对"货郎担问题"的深入解析

A Study on Traveling Salesman Problem

汪林林　张　林

（重庆邮电学院计算机系　重庆400065）

Abstract　The Traveling Salesman Problem (TSP) is one of the most difficult problems that many scholars all over the world are studying. This paper points out the disparity between the definition and the classical solution of TSP and its practical applications, and the presents a new definition of TSP and its effective algorithm conforming to practical applications, thus making TSP practically more valuable.

Keywords　TSP, Algorithm, Branch-and-bound method, Time-complexity of algorithm

1. 问题的提出

"货郎担问题"是世界难于解决的著名难题之一，至今仍有不少学者在研究它。最早由 K. Menger 提出该问题的基本描述是：某售货员要到若干个村庄售货，各村庄之间的路程是已知的，售货员从他所在的商店出发，到各村庄售货一次然后返回商店。为了提高效率，求出他应选择一条总路程最短的线路。根据此描述，用图论方法给出了如下的形式描述：设 $G(V,E)$ 是一个具有边成本为 C_{ij} 的有向图，V 是 G 的结点集，E 是 G 的边集，i 和 j 为图中结点标号，C_{ij} 的定义如下：对于所有的 i 和 j，$C_{ij}>0$。若 $(i,j)\in E$，则 C_{ij} 为该边的成本，若 $(i,j)\notin E$，则 $C_{ij}=\infty$。令 $|V|=n$，并假定 $n>1$。G 的一条周游路线是包含 V 中每个结点的一个有向环。周游路线的成本是此路线上所有边的成本和，货郎担问题是求具有最小成本的周游路线，即最优货郎担问题。

显然用这样的定义来描述货郎担问题比实际情况狭隘了。实际的情况是货郎在一段线路上可往返，因此应用无向图来描述。另外，如果有如图1所示的村庄路线图。货郎仍可选择一条最优线路，只不过村1要被访问2次。而在传统的描述和定义中对此实际情况是求不出解的。以后又提出旅行商问题对货郎担问题定义进行了修正，但没有提出该问题的算法分析。下面我根据问题的实际情况提出精确的最优货郎担问题的定义、算法及其算法分析，使该问题的解决得以完善。

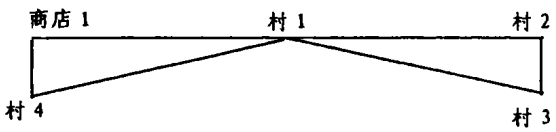

图1　村庄路线图

2. 最优货郎担问题的定义和算法

2.1　最优货郎担问题的定义

对最优货郎担问题的新描述为：某售货员要到若干村庄售货，各村庄之间的路程是已知的，售货员从他所在商店出发，到每个村庄至少售货一次然后返回商店，问他应选择一条什么路线才能使所走的总路程最短？因为考虑到一条路人是可以往返的，所以用无向图来描述如下：设 $G=(V,E)$ 是一个具有成本边 C_{ij} 无向图，V 是结点集，E 是 G 的边集，i 和 j 是 G 的结点标号。C_{ij} 定义如下：对所有的 i 和 j，$C_{ij}>0$，若 $(i,j)\in E$，这 C_{ij} 为该边的成本，若 $(i,j)\notin E$，这 $C_{ij}=\infty$。令 $|V|=n$，并假定 $n>1$。最优货郎担问题可规化求一源点 V_0 结点经过其余结点最少一次，最后返回到 V_0 结点的最短路径。

这里首先介绍一个定义及一个定理。

定义1　在带权图 $G=(V,E)$ 中，若 G 的任意一对结点 u 和 v，对 G 的其余结点 x 均满足：

$$\omega(u,v)\leqslant\omega(u,x)+\omega(x,v)$$

则称 G 为满足三角不等式的图。（其中 $\omega(u,v)$ 为 u,v 的权，即 u,v 两结点边长度）[1]

