第 27 卷第 11 期
2010 年 11 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 27 No. 11
Nov. 2010

文章编号: 1000−8152(2010)11−1557−07

时间依赖型车辆路径问题的一种改进蚁群算法

段征宇, 杨东援, 王　上

(同济大学 交通运输工程学院, 上海 200092)

摘要: 时间依赖型车辆路径规划问题(TDVRP), 是研究路段行程时间随出发时刻变化的路网环境下的车辆路径优化. 传统车辆路径问题(VRP)已被证明是NP-hard问题, 因此, 考虑交通状况时变特征的TDVRP问题求解更为困难. 本文设计了一种TDVRP问题的改进蚁群算法, 采用基于最小成本的最邻近法(NNC算法)生成蚁群算法的初始可行解, 通过局部搜索操作提高可行解的质量, 采用最大–最小蚂蚁系统信息素更新策略. 测试结果表明, 与最邻近算法和遗传算法相比, 改进蚁群算法具有更高的效率, 能够得到更优的结果; 对于大规模TDVRP问题, 改进蚁群算法也表现出良好的性能, 即使客户节点数量达到1000, 算法的优化时间依然在可接受的范围内.

关键词: 时间依赖型车辆路径规划问题; 蚁群算法; 最邻近算法

中图分类号: U491　　**文献标识码:** A

Improved ant colony optimization algorithm for time-dependent vehicle routing problem

DUAN Zheng-yu, YANG Dong-yuan, WANG Shang

(School of Transportation Engineering, Tongji University, Shanghai 200092, China)

Abstract: Time-dependent vehicle routing problem (TDVRP) is concerned with vehicle routing optimization in road networks with fluctuant link travel time. The traditional vehicle routing problem (VRP) has been proven to be an NP-hard problem, so it is difficult to solve TDVRP in considering traffic conditions. We design an improved ant colony optimization algorithm (ACO) for TDVRP. It uses nearest neighbor algorithm based on minimum cost(NNC algorithm) to generate the initial solution, improves feasible solution by local search operations, and updates pheromone with max/min ant system strategy. Test results show that compared with the nearest neighbor algorithm and genetic algorithm, the improved ACO algorithm is more efficient and able to get better solutions. Furthermore, this improved ACO algorithm show good performance in large scale TDVRP instances, even if the customer number of TDVRP reaches 1000, the computation time is still in an acceptable range.

Key words: time-dependent vehicle routing problem; ant colony optimization algorithm; nearest neighbor algorithm

1　引言(Introduction)

1.1　时间依赖型车辆路径规划问题简介(Brief introduction of TDVRP)

以往的VRP研究大多是基于静态路网的假设, 即路段行程时间被认为是静态的, 在路径的制定和执行过程中不会发生变化. 但在实际路网中, 由于受到交通流量、天气、突发事故等因素的影响, 路段行程车速, 行程时间往往经常发生变化. 在动态路网中, 两节点的行程时间不仅与它们之间距离有关, 还依赖于出发时间[1].

时间依赖型VRP问题(time dependent vehicle routing problem, TDVRP), 是研究路段行程时间随出发时间变化的路网环境下, 如何安排车辆配送路线, 以达到一定的优化目标(如路程最短、费用最少、时间最少、使用车辆最少等)[1,2].

传统VRP问题已被证明是NP-hard问题, 而TDVRP问题比传统VRP问题更复杂, 求解也更为困难. 随着现代物流的发展, 城市交通负载的日益繁重, 传统的VRP模型难以满足快捷, 高效配送的要求, 因此, 研究TDVRP问题十分必要.

1.2　研究回顾(Literature review)

TDVRP问题的研究是由时间依赖型旅行商问题(time dependent traveling salesman problem, TDTSP)发展而来, Bowman首次提出了TDTSP问题[3], 直到1978年才由Picard等第一次提出精确解法. Malandrak等[4]作出重要贡献, 给出了TDVRP的严格数学模型, 并设计了最邻近法和割平面启发式算法, 可以处理10∼25个客户节点的问题. Soojung Jung[2]在

收稿日期: 2009−11−07; 收修改稿日期: 2010−03−15.
基金项目: 国家自然科学基金重点资助项目(50738004); 国家“863”计划资助项目(2007AA11Z245).

