A Fully Adaptive Minimal Routing Algorithm in a Crossed Cube

Kaito Satoh¹, Kousuke Mouri², and Keiichi Kaneko²

¹Graduate School of Engineering, Tokyo University of Agriculture and Technology, Koganei-shi, Tokyo, Japan
²Institute of Engineering, Tokyo University of Agriculture and Technology, Koganei-shi, Tokyo, Japan

Abstract—A crossed cube can connect the same number of vertices as a hypercube while keeping the diameter almost half of the hypercube. In this paper, we propose an algorithm that can classify the neighboring vertices of an arbitrary vertex into those on the shortest paths, the sidetrack paths, and the backtrack paths to the destination vertex. The algorithm takes time that is proportional to the dimension of the crossed cube. By using the algorithm, we can achieve the fully adaptive minimal routing in a crossed cube.

Keywords: Adaptive Routing, Hypercube, Interconnection Networks, Massively Parallel Systems, Parallel Processing

1. Introduction

Instead of sequential computation, parallel processing systems, especially massively parallel systems are focused on recently. Along this paradigm shift, many people are paying attention to the topologies for interconnection networks of massively parallel systems that have been proposed so far. The topologies are mainly divided into two categories. That is, the first one is based on the vertices of permutations such as the star graph [1], [2], the rotator graph [3], [4], the pancake graph [5], [6], the substring reversal graph [7], the bubble-sort graph [8], [9], and so on. The second one is based on the vertices of binary sequences such as the crossed cube [10], the Möbius cube [11], [12], [13], the twisted cube [14], the locally twisted cube [15], the spined cube [16], [17], the dual-cube [18], [19], [20], and so on.

The crossed cube proposed by Efe [10] is one such topology. It is a variant of the hypercube [21], which was a very popular topology. A crossed cube can connect the same number of vertices as a hypercube while keeping the diameter almost half of the hypercube. Hence, it is enthusiastically studied [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36].

Efe has introduced a shortest path routing algorithm in an n-dimensional crossed cube CQn [10]. For a current vertex and a destination vertex, the algorithm can find in O(n) time at most two neighboring vertices of the current vertex that are on the shortest paths to the destination vertex. On the other hand, Chang et al. [22] have proposed another algorithm, which can generate a shortest path in O(n) time in CQn. However, these algorithms are impractical because they cannot provide a sufficient number of selections in routing. Therefore, in this paper, we propose an algorithm in CQn. For a current vertex and a destination vertex in CQn, the algorithm classifies all neighboring vertices of the current vertex into those on the shortest paths, the sidetrack paths, and the backtrack paths to the destination vertex in O(n) time.

The rest of this paper is structured as follows. In Section 2, we introduce necessary definitions and a lemma. In Section 3, we show a core lemma in this paper followed by the description of our algorithm based on it. In Section 4, we conclude our study and give future works.

2. Preliminaries

In this section, we introduce definitions of the crossed cubes and other requisite notions as well as an important lemma.

Definition 1: For a pair of 2 bits, (u₁, u₀) and (v₁, v₀), a relation (u₁, u₀) ∼ (v₁, v₀) holds if and only if either (u₁, u₀) = (v₁, v₀) = (0, 0), (u₁, u₀) = (v₁, v₀) = (1, 0), or {(u₁, u₀), (v₁, v₀)} = {(0, 1), (1, 1)} holds.

Definition 2: An n-dimensional crossed cube CQn consists of a vertex set V(CQn) = {0, 1}ⁿ. For two vertices u = (uₙ₋₁, uₙ₋₂, ..., u₀), v = (vₙ₋₁, vₙ₋₂, ..., v₀) ∈ V(CQn), the edge (u, v) is included in the edge set E(CQn) if and only if ∃k ∈ {0, 1, 2, ..., n−1} such that uᵢ = vᵢ (k + 1 ≤ ∀i ≤ n − 1), uₖ = vₖ, uₖ₋₁ = vₖ−₁ (if k is odd), and (u₂k⁺₁, u₂k) ∼ (v₂k⁺₁, v₂k) (0 ≤ ∀i ≤ k − 1 where k = [k/2]). By using k, the neighboring vertex v is specified by u(k).