定理1　若带权图满足三角不等式，则它的最优货郎担问题与最优 H 回路（哈密尔顿环）相等（当存在回路时）[1]。

一般说，最优货郎担回路与最优 H 回路是有区别的，但仍可以将最优货郎担问题化归为最优 H 回路问题求解。方法是由给定的图 $G=(V,E)$ 构造一个新的完全图 $G'=(V',E')$，其中，$V'=V$，E' 中的每一条边 (u,v)，它的权 $\omega(u,v)$ 等于 G 中 u 与 v 之间的距离（u 与 v 之间的距离就是 u 与 v 之间一条最短路径上的边权之和），同时，在构造图 G' 的过程中，对每一条边 (u,v) 要保存其对应在图 G 中的最短路径（如，设在图 G 中 u,v 之间的最短路径为 $uxyv$，最短路径上的边权之和为15，则在图 G' 中，边 uv 就记为15_{uxyv}）。下面再介绍一个定理。

定理2　G 的最优货郎担回路的权和 G' 的最优 H 回路的权相同[1]。

因此，我们就可以把问题转为求图 G' 的最优 H 回路问题。在求出图 G' 的最优 H 回路后，由于 G' 中的每条边均保存有其对应在图 G 中的最短路径，因此，这样就可以直接求出 G 的最优货郎担回路。

2.2　最优货郎担问题的算法

下面将采用 LC 分枝-限界法来求解图 G' 的最优 H 回路问题。

2.2.1　*算法思想*　分枝-限界法就是根据一些约束条件，只产生解的部分状态空间树，从而加速搜索过程，即通过计算一小部分可行解即可从中找到最优解，因而相对地减少了计算量。这里分枝和限界是算法的基础。现分述如下。

(1)限界。所谓限界就是求出问题的上、下界，通过当前得到的限界值排除一些次优解，为最终获得最优解提示方向。

汪林林　教授，主要研究领域：计算机数据库系统、GIS 系统、计算机网络、专家系统。张　林　讲师。

对于上界，可以任取一 H 回路，令其权值为上界的初始值。在算法中，U（上界）的初值可置为∞，然后随着算法的执行而不断变化。

对于下界，可从图的距离矩阵着手给以定义。下面先给出归约矩阵的定义：如果矩阵的一行（列）至少包含一个零且其余元素均非负，则此行（列）称已归约行（列），所有行和列均为归约行和列的矩阵成为归约矩阵。在算法中，可将矩阵的行（或列）中各元素减去本行（或列）中最小元素之值，即可对该行（或列）归约。从第 i 行（或第 j 列）各元素减去的最小值，称为第 i 行（第 j 列）的约数，记作 $r(i)$（或 $r'(j)$）。那么，矩阵归约后，称各行、各列的约数之和 $r=\sum_{i=1}^{n}r(i)+\sum_{j=1}^{n}r'(j)$ 为源矩阵的约数。

因为 H 回路是通过图的每个结点一次且仅一次的回路，对图的距离矩阵的每一行，任何一个 H 回路必然含该行的一个且仅一个元素，对列的情况也是一样，求图的最优 H 回路，相当于在距离矩阵中找出 n 个元素，使它们既不同行也不同列，而且它们之和为最小值。下面再介绍一个定理。

定理3 设 $D=(d_{ij})$ 是给定图的距离矩阵，$D'=(d'_{ij})$ 是 D 的归约矩阵，若 C 是 G 的任何一条 H 回路，则有：

$$\sum_{(v_i,v_j)\in C}d_{ij}=\sum_{(v_i,v_j)\in C}d'_{ij}+r$$

从定理3可知，若 C 是原有矩阵 D 中形成的最优 H 回路，则在归约矩阵 D' 中仍然是最优 H 回路。其次，若 D' 各行各列的零元素构成一条 H 回路，则这条 H 回路就是最优 H 回路，这是因为零元素在所在的行或列中是对应最小权的边，由此可知，任何一条 H 回路的权不可能小于 r，因此 r 就是矩阵归约前 H 回路权的下界。

若矩阵归约后，零元素还不能形成 H 回路，我们可以从矩阵中选择一些边构成回路，为此我们可以构造一棵二元树，这就是下面要介绍的分枝。

(2)分枝。二元树是这样构造的，结点表示 H 回路的集合，根结点表示所有 H 回路的集合，标有 (i,j) 的结点表示含 (V_i,V_j) 边的所有回路构成的子集，标有 $\overline{i,j}$ 的结点则表示不含 (V_i,V_j) 边的所有 H 回路构成的子集，若结点 (i,j) 继续分枝，下面标有 $\overline{k,l}$ 的点表示含 (V_i,V_j)，但不含边 (V_k,V_l) 边的所有 H 回路构成的子集，而标有 (k,l) 的点则表示含 (V_i,V_j) 边又含 (V_k,V_l) 边的所有回路构成的子集，如图2所示。如果继续分枝下去，则子集将越分越小。这时须用下界进行检查，即在分枝过程中，当某一回路子集的下界大于另一子集的下界时，则前者对应的节点不在分枝。

