基于Spark的自适应蚁群算法对CVRP问题的求解

Spark-Based Adaptive Ant Colony Algorithm for Solving CVRP Problems

徐涛/XU Tao[1]，孙鉴/SUN Jian[1,2]，刘陈伟/LIU Chenwei[1]
（1. 北方民族大学，中国 银川 750021；
2. 图像图形智能处理国家民委重点实验室，中国 银川 750021）
(1. North Minzu University, Yinchuan 750021, China;
2. The Key Laboratory of Images and Graphics Intelligent Processing of State Ethnic Affairs Commission, Yinchuan 750021, China)

DOI:10.12142/ZTETJ.202206015
网络出版地址：https://kns.cnki.net/kcms/detail//34.1228.tn.20221223.1314.001.html
网络出版日期：2022-12-27
收稿日期：2022-10-08

摘要：为解决大规模带容量限制的车辆路径问题（CVRP），提出一种基于Spark平台的自适应蚁群算法。该算法利用改进的自适应状态转移规则和动态的信息素更新策略，减轻固定参数的弊端；结合2-opt进行局部搜索优化；在Spark集群上分布式并行实现该算法，利用Spark提供的应用程序编程接口（API）实现对蚁群弹性分布式数据集（RDD）的各种操作，实现蚁群分布式计算。在标准数据集CVRPLib的实验结果表明，该算法使得大规模算例问题求解速度有显著提升。

关键词：Spark；车辆路径问题；蚁群算法；2-opt；并行计算

Abstract: An adaptive ant colony algorithm based on Spark platform is proposed to solve the large-scale capacitated vehicle routing problem (CVRP). First, the algorithm uses improved adaptive state transfer rules and dynamic pheromone update strategy to alleviate the drawbacks of fixed parameters; then, it combines 2-opt for local search optimization; finally, the algorithm is implemented in distributed parallel on Spark cluster, using the application programming interface (API) provided by Spark to realize various operations on ant colony resilient distributed datasets (RDDs) to achieve ant colony distributed computation. The experimental results on the standard dataset CVRPLIB show that the algorithm has a significant improvement in the speed of solving the large-scale arithmetic problem.

Keywords: Spark; vehicle routing problem; ant colony algorithm; 2-opt; parallel computing

作为车辆路径问题（VRP）的一个分支，带有容量限制的车辆路径问题（CVRP）和旅行商问题（TSP）相比，约束条件复杂，问题规模大，求解难度高。VRP在物流配送中具有至关重要的作用，是运筹学和组合优化领域的研究热点，受到业界广泛关注。

VRP[1]是一种NP-hard问题，主要的解决方法有精确式算法、元启发式算法、机器学习算法等。分支限界法[2-3]及动态规划法[4-5]是精确式算法[6]的代表，可以有效求解小规模的CVRP问题。例如，对于数据集CVRPLIB①中A类（set A）的27个实例，其节点规模在32~80之间[7]。但面对大规模问题，如数据集CVRPLIB中X类（Uchoa et al）的99个实例，节点规模在101~1 001之间[8]，计算复杂度较高，计算资源受限。强化学习[9-12]和深度学习[13-14]是机器学习算法的代表，但面对大规模CVRP的求解时间较长。蚁群算法[15]、模拟退火算法[16]、遗传算法[17]是元启发式算法的代表。此类算法更适用于大规模客户点的场景，能够在一个可接受的时间内得到一个相对可以接受的解。而蚁群算法由于具有蚂蚁搜索路径过程中并行的特点，结合分布式计算框架，可以有效提高解的效率。

蚁群算法本质上是一种并行算法，其中每只蚂蚁搜索路径的过程彼此独立。黄震华等[18]利用图形处理器（GPU）多线程并发的优势，提出中央处理器-图形处理器（CPU-GPU）异构环境。该环境由CPU负责控制，同时由GPU完成大部分计算任务。每只蚂蚁被分配到统一计算设备架构

基金项目：国家自然科学基金项目（62062002）；宁夏自然基金（2022AAC03289）北方民族大学中央高校基本科研业务费专项资金（FWNX09）；北方民族大学校级一般项目（2021XYZJK01）