Let u = (1, 0, 0, 1, 1, 0) be a vertex in CQ₆. Then, it has six neighboring vertices: u⁽⁵⁾ = (0, 0, 1, 1, 1, 0), u⁽⁴⁾ = (1, 1, 1, 1, 1, 0), u⁽³⁾ = (1, 0, 1, 1, 1, 0), u⁽²⁾ = (1, 0, 0, 0, 1, 0), u⁽¹⁾ = (1, 0, 0, 1, 0, 0), and u⁽₀⁾ = (1, 0, 0, 1, 1, 1).

Figure 1 shows an example of a 4-dimensional crossed cube CQ₄ where a vertex (u₃, u₂, u₁, u₀) is denoted by the juxtaposition of the elements u₄úa₁ua₀ to save space.

Table 1 shows the comparison of an n-dimensional crossed cube CQn with an n-dimensional hypercube Qn, an n-dimensional 0-möbius cube 0-MQn, an n-dimensional 1-möbius cube 1-MQn, an n-dimensional twisted cube TQn, an n-dimensional locally twisted cube LTPn, an n-dimensional spined cube SQn, and a (2n − 1)-dimensional dual-cube DQn. In the table, ‘#Vert.’, ‘Deg.’, ‘Reg.’, and ‘Sym.’ represent ‘Number of Vertices’, ‘Degree’, ‘Regularity’, and ‘Symmetry’, respectively. ‘Density’ is defined by ‘#Vert.’/(‘Deg.’ × ‘Diameter’). Except for DQ₄, CQ₄ has
almost same density as other topologies, which is almost twice of \( Q_n \). In addition, \( CQ_n \) has the property of vertex and edge symmetry. On the other hand, \( DQ_n \) attains the best density, which is much larger than other topologies. However, \( DQ_n \) has just vertex-symmetric property. Moreover, it lacks the property of incremental expandability, which is pointed out to be important as a topology for interconnection networks by Duato et al. [37].

Table 1: Comparison of \( CQ_n \) with \( Q_n \), 0-MQ\( n \), 1-MQ\( n \), \( TQ_n \), \( LTQ_n \), \( SQ_n \), and \( DQ_n \).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( CQ_n )</td>
<td>( 2^n )</td>
<td>( n )</td>
<td>( (n+1)/2 )</td>
<td>( 2^n/(n+1)/2 )</td>
<td>✓</td>
</tr>
<tr>
<td>( Q_n )</td>
<td>( 2^n )</td>
<td>( n )</td>
<td>( n )</td>
<td>( 2^n/n^2 )</td>
<td>✓</td>
</tr>
<tr>
<td>0-MQ( n )</td>
<td>( 2^n )</td>
<td>( n )</td>
<td>( (n+2)/2 )</td>
<td>( 2^n/(n+2)/2 )</td>
<td>✓</td>
</tr>
<tr>
<td>1-MQ( n )</td>
<td>( 2^n )</td>
<td>( n )</td>
<td>( (n+1)/2 )</td>
<td>( 2^n/(n+1)/2 )</td>
<td>✓</td>
</tr>
<tr>
<td>( TQ_n )</td>
<td>( 2^n )</td>
<td>( n )</td>
<td>( n )</td>
<td>( 2^n/(n+1)/2 )</td>
<td>✓</td>
</tr>
<tr>
<td>( LTQ_n )</td>
<td>( 2^n )</td>
<td>( n )</td>
<td>( (n+3)/2 )</td>
<td>( 2^n/(n+3)/2 )</td>
<td>✓</td>
</tr>
<tr>
<td>( SQ_n )</td>
<td>( 2^n )</td>
<td>( n )</td>
<td>( (n+3)/2 )</td>
<td>( 2^n/(n+3)/2 )</td>
<td>✓</td>
</tr>
<tr>
<td>( DQ_n )</td>
<td>( 2\cdot2^{n-1} )</td>
<td>( 2^n )</td>
<td>( 2^n/n^2 )</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

