第 35 卷 第 21 期 计 算 机 工 程 2009 年 11 月
Vol.35 No.21 Computer Engineering November 2009
·人工智能及识别技术· 文章编号：1000—3428(2009)21—0195—02 文献标识码：A 中图分类号：TP301.6

求解度约束最小生成树的改进 ACS 算法

王志杰，全惠云
(湖南师范大学数学与计算机科学学院，长沙 410081)

摘 要：针对蚂蚁系统算法求解度约束最小生成树时收敛速度慢和早熟问题，提出一种改进的蚁群系统算法 UDA-ACS。该算法在保留蚁群系统算法优点的基础上，通过增大能见度的影响力、采用动态负反馈机制和赋予不同初始信息素的方法解决上述问题。理论分析和实验结果证明，该算法的求解质量和速度比蚂蚁系统算法更优越。

关键词：蚂蚁系统算法；度约束最小生成树；蚁群系统算法

Improved ACS Algorithm for Solving Degree-Constrained Minimum Spanning Tree

WANG Zhi-jie, QUAN Hui-yun
(School of Mathematics and Computer Science, Hunan Normal University, Changsha 410081)

【Abstract】 Aiming at the problems of slow converge-rate and precocity when using Ant System(AS) algorithm to solve Degree-Constrained Minimum Spanning Tree(DCMST) problem, this paper presents an improved algorithm named UDA-ACS(Unequal initial pheromone strategy, Dynamic negative feedback mechanism, Adequately augment the effect of visibility-Ant Colony System). It retains the advantages of Ant Colony System(ACS) algorithm, and adopts the method of augmenting the effect of visibility, dynamic negative feedback mechanism and giving with unequal initial pheromone. Theory analysis and experimental result prove that it gains better solving quality and higher speed than AS algorithm.

【Key words】 Ant System(AS) algorithm; Degree-Constrained Minimum Spanning Tree(DCMST); Ant Colony System(ACS) algorithm

1 概述

度约束最小生成树(Degree-Constrained Minimum Spanning Tree, DCMST)[1]是一个 NP 难问题，该问题在典型的组合优化问题求最小生成树的基础上加入了一组限制条件来限定每个顶点度的值。

对 DCMST 问题求解的算法很多，包括分支定界算法、遗传算法等，但这些算法在度限制极其苛刻且网络呈稀疏的状态下可行解的数目大多很少甚至找不到可行解。由于蚂蚁系统(Ant System, AS)[2]算法本身所具有的正反馈性、协同性和隐并行性能够克服以上弊病，因此文献[3]提出了一种求解该问题的 AS 算法，表现出较好的性能。虽然该算法与其他算法相比有一定的优越性，但蚁群算法本身是不断发展的，如蚁群系统(Ant Colony System, ACS)[4]算法就是对 AS 算法的一种改进，它属于全局启发式算法。本文提出了改进的 ACS 算法——UDA-ACS(Unequal initial pheromone strategy, Dynamic negative feedback mechanism, Adequately augment the effect of visibility-Ant Colony System)，并将其运用于 DCMST 问题的求解中。

2 基于 AS 算法的 DCMST 问题求解

算法中的一些变量和常量说明如下：m 为蚂蚁的数量；$\Delta\tau^k(ij)$ 表示蚂蚁 k 在相邻时刻中在边 (i,j) 上留下的单位长度轨迹信息素量；α 为轨迹的相对重要程度；β 为启发信息的相对重要程度；ρ 为轨迹的持久性 $(0\le\rho\le1)$；$1-\rho$ 为衰减度；Q 为蚂蚁所留轨迹数量的一个常数；Z_{ij} 为目标函数值；η_{ij} 为边 (i,j) 的能见度，且 $\eta_{ij}=\frac{1}{w_{ij}}$；$\tau_{ij}(t)$ 为 t 时刻边 (i,j) 上的信息素数量；$p_{ij}^k(t)$ 为 t 时刻蚂蚁 k 转移概率，即

$$p_{ij}^k(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^\alpha[\eta_{ij}(t)]^\beta}{\sum_{s\in S}[\tau_s(t)]^\alpha[\eta_{ss}(t)]^\beta} & \text{if } j\in S\\ 0 & \text{otherwise}\end{cases} \tag{1}$$

算法步骤如下：

步骤 1 初始化变量，并将 m 只蚂蚁置于 n 个节点上。

步骤 2 将每只蚂蚁的初始出发点置于当前解集中；对每只蚂蚁按概率选择符合度限制的顶点，若存在，置于当前解集中，更新当前度限制。重复执行该步直到所有蚂蚁解构造完成。