① http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

（CUDA）的不同线程上。虽然能实现蚂蚁并行，但是这种异构环境依旧是单GPU多线程的单机环境。吴昊等[19]提出基于大数据环境MapReduce框架的蚁群算法。该算法用Map函数并行每只蚂蚁求解过程，用Reduce函数表达求得较优解和改变信息素过程，同时应用云计算的管道能力，把Reduce函数的输出信息作为下一代Map函数的输入，以进行下一代循环。这种蚁群算法在求解大规模的TSP问题上取得较好的结果。与MapReduce平台蚁群优化算法相比，王诏远等[20]提出的基于Spark平台的蚁群优化算法在解决TSP问题上的执行速度提升了10倍以上。

虽然上述并行算法能较好地解决CVRPLIB中A类及X类的VRP，但这些算法仍存在求解时间长、精度不高、平台架构限制大、扩展性差等缺陷。为此，本文中我们提出一种基于Spark平台的自适应蚁群算法（SPAMACO）。

本文中，我们首先针对蚁群算法信息素导向和启发信息权重进行自适应变化设计，摆脱固定参数的弊端；分段设计信息素强度变化，及时跳出局部最优解；针对求解精度差的问题，在产生局部解时利用2–opt算子增强算法局部搜索能力；针对大规模数据集执行时间过长问题，在Spark平台实现该并行算法，将蚁群封装成弹性分布式数据集（RDD），实现蚁群在分布式集群环境下的并行构造可行解，在减少算法执行时间的前提下保证了算法的求解精度。

1 技术介绍

1.1 CVRP问题描述

在图 $G=(V,E)$ 中，顶点 $V=\{v_0,v_1,\cdots,v_n\}$，边 $E=\{(v_i,v_j),i\neq j\}$。其中，v_0 表示仓库，v_n 为第 n 个客户点。每个客户的需求为 c_i，d_{ij} 表示 (v_i,v_j) 的欧几里得距离，每辆车的最大载重为 D。约束条件如下：

（1）每辆车都必须从仓库出发，并且返回仓库；

（2）每个客户点有且只有一辆车可以进行配送服务；

（3）一条线路上客户的总需求不能超过车辆的容量限制。

CVRP的目标是在这些节点当中确定一条最短的路径。集合 $K=\{1,2,3,\cdots,k\}$ 表示运输车辆，车辆总数为 k。集合 $P=\{p_i=\{v_{i1},v_{i2},\cdots,v_{ir\,i}\}|r\in K\}$，使 P 中的 k 条路径距离和最小。其中，r_i 为由车辆 i 配送的客户节点数，v_{ij} 为车辆 i 经过的第 $j+1$ 个节点。建立模型的目标使车辆路径总长度最短。

$$\min z=\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{l=0}^{k}d_{ij}x_{ij}^{l},i\neq j \quad (1)$$

$$\text{s.t.}\sum_{i=1}^{n}\sum_{j=0}^{n}c_i x_{ij}^{l}\leqslant D \quad (2)$$

$$p_1\cup p_2\cup\ldots\cup p_n=V, \quad (3)$$

$$p_i\cap p_j=\varnothing,\forall i,j\in K,i\neq j, \quad (4)$$

其中，公式（1）为目标函数。x_{ij}^{l} 的取值为1或0，表示序号为 l 的车辆是否经过 (i,j) 边，1为经过，0为经过。公式（2）表示单量车货运量不能超过最大载重量。公式（3）和公式（4）约束每客户节点有且仅有一辆车送货。

1.2 蚁群算法

蚁群算法是M. DORIGO教授于1992年提出的。该算法源于蚂蚁觅食行为。蚂蚁在寻找食物时，会在经过的路径上释放一种信息素，并且能够感知其他蚂蚁释放的信息素。信息素浓度的大小表明路径的远近。信息素浓度越高，对应的路径距离越短。通常蚂蚁会以较大的概率优先选择信息素浓度高的路径，并且释放一定的信息素，使该条路径上的信息素浓度增高，进而使自己能够找到一条由巢穴到食物源最近的路径。但是路径上的信息素浓度会随着时间推移而逐渐衰减。

在基本蚁群算法中，人工蚂蚁在寻找路径时也会像真实蚂蚁一样，根据路径上信息素的浓度选择方向。浓度越高，选择该方向的的概率就越大。在 t 时刻，蚂蚁 k 从城市 i 到城市 j 的转移概率 $p_{ij}^{k}(t)$ 如公式（5）所示：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum\limits_{u\in Q_k(i)}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}, & j\in Q_k(i)\\ 0, & \text{否则}\end{cases}, \quad (5)$$