其博士论文中系统研究了TDVRP问题, 设计了一种遗传算法, 求解了30个节点规模的TDVRP问题. Ichoua等[5]将“先入先出”(FIFO)准则引入TDVRP问题, 采用速度时间依赖函数表示动态路网, 解决了Malandraki的模型需要在节点处等待的不足; 采用改进并行禁忌搜索算法, 求解了100个节点规模的带时间窗TDVRP问题. Donati等[6,7]设计了TDVRP问题的多蚁群系统优化算法, 构建了具有5种道路类型, 4个时段函数的动态路网, 对100个节点规模带时间窗的TDVRP问题进行了求解.

总的看来, TDVRP问题研究还处于初级阶段, 面向大规模网络的高效算法是目前的研究热点.

1.3 动态路网的表示(Description of dynamic network)

本文采用Ichoua等[5]的基于行程速度的时间依赖函数方法, 来表示动态路网. Ichoua的模型如图1所示, 把路段行程速度表示为分段函数的形式(图a), 由此得到连续的路段行程时间函数(图b), 使得路网满足FIFO特性.

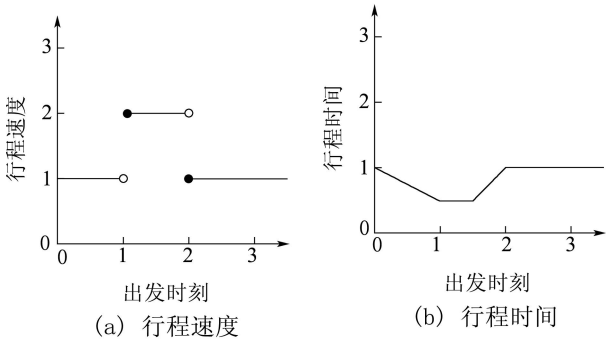

图 1 时间依赖型行程函数[5]

Fig. 1 Time-dependent travel function [5]

1.4 TDVRP 问题的数学模型(Mathematical model for TDVRP)

本文研究的是硬时间窗TDVRP问题, 也就是不能违反客户的时间窗要求. TDVRP问题是一个多目标优化问题, 在本文中, 优化目标的优先顺序为: 车辆数量, 总配送时间, 总行程时间, 总距离, 总剩余容量. 在优化计算时, 需要把多个目标转化为一个目标. 在这里采用加权求和的方式, 重点考虑“车辆数量”和“总配送时间”. 在本文中, 认为“车辆数量”比“总配送时间”更重要, 即采用更少“车辆数量”的配送方案即使“总配送时间”较大, 本文也认为这个方案较优.

下面给出TDVRP问题的一种混合整数规划模型[1,2,4]:

$$\min z = \alpha_1 \cdot K + \alpha_2 \cdot ST, \tag{1}$$

其中:

$$ST = TT + WT + SVT,$$

$$TT = \sum_{i\in SD}\sum_{\substack{j\in DE\\ j\neq 1}}\sum_{m=1}^{M}(R_{ij}^m \times x_{ij}^m),$$

$$WT = \sum_{i\in D}\max(e_i - t_i, 0),$$

$$SVT = \sum_{i\in D} svt_i,$$

s.t.

