文章编号: 1007-130X(2008)04-0043-03

蚁群算法解决指派问题的研究和应用*

Research and Application of the Ant Colony Algorithm in the Assignment Problem

殷人昆[1], 吴 阳[2,3], 张晶炜[3,4]
YIN Ren-kun[1], WU Yang[2,3], ZHANG Jing-wei[3,4]
(1 清华大学计算机系, 北京 100084; 2. 中科院研究生院, 北京 100049;
3. 海军后勤技术装备研究所, 北京 100072; 4. 海军航空工程学院信息融合技术研究所, 山东 烟台 264001)
(1. Department of Computer Science and Technology, Tsinghua University, Beijing 100084; 2. Graduate School, Chinese Academy of Sciences, Beijing 100049; 3. Institute of Naval Logistic Technology and Equipment, Beijing 100072; 4. Institute of Information Fusion, Naval Aeronautical Engineering Institute, Yantai 264001, China)

摘 要: 指派问题是在生产和生活中经常出现的问题。本文建立了指派问题的数学模型, 对现有的解决指派问题的蚁群算法进行了分析, 并设计了一种改进的解决指派问题的蚁群算法, 有效地提高了蚁群算法解决指派问题的准确性和效率, 并通过实验结果验证了应用蚁群算法解决指派问题的可行性和先进性。

Abstract: The assignment problem is a very important one that frequently appears in mass production and people's daily life. The paper constructs the model of the assignment problem, and analyzes the existing ant colony algorithm applied in the problem. It designs a modified application of the ant colony algorithm in solving the assignment problem, which improves the accuracy and efficiency of the algorithm effectively. It also gives a brief analysis of the feasibility and advantages of using the ant colony algorithm in solving the assignment problem according to the results of experiments.

关键词: 蚁群算法; 指派问题; 组合优化; 匈牙利法
Key words: ant colony algorithm; assignment problem; combinatorial optimization; Hungarian approach
中图分类号: TP301.6 文献标识码: A

1 引言

指派问题是组合优化问题中的一个分支, 它在生产和生活中经常出现。传统的解决指派问题的方法是使用匈牙利法。然而, 匈牙利法在处理一些特殊数据时算法不能收敛, 无法找出最优解[1]。此外, 该方法适合于在问题规模不大的情况下进行手工计算, 在计算机上的编程实现比较困难[2]。

蚁群算法是近年来流行起来的一种针对难解的离散优化问题的元启发式算法。蚁群算法可以应用到任何组合优化问题中[3], 如旅行商问题(TSP)、顺序排列问题(SOP)、多重背包问题(MKP)、网络路由问题(NRP)等都有成功的应用。

对于各种组合优化问题, 真正的任务是如何把某个问题描述成可以被人工蚂蚁用来构建的表达方式[3]。本文根据蚁群算法的原理重新构建了解决指派问题的蚂蚁路径图, 减少了蚂蚁每次寻找路径的计算次数, 提高了蚂蚁路径搜索的效率, 从而在整体上提高了蚁群算法解决指派问题的性能, 并通过实验结果对应用蚁群算法解决指派问题的可行性和优势进行了分析。

2 指派问题

指派问题是现实生活中常会遇到的一类问题, 有着十分广泛的应用。例如, 在生产过程中, 有若干项生产任务和若干个生产单位, 如何将每个任务分配给适当的生产单位, 使得生产成本最小, 就是一类指派问题的特例。

指派问题的标准形式(以任务和单位为例)如下: 设有 n 个单位和 n 项任务, 已知第 i 个单位完成第 j 项任务的费

* 收稿日期: 2007-09-01; 修订日期: 2007-10-23
基金项目: 国家自然科学基金资助项目(60672139, 60672140)
作者简介: 殷人昆(1975-), 男, 北京人, 硕士, 工程师, 研究方向为算法。
通讯地址: 100084 北京市清华大学计算机系; Tel: 13501375055; E-mail: doggod@126.com
Address: Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China

用为 $C_{ij}(i,j=1,2,\cdots,n)$，要求一个单位和任务之间一一对应的指派方案，使完成这些任务的总成本最小。

称矩阵：

$$\boldsymbol{C}=(c_{ij})_{n\times n}=\begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}$$

