文章编号: 1001-0920(2007)09-1061-04

基于规模压缩的混合蚁群算法

严建峰, 李伟华, 杜 北
(西北工业大学 计算机学院, 西安 710072)

摘 要: 为了提高蚁群算法处理大规模问题的性能,提出一种基于规模压缩的混合蚁群算法. 根据 TSP 问题的最优解与次优解共享部分路径片断的原理,设计城市压缩算法,减少了 TSP 问题的城市处理量. 在求解过程中,引入最优解的区域特征的概念,采用优化状态转移规则,压缩了解空间. 仿真实验结果证明,采用所提出算法得到解的质量和收敛速度都有显著提高.

关键词: 蚁群算法; 规模压缩; 路径片断; 区域特征
中图分类号: TP391 **文献标识码**: A

Hybrid ant colony algorithm based on scale compression

YAN Jian-feng, LI Wei-hua, DU Bei
(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China. Correspondent: YAN Jian-feng, E-mail: silent.valley@hotmail.com)

Abstract: To improve the performance of ant colony algorithm in solving large-scale TSP problem, a hybrid ant colony algorithm based on scale compression is proposed. Genetic algorithm is used to generate a suboptimal solution set and calculate their intersection. By eliminating all cities mapped by the elements among the intersection in the primal TSP problem, the original problem is converted into a new one with smaller scale. In addition, an optimal state transition rule is designed based on regional characteristics of optimal solutions to accelerate convergence speed. Simulation results show the approach possesses high searching ability and excellent convergence performance.

Key words: Ant colony algorithm; Scale compression; Segment; Regional character

1 引 言

蚁群算法(ACA)是受蚂蚁搜索食物过程启示而提出的启发式优化算法,属于模拟进化算法. 旅行商问题(TSP)和蚁群系统有很多相似性,是蚁群算法最早的成功应用. 蚁群算法具有分布式计算的特点且易与其他模拟进化算法结合,适用于求解可用图表示的 NP-hard 问题. 蚁群算法及各类改进算法已被广泛应用于求解车辆路由问题、模糊规则提取、供应链管理和图像识别等领域,并且取得了较好的结果.

与其他模拟进化算法类似,蚁群算法同样存在计算时间长、容易出现停滞的缺点,特别当处理大规模问题时,算法收敛速度过慢. 现有的研究从几个方面改进蚁群算法的性能:结合其他模拟进化算法和理论(如遗传算法[1]和云模型理论[2]),限制蚁群算法陷入局部最优解;采用多信息素策略,使算法不易陷入局部最优;丰富蚂蚁基本行为来加快收敛速度[3];采用优化信息素更新策略[4]和自适应改变信息素挥发参数[5]来加快收敛速度并且防止早熟.

这些改进算法主要考虑从蚁群系统本身的信息素控制策略和避免蚁群系统陷入最优解两个方面进行优化,由于模拟进化算法本身迭代求解的运行机制限制,改进后的蚁群算法虽然能够取得较高质量的解,但在处理大规模问题时收敛速度仍不尽人意.

本文从不同角度,提出一种基于规模压缩的混合蚁群算法(SCH-ACA). 算法以经典 TSP 问题为背景,从两方面对问题规模进行压缩:1) 产生 n 城市 TSP 问题的一组初始次优解,并提取其共同路径片断,通过城市压缩算法,将 n 城市 TSP 问题转化为 $n-u$ 城市 TSP 问题;2) 根据解的区域特征原理,采用优化的状态转移规则,压缩解空间并求解. 本文的算法是通过压缩城市数量和解空间两方面进行算法优

收稿日期: 2006-06-09; **修回日期**: 2006-09-11.
基金项目: 国家部委基金项目(9140A17050206HK03).
作者简介: 严建峰(1978—),男,江苏昆山人,博士生,从事智能决策系统的研究; 李伟华(1951—),男,湖北黄冈人,教授,博士生导师,从事智能决策系统、网络安全的研究.

化,因此在处理大规模问题时,算法显示出很好的时间性能.仿真实验表明在求解大规模问题时,算法在时间性能和所获解的质量上都有显著提升.

