Vol. 16 No. 12
Dec. 2004
系 统 仿 真 学 报
JOURNAL OF SYSTEM SIMULATION

对一类带聚类特征 TSP 问题的蚁群算法求解

胡小兵[1]，黄席樾[2]
([1] 重庆大学数理学院，重庆 400044；[2] 重庆大学自动化学院，重庆 400044)

摘 要： 蚁群算法是近几年提出的一种新型的模拟进化算法，初步的研究表明该算法具有极强的鲁棒性和发现较好解的能力，但同时也存在收敛速度慢的缺点。针对带聚类特征的 TSP 问题，提出了一种新型的蚁群算法。该算法利用 TSP 问题本身所具有的聚类特征，从数据域上将其分解成多个子问题，对每个子问题分别采用蚁群算法并行求解，最后将所有子问题的解按一定规则合并成问题的解。对带聚类特征 TSP 问题的仿真实验表明该算法的收敛速度得到了极大的提高。

关键词： 蚁群算法；聚类；旅行商问题；组合优化问题；局部搜索
文章编号： 1004-731X (2004) 12-2683-04 **中图分类号：** TH116 **文献标识码：** A

Solving TSP with Characteristic of Clustering by Ant Colony Algorithm

HU Xiao-bing[1], *HUANG Xi-yue*[2]
([1] School of Mathematics &Science, Chongqing University, Chongqing 400044, China;
[2] School of Automation, Chongqing University, Chongqing 400044, China)

Abstract: Ant colony algorithm (ACA) is a novel simulated evolutionary algorithm which was proposed in recent years. Preliminary study has shown that the algorithm is very robust and has great capabilities in searching better solution, but at the same time there are some shortcomings such as converging slowly in the algorithm. To tackle traveling salesman problem (TSP) with characteristic of clustering, a new ACA algorithm is proposed. First the TSP problem is divided into several sub-problems by clustering processing, and then all the sub-problems will be solved in parallelization by ACA algorithm, respectively. At last all the solutions of each sub-problem will be merged into the solution of the TSP problem by some rules. Simulated experiment on TSP with characteristic of clustering shows that the convergence rate of the new algorithm has been greatly improved.

Keywords: ant colony algorithm; clustering; traveling salesman problem; combinatorial optimization problem; local search

引 言

上个世纪 90 年代初，意大利学者 M. Dorigo[1]等人提出了一种新型的模拟进化算法——蚁群算法(Ant Colony Algorithm, ACA)，并将该算法应用于著名的旅行商问题(Traveling Salesman Problem, TSP)，获得了较好的实验结果，但同时也发现该算法存在收敛速度慢、易出现停滞现象等缺陷[2]。针对该算法的不足，许多学者提出了改进的蚁群算法[3-5]，从一定程度上提高了算法的收敛速度，消除了算法中的停滞现象。但由于该算法的时间复杂度(为 $O(N\cdot n^3)$，n 为问题规模，N 为算法迭代次数)较高[2]，很难用于求解大规模的 TSP 问题（包括其它组合优化问题）。本文针对一类带聚类特征的 TSP 问题，提出了一种新型的蚁群算法——带聚类处理的蚁群算法（Clustering Processing Ant Colony Algorithm, CPACA）。CPACA 算法首先对 TSP 中的城市进行聚类处理，将 TSP 问题分解成许多小规模的子问题（子问题数目与城市的聚类数相等），然后利用蚁群算法对每个子问题并行求解，并将所有子问题的解按一定规则合并成待求解问题的解。该过程利用问题（数据）本身的（聚类）特征，对问题进行分解，利用蚁群算法对小规模问题具有较快的求解速度，并行求解每一个子问题，极大地加快了算法的求解速度。

收稿日期： 2003-12-12 **修回日期：** 2004-06-26
作者简介： **胡小兵**(1975-)，男，湖北京山人，讲师，博士，研究方向为计算智能、机器人控制技术；**黄席樾**(1943-)，男，重庆奉节人，回族，教授，博导，博士，研究方向为机器人控制技术、人工智能、机器视觉。