$$\sum_{\substack{i\in SD\\ j\neq i}}\sum_{m=1}^{M} x_{ij}^m = 1,\ j \in DE, \tag{2}$$

$$\sum_{\substack{i\in DE\\ j\neq i}}\sum_{m=1}^{M} x_{ij}^m = 1,\ i \in SD, \tag{3}$$

$$\sum_{k=1}^{K} k(v_{ik} - v_{jk}) \leqslant B(1 - \sum_{m=1}^{M} x_{ij}^m),$$
$$i \in SD,\ j \in DE,\ i \neq j, \tag{4}$$

$$\sum_{k=1}^{K} k(v_{ik} - v_{jk}) \geqslant B(\sum_{m=1}^{M} x_{ij}^m - 1),$$
$$i \in SD,\ j \in DE,\ i \neq j, \tag{5}$$

$$\sum_{k=1}^{K} v_{ik} = 1,\ i \in D, \tag{6}$$

$$\sum_{i\in D}(q_i \times v_{ik}) \leqslant Q_k,\ k = 1, 2, \cdots, K, \tag{7}$$

$$\sum_{i\in S}\sum_{j\in DE}\sum_{m=1}^{M} x_{ij}^m = \sum_{i\in SD}\sum_{j\in E}\sum_{m=1}^{M} x_{ij}^m, \tag{8}$$

$$t_j \leqslant (\max(e_i, t_i) + svt_i + R_{ij}^m) + B(1 - x_{ij}^m), \tag{9}$$

$$t_j \geqslant (\max(e_i, t_i) + svt_i + R_{ij}^m) + B(x_{ij}^m - 1), \tag{10}$$

$$\sum_{\substack{i,j\in SC\\ i\neq j}}\sum_{m=1}^{M} x_{ij}^m \leqslant |SC| - 1,\ SC \in V. \tag{11}$$

符号的含义:

1) 常量.

α_1, α_2: 加权系数, 在本文中, 认为“车辆数量”比“总配送时间”更重要, 因此, 将α_1设置为比α_2更大的值, 具体取值将在后面的算例中给出;

$[0, T]$: 配送中心的工作时间;

$[e_i, l_i]$: 客户节点i要求的时间窗;

Q_k : k号车的载重量;

M: 出发时间分段数;

B: 无穷大的数, 处理为一个足够大的数;

2) 集合.

S: 始发节点集合;

E: 终到节点集合;

D: 需求节点集合;

SD: 始发节点与需求节点的集合;

DE: 需求节点与终到节点集合;

SDE: 所有节点的集合;

V: 客户节点集合;

SC: 是客户节点集合V的任一子集, $|SC|$ 表示SC所包含客户的数量.

3) 决策变量.

K: 使用的车辆数;

ST: 总配送时间, 即总行程时间, 总等待时间和总服务时间之和;

TT: 总行程时间, 即所有车辆的路径行程时间之和;

WT: 总等待时间, 即所有车辆的配送等待时间之和;

SVT: 总服务时间, 即所有车辆的配送服务时间之和;

x_{ij}^m: 若车辆在m时段内出发, 从i驶向j, 值为1; 否则为0;

$t_i(i \in SDE)$: 配送车辆到达i点的时刻;

svt_i: 客户节点i的服务时间;

q_i: 客户节点i的配送需求量;

v_{ik}: 若i点由k号车访问, 值为1; 否则为0;

R_{ij}^m: 在m时段内从i出发到j的路段行程时间, 其中: $i \in SD$, $j \in DE$, $i \neq j$.

如式(1)所示, 目标函数由2部分组成:

1) 使用车辆数量的加权值: $\alpha_1 \cdot K$;

2) 总配送时间的加权值: $\alpha_2 \cdot ST$, ST由总行程时间(TT), 总等待时间(WT)和总服务时间(SVT)求和得到.

约束条件可以分为5部分:

1) 需求约束: 要求每个客户由一辆车服务, 且仅服务一次如式(2)(3);

2) 对车辆的约束: 同一条路线上的客户由同一辆车服务如式(4)(5), 每个客户只能由一辆车服务如式(6), 车辆的总装载量不能超过该车的容量如式(7);

3) 路径约束: 出发车辆数与返回车辆数相等, 即车辆从始点出发, 完成配送任务后全部回到终点如式(8);

4) 时间窗约束: 到达某客户的时刻由上一个客户的到达时刻, 服务时间和路段行程时间确定如式(9)(10);

5) 奇异子回路排除约束: 防止在配送路径中形成子回路如式(11).