**Definition 3:** For two vertices \( u = (u_{n-1}, u_{n-2}, \ldots, u_0) \) and \( v = (v_{n-1}, v_{n-2}, \ldots, v_0) \) in \( V(CQ_n) \), let \( h \) (\( 0 \leq h \leq n-1 \)) be the bit position such that \( u_{i} = v_{i} \) (\( h+1 \leq i \leq n-1 \)) and \( u_{h} = v_{h} \). Also, let \( \tilde{h} = \lfloor h/2 \rfloor \) and \( \hat{n} = \lfloor (n-1)/2 \rfloor \). Then, the function \( \rho(u, v, j) (0 \leq j \leq \hat{n}) \) is defined as follows:

\[
\rho(u, v, j) = \begin{cases} 
0 & (j > \hat{n}) \\
2 & (j = \hat{n}, (u_{2h+1}, u_{2h}) = (v_{2h+1}, v_{2h})) \\
1 & (j = \hat{n}, (u_{2h+1}, u_{2h}) \neq (v_{2h+1}, v_{2h})) \\
0 & (j < \hat{n}, (u_{2j+1}, u_{2j}) \approx (v_{2j+1}, v_{2j})) \\
1 & (j < \hat{n}, (u_{2j+1}, u_{2j}) \neq (v_{2j+1}, v_{2j}))
\end{cases}
\]

where for a pair of 2 bits, \((u_{2j+1}, u_{2j})\) and \((v_{2j+1}, v_{2j})\), a relation \((u_{2j+1}, u_{2j}) \approx (v_{2j+1}, v_{2j})\) holds if and only if either one of the three conditions below holds:

1. \( (u_{2j+1}, u_{2j}) = (v_{2j+1}, v_{2j}) = (0, 1) \) or \((1, 1)\), and \( \sum_{i=j+1}^{n} \rho(u, v, i) \) is even.
2. \( (u_{2j+1}, u_{2j}) = (v_{2j+1}, v_{2j}) = (0, 1) \), \((1, 1)\), and \( \sum_{i=j+1}^{n} \rho(u, v, i) \) is odd.
3. \( (u_{2j+1}, u_{2j}) = (v_{2j+1}, v_{2j}) = (0, 0) \) or \((0, 1)\).

By calculating from \( \hat{n} \) to \( j = 0 \), \( \rho(u, v, j) (0 \leq j \leq \hat{n}) \) can be obtained in \( O(n) \) time. Figure 2 shows the algorithm. For example, for two vertices \( u = (0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0) \) and \( v = (0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0) \) in \( CQ_{20} \), \( h = 15 \) and \( \hat{n} = 7 \). Then, from Definition 3, we can have \( \rho(u, v, 9) = \rho(u, v, 8) = 0, \rho(u, v, 7) = 2, \rho(u, v, 6) = 0, \rho(u, v, 5) = \rho(u, v, 4) = 1, \rho(u, v, 3) = \rho(u, v, 2) = 0, \rho(u, v, 1) = 1, \) and \( \rho(u, v, 0) = 0 \).

**Lemma 1:** For two vertices \( u \) and \( v \) in \( CQ_n \), let \( \rho(u, v) = \sum_{j=0}^{\hat{n}} \rho(u, v, j) \). Then, \( d(u, v) = \rho(u, v) \) holds.

(Proof) See the proof by Chang et al. [22].

From Lemma 1, we can calculate the distance between an arbitrary pair of vertices in \( O(n) \) time. The distance between \( u \) and \( v \) in the above example is equal to \( \sum_{j=0}^{9} \rho(u, v, j) = 5 \).

