带时间窗车辆路径问题的分布式多 agent 蚁群算法*

金 淳, 张 雨, 王 聪
(大连理工大学 系统工程研究所, 辽宁 大连 116024)

摘 要: 针对带时间窗车辆路径问题(VRPTW)算法在求解效率、求解复杂度、求解大规模问题方面存在的不足,提出一种改进的分布式多 agent 蚁群算法,以提高算法精度和速度为研究目的。本算法在传统蚁群算法的基础上,为提高算法精度,改进了状态转移规则,结合了邻域搜索算法;为提高算法速度,将本算法设计为分布式结构,利用多分布式 agent 系统实现了分布式求解 VRPTW 问题。针对国际标准算例设计了四个实验,结果表明,本算法在精度、速度、可靠性以及求解大规模问题方面具有明显优势。本研究为有效求解大规模、复杂 VRPTW 问题提供了一种新思路和可行的方法。

关键词: 带时间窗车辆路径问题; 蚁群算法; 分布式算法; 代理
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2018)03-0666-05
doi:10.3969/j.issn.1001-3695.2018.03.006

Distributed multiagent-based ant colony algorithm for vehicle routing problem with time windows

Jin Chun, Zhang Yu, Wang Cong
(*Institute of Systems Engineering*, *Dalian University of Technology*, *Dalian Liaoning* 116024, *China*)

Abstract: Because of the disadvantage in the previous research of the vehicle routing problem with time windows (VRPTW), like limited solving efficiency, limited complexity, and difficulty to solve large-scale problems, this paper proposed a distributed multi agent ant colony algorithm which aimed at improving the accuracy and speed of the algorithm. Based on the traditional ant colony algorithm, this paper improved the state transition rule and combined the neighborhood search algorithm to improve the accuracy. At the same time, it designed the algorithm as a distributed structure to improve the speed. So, it could utilize multi agent distributed system to distributed solve the VRPTW problem. The results of four experiments which designed for the international standard show that this algorithm has obvious advantages in accuracy, speed, reliability and solving large-scale problems. This research provides a new idea and a feasible method for solving large-scale and complex VRPTW problems.

Key words: vehicle routing problem with time windows(VRPTW); ant colony algorithm; distributed algorithm; agent

0 引言

带时间窗车辆路径问题(vehicle routing problem with time windows, VRPTW)是车辆路径问题(vehicle routing problem, VRP)的一种扩展类型,也是典型的 NP-hard 难题。由于 VRPTW 问题比传统 VRP 问题在物流管理与运输组织优化中更具有实用性,所以受到学者们的广泛关注和不断探索。与 VRP 问题相同,VRPTW 问题的求解方法基本上可分为精确算法和启发式算法两大类。精确算法求解精度高,但由于其计算难度与计算量随着问题的节点数的增加呈指数级增加,求解效率难以保证[1]。相比之下,启发式算法则具有全局搜索能力强、求解效率高的特点,求解的实用性更强。因此,目前大部分研究者主要集中于如何构造高质量的启发式算法上,如遗传算法[2]、蚁群算法[3]、模拟退火[4]、粒子群算法[5]及蝙蝠算法[6]等。目前,VRPTW 求解算法大多运行在集中式系统上,求解规模通常在 100 个节点以内,甚至只用十几个节点的算例进行验证,对于规模较大的问题求解效率有限[7~11]。如马秋卓等人[12]在考虑低碳问题的情况下提出解决实际问题的方法,然而其研究仅以 11 个节点的模型为研究对象。Kwon 等人[13]运用禁忌搜索算法求解 VRP 问题,求解节点数在 50 ~ 100,并且没有考虑到时间窗的约束。

因此有些学者开始探索如何有效求解大规模 VRPTW 问题,例如饶卫振等人[14]提出一种快速改进贪婪算法求解了当前 24 个最大规模的 CVRP 算例;李净等人[15]采用基于遗传算法的路径调整方法来提高算法的收敛速度;傅成红等人[16]提出用毗邻信息指导的动态候选集规模改进禁忌搜索算法,提高了算法的自适应能力,节省了搜索时间;段征宇等人[17]针对时间依赖型 VRPTW 问题提出一种改进蚁群算法,对大规模问题也能在短时间内求得可行解。

