文章编号: 1000-6788(2012)11-2540-10 中图分类号: TP301.6 文献标志码: A

求解双层 CARP 优化问题的知识型蚁群算法

邢立宁, 陈英武, 姚 锋, 贺仁杰, 姜 江

(国防科技大学 信息系统与管理学院 管理系, 长沙 410073)

摘 要 双层 CARP 优化问题不仅要解决微观路径优化问题, 还要解决宏观配置优化问题, 最大程度地降低整体系统的固定成本和运行成本. 提出了一种求解双层 CARP 优化问题的知识型蚁群算法: 构建了一个动态参数决策模型, 并采用该模型为每次迭代动态地选择一组合适的参数; 基于弧段聚类知识和弧段顺序知识来构建可行解; 采用 2-Opt 方法对每次迭代中的最优解进行局部优化. 实验结果表明知识型蚁群算法在优化性能方面优于其他几种方法.

关键词 弧段顺序; 弧段聚类; 动态参数调整; 宏观配置优化; 微观路径优化; 蚁群算法

The knowledge-based ant colony optimization to double layer capacitated arc routing problems

XING Li-ning, CHEN Ying-wu, YAO Feng, HE Ren-jie, JIANG Jiang

(Department of Management Science and Engineering, College of Information System and Management, National University of Defense Technology, Changsha 410073, China)

Abstract The double layer capacitated arc routing problem considers a high-level configuration problem and a low-level routing problem, and its objective is minimize fixed costs and running costs of the whole system. A Knowledge-based Ant Colony Optimization (KACO) was proposed to the Double-layer Capacitated Arc Routing Problems. The exploitation of heuristic information, dynamic parameter adjustment and local optimization characterized the KACO. The dynamic parameter adjustment decreased the sensitivity of parameters to final experimental results. The feasible solution was constructed with the guidance of arc cluster knowledge and arc priority knowledge. Local optimization based on two-Opt heuristic largely improved the performance of KACO. In order to validate the performance of KACO, 87 benchmark problems were solved by KACO and some heuristic methods. Experimental results suggest that KACO outperforms these methods.

Keywords arc priority; arc cluster; dynamic parameter adjustment; high-level configuration optimization; low-level routing optimization; ant colony optimization

1 引言

在管理科学、计算机科学和工程学等学科及诸多应用领域中, 不断涌现出许多复杂的组合优化问题. 鉴于实际工程问题的复杂性、约束性、非线性和建模困难等特点, 探索高效率的优化算法已成为相关学科的主要研究方向之一 [1]. 智能优化方法通过模拟某些自然现象或过程而产生、发展并不断成熟, 为解决复杂工程问题提供了新思路和新手段 [2]. 智能优化方法的出现极大地丰富了最优化技术, 也为那些用传统优化技术难以处理的组合优化问题提供了切实可行的解决方案.

近年来, 越来越多的学者开始研究智能优化过程中演化与学习之间的交互 [3]. 根据现有文献, 研究人员一般通过两种途径来实现演化与学习之间的交互 [3]: 学习已获得个体的一些优良特征, 然后采用这些优

收稿日期: 2010-07-02
资助项目: 国家自然科学基金重点项目 (71031007); 国家自然科学基金 (70971131, 71101150, 70801062); 高等学校博士学科点专项科研基金 (20104307120019)
作者简介: 邢立宁 (1980–), 男, 陕西西安人, 汉, 博士, 讲师, 研究方向: 智能优化方法、管理理论与管理决策技术, E-mail: xinglining@gmail.com; 通讯作者: 陈英武 (1963–), 男, 湖南益阳人, 汉, 博士, 教授, 博士生导师, 研究方向: 管理理论与管理决策技术, E-mail: ywchen@nudt.edu.cn.

良特征来改进后续演化中产生的个体; 学习已获得个体的一些不良特征, 然后采取措施防止后续演化中的个体具有这些不良特征. 很多学者都尝试采用外部存储器 (memory) 来实现演化与学习之间的交互机制 [4−5]. Chung 等 [4] 将已获得个体的优良特征看作是信念 (beliefs), 将这些信念保存到外部存储器中, 然后采用这些信念来改进后续演化中产生的个体. Branke[5] 将一些已获得的较优个体保存到外部存储器中, 然后采用这些较优个体来改进后续演化中产生的新个体. 一些学者基于案例 (case) 来实现演化与学习之间的交互 [6]. Louis 等采用一种基于案例的推理 (case-based reasoning) 方法从基于案例的存储器 (case-based memory) 中选择特征来改进后续产生的个体 [6]. 一些学者采用学习演化模型 (learnable evolution model, LEM) 来实现演化与学习之间的交互 [7−8]. 在 Michalski 构建的学习演化模型中, 主要采用机器学习方法来产生后续演化中的个体 [7]. Ho 等提出了一种学习型遗传框架 (learnable genetic architecture, LEGA) 来实现演化与学习之间的交互 [3]. 现有研究表明: 演化与学习之间的交互是可能的; 可采用多种途径来实现演化和学习之间的交互, 例如: 版本空间 (version space)[9]、基于案例的存储器 [6]、Q- 学习系统 (Q-learning system)[10−11] 和 AQ- 学习系统 (AQ-learning system)[6] 等; 通过演化和学习之间的交互可有效提高智能优化方法的优化效率 [3].

现有智能优化方法还没有大量直接地挖掘、存储和应用待求解问题的相关领域知识, 因此还不能最有效地得到优化问题最优解. 鉴于此, 本文提出一种知识型蚁群算法: 采用知识模型和蚁群算法相结合的集成建模思路, 以蚁群算法为基础, 同时突出知识模型的作用, 将蚁群算法和知识模型进行优化组合、优势互补, 以提高知识型蚁群算法的效率.