1 CPACA 的原理与实现

1.1 CPACA 的原理

因蚁群算法对小规模 TSP 问题（如 30 城市的 TSP 问题 Oliver30）有极高的性能[2]，如果能将大规模 TSP 问题分解成一些小规模子问题（文中指的是两座城市间的最短路径问题，其求解方式与 TSP 问题类似），对每个小规模子问题分别利用蚁群算法并行求解，最后将每个小规模子问题的解合并成待求解问题的解，这样将大大提高算法的性能。如何分解大规模 TSP 问题成为该算法的关键。为了找出分解 TSP 问题的规律，首先人为地构造两类具有不同分布特征的 TSP 问题。图 1 为球状分布 TSP 问题的最优解，城市可聚成 4 类，图 2 为线状和球状混合分布 TSP 问题的最优解，可聚成 3 类。

通过对图 2、图 3 的分析，发现 TSP 问题最优解的结构有如下的规律：

1) 设规模为 n 的 TSP 问题按城市的分布可聚成 c 类，设类 $i(i=1,2,\cdots,c)$ 中的城市数为 w_i，显然有 $\sum_{i=1}^{c} w_i = n$，则在类 $i(i=1,2,\cdots,c)$ 中，有且只有两座城市 $u_1^i, u_{w_i}^i$ 分别与另外两个不同的类相连。为方便起见，称城市 $u_1^i, u_{w_i}^i$ 为类 i 的

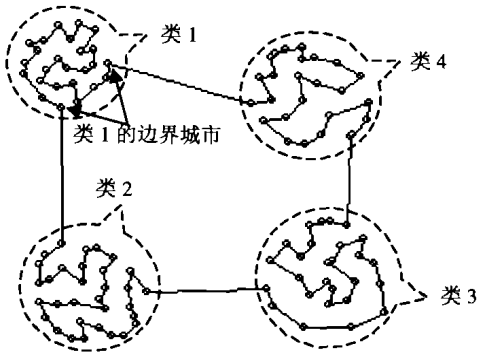

图 1 110 城市的 TSP 问题，使用 ACS 算法迭代 1000 次后的结果

图 2 44 城市的 TSP 问题，使用 ACS 算法迭代 1000 后的结果

边界城市；

2) 在类 $i(i=1,2,\cdots,c)$ 中，有一条从边界城市 u_1^i 经过剩下的城市一次且仅一次最后到达边界城市 $u_{w_i}^i$ 的路径，该路径是待求解 TSP 问题解的组成部分，且是类 i 中从城市 u_1^i 到城市 $u_{w_i}^i$ 的最短路径。因为如果该路径不是最短的，则 TSP 问题的解也非最优；

3) 如果将类 $i(i=1,2,\cdots,c)$ 本身看成一个超级城市 i，则由这 c 个超级城市构成一个新的 TSP 问题，且该新 TSP 问题的解即构成所有类的连接顺序。例如在图 1 中，4 个超级城市（类）构成一个新的 TSP 问题，每一类通过边界城市与其它类的边界城市相连。因此，新 TSP 问题的最优解即构成这些超级城市连接的顺序。图 1 中的顺序为：城市 1（类 1）→城市 2（类 2）→城市 3（类 3）→城市 4（类 4）→城市 1（类 1）。该连接顺序是新 TSP 问题的一个解，否则待求解 TSP 问题的解非最优。

显然，具有不同分布特征的 TSP（图 2）也满足以上的 3 条规律。因此，从上面的分析中受到启发，在求解带聚类特征的 TSP 问题时，首先对问题进行聚类处理，对所有的类，按规律 3）确定其连接顺序，再求出每类的边界城市，最后对类 $i(i=1,2,\cdots,c)$ 采用蚁群算法并行求解，生成一条从边界城市 u_1^i 到边界城市 $u_{w_i}^i$ 的最短路径，并将所有类中的路径按连接顺序合并起来，便生成了待求解 TSP 问题的一个可行解。

