文章编号: 1001-0920 (2012) 11-1633-06

带容量约束车辆路由问题的改进蚁群算法

王沛栋[1,2], 唐功友[1], 李 扬[1]

(1. 中国海洋大学 信息科学与工程学院，山东 青岛 266100；2. 青岛市产品质量监督检验所，山东 青岛 266101)

摘 要: 提出一种带容量约束车辆路由问题 (CVRPs) 的改进蚁群算法. 该算法使用一种新的蚂蚁位置初始化方式, 增加了蚂蚁走出最优路径的可能性. 在搜索过程中, 以客户之间路径的节省量作为启发式信息. 信息素更新采用一种动态更新的方法, 能够根据当前车辆所构建路径的情况对信息素进行更新, 避免算法陷入停滞状态. 局部搜索除使用 2-opt 方法外, 针对不同车辆访问的客户, 还增加了交换搜索和插入搜索以扩大搜索范围. 仿真实验验证了所提出算法的有效性.

关键词: 车辆路由；路径规划；蚁群算法；带容量约束车辆路由问题

中图分类号: TP18 **文献标志码:** A

Improved ant colony algorithm for capacitated vehicle routing problems

WANG Pei-dong[1,2], *TANG Gong-you*[1], *LI Yang*[1]

(1. College of Information Science and Engineering，Ocean University of China，Qingdao 266100，China；2. Qingdao Supervision and Testing Center of Product Quality，Qingdao 266101，China. Correspondent：WANG Pei-dong，E-mail: qdrnocking@yahoo.com.cn)

Abstract: An improved ant colony algorithm is proposed for capacitated vehicle routing problems(CVRPs). A new initialization of vehicle's position with an optimal and random selection increases the possibility of obtaining the optimal path. In the process of searching, the ants are more sensitive to the optimal path, because the saving path among customers is chosen as the heuristic information. The method of local and global dynamic phenomenon update is used in order to adjust the distribution of phenomenon according to vehicle routes. Except the method of 2-opt, insertion and exchange search methods are also used to expand the scope of the search for the clients on different vehicle visits. The simulation results show the effectiveness of the proposed algorithm.

Key words: vehicle routing problem；path plannin；ant colony algorithm；CVRPs

1 引 言

带容量约束的车辆路由问题 (CVRPs) 是在物流运输和资源配置等方面具有广泛应用的组合优化问题, 也是车辆路由问题 (VRPs) 中的基本问题之一. 其任务是在某些约束条件下, 规定车辆由车场出发向多个客户进行配送服务, 设计车辆的输送路线, 使车辆的运输成本最小.

目前求解 CVRPs 有许多方法, 大体可以分为两类算法: 精确算法和启发式算法. 精确算法主要有动态规划和分支限界法[1]等; 启发式算法包括: 遗传算法[2-3]、蚁群算法[4-10]等. 文献 [2] 和 [3] 分别提出了改进遗传算法和混合遗传算法来求解 CVRPs, 相比原遗传算法, 性能有了较大的提高. 文献 [6] 首次将蚁群算法应用于 CVRPs, 提出了一种基于蚂蚁系统 (AS)[4] 的改进蚁群算法, 以客户间的路径节省量作为启发式信息, 每次迭代后使用 2-opt 对最优解进行搜索. 文献 [7] 在 [6] 的基础上加以改进, 在每次搜索过程中添加候选列表, 加快了最优解的收敛速度. 文献 [8] 和 [9] 分别在蚂蚁系统和蚁群系统 (ACS)[5] 的基础上提出了多蚁群的方法来求解 CVRPs. 文献 [10] 利用改进的蚁群系统来求解 CVRPs. 文献 [11] 提出了一种基于 SR-GCWS (simulation in routing via the generalized Clarke & Wright savings) 的混合方法来求解 CVRPs.

收稿日期: 2011-07-15；**修回日期:** 2011-10-12.

基金项目: 国家自然科学基金项目(61074092)；山东省自然科学基金项目(ZR2010FM019)；山东省科技发展计划项目(2008GGB01192).

