基于模拟退火的自适应离散型布谷鸟算法求解旅行商问题

张子成[1],韩 伟[1],毛 波[1,2,3]
(1. 南京财经大学信息工程学院,江苏南京 210046;2. 现代粮食流通与全协同创新中心,江苏南京 210046;
3. 江苏省粮食大数据挖掘与应用重点实验室,江苏南京 210046)

摘 要: 提出了一种基于模拟退火的自适应离散型布谷鸟算法求解旅行商问题. 该算法在布谷鸟搜索算法原理的基础上,构造了旅行商问题的路径求解策略. 由于算法的局限性,随着算法的调整和迭代次数的增加,容易破坏已形成的路径,从而使得算法通用性不强. 针对这一局限性,本文提出了一种自适应局部调整算子和全局随机扰动策略. 采用简单的 2-opt 算子作为局部优化算子加快算法收敛速度,引入模拟退火机制防止算法陷入局部最优. 采用标准 TSPLIB 多组数据进行测试,并与有代表性的优化算法进行结果比较. 实验结果证明了该算法在精度和稳定性方面的优势.

关键词: 布谷鸟算法; 旅行商问题; 2-opt 算子; 局部调整; 全局随机扰动
中图分类号: TN92 **文献标识码:** A **文章编号:** 0372-2112 (2018)08-1849-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.08.008

Adaptive Discrete Cuckoo Algorithm Based on Simulated Annealing for Solving TSP

ZHANG Zi-cheng[1], HAN Wei[1], MAO Bo[1,2,3]
(1. *School of Inormation Engineering, Nanjing University of Finance and Economics, Nanjing, Jiangsu* 210046, *China*;
2. *Modern Grain Circulation and Full Cooperation Innovation Center, Nanjing, Jiangsu* 210046, *China*;
3. *Key Laboratory of Large Data Mining and Spplication of Grain, , Nanjing, Jiangsu* 210046, *China*)

Abstract: An adaptive discrete cuckoo algorithm based on simulated annealing is proposed to solve the traveling salesman problem. The proposed algorithm constructs the path solving strategy of traveling salesman problem based on the principle of cuckoo search algorithm. Due to the limitation of algorithm, with the increasing of the number of iterations, it is inclined to destroy the formed paths, which makes algorithm can not be commonly used in variant applications. To overcome this shortcoming, this paper adopt an new strategy which adjust operator locally and disturb parameters randomly . A simple 2-opt operator is used as the local optimization operator to accelerate the convergence rate of the algorithm. The simulated annealing mechanism is introduced to prevent the local optimum in early iterations. The algorithm is tested by the standard TSPLIB multi-group data, comparing with several representative traveling salesman problem algorithm , the experimental results show the advantages of the algorithm in terms of accuracy and stability.

Key words: cuckoo search algorithm; traveling salesman problem; 2-opt optimization; partial adjustment; global random disturbance

1 引言

旅行商问题(Traveling Salesman Problem,简称 TSP)[1]是一个典型的 NP-Hard 难题. 它决定了推销员旅行的最短路径. 在物流配送、车辆路径优化等方面具有重要的现实意义. 目前,该问题已广泛应用于物流配送路线设计、仓储货物装卸、集装箱装载、印制线路钻探等多个交叉领域. 考虑到 TSP 能在多个领域应用,许多

收稿日期:2017-07-06;修回日期:2018-03-13;责任编辑:马兰英
基金项目:国家科技支撑计划农资物流防伪和溯源关键技术与装备研发(No. 2015BAD18B02)

研究学者从数学、计算机科学、工业工程、管理科学等多个学科集中研究解决 TSP[2,3]. 并且布谷鸟搜索算法具有控制简单、收敛速度快、全局优化能力强等优点,对连续函数优化问题具有良好的性能[4]. 许多专家学者尝试用多种方法来求解 TSP. 一些学者将确定性算法,如:分枝定界法[4]和动态规划法[5]用于求解 TSP. 然而,这些方法的复杂度呈指数增长,对大规模 TSP 来说完全无效. 近几十年来,研究者从仿真机制中得到了启发,提出了一系列智能优化算法,如遗传算法(GA)[6]、模拟退火算法(SA)[7]、蚁群算法(AC)[8]等,并用于求解 TSP. 随着智能优化算法的不断发展,许多新的群智能优化算法应运而生,如布谷鸟搜索(CS)算法[9],萤火虫优化(GSO)算法[10],粒子群优化(PSO)算法[11]等. 这些群智能算法被广泛地用去求解 TSP:Mostafa Mahi[11]将粒子群算法和蚁群算法结合,用 3-opt 优化算子作为局部优化算子并用于解决 TSP. 但是,蚁群算法迭代缓慢以及 3-opt 优化算子的高计算复杂性增加了算法的运行时间;一些研究人员利用蚁群算法求解 TSP,对得到的解进行排序,但只更新最优路径的信息素. 它提高了收敛速度,但在某些条件下很容易陷入局部最优[12];Yong Wang[13]提出了一种混合算法求解 TSP,将遗传算法作为一种全局优化策略. 他建议使用两种不同的局部优化算子来优化已获得的最优路径. 该算法在求解小规模 TSP 时具有一定优势,但是对于大规模 TSP 效果不是很理想. Marinakis[14]基于概率分析 TSP,提出了一种混合粒子群优化算法.