1.2 CPACA 的实现

1.2.1 聚类算法的选择

在 CPACA 算法中，首先需根据 TSP 问题中城市的分布特征选择不同的聚类算法，且应为基于距离的聚类算法 (distance-based clustering algorithm)。对于类内模式为球状分布的 TSP 问题，选择 C-均值法。图 1 中的类内模式即为这种情况；对于非球状的内类模式，如条状和线状分布特性的 TSP 问题，选用近邻函数法[6]。

1.2.2 类连接顺序

为了将所有类中的路径重新合并成最终问题的解，必须先求出各类的连接顺序。如图 1 所示，在 TSP 问题的最优解中，各类的连接顺序为：类 1→类 2→类 3→类 4→类 1。显然，为使 TSP 问题的解最优，类与类之间的所有连接边的长度之和应最小，该问题等价于求解以每一类作为一个“超级城市”的 TSP 问题的解。因此，可采用下面的方法确定类之间的连接顺序。

设可将 TSP 中的城市聚成 $U_1,U_2,.....,U_c$ 类（c 为类的个数），$K_i(i=1,...,c)$ 为每类的中心，其中 K_i 可能不是 TSP 中的城市。则将城市 $K_i(i=1,...,c)$ 看成一个 TSP 问题，并用蚁群算法对其求解，设求得的解为 $K_{i_1},K_{i_2},\cdots,K_{i_c},K_{i_1}$（其中 $i_1,i_2,\cdots,i_c$ 为 $1,2,\cdots,c$ 的一个排列），则排列 $i_1,i_2,\cdots,i_c$ 即为各类的连接顺序。

1.2.3 类中边界城市的确定

设类 p、类 q $(p,q=1,2,\cdots,c$ 且 $p\neq q)$ 按连接顺序相邻，$u_i^p(i=1,2,\cdots,w_p)$ 为类 p 中的城市，$u_j^q(j=1,2,\cdots,w_q)$ 为类 q 中的城市，则类 p、类 q 中的一个边界城市可按(1)式确定：

$$\{u_k^p,u_l^q\}=\arg\min_{\substack{i=1,2,\cdots,w_p\\ j=1,2,\cdots,w_q}} d(u_i^p,u_j^q) \tag{1}$$

其中 u_k^p 为类 p 中的一个边界城市，u_l^q 为类 q 中的一个边界城市；U_p，U_q 分别为类 p，类 q 中的城市集合；w_p、w_q 分别为类 p、类 q 中的城市数目。

当各类的连接顺序确定后，对任意类 i 只有一个直接前驱类和一个直接后继类，利用（1）式与其直接前驱类运算，可以求出边界城市 u_1^i，通过其直接后继类，可以求出第二个边界城市 $u_{w_i}^i$。

1.2.4 求解类内最短路经的蚁群算法

采用改进的蚁群算法来求解类内从第一个边界城市经所有剩下的城市一次且仅一次后到达第二个边界城市的最短路径。与求解 TSP 问题的蚁群算法相比，主要差别在最短路径的生成上。文中所有蚂蚁都从第一个边界城市出发，逐步选择除第二个边界城市之外的城市，从而生成问题的可行解。随机状态转移规则、信息素更新规则都与 Ant-cycle 模型相同。

1.2.5 局部搜索策略

实验结果显示，在搜索解的过程中，CPACA 算法能以极快的速度进入最优解所在的子空间，其后的搜索速度相对较慢，故在 CPACA 算法中引入了局部搜索策略 2-opt。

CPACA 算法的实现描述如下：

步骤 1：根据 TSP 问题中城市分布的先验知识，选用 C-均值或近邻函数法对 TSP 问题中的城市进行聚类处理。设可将城市聚成 $U_1,U_2,.....,U_c$ 类（其中 c 为类的个数），且 $K_i(i=1,...,c)$ 为每一类的中心，其中 K_i 可以不是 TSP 问题中的城市；