其中，$Q_k(i)=\{1,2,\cdots,n\}-\tau_k$ 表示蚂蚁 k 下一步允许选择的城市；列表 τ_k 记录这此次迭代中蚂蚁 k 的已访问城市，在接下来的遍历中不能再次访问列表 τ_k 中的城市；$\tau_{ij}(t)$ 表示 t 时刻路径 (i,j) 上信息素的数量；η_{ij} 表示启发因子，一般取路径 (i,j) 距离的倒数；α 和 β 分别表示路径上的信息素累积量和启发信息的权重比例。

当所有蚂蚁均完成遍历任务后，各条路径上的信息素数量根据公式（6）和公式（7）进行更新。

$$\tau_{ij}(t+n)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t), \quad (6)$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}, \quad (7)$$

其中，$\rho(0<\rho<1)$表示路径上信息素的蒸发比例，$1-\rho$表示信息素的残余量比例；$\Delta\tau_{ij}^{k}$表示此次迭代中蚂蚁k在路径(i,j)上的信息素残余量，即增加量；$\Delta\tau_{ij}(t)$表示此次迭代中o只蚂蚁在路径(i,j)上的信息素累计增加量。

$\Delta\tau_{ij}^{k}$有3种更新模型，分别是蚁周模型、蚁量模型和蚁密模型。本文采用的是蚁周模型，如公式（8）所示。

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{当蚂蚁}k\text{在本次寻找路径中经过路径}(i,j)\text{时}\\ 0, & \text{否则}\end{cases}\tag{8}$$

其中，Q为常数，L_k表示蚂蚁k在此次迭代中所走过的路径长度。

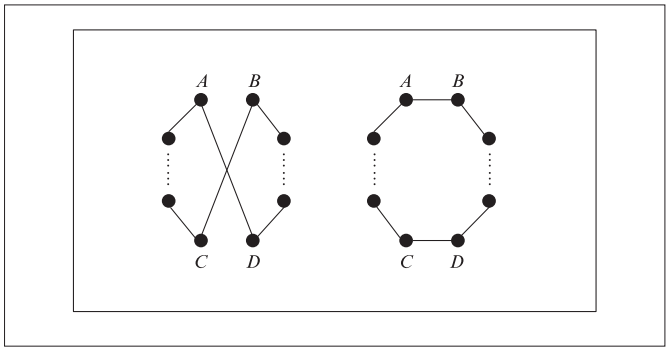
在蚁群算法求解CVRP方面，WANG X.[21]等提出一种新的ACO算法来求解CVRP。该算法允许蚂蚁多次进出仓库，直到它们访问所有客户。这简化了构建可行解的过程。LIN N.[22]等针对CVRP提出一种有效的顺序感知混合遗传算法。该算法的特征为改进的初始化策略和特定问题的交叉算子。前者结合了扫描算法，后者结合了邻域搜索启发式。S. AKPINAR[23]提出一种新的混合算法。该算法结合蚁群优化的解构造机制，执行大邻域搜索（移除和插入算子）方法来求解CVRP。R. GOEL[24]提出用蚁群和萤火虫算法（HAFA）的混合算法来求解CVRP，并在CVRP和带时间窗车辆路径问题（VRPTW）上做了验证。

1.3 局部搜索算子2-opt

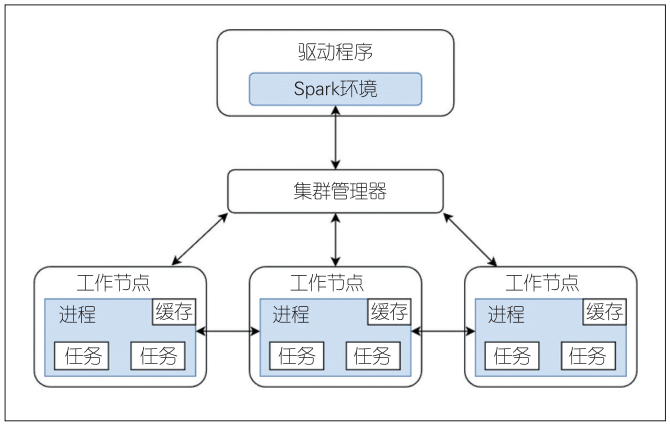
2-opt算法是一种随机性算法，其基本思想是随机取两个元素进行优化，在路径问题求解上得到广泛应用。如图1所示，路径AD、BC。如果满足$AD+BC>AB+DC$，则删除AD、AC并连接AB、DC，生成更短的路径。