2 基于规模压缩的混合蚁群算法

TSP 问题可描述为寻找遍历 n 个城市的最短路径.设 X 是城市编号的集合,TSP 的数学描述为搜索自然数集 $X = \{1,2,\cdots,n\}$ 的一个排列 $C = \{c_1, c_2, \cdots, c_n\}$ 或路径 $c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_n$,使路径 $L = \sum_{i=1}^{n-1} d(c_i, c_{i+1}) + d(c_n, c_1)$ 最短(d 为城市 c_i 到 c_{i+1} 的距离).TSP 问题是 NP-hard 问题,若采用穷举法,以每秒 1 亿次的计算速度来估算,当 TSP 问题包含 20 个城市时,求解时间长达近 400 年.有效地解决 TSP 问题对解决其他 NP-hard 问题具有重要的启发意义.

2.1 城市压缩算法原理

蚁群系统采用信息素机制,不断强化已获取的路径信息,充分利用已有知识来优化解的质量,蚁群系统的迭代求解机制是次优解向最优解不断优化的过程.作者在大量仿真试验中观察到,最终的最优解和中间过程的次优解之间,往往有大量重复的路径片断.

定义 1 路径片断是一个不可分割的城市序列 $SEG_i = c_{pi} \rightarrow \cdots \rightarrow c_{qi}, 1 < p_i < q_i < n$,即 TSP 问题的一个解 $c_1 \rightarrow \cdots \rightarrow c_{pi} \rightarrow \cdots \rightarrow c_{qi} \rightarrow \cdots \rightarrow c_n$ 等价于 $c_1 \rightarrow \cdots \rightarrow SEG_i \rightarrow \cdots \rightarrow c_n$.

根据定义,路径片断 SEG_i 具有原子性特性:如果蚂蚁选择了 SEG_i 的第一个城市 c_{pi} 作为下一个近邻城市,则必须按照固定顺序 $c_{pi} \rightarrow \cdots \rightarrow c_{qi}$ 一次性遍历完 SEG_i,即可以把 SEG_i 看作一个入口为 c_{pi},出口为 c_{qi} 的临时城市 $Ctemp_i$.根据该思想,设次优解 Sol_1 与最优解 Sol_2 有共同路径片断 SEG_i,在次优解 Sol_1 向最优解 Sol_2 演化的过程中,用 $Ctemp_i$ 替换整个 SEG_i 片断后再对问题求解,可得到一个中间过程最优解 Sol_2';最后,用片断 SEG_i 替换 Sol_2' 中的 $Ctemp_i$,便可获得原问题的最优解 Sol_2.通过一次替换,蚁群系统处理同样的一个 TSP 问题时,需处理的城市数量减少了 $q_i - p_i$ 个,而解的质量并没有受到任何影响.若次优解 Sol_1 与最优解 Sol_2 有 r 个共同路径片断,则通过压缩处理,减少的城市处理量总计为 $u = \sum_{i=1}^{r}(q_i - p_i)$.

本文的思想就是通过寻找所有共同路径片断,在不影响解的质量的前提下,压缩城市数量,把原问题从 n 城市 TSP 问题转化为 $(n-u)$ 城市的新问题,提高算法收敛速度.由算法复杂性分析理论,如不采取任何处理,则 m 个蚂蚁要遍历 n 个城市,经过 N 次循环,其算法复杂度为 $O(N * m * n^2)$;而算法优化后的复杂度为 $O(N * m * (n-u)^2)$.在大量仿真实验中观察到,通过规模压缩处理,减少的城市处理量占城市总量的 20 % 左右,对算法性能有明显提升.

2.2 城市压缩算法设计

次优解与最优解的共同路径片断交集是城市压缩算法的基础,但是在算法初始化阶段得不到最优解,因此也不能直接得到这样理想的路径片断交集.在算法中,使用一组次优解共同路径片断集合作为一个近似逼近解来替代次优解与最优解共同的路径片断交集.该逼近近似解替代机制基于以下假设:若 h 个不相关的次优解都包含路径片断 SEG_i,则最优解将以较大概率包含 SEG_i,而最优解包含 SEG_i 的概率是这组相互不相关的次优解的数量 h 的一个非递减函数 $f(h)$.表 1 所示的多组对比实验验证了假设的正确性.