布谷鸟搜索(CS)算法具有收敛速度快、易于控制、强大的全局优化能力,在连续函数优化问题上表现出良好的性能[14]. CS 算法已成功地用于优化各种组合优化问题. Aziz Ouaarab[14]提出了一种离散型 CS 算法用于解决 TSP. 针对离散型问题,在 CS 算法基础上设计了一种步长移动策略,使用 2-opt 和双桥算子作为局部优化算子. Saban 等人[15]引入并行协同的混合算法(paco-3opt)求解 TSP. 该算法是基于蚁群算法的框架,使其避免进入局部最优. 实验结果表明,该算法具有更好的性能. Yassine Saji[16]提出了一种离散型的蝙蝠算法,该算法嵌入了连续函数优化算法的机制,测试数据来自 TSPLIB,实验结果表明了它的有效性.

因此,以上论述证明了中小型规模求解精度较高,大规模城市的求解便会陷入局部最优. 2-opt 优化算子优化路径具有快速收敛的特点. 然而,使用变异、交叉和其他算子生成新路径可能会破坏已经形成的最佳路径. 因此,我们提出了一种自适应的局部调整算子并结合 2-opt 算子,并在路径上加入全局随机扰动策略,以保持种群的多样性,避免陷入局部最优.

2 布谷鸟算法 和 2-opt 优化算子

2.1 布谷鸟算法

布谷鸟搜索算法(CS)是由杨提出的一种新型的智能优化算法. Maribel[17]用遗传算法和 CS 算法对比来验证其有效性. 布谷鸟算法用莱维飞行和基于偏好的随机游动来平衡全局搜索和局部搜索. 但该算法在后期仍存在收敛速度慢的缺点. 优化算法参数是提高算法性能的一个研究热点. Maribel[18]提出了一个 FCS 系统动态调整参数. 实验结果表明,与传统的 CS 算法相比,FCS 的性能有所提高. Claudia[19]结合 Sobel 技术与区间二型模糊逻辑来优化 CS,并将改进算法应用于数字图像的边缘检测. CS 模拟布谷鸟寄生习性,可有效解决优化问题. CS 具有结构简单、参数少、跳出局部最优能力强等优点. 建立理想化规则如下:(1)每只布谷鸟每次只产生一个卵,随机选择一个寄生巢孵化它. (2)在随机选择的寄生巢中,最好的巢将保留到下一代. (3)可用寄生巢的数目是固定的,寄生巢的主人发现外来卵的概率为 Pa.

布谷鸟找到鸟窝路径并根据上述理想化规则更新位置,位置更新公式如下:

$$X_i^{(t+1)} = X_i^{(t)} + T \oplus \text{Levy}(\lambda) \tag{1}$$

式(1)中,T 代表步长 并且 $T>0$,$\oplus$是点对点乘法,Levy()是搜索路径并且服从莱维分布[20,21],伪代码描述如算法 1. 自适应局部调整算子如算法 2.

算法 1 Cuckoo Search algorithm

1. **Begin**
2. Objective function,$f(x)$,$X=(X_1,X_2,\cdots,X_d)^{\mathrm{T}}$ d is the dimension of the problem
3. Generate initial population of n host nests
   $X_i(i=1,2,\cdots,n)$
4. **While** (t < Max Generation) **or** (stop criterion)
5. Get a cuckoo randomly by Lévy flights evaluate its quality/fitness $f(X_i)$
6. Select a nest among n (say,j) randomly.
7. **If** $f(X_i)>f(X_j)$
8. replace the nest j with a new solutions
9. **end**
10. A fraction (Pa) of worse nests are abandoned and new ones are built
11. Keep the best solutions(or nests with quality solutions)
12. Rank the solutions and find the current best
13. **end while**
14. Post process results and visualization
15. **End**

算法 2 自适应局部调整算子

1. **Begin**

2. **for** $i=1$ to$[n/s]$
3. Creating an array cs$[0,\cdots,s-1]$
4. **for** $j=(i-1)\times s$ **to** $i\times s-1$
5. $cs[j]=C[j]$
6. **end for**
7. cs_p and cs_q are two random cities in cs
8. r is a random number between 0 and 1
9. **If** $r>w$
10. Swap cs_p and cs_q
11. **end if**
12. **end for**
13. **if** $k'\geq 2$
14. cs_p and cs_q are two random cities in cs
15. r is a random number between 0 and 1
16. **If** $r>w$
17. Swap cs_p and cs_q
18. **end if**
19. **end if**
20. **End**

2.2　2-opt 优化算子

2-opt 优化算子[22]是一种简单实用的 TSP 局部路径优化算法,其目的是消除路径中的交叉边缘.