作者简介: 王沛栋(1982−), 男, 博士生, 从事控制理论与应用、智能控制的研究；唐功友(1953−), 男, 教授, 博士生导师, 从事控制理论与应用、智能控制等研究.

本文提出一种改进的蚁群优化算法. 首先, 使用一种新的车辆位置初始化方式, 增加了车辆走出最优路径的可能性; 然后, 搜索过程采用客户间的路径节省量作为启发式信息; 同时, 信息素更新使用一种动态更新的方法以调配路径上的信息素; 最后, 在2-opt的基础上, 局部搜索增加了交换搜索和插入搜索以扩大搜索范围. 实验结果表明, 与其他算法相比, 本文算法在解的质量和收敛速度上都显示出了良好的性能.

2 问题描述及数学模型

CVRPs是由一个车场的车辆向多个客户进行配送服务的问题. 在已知车场和客户的位置、客户的需求量以及车辆的数量和最大负载的前提下, 设计车辆配送路线以满足所有客户的需求, 使运输成本最小化, 即总代价最小(行车总距离最短或耗费时间最少). 用数学模型表达, CVRPs可描述为从车场0出发的m辆车为n位客户服务的问题. 设d_i为客户i的需求量, c_{ij}为客户i到客户j的运输费用, b_k为车辆k的负载量, 而

$$x_{ij}^k=\begin{cases}1, \text{ 车辆 } k \text{ 服务客户 } i \text{ 后接着服务客户 } j;\\ 0, \text{ 其他.}\end{cases} \tag{1}$$

CVRPs的目标是为车队寻找最优的运输路径, 使运输成本J达到最小值, 即

$$J=\min\sum_{k=1}^{m}\sum_{\substack{i,j=0\\ i\neq j}}^{n}c_{ij}x_{ij}^k, \tag{2}$$

且应满足以下约束条件:

$$\sum_{j=1}^{n}x_{0j}^k=\sum_{j=1}^{n}x_{j0}^k=1,\ k\in(1,2,\cdots,m); \tag{3}$$

$$\sum_{k=1}^{m}\sum_{\substack{j=0\\ j\neq i}}^{n}x_{ij}^k=1,\ i\in(1,2,\cdots,n); \tag{4}$$

$$\sum_{k=1}^{m}\sum_{\substack{i=0\\ i\neq j}}^{n}x_{ij}^k=1,\ j\in(1,2,\cdots,n); \tag{5}$$

$$\sum_{i=1}^{n}d_i\sum_{\substack{j=0\\ j\neq i}}^{n}x_{ij}^k\leqslant b_k,\ k\in(1,2,\cdots,m). \tag{6}$$

式(3)表示车辆必须从车场出发, 完成任务后返回车场; 式(4)和(5)表示每个客户都被访问一次且只能被访问一次; 式(6)表示每辆车对客户的服务不超过自身的负载量.

3 算法设计

3.1 蚁群算法求解CVRPs

为了提出求解CVRPs的改进蚁群优化算法, 先对利用传统的蚁群算法求解CVRPs的局限性进行分析.

利用蚁群算法求解CVRPs时, 一般将蚂蚁视为服务车辆, 信息素τ_{ij}表示车辆处于客户i时, 客户j对车辆的吸引程度. 在车辆负载满足的条件下, 车辆根据路径选择策略选择客户; 否则将返回车场, 重新选择客户. 当所有的客户选择完毕时, 算法进行局部搜索, 然后对信息素进行更新, 从而完成一次迭代. 经过数次迭代后, 可以得到一个优化解. 与其他智能算法相比, 蚁群算法体现出较高的性能, 但是在求解CVRPs时仍然存在以下局限性.

1) 在算法初始化中的局限性. 在CVRPs中, 由于车辆和客户都有负载要求, 使得车辆对客户访问存在一个数量和顺序问题, 初始客户的选取直接影响CVRPs的求解. 图1为一个CVRPs最终获得的最优路径, 定义在CVRPs获得的最优路径中, 与车场直接相连的点为关键节点, 如图1中黑色节点所示, 其他节点为非关键节点. 显然, 当车辆选取的初始客户处于非关键节点时, 它不可能走出如图1所示的路径.

