アルゴリズム 44−1
（1995. 3. 17）

ハイパーキューブ上の単距離置換のルーティング

玉木 久夫
日本ＩＢＭ東京基礎研究所

ハイパーキューブ上での置換ルーティングの特別な場合として、すべてのパケットの始点と終点との距離が短い場合を考察する。始点と終点との最大距離が d であるような置換を距離 d 置換と呼ぶ。目的は距離制限を活かして、制限のない場合よりも少ないステップ数のルーティング法を得ることである。本稿では距離３置換および距離４置換について次の予備的な結果を与える。(1) λを任意の正整数とする。n が $18\lambda^2+6\lambda$ を越えない限り n 次元ハイパーキューブ上の任意の距離３置換に対して、$2+\lambda$ステップのルーティングが存在する（オフラインで能率良く求めることができる）。特に$\lambda=1$ と置くとこれは $n\leq 24$ である限り最適のステップ数３を与える。(2) n 次元ハイパーキューブ上の任意の距離３置換を $O(\sqrt{n})$ ステップで、距離４置換を $O(n^{5/6})$ ステップでルートするオンラインのルーティングアルゴリズムを与える。

Routing Short-Distance Permutations on the Hypercube

Hisao Tamaki

IBM Research, Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato-shi, Kanagawa, 242 Japan.
Email:htamaki@trl.ibm.co.jp

We consider a special case of permutation routing on the hypercube, where the source-destination distance of every packet is small. We call a permutation π on the set of hypercube nodes a *distance-d* permutation if the distance between v and $\pi(v)$ is at most d for every node v. Our goal is to exploit the distance constraint to obtain a routing with fewer steps than in the general case. In this note, we give the following preliminary results on distance-3 and distance-4 permutations.

(1) Let λ be an arbitrary positive integer. As long as n does not exceed $18\lambda^2+6\lambda$, any distance-3 permutation on the n-dimensional hypercube can be off-line routed in $2+\lambda$ steps. Putting $\lambda=1$ in particular, this gives an optimal 3-step routing as long as $n\leq 24$. (2) We give an on-line algorithm that routes an arbitrary distance-3 permutation on the n-dimensional hyperbube in $O(\sqrt{n})$ steps and an arbitrary distance-4 permutation in $O(n^{5/6})$ steps.

1 Introduction

Packet routing on an interconnection network is a fundamental algorithmic problem in parallel computation. Let a network be represented by a digraph G. We are given packets residing on the vertices of G and asked to deliver each packet to its specified destination vertex. The routing process consists of synchronous steps. In each step, each packet can either stay on its current vertex or go across a single edge in the direction of the edge. Multiple packets on a single vertex may go out simultaneously as long as no edge is shared by more than one packet. Similarly, multiple packets may arrive simultaneously to a single vertex as long as they come in across distinct edges. A routing algorithm is *off-line* if it sees all the source-destination pairs of packets in deciding their moves. It is *on-line*, if it is executed in a distributed manner so that each vertex decides its action based solely on the packets it has received so far and on its own previous behavior. A *permutation routing* problem is a routing problem in which every vertex is a source of at most one packet and a destination of at most one packet. Permutation routing has been a favorite subject of theoretical study (see [3] for an extensive discussion).

In this note, we are concerned with permutation routing on the *hypercube* network. The n–dimensional hypercube $Q_n$is a digraph with 2^n vertices where each vertex is identified with an n–bit string and there is an edge from u to v if and only if the bit strings u and v differ at exactly one bit position. A classical result of Beneš [1] provides an off-line algorithm that routes any permutation on the n–dimensional hypercube in $2n - 1$ steps. The randomized routing scheme of Valiant and Brebner [5] provides an on-line algorithm that routes any permutation in $O(n)$ steps with high probability. These algorithms are optimal (up to a constant factor) as long as general permutations are concerned. The question we address here is whether we can reduce the number of steps if the given permutation is known to have short source-destination distance. More precisely, an instance of the permutation routing problem on Q_n is said to be *distance-d* if the distance on Q_n from the source of each packet to its destination is at most d. Can we route every distance-d permutation in $o(n)$ steps (or, better, in $O(d)$ or even exactly d steps) when d is small?