1.4 Spark计算框架

Spark是Apache软件基金会三大分布式计算系统开源项目，是基于内存的分布式计算平台。与MapReduce相比，Spark可以将计算中间结果存储在内存中，不需要反复读写分布式文件系统（HDFS），提高了计算速度，因此适合需要迭代处理的算法。Spark运行架构如图2所示。



▲图1 2-opt算法优化过程



▲图2 Spark运行架构

Spark运行架构包括集群资源管理器、执行作业任务的工作节点、每个应用的任务控制节点和每个工作节点的执行进程。Spark所采用的进程有两个优点：一是用多线程执行具体任务，减少了启动开销；二是在执行进程中，会把内存和磁盘共同作为存储设备，有效减少了输入/输出（I/O）开销，提高了读写性能。

2 基于Spark框架的自适应蚁群算法设计

2.1 转移概率$p_{ij}^{k}(t)$的改进

本文采用自适应调整选择概率来提高收敛性能。公式（9）中先进行的状态转换的选择操作，使得搜索个体倾向选择信息素浓度较高且路径长度较短的城市f。

$$f=\begin{cases}\dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum\limits_{u\in Q_k(i)}\left[\tau_{iu}(t)\right]^{\alpha}\left[\eta_{iu}(t)\right]^{\beta}}, & q<q_0\\ \left\langle p_{ij}^{k}(t)\right\rangle, & \text{否则}\end{cases}\tag{9}$$

其中，q是$[0,1]$区间的均匀分布的随机数；$q_0=2-\arctan(ni+2)$，$(ni=1,2,\cdots,iterations)$，$iterations$为最大迭代次数。

由公式（9）可知，在开始搜索进行城市f的选择时，系统先产生一个随机数$(0\leqslant q\leqslant 1)$，以判断$q$和$q_0$的大小，然后确定转移方向。当$q<q_0$时，根据公式（9）选择最好路径，

否则采用公式（10）的自适应转移概率$\left\langle p_{ij}^{k}(t)\right\rangle$进行更新。

$$\left\langle p_{ij}^{k}(t)\right\rangle=\begin{cases}\dfrac{\left[\tau_{ij}(t)\right]^{\delta}\left[\eta_{ij}(t)\right]^{\gamma}}{\sum\limits_{u\in Q_k(i)}\left[\tau_{iu}(t)\right]^{\delta}\left[\eta_{iu}(t)\right]^{\gamma}}, & j\in Q_k(i)\\ 0, & \text{否则}\end{cases} \tag{10}$$

其中，信息素积累$\delta=\dfrac{2\alpha ni}{iterations}+\alpha$，启发信息$\gamma=\dfrac{\beta ni}{iterations}+\alpha$。随着迭代次数增加，权重比例不断变化。这样可实现自适应变化，消除固定参数的弊端。

2.2 信息素强度的动态调整

在前期，算法收敛比较慢；在后期，路径信息素浓度过大（容易造成局部最优）。对此，本文采用公式（11）中的分段函数代替常数Q。

$$Q_i=\begin{cases}Q, & iterations/3 > ni\\ \dfrac{Q}{2}, & iterations/2 > ni > iterations/3\\ \dfrac{Q}{4}, & iterations > ni > iterations/2\end{cases}。 \tag{11}$$