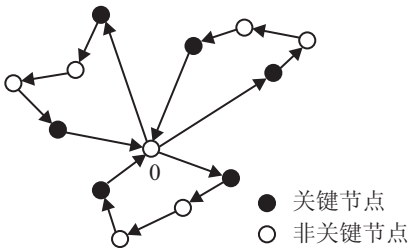

图1 CVRPs路径

使用蚁群算法求解CVRPs时, 车辆位置初始化方式目前有以下2种:

① 全部放置方式[6-7]. 该方式能够使车辆从每个客户开始遍历, 扩大了搜索的范围, 但是当处理较大规模问题时, 时间复杂度会大幅提高, 而且处于非关键节点上的车辆比处于关键节点上的车辆要多, 这会增加车辆在搜索过程中的相互干扰.

② 随机放置方式[8-10]. 该方式使用较少的车辆随机地放置在客户节点上, 虽然能够降低算法的时间复杂度, 但由于车辆的随机放置, 每次得到的当前迭代最优路径可能存在较大的差异, 这会减慢最优解的收敛速度.

2) 信息素更新过程的局限性. 目前, 求解CVRPs的蚁群算法使用的信息素更新方式有蚂蚁系统信息素的更新方式[6-7]和蚁群系统信息素的更新方式[8-10]等. 然而这些方法都普遍存在同样的问题, 即在迭代过程中, 当新的最优路径还未出现时, 当前最优路径上的信息素在更新策略的作用下会不断增强, 可能会产生两种消极的结果: ① 当前最优路径上信息素可能会因过度加强而导致算法停滞; ② 当新的最优路径出现时, 其路径上的信息素强度可能会远低于原最优

路径上的信息素强度, 有时经过数次迭代这种情况依然没有得到缓解. 总之, 目前的更新方式使当前最优路径的变化不能迅速地表现在路径的信息素分布上, 降低了搜索的效率.

3) 局部搜索过程的局限性. 2-opt 方法是求解 CVRPs 经常使用的一种方法, 文献 [6-7, 9] 将其应用于迭代后的每个车辆所构建路径的回路上, 文献 [8, 10] 将其应用于迭代后的当前最优路径的回路上. 然而通过实验分析发现, 在多数未达到最优的解中, 与最优解的差异并非产生在车辆访问的回路之中, 而是在回路之间.

3.2 算法改进

针对蚁群算法求解 CVRPs 的 3 种局限性, 本文提出 3 种新的策略对其加以改进, 路径选择使用蚁群系统 (ACS)[5]路径选择模型.

3.2.1 车辆位置初始化方式

将所有车辆 (车辆数量为 $e, e < n$) 置于车场 0, 在 t 时刻, 车辆 k ($k \in \{1, 2, \cdots, e\}$) 下一步可能选择的客户为 j, 其初始客户选择公式为

$$\begin{cases} j = \underset{y \in \text{allow}(0)}{\arg\max} \{\tau_{0y}(t)\},\ q \leqslant q_0; \\ p_{0j}^k = \dfrac{\tau_{0j}(t)}{\sum\limits_{y \in \text{allow}(0)} \tau_{0y}(t)},\ q > q_0. \end{cases} \tag{7}$$

其中: $\tau_{0j}(t)$ 表示 t 时刻边 $(0, j)$ 上的信息素值, p_{0j}^k 表示车辆 k 在车场 0 选中客户 j 的概率, allow(0) 表示车辆处于车场 0 时满足负载条件的候选客户集合, q 表示一个 $(0, 1)$ 的随机值, q_0 表示设定的阈值. 式 (7) 使用最优选择与随机选择相结合的方式初始化车辆位置, 车辆会以较大的概率 q_0 选择与车场相连信息素最大的客户, 而以 $(1 - q_0)$ 较小的概率按照随机比例原则选择初始客户. 该方法能够利用当前路径上信息素分布初始化车辆的位置. 经过实验发现, 在算法运行初期, 其效率与随机放置方式相当; 而随着迭代的进行, 路径上信息素正反馈的增强, 该方法会逐步引导车辆聚集到关键节点附近进行搜索.