步骤 3 计算各蚂蚁的目标函数值，记录当前最好解，对各蚂蚁及其弧，计算 $\Delta\tau_{ij}$，再对各弧计算 $\tau_{ij}(t+n)$。

步骤 4 对各只蚂蚁及其弧 (i,j)，计算 $\Delta\tau_{ij}\leftarrow\Delta\tau_{ij}+\Delta\tau^k(ij)$。

步骤 5 对各弧 (i,j)，计算 $\tau_{ij}(t+n)=\rho\tau_{ij}(t)+\Delta\tau_{ij}$。

步骤 6 置 $t=t+n$，对各弧置 $\Delta\tau_{ij}=0$，$nc\leftarrow nc+1$。

步骤 7 若 nc 大于最大循环次数 $NcMax$ 或蚁群全部收敛到一条路径，则输出最好解；否则，转步骤 2。

3 ACS 算法思想

蚁群系统算法是对蚂蚁系统算法改进之后的产物，其总体上与蚂蚁系统算法思路一致，但 ACS 算法在控制蚂蚁对下

作者简介：王志杰(1980－)，男，硕士研究生，主研方向：智能算法；全惠云，教授

收稿日期：2009-02-26 **E-mail：**zjwang1980@163.com

一节点的选择和信息素更新机制方面引入了一些新的思想和公式，其基本策略如下：

(1)如果存在一个节点 j 属于候选列表，则选择下一节点 j，$j \in J_i^k$，且

$$j=\begin{cases}\arg\max\limits_{j\in allowed_k}\left\{[\tau_{ij}]^{\alpha}\cdot[\eta_{ij}]^{\beta}\right\} & \text{if } q\le q_0\\ 0 & \text{otherwise}\end{cases} \tag{2}$$

其中，$p_{ij}^k(t)$ 采用式(1)计算，i 为当前节点；否则选择最近的节点 j，$j \in J_i^k$。

(2)每次遍历之后每只蚂蚁 k 应用局部更新规则：

$$\tau_{ij}(t)\leftarrow\rho\cdot\tau_{ij}(t)+(1-\rho)\cdot\tau_{ij}(0) \tag{3}$$

(3)如果找到更好的 Z_k，则更新 Z_k^+ 和 T^+。

(4)对每条边 $(i,j)\in T^+$，根据规则更新信息素轨迹：

$$\tau_{ij}(t)\leftarrow\rho\cdot\tau_{ij}(t)+(1-\rho)\cdot\Delta\tau_{ij}(t) \tag{4}$$

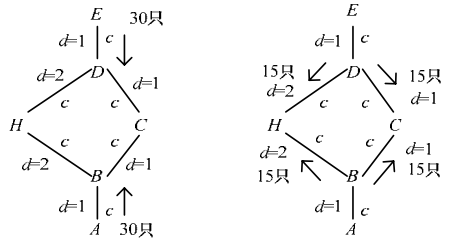
其中，$\Delta\tau_{ij}(t)=\dfrac{1}{Z_k^+}$。

4 UDA-ACS 算法

4.1 对 ACS 算法的改进

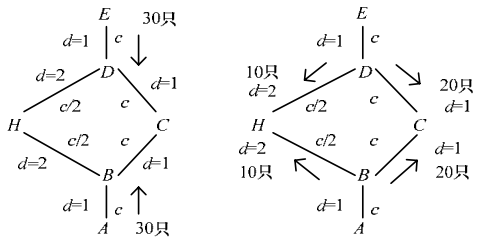
针对 ACS 算法“求解全局最优解速度慢”和“早熟”现象，在结合其优点的同时进行以下改进：

(1)ACS 算法在信息素初始化时主要采用为每条边赋予相同的信息素初始值常量 c，如图 1(a)所示，忽略了解的分布特征，这样容易导致前期收敛速度较慢。本文算法为较短的边赋予稍大的初始信息素、为较长的边赋予稍小的初始信息素来加速前期收敛速度，如图 2(a)所示。通过对比图 1(b)和图 2(b)可以得出，改进的算法由于起初就赋予不同的初始信息素，因此蚂蚁在前期进行路径选择时信息素的指导作用能更好地发挥，从而降低了算法前期蚂蚁盲目随机选择的可能性，改进后的策略显然更有利于蚂蚁在算法前期尽快找到较优解，提高了算法的收敛速度。



(a)蚂蚁到达分叉点之前　　(b)蚂蚁到达分叉点之后

图 1　ACS 算法信息初始化方法



(a)蚂蚁到达分叉点之前　　(b)蚂蚁到达分叉点之后

图 2　UDA-ACS 算法信息初始化方法

初始信息素具体设置方法如下：

$$\tau_{ij}(t)=\begin{cases}c & \text{if } w_{ij}=\min\ w_{ij}\\ \left(\dfrac{\min\ w_{ij}}{w_{ij}}\right)\cdot c & \text{otherwise}\end{cases} \tag{5}$$

