最短路径树的马尔可夫有限阶段决策算法*)

刘甜甜[1] 贾智平[1] Edwin H.-M. Sha[2]
(山东大学计算机科学与技术学院 济南 250061)[1]
(美国德州大学达拉斯分校计算机科学系 理查森德克萨斯 75083)[2]

摘 要 本文从决策的角度出发,结合马尔可夫决策过程理论,建立了计算最短路径树(SPT)的有限阶段决策模型。引入一个辅助图:反转图,结合它修改了模型的理论求解算法,提出了 SPT 反转递归迭代算法,并证明了算法的正确性。在此基础上,又提出了不使用反转图的改进模型和算法。算法的时间和空间复杂度分析表明:本文提出的算法具有分布式并行计算的特点,可以均衡各节点的工作负载,降低时间和空间复杂度,并可以有效防止环路的产生,因此可以有效应用于资源匮乏的嵌入式互连环境和对等网络环境中。

关键词 最短路径树,马尔可夫决策过程,有限阶段模型,反转图,分布式并行计算

Markov Finite Horizon Decision Algorithm of Shortest Path Tree

LIU Tian-Tian[1] JIA Zhi-Ping[1] Edwin H.-M. Sha[2]
(School of Computer Science and Technology, Shandong University, Shandong 250061)[1]
(Department of Computer Science, University of Texas at Dallas, Richardson, Texas75083)[2]

Abstract The idea of decision-making is focused on in this paper. Combining with Markov decision process theory, a SPT finite horizon decision model is established. A new auxiliary graph: Reverse Graph is introduced, combining with which the theoretical solving algorithm of the model is modified, a SPT reverse recursion iterative algorithms is proposed and its validity is proved. Base on these, an improved model and algorithm without Reverse Graph are also put forward, and time and space complexity of the two algorithms are analyzed. The theoretical analysis results show that these algorithms are characteristic by distributed parallel computation, which can balance workload between all nodes, reduce time and space complexity, and is loop-free. So it can be effectively applied in the embedded connection environment with limited resources and the peer-to-peer network environment.

Keywords Shortest path tree, Markov decision process, Finite horizon model, Reverse graph, Distributed parallel computation

1 最短路径树 SPT 的研究

最短路径树(SPT)问题一直是图论[1]、组播路由协议[2,3]等领域的经典问题。所谓最短路径树[1]是指从图中一个确定的源节点出发,计算它到其它所有节点的最短路径,从而构成的一棵树结构。上世纪 50 年代后期,Ford[4],Moore[5],Bellman[6],Dijkstra[7]先后给出了求解这一问题的基本算法。随后人们又对 SPT 的动态更新[8,9]、数据结构实现改进[10,11]等方面进行了研究探讨,提出了很多改进的算法。

计算最短路径,从整体上看是选择从源节点到目的节点的一条路径。但具体到每一个节点,其实是一个决策问题:决定从它的出边中选择哪一条作为整条路径的下一个经过的边。这样,每个节点在照顾全局的前提下进行本地决策,产生一个决策序列,从而确定整条路径。研究中发现,这个决策过程符合马尔可夫决策过程。而且使用这种本地决策的分布式算法可以将计算 SPT 的工作分布到各个节点之上,有利于平衡各个节点的工作负载,降低单个节点的工作强度和存储耗费。因此,本文结合马尔可夫决策理论来求解 SPT 问题,以适应资源匮乏型嵌入式互连环境和对等网络环境(P2P)的路由需求。

2 SPT 的马尔可夫有限阶段决策模型

2.1 马尔可夫决策过程

马尔可夫决策过程[12]简称"马氏过程",是序列决策的一种。序列决策是这样一个过程:在时刻 t 系统处于一定状态,根据这个状态选取一个行动,称为"决策"。该决策对系统的运行有两个影响:一个是产生了即得到的报酬或费用,另一个是系统的状态会按照与这个行动有关的一个概率规律在下一个阶段即时刻 $t+1$ 转移到一个新的状态。这时面临与开始时相同的问题,即选取 $t+1$ 时刻的决策。这样循环下去,从而确定一个过程的决策序列,称为一个"策略"。策略的全体构成策略空间 Π。而序列决策问题就是要在第一个决策时刻之前就预先选好一个策略,使得决策序列对应的报酬序列的某个效用函数值——准则在这个策略下达到最优。