3.2.2 路径选择方式

本文采用 ACS[5]路径选择模型进行路径选择, 启发因子 η_{ij} 为客户 (i, j) 间的路径节省量, 用来引导车辆向路径更为节省的客户移动, 有

$$\eta_{ij} = D_{i0} + D_{0i} - D_{ij}, \tag{8}$$

其中 D_{ij} 表示客户 i 到客户 j 的距离. 车辆在路径选择之前会筛选满足负载条件的客户作为候选, 如果没有满足条件的客户, 则返回车场. 当车辆 k 在 t 时刻处于客户 i 时, 下一步可能的客户为 j, 其路径选择公式为

$$\begin{cases} j = \underset{y \in \text{allow}(i)}{\arg\max} \{\tau_{iy}(t)\eta_{iy}^{\beta}(t)\},\ q \leqslant q_1; \\ {p_{ij}}^k = \dfrac{\tau_{ij}(t)\eta_{ij}^{\beta}(t)}{\sum\limits_{y \in \text{allow}(i)} \tau_{iy}(t)\eta_{iy}^{\beta}(t)},\ q > q_1. \end{cases} \tag{9}$$

其中 β 为权值系数, 用来决定启发因子的重要程度.

式 (9) 使车辆以较大的概率 q_1 向信息素和启发因子乘积最大的客户移动, 而以较小的概率 $(1-q_1)$ 使用随机比例原则选择客户. 这种方式既保证了车辆前进的方向性, 又增加了车辆搜索的多样性.

3.2.3 信息素更新方式

本文提出一种局部更新和全局动态更新相结合的信息素更新方式.

1) 局部更新.

在路径构建过程中, 车辆每经过一条边 (i, j) 都会利用下式来更新该边上的信息素:

$$\tau_{ij}(t+1) = (1 - \varepsilon)\tau_{ij}(t) + \varepsilon\tau_0. \tag{10}$$

其中: ε 为局部更新的蒸发率, τ_0 为信息素的初始值. 信息素局部更新的作用在于, 车辆每经过一条边 (i, j), 该边的信息素将会减少, 从而降低其他车辆选中该边的概率, 这将增加车辆探索未使用过边的机会.

2) 全局动态更新.

当一次迭代结束后, 将当前最优路径按下式进行信息素全局更新:

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}, \tag{11}$$

$$\Delta\tau_{ij} = \frac{L_1 - L_g}{L_g}. \tag{12}$$

其中: ρ 为全局更新的蒸发率, $\Delta\tau_{ij}$ 为全局更新信息素增量, L_1 为当前迭代最优路径长度, L_g 为当前最优路径长度. 更新当前最优路径上的信息素在于对最优路径的继续开发, 将当前最优路径对信息素的正反馈保留到下一次的迭代中, 直到有更优的路径取代为止.

式 (11) 和 (12) 能够根据车辆所构建路径的情况动态地调整当前最优路径上的信息素. 在迭代过程中, 当一条更优路径出现后, L_1 和 L_g 的差值开始可能会比较大, 这说明当前最优路径与大多数车辆走出的路径存在较大的差异, 走出当前最优路径的车辆所占比例很小. 此时, 式 (12) 能够加大当前最优路径上信息素的强度, 以吸引更多的车辆. 而随着迭代的进行, 当前最优路径上的信息素不断增强, 更多的车辆会聚集到这条路径上来, L_1 和 L_g 的差值会逐渐减小, 此时, 式 (12) 能够相应地减小信息素增量, 直至减小到 0. 当式 (12) 减小到 0 时, 当前最优路径上只进行信息素蒸发, 对信息素进行削弱. 该方法能够使当前最优路径上的信息素强度更为突出, 但又不会因过度加强而造

成算法的停滞, 使当前最优路径的变化更快地反映在信息素的分布上.

