

星图应用哈密尔顿拉丁方的并行路由算法

王丹丹, 郭大昌, 王 静

(广东工业大学 应用数学学院, 广东 广州 510006)

摘要: 提出了一种星图的信息路由算法. 在星图中, 从一个源节点到一个目的节点传递 k 个数据包, 令第 i 个数据包将沿着第 i 条路径传输 ($1 \leq i \leq k$). 对所有的数据包, 要保证每个数据包的路径与其余数据包的路径不相交. 为了构造这样的路由, 提出了应用哈密尔顿循环拉丁方的星图信息路由算法, 并给出该算法的时间复杂度是 $O(n^2)$.

关键词: 并行算法; 星图网络; 哈密尔顿拉丁方; 最短路径

中图分类号: O157.5

文献标志码: A

文章编号: 1007-7162(2010)04-0062-06

近年来, 大规模超级计算机系统的出现, 对网络系统拓扑结构的要求越来越高, 使用图论思想和方法在网络设计和分析中飞速增长, 推动了网络拓扑结构和图论的深入研究, 新的网络结构、新的概念和问题层出不穷, 引起了人们的极大兴趣, 并产生了多种互联模式. 这些模式开发出来以便对一些重要的参数进行优化, 例如度、直径、容错性、硬件成本以及特殊的应用需要. 星图^[1-3]是最典型的一类网络拓扑结构, 具有正规性、对称性、可嵌入性等, 更具有直径短、结点的度和结点存储空间小等优点.

本文提出了一个在星图网络上路由消息的代数方法. k 个数据包同时从源节点传递到目的节点, 这里第 i 个数据包将沿着第 i 条路径. 为了保证所有的数据包到达目的节点又快又安全, 第 i 条路径必须满足与其它路径不相交. 为此, 采用了在 Cayley 图的节点运算和一种哈密尔顿循环拉丁方的特殊矩阵.

1 星图网络的一些基本概念

定义 1(星图) n 维星图 S_n 满足:

(1) n 维星图的顶点集 V 对应集合 $\{1, 2, \dots, n\}$ 上所有的置换, 即

$V = \{u_1 u_2 \dots u_n \mid u_i \in \{1, 2, \dots, n\} (i = 1, 2, \dots, n), \text{且当 } i \neq j (1 \leq i, j \leq n) \text{ 时}, u_i \neq u_j\}$.

(2) n 维星图的边集

$E = \{(u, v) \mid u, v \in V, \text{且存在 } i > 1, u_1 = v_i, u_i = v_1, \text{对任何 } j \neq i, u_j = v_j\}$.

其中在 S_n 中, 两个顶点 u, v 相邻, 当且仅当这两个顶点所对应的置换仅在第一维和另外一维上的

元素不同, 即其中一个置换右乘以 $T = \{(1, 2), (1, 3), \dots, (1, n)\}$ 中某一个对换后就得到另一个置换. 为方便起见, 用 $\sigma(i) (i = 2, 3, \dots, n)$ 来记 $(1, i)$, 表示将节点第 1 维上的数字和第 i 维的数字进行交换. 这里规定, 在 n 维星图上, 节点上最左端的维是第 1 维, 最右端的维为第 n 维, 也就是满足低维在左, 高维在右. 例如在 S_3 中, $123 [\sigma(3)] = 321$, 即节点 123 通过交换第 1 维和第 3 维得到 321, 从而节点 123 和点 321 相邻. 因此, 星图的每个点通过作用 $\sigma(i)$ 连接到另外的顶点上. 所以星图 S_n 是群 V 和生成集 T 所对应的 Cayley 图^[4].

图 1 和图 2 给出了二阶星图 S_2 和三阶星图 S_3 .



图 1 二阶星图

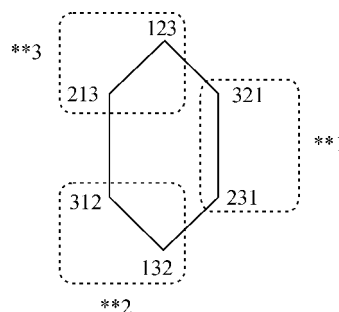


图 2 三阶星图

收稿日期: 2010-11-16

作者简介: 王丹丹(1984-), 女, 硕士研究生, 主要研究方向为图论.

定义 2^[4] 有限集合 A 上的一个一一变换叫做 A 的一个置换.