尽管如此,目前大多针对大规模 VRPTW 问题求解的研究依然存在一些缺陷,例如:求解问题的复杂度不够,目标函数和约束条件简单[18];问题的规模不够,问题所包含的节点数不能过多;算法测试时往往多关注精度指标而忽视运行时间指标[19];求解算法只考虑算法设计本身,只在一台集中式计算机

收稿日期: 2016-11-22; **修回日期**: 2017-01-18 **基金项目**: 国家自然科学基金资助项目(71271041)

作者简介:金淳(1963-),男,教授,博士,主要研究方向为物流与供应链管理、系统优化与仿真(jinchun@dlut.edu.cn);张雨(1992-),男,硕士,主要研究方向为物流系统优化.

上运行,而忽视对运行资源向分布并行的扩展[20]。当今的电子商务日益普及,电商的竞争压力日益激烈,一方面配送任务更加繁重,另一方面客户对于配送时间的要求也越来越高。为此,如何又好又快地规划配送计划,从而降低成本,提高竞争力,是目前商家和顾客都十分关心的问题。

为此,本文针对 VRPTW 问题,以既提高精度又提高速度为目标,设计了有效求解 VRPTW 问题的改进蚁群算法,主要措施包括融合邻域搜索算法、改进状态转移规则、构建多 agent 分布式算法架构等。本文利用分布式蚁群算法高效求解 VRPTW 问题,为高质量快速求解大规模 VRPTW 问题提供了新思路。

1　VRPTW 问题描述及数学模型

VRPTW 问题是指一定数量的客户,各自有不同数量的货物需求,配送中心向客户提供货物,由一个车队负责分送货物,组织适当的行车路线,目标是使得客户的需求得到满足,并能在一定的约束下达到诸如路程最短、成本最小、耗费时间最少等目的。VRPTW 问题具体描述如下[21]:设有一配送中心有一个车队,共有 K 辆货车,车辆容量分别为 $Q_k(k=1,2,\cdots,K)$,有 L 位顾客,第 i 位顾客需求量为 $B_i(i=1,2,\cdots,L)$,且 $\max D_i < \max Q_k$。任务 i 最早开始服务时间为 ET_i,任务最晚开始服务时间为 LT_i,任务总服务时间为 WT_i。车辆从场站出发对客户进行配送服务最后返回场站,要求所有顾客都被配送,每位顾客一次配送完成,不能违反车辆容量的限制,且开始服务时间晚于最早开始服务时间、早于最晚开始服务时间。目的是所有车辆路线的总距离最小。

本文采用文献[22]中所定义的数学模型,将车场编号为 0,顾客编号为 $1,2,\cdots,L$,顾客及车场均以点 $i(i=0,1,\cdots,L)$ 来表示;c_{ij} 表示从点 i 到 j 的运输成本,其含义可以是距离、费用、时间,文中代表距离成本,t_i 表示车辆到达顾客 i 的时间。定义变量如下:

$$x_{ijk}=\begin{cases}1 & \text{车辆 } k \text{ 从点 } i \text{ 行驶到点 } j\\ 0 & \text{其他}\end{cases}$$

$$y_{ki}=\begin{cases}1 & \text{点 } i \text{ 的顾客需求由车辆 } k \text{ 完成}\\ 0 & \text{其他}\end{cases}$$

则 VRPTW 问题模型的目标函数和约束条件如下:

$$\text{obj.}\quad \min Z=\sum_{i=0}^{L}\sum_{j=0}^{L}\sum_{k=1}^{L}c_{ij}x_{ijk} \tag{1}$$