3.2.4 局部搜索策略

为了扩大搜索的范围, 加快最优解的收敛速度, 本文提出了插入搜索和交换搜索相结合的搜索方式. 当完成一次迭代后, 对每个车辆所构建的路径的回路进行 2-opt 搜索. 如果连续 N 次迭代当前最优路径长度仍然没有发生改变, 则对当前最优路径进行插入搜索和交换搜索.

1) 插入搜索.

将当前最优路径上的客户依次插入到其他车辆所访问的路径中, 每次插入前都要进行负载审核, 即验证插入后车辆是否超出车辆的负载限制. 如果审核通过, 则对新的回路进行 2-opt 搜索, 并记下插入前后的路径节省量; 否则, 将客户插入到下一个车辆访问的路径中. 当所有的客户都完成插入搜索后, 选取节省量最大的路径作为当前最优路径, 某次迭代使用插入搜索的过程如图 2 所示. 该图示意了客户 b 由回路 A 中转移至回路 B 中的插入搜索过程.

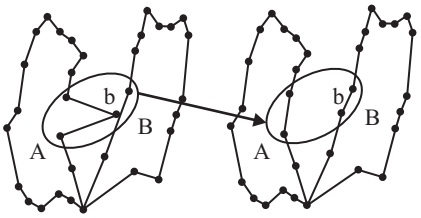

图 2 插入搜索过程

2) 交换搜索.

将当前最优路径上的客户与其他车辆所访问的客户进行交换, 每次交换前要进行负载审核, 如果审核通过, 则对新的回路进行 2-opt 搜索, 并记下交换前后的路径节省量; 否则, 与下一个客户进行交换. 当所有客户都完成交换搜索后, 选取节省量最大的路径作为当前最优路径, 某次迭代使用交换搜索的过程如图 3 所示. 图 3 示意了客户 s 与客户 t 交换了归属的回路交换搜索过程.

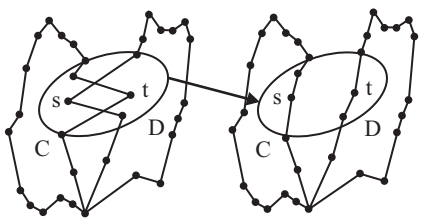

图 3 交换搜索过程

在本文算法中, 2-opt 方法主要应用于单回路搜索, 而插入搜索与交换搜索则应用于回路之间. 由实验结果发现, 与原先的方法相比, 本文方法虽然会消耗更多的时间, 但是它能够扩大搜索范围, 加快最优解的收敛速度.

3.3 算法步骤

根据以上改进策略的描述, 改进算法的步骤如下.

Step 1: 参数初始化. 令迭代计数器 NC $=0$, 设置当前最优路径长度 shortest_len, 最大的迭代次数 NC_CONST, 距离 length$[i][j](i,j=0,1,\cdots,n)$, 客户服务需求量 load$[j](j=1,2,\cdots,n)$, 当前最优路径表 tabu_min, 计算信息素初值 τ_0 和客户间节省量 η_{ij} $(i,j=1,2,\cdots,n)$. 将 e 个车辆放置在车场, 初始化每个车辆的禁忌表 ant$[k]$.tabu$(k=1,2,\cdots,e)$, 所走路径长度 ant$[k]$.length$(k=1,2,\cdots,e)$, 车辆的负载量 ant$[k]$.load$(k=1,2,\cdots,e)$ 以及路径上的信息素 τ_{ij} $(i,j=1,2,\cdots,n)$.

Step 2: 车辆位置初始化. 所有车辆按式 (7) 进行客户选择, 将所选客户添加到 ant$[k]$.tabu 中, 并更新 ant$[k]$.length 和 ant$[k]$.load 的值.

Step 3: 路径选择. 如果满足负载条件, 则车辆按式 (9) 进行客户选择; 否则车辆返回车场. 将所选客户或车场添加到 ant$[k]$.tabu 中, 更新 ant$[k]$.length 和 ant$[k]$.load 的值.

Step 4: 信息素局部更新. 车辆每做出一次选择, 便根据式 (10) 对刚刚走过的路径 (i,j) 进行信息素局部更新.