为该问题的成本矩阵，第 i 行各元素表示第 i 个单位完成各任务所需成本，第 j 列各元素表示第 j 项任务由各单位完成所需成本。

这一问题的数学模型如下：

引入 n^2 个 0-1 变量 x_{ij}，令：

$$x_{ij}=\begin{cases}1, \text{指派第 } i \text{ 个单位完成第 } j \text{ 项任务}\\ 0, \text{不指派第 } i \text{ 个单位完成第 } j \text{ 项任务}\end{cases} i,j=1,2,\cdots,n \tag{1}$$

这样，问题的数学模型是：

$$\min Z=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij},$$

$$s.t.\begin{cases}\sum_{i=1}^{n}x_{ij}=1, j=1,2,\cdots,n\\ \sum_{j=1}^{n}x_{ij}=1, i=1,2,\cdots,n\\ x_{ij}\in\{0,1\}, i,j=1,2,\cdots,n\end{cases} \tag{2}$$

3 蚁群算法解决指派问题

3.1 蚁群算法概述

蚁群算法(Ant Colony System)是一种元启发式算法，是由意大利学者 Dorigo M 等人于 1991 年首先提出来的。它模拟自然界中真实蚂蚁的觅食行为，采用具有记忆的人工蚂蚁，通过个体之间的信息交流与相互协助来寻找蚁穴到食物源的最短路径。

蚁群算法的基本原理为：蚂蚁之间通过信息素(Pheromone)进行间接交流而达到合作。蚂蚁在行进途中会根据信息素的浓度选择路径，同时也释放自己的信息素。这就是说，路径上的信息素浓度越大，后来的蚂蚁选择该路径的概率也越大，经过一段时间蚂蚁经常选择的路上的信息素浓度会越来越大；另一方面，信息素也会随着时间的推移而挥发，而蚂蚁很少选择的路径上的信息素浓度会越来越低。因此，当大量蚂蚁觅食时就会表现出一种信息的正反馈现象，指导蚂蚁最终找到一条从蚁穴到食物源的最短路径。

3.2 现有解决指派问题的蚁群算法

文献[2]描述了一种利用蚁群算法解决指派问题的方法，这种算法基本按照文献[3]中的描述，将所有任务和单位都作为图中的节点。其中，任务节点的序号 $i\in[1,n]$，单位节点的序号 $j\in[n+1,2n]$，任务与任务或单位与单位之间的距离被设为无限大(在实际编程中取最大整数)，任务到单位的距离被设为 0，单位到任务的距离 $d_{i[j]}$ 设为成本系数。

这种蚂蚁搜索图的设计存在三点不足：

(1) 根据实际运行经验，蚂蚁在搜索路径时很有可能通过距离最大值搜索至相同的节点，虽然概率比较低，但这种情况时有发生。一旦出现了这种情况，该蚂蚁所搜索的路径就不再有意义，从而浪费了算法的资源，影响了算法的效率。

(2) 蚂蚁在选择下一节点时，将所有的节点都纳入候选列表，而后通过轮盘赌的方式进行选择。然而，选择结果很有可能是不符合公式(2)的约束条件的，会出现有的任务被重复选择，或有的单位被赋予执行多项任务。在下一节点选择结果出现后，再判断其是否符合约束条件：如果符合则继续，不符合则删去该节点，退回到上一步重新选择。这种蚂蚁路径搜索的算法显然效率比较低。

(3) 图中任务到单位的距离设为 0，表示在任务节点到单位节点的启发式信息非常高，诱导蚂蚁在任务节点选择下一节点时以高概率选中单位节点。然而，根据蚁群算法蚂蚁转移规则[4]，蚂蚁 k 在 t 时刻在节点 i 选择节点 j 的概率 $p_{ij}^{k}(t)=[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}/(\sum_{l\in N_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta})$。其中，$\eta_{ij}$ 表示一个预先给定的启发式信息，它的值等于两节点之间距离的倒数 $1/d_{ij}$。然而，根据该图的定义，任务到单位的距离 $d_{ij}=0$，会使得 $1/d_{ij}$ 的计算出错，所以必须在程序当中增加 d_{ij} 是否为 0 的判断，从而增加了算法的时间复杂度。