2 双层 CARP 优化问题

车辆优化问题主要分为两类 [12]: 以抽象点为服务对象的车辆路径优化问题 (vehicle routing problem, VRP) 和以道路为服务对象的弧路径优化问题 (arc routing problem, ARP). 在现实生活中, 广泛地存在着许多典型的 CARP(capacitated arc routing problem) 优化问题: 邮件配送服务和校车接送服务的规划问题; 街道清扫车、垃圾收集车、铺沙车和除雪机的路由问题; 天然气管道、石油管道和电力线路的巡查问题等. 在标准 CARP 问题的基础上, 本文构建了一种双层 CARP 优化问题. 该问题以一个城市或地区为研究对象, 首先考虑系统的宏观配置问题 (确定仓库位置、仓库数目和车辆数目), 然后考虑服务的完成问题 (确定所有车辆的最优行驶路径, 使得所有需要服务的弧段都能被及时服务).

双层 CARP 优化问题可抽象地描述为: 在一个给定的有向图中, 每条弧段都存在一个空车成本; 对于两条相邻的弧段, 从一条弧段转向另外一条弧段都会产生一个惩罚项; 在该有向图中, 部分 (或全部) 弧段需要被服务; 对于每条需要被服务的弧段, 它的需求量和服务成本是已知的; 多台完全相同的车辆 (相同的购置成本、限定容量和行驶速度) 被用来处理各条弧段上的服务需求; 在给定的期限内, 相同的任务被多次重复执行; 仓库位置、仓库数目和车辆数目等都是决策变量; 在单次任务中, 任意车辆所处理服务的总需求量不能大于车辆的限定容量; 在一个服务周期内, 每条需要服务的弧段都必须被服务到; 每辆车在执行完任务后都必须回到它的出发仓库; 任何车辆的行驶时间都不能超过最大服务时间; 部分服务是不允许的; 在宏观层面上, 需要确定最合适的仓库位置、仓库数目和车辆数目; 在微观层面上, 需要确定所有车辆的最优行驶路径, 使得所有需要服务的弧段都能被及时服务; 双层 CARP 优化问题的优化目标是最小化固定成本和运行成本之和.

2.1 输入条件

- 一个包含 N_V 个节点和 N_A 条弧段的有向图 $G=(V,A)$, 这里 V 表示节点的集合, A 表示弧段的集合; 每条弧段 $u\in A$ 都存在一个空车成本 $d(u)$.
- 对于两条相邻的弧段, 从弧段 $u\in A$ 转向弧段 $v\in A$ 的惩罚项为 $pen(u,v)$.
- 弧段集合 $R\subseteq A$ 内的所有弧段都需要被服务; 对于每条需要被服务的弧段 $u\in R$, 它的需求量和服务成本分别为 $q(u)$ 和 $s(u)$.
- 每个仓库的构建成本为 C_1, 每辆车的购置成本为 C_2.
- 车辆的限定容量和行驶速度分别为 Q 和 S.
- 行驶距离和空车成本之间的转换系数为 α.
- 每个服务周期内的最大服务时间为 T.
- 在给定的期限内, 相同的任务被重复执行 M 次.

2.2 输出条件

- 仓库的数目 N_1 和每个仓库的位置 $D=\{d_1,d_2,\cdots,d_{N_V}\}$, 这里 d_i 是一个布尔变量, $d_i=1$ 表示节点 i 应该被选作一个仓库 (在节点 i 附近建立仓库).
- 车辆的数目 N_2 和车辆的分布 $DT=\{dt_1,dt_2,\cdots,dt_{N_1}\}$, 这里 dt_i 表示存放在第 i 个仓库内的车辆数目.
- 最优的车辆路径集合 $L=\{l_1,\cdots,l_j,\cdots,l_{N_2}\}$ 和路段服务标识码 $f_j(\mu_{ij})$, 这里, $l_j=\{\mu_{1j},\cdots,\mu_{ij},\cdots,\mu_{(m_j)j}\}$, μ_{ij} 表示路径 l_j 上第 i 条弧段的序号, m_j 表示路径 l_j 上弧段的总数目, $f_j(\mu_{ij})$ 是一个布尔变量. 车辆在遍历路径 l_j 时, 如果配送员处理了弧段 μ_{ij} 上的服务需求, 则 $f_j(\mu_{ij})=1$; 反之, $f_j(\mu_{ij})=0$.
- 每条路径的起始节点序号 (每台车辆的出发仓库序号).

2.3 目标函数

双层 CARP 优化问题的优化目标是最小化总成本 F,

$$F=CF+CR \tag{1}$$

其中, CF 代表固定成本,

$$CF=C_1N_1+C_2N_2 \tag{2}$$

CR 代表运行成本,

$$CR=M\sum_{j=1}^{N_2}Cost(l_j) \tag{3}$$

$Cost(l_j)$ 代表路径 l_j 的总成本,

$$Cost(l_j)=\sum_{i=1}^{m_j}s(\mu_{ij})f_j(\mu_{ij})+\sum_{i=1}^{m_j}d(\mu_{ij})+\sum_{i=1}^{m_j-1}pen(\mu_{ij},\mu_{(i+1)j}) \tag{4}$$

2.4 约束条件

- 每台车辆在执行完任务后都必须回到它的出发仓库. 对于每条路径 $l_j(1\le j\le N_2)$, 第一条弧段的起始节点必须和最后一条弧段的末尾节点相同.
- 在单次任务中, 任意车辆所处理服务的总需求量不能大于车辆的限定容量 Q. 对于任意路径 $l_j(1\le j\le N_2)$,

$$\sum_{i=1}^{m_j}q(\mu_{ij})f_j(\mu_{ij})\le Q \tag{5}$$

- 在单个服务周期内, 任何车辆的行驶时间都不能超过最大服务时间 T, 即

$$\max_{1\le j\le N_2}\left\{\sum_{i=1}^{m_j}d\left(\mu_{ij}\right)\right\}\le\frac{S\times T}{\alpha} \tag{6}$$

- 从每个仓库派出的车辆数目不能超过该仓库储存的车辆数目. 如果第 i 个仓库内有 dt_i 台车辆, 那么从第 i 个仓库内最多派出 dt_i 台车辆.
- 部分服务是不允许的. 在单个服务周期内, 每条需要服务的弧段必须被服务且只能被服务一次 (任意弧段在单个服务周期内不能被多台车辆同时服务).

