# 一种求解最大团问题的蚁群算法

曾 艳
(西安邮电学院 计算机学院, 陕西 西安 710121)

摘要: 将最大团问题看作子集类问题, 提出了基于子集类问题的特殊蚁群算法用于求解最大团问题。该算法将信息素和局部启发信息与图的顶点相关联, 而不再与边相关联, 从而提高算法的运行速度。仿真实验研究表明, 该算法较传统求解最大团问题的蚁群算法有着更短的运行时间, 较强的求解能力, 更适合用于求解最大团问题。

关键词: 蚁群算法; 最大团问题; 子集类问题
中图分类号: TP301.6 文献标识码: A 文章编号: 1007-3264(2010)03-0089-03

## 0 引言

最大团问题(Maximum Clique Problem, MCP)是一个著名的组合优化问题, 这类问题的求解一直困扰着人们。近些年, 人们用神经计算、遗传算法、分子生物计算等方法来求解 NP-完全问题也取得了一些进展。

"蚁群优化"(Ant Colony Optimization algorithm, ACO)是意大利学者 M. Dorigo 等人[1]在真实蚂蚁觅食行为的启发下提出的一种基于种群的元启发式搜索算法。最早用于解决 TSP 问题[2]。这类问题可以被看作排序类问题(ordering problems), 其可行解为所有顶点组成的一个序列, 这个序列表达了"顺序和路径"的概念。而最大团问题和 TSP 问题不同, 它是子集类问题(subset problem), 在子集问题中, 所考虑问题的解是顶点集的一个满足特定条件的子集。到目前为止只有很少人尝试使用 ACO 来解决子集问题。本文提出的算法使用了一种针对子集类问题的 ACO 思想[3]求解最大团问题, 取得了较为满意的试验结果。

## 1 问题描述

最大团问题(MCP): 给定一个无向图 $G = (V, E)$, 其中 $V = \{1, \cdots, n\}$ 是图 $G$ 的顶点集, $E \subseteq V \times V$ 是边的集合。一个团是顶点集 $V$ 的一个子集, 记为 $C$, $C \subseteq V$, 要求 $C$ 中任意两个顶点之间有边相连。也就是说由团 $C$ 中顶点及其边组成的子图是完全图。最大团问题就是要找到具有最多顶点数的子集 $C$。

## 2 最大团问题的蚁群算法

最大团问题可以被看作是求顶点集 $V$ 的一个满足特定条件的子集 $C$, 它的可行解是一个集合, 其中元素是没有顺序概念的。显然, 最大团问题就是一个典型的子集问题。

在子集问题中并不特别注意顶点的顺序。因而, 在构建过程中, 选择下一个要添加进解中的顶点时, 并不需要考虑最近刚加入的顶点。因此在子集问题中, 信息素通常与顶点相关联, 而不是与边相关联。

2003 年, Fenet 和 Solnon 提出了求解最大团问题的蚁群算法 AntClique[5], 该算法仍然将信息素留在边上, 信息素 $\tau_{ij}$ 是指把结点 $i$ 和结点 $j$ 分配到同一个团中的期望。没有使用局部启发信息, 这使得迭代初期各候选顶点的选择概率几乎相等, 这样算法在迭代初期有一定的盲目性, 往往需要更多的迭代次数才能得到最优解。针对这些不足及最大团问题的特点, 本文提出的算法 AntMCP 做了如下改进: 第一, 增加了局部启发信息 $\eta$; 第二, 信息素 $\tau$ 和局部

收稿日期: 2009-11-25
作者简介: 曾 艳(1975-), 女, 湖北钟祥人, 西安邮电学院计算机学院助教, 硕士。

启发信息 η 不是留在边上, 而是留在顶点上。这样, 变量 τ 和 η 由二维降为一维, 既可节省存储空间, 又可提高运行速度, 大量实验表明, 该算法运算速度更快, 效率更高。

2.1 概率选择公式

$$p(v_i) = \frac{\tau_{\alpha i}\eta_i^{\beta}}{\sum_{j \in Candidate} \tau_j^{\alpha}\eta_j^{\beta}} \quad (1)$$

其中, $\tau_i$ 为顶点 $v_i$ 上的信息素的强度。

$$\eta_i = \frac{Q \cdot \deg(v_i)}{\underset{j=1}{\overset{n}{\mathrm{Max}}}(\deg(v_j))} \quad (2)$$

$\eta_i$ 为顶点 $v_i$ 上的局部启发信息。其中 $\deg(v_i)$ 为顶点 $v_i$ 的度; $Q$ 为常量, 可知 $\eta_i \in (0, Q]$; (本文中实验数据是在 $Q$ 取为 100 时获得的)。