当最优解在一段时间内不变化时，搜索过程可能陷入局部最优困境。此时，分段的信息素设定可以增加或者减少信息素的数量影响。这使系统可以跳出局部最优困境。

2.3 自适应蚁群算法的并行化实现

用Spark平台可以完成自适应蚁群算法的并行实现。我们采用RDD进行操作，实现蚁群并行构建求解过程。

具体的SPAMACO步骤如下：

1）读取CVRPLIB数据文件，初始化车辆数量k、车辆容量D，构建坐标点（x,y）、客户点需求$demand$，计算各点距离，初始化信息素矩阵；

2）初始化参数；

3）封装蚂蚁为Ant类，利用Spark的.parallelize()操作将蚂蚁装换为分布式的RDD；

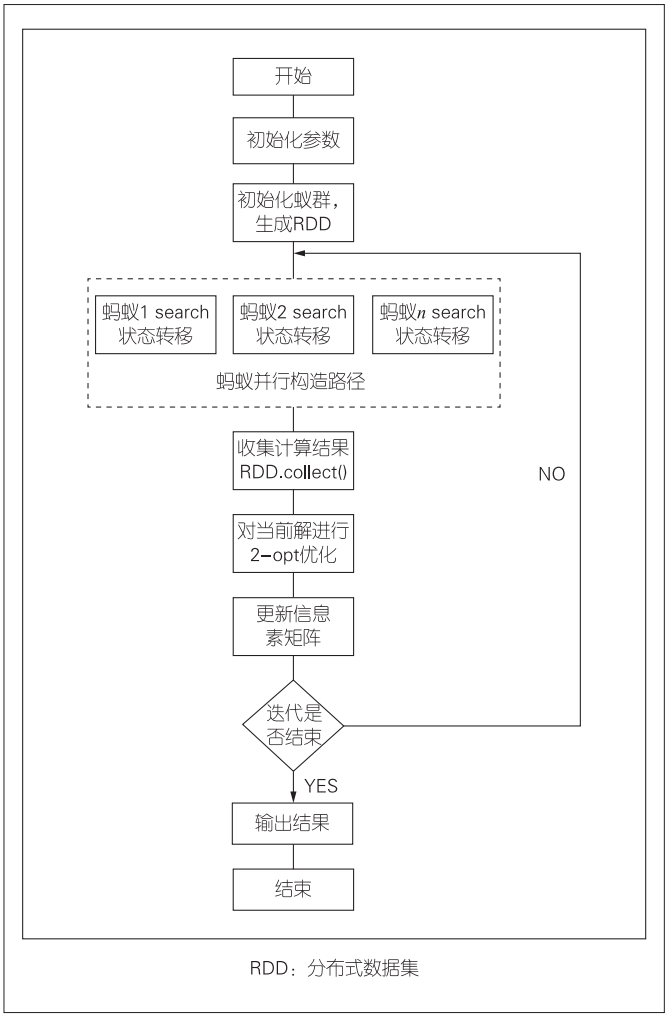
4）在每轮迭代中通过Spark的.map()方法实现并行求解过程；

5）每轮迭代结束后通过Spark的.collect()方法收集计算结果，对其进行2-opt局部优化；

6）更新信息素并广播；

7）判断迭代是否结束，结束输出全局最优解，否则返回步骤4继续并行计算求解。

综上所述，算法流程如图3所示。



▲图3 基于Spark的自适应蚁群算法求解CVRP流程图

3 数值实验

3.1 实验设计

本文基于Spark分布式集群实现SPAMACO。实验环境由8台虚拟机组成，其中包括1台master主节点（负责任务控制节点的运行和管理），7台执行节点。

硬件配置：每台虚拟机CPU为四核四线程，内存为4 GB，磁盘容量为40 GB。操作系统是64位的Ubuntu18.04。

软件配置：Spark2.4.0、hadoop3.1.3、java1.8.0_162。实验中，我们采用具有函数式编程特性的python语言，同时为验证SPAMACO算法的有效性，采用自适应动态搜索蚁群算法（ADACO）[25]和自适应贪婪蚁群算法（GSACO）[26]作为对比实验。其中，为了保证公平性，GSACO和ADACO的参

数选用原文中的参数设置。3个算法的迭代次数均为200次。

3.2 小规模算例实验结果

对于小规模算例，本文采用CVRPLIB中的A类数据集（SET A Benchmark），并选取A-n60-k9、A-n61-k9、A-n62-k9、A-n62-k10、A-n63-K9、A-n64-k9、A-n65-k9、A-n69-k9、A-n80-k10进行求解。实验结果如表1所示，其中V表示当前算法最优解，T表示程序运行时间。