马氏过程的特点是无后效性,可以用五元组 $\{T,S,A(i),p(\cdot|i,a),r(i,a)\}$ 表示,其中 T 为决策时刻集;S 为所有可

*)美国国家科学基金(NSF EIA-0103709)、山东省重大科技攻关项目(2005GG1101001)。**刘甜甜** 硕士研究生,主要研究方向:嵌入式系统、分布式计算;**贾智平** 教授,CCF 高级会员,主要研究方向:嵌入式系统、分布式计算、实时系统;**Edwin H.-M. Sha** 教授,主要研究方向:并行优化、嵌入式系统。

能状态集，$i\in S$；$A(i)$为系统在状态i时的可用行动集行动，a是一个行动；$p(\cdot|i,a)$是转移规律，$p(j|i,a)$表示当前状态为i、选择行动a、到下一个决策时刻转移到状态j的概率。$r(i,a)$是实值报酬函数，为正值时表示收入，为负值时表示费用。$A(i)$，$p(j|i,a)$，$r(i,a)$只依赖于当前的系统状态和选取的行动，与过去的历史无关。这个模型看似过于受限，但其实已涵盖大部分的序列决策模型。而路径选择的问题也正是符合这一模型的特点。马氏过程的策略称为“马氏策略”。

定义1 决策函数f　对于每个$i\in S$，有$f(i)\in A(i)$，则称f为确定性的决策规则或决策函数。f的全体记作F。

定义2 马氏策略π　一个决策函数序列$\pi=(f_0,f_1,f_2,\cdots)$，$f_t\in F$，$t\in N$。其中f_t是决策时刻t的决策函数，不依赖时刻t以前系统的历史，$t\in N$。马氏策略的全体集合记作$\prod_m^d$，称为马氏策略类。

2.2 SPT有限阶段模型

路径选择的过程便是一类序列决策过程。对于一个给定有向图上的给定的源节点和目的节点。假设路径已经确定到达某一中间节点，此刻要做的选择就是要在这一节点的出边上选择一条来继续构造这条路径，最终到达目的节点，从而多个中间节点的这些决策构成一个策略，即对应一条最优路径。而单个决策节点处的可选行动集(出边)、转移概率和报酬(见后文分析)不依赖于已经做好的选择，因此也正满足马氏过程的无后效性。路径的跳数虽然是不可预知的，但网络传送数据往往要避免过多的跳步，那么选择时刻便可以看作是有限的(ttl)，因此可以建立最短路径的有限阶段的马氏决策模型。

定义3 马氏有限阶段决策模型[13]　是指离散时间决策时刻的马氏决策模型，其阶段是有限的。该模型使用马氏决策五元组$\{T,S,A(i),p(\cdot|i,a),r(i,a)\}$定义，其中$T=\{0,1,\cdots,N-1\}$，$0<N<\infty$，$r(i,a)$是有界报酬函数。在选定一个策略后，决策者在阶段$0,1,\cdots,N$时依一定的概率收到一串报酬，将其累加就是该模型的具体效用函数。

设有向图$G(V,E,w)$，如图1所示，$|V|$，$|E|$分别表示节点和边的数目，$w(e_{ij})$是边e_{ij}的权重，已知源节点为s。方便起见，对节点使用数字编号表示，从0开始编号，则$s\in[0,|V-1|]$。

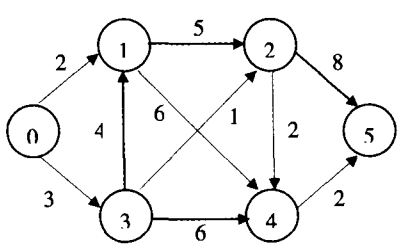


图1　有向图G(V,E,w)

根据马氏有限阶段决策的基本模型和路径选择的具体应用，SPT决策模型建立如下：

决策时刻t：$t\in T$，$T=\{0,1,\cdots,ttl\}$，其中ttl表示跳数的最大允许值，在此有$ttl=|V-1|$，即最多允许跳过除去源节点的所有的节点。

可能的状态：$S=V=[0,1,\cdots,|V-1|]$，即一个节点表示一个状态。

可能的行动集：$A(i)=\{e_{ij}\underset{j\in s}{|}e_{ij}\in E\}$，即一个状态的行动集是它代表的节点的所有出边。

报酬值：$r_t(i,a)=r_t(i,e_{ij})=w(e_{ij})$，$i,j\in S$，$e_{ij}\in E$，$t\in[0,ttl]$，其中$w(e_{ij})$是边的权重。