$$\text{s.t.}\quad \sum_i B_i y_{ik}\leqslant q_k \quad \forall k \tag{2}$$

$$\sum_k\sum_j x_{ijk}=1 \quad i=1,2,\cdots,L \tag{3}$$

$$\sum_k y_{ki}=1 \quad i=1,2,\cdots,L \tag{4}$$

$$\sum_i x_{ijk}-\sum_i x_{jik}=0 \quad j=1,2,\cdots,L;\ \forall k \tag{5}$$

$$\sum_i x_{ijk}=y_{kj} \quad j=0,1,\cdots,L;\ \forall k \tag{6}$$

$$\sum_j x_{ijk}=y_{ki} \quad i=0,1,\cdots,L;\ \forall k \tag{7}$$

$$ET_i\leqslant t_i\leqslant LT_i \quad i=1,2,\cdots,L \tag{8}$$

$$x_{ijk},y_{ijk}\in(0,1) \quad i,j=0,1,\cdots,L;\ \forall k \tag{9}$$

其中:目标函数式(1)为配送成本最小;约束式(2)为货车容量约束,每辆货车每次配送货物总量不得超过其装载总量;约束式(3)(4)保证每个客户点只能由一辆车配送一次,不能重复配送;约束式(5)保证整个运送路径中不存在局部的回路;约束式(6)(7)确保每个顾客都被配送到;约束式(8)为时间窗约束。

2　算法设计

2.1　基本思路

本算法在传统蚁群算法的基础上进行改进,试图从精度和速度两个方面提升求解 VRPTW 问题的性能。主要设计思路如下:

a)为提升算法精度,本文设计了新的状态转移规则,由随机影响因子、信息素影响因子、距离影响因子三个维度对蚂蚁的状态转移进行影响;同时,在可行解的基础上引入邻域搜索,优化可行解,提高收敛速率。

b)为提升算法速度,本文在算法结构上将计算与控制分离,构建有效的多 agent 结构,使算法能够将计算部分部署到多台设备上,实现分布式并行计算。

c)另外,由于分布并行计算可以扩充足够的计算资源,为此可采用比传统蚁群算法更大的蚁群规模来扩大算法的搜索空间,所得到的可行解范围更大,这样可避免过早陷入局部最优解,对求解精度及收敛速率也有一定的提高。

2.2　改进的状态转移规则

在搜索过程中,蚂蚁按照状态转移规则选择下一个节点。如式(10)所示,每个节点有各自的状态转移概率,蚂蚁选择概率最大的节点作为下一节点[23]。

设蚂蚁 k 当前节点为 i,在选择下一节点 $i+1$ 前,第 n 个节点的转移概率 C_n 为

$$C_n^k=(\varepsilon)^{\gamma}+[P_{i(i+1)}]^{\alpha}+[D_{i(i+1)}/\text{Max}D]^{\beta} \tag{10}$$

其中:ε 是 0 ~ 1 的随机数。γ 为随机数影响因子,γ 越大意味着蚂蚁选择下一节点的随机性越大,即解变异的可能性越大。

$P_{i(i+1)}$ 为节点 i 到 $i+1$ 边上的信息素浓度,初始值为 1。该浓度随时间衰减,且每次优化的最优路径上的浓度值增加一定数值。α 为信息素影响因子,α 越大意味着蚂蚁选择下一节点受信息素影响程度越大,即受之前解的影响程度越大。

$S_{i(i+1)}$ 为节点 i 到 $i+1$ 边的长度,即两点间距离。Max$D$ 为所有节点中距离最长的两节点的距离。β 为距离影响因子,β 越大意味着蚂蚁选择下一节点受距离影响程度越大,即越容易选择距离更近的节点。

2.3　邻域搜索算法

邻域搜索的作用是扩展当前解的搜索空间,减小算法陷入局部最优的可能性,使得算法能够求得较好的解[24]。

设 p 是基于多 agent 的分布式蚁群算法求解的 VRPTW 问题的一个局部最优解,在此基础上运用邻域算法,则 p 的 k 变换邻域定义为

$$N_k(p)=\{q\mid \text{将 } p \text{ 中的 } r \text{ 条边变换成 } p \text{ 外的 } r \text{ 条边得到 } q,\ \text{且 } q\in S,0\leqslant r\leqslant k\} \tag{11}$$

本文在上述定义的基础上作出以下修改:p 中的 r 条边断开,即形成 $r+1$ 集合,把这 $r+1$ 个集合全排列,得到的每一个排列作为一个 q(可能的解),所组成的新边就相当于 p 外的 r 条边。2 变换邻域 $N_2(p)$ 的规模大小为 $(n-1)(n-2)/2$,而 3 变换邻域 $N_3(p)$ 的规模大小为 $(n-1)(n-2)(n-3)/2$。