2.5 双层 CARP 优化问题的求解框架

双层 CARP 优化问题主要考虑: (1) 宏观配置优化问题, 确定仓库位置、仓库数目和每个仓库应配置多少车辆等配置方面的问题; (2) 微观路径优化问题, 确定所有车辆的最优行驶路径, 使得所有需要服务的弧段都能被及时服务. 本文采用分层求解框架来构造具体的求解方法. 在分层求解框架下, 首先需要求解宏观配置优化问题, 然后再求解微观路径优化问题.

文献 [13] 构建了 Extended Random Path-Scanning (ERPS) 和 Extended Random Ulusoy's Heuristic (ERUH) 两种改进启发式方法来求解多仓库 CARP 问题. 根据实验结果, ERPS 和 ERUH 在优化结果方面没有明显差异; 但在计算效率方面存在明显差异 (在相同次数的迭代中, ERPS 所消耗的时间比 ERUH 要少); 因此本文采用 ERPS 方法来计算某配置方案下各种路径方案的目标值.

根据现有研究成果, 蚁群算法是一种能快速有效地求解离散优化问题的智能优化方法. 微观路径优化问题是一种典型的离散优化问题, 因此可基于蚁群算法来构建求解方法. 为了最大程度地提高求解效率, 本文

提出了一种将蚁群算法和知识模型有效集成起来的知识型蚁群算法, 采用知识型蚁群算法来求解给定配置方案下的微观路径优化问题.

综上, 在求解宏观配置优化问题时, 采用 ERPS 方法来计算某配置方案下各种路径方案的目标值; 在求解微观路径优化问题时, 使用不同方法来求解给定配置方案约束下的微观路径优化问题. 为了便于描述, 下文中提及到的 "采用 KACO 方法 (其他类同) 求解双层 CARP 优化问题" 就是指采用 ERPS 方法来求解宏观配置优化问题, 同时使用 KACO 方法来求解微观路径优化问题.

3 知识型蚁群算法

知识型蚁群算法从功能上可分为两个模块: 智能优化模型 (蚁群算法) 和知识模型 (知识矩阵的初始化、更新和应用). 智能优化模型主要负责从广阔的可行解空间中搜索并识别出一些准最优解; 知识模型从已获得的准最优解中学习 (抽取) 一些可用知识, 然后采用已获得的知识来指导后续优化过程. 知识型蚁群算法的优化流程可概括如下:

- 知识初始化, 初始化记录弧段顺序知识、弧段聚类知识和算子知识的矩阵;
- 动态参数调整, 根据参数组合的优化绩效为下次迭代随机选择一个参数组合;
- 可行方案构造, 在已有知识的指导下采用蚁群优化机制构建一组可行方案;
- 可行方案改进, 采用 2-Opt 方法对每次迭代中的最优解进行局部优化;
- 知识更新, 采用已获得的准最优解来更新弧段顺序知识、弧段聚类知识和算子知识.

3.1 知识初始化

弧段顺序知识 (arc priority knowledge, APK) 就是描述必需服务弧段之间服务顺序的一种累积知识. 本文采用一个大小为 $N_R \times Dim$ 的矩阵 APK 来表示弧段顺序知识; 其中, N_R 表示所有必需服务弧段的数目, Dim 表示矩阵 APK 的列宽度 (根据实验结果, 列宽度设置为 20 比较合理). 弧段顺序知识矩阵 APK 主要包括两部分: 左半部分表示距离当前弧段最近的 10 条弧段; 右半部分表示给定弧段序列在已获得的准最优解中出现的次数. 在初始化操作阶段, 首先计算与每条弧段距离最近的 10 条弧段, 并将计算结果保存到弧段顺序知识矩阵 APK 的左半部分 (静态知识); 然后将所有给定弧段序列在已获得的准最优解中出现的次数初始化为 1(动态知识).

弧段聚类知识 (arc cluster knowledge, ACK) 就是描述必需服务弧段之间聚类性质的一种累积知识. 此处提及到的 "聚类" 可这样理解: 将所有必需服务弧段划分为不同的群组, 每个群组中的必需服务弧段将由同一台车辆来完成服务. 本文采用一个大小为 $N_R \times Dim$ 的矩阵 ACK 来表示弧段聚类知识; 其中, N_R 表示所有必需服务弧段的数目, Dim 表示矩阵 ACK 的列宽度 (根据实验结果, 列宽度设置为 20 比较合理). 弧段聚类知识矩阵 ACK 主要包括两部分: 左半部分表示距离当前弧段最近的 10 条弧段; 右半部分表示给定弧段序列在已获得的准最优解中被划分到同一个群组中的次数. 在初始化操作阶段, 首先计算与每条弧段距离最近的 10 条弧段, 并将计算结果保存到弧段聚类知识矩阵 ACK 的左半部分 (静态知识); 然后将所有给定弧段序列在已获得的准最优解中被划分到同一个群组中的次数初始化为 1(动态知识).

参数知识主要指各个参数组合优化绩效的一种累积知识. 为了降低参数对实验结果的敏感性, 本文采用多个不同的参数组合来实施演化过程, 同时根据已获得的优化结果来确定下次迭代所使用的参数组合. 在单次迭代完成后, 如果全局最优解被改进, 那么当前迭代被称为一次成功的迭代. 在求解当前实例的过程中, 采用给定参数组合获得的成功迭代次数可看作是该参数组合的优化绩效. 本文采用一个大小为 N_X 的矩阵 PPC 来表示参数知识, 其中, N_X 表示备选参数组合的数目, $PPC(i)$ 表示第 i 个参数组合的优化绩效. 在初始化操作阶段, 将所有参数组合的优化绩效都初始化为 1.