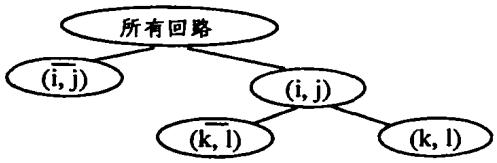

图2 分枝示意图

一般情况下，先选择零元素 $d'_{ij}=0$ 所对应的边 (V_i,V_j) 进入回路，选定之后，可以删除所在行列的元素（或将这些元素都置∞），在以后的讨论中不再考虑，此外，当 (V_i,V_j) 选入后，为了避免以后可能将边 (V_j,V_i) 也选入回路而形成 $((V_i,V_j),(V_j,V_i))$ 子回路，这时应置 $d_{ji}=\infty$，这种处置应随每一条边的选入一同进行。

在归约矩阵 D' 中，每一行列都有零元素，应选择哪一个元素所对应的边进入回路通常可如下选择。

在所有 $d'_{ij}=0$ 的元素中，选择使得 $\Delta(i,j)=\min_{k\neq i}\{d'_{ik}\}+\min_{k\neq j}\{d'_{kj}\}$ 值最大的所对应的边 (V_i,V_j)。

这是因为当边 (V_i,V_j) 选入回路后，这条边所在的行和列的其他边都不能进入回路，这样选择时，与它同一行的最小边和同一列的最小边的权和是所有 $d'_{ij}=0$ 的边的最大的，这就保证了在去掉第 i 行第 j 列的元素后，权之和的下界升较慢。

2.2.2 具体算法 根据上面的分析，下面给出本题的 LC 分枝-限界算法。

```
Procedure LCROAD(D, n)
//D 是给定图 G 的成本矩阵，n 是图 G 的结点数//
1  real D(1:n,1:n), D' (1:n,1:n),A1(1:n,1:n), AE(1:n,1:n)
2  real AN+1(1:n,1:n), A'N+1(1:n,1:n), AN+2(1:n,1:n), A'N+2
   (1:n,1:n) //U 代表上界//
3  real U //E 为当前扩展结点//
4  integer E, N, ans //ans 为回答结点//
5  定义 L 为活结点表，ET 用来存放回路的边，PARENT 用来存放
   状态空间树中各结点的父结点，函数 C(X)用来存放下界
6  将图 G 的成本矩阵 D 转化为图的距离矩阵 D' //即得到完全图
   G'//
7  E←1; L ←Φ; PARENT(1)← 0; U←∞
8  设 A1是距离矩阵 D' 的归约矩阵，r1是 D 的约数
9  C(1) ← r1; N←0
10 while C(E)< U do //当前扩展结点的下界小于上界时，AE是当
   前扩展结点 E 的归约矩阵//
11 设(i,j)是 E 的归约矩阵 AE中满足 AE(i,j) = 0 且使得 AE(i,j)
   最大的一条边
12 ET(N+1) ←(i, j); ET(N+2) ←(0, 0) //下面生成结点 E 的2
   个儿子//
13 AN+1← AE; AN+2← AE
14 将 AN+1的第 i 行及第 j 列的一切元素置∞，并且 AN+1(j,i) ←∞
15 将 AN+1归约成 A'N+1，设约数为 rN+1
16 C(N+1) ← C(E) + rN+1; PARENT(N+1) ←E
17 if N+1是一个解结点且 C(N+1) < U then
18 U ←C(N+1); ans ← N+1
19 else if N+1不是解结点且 C(N+1) < U then
20 将(N+1)加入表 L 中
21 end if
22 end if
23 AN+2(i,j)←∞
24 将 AN+2归约成 A'N+2，设约数为 rN+2
25 C(N+2)←C(E)+rN+2;PARENT(N+2)←E
26 if C(N+2)<U then
27 将(N+2)加入表 L 中
28 end if
29 N←N+2
30 从表 L 中删除 C(X)≥U 的一切结点 X//删除不符合限界条件
   的结点//
31 if L≠Φ then //取出下一结点//
32 从表 L 中选出有最小下界函数值 C(X)的结点 X 赋给 E，并从 L
   中删除这个结点
33 else
34 C(E)←∞//已没有活结点，即已找到最优 H 回路//
35 repeat
36 print('Least Cost =',U)//当前 U 就是最优货郎担问题的权值
   //
37 while ans≠0 do //打印出最优货郎担回路//
38 print(边 ET(ans)所保存的最短路径)
39 ans←PARENT(ans)
40 repeat
41 end LCROAD
```