通过总结当前蚁群算法解决指派问题的不足，本文重新设计了一种以成本矩阵元素为蚂蚁搜索图节点，并将信息素放在节点上的蚁群算法，有效地提高了蚁群算法解决指派问题的算法效率。

3.3 以成本矩阵元素为节点的蚁群算法

首先将成本矩阵中的每一个值作为一个节点 v_{ij}，作一无向完全图，任意两点之间都连通。与 TSP 图不同，该图中的成本值和信息素都放在节点上，每个节点都赋予一个二维坐标，表示第 i 项任务由第 j 个单位完成，每个节点都与成本矩阵中的元素相对应。

蚂蚁在选路时，每选择一个节点，便将该节点和与该节点具有相同行坐标和列坐标的节点纳入禁忌表，不作为下一步选择的对象，从而逐步剔除不符合约束条件的节点，同时减少蚂蚁选择下一节点的范围，减少节点搜索次数。

具体采用蚁群算法的框架表述如下：

(1) 信息初始化：算法开始运行时，赋予每个节点相等数量的信息素 t_0。实验证明，取 $t_0=1/(n\sum_{i=1}^{n}c_{ii})$ 可达到比较好的收敛性；同时，初始化禁忌节点集为空。

(2) 选择策略：蚂蚁从剩余节点集合中，按以下选择策略选择节点 v_{ij}：

$$(i,j)=\begin{cases}\arg\max_{v\in V_i^k}\{t(v)[1/c(v)]^{\beta}\}, & q\leqslant q_0\\ (l,m), & q>q_0\end{cases} \tag{3}$$

其中，V_i^k 是剩余节点，它等于{所有节点}-{禁忌节点}。蚂蚁每选择一个节点，不仅将该节点纳入禁忌节点集合，而且将与该节点具有相同行坐标和列坐标的节点都纳入禁忌节点集合。这样做的目的是保证任何一个单位只完成一项任务，而任一任务只由一个单位完成。$t(v)$ 表示 v 点的信息素值，$c(v)$ 表示 v 点的成本值，β 是给定参数，而

$\arg\max_{v\in V_i^k}\{t(v)[1/c(v)]^{\beta}\}$ 表示在剩余节点中 $t(v)[1/c(v)]^{\beta}$ 值最大的节点下标；q_0 为给定参数，$0<q_0<1$，q 是 $(0,1)$ 内服从均匀分布的随机变量；(l,m) 依照以下概率在剩余节点内随机取值：

$$p_{lm}=\frac{[t_{lm}]^{\alpha}[\eta_{lm}]^{\beta}}{\sum_{s_{ij}\in V_i}[t_{ij}]^{\alpha}[\eta_{ij}]^{\beta}},\ s_{lm}\in V_i \qquad (4)$$

其中，t_{lm} 表示节点 v_{lm} 上的信息素；$\eta_{lm}=1/c_{lm}$ 是与成本有关的启发式信息；α、β 是两个参数，它们决定了信息素和启发式信息的相对影响力。通过测试表明，取 $\alpha=5$、$\beta=1$ 时，配合一个比较高的信息素挥发率，本算法会得到比较好的性能。若 $q_0=0.9$，式(3)表明信息素最大的节点以高概率 0.9 被蚂蚁选中，其余的节点以 0.1 的概率按式(4)参与选择。

(3) 局部更新信息素：当蚂蚁选中节点 v_{ij} 后，就更新节点 v_{ij} 上的信息素：

$$t_{ij}=(1-\xi)t_{ij}+\xi t_0,\ 0<\xi<1 \qquad (5)$$

其中，ξ 是给定参数。每当蚂蚁选中一个节点后，就按式(5)减少节点上的信息素，从而增加蚂蚁选择其它节点的概率。

(4) 局部搜索：当 m 只蚂蚁搜索完成所有节点后，则求得 m 个解。为了尽可能遍历所有解，分别在这些解的邻域中采用局部搜索算法，求出局部最优解。

(5) 全局更新信息素：当所有的 m 只蚂蚁都得到局部最优解后，全局更新节点上的信息素，通过下式完成：

$$t_{ij}=(1-\rho)t_{ij}+\rho\Delta t_{ij}^{bs},\quad \forall(i,j)\in T^{bs} \qquad (6)$$