2 TDVRP问题的蚁群算法设计(ACO algorithm for TDVRP)

2.1 算法步骤(Algorithm process)

下面给出一种TDVRP问题的蚁群算法(ant colony optimization, ACO), 步骤如下:

Step 1　初始化.

设定蚁群算法参数, 通过随机方法或构造算法生成初始可行解, 设定更新每条路段的初始信息素强度值 $\tau_{ij}(0)$. 然后, 采用“蚁周模型”(ant-cycle model)更新规则[8], 对各路段的信息素进行更新.

在本文中, 采用NNC算法(见下节)产生一定数量的高质量可行解, 然后取最优解作为蚁群算法的初始解.

Step 2　通过蚂蚁移动, 构造配送路径, 并进行局部更新信息素.

每只蚂蚁从配送中心出发, 根据转移概率选择下一个节点, 若满足容量和时间窗约束条件, 则选择该节点, 并将该节点记录到禁忌表中; 否则, 返回配送中心, 重新开始一条新的路径, 直至所有客户节点都被访问, 该蚂蚁就完成了一次周游.

转移概率由“伪随机比例规则”[8,9]确定, 其优点是既可以利用先验知识, 又可以进行有倾向性的探索. 局部更新, 是指蚂蚁从当前节点移动到下一个节点, 便对经过的路段进行一次信息素更新.

Step 3　局部搜索操作.

在每只蚂蚁构造完成配送路径以后, 对配送路径进行局部搜索操作. 局部搜索操作采用: 2-opt方法、or-opt方法、一条路径内的节点移动法和λ-interchange方法, 对可行解进行邻域搜索, 以获得更优的解.

Step 4　全局更新信息素.

全局更新, 是在所有蚂蚁都完成周游以后, 对最优路径的组成路段进行信息素更新.

Step 5　判断终止条件.

判断是否达到预定的最大迭代次数, 若符合, 则进入下一步; 否则, 返回到Step 2.

Step 6　输出最优解.

输出最优解, 并结束计算.

为了避免蚁群算法出现“过早收敛”的问题, 本文采用“最大–最小蚂蚁系统”[8,10] 的信息素更新策略, 将路段信息素量限制在[$\tau_{\min}$, $\tau_{\max}$]以内.

$$\tau_{\max} = \frac{1}{f(s^{\text{opt}})}. \tag{12}$$

其中$f(s^{\rm opt})$为每次循环得到的最优路径的总配送时间, 即$f(s^{\rm opt})={\rm ST}$, 也就是一个动态更新的$\tau_{\max}$.

$\tau_{\min}$由式(13)定义如下:

$$\tau_{\min}=\frac{\tau_{\max}(1-\sqrt[n]{P_{\rm best}})}{(n/2-1)\sqrt[n]{P_{\rm best}}}. \tag{13}$$

其中: n为客户节点数, $P_{\rm best}$为选择最优解的概率.

若已知$P_{\rm best}$就能够求得$\tau_{\min}$, Stüzle[10]通过TSP问题算例的计算表明, $P_{\rm best}$较大时, 即$\tau_{\min}$较小时, 能获得较好的结果, 在本文中, 取$P_{\rm best}=0.05$.

2.2 NNC算法(NNC algorithm)

基于最小成本的最邻近法(Nearest neighbor algorithm base on minimum cost, NNC算法)是一种构造算法, 可以产生比较高质量的可行解. 最早由Solomon[11]提出并应用于VRP问题, 下面将其扩展到TDVRP问题, 算法步骤如下:

Step 1 对于给定出发时间, 在可用车辆中选派第一辆车, 从配送中心出发.

Step 2 选择“距离”最后一次访问节点最近的未访问节点, 若满足约束条件, 则将该节点插入到当前路径中.