α 为信息素权重; β 为局部启发信息权重; 这两个量均为常量。

Candidate 为候选解的集合, 其确定方法如下:

(a) 随机选择一个顶点 $v_f$ 加入 $C$ 后, $Candidate \leftarrow \{v_i \mid (v_f, v_i) \in E\}$。

(b) 当从 $Candidate$ 集合中选中 $v_i$ 加入 $C$ 后, $Candidate \leftarrow Candidate \cap \{v_j \mid (v_i, v_j) \in E\}$。

2.2 信息素策略

给每个顶点 $v_i$ 设置一个信息素变量 $\tau_i$。本文采用的是 MAX-MIN 蚁群系统[4] 中的方法, 信息素的取值范围固定在 $\tau_{max}$, $\tau_{min}$ 之间。$\tau_i$ 的初始值设为 $\tau_{max}$。

更新信息素是一次循环结束之后, 根据本次循环所有蚂蚁构造的团更新各个顶点上的信息素。信息素的更新分为两步:

(a) 信息素挥发, 在自然环境中, 蚂蚁留在路上的信息素会随时间的推移而挥发, 这个步骤就是模拟挥发现象。

$$\tau_i(t+1) = \rho\tau_i(t) \quad \rho \in (0, 1] \quad (3)$$

其中, ρ 为挥发系数, 为常数, 可根据经验给定, 一般在(0.9, 1) 之间。

(b) 增加信息素, 当蚂蚁走过某条路径时, 就会在该路径上留下信息素, 该路径上的信息素量就会增加。由于在最大团问题中没有路径的概念, 算法需要满足: 子集 $C$ 越优秀, 其中顶点的被选择概率越高。所以, 对蚂蚁找到的最大团 $C_{best}$, $C_{better}$ 中的顶点的信息素进行增加, 增加量为:

$$\Delta\tau_i = \frac{1}{1 + |C_{best}| - |C_{better}|} \quad i \in C_{best} \cup C_{better} \quad (4)$$

$|C_{best}|$ 为目前为止找到的最优解; $|C_{better}|$ 为本次循环中所有蚂蚁找到的最优解。

2.3 算法描述

```
procedure Vertex_AntClique
    Initialize //初始化信息素和局部启发信息
    Repeat
        For k in 1..nbAnts do:
            Choose randomly a first vertex v_f ∈ V
            C_k ← {v_f}
            Candidate ← {v_i | (v_f, v_i) ∈ E}
            While Candidate ≠0 do
                Choose a vertex v_i ∈ Candidate with probability p(v_i);
                C_k ← C_k ∪ {v_i}
                Candidate ← Candidate ∩ {v_j | (v_i, v_j) ∈ E}
            End of while
        End of for
        Update pheromone w.r.t. {C1, ..., C_nbAnts}
        C_nbAnts}
    Until max cycles reached or optimal solution found
End of procedure
```

其中 $p(v_i)$ 由公式(1) 确定。

## 3 实验结果

下面用 DIMACS benchmarks 中的部分实例来测试本文中提出的 AntMCP 算法。

表 1 AntMCP 和 AntClique 的求解性能比较

| 例 图 | 已知最优解 | AntMCP | | Ant Clique |
|---|---|---|---|---|
| | | a= 2 | a= 3 | |
| C500.9 | ≥57 | 57(55.00) | 56(54.64) | 56 |
| C1000.9 | ≥68 | 67(65.48) | 67(64.00) | 67 |
| C2000.9 | ≥78 | 77(74.48) | 75(72.80) | 73 |
| C2000.5 | ≥16 | 16(15.00) | 16(15.04) | 16 |
| C4000.5 | ≥18 | 16(15.88) | 16(15.92) | 17 |
| MANN-a27 | 126 | 126(125.48) | 126(125.40) | 126 |
| MANN_a45 | 345 | 344(342.60) | 343(342.32) | 341 |
| MANN_a81 | ≥1098 | 1095(1093.26) | 1098(1097.38) | 1092 |
| gen200_p0.9_44 | 44 | 44(42.00) | 44(40.64) | 44 |
| gen200_p0.9_55 | 55 | 55(55.00) | 55(54.44) | 55 |
| gen400_p0.9_55 | 55 | 55(51.56) | 53(50.92) | 52 |
| gen400_p0.9_75 | 75 | 75(75.00) | 75(73.28) | 75 |
| Hamming8-4 | 16 | 16(16.00) | 16(16.00) | 16 |
| hamming10-4 | 40 | 35(33.96) | 40(38.56) | 40 |
| Keller4 | 11 | 11(11.00) | 11(11.00) | 11 |
| Keller5 | 27 | 27(27.00) | 27(27.00) | 27 |
| Keller6 | ≥59 | 51(48.27) | 59(55.48) | 55 |