假设 $c_m(m=0,1,\cdots,n-1)$ 代表每个城市所在的点,$d(c_i,c_j)$ 代表城市 c_i 和城市 c_j 之间的距离. 2-opt 算子优化步骤描述如下:

Step 1:随机生成可行解 $c=(c_0,\cdots,c_i,c_{i+1},\cdots,c_j,c_{j+1},\cdots,c_{n-1})$ (如图 1),令 $i=j=0$.

Step 2:随机选择一条边 $E_1:(c_i,c_{i+1})$,其中 $i<n-2$.

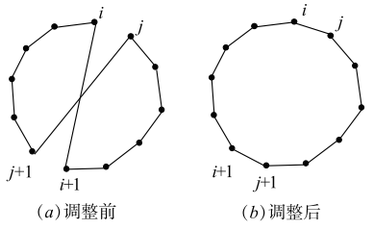
Step 3:随机选择另一条边 $E_2:(c_j,c_{j+1})$,其中 $j<n-2$.

Step 4:如果 $|j-(i+1)|\geq 2$ 和 $d(c_i,c_j)+d(c_{i+1},c_{j+1})<d(c_i,c_{i+1})+d(c_j,c_{j+1})$ 成立,则删除边(c_i,c_{i+1})和(c_j,c_{j+1}),并连接边(c_i,c_j)和(c_{i+1},c_{j+1}).

Step 5:以顶点 c_j作为 E_2 边遍历开始的城市,置 $j=j+1$. 重复执行 Step 3 ~ Step4 直到 $j=n-1$,则 $j+1=0$.

Step 6:以顶点 c_i作为 E_1边遍历开始的城市,置 $i=i+1$. 重复执行 Step 2 ~ Step5 直到 $i=n-1$,则 $i+1=0$.

Step 7:重复执行 Step 2 ~ Step6 直到没有交叉边(如图 1).



(a)调整前　　(b)调整后

图 1　使用2-opt算子优化前后区别

3　离散型自适应布谷鸟算法求解 TSP

第 2 部分描述的布谷鸟搜索算法适用于求解连续函数问题. 然而,TSP 是一个组合优化问题. 因此,有必要将连续布谷鸟优化算法转换成一个离散的布谷鸟优化算法. 转换过程所涉及的步骤如下所述. 首先,为了将鸟巢看成一个可行解,对 N 个巢进行编码和初始化. 随后,自适应调整算子结合 2-opt 优化算子替代 Levy 飞行作为鸟巢移动策略. 适应度值较大的鸟保留下来,否则继续使用旧鸟巢. 寻找一个巢,它实现了一个全局随机扰动策略结合 2-OPT 优化巢. 以下各部分介绍了 TSP 的表示方式、路径初始化、评价函数和自适应局部调整算子的细节. 提出了基于模拟退火算法的自适应离散布谷鸟求解 TSP.

3.1　表示方式

TSP 的向量表示方式分为以下三种:近邻表示、次序表示、路径表示. 本文选取可与本文采用的优化算子相结合的路径表示,例如周游的城市的顺序为 $C_1-C_5-C_4-C_2-C_3$,则算法的一个可行解记为(1,5,4,2,3).

3.2　路径初始化

为加快算法的收敛速度,本文采用轮盘赌方法初始化路径. 首先确定出发城市,并建立禁忌表,按概率选择下一个城市,并与前一城市的距离越短则选择的概率越大,并将已经访问过的城市加入禁忌表,下次不再访问,如此循环,直至禁忌表包含所有城市.

3.3　评价函数

在 TSP 中,路径长度越短则表示解的质量越高,即路径长度与解的适应度成反比,所以评价函数定义为路径长度的倒数.

3.4　自适应局部调整算子

本文针对离散型算法在求解 TSP 时对路径进行整体调整容易破坏已形成的较优路径的缺点,设计一种自适应型局部调整算子来对一条路径进行局部调整. 设 $C=\{C_1,C_2,\cdots,C_n\}$ 为 n 个城市的 TSP 的一条路径,s 是一个输入参数,表示按每段 s 个城市将路径 C 分为 $[n/s]$段并将剩余 $k=n\%s$ 个城市称为剩余段,在$[n/s]$段中任意取两个城市,若 $k\geq 2$,则也在剩余段中任意取两个城市. 对于每一段所取出的城市,按 $w,w\in(0,1)$ 抉择是否进行交换,生成一个属于(0,1)的随机数 r,若 $w>r$ 则交换取出的城市,否则不交换. w 的取值随算法迭代次数的变大而变大,是一个需要自适应的参数,w 的定义式如下:

$$w=\alpha_{\min}+\frac{\text{iter}}{\text{max\_iter}}\times(\alpha_{\max}-\alpha_{\min}) \tag{2}$$

其中 $\alpha_{\max}$ 和 $\alpha_{\min}$ 是经验参数,经大量实验验证 $\alpha_{\max}=0.9$,$\alpha_{\min}=0.5$ 时算法运行结果最佳,iter 表示当前迭代

次数,max_iter 为最大迭代次数.

自适应型局部调整算子伪代码如 Algorithm 2.