3.2 动态参数调整

动态参数调整的主要思路可概括为: 在知识型蚁群算法每次迭代之前, 采用轮盘赌法 (基于每个参数组合的工作绩效) 随机地从多组待选参数组合中随机选择一个参数组合作为本次迭代的工作参数; 如果当前最优方案在本次迭代中被改进, 则增加本次迭代所使用参数组合的优化绩效值. 在采用知识型蚁群算法求解双层 CARP 优化问题时, 采用正交表设计了多组不同的参数组合. 在优化过程中动态地调整各个参数的取值, 是有效提高蚁群算法优化绩效的一种简单易行的方法.

3.3 可行方案构造

可行方案构造阶段的主要任务是: 在已获得的启发式知识的指导下, 采用蚁群优化机制构建双层 CARP 优化问题的一组可行方案. 可行方案构造阶段的伪代码如图 1 所示.

1) 允许选择的弧段集合. 在构建当前方案 (路径集合) 的当前路径时, 将所有可供选择的必需服务弧段的集合称为允许选择的弧段集合. 假设 B 表示必需服务弧段的服务标志, $B(i)=1$ 表示必需服务弧段 i 已经被服务; $B(i)=0$ 表示必需服务弧段 i 还没有被服务; 当完成对必需服务弧段 μ 的服务后, 允许选择的弧段集合表示为 $Allow(\mu,B)$.

2) 状态转移规则. 在完成对当前弧段 μ_i(当前状态) 的服务后, 状态转移规则指定了如何从允许选择的弧段集合 $Allow(\mu_i,B)$ 中选择下一条需要服务的弧段 (下一个状态). 当完成对当前弧段 μ_i 的服务后, 人工蚂蚁均按照以下概率分布从允许选择的弧段集合 $Allow(\mu_i,B)$ 中随机选择下一条需要服务的弧段 μ_j.

$$\Pr(\mu_i,\mu_j,B)=\begin{cases}\dfrac{TF(\mu_i,\mu_j)}{\sum\limits_{\mu_k\in Allow(\mu_i,B)}TF(\mu_i,\mu_k)}, & \mu_j\in Allow(\mu_i,B)\\ 0, & \mu_j\notin Allow(\mu_i,B)\end{cases}\tag{7}$$

其中,

$$TF(\mu_i,\mu_j)=[TFA(\mu_j)]^a\times[TFB(\mu_i,\mu_j)]^b\times[TFC(\mu_i,\mu_j)]^c\tag{8}$$

$TFA(\mu_i,\mu_j)$ 表示内在的确定信息的归一化结果:

$$TFA(\mu_i,\mu_j)=\frac{DI(\mu_i,\mu_j)}{\max\limits_{\mu_k\in Allow(\mu_i,B)}\{DI(\mu_i,\mu_k)\}}\tag{9}$$

$TFB(\mu_i,\mu_j)$ 表示累计的弧段聚类知识的归一化结果:

$$TFB(\mu_i,\mu_j)=\frac{ACK(\mu_i,\mu_j)}{\max\limits_{\mu_k\in Allow(\mu_i,B)}\{ACK(\mu_i,\mu_k)\}}\tag{10}$$

$TFC(\mu_i,\mu_j)$ 表示累计的弧段顺序知识的归一化结果:

$$TFC(\mu_i,\mu_j)=\frac{APK(\mu_i,\mu_j)}{\max\limits_{\mu_k\in Allow(\mu_i,B)}\{APK(\mu_i,\mu_k)\}}\tag{11}$$

本文从以下几个方面来刻画内在的确定信息,

$$DI(\mu_i,\mu_j)=\begin{cases}q(\mu_j), & 0\le r<\dfrac{1}{6}\\ \dfrac{1}{q(\mu_j)}, & \dfrac{1}{6}\le r<\dfrac{1}{3}\\ d(\mu_j)+s(\mu_j)-pen(\mu_i,\mu_j), & \dfrac{1}{3}\le r<\dfrac{1}{2}\\ \dfrac{1}{d(\mu_j)+s(\mu_j)+pen(\mu_i,\mu_j)}, & \dfrac{1}{2}\le r<\dfrac{2}{3}\\ \dfrac{q(\mu_j)}{d(\mu_j)+s(\mu_j)+pen(\mu_i,\mu_j)}, & \dfrac{2}{3}\le r<\dfrac{5}{6}\\ \dfrac{d(\mu_j)+s(\mu_j)-pen(\mu_i,\mu_j)}{(\mu_j)}, & \dfrac{5}{6}\le r\le 1\end{cases}\tag{12}$$

$r\in[0,1]$ 表示服从均匀分布的一个随机浮点数, a 表示确定性知识的权重, b 表示启发式知识 ACK 的权重, c 表示启发式知识 APK 的权重. 当完成对当前弧段 μ_i 的服务后, 本文采用伪随机比例规则 (pseudo-random-proportional state transition rule)[14] 从允许选择的弧段集合 $Allow(\mu_i,B)$ 中随机选择下一条需要服务的弧段 μ_j. 经过本文的多次实验, 将伪随机比例系数 q_0 设置为 0.6 较为合理.

3.4 可行方案改进

在知识型蚁群算法的每次迭代之后, 采用局部优化方法 2-Opt 对本次迭代所得到的局部最优解进行改进. 局部优化操作主要包括两个阶段: 单条路径上的局部优化 (图 2) 和两条路径间的局部优化 (图 3). 在图 2 和图 3 中, 粗线表示必需服务弧段, 细线表示两条弧段之间空车成本最小的路径. 局部优化操作通过以上

步骤 1. 初始化局部最优解, 并将其目标值 (固定成本和运行成本之和) 设置为正无穷.
步骤 2. 采用已获得的启发式知识, 构建当前人工蚂蚁的路径集合 (当前方案).
　　步骤 2.1. 选择一个合适的仓库. 到所有未处理的必需服务弧段距离最近的仓库将被选为当前仓库; 如果有多个可选仓库, 则从中随机选择一个仓库. 本操作中所选择的仓库即为当前路径的出发点.
　　步骤 2.2. 确定允许选择的弧段集合.
　　步骤 2.3. 根据**状态转移规则**从允许选择的弧段集合中选择一条必需服务弧段.
　　步骤 2.4. 将步骤 2.3 中选择的弧段加入到当前路径中.
　　步骤 2.5. 如果当前路径还没有违反容量约束或服务时间约束, 则转至步骤 2.2; 反之, 使当前路径沿最短路由快速返回到出发点, 结束当前路径的构造过程.
　　步骤 2.6. 如果还存在一些没有被处理的必需服务弧段, 则转至步骤 2.1.
步骤 3. 如果当前方案好于局部最优方案, 则更新局部最优方案.
步骤 4. 如果还有人工蚂蚁没有构造自己的路径集合, 则转至步骤 2.
步骤 5. 如果局部最优方案好于全局最优方案, 则更新全局最优方案.

