.

对上述实例,分别取蚂蚁数 M = 30、40、50、60、70、80、90、100,其它参数 $nc_{\max}$ = 50,τ_{ij} = 20,MAX = 80,MIN = 2,α = 2,β = 1,γ = 0.5,λ = 0.2,ρ_1 = 0.85,ρ_2 = 0.95,Q_1 = 5,Q_2 = 50,Q_3 = 100,r_0 = 1,各运行50次,结果如表4所示.

表4 蚂蚁数 M 对蚁群算法性能的影响分析表

Table 4 Impact analysis of the number of ant for performance of ACA

蚂蚁数量	平均配送总里程/km	平均使用车辆数	平均迭代步数	平均计算时间/s
30	62.83	4	29.7	0.15
40	55.54	4	28.7	0.28
50	46.49	4	43.9	0.47
60	44.48	4	43.6	0.71
70	44.31	4	42.9	1.87
80	43.14	4	41.8	2.14
90	42.74	4	32.8	2.52
100	42.65	4	30.5	3.86

由表4可以看出,前4种蚂蚁数下,蚁群算法所得到的平均计算结果差别比较明显,由66.83下降到44.48,解的质量明显提高,达到30%,算法的平均时间由0.15 s增加到0.71 s.总体来看,当蚂蚁数比较小的时候,解的质量较差.后4种蚂蚁数下,算法的结果差别不大,由44.31下降到42.65,仅有3.5%,而计算时间却增加了1倍.蚂蚁数较大时,解的质量比较高,但随着蚂蚁数的不断增加,其解质量的提高越来越不明显,而计算时间却大幅度增加.因此就本实例而言,采用60~80个蚂蚁就可以了.可见,采用蚁群算法求解VRP时,蚂蚁数量对算法搜索性能的影响是比较明显的,过少的蚂蚁数量不利于取得较好的计算结果,而过多的蚂蚁数量尽管会使算法的寻优结果有小幅度的提高,但却会极大影响计算时间.

2) 算法策略对蚁群算法搜索性能的影响

这里主要分析可行解获取策略对大规模VRP的蚁群算法性能的影响.在实验中发现,若不采取任何可行解获取策略,较大规模问题的蚁群算法基本发现不了可行解,因此对大蚂蚁数策略、蚂蚁初始分布均匀策略、多周期蚂蚁组合策略以及近似解可行化策略等4个策略以及它们之间的组合进行实验比较.

具体分析6种算法.其中,第1种算法采取小蚂蚁数(M = 40)和初始蚂蚁分布均匀策略,第2种算法采取大蚂蚁数(M = 80)和初始蚂蚁分布均匀策略,第3种算法在第1种的基础上,增加多周期蚂蚁组合策略,第4种算法在第1种的基础上,增加近似解可行化策略,第5种算法在第2种的基础上,增加近似解可行化策略,第6种算法在第4种的基础上,添加多周期蚂蚁组合策略.分别对前述实例求解,最大迭代次数为20,各运行20次,结果如表5所示.

表5 算法策略对蚁群算法性能的影响分析表

Table 5 Impact analysis of the four techniques for performance of ACA

算法策略	平均配送总里程/km	找不到解的迭代数	平均无可行解数	平均计算时间/s
1	64.61	13	18.6	0.16
2	56.43	0	11.9	2.08
3	57.83	0	14.7	0.51
4	48.03	0	0	0.36
5	43.13	0	0	2.86
6	46.26	0	0	0.72

由上表不难看出,在仅使用初始蚂蚁均匀分布策略时,平均配送总里程最高,达到64.61 km,由于蚂蚁数量较少,在20次运行中,有13次未得到任何解,并且在得到解的那7次运行中,平均有18.6次迭代未找到任何可行解,即只有1.4次迭代找到了可行解.在使用大蚂蚁数策略后,情况有较大改观,尽管计算时间有较大攀升,达到2.08 s,但解的质量却提高到了56.43 km,每次运行都找到了解,并且各次运行中平均未找到可行

解次数也降低到了 11.9 次.第 3 种算法相对第 1 种而言,从计算结果来看,虽然计算时间有一点延长,但其求解质量有较大提高,而相对第 2 种而言,其计算结果和平均无可行解次数均比较接近,而计算时间却有较大幅度的节省.后 3 种算法由于引进了近似解可行化策略,因此不存在无可行解的问题,故找不到解的迭代数和平均无可行解数这两项均为 0.第 4 种算法的计算结果质量较前 3 者有很大提高,由 56.43 km 降低到 48.03 km,下降了 14.9%.在各种算法中,计算结果最好的为第 5 种算法,其平均计算结果为 43.13 km,离问题最好解 41.86 km 相当接近,其原因在于它既采用了大蚂蚁策略,又采用了近似解可行化策略,因而计算时间最长.

总之,从以上实验可以看出,通过蚂蚁的选择机制、信息素更新机制以及蚂蚁群之间的协同机制,蚁群算法具有很强的发现较好解的能力.并且通过对算法的选择机制、更新机制以及协调机制的进一步改进,融合爬山法以及节约法,能够有效提高算法的计算效率,避免过早收敛.

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到 VRP 中来.通过分析 VRP 与 TSP 的区别,构造了适于求解 VRP 的蚁群算法.并且为减少算法计算时间,避免算法停滞,对算法的选择机制、更新机制以及协调机制作了进一步研究,并融合了爬山法以及节约法等方法.实验结果表明本文所设计的自适应混合蚁群算法不但具有很强的发现较好解的能力,还具有较高的计算效率,计算结果相当稳定.

参 考 文 献:

[1]Dantizig G, Rmser J. The truck dispatching problem[J]. Management Science, 1959, 6(1): 80—91.