表 1 参数 g,h 对算法的影响

g/ %	h	C_num	C_per/ %	time	length
5	7	29	12.9	394.2	3 924
10	7	43	19.1	384.5	3 930
15	7	52	23.1	377.8	4 010
20	7	59	26.2	371.0	4 021
10	3	43	19.1	355.4	4 015
10	6	43	19.1	369.1	3 927
10	9	43	19.1	403.4	3 924
10	12	43	19.1	452.6	3 924

为了获得近似逼近解,需要产生一组次优解.文献[6]提到一种利用蚁群并行计算得到多个次优解的机制,但是该机制一方面基于大规模并行系统求解,不具有普遍适用性;另一方面,由蚁群算法得到次优解的过程,收敛速度不能让人满意.

遗传算法是一种比较成熟的模拟进化算法,易与蚁群算法结合,并且本身具有快速性、随机性、全局收敛性的特点,而产生的解相关性不强,符合算法要求.采用遗传算法可产生多个次优解,进行片断提取,得到共同路径片断集合.本文选用贪心遗传算法产生一组初始次优解,具体实现可参考文献[7].

获得一组次优解后,共同片断提取算法描述如下:1) 设有 s 个次优解,任选一个次优解 Sol_1 作为基准次优解,并把 $Sol_1 = \{c_{11}, c_{12}, \cdots, c_{1n}\}$ 映射到自然排列 $X_1 = \{1,2,\cdots,n\}$,即城市 c_{11} 的编号为 1,城市 c_{12} 的编号为 2,依次类推.2) 根据基准次优解 Sol_1 与自然排列 X_1 的映射所规定的城市与编号间的对应

关系,同样建立起其他所有次优解 $Sol_i(i=2,3,\cdots,s)$ 到某个排列 $X_i(i=2,3,\cdots,s)$ 的映射. 3) 找出排列 $X_i(i=2,3,\cdots,s)$ 中连续自然数片断的交集,并根据 Sol_1 与 X_1 的映射关系,把这些连续自然数片断转化成城市序列片断,便得到这组次优解的共同片断.

假设得到 1 组 3 个次优解如下:

$$Sol_1 = \{c_1, c_5, c_4, c_2, c_3, c_6\},$$

$$Sol_2 = \{c_3, c_6, c_5, c_4, c_2, c_1\},$$

$$Sol_3 = \{c_2, c_5, c_4, c_3, c_6, c_1\}.$$

并设 Sol_1 为基准次优解,则 Sol_1 所对应自然排列为 $X_1 = \{1,2,3,4,5,6\}$. 根据基准次优解 Sol_1 与自然排列 X_1 映射规则,那么次优解 Sol_2 与 Sol_3 分别映射到排列 $X_2 = \{5,6,2,3,4,1\}$, $X_3 = \{4,2,3,5,6,1\}$. X_1 与排列 X_2 和 X_3 中包含的相同的自然数片断集合包含两个元素,即 $\{\{2,3\},\{5,6\}\}$,则根据映射的一一对应原理,可以提取 Sol_1, Sol_2 和 Sol_3 的两个共同路径片断 $SEG_1 = c_5 \rightarrow c_4$, $SEG_2 = c_3 \rightarrow c_6$.

2.3　解空间压缩算法

蚁群系统中,位于城市 i 的蚂蚁 k 移动到下一个城市时,根据一定转移概率 $P_{ij}^k(t)$ 从集合 $allowed_k$ 中选择城市 j 作为目标城市. 这里, $allowed_k$ 表示蚂蚁 k 未遍历的城市集合,均值为 $n/2$,当城市规模较大时,解空间较大,影响蚁群算法的性能.

而根据对蚁群算法得到最优解的大量统计[8],最优解具有区域特征,即最优解中每一个城市都选择距离自己最近的前 Max 个城市中的一个作为近邻城市 (100 城市以下, Max = 9; 300 城市以下, Max = 15; 1 000 城市以下 Max = 20). 当 n 较大时, Max 比 $n/2$ 小得多,因此蚂蚁只需从 Max 而不是平均 $n/2$ 个城市中选择近邻城市. 在解空间压缩算法中,对每个城市 i 应维护临时数据结构 Allow(i),记录距离该城市 i 最近的前 Max 个城市. 位于城市 i 的蚂蚁 k 直接从 Allow(i) 中选择一个作为下一个需遍历城市. 对于大规模问题,通过采用解的区域特征,将解空间由原始的 $n!$ 缩小为 $n^{\max c}$,进一步加快算法的收敛速度.