$r_N(i)=0$，$i\in S$，$N\in[0,ttl]$，N是结束时刻。

转移概率：$p_t(j'|i,e_{ij})=\begin{cases}1, j'=j\\0, \text{其他}\end{cases}$　$i,j,j'\in S$，$e_{ij}\in E$，$t\in[0,ttl]$。

这样，就建立好了最短路径的有限阶段马尔可夫模型。可以看到：报酬值，转移概率都不依赖于已经作好的决策。

2.3 模型的最优方程求解

对于上述有限阶段模型，文[13]给出了它的理论求解方法。

定义从时刻t到时刻N的期望报酬总和$u_t^\pi(h_t)$为：

$$u_t^\pi(h_t)\equiv E_\pi\{\sum_{n=t}^{N-1}R_n(\pi)+R_N(iN)|h_t,Y_t=i_t\}\qquad(1)$$

其中i_t，π，$R_n(\pi)$分别与2.1小节中给出的符号含义相同，$R_N(i_N)=r(i_N,a_N)$，E_π是期望算子，h_t是从时刻0到时刻t的一条轨迹$h_t=(i_0,a_0,i_1,a_1,\cdots,i_{t-1},a_{t-1},i_t)t\geqslant 0$。

定义最优方程为：

$$u_t(h_t)=\sup_{a\in A(i_t)}\{r_t(i_t,a)+\sum_{j\in S}p_t(j|i_t,a)u_{t+1}(h_t,a,j)\}\qquad(2)$$

对于$t=N$，加上边界条件：$u_N(h_N)=r_N(i_N)$　(3)

其中$h_N=(h_{N-1},a_{N-1},i_N)$。当式(2)的sup可以达到时，使用符号max或min。

在此要说明的是：式(1)，式(2)，式(3)都是通用公式，在策略空间$\prod$中进行选择，所以它们是h_t的函数，是与历史有关的。后文中将针对马氏过程的特点进行部分修改。

引入记号：$\arg\min_{x\in X}(x)=\{x'\in X|g(x')\leqslant g(x),\forall x\in X\}$。

定理1　假设u_t^* $(t\leqslant N)$是方程(2)的解，而且满足边界条件(3)，那么

(1)对于$t=0,1,\cdots,N$，$u_t^*(h_t)$对历史h_t的依赖只是与h_t的最后一个元素$i_t\in S$有关；

(2)对$t=0,1,\cdots,N$和所有的$i_t\in S$，存在$a_t^*(h_t)\in A(i_t)$满足

$$a_t^*(h_t)\in\arg\max_{a\in A(i_t)}\{r_t(i_t,a)+\sum_{j\in S}p_t(j|i_t,a)u_{t+1}^*(h_t,a,j)\}$$

那么存在最优策略而且它是马氏策略。

定理1说明了：有限决策模型在最优方程式(2)、(3)有解时存在最优的马氏策略，并且可以通过最优方程的解求解出这个最优的马氏策略。文[13]证明了定理1的正确性，因此后文将在这个定理成立的基础上，对公式进行部分修改，建立SPT有限阶段模型在马氏策略类$\prod_m^d$中的最优策略求解算法。

3 SPT反转递归迭代算法的提出与描述

3.1 多源单目SPT算法

在马氏决策理论中，有限阶段模型的最优策略求解一般采用向后递归迭代算法[12,13]。

首先，针对特定的模型对最优方程进行推导化简：

$$\begin{aligned}u_t(h_t)&=u_t(i_t)\\&=\sup_{a\in A(i_t)}\{r_t(i_t,a)+\sum_{j\in S}p_t(j|i_t,a)u_{t+1}(i_t,a,j)\}\\&=\min_{e_{uj}\in A(i_t)}\{r_t(i_t,e_{uj})+\sum_{j\in S}p_t(j|i_t,e_{uj})u_{t+1}(j)\}\\&=\min_{e_{uj}\in A(i_t)}\{r_t(i_t,e_{uj})+p_t(j|i_t,e_{uj})u_{t+1}(j)\}\\&=\min_{e_{uj}\in A(i_t)}\{r_t(i_t,e_{uj})+u_{t+1}(j)\}\end{aligned}$$