Step 3 重复Step 2, 若已到达当前车辆的容量限制, 则增派一辆车; 若所有节点都已访问, 则结束计算.

节点之间的“距离”定义为: 两节点的行程时间, 时间窗的接近程度, 后一个节点时间窗紧迫性的加权求和. 也可以将这里的“距离”, 看作节点的“插入成本”.

“时间窗的接近程度”是指: 后一个节点的“开始服务时间”与前一个节点的“完成服务时间”之差.

节点的“时间窗紧迫性”是指: 节点的“最迟服务时间”与节点的“开始服务时间”之差.

那么, 两节点间的“距离”由式(14)计算:

$$c_{ij}=\delta_1 t_{ij}+\delta_2 T_{ij}+\delta_3 E_{ij}, \tag{14}$$

其中:

c_{ij}: 节点i,j之间的“距离”;

t_{ij}: 从节点i到j的行程时间;

T_{ij}: 节点i,j的时间窗的接近程度;

E_{ij}: 节点j的时间窗紧迫性;

$\delta_1,\delta_2,\delta_3$: 加权系数, 满足$\delta_1+\delta_2+\delta_3=1$.

T_{ij}由式(15)得到

$$T_{ij}=b_j-(b_i+s_i), \tag{15}$$

其中: b_i为点i的开始服务时间, s_i为节点i的服务时间.

E_{ij}由式(16)得到

$$E_{ij}=l_j-(b_i+s_i+t_{ij}), \tag{16}$$

其中l_j为节点j的最迟服务时间.

2.3 局部搜索算法(Local search algorithms)

局部搜索(local search)是通过对已有可行解邻域的搜索, 来获得更优的可行解的过程[12]. 下面简要介绍蚁群算法中所采用的几种局部搜索算法:

2-opt方法: 将原路径中的2条边删除, 然后, 用另外2条边代替, 以获得更优的路径.

or-opt方法: 将原路径中的某个子路径移动到新的位置, 以获得更优的路径.

一条路径内的节点移动法: 将原路径的某节点移动到新的位置, 以获得更优的路径.

λ-interchange方法: 从两条路径中各选择一条子路径, 子路径长度不超过λ, 然后进行互换, 以获得更优的路径.

3 算法参数的标定(Calibration of algorithm parameters)

3.1 测试算例的构造(Construction of test problems)

对于TDVRP问题, 目前还没有标准的基准算例, 本文对传统VRP问题的Solomon基准算例进行修改, 增加“速度时间依赖函数”, 得到TDVRP问题的测试算例.

参照文献[7]的做法, 定义速度时间依赖函数如下: 采用阶梯函数表示, 分为5类函数, 分别代表5种不同类型的道路, 每类函数分为4个等分时段, 表示路网的早晚高峰特征. 为了保证修改后的算例满足原来的时间窗设计, 将各种类型道路的速度最小值设为1. 由此, 得到速度时间依赖函数如表1所示.

表 1 速度时间依赖函数

Table 1 Time-dependent function of travel speed

类型	v_1	v_2	v_3	v_4
1	1.80	2.20	1.60	2.40
2	1.60	2.40	1.80	2.20
3	1.40	2.60	1.00	3.00
4	1.20	2.80	1.40	2.60
5	1.00	3.00	1.20	2.80

各路段所属的类型由式(17)确定, 这种方法可以使得各类道路在网络中均匀分布.

$$\mathrm{Grade}(i,j)=\mathrm{Mod}((i+j),5). \tag{17}$$

其中: Grade(i, j)为路段(i, j)的道路类型, Mod(a, b)为求a除以b得到的余数.

3.2 测试环境(Test environment)

本文的测试环境如下(台式机):

CPU: Intel Core 2 Duo(双核), 2.20 GHz; 内存: 2 GB; 操作系统: Windows Server 2003; 编程语言: C#(Microsoft.Net平台).