根据伪代码可以看出自适应局部算子的时间复杂度为$O(n^2)$,将指数增长的解搜索空间转化为线性增长的解搜索空间,是求解 TSP 的有效性算子.

3.5 全局随机扰动策略

在本文算法中,当寄主发现新来鸟蛋时以全局随机扰动策略调整路径. 全局随机扰动策略为:按 3.1.4 节提出的方法将 $C=\{C_1,C_2,\cdots,C_n\}$ 进行分段,每段取两个城市,取一个随机偶整数 m,若 $k\geq 2$,则 $m\in(2,[n/s]+1)$,否则 $m\in(2,[n/s])$,然后从分好的城市段中随机取 m 段,将这 m 个城市段两两配对,分别与组内另一个城市段的选中城市进行对应交换,全局随机扰动策略伪代码如算法 3.

算法 3 全局随机扰动策略

1. **Begin**
2. **If** $k\geq 2$
3. Create a matrix re with the size $([n/s]+1)\times 2$
4. **else**
5. Create a matrix re with the size$[n/s]\times 2$
6. **end if**
7. **for** $i=1$ to $[n/s]$
8. Creating an array cs$[0,\cdots,s-1]$
9. **for** $j=(i-1)\times s$ **to** $i\times s-1$
10. cs$[j]=C[j]$
11. **end for**
12. cs_p and cs_q are two random cities
13. insert cs_p and cs_q into the *i-th* row of re
14. **end for**
15. **If** $k'\geq 2$
16. Create a random number $m\in(2,[n/s]+1)$
17. Creating an array se$[0,\cdots,m-1]$
18. Insert a number of m different positive integers from 2 to $[n/s]+1$ into Se
19. **else**
20. Create a random number $m\in(2,[n/s])$
21. Creating an array se$[0,\cdots,m-1]$
22. Insert a number of m different positive integers from 2 to $[n/s]$ into Se
23. **end if**
24. num = 0
25. while num < m
26. swap C[re[se[num]][0]]and C[re[se[num+1]][0]]
27. swap C[re[se[num]][1]]and C[re[se[num+1]][1]]
28. num = num + 2
29. **end while**
30. **End**

3.6 模拟退火算法

模拟退火算法是由 Metropolis 等人在 1953 年提出并于 1982 年应用于工业领域的一种仿自然过程的算法. 模拟退火算法的思想来源于固体退火原理. 模拟退火通过温控参数解决了优化算法在探索(explore)和利用(exploit)上的两难问题,在早期通过较高的温度使得算法在全局范围搜索,而在后期算法能较快的收敛到当前的找寻到的最优区域内,模拟退火伪代码如算法 4.

算法 4 模拟退火算法

1. **Begin**
2. $t<-0$
3. initialize T
4. select an initial point randomlyν_c
5. estimate ν_c
6. **repeat**
7. **repeat**
8. select a new point ν_n in the area of ν_c
9. **If** eval$(\nu_c)<$eval(ν_n)
10. **then**$\nu_c<-\nu_n$
11. **else if** random$[0,1]<(\text{eval}(\nu_n)-\text{eval}(\nu_c))/T$
12. **then** $\nu_c<-\nu_n$
13. **until** reach the end condition
14. $T<-T\times\alpha$
15. $t=t+1$
16. until Satisfy the rules of downtime
17. **End**

3.7 基于模拟退火的离散型自适应布谷鸟算法

SA-ADCS 算法描述如下:

Step 1:设置算法参数:发现概率 P_a,最大迭代次数 max_iter,经验参数 $\alpha_{\max}$ 和 $\alpha_{\min}$,鸟窝数 N,每段的城市数 s. 用 3.2 节方法生成初始路径,并以 3.1 节所提表示方法对解进行编码,并计算出每个解的适应度.

Step 2:对每个鸟窝内的解用 3.4 节的自适应型局部调整算子经行局部调整后再用 2-opt 优化调整后的路径,计算机新路径的适应度,按照模拟退火算法依概率选择是否接受该路径,若当前路径适应度比全局最优解大,以当前路径为全局最优解.

Step 3:计算每个鸟窝的发现概率,若鸟窝被发现,则用 3.5 节的全局随机扰动策略对该鸟窝内的路径进行全局的扰动并用 2-opt 优化扰动后的路径,若比原路径适应度大则用扰动后路径替换原路径,若比原路径适应度小则丢弃. 若当前路径适应度比全局最优解大,以当前路径为全局最优解.

Step4:检查算法是否到达最大迭代次数,若到达,停止算法,输出全局最优解,若未到达重复 step2 ~ step3.

3.8 算法时间复杂度分析

2-opt 优化算法作为一种局部优化算法其时间复杂

度为$O(n^2)$,本文算法在设计局部调整策略和全局调整策略时均与 TSP 的规模有关,因此局部调整策略和全局调整策略的时间复杂度为 $O(n)$. 本文算法在对可行解进行局部调整策略和全局调整策略时均采用了 2-opt 算法对新解进行局部优化,故本文算法的时间复杂度为$O(n^2)$. 求解 TSP 的确定性算法的时间复杂度呈现指数增长的趋势,本文算法可在多项式时间内求解 TSP,是求解 TSP 的一种有效算法.