其中，ρ 是挥发系数，$0<\rho\leqslant 1$；$\Delta t_{ij}^{bs}=1/C^{bs}$ 是信息素增量；C^{bs} 表示当前为止的全局最优解的成本和；T^{bs} 表示局部搜索的最优路径。信息素的更新只在构成 T^{bs} 的边上执行，而不是在所有边上执行。这种方式将信息素更新的算法复杂度从原来蚂蚁系统的 $O(n^2)$ 降低到 $O(n)$。同时，信息素的更新和挥发合并在了一起，对于信息素的增量也乘以参数 ρ，这使得更新后的信息素量在旧的信息素量与新释放的信息素量之间。

(6) 求全局最优解：到当前迭代次数为止，所建立的所有局部最优解中，值最小的解作为当前迭代次数的全局最优解，将其记录到最优解列表当中，直到达到最大迭代次数。

4 实验

4.1 实验 1

实验成本矩阵如下：

$$C=\begin{pmatrix}
204&230&106&102&119&216&106&218&161&149\\
140&120&199&102&243&155&124&183&115&144\\
114&163&102&190&141&214&228&110&122&149\\
163&228&102&156&221&207&114&249&184&168\\
223&208&114&194&172&172&167&241&128&237\\
108&234&236&117&221&238&178&197&238&154\\
222&158&106&167&160&110&177&149&229&217\\
126&179&103&117&110&225&170&219&198&210\\
228&172&101&206&170&142&126&157&111&117\\
201&197&147&229&111&109&175&179&120&238
\end{pmatrix}$$

算法参数设置如下：迭代次数为 50，$\alpha=5$，$\beta=1$，挥发系数 $\rho=0.9$，蚂蚁数量为 20，$q_0=0.35$，$\xi=0.2$，$t_0=5.757\ 1e-005$。全局最优解收敛过程如图 1 所示。