注: 表中的数据格中, 括号前的数据为运行 25 次中的最优结果, 括号内数

据为运行 25 次的平均结果。

表 1 中列出的数据是在蚂蚁数量为 30 只, $\beta = 2$, $\rho = 0.95$, $\tau_{max} = 4$, $\tau_{min} = 0.01$, 最大循环次数为 3000, 运行 25 次。和 AntClique[5] 中的运行次数基本一致, 以方便比较。从表 1 中的实验结果可以看出, 只要参数选择的比较合适, 算法能较好的解决最大团问题。特别地, 对于图 C500.9, $\alpha = 2$, $\beta = 2$, 能找到最大团 57; 对于图 Keller6, $\alpha = 3$, $\beta = 2$, 找到的最优解为 59; 对于复杂图 MANN_a81, 能找到 1098, 均优于 AntClique 算法[5] 的 56、55、1092。

表 2 是两种算法的运算速度的比较, 其运算规模均为: 7 只蚂蚁 × 3000 次迭代; 硬件环境为: Pentium 4 CPU 2.66GHz, 512MB 内存; 软件环境为: 操作系统 windows XP, VC++ 6.0 编程实现。

表 2　AntMCP 和 AntClique 的运行时间比较

| 例 图 | \|V\| 顶点数 | \|E\| 边数 | AntMCP 运行时间(秒) | AntClique 运行时间(秒) |
|---|---|---|---|---|
| C500.9 | 500 | 112332 | 13 | 68 |
| C1000.9 | 1000 | 450079 | 28 | 196 |
| C2000.9 | 2000 | 1799532 | 57 | 630 |
| C2000.5 | 2000 | 999836 | 19 | 458 |
| MANN-a27 | 378 | 70551 | 36 | 149 |
| MANN_a45 | 1035 | 533115 | 269 | 1151 |
| gen200_p0.9_44 | 200 | 17910 | 5 | 18 |
| gen200_p0.9_55 | 200 | 17910 | 6 | 20 |
| gen400_p0.9_55 | 400 | 71820 | 10 | 50 |
| gen400_p0.9_75 | 400 | 71820 | 13 | 55 |
| Hamming8-4 | 256 | 20864 | 3 | 11 |
| hamming10-4 | 1024 | 434176 | 17 | 164 |
| Keller4 | 171 | 9435 | 2 | 5 |
| Keller5 | 776 | 225990 | 10 | 87 |

注: 表中最后一列用于比较的数据为 AntClique[5] 在各种参数下的最好结果。

由表 2 中的数据可以看出, 由于 AntMCP 将信息素留在顶点上, 使得算法 AntMCP 的运算速度较最大团问题的传统蚁群算法 AntClique 有着显著的提高。

## 4　结论

仿真实验数据研究表明, 该算法确实能提高算法的求解速度和能力。将信息素留在顶点上, 而非边上, 信息素变量 $\tau$ 从二维 $\tau_{ij}$ 降为一维 $\tau_i$, 节省了存储空间, 更重要的是提高了运行速度。增加了局部启发信息, 消除了迭代初期的盲目性, 使得在同等运算规模下, 提高了算法的求解能力。

参　考　文　献

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

[2] Dorigo M, Gambardella LM. Ant colonies for the traveling salesman problem[J]. BioSystems, 1997, 43(2): 73-81.

[3] Guillermo Leguizamon, Zbigniew Michalewicz. A New Version of Ant System for Subset Problems[C]. Proceedings of the congress on Evdutionary Computation, 1999.

[4] Stützle T, Hoos H. The MAX-MIN ant system and local search for the traveling salesman problem [C]. IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference. IEEE Press, 1997.

[5] S. Fenet and C. Solnon Searching for Maximum Cliques with Ant Colony Optimization[C]. Applications of Evolutionary Computing, LNCS 2611, Springer 2003.

# Ant colony algorithm for maximum clique problem

ZENG Yan

(School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, China)

**Abstract:** In this paper, a new version of ant colony algorithm for subset problem is proposed for solving the maximum clique problem. The pheromone trail and heuristic information of the new version ACO should be stored on each vertex instead of edge. The simulation result shows that the proposed ant colony optimization is effective and efficient for solving maximum clique problem.

**Key words:** ant colony optimization algorithm; maximum clique problem; subset problem