引入符号：$Another(e_{ij})=j$，即表示边e_{ij}的入端点j。

基本的向后递归迭代算法是计算特定开始状态和终止状态之间的最优决策序列，对应于上文提出的特定模型，即计算单个源节点和单个目的节点之间的一条最短路径。在此，选择图中任一节点 d 作为目的节点即终止状态。对基本算法进行修改，得到计算单源单目最短路径的算法 1 如下：

步骤 1　令 $t=0, i_t=d$

初始化，$u_k^*(n)=\infty, k=0,1,\cdots,ttl, n=0,1,\cdots,|V-1|$，

$n\neq d$，

$u_0^*(d)=0$。

步骤 2　令 $t+1\Rightarrow t$，如果 $t=ttl+1$，进入步骤 5；否则，进入步骤 3。

步骤 3　对所有 $i_t\in S, i_t\neq d$，如果 $\forall e_{ij}\in E, \exists u_{t-1}^*(j)\neq\infty$，计算：

$$u_t^*(i_t)=\min_{e_{ij}\in A(i_t)}\{r_t(i_t,e_{ij})+u_{t-1}^*(j)\},$$

并且记：

$$A_t^*(i_t)=\arg\min_{e_{ij}\in A(i_t)}\{r_t(i_t,e_{ij})+u_{t-1}^*(j)\};$$

任意取定 $f_t^*(i_t)\in Another(A_t^*(i_t))$，这样就定义了 t 时刻的决策规则 f_t^*。

进入步骤 4。

步骤 4　如果此时存在 $i_t\neq s$，有 $u_t^*(i_t)\neq\infty$，返回步骤 2，否则进入步骤 5。

步骤 5　对所有 $i\in S, i\neq d$，计算：$u_{opt}(i)=\min_{t\in[0,ttl]}\{u_t^*(i)\}$ 和相应的 $f_{opt}(i)$，算法终止。

对算法 1 的说明：t 是算法的控制变量，它既能控制算法的终止，又能有效结束环路的重复计算，同时指示了当前决策时刻某路径中边的条数，在此，称 t 为“阶段数”。$u_t^*(i_t)$ 是节点 i 到目的节点 d 的 t 阶段最短距离（后文简称“阶段距离”），$A_t^*(i_t)$ 中保存的是使之具有最短距离 $u_t^*(i_t)$ 的可选择行动。如果 $A_t^*(i_t)$ 中的元素个数大于 1，则表明 i 到 d 的 t 阶段最短路径不止一条，这时可以任意选择一条的入端点作为当前的决策行动 $f_t^*(i_t)$。步骤 4 也用来控制算法的流程，当已无有效的 $u_t^*(i_t)$ 可用时，可以及时的跳出由 t 控制的阶段循环。最终，每个节点在多个阶段距离之间选择最小的 $u_t^*(i)$ 作为最优决策，即最短距离。这样节点 d 计算出了从 s 到它的最短距离。

算法 1 本身的目的是为了计算以 s 为源节点，以 d 为目的节点的单条最短路径。但研究发现：它实际上可以计算出其他所有节点到 d 的最短距离，即以其他所有节点为源节点，以 d 为目的节点的“多源单目最短路径树”，后文中将给出这个结论的证明。

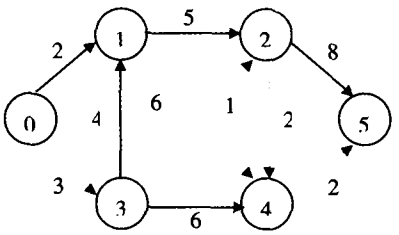


图 2　算法 1 计算出的“多源单目最短路径树”

图 2 是对图 1 中的 G，取定 $s=0, d=5$ 时算法 1 的计算结果。

从图 2 中可以看到：如果各点最终选择的 $u_{opt}(i)\neq\infty$，则 $u_{opt}(i)$ 就是该节点 i 到 d 的最短距离，而 $\pi(i)=(i, f_{opt}(i), f_{opt}(f_{opt}(i)), f_{opt}(f_{opt}(f_{opt}(i))),\cdots, d)$，则对应着这条最短路径；如果 $u_{opt}(i)=\infty$，则不存在从 i 到 d 的通路。

这样，$\pi(i)$ 形成的各条路径组成了一棵树，是以各个节点为叶，以目的节点 d 为根的一棵“多源单目最短路径树”。这棵树与要计算的“单源多目最短路径树”有点形似，它可以看作是一棵长反了的“单源多目最短路径树”。于是，我们得到启发：如果将源目节点互换，同时将有向图各边的方向反转，运行算法 1，就能得到要求的 SPT 的树形。