例如从 A 到 A 上的一个置换为: $1 \rightarrow 4, 2 \rightarrow 1, 3 \rightarrow 2, 4 \rightarrow 3$, 那么可以将这个置换写为 $\sigma = \begin{pmatrix} 1234 \\ 4123 \end{pmatrix}$.

定义 3^[4] 集合 S 的一个置换称为长度为 n 的环, 如果存在 $a_1, a_2, \dots, a_n \in S$ 有 $a_1\sigma = a_2, a_2\sigma = a_3, \dots, a_{n-1}\sigma = a_n, a_n\sigma = a_1$, 并且 $x\sigma = x (x \in S, x \notin \{a_1, a_2, \dots, a_n\})$. 本文把 σ 记为 $\sigma = (a_1, a_2, \dots, a_n)$.

定理 1 假设从源节点 u 到目的节点 v 的置换为 (u, v) , A 表示目的节点第 1 维上的数字, X 表示目的节点其它任何一维上的数字, c 为长度至少为 2 的循环置换的个数 ($c \geq 1$), m 表示所有循环置换中数字个数的和, 即源节点与目的节点错位数字的个数, 显然 $m \leq n$. 源节点 u 到目的节点 v 间的距离用 $d(u, v)$ 表示, 则

$$d(u, v) = c + m - \begin{cases} 0, & \text{如果 } A \text{ 和源节点的第一维相同;} \\ 2, & \text{否则.} \end{cases}$$

证明 由于星图的高度对称性, 寻找任意两节点之间的最短路径就可以映射为特定的节点到特殊节点 $v = 123 \dots n$ 之间的路径. 下面分两种情况进行证明.

(1) 源节点的第 1 维数字和目的节点的第 1 维数字相同, 也即同为 1 时:

假设源节点的置换为 $u = x_1 x_2 x_3 \dots x_n, x_1 = 1, x_i \in \{2, 3, \dots, n\} (i = 2, 3, \dots, n)$, 且当 $i \neq j (2 \leq i, j \leq n)$ 时, $x_i \neq x_j$, 目的节点为恒等置换 $v = 123 \dots n$. $C_1, C_2, \dots, C_c$ 为 c 个不相交的循环置换, 用 $|C_i|$ 表示循环置换 $C_i (i = 1, 2, \dots, c)$ 的长度, 且 $|C_i| \geq 2$. m 为 c 个循环置换中数字个数的总和, 且满足 $m < n$. 再设 $2 < k_1 < k_2 < \dots < k_c \leq n$, 不失一般性, 这里假设 $C_1 = (2, 3, \dots, k_1), C_2 = (k_1 + 1, k_1 + 2, \dots, k_2), \dots, C_c = (k_{c-1} + 1, k_{c-1} + 2, \dots, k_c)$, 并且 $k_c = m + 1$, 才能保证这些循环长度的总和为 m . 可知

$$C_1(2, 3, \dots, k_1) = (2, 3)(2, 4) \dots (2, k_1) = (1, 2)(1, 3)(1, 4)(1, 5) \dots (1, k_1 - 1)(1, k_1)(1, 2).$$

同理可以得到:

$$\begin{aligned} C_2 &= (1, k_1 + 1)(1, k_1 + 2)(1, k_1 + 3) \dots (1, k_2) \\ &\quad (1, k_1 + 1), \\ &\quad \vdots \\ C_c &= (1, k_{c-1} + 1)(1, k_{c-1} + 2)(1, k_{c-1} + 3) \dots \\ &\quad (1, k_{c-1})(1, k_c)(1, k_{c-1} + 1). \end{aligned}$$

由上面可知循环 C_1 由 k_1 个连续对换来完成, 这些对换是分别通过交换第 1 维和第 2, 3, $\dots, k_1$ 维

及第 2 维上的数字得到的. 对换的个数是循环 C_1 的长度加 1, 即 $k_1 = |C_1| + 1, C_2$ 由 $k_2 - k_1 + 1 = |C_2| + 1$ 个对换来完成, 分别通过交换第 1 维和第 $k_1 + 1, k_1 + 2, \dots, k_2$ 维及 $k_1 + 1$ 维上的数字得到, $\dots, C_c$ 是由 $k_c - k_{c-1} + 1$ 个对换完成的. 因此, 从源节点 p 到目的节点 l 的置换可以表示为