图 1　可行方案构造阶段的伪代码

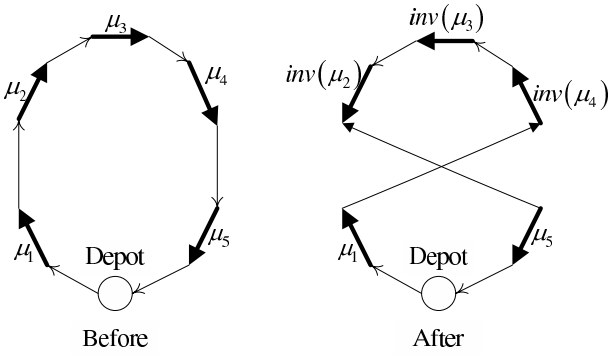


图 2　单条路径上的局部优化

两个阶段的操作, 不断地尝试调换部分必需服务弧段的操作顺序, 从而尽可能地提高当前局部最优解的质量. 本文采用首次改进 (first improvement) 作为每个阶段的终止准则; 即在每个阶段不断尝试改进的过程中, 一旦当前局部最优解获得改进, 则该阶段的优化操作立即结束. 当两个阶段的优化操作都结束后, 局部优化阶段的操作也就立即结束. 在图 2 和图 3 中, 如果弧段 μ 代表某双向弧段某个方向的弧段, 则 $inv(\mu)$ 表示弧段 μ 的反向弧段; 如果弧段 μ 是一条单向弧段, 则 $inv(\mu) = \mu$.

3.5　知识更新

在知识型蚁群算法中, 需要对算子知识、弧段聚类知识和弧段顺序知识进行不断地更新. 算子知识的更新相对比较简单. 在采用知识型蚁群算法求解某实例的过程中, 如果全局最优解在当前迭代中被改进, 则将本次迭代所使用参数组合的优化绩效值增加 1, 这样就完成了对算子知识的更新. 参照现有蚁群算法对信息素的更新方式, 对于弧段聚类知识和弧段顺序知识的更新也经历全局更新、局部更新和信息挥发三个阶段的操作.

1) 全局更新操作. 在全局更新操作中, 只有构建了全局最优方案的人工蚂蚁才能在相关弧段上释放一些信息素 (对当前知识进行全局更新); 即只有全局最优解被改进后, 才能对弧段聚类知识和弧段顺序知识进行全局更新操作. 全局更新规则定义如下:

$$ACK(i,j) = \begin{cases} ACK(i,j) + Q_G, & \text{若 } i,j \in GBS_T \\ ACK(i,j), & \text{其它} \end{cases} \tag{13}$$

$$APK(i,j) = \begin{cases} APK(i,j) + Q_G, & \text{若 } (i,j \in GBS_T) \text{ 和 } (i \to j) \\ APK(i,j), & \text{其它} \end{cases} \tag{14}$$

这里, $i,j \in GBS_T$ 表示弧段 i 和弧段 j 在全局最优方案的同一条路径中被服务, $i \to j$ 表示弧段 i 和弧段 j 被依次服务且弧段 i 先于弧段 j 被服务, Q_G 表示全局更新阶段启发式信息的增量水平.

2) 局部更新操作. 在每次迭代完成后, 知识型蚁群算法都要对弧段聚类知识和弧段顺序知识进行局部更新操作. 在局部更新操作中, 只有构建了局部最优方案 (单次迭代中的最优方案) 的人工蚂蚁才能在相关弧

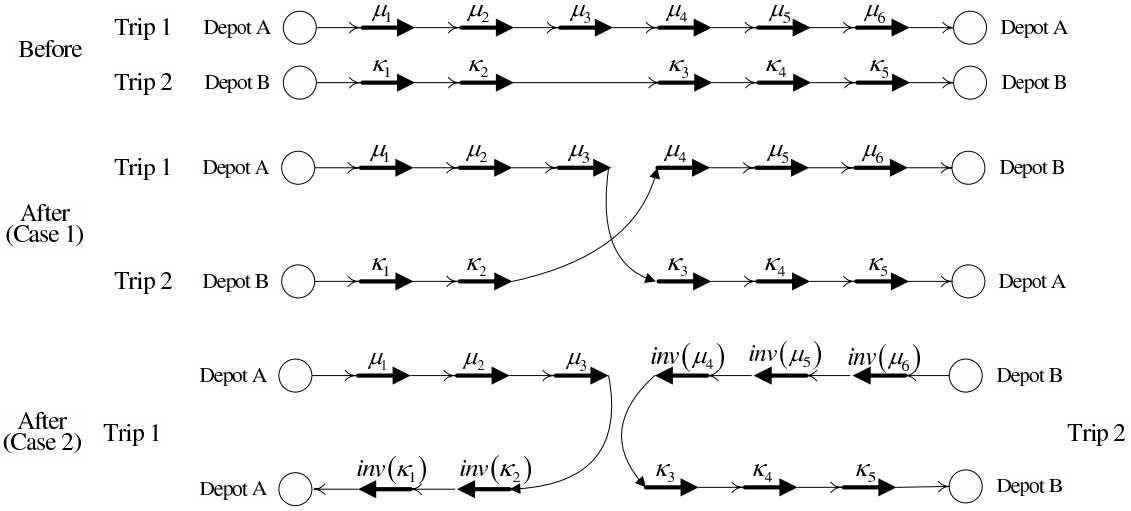


图 3 两条路径间的局部优化

段上释放一些信息素 (对当前知识进行局部更新). 局部更新规则定义如下:

$$ACK(i,j)=\begin{cases} ACK(i,j)+Q_L, & \text{若 } i,j\in LBS_T \\ ACK(i,j), & \text{其它} \end{cases} \tag{15}$$

$$APK(i,j)=\begin{cases} APK(i,j)+Q_L, & \text{若 } (i,j\in LBS_T) \text{ 且 } (i\to j) \\ APK(i,j), & \text{其它} \end{cases} \tag{16}$$