2.4　分布式系统结构

蚁群算法本身能够很好地适应分布式系统,通过合理的系

统设计,能够将算法的控制部分与计算部分分离,从而部署在不同的物理机器上[25,26]。

分布式蚁群算法分为 control agent 和 ant agent,系统结构如图 1 所示。Control agent 负责初始化部分参数,控制信息素浓度变化,给各 ant agent 发送计算命令及协调其工作;ant agent负责实现优化计算过程,找出当前局部最优解,反馈给 control agent。Ant agent 共 M 个,分布在 X 台计算机上,每个计算机上的 ant agent 个数不定。

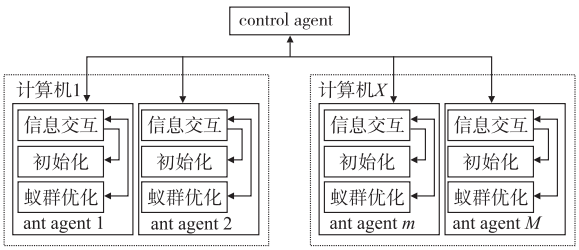

图1 多agent分布式蚁群算法系统结构

2.5 分布式算法流程

多 agent 分布式蚁群算法流程如图 2 所示。

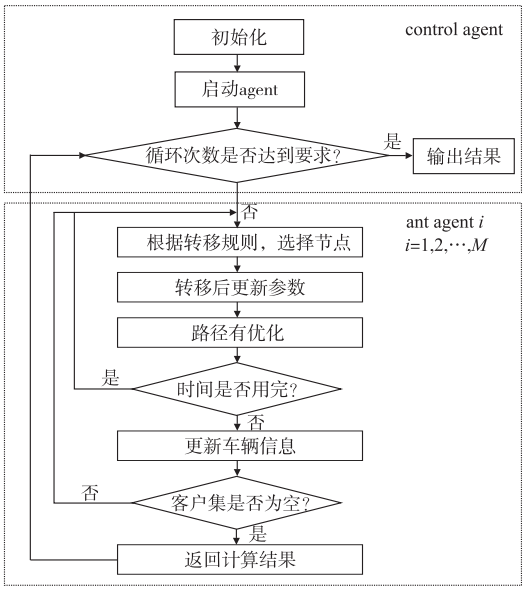

图2 多agent分布式蚁群算法程序

其具体计算步骤如下:

a) Control agent 启动各 ant agent,初始化蚂蚁信息,蚂蚁个数为 M,种群大小为 N,迭代次数为 T,准备接收计算信息。

b) Control agent 初始化各参数 α、β、γ,货车容量 Q,读入数据信息,生成距离矩阵 $\{S_{ij}\}$、信息素矩阵 $\{P_{ij}\}$ 及最大距离 Max$D$。将初始化信息发送给各 ant agent,等待 ant agent 返回计算结果。

c) Ant agent 收到初始化信息后开始计算。将蚂蚁 k 放置在初始点,即场站(depot)。蚂蚁根据状态转移规则选择下一节点 i,若节点 i 的需求量 B_i 小于货车当前容量 Q,且货车到节点 i 的时间大于节点 i 的最早开始服务时间 TS_i,小于节点 i 的最晚开始服务时间 TE_i,则转移至下一节点;否则,若不符合前者,蚂蚁下一节点为初始点,转至步骤 e),若仅不符合后者,蚂蚁重新选择下一节点。

d) 蚂蚁转移到节点 $i+1$ 后更新当前时间,当前时间为原时间加上节点 i 到 $i+1$ 的时间再加上节点 $i+1$ 的总服务时间 TU_i。同时更新当前容量,当前容量为原容量 Q 减去节点 $i+1$ 的需求量 B_i。从待解集即顾客集 C_n 中去除该点,并记录该次转移路径。

e) 应用 2 变换邻域的策略优化回路,通过每次交换两条边来改变当前路径,即当 $S_{ij}+S_{(i+1)(j+1)}<S_{i(i+1)}+S_{j(j+1)}$ 时,两边互相交换,直到线路长度不再缩短为止。

f) 当蚂蚁的时间超过场站的最晚开始服务时间 TE_0,蚂蚁的下一节点为初始点,转至步骤 e)。并重新设置蚂蚁的时间为 0,所用车辆数加 1,转至步骤 c)。

g) 当待解集为空时计算结束,记录各蚂蚁目标函数 L 值,记录当前最优解,发送给 control agent。

h) 若循环次数 $t<T$,control agent 根据信息素更新策略,更新移动路径上的信息素浓度,发送更新后的信息素浓度及当前最优解给 ant agent,转至步骤 c)。若 $t=N$,则计算结束,转至步骤 i)。