$$\begin{aligned} d = \begin{pmatrix} v \\ u \end{pmatrix} &= \begin{pmatrix} 123 \dots n \\ x_1 x_2 x_3 \dots x_n \end{pmatrix} = C_1 C_2 \dots C_c = (2, 3, \\ &\quad \dots, k_1)(k_1 + 1, k_1 + 2, \dots, k_2) \dots (k_{c-1} + 1, k_{c-1} + \\ &\quad 2, \dots, k_c) = [\sigma(2) \sigma(3) \sigma(4) \dots \sigma(k_1 - \\ &\quad 1) \sigma(k_1) \sigma(2)] [\sigma(k_1 + 1) \sigma(k_1 + \\ &\quad 2) \dots \sigma(k_2) \sigma(k_1 + 1)] \dots [\sigma(k_{c-1} + 1) \sigma(k_{c-1} + \\ &\quad 2) \sigma(k_{c-1} + 3) \dots \sigma(k_c - 1) \sigma(k_c) \sigma(k_{c-1} + 1)], \end{aligned}$$

故从 u 到 v 的置换可以用一系列的对换 $\sigma(i) (i = 2, 3, \dots, n)$ 的乘积来表示, 也即从源节点出发, 从左到右依次乘以这些对换, 就可以到达目的节点. 并且这些对换的个数为所有循环置换的长度之和加 c , 即

$$\begin{aligned} d(u, v) &= (|C_1| + 1) + (|C_2| + 1) + \dots + (|C_c| + \\ &\quad 1) = k_1 + (k_2 - k_1 + 1) + (k_3 - k_2 + 1) + \dots + \\ &\quad (k_c - k_{c-1} + 1) = k_c + c - 1 = (m + 1) + c - 1 = c + m. \end{aligned}$$

(2) 当源节点和目的节点第 1 维上的数字不同时:

这里同样假设源节点的置换为 $u = x_1 x_2 x_3 \dots x_n, x_i \in \{1, 2, \dots, n\} (i = 1, 2, 3, \dots, n)$, 且当 $i \neq j (1 \leq i, j \leq n)$ 时, $x_i \neq x_j$, 其中 $x_1 \neq 1$. 目的节点为恒等置换 $v = 123 \dots n, C_1, C_2, \dots, C_c$ 为 c 个不相交的循环置换, $|C_i|$ 表示循环置换 $C_i (i = 1, 2, \dots, c)$ 的长度, 且 $|C_i| \geq 2$. m 为 c 个循环置换中数字个数的总和, $m \leq n$. 再设 $2 < k_1 < k_2 < \dots < k_c$, 不失一般性, 假设 $k_c < x_1 < n$, $C_1 = (1, x_1, 2, 3, \dots, k_1), C_2 = (k_1 + 1, k_1 + 2, \dots, k_2), \dots, C_c = (k_{c-1} + 1, k_{c-1} + 2, \dots, k_c)$. 这里 $k_c = m - 1$, 才能保证这些循环长度的总和为 m .

从源节点到目的节点的置换为

$$\begin{aligned} d = \begin{pmatrix} u \\ v \end{pmatrix} &= \begin{pmatrix} 123 \dots n \\ x_1 x_2 x_3 \dots x_n \end{pmatrix} = C_1 C_2 C_3 \dots C_c = \\ &\quad (1, x_1, 2, 3, \dots, k_1)(k_1 + 1, k_1 + 2, \dots, k_2) \dots \\ &\quad (k_{c-1} + 1, k_{c-1} + 2, \dots, k_c) = [\sigma(x_1) \sigma(2) \sigma(3) \\ &\quad \dots \sigma(k_1)] [\sigma(k_1 + 1) \sigma(k_1 + 2) \dots \sigma(k_2 - \\ &\quad 1) \sigma(k_2) \sigma(k_1 + 1)] \dots [\sigma(k_{c-1} + 1) \sigma(k_{c-1} + \\ &\quad 2) \sigma(k_{c-1} + 3) \dots \sigma(k_c - 1) \sigma(k_c) \sigma(k_{c-1} + 1)], \end{aligned}$$

故只要从源节点出发从左到右依次乘以上面的对换就可以到达目的节点, 也即它们之间的距离为所有对换的个数, 因而

$d(u, v) = (|C_1| - 1) + (|C_2| + 1) + \cdots + (|C_c| + 1) = k_1 + (k_2 - k_1 + 1) + (k_3 - k_2 + 1) + \cdots + (k_c - k_{c-1} + 1) = k_c + (c - 1) = c + m - 2$.