2.4　算法实现

具体算法实现如下:

1) 利用贪心遗传算法产生一组次优解.

2) 利用片断提取算法,得到次优解的共同路径片断交集 S_{Set}.

3) 对交集 S_{Set} 中每一个元素,用临时城市 $Ctemp_i$ 替代路径片断 SEG_i,把包含 n 个城市的原问题 P 转化为包含 $n-u$ 个城市的新问题 P'.

4) 利用解空间压缩算法,根据 P' 的城市个数 $n-u$ 确定参数 Max,为每个城市维护 $allow_i$.

5) 利用蚁群算法,求解 P',得到 P' 的最优解 Sol'_{best}.

6) 用 SEG_i 替换 Sol'_{best} 中的所有临时城市 $Ctemp_i$,得到原问题 P 的最优解 Sol_{best}.

3　仿真实验

SCACA 的参数除了基本蚁群算法的参数 α,β,ρ 和 Q 之外,另外必须考虑片断长度的上限 g(即片断包含的城市数目占城市总数的比值) 和次优解的个数 h. 若 g 过大,过长的片断可能包含实际最优解,导致算法陷入局部最优解,影响算法性能; 若 g 过小,太短的片断引起的算法性能提升不显著. 而 h 若取的过大,会增加产生初始解的计算时间; 若 h 值过小,则会影响共同路径片断的质量. 本文取 $Q=100$, $\alpha=\rho=0.1$, $\beta=2$,对 g 和 h 进行测试,求解 TSP225 问题,实验结果如表 1 所示. 参数说明如下: C_num 和 C_per 指减少的城市处理量和百分比, time 表示算法所用时间, length 表示最优路径长度.

从表 1 可以看出,随着参数 g 增大,算法压缩的城市数量有显著的增加,所用时间有所减少,但是算法取得最优路径的可能变差. 在本例中,当 g 取 10 %,算法取得了较优的结果,同时所用时间也较少; 而随着参数 h 增大,算法获得解的质量更优,但耗时也更长; 当 h 到达一定界限时 (本例中 $h=6$),再增加参数 h 的值对算法求解精度提升无显著贡献. 这充分验证了本文的假设: 最优解包含共同路径片断的概率是这些相互不相关的次优解的数目 h 的一个非递减函数 $f(h)$.

为了便于算法性能对比,采取不同规模的 TSP 问题分别采用 ACS 和 SCH-ACA 进行对比测试,其中 ACS 的参数设置为: $Q=100$, $\alpha=\rho=0.1$, $\beta=2$; SCH-ACA 的参数设置为: $Q=100$, $\alpha=\rho=0.1$, $\beta=2$, $g=10\%$, $h=6$; 蚂蚁数目都取 10. 对每个 TSP 问题,算法都运行 10 次取平均值,每次运行迭代 300 次,结果如表 2 所示. 参数说明如下: Best solution 指已知的最优解, time, length 和 error 分别代表所用的时间、路径长度以及该路径长度与已知最优解的相对误差, time reduced 指改进算法与基本算法相比,节省的时间比值.

图 1 和图 2 分别给出了应用原始蚁群算法和基于规模压缩的混合蚁群算法 (SCH-ACA) 求解 TSP225 问题和 CH130 问题的进化曲线. 从实验结果看, SCH-AAC 在收敛时间上大大优于 ACS 算法,并且解的质量也优于 ACS 算法; SCH-ACA 算法的不足之处在于需要优化设置的参数过多,增加了参数组合优化的难度.