i) 输出当前最优解。

其中,各步骤的计算时间复杂度如下:步骤 a) 为 $O(N_n)$,步骤 b) 为 $O(k_n)$,步骤 c) 为 $O(k_n)$,步骤 d) 为 $O(1)$,步骤 e) 为 $O(k)$,步骤 f) 为 $O(1)$,步骤 g) 为 $O(k)$,步骤 h) 为 $O(n)$,步骤 i) 为 $O(1)$。所以本算法计算时间复杂度为:$O(N_n)+N[k(O(k_n)+O(k_n)+O(1)+O(k)+O(1)+O(k))+O(n)+O(1)]$,由于 $k\leqslant n$,可得本算法时间复杂度为 $O(N_n^2)$。

3 算例及结果分析

3.1 实验设计及算例说明

本文设计的多 agent 分布式蚁群算法用 Java 语言开发,在开源的多 agent 开发环境 JADE(Java-based agent development environment)上实现[25],其逻辑结构如图 1 所示。

算法实验分为四类,即算法的精确度比较、算法的运行效率比较、算法结果的可靠性实验、大规模 VRPTW 问题求解实验。实验用算例采用 Solomon 和 Gehring & Homberger's 标准算例。Solomon 标准算例(http://www.sintef.no/Projectweb/TOP/VRPTW/solomon-benchmark/)包括 25、50、100 三个规模,涵盖客户点随机分布(R 类)、群集分布(C 类)和群集—随机分布(RC 类)三个类别,且 1 类算例为紧时间窗约束、2 类算例为松时间窗约束[27]。Gehring & Homberger's 标准算例(http://www.sintef.no/projectweb/top/vrptw/homberger-benchmark/)是在 Solomon 标准算例基础上对算例规模进行了扩充,包括 200、400、600、800、1 000 五个规模[28]。

上述两套标准算例中,车辆数(vehicle number)最大均为 25,容量(capacity)均为 200,信息包括节点编号(customer cust No.)、节点的 X 轴坐标(Xcoord.)和 Y 轴坐标(Ycoord.)、节点需求数量(demand)、节点最早开始服务时间(ready time)、节点最晚开始服务时间(due date)、节点服务总时间(service time),车辆行驶时间等于两节点之间的距离。

3.2 实验及结果分析

3.2.1 求解精度比较实验

为验证本文所提出算法的精确度,选取文献[29]中所使用和提及的量子蚁群算法(QACA)和传统蚁群算法(ACA)进行精度对比。针对规模为 100 的 11 个 R2 类型的 Solomon 标

准算例进行求解,并将结果与文献[29]进行对比。

本实验参数设置:$\alpha=1$,$\beta=0.8$,$\gamma=0.2$,计算机数 X 为 1,ant agent 个数 M 为 4,蚁群规模 N 为 25,迭代次数 T 为 400,节点数 C 为 100。由于求解精度不受算法结构影响,所以选择在一台计算机上使用多 agent 并行计算。实验设备的 CPU 为 Intel Core i7-3770MCPU@3.40 GHz 四核处理器,操作系统为 Windows 10 专业版。算法运行 20 次,取结果的平均值。实验结果如表 1 所示。

表 1　本文算法、QACA、ACA 对 Solomon 标准算例中 11 个算例的测试结果

算例	最优结果	本算法结果	误差	误差率/%	QACA	ACA
R201	1 253.23	1 335.31	75.12	6.55	1 399	1 457
R202	1 202.52	1 293.43	90.91	7.56	1 239	1 389
R203	924.64	988.62	63.98	6.98	1 037	1 106
R204	825.52	885.28	59.76	7.24	955	1 067
R205	994.43	1 055.18	60.75	6.11	1 205	1 196
R206	906.14	970.65	64.51	7.12	1 181	1 154
R207	894.89	948.40	53.51	5.98	977	1 081
R208	726.82	804.95	78.13	10.75	867	1 003
R209	909.16	982.52	73.36	8.07	1 030	1 099
R210	939.37	1 005.40	66.03	7.03	1 112	1 141
R211	885.71	937.87	52.16	5.89	976	1 073