▼表1 小规模算例实验结果

算例	ADACO		GSACO		SPAMACO	
	V	T/s	V	T/s	V	T/s
A-n60-k9	1 505.99	70.51	1 490.72	63.36	1 434.19	77.37
A-n61-k9	1 193.36	76.67	1 181.34	66.16	1 107.43	77.64
A-n62-k9	1 432.71	81.84	1 395.79	66.85	1 381.43	74.59
A-n62-k10	1 819.90	81.29	1 784.36	70.17	1 769.49	80.28
A-n63-k9	1 413.09	82.00	1 463.69	69.93	1 388.44	81.78
A-n64-k9	1 581.80	85.21	1 511.13	74.18	1 445.88	81.77
A-n65-k9	1 359.61	92.16	1 348.06	76.75	1 320.60	84.86
A-n69-k9	1 354.16	105.92	1 240.56	96.01	1 305.17	87.38
A-n80-k10	2 098.46	160.89	1 898.82	138.18	1 833.32	114.72

ADACO：自适应动态搜索蚁群算法　　GSACO：自适应贪婪蚁群算法
SPAMACO：基于Spark平台的自适应蚁群算法

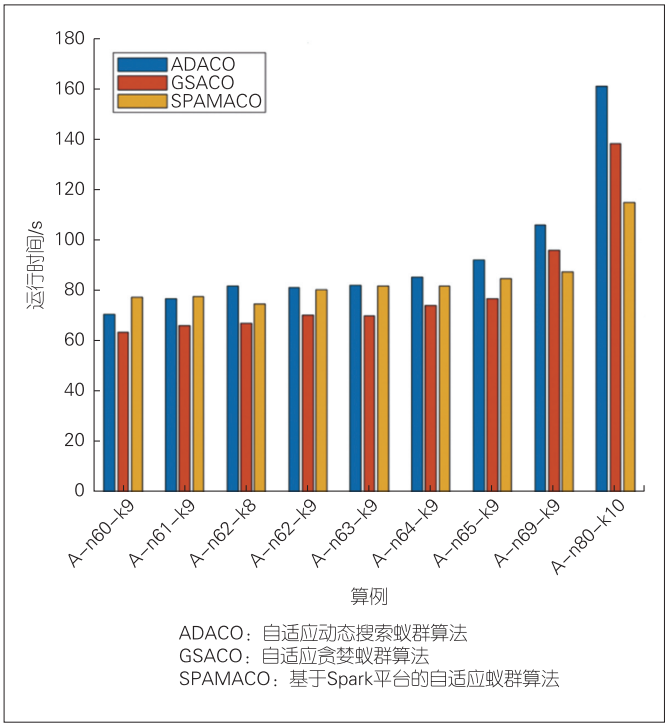
从表1可以看出，和ADACO、GSACO相比，SPAMACO在求解精度上有所提升。但是由于受到Spark集群初始化时间、广播时间和组间通信等的影响，在小规模算例上，SPAMACO的运行时间优势并不明显。在客户点大于65后，并行的执行时间优势逐渐体现。在执行A-n80-k10算例时，本文算法执行时间比ADACO低28.8%，比GSACO低17%。