例 若有源节点 $u = 13254$, 目的节点 $v = 12345$, 则 $\begin{pmatrix} 12345 \\ 13254 \end{pmatrix} = (23)(45)$, $A = 1, c = 2, m = 4$, $d(u, v) = c + m = 2 + 4 = 6$, 则从 u 到 v 的距离为 6, 即 u 与 v 之间的最短路径为 6.

定义 4^[8] 在星图中, 任意两点间的通信可以用一个置换来表示, 且置换可以由源节点以 p 个循环到达目的节点, $(\underbrace{X \cdots XA}_{k_1}) (\underbrace{X \cdots X}_{k_2}) \cdots (\underbrace{X \cdots X}_{k_c})$, 其中, $k_1 \geq 0, k_2 \geq k_3 \geq \cdots \geq k_c, A$ 表示目的节点的第 1 维, X 表示目的节点的其它任一维, 并将该形式称为一个模式.

在星图中, 源节点到目的节点的第一维主要分为两种模式: 一种是源节点到目的节点的第一维不相同, 记为 $(\underbrace{X \cdots XA}_{k_1}) (\underbrace{X \cdots X}_{k_2}) \cdots (\underbrace{X \cdots X}_{k_c})$, 另一种是源节点到目的节点的第一维相同, 记为 $(\underbrace{X \cdots X}_{k_1}) (\underbrace{X \cdots X}_{k_2}) \cdots (\underbrace{X \cdots X}_{k_c})$.

例 若有源节点 $u = 13254$, 目的节点 $v = 12345$, 则 u 与 v 间的置换为 $\begin{pmatrix} 12345 \\ 13254 \end{pmatrix} = (23)(45)$, 即 u 经过 2 个循环到达 v , 这 2 个循环的结果是 $(23)(45)$, 这属于源节点到目的节点第一维相同.

在下面的章节中, 主要讨论在星图中传递信息, 源节点到目的节点的第一维相同的情况, 即这种模式 $(\underbrace{X \cdots X}_{k_1}) (\underbrace{X \cdots X}_{k_2}) \cdots (\underbrace{X \cdots X}_{k_c})$ 的最短路径与不相交路径.

2 最短路径算法设计

定义 5 星图上的路径函数如下: 假设目的节点 $v = 123 \cdots n$, 源节点 $u = i_1 i_2 i_3 \cdots i_n, i_1 = 1, 2 \leq i_k \leq n (2 \leq k \leq n)$, 且当 $k \neq j (2 \leq k, j \leq n)$ 时, $i_k \neq i_j$, 则 $\sigma = \begin{pmatrix} 123 \cdots n \\ i_1 i_2 i_3 \cdots i_n \end{pmatrix} = C_1 C_2 \cdots C_c e_1 e_2 \cdots e_m$, 其中 $C_1, C_2, \cdots, C_c$ 为 c 个不相交的循环, 且 $|C_i| \geq 2, |C_i|$ 表示循环 $C_i (i = 1, 2, \cdots, c)$ 的长度, $e_m (1 \leq m \leq l)$ 表示长度为 1 的循环, 也就是相对于目的结点已经在正确位置上的数字.

定义 6 节点 u 和节点 v 的相关地址 r 记作 $r =$

$u \oplus v = d(u, v) = c + m$, 其中 c 为长度至少为 2 的循环置换的个数 ($c \geq 1$), m 表示所有循环置换中数字个数的和, 即源结点与目的结点错位数字的个数, 显然 $m \leq n$.

例 若 $u = 1325647, v = 1234567$, 则 $\begin{pmatrix} 1234567 \\ 1325647 \end{pmatrix} = (23)(456)$, $r = u \oplus v = d(u, v) = c + m = 2 + 5 = 7$.

定义 7 设 $T(u, S)$ 是数据从节点 u 到目的节点的逻辑传输路径, 其中 S 是一个包含一系列操作的集合, 通过 S , 数据包可以到达目的节点, S 中元素的顺序决定了 $T(u, S)$.