由表 1 可以看出,与量子蚁群算法(QACA)和传统蚁群算法(ACA)相比,本文所提出的算法在针对客户点随机分布宽时间窗约束算例上求解精确度明显更加优异。相比于 Solomon 标准算例,本文所用算法相对于目前已知最优解误差在 5.89% ~10.47%,最大误差为 R208 的 10.47%,因此可以认为:本算法在求解精度上有较好的保证。

3.2.2　算法运行速度比较实验

为验证本算法的运行速度,选择在相同实验环境下通过不同的结构执行三种算法,即本文提出的算法、集中式改进蚁群算法、集中式多 agent 并行改进蚁群算法。其中,后两者在一台计算机上实现,CPU 为 Intel Core i7-3770MCPU@3.40 GHz 四核处理器,操作系统为 Windows 10 专业版。本文的分布式多 agent改进蚁群算法在三台计算机上实现,其中,有两台 CPU 为 Intel Core i5-3230MCPU@2.60 GHz 双核处理器,一台 CPU 为 Intel Core i7-3770MCPU@3.40 GHz 四核处理器,操作系统均为 Windows 10 专业版。

算法设置的参数同实验 1。算法运行 20 次,取结果的平均值。实验结果如表 2 所示,时间单位为 s。

表 2　不同结构下三种算法的求解时间

算例	集中式	多 agent	分布式多 agent
R201	201	94	21
R202	214	97	22
R203	197	80	17
R204	180	81	19
R205	248	109	26
R206	219	102	24
R207	185	79	19
R208	191	83	20
R209	195	86	21
R210	204	89	22
R211	193	91	22

由表 2 可以看出,与集中式改进蚁群算法相比,多 agent 并行蚁群算法由于利用多 agent 的并行运算,充分发挥 CPU 的多核特性,将计算时间缩短了一半以上;而本文的分布式多agent 改进蚁群算法比前两种算法具有更为明显的速度优势,运行速度是多 agent 并行蚁群算法的 4 倍左右,是集中式改进蚁群算法的 9 倍左右。本实验仅选取三台计算机进行 10 个 agent 并行分布计算,就将计算时间缩短至原时间的约 1/9。如果在更大规模计算机集群下,本算法将能够极大地提高计算速度。

在计算速度提高的情况下,相同时间内,相较于传统算法,本算法还可以采用更大的蚁群规模数以及更为复杂的邻域搜索算法,从而扩大解的范围,避免陷入局部最优解,提高解的质量。

3.2.3　求解结果的可靠性验证

为了验证本文所用算法求解结果的可靠性,选择 R204 进行详细实验。记录并统计本文所用算法在计算 R204 标准算例时的收敛情况、车辆使用情况、最终解的详细信息等,并与文献[29]中所记录的结果进行比较。本次实验最终解为 871,收敛情况如图 3 所示,车辆使用情况及最终解的路径顺序如表 3 所示。

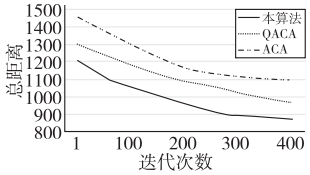

图3　R204收敛比较

表 3　R204 详细实验结果

车辆编号	路径的节点顺序
1	0,27,28,26,40,21,73,72,75,22,41,2,97,42,14,44,38,43,15,57,98,91,16,86,17,45,46,36,49,64,63,32,90,10,70,1,50,3,68,80,24,29,78,34,35,71,65,66,20,30,51,33,81,77,76,69,0
2	0,89,6,94,59,95,92,93,61,5,83,60,18,52,7,88,31,62,11,19,47,48,82,8,99,85,37,87,96,84,100,13,58,53,12,54,55,25,39,67,23,56,74,4,79,9,0

由表 3 可以看出,针对实验算例 R204,本算法在收敛速度上优于量子蚁群算法和传统蚁群算法。而本实验所取得的最终结果包括了共 100 个客户集合,共用了 2 辆车,作为对照,目前已知最优结果使用车数为 2,量子蚁群算法所用车辆为 3,传统蚁群算法所用车辆为 3。该实验说明针对具体算例,本算法达到了精确算法的求解精度水平,结果具有可靠性。