3. Classification of Neighboring Vertices

In this section, for an arbitrary vertex \( u \) and a destination vertex \( v \) in \( CQ_n \), we give a method to classify the elements of the neighboring vertex set \( N(u) \) of \( u \) into three subsets in \( O(n) \) time depending on the distances to \( v \): \( N_{-1}(u, v) = \{ n \mid n \in N(u), d(n, v) = d(u, v) - 1 \}, N_{0}(u, v) = \{ n \mid n \in N(u), d(n, v) = d(u, v) \}, \) and \( N_{+1}(u, v) = \{ n \mid n \in N(u), d(n, v) = d(u, v) + 1 \} \).

**Lemma 2:** For two vertices \( u \) and \( v \) in \( CQ_n \), let \( u^{(k)} \) be a neighboring vertex of \( u \). Then, \( \rho(u^{(k)}, v, j) = \rho(u, v, j) \) holds for \( j \geq k \) for \( k = \lfloor k/2 \rfloor \).

(Proof) Let \( u = (u_{n-1}, u_{n-2}, \ldots, u_0) \) and \( u^{(k)} = (u_{n-1}', u_{n-2}', \ldots, u_0') \). Then, \( (u_{2j+1}', u_{2j+2}') \) holds for \( j \geq k \geq h+1 \). Hence, \( \rho(u^{(k)}, v, j) = \rho(u, v, j) \) holds.

See Figure 3.

**Lemma 3:** For two vertices \( u \) and \( v \) in \( CQ_n \), let \( u^{(k)} \) (\( 0 \leq k \leq n-1 \)) be the neighboring vertices of \( u \). If \( \rho(u, v, j) (0 \leq j \leq \hat{n}) \) are all given, we can calculate \( \rho(u^{(k)}, v) (0 \leq k \leq n-1) \) in \( O(n) \) time.

(Proof) For \( u = (u_{n-1}, u_{n-2}, \ldots, u_0) \) and \( v = (v_{n-1}, v_{n-2}, \ldots, v_0) \), let \( h = \max \{ i \mid u_i = \tau_1 \}, \hat{n} = \lfloor h/2 \rfloor \),
Fig. 3: $\Delta p = \rho(u(k), v, j) - \rho(u, v, j)$ for $j (\tilde{n} \geq j \geq k+1)$. 

$k = \lfloor k/2 \rfloor$, and $u(k) = (u_{n-1}, u_{n-2}, \ldots, u_0)$. Then, we prove this lemma by three cases: $k < h$, $k = h$, and $k > h$.

Case 1 ($k < h$) From Definition 3, if $k < h$, then $\rho(u, v, k) = 0$ or 1.

Case 1a First, let us assume $\rho(u, v, k) = 0$. From Definition 3, if $k < h$ and $\rho(u, v, k) = 0$, then $(u_{2k-1}, u_{2k}) \approx (v_{2k-1}, v_{2k})$ holds. Also, from Lemma 2, $\sum_{j=k+1}^{\tilde{n}} \rho(u, v, j) = \sum_{j=k+1}^{\tilde{n}} \rho(u(k), v, j)$ holds. Since $(u_{2k-1}, u_{2k}) = (u_{2k-1}, \bar{u}_{2k})$ or $(\bar{u}_{2k-1}, u_{2k})$ (see Table 2), $(u_{2k-1}, u_{2k}) \neq (v_{2k-1}, v_{2k})$ holds from Definition 3. Therefore, $\rho(u(k), v, k) = 1$, holds, and $\rho(u(k), v, k) = \rho(u, v, k) + 1$ holds.

Table 2: $(u_{2k-1}, u_{2k})$ in $\rho(u, v, k)$.