表 2 ACS 和 SCH-ACA 算法性能对比

TSP Instance	Best Solution	ACA			SCH-ACA			time reduced/ %
		time	length	error/ %	time	length	error/ %	
Berlin52	7 542	22.1	8 165	8.3	9.7	7 561	2.5	56.1
Pr76	108 159	41.6	129 562	19.8	14.7	108 277	1.1	64.7
Rd100	7 910	70.1	9 414	19.1	25.5	7 910	0	63.6
Ch130	6 110	112.8	7 704	26.1	36.1	6 122	1.9	68.0
Ch150	6 528	113.7	7 541	15.5	36.2	6 544	2.5	68.2
Tsp225	3 919	315.8	5 287	34.9	91.8	3 924	1.3	70.9

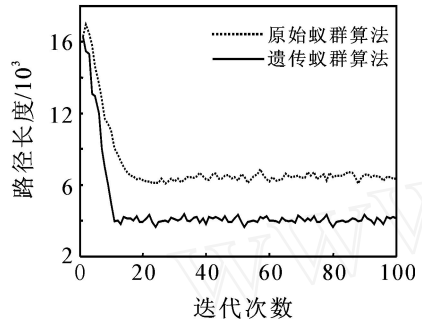

图 1 TSP225 进化曲线

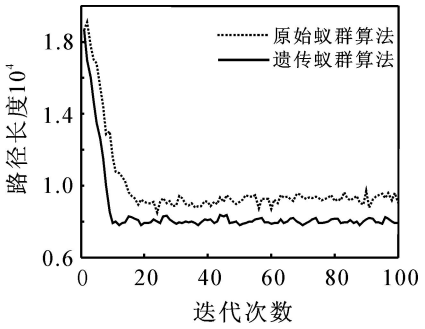

图 2 CH130 进化曲线

4 结 论

仿真实验证明,在求解 TSP 问题时,基于规模压缩的混合蚁群算法通过规模压缩处理,该系统可以减少约 20 %的城市处理量;结合解空间压缩处理,相比传统蚁群算法,蚁群系统收敛的时间缩短了 60 %左右,并且解的质量有显著改善.

参考文献(References)

[1] 陈宏建, 陈崚, 徐晓华,等. 改进的增强型蚁群算法[J]. 计算机工程, 2005, 31(2): 176-178.
(Chen Hong-jian, Chen Ling, Xu Xiao-hua, et al. An improved augment ant colony algorithm[J]. Computer Engineering, 2005, 31(2): 176-178.)

[2] 段海滨, 王道波, 于秀芬,等. 基于云模型理论的蚁群算法改进研究[J]. 哈尔滨工业大学学报, 2005, 37(1): 115-120.
(Duan Hai-bin, Wang Dao-bo, Yu Xiu-fen, et al. Improvement of ant colony algorithm based on cloud models theory[J]. J of Harbin Institute of Technology, 2005, 37(1):115-120.)

[3] 胡小兵, 黄席樾. 基于混合行为蚁群算法的研究[J]. 控制与决策, 2005, 20(1): 69-72.
(Hu Xiao-bing, Huang Xi-yue. On hybrid behavior based ant colony algorithm[J]. Control and Decision, 2005, 20(1): 69-72.)

[4] Sun Jun, Xiong Sheng-wu, Guo Fu-ming. A new pheromone updating strategy in ant colony optimization [J]. Machine Learning and Cybernetics, 2004, 1: 620-625.

[5] 冯远静, 冯祖仁, 彭勤科. 一类自适应蚁群算法及其收敛性分析[J]. 控制理论与应用, 2005, 22(5): 713-718.
(Feng Yuan-jing, Feng Zu-ren, Peng Qin-ke. Adaptive ant colony optimization algorithms and its convergence [J]. Control Theory & Applications, 2005, 22(5): 713-718.)

[6] 萧蕴诗, 李炳宇, 吴启迪. 求解 TSP 问题的模式学习并行蚁群算法[J]. 控制与决策, 2004, 19(8): 885-888.
(Xiao Yun-shi, Li Bing-yu, Wu Qi-di. Parallel model-learning ant colony optimization algorithm for TSP[J]. Control and Decision, 2004, 19(8): 885-888.)

[7] 魏英姿, 赵明扬. 求解 TSP 问题的贪心遗传算法[J]. 计算机工程, 2004, 30(19): 19-34.
(Wei Ying-zi, Zhao Ming-yang. A novel greedy genetic algorithm for traveling salesman problem[J]. Computer Engineering, 2004, 19(8): 885-888.)

[8] 全惠云, 文高进. 求解 TSP 的子空间遗传算法[J]. 数学理论与应用, 2002, 22(1): 36-39.
(Quan Hui-yun, Wen Gao-jin. Subspace genetic algorithm for TSP [J]. Mathematical Theory and Application. 2002, 22(1): 36-39.)