3.3 算法参数的确定(Algorithm parameters value)

蚁群算法需要确定的参数有: 信息强度参数α, 能见度参数β, 信息素挥发系数ρ, 伪随机比例规则q_0, 蚂蚁数量m.

采用试算法确定蚁群算法的最优参数, 即每次给出某参数的一组取值, 固定其他参数, 根据算例的计算结果确定该参数的最优取值. 测试算例选取Solomon基准中的C101, C201, R101, R201, RC101, RC201. 对于目标函数(1), 取$\alpha_1 = 10000$, $\alpha_2 = 1$. 得到各参数的取值如表2所示, 其中n为客户节点数.

表 2　蚁群算法的参数取值

Table 2　Parameters value of ACO

	α	β	ρ	q_0	m
取值	1	2	0.9	0.2	n/10

4 算法性能分析(Analysis of algorithm performance)

4.1 初始解的影响(Impact of initial solution)

分别采用随机方法, NNC算法生成初始解, 对RC101算例进行计算, 得到表3所示的结果. 为考察初始解的影响, 这里不采用局部搜索操作, 迭代次数为200次.

由表可见, 基于NNC算法初始解的蚁群算法, 计算结果比基于随机初始解的蚁群算法有较大提高, 这说明高质量的初始解有利于加快算法的收敛速度.

表 3　初始解对蚁群算法的影响

Table 3　Impact of initial solution to ACO

初始解生成	车辆数	配送时间	行程时间	距离	剩余容量	计算时间/s
随机	16	2,819.15	1,130.55	2,086.35	1,476	12.90
NNC	12	2,307.97	1,090.38	1,983.40	676	14.00

4.2 局部搜索操作的影响(Impact of local search operations)

下面考察局部搜索操作对蚁群算法的影响, 分别基于随机初始解, NNC算法生成的初始解, 在采用局部搜索操作和不采用局部搜索操作两种情况下, 对RC101算例进行计算, 迭代次数为200次, 得到表4所示的结果.

由表可见, 对于随机初始解, 采用局部搜索操作后, 可行解得到较大改进; 而NNC法生成的初始解, 尽管初始解的质量较好, 采用局部搜索操作后, 结果仍能得到进一步优化. 采用局部搜索操作后, 算法的计算时间增加了约80%.

表 4　局部搜索操作对蚁群算法的影响

Table 4　Impact of local search operations to ACO

初始解生成	局部搜索	车辆数	配送时间	行程时间	距离	剩余容量	计算时间/s
随机	否	16	2,819.15	1,130.55	2,086.35	1,476	12.90
	是	13	2,435.61	1,059.10	1,961.61	876	23.45
NNC	否	12	2,307.97	1,090.38	1,983.40	676	14.00
	是	12	2,272.43	1,057.18	1,907.08	676	23.80

5 测试算例(Experimental results)

5.1 无时间窗TDVRP算例(Instances of TDVRP without time windows)

文献[1]给出了一个无时间窗TDVRP问题的算例, 包括20个客户节点, 配送中心具有时间窗约束, 路段行程成本以5个等分时段的阶梯函数形式描述. 原文设计了一种免疫遗传算法进行求解, 考察了车辆出发时间可变情况下的配送路径优化[1].

本文将采用蚁群算法对该算例进行求解, 在这里只考虑出发时刻为0的配送路径优化, 不考虑出

发时间可变的情况. 对于目标函数式(1), 由于优化后的“配送成本”的值域一般为[0, 1000], 所以: 取$\alpha_1 = 1000$, $\alpha_2 = 1$.

计算结果如表5所示(迭代次数为200次), 由表可见, 本文的结果在配送成本, 路径距离上均优于原文的结果, 其中“配送成本”降低了6.19%. 此外, 蚁群算法的计算时间仅为2.31 s, 这说明算法具有良好的性能.