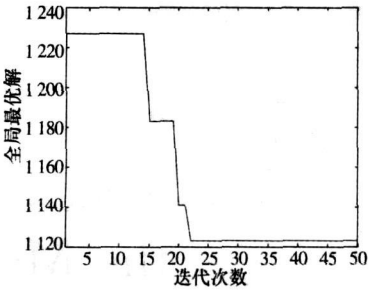

图 1　实验 1 算法全局最优解收敛过程

计算结果为 1 123，路径为：4—2—8—7—9—1—6—3—10—5。

通过 20 次计算，算法平均耗时 0.895 3 秒，迭代到最优解的平均迭代次数为 25 次，搜索到最优解的百分比为 100%。

4.2 实验 2

该实验采用文献[1]中造成匈牙利法不能收敛的指派问题，测试蚁群算法解决该问题的能力。该问题的成本矩阵如下：

$$C=\begin{pmatrix}
4&2&6&2&9&9&8&5&6&8&6&9&7&1&9&9&9&5&2&8&4&2\\
5&9&6&3&8&9&6&4&6&6&6&6&1&2&2&9&6&4&1&5&3&2\\
1&6&2&8&4&3&1&2&9&2&7&4&6&9&3&6&9&4&8&3&4&9\\
6&4&9&5&4&4&4&1&2&1&3&3&5&6&5&5&1&8&4&2&8&6\\
7&7&2&6&1&3&1&4&8&7&2&3&1&5&9&8&9&5&1&7&3&6\\
9&6&2&8&5&4&8&4&8&1&7&5&4&2&1&9&9&4&8&4&5&2\\
3&1&9&8&7&3&9&5&3&6&7&1&2&5&7&1&9&4&2&4&5&3\\
2&2&5&8&9&8&5&4&7&8&9&6&7&9&1&9&7&6&1&5&4&1\\
5&3&5&2&9&4&8&9&8&4&3&4&4&1&8&5&1&4&1&4&2&5\\
8&8&2&6&1&2&3&2&9&8&3&3&9&4&7&7&1&6&7&7&3&5\\
4&1&6&2&8&1&7&5&7&6&3&3&5&1&7&1&7&6&4&4&2&6\\
5&4&6&4&1&7&3&9&1&8&2&9&9&5&9&4&5&2&3&9&1&5\\
8&7&9&7&9&5&5&1&5&9&2&4&1&9&6&6&9&3&3&9&2&2\\
2&9&7&7&8&3&2&3&3&5&8&9&9&5&6&8&9&2&8&4&7&8\\
7&2&2&3&5&4&6&6&2&1&7&6&8&8&2&2&2&8&9&3&8&7\\
8&5&2&5&6&4&2&1&1&2&2&3&4&6&3&6&4&8&6&2&7&7\\
7&5&4&5&1&7&4&2&1&9&2&7&1&6&6&3&7&9&2&8&1&4\\
7&2&2&5&6&6&7&9&6&4&5&6&4&3&7&5&3&4&2&6&2&7\\
4&4&6&2&6&2&1&9&8&4&7&8&8&4&9&5&5&3&4&2&4&6\\
6&4&6&1&3&7&8&5&2&4&7&6&6&9&3&1&8&8&6&1&5&1\\
5&9&9&9&2&1&2&6&2&8&4&1&2&2&2&2&3&9&1&8&9&8\\
4&2&8&9&7&9&7&5&8&7&1&1&7&9&7&1&4&7&5&6&1&5
\end{pmatrix}$$

算法参数设置如下：迭代次数为 50，$\alpha=5$，$\beta=1$，挥发系数 $\rho=0.9$，蚂蚁数量为 20，$q_0=0.35$，$\xi=0.2$，$t_0=4.132\ 2e-004$。全局最优解收敛过程如图 2 所示。

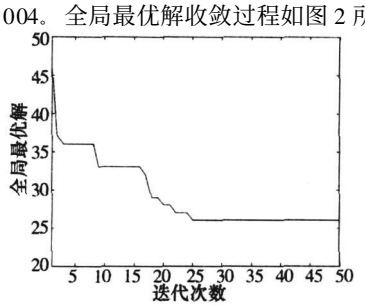

图 2　实验 2 算法全局最优解收敛过程

(下转第 112 页)

仿真程序中,令传感器节点的感知半径 $R_s = R = 1m$。与图 1 相对应,图 5 显示了 35 个节点形成的体心立方格结构传感器网络,完全覆盖大小为$(8\sqrt{5})^3 m^3$ 的立方体空间。

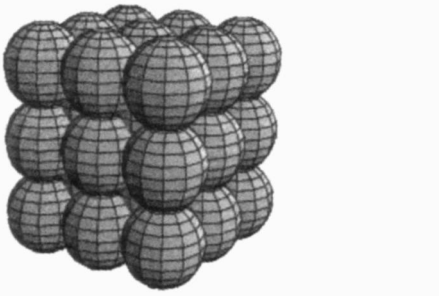

图 5　35 个球体的体心立方格覆盖

本文提出的随机节点组织策略是以体心立方格 *Voronoi* 单元对空间的划分为基础的。图 6 以线框图显示了虚拟体心立方格格点及其 *Voronoi* 单元对空间的划分形式。其中,黑点代表格点位置,深色部分突出显示一个截头八面体形状的 *Voronoi* 单元。

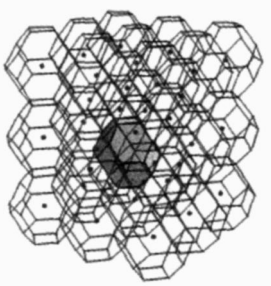

图6　体心立方格 *Voronoi* 单元对空间的划分

6　结束语

针对三维传感器网络部署问题研究薄弱的情况,本文基于体心立方格结构,提出确定部署网络中节点位置的计算公式和随机部署网络的节点组织策略。因为体心立方格是目前已知的最优空间球覆盖形式,所以根据该结构确定部署形成的网络可以在节点数固定的条件下最大化网络覆盖区域。而根据虚拟体心立方格 *Voronoi* 单元划分空间,在满足覆盖要求的条件下可以最小化活动节点个数。以后的工作中,将进一步研究无需位置信息的空间传感器节点部署和组织策略。

参考文献:

[1] Akyildiz I F, Pompili D, Melodia T. Underwater Acoustic Sensor Networks: Research Challenges [J]. Ad Hoc Networks, 2005, 3(3): 257-279.