在 S_7 中任给一个开始节点 u 和一系列操作的集合 S , 如果要传递一个数据包到目的节点要经过中间节点, 假设节点 u 和集合 S 分别是 1325647 和 $\langle 2, 3, 4, 5, 6 \rangle$, 在 S 利用路由函数连续运算, 那么数据传递路径是 $1325647 \xrightarrow{\sigma(2)} 3125647 \xrightarrow{\sigma(3)} 2135647 \xrightarrow{\sigma(4)} 5132647 \xrightarrow{\sigma(5)} 6132547 \xrightarrow{\sigma(6)} 4132567 \xrightarrow{\sigma(4)} 2134567 \xrightarrow{\sigma(2)} 1234567$.

3 在星图网络并行计算中采用哈密尔顿拉丁方

在星图上从源节点传递 k 个数据包到目的节点, 为了既安全又快速地传递数据, 要构造一组 k 个不相交路径和最短路径. 首先, 一个节点传递的 k 个数据包, 通过一组不相交的路径传递到这个节点的 k 个邻节点. 然后通过初始阶段执行 k 个不同的运算, 并在结束阶段执行 k 个不同的运算, 可以生成这些路径, 通过这些路径可以到达目的节点. 图 3 是从源节点应用运算生成 k 条路径到达目的节点的结构图.

定义 8 设 O_s 是运行计算开始的节点集合, O_d 是运行计算结束的节点集合; 从 O_s 同时传递 k 个数据包, 使 k 个数据包到达目的节点. 集合定义如下:

$$O_s = \{s_1, s_2, \cdots, s_k\}; O_d = \{p_1, p_2, \cdots, p_k\}.$$

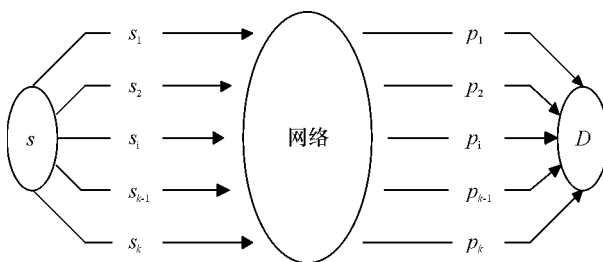


图3 运算运行的开始阶段和最后阶段

第 i 个数据包沿第 i 条路径传递, 在开始节点应用运算 s_i 得到第一个中间节点, 应用运算 p_i 从最后的中间节点传递数据到达目的节点. 在某些情况下, s_i 和 p_i 相同.

采用了哈密尔顿循环拉丁方找到了网络中 m 个集合的最短路径和不相交路径. 拉丁方是一种特殊的 $m \times m$ 矩阵, 有 m 个不同元素, 任意一个元素不会在每行每列出现两次. 这里的 m 是指拉丁方的阶. 下面定义循环拉丁方.

定义 9 ^[8] 哈密尔顿循环拉丁方 (HCLS) 记为 M_1 : 任给 m 个不同点, $a_0, a_1, \dots, a_{m-2}, a_{m-1}$, 而哈密尔顿循环 $a_i \rightarrow a_j \rightarrow \dots \rightarrow a_k \rightarrow a_i$ 是随机选择. 在哈密尔顿循环的每一行能得到哈密尔顿路径. 假设开始位置是 a_k ($0 \leq k \leq m-1$), 那么每行都不在同样的位置. 如果一个哈密尔顿路径是 $a_i \rightarrow a_j \dots \rightarrow a_k$, 可以得到 $[a_i a_j \dots a_k]$.

从 HCLS 的定义中, 又能得到改进的哈密尔顿循环矩阵 (MHCM), 记为 M_2 .

定义 10 ^[8] 任给 $M_1 = (a_{ij})$, M_2 的结构如下: $M_2 = (A_{ij})$, $A_{ij} = (a_{i,0}, a_{i,1}, \dots, a_{i,j-1}, a_{i,j})$, $a_{i,j} \in Z_m$, $0 \leq i \leq m-1, 0 \leq j \leq m-2$.

下面以例子说明定义 9 和定义 10.

例 如果哈密尔顿路径是 $2 \rightarrow 3 \rightarrow 1 \rightarrow 4$, 那么 M_1 和 M_2 的结构如下:

$$M_1 = \begin{pmatrix} 2 & 3 & 1 & 4 \\ 3 & 1 & 4 & 2 \\ 1 & 4 & 2 & 3 \\ 4 & 2 & 3 & 1 \end{pmatrix},$$

$$M_2 = \begin{pmatrix} (2) & (2,3) & (2,3,1) \\ (3) & (3,1) & (3,1,4) \\ (1) & (1,4) & (1,4,2) \\ (4) & (4,2) & (4,2,3) \end{pmatrix}.$$

定义 11 生成不相交路径矩阵 (MGNDP), 记为 M_3 , 满足以下条件:

$M_3 = (A_{i,j})$, $A_{i,j} \subset \{Z_m \cup \{s\}\}$, $0 \leq i \leq m-1, 0 \leq j \leq m-2, s$ 表示为当前节点.