[2]Golden B L, Assad A. Vehicle Routing: Methods and Studies[M]. Amsterdam: Elsevier Science Publishers B. V. 1998.

[3]Laport G. The vehicle routing problem: An overview of exact and approximate algorithms[J]. European Journal of Operational Research, 1992, 59(4): 345—358.

[4] Bodin L D, Golden B L, Assad A A, *et al*. Routing and scheduling of vehicles and crews: The state of art[J]. Computers&Operations Research, 1983, 10(3): 63—211.

[5]郭耀煌, 李　军. 满载问题车辆路线安排[J]. 系统工程学报, 1995, 10(2): 106—118.
Guo Yaohuang, Li Jun. Vehicle routing with full lodads[J]. Journal of Systems Engineering, 1995, 10(2): 106—118. (in Chinese)

[6]李　军. 车辆调度问题的分派启发式算法[J]. 系统工程理论与实践, 1999, 19(1): 27—33.
Li Jun. Generalized assignment heuristics for vehicle scheduling[J]. Systems Engineering—Theory & Practice, 1999, 19(1): 27—33. (in Chinese)

[7]郎茂祥. 车辆路径问题的禁忌搜索算法研究[J]. 管理工程学报, 2004, 18(1): 81—84.
Lang Maoxiang. Study on the tabu search algorithm for vehicle routing problem[J]. Journal of Industrial Engineering and Engineering Management, 2004, 18(1): 81—84. (in Chinese)

[8]Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by a Colony of Cooperating Agents[C]. IEEE Trans. on System, Man, and Cybernetics, 1996, 26(1): 29—41.

[9]Stutzle T, Hoos H. The MAX - MIN Ant System and Local Search for the Traveling Salesman Problem[C]. Proc. of ICEC'97, 1999 IEEE 4th Int. Conf. on Evolutionary Computation. IEEE Press, 1997. 308—313.

[10]Dorigo M, Gambardella L M. Ant colonies for the traveling salesman problem[J]. Bio. Systems, 1997, 43(2): 73—81.

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

[12]Colorni A, Dorigo M. Ant system for job shop scheduling[J]. Operations Research, 1994, 34(1): 39—53.

[13]Costa D, Hertz A. Ant can color graphs[J]. Journal of the Operational Res. Soc., 1997, 48(3): 295—305.

[14]Bilchev G, Parmee I C. Adaptive Search Strategies for Heavily Constrained Design Spaces[C]. Proceedings of 22nd International Conference on Computer Aided Design 95. Ukraine, Yelta, 1995. 230—235.

[15]Botee H M, Bonabeau E. Evolving ant colony optimization[J]. Complex Systems, 1998, 1(2): 149—159.

[16]Dorigo M, Luca M. A Study of Some Properties of Ant-Q[C]. Proc. of 4th Int. Conf. on Parallel Problem Solving form Nature (PPSN). Springer Verlag, Berlin, 1996. 656—665.

[17]Stutzle T, Hoos H. MAX-MIN ant system[J], Future Generation Computer Systems Journal, 2000, 16(8): 889—914.

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

[19]马 良. 来自昆虫世界的寻优策略—蚂蚁算法[J]. 自然杂志, 1999, 21(3): 161—163.
Ma Liang. Ant algorithm-optimization strategy from the insect colony[J]. Nature Magazine, 1999, 21(3): 161—163. (in Chinese)

[20]张纪会, 徐心和. 一种新的进化算法 - 蚁群算法[J]. 系统工程理论与实践, 1999, 36(3): 84—87.
Zhang Jihui, Xu Xinhe. A new evolutionary algorithm ant colony algorithm[J]. Systems Engineering—Theory & Practice, 1999, 36(3): 84—87. (in Chinese)

[21]马 良, 项培军. 蚂蚁算法在组合优化中的应用[J]. 管理科学学报, 2001, 4(2): 32—37.
Ma Liang, Xiang Peijun. Applications of the ant algorithm to combinatorial optimization[J]. Journal of Management Sciences in China, 2001, 4(2): 32—37. (in Chinese)

[22]覃刚力, 杨家本. 自适应调整信息素的改进蚁群算法[J]. 信息与控制, 2002, 31(3): 198—210.
Qin Gangli, Yang Jiabe. An improved ant colony algorithm based on adaptively adjusting pheromone[J]. Information and Control, 2002, 31(3): 198—210. (in Chinese)

[23]Clarck G, Wright J W. Scheduling of vehicles form a central depot to a number of delivery points[J]. Operations Research, 1964, 12(8): 568—581.

Design and realization of a hybrid ant colony algorithm for vehicle routing problem

LIU Zhi-shuo, SHEN Jin-sheng, GUAN Wei

Insititute of System Engineering of Beijing Jiaotong University, Beijing 100044, China

Abstract: Ant Colony Algorithm(ACA) is a novel simulated evolutionary algorithm which shows many promising properties and can solve Traveling Salesman Problem(TSP) efficiently. On the basis of analyzing the difference between VRP and TSP, an Adaptive Hybrid Ant Colony Algorithm(AHACA) is proposed to solve VRP, which is improved from basic ACA by improving the basic rules and integrating 2-opt local search method and C-W algorithm in order to decrease computing time and avoid stagnation behavior of basic ACA. Moreover, the problem of acquiring feasible solution is also discussed, and four resolutions such as Mass Ant, Feasibility Process of Approximate Solutions etc. are also introduced. Simulation results show that the AHACA is feasible and valid for VRP.

Key words: vehicle routing problem; traveling salesman problem; ant colony algorithm; 2-opt; feasibility process of approximate solutions