(2)ACS 算法在求解后期各蚂蚁所寻优的路径已经基本相同，此时想进一步找到更好的解较困难，若找到的解不是真正的最优解，往往容易陷入局部最优，为了尽可能避免这一缺陷、提高解的质量，本文算法在运行中期，即当所有蚂蚁中有 r 只蚂蚁所走路径相同时就引入动态的负反馈机制：

$$\tau_{ij}(t)\leftarrow\rho\cdot\tau_{ij}(t)+(1-\rho)\cdot\Delta\tau_{ij}(t)\cdot\varepsilon \tag{6}$$

其中，$\Delta\tau_{ij}(t)=\dfrac{1}{Z_k^+}$；$\varepsilon=\dfrac{(m-r)}{m}$，$\dfrac{m}{3}\le r\le\dfrac{2m}{3}$。

(3)ACS 算法在求解后期，各路径上的信息素已经基本趋于稳定，此时信息素对蚂蚁选择下一节点的指导作用过强，易忽视对更优节点的选择，影响求解质量。为了弥补这一缺陷，本文算法在运行后期，即当所有蚂蚁中有 h 只蚂蚁所走路径相同时，适当增大能见度对蚂蚁路径选择的影响力：

$$\eta_{ij}=\frac{D}{w_{ij}} \tag{7}$$

其中，$D=6(h-\dfrac{m}{2})$，$\dfrac{2m}{3}<r<m$。

4.2 UDA-ACS 算法

UDA-ACS 算法的具体步骤如下：

步骤 1 初始化时间变量 t、循环次数 nc、信息素变化值 $\Delta\tau_{ij}$；根据式(5)设置初始信息素值 $\tau_{ij}(t)$，设置最大循环次数 $NcMax$、概率选择控制系数 q_0；为负反馈机制开关系数 r、能见度增大开关系数 h、最优解的路径 T^+和目标函数值 Z^+赋初值，并将所有蚂蚁随机放到各节点上，为每个节点生成候选列表。

步骤 2 将每只蚂蚁的初始出发点置于当前解集中，判断是否还存在节点属于候选列表，若存在于候选列表，判断能见度开关系数 h，按式(2)选择符合度限制的下一节点，否则，选择不在候选列表中权值最小且符合度限制的节点作为下一节点 j，将 j 置于当前解集中，更新当前度限制，按式(3)更新局部信息素。重复这一步直到所有蚂蚁构造解完成。

步骤 3 计算各蚂蚁的目标函数值 Z_k 以及所有蚂蚁中趋于同一路径的个数，并将值赋给 h 和 r，记录当前最好解并与 Z^+进行比较，若更优则更新 Z^+和 T^+。

步骤 4 对 T^+中各弧 (i,j)，判断负反馈开关系数 r 是否满足式(6)，若满足，则按式(6)更新，否则，按式(4)更新。

步骤 5 置 $t=t+n$，对各弧 (i,j)，置 $\tau_{ij}(t+n)=\tau_{ij}(t)$，$nc\leftarrow nc+1$。

步骤 6 若 nc 大于预定迭代次数或蚁群全部收敛到一条路径，则输出最好解；否则，转步骤 2。

5 实验结果与分析

实验的运行环境如下：CPU 为赛扬 766 MHz，内存为 512 MB，操作系统为 Windows XP SP2，开发工具为 Microsoft Visual C++ 6.0，编程语言为 C++。并采用文献[3]的例子检验本文算法求解 DCMST 问题的优越性。

例：n=8，度限制为{1, 3, 3, 1, 1, 3, 1, 3}，权矩阵为

$$\boldsymbol{w}=\begin{bmatrix}\infty & 12 & 35 & 41 & 47 & 10 & 13 & 89\\ 12 & \infty & 78 & 93 & 43 & 54 & 91 & 87\\ 35 & 78 & \infty & 59 & 10 & 44 & 72 & 5\\ 41 & 93 & 59 & \infty & 60 & 39 & 95 & 64\\ 47 & 43 & 10 & 60 & \infty & 26 & 3 & 55\\ 10 & 54 & 44 & 39 & 26 & \infty & 51 & 57\\ 13 & 91 & 72 & 95 & 3 & 51 & \infty & 5\\ 89 & 87 & 5 & 64 & 55 & 67 & 5 & \infty\end{bmatrix}$$

（下转第 199 页）