- (1) $|A_{i,j}| = j+1$;
- (2) $A_{i,j} \neq A_{k,l}$, 如果 $i \neq k$;
- (3) $A_{i,j} \subset A_{i,j+1}$;
- (4) $A_{i,j+1} = A_{i,j} \cup \{s\}$, 如果 $s \in A_{i,j}$.

根据文献 [6-7], M_2 满足 M_3 的条件, 在超立方体上构造了一组节点不相交路径. M_1 属于拉丁方,

元素集合的第一列和第一行相同. 在星图网络上, M_1 中的一个元素代表了一个运算. 同样, 定义 8 中 O_s 和 O_d 分别代表了第一列的所有元素和最后一列的所有元素.

例 若 S_7 中, 源节点 $u = 1325647$, 目的节点 $v = 1234567$, 根据定义 7, 可得到运算序列 S 是 $\langle 2, 3 \rangle, \langle 4, 5, 6 \rangle$, 生成一组不相交路径. 用 S 中不同运算, 根据定义 9, 可得到 (2×2) 循环拉丁方, $M_1 = \begin{bmatrix} (2,3) & (4,5,6) \\ (4,5,6) & (2,3) \end{bmatrix}$, 其中 $(2,3), (4,5,6)$ 也是循环拉丁方.

下面的例子充分说明了这种循环在路径中怎么把数据传递到目的节点. 根据定义 8, 可以得到 $O_s = O_d = \{2, 3, 4, 5, 6\}$.

路径 1: $T(u, \langle 2, 3, 4, 5, 6, 4, 2 \rangle) : u \rightarrow 3125647 \rightarrow 2135647 \rightarrow 5132647 \rightarrow 6132547 \rightarrow 4132567 \rightarrow 2134567 \rightarrow v$.

路径 2: $T(u, \langle 4, 5, 6, 2, 3, 2, 4 \rangle) : u \rightarrow 5321647 \rightarrow 6321547 \rightarrow 4321567 \rightarrow 3421567 \rightarrow 2431567 \rightarrow 4231567 \rightarrow v$.

从定义 8 中, 由于 $O_s = O_d$ 中包含了 $\{2, 3\}$, 根据路径 1, $(2, 3)$ 可以构造拉丁方 $\begin{pmatrix} (1,2) & (1,3) \\ (1,3) & (1,2) \end{pmatrix}$, 于是, 可以找到一条新的点不相交的最短路径如下:

路径 3: $T(u, \langle 3, 2, 4, 5, 6, 4, 3 \rangle) : u \rightarrow 2315647 \rightarrow 3215647 \rightarrow 5213647 \rightarrow 6213547 \rightarrow 4213567 \rightarrow 3214567 \rightarrow v$.

同理, 如上根据路径 2, $(4, 5, 6)$ 可以构造拉丁方为 $\begin{pmatrix} (1,4) & (1,5) & (1,6) \\ (1,5) & (1,6) & (1,4) \\ (1,6) & (1,4) & (1,5) \end{pmatrix}$, 此时, 可以找到另外两条点不相交的最短路径如下:

路径 4: $T(u, \langle 5, 6, 4, 2, 3, 2, 5 \rangle) : u \rightarrow 6325147 \rightarrow 4325167 \rightarrow 5324167 \rightarrow 3524167 \rightarrow 2534167 \rightarrow 5234167 \rightarrow v$.

路径 5: $T(u, \langle 6, 4, 5, 2, 3, 2, 6 \rangle) : u \rightarrow 4325617 \rightarrow 5324617 \rightarrow 6324517 \rightarrow 3624517 \rightarrow 2634517 \rightarrow 6234517 \rightarrow v$.