这里, $i,j\in LBS_T$ 表示弧段 i 和弧段 j 在局部最优方案的同一条路径中被服务, $i\to j$ 表示弧段 i 和弧段 j 被依次服务且弧段 i 先于弧段 j 被服务, Q_L 表示局部更新阶段启发式信息的增量水平.

3) 信息挥发操作. 在每次迭代完成后, 知识型蚁群算法都要对弧段聚类知识和弧段顺序知识进行信息挥发操作. 在知识型蚁群算法中, 为了防止优化过程陷入到停滞状态 (stagnation state), 本文将弧段聚类知识矩阵和弧段顺序知识矩阵中动态知识 (知识矩阵的右半部分) 的水平限定在区间 $[\tau_{\min},\tau_{\max}]$ 内. 信息挥发规则定义如下:

$$\begin{aligned} &ACK\,(i,j)=\max\{\tau_{\min},\min\{\tau_{\max},(1-\rho)\,ACK\,(i,j)\}\} \\ &1\le i\le N_R,\quad 0.5Dim<j\le Dim \end{aligned} \tag{17}$$

$$\begin{aligned} &APK\,(i,j)=\max\{\tau_{\min},\min\{\tau_{\max},(1-\rho)\,APK\,(i,j)\}\} \\ &1\le i\le N_R,\quad 0.5Dim<j\le Dim \end{aligned} \tag{18}$$

这里, ρ 表示启发式信息的挥发系数 $(0<\rho<1)$.

3.6 停止准则

知识型蚁群算法的停止准则可定义为: 当下列两个条件中任意一个条件满足时, 优化过程立即终止.

i) 最大的 MI 次迭代已经消耗完毕;

ii) 全局最优解在连续 SI 次迭代后没有被改进.

4 实验结果及分析

本章主要将 KACO 和 HHEA[13] 和 MMAS[15] 的优化绩效进行对比分析, 从而充分地展示了知识型蚁群算法卓越的优化绩效.

4.1 实验设计和参数设置

本文构建了 87 个双层 CARP 优化问题的测试实例, 分别对 HHEA、MMAS 和 KACO 方法的优化绩效进行验证. 由于这些测试实例的最优值是未知的, 因此使用最优下限估计方法来估计每个实例的最优下限值, 然后采用最优下限值替代最优值来估计各种方法的优化误差. 本文采用 Visual C++ 语言分别实现了 HHEA、KACO 和 MMAS 这三种方法. 本文所有实验均是在台式计算机中完成的, 该电脑的处理器为奔腾双核 2.0G, 内存为 2.0G. 本文采用每种方法对每个实例均求解 50 次, 然后采用 50 次计算结果的平均值作为

最终的实验结果. 本文采用威氏符号秩次检验 (the Wilcoxon signed ranks test) 方法来完成对不同方法优化绩效的统计分析.

在本章的实验中, 当采用知识型蚁群算法来求解双层 CARP 优化问题时, 将人工蚂蚁的数目 $AntSize$ 设置为 50 个; 将弧段聚类知识矩阵和弧段顺序知识矩阵中动态知识的水平限定在区间 $[1, 100]$ 之内; 终止条件中的最大迭代次数 MI 设置为 200 次, 连续迭代次数 SI 设置为 50 次.

为了对 HHEA、MMAS 和 KACO 方法这五种方法的优化绩效进行比较客观地评价, 统一采用最大计算时间来定义这些方法的终止准则. 也就是说, 当采用某种方法求解某个测试实例时, 一旦预先定义的该测试实例的最大计算时间消耗完毕, 那么立即停止当前的计算过程并记录当前的优化结果. 在本文中, 测试实例 C1-C29 的最大计算时间被设置为 250 秒; 测试实例 C30-C63 的最大计算时间被设置为 1000 秒; 测试实例 C64-C87 的最大计算时间被设置为 2000 秒.

采用 HHEA、MMAS 和 KACO 方法求解 87 个测试实例的平均计算误差如表 2 至表 3 所示. 在求解某测试实例时, 如果右边方法的优化结果明显优于左边方法的优化结果 (置信度为 0.95), 则在这些表中将右边方法的平均计算误差采用粗体字来标注.

4.2 HHEA 和 KACO 的实验结果

从表 1 的实验结果中不难看出, KACO 的整体优化绩效明显好于 HHEA 的整体优化绩效. 在求解一些小规模的测试实例 (例如测试实例 C1-C23) 时, HHEA 和 KACO 在整体优化绩效方面的差异并不是很明显; 在求解一些大规模的测试实例 (例如测试实例 C64-C87) 时, HHEA 和 KACO 在整体优化绩效方面的差异非常明显; 这一点说明了 KACO 方法更适合于求解大规模的测试实例.

虽然 HHEA 和 KACO 都属于现代启发式方法, 但最终的实验结果表明, KACO 比 HHEA 更适合于求解双层 CARP 优化问题. 根据现有的众多研究成果, 蚁群算法特别适合于在离散优化问题的解空间进行多点非确定性搜索; 实验结果也证明了这个结论的正确性.