4　仿真实验

本文提出了一种基于模拟退火的离散型布谷鸟算法(SA-ADCS)求解 TSP,并用国际标准的 TSPLIB 数据集进行仿真实验. 实验环境为 matlab2010b Intel(R) Pentium(R) CPU G3220 3GHz,and 4GB of RAM.

4.1　参数设置

算法参数设置如表 1 所示.

表 1　SA-ADCS 参数设置

参数定义	参数名	值
种群规模	N	15
发现概率	P_a	0.25
最小概率	$\alpha_{\min}$	0.4
最大概率	$\alpha_{\max}$	0.9
迭代次数	max_iter	200
初始温度	T	200 × citynumber
衰减因子	A	0.85

算法中 N 和 max_iter 与 DBA 保持一致,P_a 参考基本布谷鸟算法,$\alpha_{\min}$ 和 $\alpha_{\max}$ 是根据大量实验得出来的经验参数,初始温度 T 的取值为 200 倍的城市规模,这样设置的根据是在城市规模较小时,算法易于收敛到最优值,故而采用较小的初始温度,但是城市规模较大的时候,算法难于收敛到最优值,故而采用较高的初始温度,使得算法在可行解区域内进行更加全面的搜索以提升算法的搜索性能.

4.2　结果与讨论

表 2 展示了 SA-ADCS 对 10 个 TSPLIB 标准数据集的测试结果. 本文对每个数据集独立运行 30 次得出实验结果. 列“Instance”表示测试所用的 TSPLIB 算例,列“Optimal”表示该算例的已知最优解,列“Best”表示该算例独立运行 30 次所得出的最优解,列“Worst”表示该算例独立运行 30 次所得出的最差解,列“Average”表示该算例独立运行 30 次所得出的平均值,列“Std”表示独立运行 30 次的值所构成的数列的标准差.

列“PDa(%)”计算公式如下:

$$\mathrm{PDa}(\%)=\frac{\text{Average-Optimal}}{\text{Optimal}}\times 100\% \tag{3}$$

列“PDb(%)”计算公式如下:

$$\mathrm{PDb}(\%)=\frac{\text{Best-Optimal}}{\text{Optimal}}\times 100\% \tag{4}$$

列“$C1\%/\mathrm{Copt}$”表示该在独立运行 30 次中,与最优解偏差在 1% 内的次数,列“Time”表示该算例独立运行 30 次所用的平均时间. 本文选择的 10 个 TSPLIB 标准数据集包含不同的城市规模,全面验证了本文所提出的 SA-ADCS 算法的性能. 表中用黑色字体标出的数据,表示 SA-ADCS 算法已经找出该 TSPLIB 标准算例的已知最优解. 通过 30 次独立测试的结果,也可以看出一个算法的稳定性,从而证明本文算法所得结果不是偶然事件.

表 2　SA-ADCS 求解 41 个 TSP 算例结果

Instance	Optimal	Best	Worst	Average	Std	PDa(%)	PDb(%)	C1%/Copt	Time(s)
pr76	108,159	**108,159**	**108,159**	**108,159**	**0.00**	**0.00**	**0.00**	**30/30**	8.04
kroA100	21,282	**21,282**	**21,282**	**21,282**	**0.00**	**0.00**	**0.00**	**30/30**	8.35
kroA150	26,524	**26,524**	26,566	26,534.17	17.07	0.04	**0.00**	30/10	21.02
kroA200	29,368	**29,368**	29,503	29,418.3	32.99	0.17	**0.00**	30/1	38.45
lin318	42,029	42,124	42,410	42,248.4	79.75	0.52	0.23	30/0	66.04
pr439	107,217	107,264	107,812	107,535.5	124.81	0.30	0.04	30/0	135.29
rat575	6773	6,846	6,901	6,882.33	20.34	1.61	1.08	0/0	396.81
rat783	8806	8,920	8,993	8,950.77	12.15	1.64	1.29	0/0	782.88
pr1002	259,045	262,140	264,186	263,351.83	477.26	1.66	1.19	0/0	1429.97
nrw1379	56,638	57,481	57,751	57,619.6	112.29	1.73	1.49	0/0	6279.51

表 3 为 SA-ADCS 算法与 DCS[23] 和 DBA[16] 的实验对比表. 每种算法独立运行 30 次. 表格第一例是算例名称,列‘optimal’代表已知最优值,列 ‘Best’代表三种算法求解的最优解,列‘Average’代表三种算法求解的平均值. 黑体标出的数据表示 SA-ADCS 已经找到了最优解或者求解结果优于 DCS 和 DBA.