Step 5: 2-opt 局部搜索. 当所有车辆都完成客户服务, 路径构建完毕时, 对每辆车所构建的路径进行 2-opt 搜索, 并更新 ant$[k]$.length 和 ant$[k]$.tabu; 否则返回Step 3.

Step 6: 插入搜索与交换搜索. 如果连续 N 次迭代当前最优路径没有发生变化, 则对当前最优路径进行插入搜索和交换搜索, 更新shortest_len和tabu_min.

Step 7: 信息素全局动态更新. 统计当前迭代中最短路径, 将 ant$[k]$.length与shortest_len 进行比较. 若 ant$[k]$.length $<$ shortest_len, 则用 ant$[k]$.length 替换 shortest_len, ant$[k]$.tabu 替换 tabu_min. 根据式 (11) 和 (12) 对当前最优路径进行信息素全局更新.

Step 8: 迭代循环. 若 NC $\leqslant$ NC_CONST, 则返回 Step 2, 开始新一次的迭代; 否则算法结束, 输出最优路径 shortest_len 和最优路径表 tabu_min.

4 仿真示例

从车辆路由公共数据集[12]中选择标准算例进行测试, 使用 BCB 和 Matlab 进行编程, 并在 CPU 为 Intel Celeron 1.0 G, 内存 512 M 的计算机上运行. 实验参数为: $\rho=0.1$, $\varepsilon=0.1$, $\beta=2$, $e=15\sim30$, $q_0=0.8$

~ 0.95, $q_1 = 0.7 \sim 0.9$, $N = 40$.

为了验证改进方法的有效性, 首先, 对算法使用3种不同的初始化方式进行性能测试, 测试算例为E-n22-k4, E-n51-k5和E-n76-k10, 其收敛曲线如图4所示.

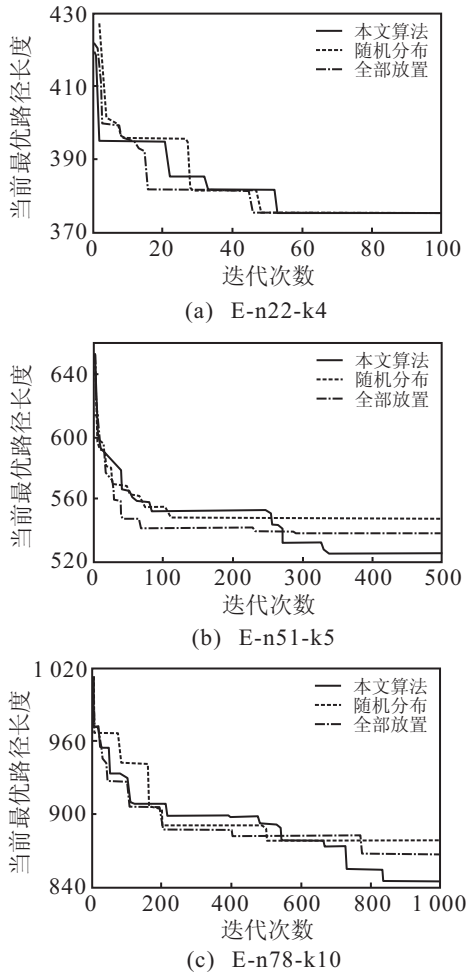

图 4　3种初始化方式性能比较

从图4可知, 在算法执行初期, 采用全部放置方式得到的解优于随机放置方式和本文方法, 这主要归因于它对所有客户的搜索, 此时由于路径上信息素分布较为平均, 而本文方法主要依赖于路径上信息素的积累, 故算法效果并不明显. 但随着迭代的进行, 最优路径上的信息素不断增强, 本文方法会引导车辆从关键节点附近的客户出发进行搜索; 对于全部放置方式, 由于从非关键节点出发的车辆较多, 他们之间的相互干扰会不断增加, 特别是在求解规模较大的算例时, 其寻优能力会大幅下降, 如图4(b)和图4(c)所示; 而采用随机放置方式, 由于车辆位置的随机放置, 每次迭代最优路径可能存在较大差异, 这会减慢最优解的收敛速度. 总之, 在处理较小规模问题时, 3种方法的效率基本相当, 全部放置方式的收敛速度更为快速; 而处理较大规模问题时, 本文方法会体现出较好的性能.