当一个数据包在第 i 维上传递, 如果再次传递到相同维时, 数据回到原始位置. 不相交路径就是应用这种思想产生的. 对于第 6 条路径, 选择在源节点 u 和目的节点 v 同一位置相同的元素 7, 进行运算, 因为在星图里面 $(1, a)(1, a) = e$, 放在开始的阶段

和最后的阶段,生成的路径与上面的路径也是不相交的,得到路径如下:

路径 6: $T(u, \langle 7, 2, 3, 2, 4, 5, 6, 4, 7 \rangle): u \rightarrow 7325641 \rightarrow 3725641 \rightarrow 2735641 \rightarrow 7235641 \rightarrow 5237641 \rightarrow 6237541 \rightarrow 4237561 \rightarrow 7234561 \rightarrow v$.

如上过程找到一组不相交路径,提出了并行路由算法在星图上生成一组不相交路径. 使用“distance”表示两点间传递信息的路由的步长数.

满足上述过程的星图路由算法(SN_Routing_Algorithm)如下:

$u \leftarrow$ 起始节点 u 的地址;

$v \leftarrow$ 目的节点 v 的地址;

O_s 在起始节点 u 上进行的操作集合;

O_d 为到达目的节点 v 需进行的操作集合.

算法过程:

(1) 计算节点 A 和 B 的相对位置 $r, r = u \oplus v$.

(2) 使用相对位置 r , 产生一个运算序列 S , 在 S 的操作下可以在一个较短的时间内到达节点 v .

(3) 为设计一个节点不相交路径的集合, 从 S 中找到一个序列 S_1 , 其中 S_1 中的元素具有唯一性. 集合 $|S_1| + C$ 最短, 并且可以产生不相交路径集合. 每条路径的长度为 $|S| + C$.

(a) 使用 S_1 , 可以建立 $(n \times n)$ 的 M_1 , 这里 $n = |S_1|$.

(b) 在 M_1 第 i 行的运算会对第 i 个数据包执行, 并且 S 中剩余的操作在这一点也会被执行, 除了第一个点和最后一个点.

(c) $O_s \leftarrow O_s - S_1, O_d \leftarrow O_d - S_1$.

(4) 建立两个节点不相交路径. 每条路径的长度为 $|S| + C$.

(a) 如果 $O_s = \emptyset$, 处理过程结束.

(b) 如果在 $O_s = S_1$ 中找到集合 $\{s_i\}$, 使用 s_i , 可以建立 $(n \times n)$ 的 M_1 , 运算执行到第一步和最后一步, S 中的操作执行到中间步骤.

(c) $O_s \leftarrow O_s - \{s_i\}, O_d \leftarrow O_d - \{s_i\}$, 转到 (a).

(5) 产生剩余的路径. 每条路径的长度为 $|S| + C + 2$.

(a) 如果 $O_s = \emptyset$, 处理过程结束.

(b) 如果在 O_s 中找到集合 $\{e_m\}$, 从集合 $\{e_m\}$ 中找到 e_i , 那么这些操作执行到第一步和最后一步, S 中的操作执行到中间步骤.

(c) $O_s \leftarrow O_s - \{e_m\}, O_d \leftarrow O_d - \{e_m\}$, 转到 (a).

定理 2 构造一组 k 条节点不相交路径的时间复杂度是 $O(n^2)$.

证明 应用上述算法生成 k 条节点不相交路径. 该算法的时间复杂度取决于两个重要步骤. 一个是设计 M_1 , 需要 $O(n)$. 另一个是 SN_Routing_Algorithm 的第 5 步. 在第 5 步, 为了使运算 e_i 在 O_s 的第一步和最后一步传递, 决定了一个有序的运算是 $S \cup \{e_i\}$, 而取有序运算 $S \cup \{e_i\}$ 可减少运算量. 由于减少过程时间复杂度是 $O(n)$, 在 O_s 中的元素个数小于 n , 完成第 5 步的时间复杂度是 $O(n^2)$. 因此, 构造 k 条节点不相交路径的时间复杂度是 $O(n^2)$.

定理 3 通过 SN_Routing_Algorithm 产生的 k 个传输路径节点不相交.

证明 若节点 u 在时间 t_0 开始传输 k 个数据包, 这些数据包在时间 t_1 到达 u 的 k 个邻居节点. 然后, 每个数据包沿着一个最短路径传递到一个邻节点. 假设有两个数据包在传输过程中达到了同一个节点(不包括目的节点). 为了产生这种情况, 以下条件必须要满足. 设 S_i 和 S_j 是两个运算序列, 这两个序列在时间 t_{ij} 从起始节点向两个任意节点传输两个数据包. t_{ij} 为时间向量, 包含 t_i 和 t_j . 其中, t_i 和 t_j 分别表示一个数据包在时间 t_i 到达, 另外一个数据包在时间 t_j 到达. 这样, S_i 和 S_j 相同. 换句话说, 如果这些序列不出现, 那么一组节点不相交的路径就可以构建出来. 通过上面对算法的描述, 可知这种网络中会产生 3 种情况的路径, 故应考虑这 3 种情况.