表 3 DCS,DBA 和 SA-ADCS 求解结果比较

	Optimal	DCS[23]		DBA[16]		SA-ADCS	
		Best	Average	Best	Average	Best	Average
pr76	108,159	**108,159**	**108,159**	**108,159**	**108,159**	**108,159** *	**108,159**
kroA100	21,282	**21,282**	**21,282**	**21,282**	**21,282**	**21,282** *	**21,282**
kroA150	26,524	**26,524**	26,569.26	**26,524**	26,560.2	**26,524** *	26,534.17
kroA200	29,368	29,382	29,446.66	**29,368**	29,449.23	**29,368** *	29,418.3
lin318	42,029	42,125	42,434.73	42,154	42,462.16	42,124 *	42,248.4
pr439	107,217	107,447	107,960.5	107,291	107,683.33	107,264 *	107,535.5
rat575	6773	6,896	6,956.73	6,862	6,903.83	6,846 *	6,882.33
rat783	8806	9,043	9,109.26	8,948	9,010.4	8,920 *	8,950.77
pr1002	259,045	266,508	268,630.03	266,146	266,412.8	262,140 *	263,351.83
nrw1379	56,638	58,951	59,349.53	58,188	58,299	57,481 *	57,619.6

表3 选择了 10 个不同规模的 tsp 算例,因此证明了 SA-ADCS 算法优于 DCS 和 DBA 算法.

为了验证该 SA-ADCS 算法的可靠性,再次与其他具有代表性的算法进行比较. 利用双边 t-检验对其他四种算法进行了分析. t 值按下面所示的公示(5)计算. 其中 $n1$,$n2$ 是样本数,在本文中它代表了算法的独立运算次数.

$$t=\frac{\mathrm{Meanbst}_1-\mathrm{Meanbst}_2}{\sqrt{\frac{(n_1-1)\mathrm{Std}_1+(n_2-1)\mathrm{Std}_2}{n_1+n_2-2}}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} \tag{5}$$

在以上检验中,当显着性水平 $\alpha=0.01$,得到 $t_{0.005}=2.660$. 当 $|t|>2.660$,SA-ADCS 被认为与比较算法有显着差异. 对于没有通过提出的算法实现理论最优值的情况进行了着重分析,其中“*”表示在比较文献中未测试,“-”表示 SA-ADCS 作为参考或其他算法达到理论上最优.

从表 4 所示的结果可以看出,在大多数情况下,与其他算法相比 SA-ADCS 的 $|t|$ 大于 2.660,只有在实例 eil76 和一些改进的算法的情况下 $|t|$ 小于 2.660,表明 SA-ADCS 在解决此类问题方面有显着的改进.

从表 5 所示的结果可以看出,对于大多数在 SA-ADCS 算法下运行 30 次的实例,每次运行的稳定性都达到最优解算法且优于 RKCS 算法[24]. 6 个实例均达到已知的最优解,而 RKCS 算法在实例 pr136 上没有达到最优解.

表 4 SA-ADCS 与其他算法的可靠性比较

Instance	DCS[23]	DBA(Mean)[16]	HGA(Standard deviation)[14]	ACE[25]	SA-ADCS
eil76	2.249	1.510	*	0.231	-
eil101	3.878	17.855	*	5.727	-
kroA200	1.533	1.845	*	*	-
lin318	5.056	7.360	40.990(107.2)	*	-
rd400	7.209	4.247(15488.0)	53.202(58.5)	*	-
fl417	11.634	12.033(460.9)	*	*	-
pr439	5.110	6.825(108011)	*		-
rat575	9.910	5.148(6908.7)	*		-
rat783	21.713	4.566(8972.1)	*	63.408	-
pr1002	23.624	34.526(266405.3)	*	-	-
nrw1379	39.223	26.851(58287.9)	*	*	-

表 5　SA-ADCS 与 RKCS[21] 算法结果比较

Instance	Optimal	RKCS[21]			SA-ADCS		
		Best	Average	PDav(%)	Best	Average	PDav(%)
eil51	426	**426**	426.9	0.21	**426**	426	0
st70	675	**675**	677.3	0.34	**675**	675	0
eil101	629	**629**	631.1	0.33	**629**	630.43	0.23
pr136	96,772	97046	97708.9	0.97	**96,772**	97,009.26	0.25
pr144	58,537	**58537**	58554.45	0.03	**58,537**	58,537	0

图 2 和图 3 分别是算例 rd400,rat575 和 PR1002 的收敛曲线. 从收敛曲线可以看出,该算法的振荡幅度较大,这是由于模拟退火算法接受更差的解. 由于贪心算法作为初始种群,在初始阶段的接受概率较高,使得算法始终在最优解附近搜索. 它保证了在后期加快收敛速度以及优质种群的保留. 从图中可以看出 SA-ADCS 算法具有较强的跳出局部最优的能力.