此外, 将本文算法与文献[7]中的更新方式进行性能对比, 测试算例为E-n22-k4, E-n51-k5和E-n76-k10, 其仿真曲线如图5所示. 其中: 图5(a)中的两种更新方式最后都得到了最优解, 但本文方法的收敛速度比文献[7]的略快; 由图5(b)和图5(c)可以看出, 本文方法在求解精度和收敛速度方面均优于文献[7]的方法.

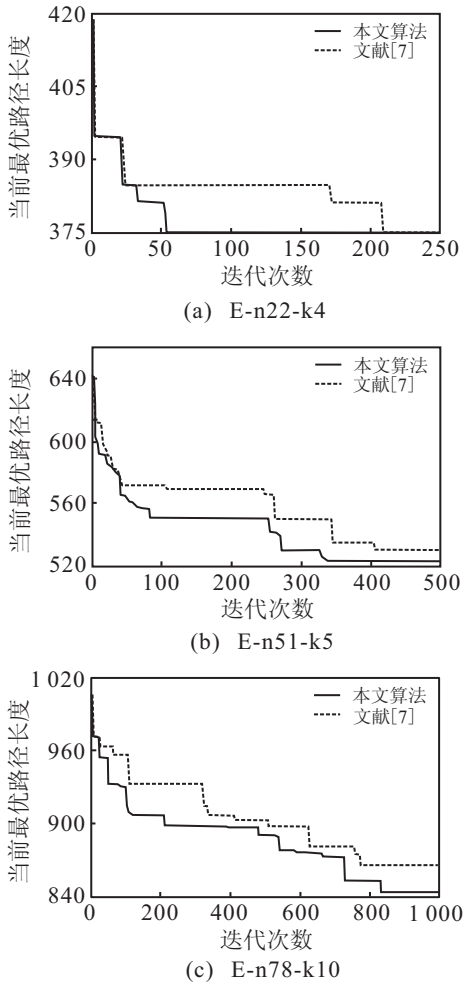

图 5　两种信息素更新方式性能比较

从算法本身分析, 文献[7]方法的原理是使当前最优路径上的信息素不断增强, 直到更优的路径出现为止, 当前最优路径上信息素可能会因过度加强而导致算法停滞. 本文方法能够根据车辆的搜索结果动态地调配最优路径上的信息素: 在当前迭代最优路径长度与当前最优路径长度相异时会增强最优路径上信息素; 相等时则会对其进行削弱.

表1列出了本文算法对部分算例的计算结果, 每个算例运行10次. 其中: Cu表示客户的数量, V 表示车辆的数量, BK表示迄今为止已知的最优解, Result表示本文算法得到的最优解, RPD表示本文算法最优解与BK偏移率, Average表示本文算法运行10次的平均解, Time表示得到最优解消耗的时间. 由表1可知, 本文算法求解全部的12个算例中, 有6个达到

表 1 本文算法计算结果

Problem	Cu	V	BK	Result	RPD/%	Average	Time/s
P-n16-k8	15	8	**451.35**	451.35	0.00	457.34	1.88
P-n19-k2	18	2	**212.65**	212.65	0.00	215.92	3.23
P-n20-k2	19	2	**217.41**	217.41	0.00	220.80	2.28
P-n21-k2	20	2	**212.71**	217.48	2.24	222.46	5.10
P-n23-k8	22	8	**531.17**	531.17	0.00	532.70	3.65
P-n40-k5	39	5	**461.72**	470.31	1.68	472.39	13.51
E-n22-k4	21	4	**375.28**	375.28	0.00	381.62	6.07
E-n51-k5	50	5	**524.61**	524.61	0.00	524.92	18.26
E-n76-k10	7	10	**835.26**	844.31	0.11	847.63	88.33
E-n101-k8	100	8	**826.14**	835.64	1.14	842.76	192.37
E-n151-k12	150	12	**1028.42**	1 038.22	1.05	1 051.90	837.41
E-n200-k17	199	17	**1291.45**	1 327.07	2.76	1 351.54	3 080.82

最优解, 最优解的偏移率最大为 2.76%.