Naturally starting from the simplest case $d = 2$, we immediately observe that a straightforward on-line algorithm routes any distance-2 partial permutation in 2 steps. This is because, for an arbitrary permutation problem not necessarily distance-2, each edge e has at most one packet that may use e in the first step and at most one packet that may use e in the last step. The next target $d = 3$ appears to be much more difficult. Neither the existence nor the absence of an off-line algorithm is known that achieves a 3-step routing for an arbitrary distance-3 permutation. We give a partial answer to this question: as long as $n \leq 24$, any distance-3 permutation on Q_n can be off-line routed in 3 steps. If we allow 4 steps, then the result holds up to $n = 84$. Theorem 2 in section 2 states these results in a more general unified form. We also give an on-line algorithm that routes an arbitrary distance-3 permutation in $O(\sqrt{n})$ steps and an arbitrary distance-4 permutation in $O(n^{5/6})$ steps.

Before diving into the main sections, we briefly discuss a type of assignment problems that will be encountered in both our off-line and on-line algorithms.

Let P be a set of packets and E be a set of edges. To each packet $p \in P$ is associated a subset E_p of E, called the *candidate edges* for p. For our application, the cardinality of E_p is uniformly s, a small constant. We are asked to assign each packet p to an edge in E_p, so that each edge is assigned at most λ packets, where λ is some predetermined capacity of each edge. The following proposition is a corollary to the Hall's Marriage Theorem.

Proposition 1 *An assignment as above exists if and only if* $|Q| \leq \lambda |\bigcup_{p \in Q} E_p|$ *for every* $Q \subseteq P$.

With s a constant and $|P|, |E| \leq m$, such an assignment can be obtained in $O(\lambda m^{1.5})$ time using the bipartite matching algorithm of Hopcroft and Karp [2]. A variant of the problem, which we call the *minimum capacity assignment* problem, is to minimize λ so that an assignment within capacity λ exists. With a straightforward binary search, this can be solved in $O(\lambda m^{2.5} \log m)$ time. If we are satisfied with an approximate solution (which is often the case), a greedy algorithm runs in $O(m \log m)$ time and achieves an assignment with capacity within s times the optimal [4].

2 Off–line routing of distance–3 permutations

Suppose an instance of the distance-3 permutation problem is given. Let P be the set of packets for which the source-destination distance is exactly 3. Then, each $p \in P$ has 6 distinct paths of length 3 from its source to destination. Let E_p denote the set of the second edges of these 6 paths. Our off-line algorithm first assigns each $p \in P$ to an edge in E_p so as to minimize the required capacity λ of the edges. This assignment determines the path each packet in P takes. Considering the congestion of at most λ on the second edge, the routing is done in $2+\lambda$ steps. Those packets with source-destination distance less than 3 are moved only in the first and the last steps, which causes no conflict.

Theorem 2 *Let* λ *be an arbitrary positive integer and suppose* $n \leq 18\lambda^2 + 6\lambda$. *Then the above off-line algorithm routes any distance-3 permutation on* Q_n *in* $2 + \lambda$ *steps.*