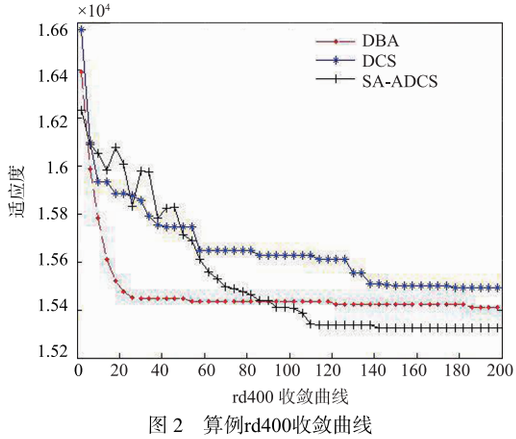


图 2　算例rd400收敛曲线

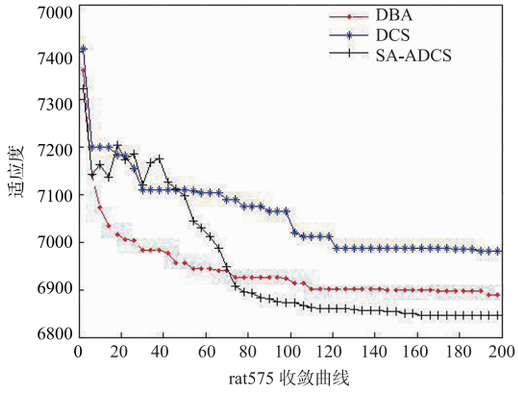


图 3　算例rat575收敛曲线

5　实际应用案例

在 SA-ADCS 求解 TSP 的理论基础之上,本节将 SA-ADCS 运用于求解杭州主城区某公司的低温奶配送研究,求解低温奶网点配送最短路径. 选取该城区的 126 家网点配送进行研究. 选取该公司 2017 年 11 月到 2017 年 12 月的配送量数据,计算该公司的网点平均单次配送量用以粗略表示各自营网点单次的配送量. 已知配送车辆最大装载量为三顿,车辆需要从配送中心出发,完成任务后再返回配送中心.

首先不考虑配送量约束,直接对所有网点进行 k-medoids 聚类,聚类结果见图 4. 其次,考虑配送量约束,直接对所有网点进行 k-medoids 聚类,聚类结果见图 5. 配送网点编号规则为:从左往右从上往下依次编号.

图4　不考虑配送量约束的k-medoids聚类结果图

显然两种情况下,考虑约束的类内配送量较为稳定,聚类效果更佳. 将图中的点进行坐标编码,并且求出各个聚类中心和聚类点集. 各聚类中心点分布情况见图 5. 在聚类结果的基础之上,进行车辆路径规划,用 SA-ADCS 进行求解,具体结果见表 6.

计算结果可知,原始里程为 1214.35,优化后的里程是 1048.42,因此 SA-ADCS 求解的结果有了一定程度的提升,为路径规化问题提供了一种新的思路,是解决

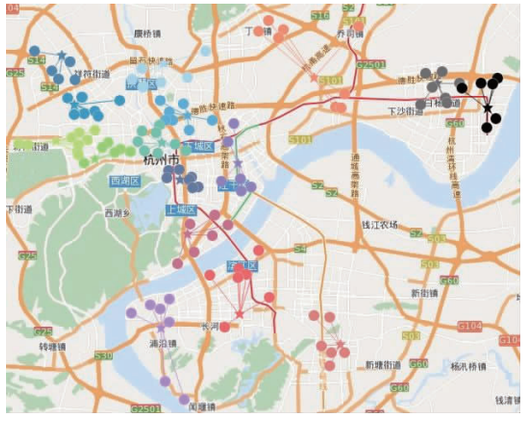


图5 考虑配送量约束的k-medoids聚类结果图

此类问题有效的方法.

表6 基于SA-ADCS的车辆路径规划结果

路线编号	配送路径	运输距离
1	0－11－5－124－1－9－123－2－0	63
2	0－14－4－8－13－29－33－31－34－22－20－0	64.5
3	0－74－10－6－17－23－15－19－7－0	56
4	0－55－50－38－39－53－54－25－26－18－0	67.72
5	0－28－49－42－56－52－45－44－51－47－43－0	74.68
6	0－110－113－69－70－109－112－114－117－0	86
7	0－73－71－75－86－72－95－83－0	58.5
8	0－104－103－106－125－107－102－105－0	66.2
9	0－32－46－40－58－41－59－0	61.1
10	0－66－65－63－62－64－68－67－0	83.5
11	0－79－82－84－78－77－98－94－0	65.04
12	0－88－97－99－96－89－92－80－101－122－0	48.27
13	0－119－111－116－108－115－120－121－119－0	81.56
14	0－36－60－57－48－37－61－0	63.76
15	0－3－30－16－21－24－27－0	47.38
16	0－93－76－81－100－87－91－85－90－0	62.21

6 结论

本文将模拟退火算法和布谷鸟算法相结合,在TSP上的求解展现出良好的性能.本文算法虽然求解精度较高,但同时也付出了较高的复杂度,在上文的复杂性分析中可以看出,本文算法时间复杂度为$O(n^3)$.笔者下一步工作主要分为两点:(1)优化算法的时间复杂度.争取在保持本文算法的求解精度的同时,降低算法的复杂度,将算法的复杂度降低到$O(n^2)$或者$O(n\log(n))$.(2)将本文算法的设计思路推广到其它领域的组合优化问题上,例如二维排样问题.

参考文献

[1] Applegate DL, Bixby RE, Chvátal V, Cook WJ. The Traveling Salesman Problem: A Computational Study (Princeton in Applied Mathematics)[M]. New Jersey: Princeton University Press, 2007.