由图4可以看出，对于小规模算例，随着客户点数的增加，3个算法的执行时间会逐步增加。其中，除A-n80-k10以外，这种增加比例几乎相同。

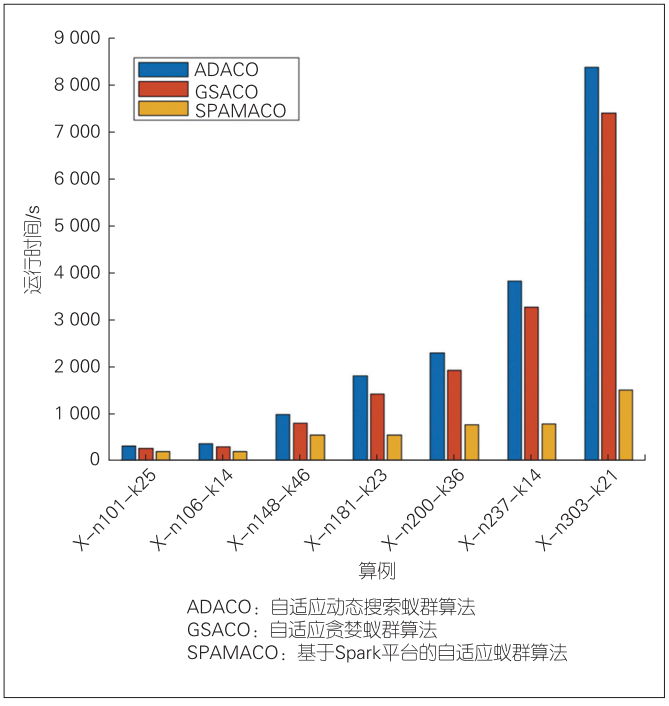
3.3 大规模算例实验结果

对于大规模算例，本文采用CVRPLIB中的X类数据集（Uchoa et al. Benchmark），选取X-n101-k25、X-n106-k14、X-n148-k46、X-n181-k23、X-n200-k36、X-n237-k14、X-n303-k21，并进行求解，结果如表2所示。

从图5可以看出，在大规模算例上（n>100），随着客户点数的增加，ADACO和GSACO的执行时间呈指数增长，而本文所提算法的时间增加比率几乎不变。这是因为此时程序运行的大部分时间都是算法执行时间，同时Spark初始化、广播时间和组间通信所用时间占比也变小了。因此Spark基



▲图4 小规模算例运行时间



▲图5 大规模算例运行时间

于内存的并行计算优势逐步体现出来。

在101个点时，本文所提算法的执行时间比ADACO快39.7%，比GSACO快29.0%。当客户点数增加时，如在200个点时，本文所提算法的执行时间比ADACO快66.6%，比GSACO快60.1%。在300个点时，本文所提算

法的执行时间比ADACO快82.1%，比GSACO快79.7%。因此，SPAMACO更适合处理大规模算例，并且随着算例规模的增加，时间提升效果明显增加。

4 结束语

本文提出了一种基于Spark框架的自适应蚁群算法SPAMACO，结合2-opt局部优化算子，在解决大规模CVRP上效果明显。数值实验表明，该算法在大规模算例应用上具有明显优势。在保证时间优势的条件下，未来我们将进一步探讨最优解。