3.2.4　大规模问题求解实验

为验证本文所用算法在求解大规模 VRPTW 问题时的表现,选用 Gehring & Homberger's 标准算例。为与上述实验相对应,均选用 R2 系列算例进行求解。经查阅,近五年关于 VRPTW 问题的研究中,所用实验规模大多数均在 100 节点以内。因此,本实验选取规模为 200、400 节点的标准算例进行大规模问题求解。实验结果如表 4 所示,时间单位为秒(s)。

表 4　大规模 VRPTW 问题求解结果

算例	节点数	求解精度			求解时间/s		
		最优解	本算法	误差率/%	集中式	多 agent	本算法
R2_2_1	200	4 483.16	4 849.86	8.2	468	205	67
R2_2_2	200	3 621.20	3 936.24	8.7	461	201	63
R2_2_3	200	2 880.62	3 145.62	9.2	471	197	68
R2_4_1	400	9 210.15	10 151.89	10.2	--	640	181
R2_4_2	400	7 606.75	8 209.73	7.9	--	607	172
R2_4_3	400	5 911.07	6 459.61	9.3	--	629	185

由表 4 可以看出,本算法对大规模 VRPTW 问题同样具有可靠的求解精度,误差率在 7.9% ~10.2%。同时,本文提出的分布式多 agent 算法能够显著地提高求解速度。而且由于分布式算法的可扩展性,在面对更为复杂的问题时,可以通过

扩展外部计算设备的方法,持续、显著地提高求解速度。

最后,由于实验室环境及时间限制,本实验尚未对更大规模(1 000 节点)的算例进行求解。在此提出今后针对求解更大规模问题算例的三方面改善措施:

a)软件层。改进算法逻辑,优化算法结构。通过改进算法的代码,降低算法时间和空间复杂度,可以提高算法求解速度。

b)平台层。本文所用的 JADE 平台仅为分布式 agent 平台的一种,可以尝试寻找更加高效的分布式平台来实现该算法。

c)硬件层。提高计算机硬件设备数量和质量。同时,分布式的可扩展性导致计算机的数量也对求解速度有着重大影响。

4 结束语

a)本文在传统蚁群算法的基础上,改进状态转移规则,结合邻域搜索算法,构建了一个可以更加有效求解 VRPTW 问题的基于多 agent 的分布式蚁群算法,实现了分布式求解 VRPTW 问题,大大提高了算法的精度和速度。

b)本文针对国际标准算例设计了四个实验,通过与既有算法的比较表明,本算法在精度、速度、可靠性以及求解大规模问题方面具有明显的优越性。

c)本文提出的基于多 agent 的分布式蚁群算法既可以提高求解精度,又可以提高求解速度,克服了传统集中式算法在两者权衡中不得不牺牲某些性能指标的缺点,而且在求解大规模 VRPTW 问题方面优势更为明显。

本研究为如何有效求解大规模、复杂 VRPTW 问题提供了一种新思路和可行的方法,也为今后求解实时动态大规模、复杂 VRPTW 问题提出了解决途径。

参考文献:

[1] 张媛媛,李建斌. 动态车队组合优化模型及精确算法[J]. 系统工程理论与实践,2007,27(2):83-91.

[2] 张群,颜瑞. 基于改进模糊遗传算法的混合车辆路径问题[J]. 中国管理科学,2012,20(2):21-128.

[3] 马建华,房勇,袁杰. 多车场多车型最快完成车辆路径问题的变异蚁群算法[J]. 系统工程理论与实践,2011,31(8):1508-1516.

[4] Mirabi M,Ghomi S M T F,Jolai F. Efficent stochastic hybrid heuristics for the multi-depot vehicle routing problem[J]. Robotics and Computer-Integrated Manufacturing, 2010,26(6):564-569.

[5] 陈玉光,陈志祥. 基于准时送货和最小耗油的配送车辆路径问题研究[J]. 中国管理科学,2015,23(S1):156-164.

[6] Yang Xinshe. A new metaheuristic bat-inspired algorithm[J]. Computer Knowledge and Technology,2010,284:65-74.

[7] 李妍峰,高自友,李军. 基于实时交通信息的城市动态网络车辆路径优化问题[J]. 系统工程理论与实践, 2013,33(7):1813-1819.