2.2.3 算法说明与分析 U 为上界值，起初值置∞，$C(X)$ 为结点 X 的下界函数，L 为活结点表，不可能产生解的结点称为死结点，只要有一个儿子不是死结点，就将此结点加入活结点表 L 中。E 为当前扩展结点，$PARENT(X)$ 为 X 的父亲，ans 为回答结点。另外，本题要求回路的起点和终点均为 V_0，而算法 LCROAD 是求一条最优货郎担回路，这时只须把结点 V_0 指定为该回路的起点即可。因为本算法的核心是用分枝-限界法求解完全图的最优 H 回路，从而进一步求出给定图的最优货郎担回路（根据给定图的距离矩阵（代表一个完全图）中所保存的最短路径获得），所以在算法中未给出由给定图 G 求出 G' 的具体方法，另外，也略去了求一矩阵的归约矩

（下转第89页）

由多个交互自主实体组成的松散耦合网。面向 agent 方法采用粗粒度的自主计算实体(agent)作为抽象机制,以社会学的观点和人们熟悉的概念(组织,角色等)进行分析、建模、设计复杂分布式系统。面向对象方法则采用细粒度的非自主计算实体(对象)为抽象机制,对复杂问题分析与建模,面向 agent 方法较面向对象方法自然、直观、简单,容易理解。

(2)面向 agent 分析与设计处理实体间知识级交互,有专用的交互语言(如 KQML 等),实体能自主确定交互的方式、范围、时间、内容;同时,存在多种组织关系(如等级制、专家共同体、市场机制等),适合复杂系统的建模。面向对象方法中对象的交互是用处于语法级对象间的消息传递来实现,对象间的组织关系少(如包含、聚合关系),缺乏足够的机制来模型化复杂系统。

(3)面向 agent 方法主要强调角色、责任、服务、目标这些抽象机制来处理复杂性,以何时完成何种目标来分析应用领域,核心是完成的目标。面向对象方法主要强调完成目标的行为类型。因为在任何应用领域中目标比行为或计划更稳定,强调目标与强调行为这看似很小的改变,却导致实质上的不同,面向 agent 的目标分析使得系统设计更稳定、健壮、模块化,能进行渐增式地开发和测试,具有可扩展性。上下文敏感的计划提供了模块性和构成性,不必改变已有的计划,为实现同一目标的新的上下文计划可加入系统中,这使得系统能处理易变性和特例。

(4) AOP(Agent Oriented Programming)是 Shoham 在 OOP 的基础上提出的[11],并将 AOP 视为一种特殊的 OOP,但存在明显不同,如:OOP 的基本部件是对象,而 AOP 的基本部件是 agent;OOP 对定义基本部件状态没有明确规定,而 AOP 中,按 agent 的具体模型(如 DBI 模型),基本部件状态可包含信念、承诺、能力、选择等;OOP 用方法引用处理消息,而 AOP 的消息类型来自言语行为理论,可包含通知、请求、提供、承诺、拒绝等,agent 通信有专用语言,如 KQML;通常 agent 在面向 agent 方法中被实现为有意图的系统,而 OOP 将对象作为类的实例。

尽管面向对象方法与面向 agent 方法有明显的不同,但对象与 agent 之间有一种天然的进化关系,有人称 agent 是活化的对象,或对象加上思维状态等。现有一些重要的面向 agent 方法研究中都利用或扩展面向对象方法来实现面向 agent 方法(如 Gaia 和 KGR 方法等)[2,9]。