表 1　采用 HHEA 和 KACO 求解 87 个测试实例的平均计算误差 (%)

测试实例	HHEA	KACO	测试实例	HHEA	KACO	测试实例	HHEA	KACO	测试实例	HHEA	KACO
C1	0.285	0.254	C23	0.528	**0.215**	C45	0.710	**0.354**	C67	3.497	**1.531**
C2	0.082	0.077	C24	0.396	0.362	C46	1.240	**0.631**	C68	2.809	**1.130**
C3	0.164	0.177	C25	0.529	**0.223**	C47	0.723	**0.354**	C69	4.612	**2.337**
C4	0.646	**0.308**	C26	0.595	**0.262**	C48	0.789	**0.323**	C70	3.919	**1.631**
C5	0.063	0.062	C27	1.810	**0.931**	C49	1.206	**0.655**	C71	2.412	**1.236**
C6	0.049	0.046	C28	2.023	**0.954**	C50	0.806	**0.424**	C72	4.056	**1.760**
C7	0.984	**0.469**	C29	1.819	**0.754**	C51	0.821	**0.454**	C73	4.346	**1.866**
C8	1.253	**0.754**	C30	0.561	**0.262**	C52	1.839	**0.962**	C74	4.277	**2.567**
C9	2.311	**0.992**	C31	1.546	**0.885**	C53	0.784	**0.439**	C75	6.511	**3.438**
C10	0.116	0.108	C32	1.640	**0.885**	C54	1.049	**0.554**	C76	1.879	**0.930**
C11	0.742	**0.354**	C33	0.185	0.169	C55	2.189	**1.093**	C77	2.859	**1.289**
C12	0.295	0.285	C34	0.245	0.231	C56	0.138	0.139	C78	3.178	**1.536**
C13	0.121	0.139	C35	3.147	**1.648**	C57	0.807	**0.447**	C79	3.543	**1.807**
C14	0.163	0.154	C36	0.221	0.200	C58	1.032	**0.393**	C80	4.295	**2.208**
C15	0.072	0.069	C37	0.225	0.223	C59	0.643	**0.331**	C81	4.000	**2.519**
C16	0.276	0.277	C38	2.479	**1.578**	C60	0.682	**0.347**	C82	2.323	**1.195**
C17	0.019	0.015	C39	0.699	**0.385**	C61	0.373	0.246	C83	3.152	**1.689**
C18	0.164	0.162	C40	0.519	**0.246**	C62	0.166	0.169	C84	4.149	**2.302**
C19	0.091	0.085	C41	0.697	**0.416**	C63	1.061	**0.477**	C85	3.697	**1.572**
C20	0.826	**0.354**	C42	0.800	**0.377**	C64	2.674	**1.501**	C86	5.174	**2.290**
C21	0.075	0.046	C43	0.632	**0.362**	C65	2.821	**1.601**	C87	5.566	**2.272**
C22	0.086	0.085	C44	0.564	**0.270**	C66	2.392	**1.083**			

4.3 MMAS 和 KACO 的实验结果

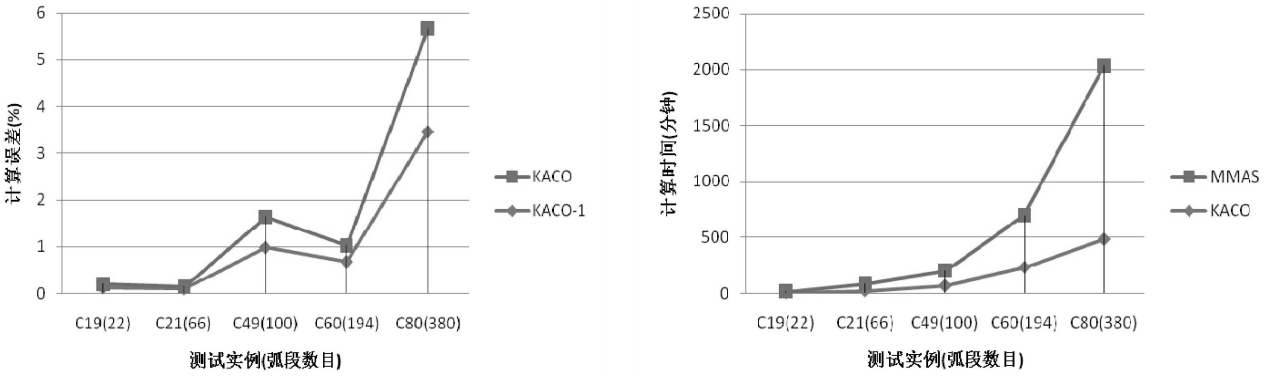
从表 2 的实验结果中不难看出, KACO 的整体优化绩效明显好于 MMAS 的整体优化绩效. 在 MMAS 优化机制的基础上, 本文将启发式信息、动态参数调整和 2-Opt 操作有效地融入到 KACO 方法. 从这个方

面来讲, KACO 的整体优化绩效应该优于 MMAS 的整体优化绩效; 实验结果也证明了这个结论的正确性.

为了进一步分析 MMAS 和 KACO 两种方法的差异, 本节选取 5 个不同规模的测试实例来进行以下分析: 1) 2-Opt 在 KACO 中所起的作用 (KACO-1 表示不含 2-Opt 的 KACO 方法); 2) KACO 和 MMAS 的计算复杂度. 在本部分的实验中, 三种方法均采用以下终止准则: 如果适应度评价次数超过 30000 次, 则算法立即结束. 本部分的实验结果如图 4 所示. 从图 4 的实验结果不难看出: 1) KACO 的优化绩效明显优于 KACO-1, 这表明 2-Opt 在 KACO 中起到了显著作用; 2) KACO 的计算复杂度明显小于 MMAS, KACO 的计算复杂度是可以接受的. 在构建可行方案时, KACO 只考虑与当前弧段距离最近的 10 条弧段; 而 MMAS 将所有弧段都列入可选对象. 正是这种原因, 导致了 MMAS 的计算复杂度要远大于 KACO.