3.2　反转变换与反转图

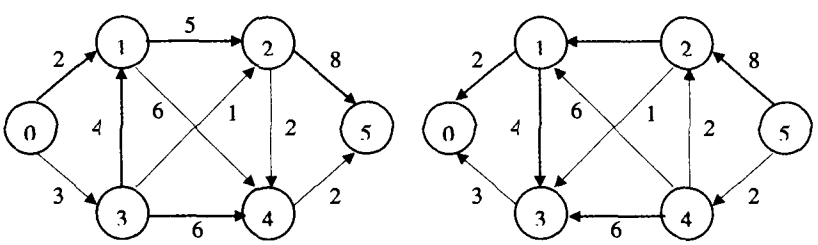
在上述启发的基础上，引入反转变换和反转图的概念，用来辅助计算 SPT。

定义 4 反转变换　将有向图的所有边的方向反转。

定义 5 反转图　一个有向图经过反转变换后得到的图叫做反转图，用符号 G^{rev} 表示。

定义 6 反转恢复　是指将反转图 G^{rev} 再经过一次反转变换恢复到原图 G 的过程。

图 3 给出了图 1 中 G 及其对应的反转图 G^{rev}。



(a)$G(V,E)$　　(b)G^{rev}

图 3　图 1 中的 G(V,E) 和它对应的反转图 G^{rev}

3.3　SPT 反转递归迭代算法

在反转图的基础上，提出算法 2：SPT 反转递归迭代算法。首先将源 s 目 d 互换，边的方向反转，运行算法 1，计算得到 G^{rev} 中以 s 为目的节点，以其它各点为源节点的“多源单目最短路径树”。然后再进行一步反转恢复，变换到原图 G。那么就可以用这棵以 s 为目的节点的“多源单目最短路径树”的树形构造出以 s 为源节点的“单源多目最短路径树”。算法 2 具体描述如下：

步骤 1：反转预处理。

步骤 1.1：对图 G 进行反转变换，得到对应的反转图 G^{rev}；

步骤 1.2：令 $s^{rev}=d, d^{rev}=s$。

步骤 2：对图 G^{rev} 和 s^{rev}, d^{rev}，运行算法 1。

步骤 3：对于所有 $i\in S, i\neq d^{rev}$，如果计算出的 $u_{opt}(i)\neq\infty$，则对应的

$\pi^{rev}(i)=(i, f_{opt}(i), f_{opt}(f_{opt}(i)), f_{opt}(f_{opt}(f_{opt}(i))),\cdots, d^{rev})$ 是从 i 到 d^{rev} 的一条最优路径；否则，不存在从 i 到 d^{rev} 的通路。

步骤4:反转恢复。将各条路径 $\pi^{rev}(i)$ 的节点序列分别逆序排列得到 $\pi(i)$,得到的便是从 d^{rev} 即 s 到 i 的最短路径,它们则共同构成了原图 G 中以 s 为源的 SPT。算法终止。

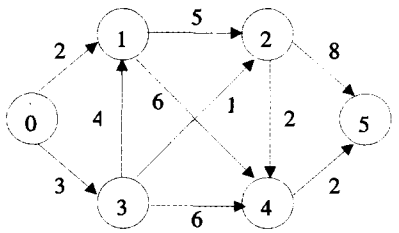


图4 算法3计算结果

图 G 以 s 为源的 SPT。

图4是对图1中的 G,取 $s=0,d=5$ 时运行算法3,得到的以 $s=0$ 为源节点的 SPT。

4 算法正确性的证明

定理2 算法1可以得到"多源单目最短路径树"。即算法结束时,每个节点都计算出了它到目的节点的最短路径。

证明:使用反证法,假设对于任一节点,不失一般性,设为 i,算法1计算得到 $u_{opt}(i)$ 对应的路径 $\pi(i)=(i,f_{opt}(i),f_{opt}(f_{opt}(i)),f_{opt}(f_{opt}(f_{opt}(i))),\cdots,d\}$ 不是从 i 到目的节点 d 的最短路径,也就是说存在一条路径 $\pi_1(i)$ 比路径 $\pi(i)$ 的距离更短,即 $\pi_1(i)$ 的距离 $u(i)_1$ 小于 $u_{opt}(i)$。