此外, 将本文算法与其他算法进行比较, 性能对比如表 2 所示. 其中: Distance 表示算法取得的最优路径长度, Average 表示对应表项的平均值. 由表 2 可知, 本文算法的平均偏移率为 1.01%, 在所有算法中仅次于 HGA, 相差 0.16%. 在计算的 5 个算例中, 本文算法对其中 3 个算例都得到了最优的结果. 在时间消耗方面, 本文算法求解前 2 个算例的时间要长于 AS, 其余 3 个算例均是所有算法中时间消耗最少的, 而且本文算法的平均时间消耗也是所有算法中最少的, 大约是文献 [2] 算法的一半.

表 2 算法性能对比

algorithm	items	E-n51-k5 BK=524.61	E-n76-k10 BK=835.26	E-n101-k8 BK=826.14	E-n151-k12 BK=1 028.42	E-n200-k17 BK=1 291.45	Average
GA[2]	Distance	524.81	849.77	840.72	1 055.85	1 387.73	–
	RPD/%	0.04	1.74	1.76	2.76	6.76	2.61
	Time/s	213	765	1 148	2 475	3 999	1 720
HGA[2]	Distance	524.61	838.89	829.47	1 034.8	1 327.78	–
	RPD/%	0	0.43	0.4	0.62	2.81	0.85
	Time/s	23	617	717	1 961	5 261	1 715.8
MAS[10]	Distance	524.61	–	840.2	1 094.8	–	–
	RPD/%	0	–	1.7	6.45	–	–
	Time/s	99	–	450	1 395	–	–
AS[7]	Distance	524.61	844.31	832.32	1 061.55	1 343.55	–
	RPD/%	0	1.08	0.75	3.22	4.03	1.82
	Time/s	6	78	228	1 104	5 256	1 334.4
SR-GCWS[11]	Distance	524.61	836.42	–	–	–	–
	RPD/%	0	0.14	–	–	–	–
	Time/s	32	21 707	–	–	–	–
本文算法	Distance	524.61	836.18	835.64	1 038.22	1 327.07	–
	RPD/%	0	0.11	1.14	1.05	2.76	1.01
	Time/s	18.26	88.33	192.37	837.41	3 080.82	843.44

由以上实验结果可以看出, 与其他算法相比, 本文算法体现出良好的性能, 并且在问题规模较大的情况下, 解的质量也显示了较高的稳定性.

5 结 论

本文提出了一种基于蚁群算法的改进优化方法, 并将其应用于求解带容量限制的车辆路由问题. 通过对标准问题的求解以及与其他方法的比较, 本文算法均体现出了良好的性能. 此外, 该算法也可以用于其他相关类型的优化问题.

参考文献(References)

[1] Laporte G, Mercure H, Nobert Y. A branch and bound algorithm for a class of asymmetrical vehicle routing problem[J]. The J of Operational Research Society, 1992, 43(5): 469-481.

[2] Baker B M, Ayechew M A. A genetic algorithm for the vehicle routing problem[J]. Computers and Operations Research, 2003, 30(5): 787-800.

[3] Wang C H, Lu J Z. A hybrid genetic algorithm that optimizes capacitated vehicle routing problems[J]. Expert Systems with Applications, 2009, 36(2): 2921-2936.

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

[5] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach for the traveling salesman problem[J]. IEEE Trans on Evolutionary Computation, 1997, 1(1): 53-66.

[6] Bullnheimer B, Hartl R F, Strauss C. Applying the ant system to the vehicle routing problem[C]. Advances and Trends in Local Search Paradigms for Optimization. Boston: Kluwer, 1998: 109-120.

[7] Bullnheimer B, Hartl R F, Strauss C. An improved ant system for the vehicle routing problem[J]. Annals of Operations Research, 1999, 89: 319-328.

(下转第1643页)