Proof: Suppose the permutation is given and let P and E_p for each $p \in P$ be defined as above. If suffices to show that an assignment with edge capacity λ exists. By Proposition 1, it in turn suffices to show that $|Q| \leq \lambda |E_Q|$ for every subset Q of P, where E_Q is defined to be $\bigcup_{p \in Q} E_p$. Suppose to the contrary that there is a subset Q of P such that $|Q| > \lambda |E_Q|$. For each $e \in E_Q$, let g_e denote the number of $p \in Q$ such that $e \in E_p$. Since $|E_p| = 6$ for each p, we have $\sum_{e \in E_Q} g_e = 6|Q| > 6\lambda |E_Q|$, so that the average of g_e over all $e \in E_Q$ must be strictly greater than 6λ. For each vertex v of the hypercube, let F_v denote the set of edges in E_Q that go out of v. Choose a vertex v such that the average of g_e over all $e \in F_v$ is strictly greater than 6λ. Among the edges in F_v, choose e_0 such that $g_{e_0} > 6\lambda$. Let P_0 be the set of all packets p such that that $e_0 \in E_p$. For each $p_i \in P_0$, $1 \leq i \leq g_{e_0}$, there are exactly two edges in $E_{p_i} \cap F_v$, of which one is e_0 and the other we call e_i. Then for each pair $i \neq j$, $1 \leq i, j \leq g_e$, the edges e_i and e_j must be distinct since otherwise the destinations of p_i and p_j are identical. Therefore, F_v contains at least $g_{e_0} + 1$ edges and therefore $\sum_{e \in F_v} g_e > 6\lambda(g_{e_0} + 1) > 6\lambda(6\lambda + 2)$. Since each packet p with $E_p \cap F_v \neq \emptyset$ contributes exactly 2 to this sum, the number of such packets is at least $3\lambda(6\lambda + 2) + 1$. Since the sources of these packets are at distance 1 from v, this is impossible under our assumption that $n \leq 3\lambda(6\lambda + 2)$. □

3 On–line routing of distance–3 and distance–4 permutations

Our online algorithm for distance–3 permutations operates in 3 phases. In the first phase (which consists of a single step), every packet at distance exactly 3 of its destination is sent out through arbitrary one of the 3 edges towards the destination. Thus, at the beginning of the second phase, every packet is at distance at most 2 of its destination. In this phase, each node v performs the following. Let P be the set of all packets on v that are at distance exactly 2 of their destinations. Then, each packet in P has exactly two candidate edges that would bring the packet closer to the destination. We approximately solve the minimum capacity assignment problem using the greedy algorithm mentioned earlier (let λ be the capacity obtained). Then we send the packets in P through the assigned edges in λ steps; other packets on v stay. The third phase completes the routing in an obvious single step.

Theorem 3 *The above algorithm routes an arbitrary distance-3 permutation on Q_n in $O(\sqrt{n})$ steps.*

Proof: We show that the optimal value of λ in the second phase of the algorithm is at most $\sqrt{(n+1)/2}$ at every vertex. Since we are using a 2-approximation algorithm for the assignment, the value of λ actually obtained is then at most $\sqrt{2(n+1)}$. Let P be as defined above. Note first that $|P| \leq n+1$. Therefore, by Proposition 1, it suffices to show $|Q| \leq \sqrt{|P|/2}|E_Q|$ for every subset Q of P, where E_Q is the set of all edges to which some packet of Q is potentially assigned. But, since each packet in P has a unique pair of edges to which it can be assigned, $|Q|$ is at most $\binom{|E_Q|}{2}$, from which it follows that $\sqrt{2|Q|} \leq |E_Q|$, or $|Q| \leq \sqrt{|Q|/2}|E_Q|$, establishing the result. $\square$

The on-line algorithm for distance-4 permutations is similar. In the second phase, an assignment problem is solved at each vertex for all the packets with distance exactly 3 of their destinations. In the third phase, the same is done for the packets with distance exactly 2 of their destinations.

Theorem 4 *The above algorithm routes an arbitrary distance-4 permutation on Q_n in $O(n^{5/6})$ steps.*

Proof: By a similar argument to the above, each edge is assigned at most $O(n^{2/3})$ packets in the second phase. Therefore, at the beginning of the third phase, each node has at most $O(n^{5/3})$ packets. Again by a similar argument, each edge is assigned at most $O(n^{5/6})$ packets in the third phase. $\square$

References

[1] V. Beneš. Permutation groups, complexes, and rearrangeable multistage connecting networks. *Bell System Technical Journal*, 43:1619-1640, 1964.

[2] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 15:120-30, 1986.

[3] F.T. Leighton. *Introduction to parallel algorithms and architectures.* Morgan Kaufmann, 1992.

[4] T. Tokuyama. *Personal communication.*

[5] L. Valiant and G. Brebner. Universal schemes for parallel communication. *Proc. 13th ACM Symposium on Theory of Computing*, 263-77, 1981.