表 5 无时间窗TDVRP问题算例的计算结果

Table 5 Experimental results of TDVRP instances without time windows

	车辆数量	配送成本	距离	剩余容量	计算时间/s
本文	6	336.38	1153.19	195	2.31
文献[1]	6	358.56	1171.61	195	—

本文得到的优化配送路径如下:

$0 \to 8 \to 10 \to 4 \to 20 \to 0$

$0 \to 7 \to 1 \to 11 \to 15 \to 0$

$0 \to 14 \to 2 \to 18 \to 17 \to 0$

$0 \to 16 \to 19 \to 6 \to 0$

$0 \to 5 \to 12 \to 3 \to 0$

$0 \to 9 \to 13 \to 0$

5.2 Solomon基准算例(Solomon instances)

采用3.1节的方法将Solomon基准算例改造为TDVRP算例, 然后分别采用蚁群算法, Malandrak[4]提出的最邻近法(NN算法)对56个算例进行计算, 得到6组算例的平均计算结果如表6所示. 算法的迭代次数为200次. 由于优化后的“总配送时间”的值域一般为[0, 10000], 目标函数(1)的系数取值为: $\alpha_1 = 10000$, $\alpha_2 = 1$.

由表6可见, 与NN算法相比, 蚁群算法的计算结果有了大幅度提高, 其中: “车辆数量”减少了40%∼80%, “总配送时间”减少了40%∼70%.

表 6 蚁群算法与NN算法的计算结果比较

Table 6 Comparison between ACO and NN algorithm

算例	算法	车辆数	配送时间	行程时间	距离	剩余容量	计算时间/s
C1	NN	26.44	18,875.90	1,563.16	2,568.20	3,478.89	0.01
	ACO	10.00	9,796.89	729.53	1,339.67	190.00	28.81
C2	NN	15.00	29,468.13	1,054.06	1,882.39	8,690.00	0.03
	ACO	3.25	9,580.43	552.67	1,041.12	465.00	82.46
R1	NN	20.67	3,455.68	1,132.73	1,991.74	2,675.33	0.01
	ACO	10.17	1,938.47	802.41	1,538.55	575.33	28.71
R2	NN	8.00	4,957.49	871.01	1,613.17	6,542.00	0.03
	ACO	2.36	1,831.10	777.93	1,505.94	905.64	81.95
RC1	NN	18.63	3,347.60	1,318.00	2,262.48	2,001.00	0.02
	ACO	10.75	2,051.24	935.02	1,755.28	426.00	27.95
RC2	NN	9.13	5,379.39	1,050.86	1,948.19	7,401.00	0.02
	ACO	2.88	2,107.34	1,033.41	2,005.95	1,151.00	63.18

5.3 大规模算例(Large scale instances)

Joerg Homberger在Solomon基准算例的基础上作了扩展, 构建了客户节点为200∼1000的大规模算例[13]. 采用3.1节的方法, 将Homberger的算例进行改造, 可以得到TDVRP问题的大规模算例. 对于目标函数式(1), 由于优化后的“总配送时间”的值域一般为[0, 100000], 所以取: $\alpha_1 = 100000$, $\alpha_2 = 1$.

下面对RC1_2_1∼RC110_1的5个算例进行计算, 结果如表7所示. 由表可见, 随着客户节点数量的增加, 算法的计算时间增长很快. 当节点为1000时, 计算时间约为20 min, 在可以接受的范围内.

表 7　大规模算例的计算结果

Table 7　Experimental results of large scale instances

算例	客户数	车辆数	配送时间	行程时间	距离	计算时间/min
RC1_2_1	200	19	6,972.34	3,466.16	6,309.55	1.37
RC1_4_1	400	39	17,359.35	8,597.18	15,389.00	3.95
RC1_6_1	600	62	41,260.36	20,560.28	34,428.16	7.96
RC1_8_1	800	80	70,062.44	37,271.13	62,153.25	12.79
RC110_1	1000	97	106,954.00	55,745.45	93,133.14	18.97

5.4　结论(Conclusions)