[8] 李峰,魏莹. 易腐货物配送中时变车辆路径问题的优化算法[J]. 系统工程学报,2010,25(4): 492-519.

[9] Xiao Yiyong, Zhao Qiuhong, Kaku I, *et al*. Development of a fuel consumption optimization model for the capacitated vehicle routing problem[J]. Computers & Operations Research, 2012,39(7):1419-1431.

[10] Rahimi-Vahed A, Crainic T G, Gendreau M,*et al*. A path relinking algorithm for a multi-depot periodic vehicle routing problem[J]. Journal of Heuristics, 2013,19(3):497-524.

[11] 徐云,刘向彬,陈晓欣,等. 固定时间窗快递车辆路径问题建模及求解[J]. 系统工程,2016,34(7):97-103.

[12] 马秋卓,王健,宋海清. 市区小范围多车辆低碳 VRP:以珠海速递公司区域收件网络为例[J]. 管理工程学报,2016,30(4):153-159.

[13] Kwon Y J, Choi Y J, LEE D H. Heterogeneous fixed fleet vehicle routing considering carbon emission [J]. Transportation Research Part D:Transport and Environment, 2013,23(8):81-89.

[14] 饶卫振,金淳. 求解大规模CVRP问题的快速贪婪算法[J]. 管理工程学报,2014,28(2):45-54.

[15] 李净,袁小华,朱云飞. 物流配送系统中车辆路径问题的实现[J]. 计算机工程与设计,2009,30(16):3783-3786.

[16] 傅成红,符卓. 一种毗邻信息改进的车辆路径问题禁忌搜索算法[J]. 系统工程,2010,28(5):81-84.

[17] 段征宇,杨东援,王上. 时间依赖型车辆路径问题的一种改进蚁群算法[J]. 控制理论与应用, 2010,27(11):1557-1563.

[18] 朱婷,王旭磊,赵来军. 带时间窗的时变多目标危险化学品道路运输路径优化[J]. 工业工程,2016,19(2):62-67.

[19] 颜瑞,张群,胡睿. 考虑三维装箱约束的车辆路径问题研究[J]. 中国管理科学,2015,23(1):128-134.

[20] Lin Canhong, Choy K L, Ho G T S, *et al*. Survey of green vehicle routing problem:past and future trends[J]. Expert Systems with Applications,2014,41(4):1118-1138.

[21] 吴耀华,张念志. 带时间窗车辆路径问题的改进粒子群算法研究[J]. 计算机工程与应用,2010,46(15):230-234.

[22] 李军,郭耀煌. 物流配送车辆优化调度理论与方法[M]. 北京:中国物资出版社,2001.

[23] 曹庆奎, 赵斐. 基于遗传蚁群算法的港口集卡路径优化[J]. 系统工程理论与实践,2013,33(7):1820-1828.

[24] 王征, 张俊, 王旭坪. 多车场带时间窗车辆路径问题的变邻域搜索算法[J]. 中国管理科学,2011,19(2):99-109.

[25] Zafar K, Baig R, Bukhari N,*et al*. Route planning and optimization of route using simulated ant agent system[J]. International Journal of Computer Applications, 2010,4(8):457-478.

[26] Ilie S, Bădică C. Multi-agent approach to distributed ant colony optimization[J]. Science of Computer Programming, 2013,78(6):762-774.

[27] Solomon M M. Algorithms for the vehicle routing and scheduling problems with time window constraints[J]. Operations Research, 1987,35(2):254-265.

[28] Gehring H, Homberger J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows[J]. Proceedings of Eurogen99 Jyvskyl University of Jyvskyl,1999,40(1):95-101.

[29] 何小锋, 马良. 带时间窗车辆路径问题的量子蚁群算法[J]. 系统工程理论与实践,2013,33(5):1255-1261.

[30] Aras N, Aksen D, Tekin M T. Selective multi-depot vehicle routing problem with pricing[J]. Transportation Research Part C,2011,19(5):866-884.

[31] Kuo Yiyo, Wang Chichang. A variable neighborhood search for the multi-depot vehicle routing problem with loading cost[J]. Expert Systems with Applications,2012,39(8):6949-6954.

[32] 李宁,邹彤,孙德宝. 带时间窗车辆路径问题的粒子群算法[J]. 系统工程理论与实践,2004,24(4):130-135.