结束语 目前面向 agent 软件工程还处在发展的早期,现有的技术和方法还很不成熟。缺乏系统性和标准化,没有贯穿软件生存周期(如可行性研究、分析、设计、实现、验证、维护等)的系统化技术与方法,缺乏对 agent 重要功能需求的支持,如:行为的反应性和主动性、形式语义、行为规范既是陈述式的又可执行、与平台无关、开放性、异质性、协作能力、模块化等[3]。目前面向 agent 软件工程面临许多问题和挑战,其中最重要的问题是[1,6,7]:(1)理解 agent 求解适合的情形;(2)有原则而非形式化的开发 agent 系统。对这些问题,许多学者进行了研究,典型的工作如 Gaia 和 KGR 方法,这表明研究面向 agent 软件工程的有效方法是利用 agent 技术自身的特性及现有软件工程的成功技术与方法。

参 考 文 献

1 Jennings N R. On agent-based software engineering. Artificial Intelligence, 2000,117(2): 277～296

2 Kinny D, Georgeff M. Modelling and design of multi-agent systems. In: M. Wooldridge and N. R. Jennings, eds. Intelligent Agent Ⅱ (LNAI Volume 1193), Springer-Verlag, Berlin, Germany, 1997. 1～20

3 Fisher M, Müller J, Schroeder M. Methodological foundations for agent-based system. The Knowledge Engineering Review, 1997,12(3): 323～329

4 Nikolaos, Skarmeas. Agents as objects with knowledge base state. London: Imperial College Press, 1999

5 Jennings N R, Wooldridge M. Agent Technology: Foundations, Applications and Markets, Berlin: Springer, 1998

6 Jennings N R, Wooldridge M. Agent-oriented software engineering. In: J. Bradshaw, ed. Handbook of Agent Technology, AAAI/MIT Press, 2001

7 Wooldridge M. Agent-based software engineering. IEE Proc. Software Engineering, 1997,144 (1): 26～37

8 Wooldridge M, Jennings N R. Intelligent agents: Theory and practice. Knowledge Engineering Review, 1995,10 (2): 115～152

9 Wooldridge M, Jennings N R, Kinny D. The Gaia methodology for agent-oriented analysis and designs. Internat. J. Autonomous Agents and Multi-Agent Systems, 2000,3(3): 285～312

10 陆汝钤,石纯一等. 面向 agent 的常识知识库. 中国科学, E 辑, 2000, (5): 453～463

11 Shoham Y. Agent-oriented programming. Artificial Intelligence, 1993, 60(1): 51～92

(上接第104页)

阵部分(归约矩阵可由前面所提到的方法求得)。这两部分在算法中都用一句话代替,因为这样更有利于把算法的核心思想体现出来。对算法的其他说明可在算法的注释中得到。

在算法中,先行后列和先列后行的归约顺序产生的归约矩阵,一般说来是不同,如果最优 H 回路是唯一的,答案不会变,否则不同归约顺序可能得出不同的最优 H 回路。

该算法的效率主要是估计出状态空间树中不受限制的结点个数,我们可以用估算法。该方法是在状态空间树中生成几条 LC 路径(按 LC 值选择路经上的结点)。设 X 是某 LC 路径上的一个结点,而且 X 在状态空间树的第 i 级。在结点 X 处用限界函数确定没受限界的儿子结点数 m_i,在队列中选择下一个 LC 值最小的结点作为树中某路径上的下一个结点。这些路径的生成在一个叶结点或某一结点的所有儿子结点都已被限界终止。用这些 m_i 总和即可以估算出这棵状态空间中不受限结点的总数。

结论 本文对货郎担问题提出了精确的、合符实际的问题描述和定义,并用图论的方法研究解决此问题的算法,并对算法时间复杂度进行了分析。更重要的是该问题的提出和解决为具有相同性质的其他问题提供了有价值的解决方法。

参 考 文 献

1 肖位枢. 图论及其算法. 北京: 航空工业出版社. 1993

2 Knuth D E. The art of computer programming. Addison-Wesley Publishing Company, Inc. 1973,3

3 Boudy J A, Murty U S R. Graph theory with application. The Macmillan pross LTD, 1976

4 邹海明,余祥宣. 计算机算法基础. 华中理工大学出版社, 1984

5 徐卓群,张乃孝,杨东青,等. 数据结构. 高等教育出版社,1985

6 Howitz E. Fundamentals of computer algorithms. Computer Science Press, Pitman, Inc. 1978

7 戴一奇. 图论与代数结构. 北京: 清华大学出版社,1995