情况 1: 设 S_{1i} 和 S_{1j} 是通过最短距离得到的两个运算序列. 由 M_3 的性质可知 S_{1i} 和 S_{1j} 一定不相同.

情况 2: 设 S_{2i} 和 S_{2j} 是通过算法(4)中得到的两个操作序列. 为证明通过情况 1 和情况 2 产生的路径是节点不相交的, 要证明 S_{2i} 和 S_{2j} 一定不相等. 考虑 S_{2j} 的第一个元素 s_k , 它不同于 S_{2j} 的第一个元素. 因此, 这些序列任何时候都不相同.

情况 3: 设 S_{3i} 和 S_{3j} 是通过算法(5)得到的两个操作序列, T 是一个操作序列, 通过 T 可以产生从源节点到目的节点一个最短路径. 那么, S_{3i} 和 S_{3j} 的第一个节点不在 T 中, 并且 S_{3i} 不包含 S_{3j} 的第一个元素, S_{3j} 也不包含 S_{3i} 的第一个元素. 所以, S_{3i} 和 S_{3j} 一定不相同. 为证明这 3 种情况产生的路径都是节点不相交的, S_{1i} 和 S_{2i} 一定不相同. 证明方法同情况 2. 考察 S_{2i} 的第一个元素, 这个元素不是 S_{3i} 的第一个元素. 因此, 这些序列任何时候都不相同.

4 结论

本文在星图网络中采用循环拉丁方构造了 k 条节点不相交路由. 该算法的时间复杂度由两个重要的步骤决定, 这两个步骤分别是设计循环拉丁方和 SN_Routing_Algorithm 的第5步. 因此, 该算法构造 k 条节点不相交路径的时间复杂度是 $O(n^2)$. 在随后的研究中, 也可以用同样的算法设计源节点和目的节点第一维不相同的并行路径和最短的不相交路径.

参考文献:

- [1] Day Khaled, Tripathi Anand. A Comparative Study of Topological Properties of Hypercubes and Star Graphs [J]. IEEE Transactions on Parallel and Distributed Systems, 1994, 5 (1): 31-38.
- [2] Chen Chi-Chang, Chen Jianer. Vertex-Disjoint Routings in Star Graphs [J]. IEEE Algorithms and Architectures for Parallel Processing, 1995, 1 (19-21): 460-464.
- [3] 徐俊明. 组合网络 [M]. 北京: 科学出版社, 2007.
- [4] 唐高华. 近世代数 [M]. 北京: 清华大学出版社, 2008.
- [5] Yeh Sheng-I, Yang Chang-Biau, Chen Hon-Chan. Fault-Tolerant Routing on the Star Graph with Safety Vectors [J]. IEEE Parallel Architectures, Algorithms and Networks, 2002 (5): 266-271.
- [6] Cho Youngjoo, Chung Ilyong. A parallel routing algorithm on circulant networks employing the Hamilton circuit latin square [J]. Information Sciences, 2006, 176 (21): 3132-2142.
- [7] Choi Dongmin, Chung Ilyong. A parallel routing algorithm on recursive cube of rings networks employing the Hamilton circuit latin square [J]. Information Sciences, 2008, 178 (6): 1533-1541.

A Parallel Routing Algorithm on Star Graph Network Employing the Hamiltonian Circuit Latin Square

Wang Dan-dan, Guo Da-chang, Wang Jing

(Faculty of Applied Mathematics, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: An algorithm for constructing the routing of a message on the star graph is proposed. The k packets are transmitted from a source node to a destination node simultaneously along paths on the star graph network, where the i th packet traverses along the i th path ($1 \leq i \leq k$). In order for all packets to arrive at the destination node quickly and securely, the i th path must be node-disjoint from all other paths. For the construction of these paths, the Hamiltonian circuit latin square (HCLS) is employed in this algorithm, which has $O(n^2)$ of the time complexity.

Key words: parallel routing algorithm; star graph network; Hamiltonian circuit latin square; the shortest path