本文从初始解的生成, 局部搜索操作的使用, “最大-最小蚂蚁系统”信息素更新策略3个方面着手, 设计了TDVRP问题的一种改进蚁群算法, 得到以下研究结论:

1) 初始解的质量对蚁群算法的效率有着重要的影响, 本文采用NNC构造算法生成蚁群算法的初始解, 有利于加快算法的收敛速度;

2) 局部搜索操作的使用, 可以增强蚁群算法的邻域搜索能力, 从而提高可行解的质量;

3) 本文的测试算例表明, 与最邻近算法, 遗传算法相比, 改进蚁群算法具有更高的效率, 能够得到更优的计算结果;

4) 本文的改进蚁群算法, 在大规模TDVRP算例中也表现出良好的性能, 即使达到1000个客户节点规模, 算法的优化时间约为20 min, 在可接受的范围内.

参考文献(References):

[1] 肖增敏. 动态网络车辆路径问题研究[D]. 成都: 西南交通大学, 2005.
(XIAO Zengmin. *Research on vehicle routing problem with dynamic networks* [D]. Chengdu: Southwest Jiaotong University, 2005.)

[2] JUNG S. *A genetic algorithm for the vehicle routing problem with time dependent travel times* [D]. College Park: University of Maryland, 2000.

[3] BOWMAN E H. Production scheduling by the transportation method of linear programming[J]. *Operations Research*, 1956, 4(1): 100 – 103.

[4] MALANDRAKI C, DASKIN M S. Time dependent vehicle routing problems: formulations, properties and heuristic algorithms[J]. *Transportation Science*, 1992, 26(3): 185 – 200.

[5] ICHOUA S, GENDREAU M, POTVIN J Y. Vehicle dispatching with time-dependent travel times[J]. *Discrete Optimization*, 2003, 44(2): 379 – 396.

[6] DONATI A V, GAMBARDELLA L M, RIZZOLI A E, et al. {Time dependent vehicle routing problem with an ant colony system}[EB/OL]. Manno, Switzerland: Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale(IDSIA), 2002, [2009–3–18]. http://www.idsia.ch/ roberto/papers/IDSIA–02–03.pdf.

[7] DONATI A V, MONTEMANNI R, CASAGRANDE N, et al. Time dependent vehicle routing problem with a multi ant colony system[J]. *European Journal of Operational Research*, 2008, 185(3): 1174 – 1191.

[8] 李士勇, 陈永强, 李研. 蚁群算法及其应用[M]. 哈尔滨: 哈尔滨工业大学出版社, 2004.
(LI Shiyong, CHEN Yongqiang, LI Yan. *Ant Colony Algorithms with Applications*[M]. Harbin: Harbin Institute of Technology Press, 2004.)

[9] DORIGO M, GAMBARDELLA L M. Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 53 – 66.

[10] STÜTZLE T, HOOS H H. Max-min ant system[J]. *Future Generation Computer Systems*, 2000, 16(8): 889 – 914.

[11] SOLOMON M M. Algorithms for the vehicle routing and scheduling problems with time window constrains[J]. *Operations Research*, 1987, 35 (2): 254 – 265.

[12] BRÄYSY O, GENDREAU M. Vehicle routing problem with time windows, part Ⅰ: route construction and local search algorithms[J]. *Transportation Science*, 2005, 39(1): 104 – 118.

[13] HOMBERGER J. Extended solomon's VRPTW instances[EB/OL]. Germany: FernUniversität in Hagen, 1998 [2009–3–18]. http://www.fernuni-hagen.de/WINF/touren/inhalte/probinst. htm.

作者简介:

段征宇　(1978—), 男, 博士, 讲师, 目前研究方向为交通运输规划与管理, E-mail: d_zy@163.com;

杨东援　(1953—), 男, 教授, 博士生导师, 目前研究方向为交通运输规划与管理, E-mail: yangdyk@yahoo.com.cn;

王　上　(1986—), 男, 硕士研究生, 目前研究方向为交通运输规划与管理, E-mail: upking86@gmail.com.