[2] Rego C, Gamboa D, Glover F, Osterman C. Traveling salesman problem heuristics: Leading methods, implementations and latest advances[J]. European Journal of Operational Research, 2011, 211(3): 427－441.

[3] Junger M, Reinelt G, Rinaldi G. Handbooks in OR&MS: Chapter4 the Traveling Sales Man Problem[M]. Amsterdam: Elsevier Science, 1995.

[4] Lawler EL, Wood D E. Branch-and-bound methods: A survey[J]. Operations Research, 1966, 14 (4): 699－719.

[5] Bellman RE, Dreyfus SE. Applied Dynamic Programming. Princeton[M]. New Jersey: Princeton University Press, 1962.

[6] Holland JH. Adaptation in Natural and Artificial Systems [M]. Ann Arbour: The University of Michigan Press 1975.

[7] Kirkpatrick S, Gelatt Jr CD, Vecchi M P. Optimization by Simulated Annealing[J]. Science, 1983, 220 (4598): 671－680.

[8] Dorigo M, Gambardella L M. A study of some properties of Ant-Q[A]. Lecture Notes in Computer Science[C]. 1996 (1141): 656－665.

[9] Yang XS, Deb S. Cuckoo search via le′vy flights[A]. Nature & biologically inspired computing, NaBIC World congress on[C]. IEEE, 2009. 210－214.

[10] Krishnand KN, Ghose D. Detection of multiple source location using aglowworm metaphor with applications to collective robotics[A]. Proc of IEEE Swarm Intelligence Symposium Piscataway[C]. IEEE Press, 2005. 84－91.

[11] Kenney J, Eberhart R. Particle swarm optimization[A]. Proceedings of IEEE Conference on Neural Networks [C]. Perth, Australia, 1995.

[12] Mostafa Mahi, Ömer Kaan Baykan, Halife Kodaz. A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem[J]. Applied Soft Computing, 2015, 30: 484－490.

[13] Bullnheimer B, Hartl R F, Strauss C. A new rank based version of ant system: A computational study[J]. Central European J for Operations Research and Economics, 1999, 7(1): 25－38.

[14] Yong Wang. The hybrid genetic algorithm with two local optimization strategiesfor traveling salesman problem[J]. Computers & Industrial Engineering, 2014, 70: 124－133.

[15] Saban Gulcu, Mostafa Mahi, Omer Kaan Baykan, Halife

Kodaz. A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem [J]. Soft Comput, 2016, 1: 1 – 17.

[16] Yassine Saji, Mohammed Essaid Riffi. A novel discrete bat algorithm for solving the travelling salesman problem [J]. Neural Comput&Applic, 2016, 27: 1853 – 1866.

[17] Maribel Guerrero, Oscar Castillo, Mario García Valdez. Cuckoo search via lévy flights and a comparison with genetic algorithms [J]. Fuzzy Logic Augmentation of Nature-Inspired Optimization Metaheuristics, 2015, 91 – 103.

[18] Maribel Guerrero, Oscar Castillo, Mario García Valdez. Fuzzy dynamic parameters adaptation in the Cuckoo Search Algorithm using Fuzzy logic [J]. CEC, 2015, 441 – 448.

[19] Claudia I González, Patricia Melin, Juan R Castro, Oscar Castillo, Olivia Mendoza. Optimization of interval type-2 fuzzy systems for image edge detection. Appl [J]. Soft Comput, 2016, 47: 631 – 643.

[20] Pavlyukevich I. L ′ evy flights, non-local search and simulated annealing [J]. Computational Physics, 2007, 226, 1830 – 1844.

[21] Pavlyukevich I. Cooling down L ′evy flights [J]. J Phys A: Math Theor, 2007, 40, 12299 – 12313.

[22] Chiang CW, Lee WP, Heh J S. A 2-Opt based differential evolution for global optimization [J]. Applied Soft Computing Computing, 2010, 10 (4): 1200 – 1207.

[23] Aziz Ouaarab · Belal" d Ahiod · Xin-She Yan. Discrete cuckoo search algorithm for the travelling salesman problem [J]. Neural Comput & Applic 2014, 24: 1659 – 1669.

[24] Aziz Ouaarab · Belal" d Ahiod · Xin-She Yang. Random-key cuckoo search for the travelling salesman problem [J]. Soft Comput 2015, 19: 1099 – 1106.

[25] Jose B. Escario , Juan F. Jimenez, Jose M. Giron-sierra. ant colony extended: experiments on the travelling salesman problem [J]. Expert System with Applications, 2015, 42: 390 – 410.

作者简介

张子成　男,硕士研究生,研究方向为人工智能、数据挖掘等.
E-mail:269627853@ qq. com

韩　伟(通信作者)　男,副教授,硕士生导师,研究方向为人工智能、运筹学,多智能体系统等.
E-mail:dallashw@ gmail. com

毛　波　男,副教授,硕士生导师,研究方向为三维模型综合简化、在线可视化、数据挖掘以及物联网应用.
E-mail:99454961@ qq. com