步骤 2：将 $K_i(i=1,...,c)$ 看成一个 TSP 问题，并使用蚁群算法对其求解，设求得的解为 $K_{i_1},K_{i_2},\cdots,K_{i_c},K_{i_1}$，此解即为合并各类中路径的连接顺序；

步骤 3：按公式（1）计算各类中的边界城市；

步骤 4：对每一类 $U_i(i=1,2,...,c)$ 中的城市，采用改进的蚁群算法并行求解从边界城市 u_1^i 到边界城市 $u_{w_i}^i$ 的一条经过所有剩下城市各一次的最短路径 $R_i(i=1,2,...,c)$；

步骤 5：对所有的类，按连接顺序 $i_1,i_2,\cdots,i_c,i_1$ 通过类与类之间的边界城市连接起来，便构成了 TSP 问题的一个可行解；

步骤 6：对解进行 2-opt 局部搜索，记录本次迭代的最好解；

步骤 7：如果本次迭代最好解优于当前最优解，则用其替换当前最优解；

步骤 8：如果结束条件满足，则退出算法，否则转到步骤 2。

CPACA 算法的时间复杂度分析如下：设 n 座城市的 TSP 问题可聚成 c 类（假设每类中城市数相等），则每类的城市数为 n/c。由文献[2]知，蚁群算法的时间复杂度为 $O(NC_{ACA}\cdot n^3)$，其中 NC_{ACA} 为 ACA 算法的迭代次数，则：

1) 如果CPACA算法中求解c个子问题的蚁群算法串行执行，其算法的时间复杂度为 $O\left(NC_{CPACA}\cdot\left(n/c\right)^3\cdot c\right)$，其中 NC_{CPACA} 是 CPACA 算法中求解每个子问题的蚁群算法发现优解的迭代次数，则算法效率提高了 $\frac{O(NC_{AS}*n^3)}{O\left(NC_{CPACA}*\left(n/c\right)^3*c\right)}=O\left(\frac{NC_{AS}}{NC_{CPACA}}*c^2\right)$ 倍。实验结果表明，待求解问题的规模越大，蚁群算法收敛到最优解的迭代次数越大，故 NC_{ACA} 大于 NC_{CPACA}，因此，算法的效率至少提高了 c^2 倍。

2) 如果CPACA算法中求解 c 个子问题的蚁群算法并行执行，则其时间复杂度为 $O\left(NC_{CPACA}\cdot\left(n/c\right)^3\right)$，算法效率提高了 $\frac{O(NC_{AS}\cdot n^3)}{O\left(NC_{CPACA}\cdot\left(n/c\right)^3\right)}=O\left(\frac{NC_{AS}}{NC_{CPACA}}\cdot c^3\right)$ 倍，因此，算法的效率至少提高了 c^3 倍。

显然，当问题可聚的类越多，CPACA 算法的效率越高，但如果问题的聚类特性不明显，则 CPACA 算法将退化为一般的蚁群算法，且 CPACA 算法中求解各子问题的蚁群算法之间不需要任何的通讯，故不存在蚂蚁级并行策略中的通讯瓶颈问题。

2 实验结果

实验选用 TSP 库（http://www.iwr.uni-heidelberg.de/ iwr /comopt/soft/TSPLIB95/TSPLIB.html）中的实例 pr107、pr136、d2103、u2319、pr2392，通过 ACS 算法与 CPACA 算法进行对比实验。针对不同的 TSP 实例，ACS 与 CACA 算法的参数设置如表 1、表 2。

采用 Visual C++ 6.0，在内存为 128 兆，奔腾 733 处理器的 PC 机上的进行模拟实验。对 pr107，pr136 问题两种算法各运行 15 次，d2103、u2319、pr2392 用两种算法各运行 4 次，得出对比实验结果如表 3 所示。

表 1 CPACA 算法针对不同问题实例的参数设置

参数 / 问题	α	β	ρ	Q	类数	蚂蚁数	CPACA 算法中蚁群算法的迭代次数
Pr107	1.0	3.0	0.1	10.0	2	30	300
Pr136	1.0	3.0	0.1	10.0	8	10	50
d2103	1.0	3.0	0.1	10.0	36	30	300
u2319	1.0	3.0	0.1	10.0	54	20	200
Pr2392	1.0	3.0	0.1	10.0	16	50	500