情况1:若 $\pi_1(i)$ 与 $\pi(i)$ 阶段数相同:即两路径中边的条数相同,那么在算法1的步骤3求 min 中选择的应该是 $\pi_1(i)$ 的距离 $u(i)_1$,这与已知的计算结果:选择了 $\pi(i)$ 对应的 $u_{opt}(i)$ 矛盾。

情况2:若 $\pi_1(i)$ 与 $\pi(i)$ 阶段数不同,那么在算法1的步骤5:不同阶段 t 间求 min 中选择的应该是 $\pi_1(i)$ 的距离 $u(i)_1$,这与已知的计算结果:选择了 $\pi(i)$ 对应的 $u_{opt}(i)$ 矛盾。

综上推出假设不成立,则算法1计算得到的各条路径 $\pi(i)$ 是从节点 i 到目的节点的最短路径,那么它们构成了"多源单目最短路径树",定理1成立。

定理3 算法2反转图中得到的各节点到 d^{rev} 即 s 的多源单目最短路径树,经过反转恢复后,恰好是原图中以 s 为源节点的最短路径树。

证明:使用反证法,反转恢复得到原图。假设对于反转图中任一节点,不失一般性,设为 i,计算得到的到 d^{rev} 即 s 的最短路径 $\pi^{rev}(i)$(定理1已证明),经反转恢复后得到的路径 $\pi(i)$ 不是原图的从 s 到 i 的最短路径,也就是说原图中存在一条从 s 到 i 的路径 $\pi_1(i)$,比 $\pi(i)$ 的长度更短。

那么将 $\pi_1(i)$ 上的各条边方向反转,即得到反转图中的一条从 i 到 s 的路径 $\pi_1^{rev}(i)$,且它的长度比 $\pi^{rev}(i)$ 还要短。那么 $\pi^{rev}(i)$ 就不是反转图中从 i 到 s 的最短路径。这与已知矛盾。

因此,假设不成立,则反转图中任意节点 i 到 s 的最短路径,经过反转恢复后,恰好是原图中从 s 到 i 的最短路径,定理2成立。

5 改进算法的提出与证明

5.1 新的 SPT 模型

有了对反转图的基本理解,我们提出了一个改进的算法,它能直接根据原图 G 来计算 SPT。需要修改的是算法模型的一部分,却省去了进行反转变换和反转恢复的计算量。

修改的 SPT 决策模型如下。

未经修改的部分:

决策时刻 $t:t\in T,T=\{0,1,\cdots,ttl\},ttl=|V|-1$;

可能的状态:$S=V=[0,1,\cdots,|V-1|]$,即一个节点表示一个状态;

修改的部分:

可能的行动集:$A(i)=\{e_{ji}\underset{j\in S}{|}e_{ji}\in E\}$,即一个状态的行动集是它代表的节点的所有入边;

报酬值:$r_t(i,a)=r_t(i,e_{ji})=w(e_{ji}),i,j\in S,e_{ji}\in E,t\in[0,ttl]$,其中 $w(e_{ji})$ 是边的权重。

$r_N(i)=0,i\in S,N\in[0,ttl]$,$N$ 是结束时刻。

转移概率:$p_t(j'|i,e_{ji})=\begin{cases}1,j'=j\\0,\text{其他}\end{cases}$, $i,j,j'\in S,e_{ji}\in E,t\in[0,ttl]$。

符号定义修改为:$Another(e_{ji})=j$,即表示边 e_{ji} 的出端点 j。

5.2 改进的 SPT 算法

根据上述新的 SPT 模型,对算法1和2进行部分修改、合并,得到改进的算法3如下:

步骤1:预处理及初始化。

步骤1.1:令 $s^{rev}=d,d^{rev}=s$。

步骤1.2:令 $t=0,i_t=d^{rev}$,

初始化:$u_k^*(n)=\infty,k=0,1,\cdots,ttl,n=0,1,\cdots,|V-1|,n\neq d^{rev}$,

$u_0^*(d^{rev})=0$。

步骤2:令 $t+1\Rightarrow t$,如果 $t=ttl+1$,进入步骤5;否则,进入步骤3。

步骤3:对所有 $i_t\in S,i_t\neq d$,如果 $\forall e_{jit}\in E,\exists u_{t-1}^*(j)\neq\infty$,计算:

$$u_t^*(i_t)=\min_{e_{jit}\in A(i_t)}\{r_t(i_t,e_{jit})+u_{t-1}^*(j)\},$$

并且记:

$$A_t^*(i_t)=\arg\min_{e_{jit}\in A(i_t)}\{r_t(i_t,e_{jit})+u_{t-1}^*(j)\};$$

任意取定 $f_t^*(i_t)\in Another(A_t^*(i_t))$,这样就定义了 t 时刻的决策规则 f_t^*。

进入步骤4。

步骤4:如果此时存在 $i_t\neq s^{rev}$,有 $u_t^*(i_t)\neq\infty$,返回步骤2,否则进入步骤5。

步骤5:对所有 $i\in S,i\neq d^{rev}$ 计算:$u_{opt}(i)=\min_{t\in[0,ttl]}\{u_t^*(i)\}$ 和相应的 $f_{opt}(i)$。

如果 $u_{opt}(i)\neq\infty$,对应的:

$\pi^{rev}(i)=(i,f_{opt}(i),f_{opt}(f_{opt}(i)),f_{opt}(f_{opt}(f_{opt}(i))),\cdots,d^{rev})$ 的节点序列逆序排列得到 $\pi(i)$,就是从 d^{rev} 到 i 的一条最优路径;否则,不存在从 d^{rev} 到 i 的通路。这些路径共同构成了图 G 中以 d^{rev} 即 s 为源的 SPT。算法终止。

5.3 改进算法的正确性

定理4 算法3和算法2的计算结果相同。

证明:算法2,3的不同之处有二:一是前者进行了反转变化,后者没有;二是计算模型有变化。下面将证明正是这两个不同,使得两个算法的计算结果其实是相同的。

设有向图 G,任一节点 i,设它的入边集合为 inE,出边集合为 $outE$。

对于算法2,步骤1首先进行反转,得到图 G^{rev},此时 i 的入边集合为 $outE$,出边集合为 inE。

对应于算法2的模型有：$A(i)=\{e_{ij}\mid e_{ij}\in E\}=inE$。

对于算法3，不进行反转，对应的模型也有：$A(i)=\{e_{ij}\mid e_{ij}\in E\}=inE$。

然后算法2,3都运行算法1的计算步骤，得到相应的距离 $u_{opt}(i)$和路径 $\pi^{new}(i)$和 $\pi(i)$。在此要注意的是：虽然两个算法中 *Another* 函数的定义形式不同，但其实都是表示当前节点选择的边的另一个端点，因此不影响各阶段 $f_t^*(i_t)$的选择。

因此，得证算法2和算法3实际的计算过程是相同的，那么它们的计算结果相同。

6 复杂度分析

下面从理论上分析算法1,2,3的复杂度。由上面的算法过程可以直观的看到：算法1,2,3都具有分布式并行计算的特点：即每一个 t 阶段有多个节点分别并行的参与计算 $u_t^*(i_t)$。

6.1 算法1的复杂度分析

算法的运行阶段：上限为 $O(|V|-1)$，这是由终止条件 $t=ttl+1$ 控制的，即源目之间可能的最多计算阶段是 $|V|-1$，此时得到的路径是跳跃所有节点到达目的节点的路径，它同时有效的终止了已生成环路的无限计算。

每个阶段参与计算的节点数：算法描述中单阶段一个节点 i_t 参与计算的条件是：$\forall e_{i_t j}\in E$，$\exists u_{t-1}^*(j)\neq\infty$。即只有已计算 $u_{t-1}^*(j)$节点的前驱节点才进行步骤3中的计算。这在算法的代码实现时可以很容易的提前做出判断。这样，每个阶段参与计算的节点数的上限为 $|V|-1$，而且这些节点都是并行工作的。

每个阶段单个节点的计算量：每个阶段参与的设备扫描后继节点，求同阶段的最优距离；计算量为 $O(\max\ indegree(i))$，其中 $indegree(i)$表示一个节点的入度。对于稀疏图有 $O(\max\ indegree(i))\leqslant O(|E|)$；对于非稀疏图有 $O(\max\ indegree(i))\leqslant O(|V|-1)$，当全互连时取等号。

最后一步求最优值的计算量：步骤5中每个节点并行的在不同阶段的距离间比较求min。复杂度为 $O(ttl)=O(|V-1|)$。对于一般的网络拓扑，计算量远小于 $O(|V|-1)$。