<table>
<thead>
<tr>
<th>$(u_{2k-1}, u_{2k})$</th>
<th>$(v_{2k-1}, v_{2k})$</th>
<th>$\sum_{j=k+1}^{\tilde{n}} \rho(u, v, j)$</th>
<th>$\sum_{j=k+1}^{\tilde{n}} \rho(u(k), v, j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(0,0)$</td>
<td>$(0,0)$</td>
<td>even or odd</td>
<td>even or odd</td>
</tr>
<tr>
<td>$(0,1)$</td>
<td>$(0,1)$</td>
<td>even</td>
<td>even or odd</td>
</tr>
<tr>
<td>$(1,0)$</td>
<td>$(1,0)$</td>
<td>odd</td>
<td>odd</td>
</tr>
<tr>
<td>$(1,1)$</td>
<td>$(1,1)$</td>
<td>even</td>
<td>even or odd</td>
</tr>
</tbody>
</table>

Now, let us consider about $\rho(u(k), v, k-1)$. From Definition 2, Table 3 shows the relationship $(u_{2j+1}, u_{2j}) \sim (u_{2j+1}, u_{2j})$ ($k - 1 \geq j \geq 0$). From Lemma 2 and the fact that $\rho(u(k), v, k) = \rho(u, v, k) + 1$ just proved above, $\sum_{j=k}^{\tilde{n}} \rho(u(k), v, j) = \sum_{j=k}^{\tilde{n}} \rho(u, v, j) + 1$ holds. Hence, from Definition 3, $(u_{2k-1}, u_{2k-2}) \approx (v_{2k-1}, v_{2k-2})$ holds if and only if $(u_{2k-1}, u_{2k-2}) \approx (v_{2k-1}, v_{2k-2})$ holds. Therefore, $\rho(u(k), v, k-1) = \rho(u, v, k-1)$ holds. Similarly, for all $j (k - 1 \geq j \geq 0), \rho(u(k), v, j) = \rho(u, v, j)$.

Fig. 4: $\Delta p = \rho(u(k), v, j) - \rho(u, v, j)$ in Case 1a.

Then, similarly to the case with $S = \emptyset$, $\rho(u(k), v, j) = \rho(u, v, j) - 1 \geq \rho(u, v, j) + 1$ can be proved. From Lemma 2, $\sum_{l=1}^{\tilde{n}} \rho(u(k), v, l) = \sum_{l=1}^{\tilde{n}} \rho(u, v, l)$ holds. If $(u_{2l+1}, u_{2l}) = (0, 1), (u_{2l+1}', u_{2l}') = (1, 1)$ holds. Otherwise, that is, if $(u_{2l+1}, u_{2l}) = (1, 1), (u_{2l+1}', u_{2l}') = (0, 1)$ holds. Hence, from Definition 3, $\rho(u(k), v, l) = \begin{cases} 0 & \rho(u, v, l) = 1 \\ 1 & \rho(u, v, l) = 0 \end{cases}$ holds. Based on the similar discussion to Case 1a, $\rho(u(k), v, j) = \rho(u, v, j) - 1 \geq 0$ can be proved. From above discussion, $\rho(u(k), v) = \begin{cases} \rho(u, v) & (S = \emptyset) \\ \rho(u, v) + 1 & (S \neq \emptyset, \rho(u, v, l) = 0) \\ \rho(u, v) - 1 & (S \neq \emptyset, \rho(u, v, l) = 1) \end{cases}$
Fig. 6: $\Delta \rho = \rho(u(k), v, j) - \rho(u, v, j)$ in Case 1c.

Case 2 ($k = h$) From Definition 3, if $k = h$, $\rho(u, v, k) = 1$ or 2.

Case 2a First, let us assume $\rho(u, v, k) = 2$. From Definition 3, if $\rho(u, v, k) = 2$, $(u_{2k+1}, u_{2k}) = (v_{2k+1}, v_{2k})$. Because $u_k = \bar{u}_k$, $(u_{2k+1}', u_{2k+1}'') = (v_{2k+1}, v_{2k})$ or $(u_{2k+1}', u_{2k+1}'') = (v_{2k+1}, v_{2k})$. Hence, $\rho(u(k), v, k) = 1$ that is, $\rho(u(k), v, k) = \rho(u, v, k) - 1$ holds. Then, from Lemma 2, $\sum^n_{j=1} \rho(u(k), v, j) = \sum^n_{j=1} \rho(u, v, j) + 1$. Moreover, similarly to Case 1a, $\rho(u(k), v, j) = \rho(u, v, j)(k - 1, 0 \geq 0)$ holds. From above discussion, $\rho(u(k), v) = \rho(u, v) - 1$ holds. See Figure 6.