表 2 ACS 算法针对不同问题实例的参数设置

参数 / 问题	β	ρ	γ	τ_0
pr107	2.0	0.1	0.1	$(107*44303)^{-1}$
pr136	2.0	0.1	0.1	$(136*96772)^{-1}$
d2103	2.0	0.1	0.1	$(2103*8450)^{-1}$
u2319	2.0	0.1	0.1	$(2319*234256)^{-1}$
pr2392	2.0	0.1	0.1	$(2392*378032)^{-1}$

（ρ 为全局更新时信息素蒸发系数，γ 为局部更新时信息素蒸发系数）

图 3 是两种算法求解 pr136 问题时的对比时间关系，图 4 为 CPACA 算法求得的最优解。

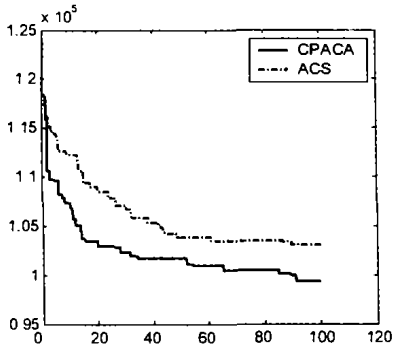

图 3 CPACA 算法与 ACS 算法收敛情况比较

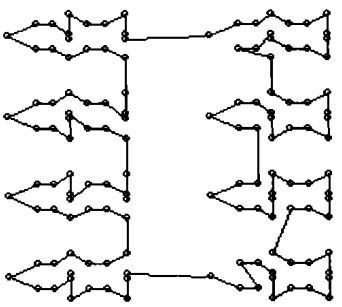

图 4 CPACA 算法求得 pr136 问题的解

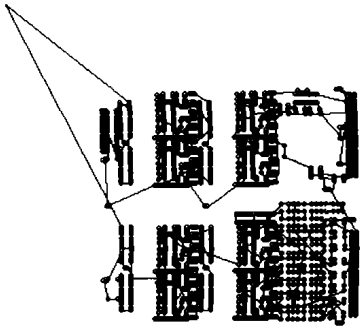

图 5 CPACA 算法求解 TSP 问题 d2103 的运行结果

表 3 CPACA 算法、ACS 算法求解 TSP 问题 pr107，pr136，d2103，u2319 和 pr2392 的对比结果

问题	运行时间/秒	ACS 算法		CPACA 算法			已知最优解
		所求的解	误差百分比 /%	所求的解	达到 ACS 的解所需时间	误差百分比 %	
Pr107	210	44661	0.81	44514	26	0.48	44303
Pr136	30	100213	3.56	97218	11	0.46	96772
d2103	26000	88306	9.77	81273	1025	1.02	80450
d2319	86000	262114	11.89	245341	50	4.73	234256
pr2392	365000	426218	12.75	392698	15000	3.88	378032

（表中的运行时间为 ACS 算法迭代 1000 次后所用的时间，CPACA 算法的解为相同时间内求得的解）

图 5 是 CPACA 算法求解 d2103 问题时运行 4 分钟后求得的问题的解，而 ACS 算法需要运行 3.5 小时才能达到与该值相近的解。

3 结论

本文提出的 CPACA 算法根据数据域所具有的聚类特征对问题进行分解，大大降低了问题的维数，从而减小问题求解时的复杂性。算法在运行初期就能以极高的速度收敛于问题解的某个子空间，从而引导算法朝正确的方向（最优解所在的子空间）进行搜索。在每次迭代时，局部搜索策略使其能够更好地发现较好的解。当问题的聚类数越多，且每类中的城市数较接近时，CPACA 算法的性能越好；但当问题的聚类特征不明显时，CPACA 算法会退化成一般的蚁群算法，故其实用性有一定的限制。

参考文献：

[1] M Dorigo, V Maniezzo, A Colorni. Positive feedback as a search strategy [R]. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.