表 2 采用 MMAS 和 KACO 求解 87 个测试实例的平均计算误差 (%)

测试实例	MMAS	KACO	测试实例	MMAS	KACO	测试实例	MMAS	KACO	测试实例	MMAS	KACO
C1	0.296	0.254	C23	0.417	**0.215**	C45	0.371	0.354	C67	1.972	**1.531**
C2	0.078	0.077	C24	0.417	0.362	C46	1.008	**0.631**	C68	2.053	**1.130**
C3	0.182	0.177	C25	0.412	**0.223**	C47	0.690	**0.354**	C69	4.640	**2.337**
C4	0.559	**0.308**	C26	0.317	0.262	C48	0.341	0.323	C70	1.659	1.631
C5	0.058	0.062	C27	1.355	**0.931**	C49	1.237	**0.655**	C71	2.249	**1.236**
C6	0.056	0.046	C28	1.031	0.954	C50	0.467	0.424	C72	2.853	**1.760**
C7	0.799	**0.469**	C29	1.396	**0.754**	C51	0.484	0.454	C73	2.911	**1.866**
C8	1.148	**0.754**	C30	0.309	0.262	C52	1.187	**0.962**	C74	3.193	**2.567**
C9	1.917	**0.992**	C31	1.168	**0.885**	C53	0.849	**0.439**	C75	6.264	**3.438**
C10	0.105	0.108	C32	1.217	**0.885**	C54	0.589	0.554	C76	1.175	**0.930**
C11	0.413	0.354	C33	0.215	0.169	C55	1.382	**1.093**	C77	2.260	**1.289**
C12	0.315	0.285	C34	0.317	0.231	C56	0.213	0.139	C78	2.549	**1.536**
C13	0.143	0.139	C35	1.769	1.648	C57	0.517	0.447	C79	2.194	**1.807**
C14	0.203	0.154	C36	0.240	0.200	C58	0.388	0.393	C80	3.538	**2.208**
C15	0.074	0.069	C37	0.234	0.223	C59	0.415	0.331	C81	4.043	**2.519**
C16	0.291	0.277	C38	2.472	**1.578**	C60	0.580	**0.347**	C82	1.983	**1.195**
C17	0.016	0.015	C39	0.413	0.385	C61	0.287	0.246	C83	1.999	**1.689**
C18	0.205	0.162	C40	0.314	0.246	C62	0.198	0.169	C84	3.767	**2.302**
C19	0.089	0.085	C41	0.465	0.416	C63	0.795	**0.477**	C85	1.840	**1.572**
C20	0.443	0.354	C42	0.667	**0.377**	C64	1.698	**1.501**	C86	3.526	**2.290**
C21	0.056	0.046	C43	0.398	0.362	C65	1.754	**1.601**	C87	3.688	**2.272**
C22	0.096	0.085	C44	0.492	**0.270**	C66	1.099	1.083			



KACO-1 和 KACO 的计算误差 **KACO 和 MMAS 的计算时间**

图 4 其他实验结果

5 结束语

在蚁群算法的基础上, 建立了知识型蚁群算法的基本框架: 蚁群优化模型按照 “邻域搜索” 策略对待优化问题的可行空间进行搜索; 知识模型从前期优化过程中挖掘有用知识, 然后采用知识来指导蚁群优化模型的

后续优化过程. 通过构建知识型蚁群算法的基本框架, 将蚁群优化模型和知识模型有效地结合起来, 极大地提高了知识型蚁群算法的优化绩效. 知识型蚁群算法的基本框架为现有优化方法改进提供了一种有益的借鉴.

参考文献

[1] 周雅兰. 现代智能优化方法研究与应用 [D]. 广州: 中山大学, 2008.

[2] 钟一文. 智能优化方法及其应用研究 [D]. 杭州: 浙江大学, 2005.

[3] Ho N B, Tay J C, Lai E M K. An effective architecture for learning and evolving flexible job-shop schedules[J]. European Journal of Operational Research, 2007, 179(2): 316–333.

[4] Chung C J, Reynolds R G. A testbed for solving optimization problems using cultural algorithm [C]// Proceedings of the 4th Annual Conference on Evolutionary Programming, Cambridge: MIT Press, 1996, 1: 225-236.

[5] Branke J. Memory-enhanced evolutionary algorithms for dynamic optimization problems [C]// Proceedings of Congress on Evolutionary Computation, Piscataway: IEEE Press, 1999, 1: 1875-1882.

[6] Louis S J, McDonnell J. Learning with case-injected genetic algorithms[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(4): 316–328.

[7] Michalski R S. Learnable evolution model: Evolution process guided by machine learning[J]. Machine Learning, 2000, 38(1): 9–40.

[8] Wojtusiak J. The LEM3 system for multitype evolutionary optimization[J]. Computing and Informatics, 2009, 28(2): 225–236.

[9] Reynolds R G. An introduction to cultural algorithms[C]// Proceedings of the Third Annual Conference on Evolutionary Programming, River Edge, NJ: World Scientific Publishing, Singapore, 1994, 1: 131–139.

[10] Kamall K, Jiang L J, Yen J, et al. Using Q-learning and genetic algorithms to improve the efficiency of weight adjustments for optimal control and design problems[J]. Journal of Computing and Information Science in Engineering, 2007, 7(4): 302–308.

[11] Juang C F, Lu C M. Ant colony optimization incorporated with fuzzy Q-learning for reinforcement fuzzy control[J]. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 2009, 39(3): 597–608.

[12] Aráoz J, Fernández E, Zoltan C. Privatized rural postman problems[J]. Computers and Operations Research, 2006, 33(12): 3432–3449.

[13] Xing L N, Rohlfshagen P, Chen Y W, et al. An evolutionary approach to the multi-depot capacitated arc routing problem[J]. IEEE Transactions on Evolutionary Computation, 2010, 14(3): 356–374.

[14] Dorigo M, Stutzle T. Ant Colony Optimization[M]. Cambridge, MA: MIT Press, 2004.

[15] Stutzle T, Hoos H H. Max-min ant system[J]. Future Generation Computer Systems, 2000, 16(8): 889–914.