Fig. 7: $\Delta \rho = \rho(u(k), v, j) - \rho(u, v, j)$ in Case 2a.

Case 2b Next, let us assume $\rho(u, v, k) = 1$ and $u_k = \bar{v}_k$. From Definition 3, if $\rho(u, v, k) = 1$, $(u_{2k+1}, u_{2k}) = (v_{2k+1}, v_{2k})$ or $(u_{2k+1}, u_{2k}) = (v_{2k+1}, v_{2k})$ holds. In addition, $(u_{2k+1}', u_{2k+1}'') = (v_{2k+1}, v_{2k})$ because $u_k = \bar{v}_k$. Hence, $\rho(u(k), v, k) = 2$, that is, $\rho(u(k), v, k) = \rho(u, v, k) + 1$ holds. Then, from Lemma 2, $\sum^n_{j=1} \rho(u(k), v, j) = \sum^n_{j=1} \rho(u, v, j) + 1$. Moreover, similarly to Case 1a, $\rho(u(k), v, j) = \rho(u, v, j)(k - 1,0 \geq 0)$ holds. From above discussion, $\rho(u(k), v) = \rho(u, v) + 1$ holds. See Figure 7.

Fig. 8: $\Delta \rho = \rho(u(k), v, j) - \rho(u, v, j)$ in Case 2b.

Case 2c Finally, let us assume $\rho(u, v, k) = 1$ and $u_k = \bar{v}_k$. From Definition 3, if $\rho(u, v, k) = 1$, $(u_{2k+1}, u_{2k}) = (v_{2k+1}, v_{2k})$ or $(u_{2k+1}, u_{2k}) = (v_{2k+1}, v_{2k})$ holds. $(u_{2k+1}', u_{2k+1}'') = (v_{2k+1}, v_{2k})$ because $u_k = \bar{u}_k = v_k$. Hence, $\rho(u(k), v, k) = 0$, that is, $\rho(u(k), v, k) = \rho(u, v, k) - 1$ holds. Now, let $S$ be $\{j \mid k > j \text{ and } \rho(u, v, j) = 1\}$. If $S = \emptyset$, $\rho(u(k), v, j) = \rho(u, v, j)(k - 1, 0 \geq 0)$ from Definition 3. If $S \neq \emptyset$, let $l = \max S$. Then, $\rho(u(k), v, j) = \rho(u, v, j)(k - 1, 0 \geq l - 1)$ holds. The relationship among $(u_{2j+1}, u_{2j})$, $(v_{2j+1}, v_{2j})$, and $(u'_{2j+1}, u'_{2j})$ ($k - 1, 0 \geq 0$) is shown in Table 5. From above discussion, $\rho(u(k), v) = \rho(u, v) + 1$ holds. See Figure 9.

Case 3 ($k > h$) If $k > h$, from Definition 3, $\rho(u, v, k) = 0$. Because $u_k' = \bar{u}_k$ and $(u_{2k+1}', u_{2k}) = (v_{2k+1}, v_{2k})$, either $(u'_{2k+1}', u'_{2k}) = (v_{2k+1}, v_{2k})$ or $(u'_{2k+1}', u'_{2k}) = (v_{2k+1}, v_{2k})$ holds. Therefore, from Definition 3, $\rho(u(k), v, k) = 1$, that is, $\rho(u(k), v, k) = \rho(u, v, k) + 1$ holds. With $j = k - 1$, $\sum^n_{j=1} \rho(u, v, i) = 1$ holds. Because $(u_{2j+1}, u_{2j}) = (v_{2j+1}, v_{2j})$, $(u'_{2j+1}, u'_{2j})$ and $(v_{2j+1}, v_{2j})$ satisfy either of (2) or (3) in Definition 3. Hence, $(u_{2j+1}', u_{2j}'') = (v_{2j+1}, v_{2j})$ holds, and $\rho(u(k), v, j) = 0$. Similarly, $\rho(u(k), v, j) = \rho(u, v, j) = 0$ holds for $j (k - 1, 0 \geq 0)$.