[2] M Dorigo, V Maniezzo, A Colorni. The ant system: Optimization by a colony of cooperating agents [J]. IEEE Transactions on Systems, Man, and Cybernetics Part B, 1996, 26(1): 29-41.

[3] M Dorigo, L M Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53–66.

[4] T Stützle. H Hoos. The MAX–MIN ant system and local search for the traveling salesman problem [C]. In T. Baeck, Z. Michalewicz, and X. Yao, editors, Proceedings of IEEE-ICEC-EPS'97, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference, pages 309–314. IEEE Press, 1997.

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

[6] 孙即祥. 现代模式识别[M]. 长沙：国防科技大学出版社, 2002.

（上接第 2682 页）

动。从过程输出与混合模型输出曲线的直观比较可以发现两者能够很好的吻合，整个运行期间采样点数据的 MSE（均方差）为 1.304e-4，其中阶段 1 的 MSE 为 2.194e-6，阶段 2 的 MSE 为 1.273e-4，阶段 3 的 MSE 为 7.675e-4，此时出现最大偏差，为输出值的 2.72%。模型在稳态阶段具有很高的精度，当有外界扰动出现时，动态调整机构发挥作用，使模型仍保持较高的精度，而当存在叠加的扰动时，模型的误差增大。从运算速度上看，当误差限设定为 0.005 时，模型在阶段 2 调整参数的时间为 45 周期，阶段 3 的调整时间约为 70 周期，而模型算法每周期大约耗时 80ms，因此实时性很好。

5 结论

具有多神经网络结构的混合模型可以描述复杂过程的非线性动态行为，对模型参数的调整算法提出了更高的要求。采用结合了混沌算法的全局搜索优势与 BFS 法的局部快速寻优特性的混合递推算法后，模型可以在线实时进行参数递推辨识，具有很强的适应性，为进一步的基于模型的复杂过程系统控制与优化打下良好基础。

参考文献：

[1] 卢荣德, 陈宗海, 王雷. 复杂工业过程计算机建模、仿真与控制的综述 [J], 系统工程与电子技术, 2002, 24(1): 27-30.

[2] 王雷, 陈宗海. 神经网络在过程系统建模中的应用综述[A]. 系统仿真技术及其应用会议论文集 [C], 2002, 4: 24-29.

[3] A A Safavi, A Nooraii, J A Romagnoli. A hybrid model formulation for a distillation column and the online optimization study [J]. Journal of Process Control, 1999, 9: 125-134.

[4] Cabassud M, *et al.* Neural networks: a tool to improve UF plant productivity [J]. Desalination, 2002, 145(1-3): 223-231.

[5] Wang Lei, Zhang Haitao, Chen Zonghai. Hybrid RBF Neural Network Based Prefractionator Modeling and Control [A]. Proceedings of ICCA'02 [C], 2002: 463-467.

[6] Pascal F, *et al.* A structural modeling approach for dynamic hybric fuzzy-first principles models [J]. Journal of process control. 2002, 12: 605-615.

[7] Michael L, Thompson, Mark A Kramer. Modeling Chemical Processes Using Prior Knowledge and Neural Networks [J]. AIChE Journal, 1994, 40(8): 1328-1340.

[8] Patnaik, P.R. An integrated hybrid neural system for noise filtering, simulation and control of a fed-batch recombinant fermentation [J]. Biochemical Engineering Journal, 2003, 15(3): 165-175.

[9] 王东生, 曹磊. 混沌、分形及其应用 [M]. 合肥：中国科学技术大学出版社, 1995.

[10] 孙德敏. 工程最优化方法及应用 [M]. 合肥：中国科学技术大学出版社, 1991.

[11] 陈宗海. 过程系统建模与仿真 [M]. 合肥：中国科学技术大学出版社, 1997.

[12] Boozarjomehry R B, Svrcek W Y. Automatic design of neural network structures [J]. Computers and chemical engineering, 2001, 25: 1075-1088.