参考文献

[1] 庞燕, 罗华丽, 邢立宁, 等. 车辆路径优化问题及求解方法研究综述 [J]. 控制理论与应用, 2019, 36(10): 1573-1584. DOI: 10.7641/CTA.2019.90120
[2] LU D. The robust vehicle routing problem with time windows: solution by branch and price and cut [J]. European journal of operational research, 2019, 275(3): 925-938. DOI: 10.1016/j.ejor.2018.12.019
[3] REIHANEH M. A branch-and-price algorithm for a vehicle routing with demand allocation problem [J]. European journal of operational research, 2019, 272(2): 523-538. DOI: 10.1016/j.ejor.2018.06.049
[4] SOYSAL M, et al. A simulation based restricted dynamic programming approach for the green time dependent vehicle routing problem [J]. Computers & operations research, 2017, 88: 297-305. DOI: 10.1016/j.cor.2017.06.023
[5] ÇIMEN M. Time-dependent green vehicle routing problem with stochastic vehicle speeds: an approximate dynamic programming algorithm [J]. Transportation research part D: transport and environment, 2017, 54: 82-98. DOI: 10.1016/j.trd.2017.04.016
[6] SAVITRI H, KURNIAWATI D A. Sweep algorithm and mixed integer linear program for vehicle routing problem with time windows [J]. Journal of advanced manufacturing systems, 2018, 17(4): 505-513. DOI: 10.1142/s0219686718500282
[7] COSSIO E, HERNANDEZJ A, OCHOA-ZEZZATT C A, et al. Comparison between instances to solve the CVRP [J]. International journal of combinatorial optimization problems and informatics, 2018, 9(2): 41.
[8] BONILHA I S, MAVROVOUNIOTIS M, MÜLLER F M, et al. Ant colony optimization with heuristic repair for the dynamic vehicle routing problem [C]//Proceedings of 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2021: 313-320. DOI: 10.1109/SSCI47803.2020.9308156
[9] 牛鹏飞, 王晓峰, 芦磊, 等. 强化学习在车辆路径问题中的研究综述 [J]. 计算机工程与应用, 2022, 58(1): 41-55. DOI: 10.3778/j.issn.1002-8331.2108-0467
[10] XU Y, FANG M, CHEN L, et al. Reinforcement learning with multiple relational attention for solving vehicle routing problems [J]. IEEE transactions on cybernetics, 2022, 52(10): 11107-11120. DOI: 10.1109/tcyb.2021.3089179
[11] PENG B, WANG J H, ZHANG Z Z. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems [M]// Communications in computer and information science. Singapore: Springer Singapore, 2020: 636-650.. DOI: 10.1007/978-981-15-5577-0_51
[12] XIN L, SONG W, CAO Z G, et al. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems [J]. Proceedings of the AAAI conference on artificial intelligence, 2021, 35(13): 12042-12049. DOI: 10.1609/aaai.v35i13.17430
[13] BDEIR A, BOEDER S, DERNEDDE T, et al. RP-DQN: an application of Q-learning to vehicle routing problems [EB/OL]. [2022-10-16]. https://arxiv.org/abs/2104.12226
[14] ZHAO J X, MAO M J, ZHAO X, et al. A hybrid of deep reinforcement learning and local search for the vehicle routing problems [J]. IEEE transactions on intelligent transportation systems, 2021, 22(11): 7208-7218. DOI: 10.1109/TITS.2020.3003163
[15] 李奕颖, 秦刚. 基于Spark的改进蚁群算法对带时间窗车辆路径问题的求解 [J]. 计算机系统应用, 2019, 28(7): 9-16. DOI: 10.15888/j.cnki.csa.007000
[16] 孙鉴, 刘淞佐, 武晓晓, 等. 基于Spark的并行模拟退火算法求解TSP [J]. 电子测量技术, 2022, 45(4): 53-58. DOI: 10.19651/j.cnki.emt.2108429
[17] 唐坤. 车辆路径问题中的遗传算法设计 [J]. 东华大学学报(自然科学版), 2002, 28(1): 66-70. DOI: 10.3969/j.issn.1671-0444.2002.01.015
[18] 黄震华, 赵振岐, 林培裕, 等. 基于异构平台的并行最大最小蚁群算法 [J]. 同济大学学报(自然科学版), 2016, 44(12): 1949-1955. DOI: 10.11908/j.issn.0253-374x.2016.12.021
[19] 吴昊, 倪志伟, 王会颖. 基于MapReduce的蚁群算法 [J]. 计算机集成制造系统, 2012, 18(7): 1503-1509. DOI: 10.13196/j.cims.2012.07.162.wuh.019
[20] 王诏远, 王宏杰, 邢焕来, 等. 基于Spark的蚁群优化算法 [J]. 计算机应用, 2015, 35(10): 2777-2780+2797. DOI: 10.11772/j.issn.1001-9081.2015.10.2777
[21] WANG X Y, CHOI T M, LIU H K, et al. Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems [J]. IEEE transactions on intelligent transportation systems, 2016, 17(11): 3132-3141. DOI: 10.1109/TITS.2016.2542264
[22] LIN N, SHI Y J, ZHANG T L, et al. An effective order-aware hybrid genetic algorithm for capacitated vehicle routing problems in Internet of Things [J]. IEEE access, 2019, 7: 86102-86114. DOI: 10.1109/ACCESS.2019.2925831
[23] AKPINAR S. Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem [J]. Expert systems with applications, 2016, 61: 28-38. DOI: 10.1016/j.eswa.2016.05.023
[24] GOEL R. A hybrid of ant colony and firefly algorithms (HAFA) for solving vehicle routing problems [J]. Journal of computational science, 2018, 25: 28-37. DOI: 10.1016/j.jocs.2017.12.012
[25] 贺智明, 郑丽, 梁文. 基于自适应动态搜索蚁群算法的车辆路径规划 [J]. 计算机工程与设计, 2021, 42(2): 543-551. DOI: 10.16208/j.issn1000-7024.2021.02.036
[26] LI W, XIA L, HUANG Y, et al. An ant colony optimization algorithm with adaptive greedy strategy to optimize path problems [J]. Journal of ambient intelligence and humanized computing, 2022, 13(3): 1557-1571. DOI: 10.1007/s12652-021-03120-0

作 者 简 介

徐涛，北方民族大学在读硕士研究生、CCF会员；主要研究方向为大数据技术等。

孙鉴，北方民族大学讲师、CCF会员；主要研究方向为大数据、存储与管理等。

刘陈伟，北方民族大学在读硕士研究生、CCF会员；主要研究方向为云计算、任务调度等。