空间复杂度：每个节点要维持一个阶段距离数组，用来保存该节点到目的节点经 t 跳步时的最短路径长度，数组的元素个数是 $|V|-1$，即最多经过 $|V|-1$ 跳步到达目的节点。同时每个节点维护自己的本地邻居表，元素个数上限也是 $|V|-1$。这样就将单点独立计算时源节点要维护的全局拓扑信息分布到了各个节点上，均衡了空间复杂度。

上述对算法的分析可以看到：这种马氏有限多阶段决策解法，虽然整体上看计算复杂度是 $O(|V|*\max\ indegree(i))$（阶段数 * 单点计算量）。但它的计算工作和拓扑信息分布在了每个设备节点上，而单阶段中各个节点又是并行工作的，因此可以减少和均衡各节点的工作负载，有效利用对等互连网络环境中的各个计算资源。并且，对于现实中一般的互连拓扑图，时间复杂度则远远小于上限 $O(|V|^2)$，甚至远小于 $O(|V||E|)$。复杂度的降低和分布计算的应用，使得本算法比传统的Dijkstra算法等更适用于资源匮乏的嵌入式互连环境和各种对等网络环境。

6.2 算法2和算法3的复杂度分析

对于算法2，因为引入了反转变换和反转恢复，增加了一定的计算量。但同样，反转变换是在多个节点上并行进行的，只需对单个节点维护的本地邻居表进行修改。因此复杂度仅有 $O(|V|-1)$。

对于算法3，因为对模型进行了改进，从而省去了反转变换的复杂度，它的复杂度与算法1相同。

总结 在上面的章节中，从决策的角度出发，重新研究了路径选择问题。利用马尔可夫决策理论，建立了路径选择的马氏有限阶段模型。修改模型的理论算法，设计了分布式并行计算，均衡工作量，降低单个节点计算和空间复杂度，并能有效避免环路产生的路径选择算法（算法1）。引入反转图，对计算单条最短路径的算法1进行扩展，得到了计算最短路径树的SPT反转递归迭代算法（算法2），并证明了它的正确性。在这个基础上，又对算法2进行改进，省去了反转变换的过程，提出了基于原图计算SPT的新模型和新算法（算法3），并证明它与算法2的计算结果是相同的。最后从理论上分析了算法1,2,3的计算量和空间复杂度，说明了算法的实际计算复杂度小于传统的最短路径算法，并且把计算工作和拓扑信息分布到了各个节点上，均衡了工作量和存储需求。这在资源匮乏型嵌入式设备互连和对等网络（P2P）的路由选择上是有实际应用价值的。

参 考 文 献

1 Bondy J A, Mutry U S R. Graph Theory with Applications. London and Basingstoke: Mac Millan Press, 1976

2 Deering S, Cheriton D. Multicast Routing in Datagram Internetworks and Extended LANs. ACM Transaction on Computer Systems, 1990, 8(2): 85～110

3 Levine B N, Garcia-Luna-Aceves J J. A Comparison of Reliable Multicast Protocols. Multimedia Systems, 1998, 6(5): 334～348

4 Ford L. Network Flow Theory: [Technical Report P-932]. The Rand Corporation, 1956

5 Moore E F. The Shortest Path Through a Maze. In: Proc of International Symposium on the Theory of Switching, Part Ⅱ. Cambridge, Massachusetts: Harvard University Press, 1957. 285～292

6 Bellman R E. On a Routing Problem. Quarterly of Applied Mathematics, 1958, 16: 87～90

7 Dijkstra E W. A note on two problems in connection with graphs. Numerische Mathematik, 1959, 1: 260～271

8 Narvaez P, Siu K Y, Teng H Y. New dynamic algorithms for shortest path tree computation. IEEE Trans. Netw, 2000, 8(6): 735～746

9 Xiao Bin, Zhuge Qingfeng, Sna Hsing-Mean E. Efficient Algorithms for Dynamic Update of Shortest Path Tree in Networking. I J Comput Appl, 2004, 11(1): 60～75

10 Nonato M, Pallottino S, Xuewen B. SPT_L shortest path algorithms: review, new proposals and some experimental results: [techn. rept]. 16/99. Dip Informatica, Universitâ di Pisa, 1999

11 Goldberg A V. A Simple Shortest Path Algorithm with Linear Average Time. Lecture Notes in Computer Science, 2001, 2161: 230～241

12 刘克. 实用马尔可夫决策过程. 北京：清华大学出版社，2004

13 Puterman M L. Markov Decision Processes. NewYork : John Wiley & Sons, 1994