[2] Akyildiz I F, Stuntebeck E P. Wireless Underground Sensor Networks: Research Challenges [J]. Elsevier's Journal of Ad Hoc Networks, 2006, 4(3): 669-686.

[3] Conway J H, Sloane N J A. Sphere Packings, Lattices, and Groups[M]. 3rd ed. New York: Springer-Verlag, 1999.

[4] Bulusu N, Heidemann J, Estrin D. GPS-Less Low Cost Outdoor Localization for Very Small Devices[J]. IEEE Personal Communications, 2000, 7(5): 28-34.

[5] Niculescu D, Nath B. Ad Hoc Positioning System (ASP) Using AOA [C] //Proc of IEEE INFOCOM' 03, 2003.

[6] Xu Y, Estrin D. Geography-Informed Energy Conservation for Ad Hoc Routing [C] //Proc of ACM MobiCom' 01, 2001.

[7] Santi P, Simon J. Silence is Golden with High Probability: Maintaining a Connected Backbone in Wireless Sensor Networks [C] //Proc of IEEE EWSN' 04, 2004.

(上接第 45 页)

通过 20 次计算,算法平均耗时 4.283 2 秒,迭代到最优解的平均迭代次数为 25 次,搜索到最优解的百分比为 100%。计算结果为 26。从图 2 中看出,该算法在迭代进行到一半时即已找到最优解。如果减少迭代次数,算法的性能还可以提高。

值得提出的是,算法多次计算出的最优路径存在不同,也就是说该题有多种最优路径的组合方式,如 14—13—1—17—7—15—16—22—19—5—6—9—8—18—10、14—19—1—17—13—15—2—22—4—5—6—21—8—18—10、14—19—1—8—13—15—2—22—17—5—16—9—11—18—10 等。这一结果证明蚁群算法可以找到不止一个最优解,即使问题的最优解只有一个,蚁群算法也可以搜索出仅次于最优解的多个次优解,而这个能力是匈牙利法无法达到的,文献[1]中利用改进的匈牙利法解该题计算出的结果为 27,仍然没有得到最佳解。

5　结束语

目前,解决指派问题最好的方法是匈牙利法。然而,在实际应用当中,匈牙利法存在一定不足,需要改进。利用蚁群算法解决指派问题,有以下几个方面的优势是匈牙利法不能达到的:

(1) 利用蚁群算法计算出的解具有多样性,在需要求多个最优解的应用中可以利用蚁群算法大量搜索路径的能力,对算法在收敛过程中的全局最优解进行记录和排序,即可求出多个最、次优解的集合,而匈牙利法很难完成这个任务。

(2) 通过实验发现,蚁群算法计算耗时比匈牙利法长,这是蚁群算法的性质决定的。然而,蚁群算法所具备的独特的并行计算能力是匈牙利法不能比拟的。蚁群算法很容易改进为多机协作的并行处理模型,在处理超大规模的指派问题时一定会显现出非常好的性能。

参考文献:

[1] 顾大权,左莉,侯太平,等. "匈牙利法"存在的问题及改进方法[J]. 微机发展, 2003, 13(4): 76-78.

[2] 黄茹. 一种解决指派问题的蚁群算法[J]. 西安邮电学院学报, 2006, 11(3): 106-109.

[3] Dorigo M, Stuizle T. 蚁群优化 Ant Colony Optimization[M]. 张军, 等译. 北京: 清华大学出版社, 2007.

[4] Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by a Colony of Cooperating Agents[J]. IEEE Trans on Cybernetics, 1996, 26(1): 29-41.

[5] 刘锋. 物流运筹学[M]. 上海: 上海交通大学出版社, 2005.