Table 5: Relationship among $(u_{2j+1}, u_{2j})$, $(v_{2j+1}, v_{2j})$, and $(u'_{2j+1}, u'_{2j})$ ($k - 1, 0 \geq 0$) with $\rho(u, v, j)$ and $\rho(u(k), v, j)$.

<table>
<thead>
<tr>
<th>$u_{2j+1}$, $u_{2j}$</th>
<th>$(u'<em>{2j+1}, u'</em>{2j})$</th>
<th>$\rho(u, v, j)$</th>
<th>$\rho(u(k), v, j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(0, 0)$</td>
<td>$(0, 0)$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(0, 0)$</td>
<td>$(1, 0)$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(0, 0)$</td>
<td>$(0, 1)$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(1, 0)$</td>
<td>$(1, 0)$</td>
<td>$1$</td>
<td>$2$</td>
</tr>
<tr>
<td>$(1, 0)$</td>
<td>$(1, 1)$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(1, 0)$</td>
<td>$(0, 0)$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(1, 1)$</td>
<td>$(1, 0)$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(1, 1)$</td>
<td>$(1, 1)$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(1, 1)$</td>
<td>$(0, 1)$</td>
<td>$1$</td>
<td>$2$</td>
</tr>
</tbody>
</table>

Table 5, if $u_{2l+1} \neq v_{2l+1} \oplus v_{2l}$, $\rho(u(k), v, l) = \rho(u, v, l)$. From Lemma 2 and $\rho(u(k), v, k) = \rho(u, v, k) - 1$ as shown above, $\sum^n_{j=1} \rho(u(k), v, j) = \sum^n_{j=1} \rho(u, v, j) - 1$. Moreover, similarly to Case 1a, $\rho(u(k), v, j) = \rho(u, v, j)(l - 1, 0 \geq j) = 0$ holds. Otherwise, if $u_{2l+1} = v_{2l+1} \oplus v_{2l}$, $\rho(u(k), v, l) = \rho(u, v, l) + 1$. From Lemma 2 and $\rho(u(k), v, k) = \rho(u, v, k) - 1$ as shown above, $\sum^n_{j=1} \rho(u(k), v, j) = \sum^n_{j=1} \rho(u, v, j) + 1$. Moreover, similarly to Case 1c, let $T$ be $\{j \mid k \leq j \leq j = v_{2j} = 1\}$. If $T = \emptyset$, $\rho(u(k), v, j) = \rho(u, v, j)(0 \leq j \leq l - 1)$ holds. Otherwise, if $T \neq \emptyset$, let $m = \max T$. Then, $\rho(u(k), v, m) = \{\rho(u, v, m) + 1 \mid \rho(u, v, m) + 1 \mid \rho(u, v, m) = 1\}$ and $\rho(u(k), v, j) = \rho(u, v, j)(k - 1, 0 \geq 0) = 0$ holds. From above discussion, $\rho(u(k), v) = \rho(u, v) + 1$ holds. See Figure 9.
Hence, the relationship \( a \) holds. From Lemma 2, \( \sum_{j=k+1}^{n} \rho(u^{(k)}, v, j) = 1 \) holds. Therefore, the relationship among \( (u_{2h+1}, v_{2h}), \) \( (v_{2h+1}, v_{2h}), \) and \( (u_{2h+1}, u_{2h}) \) is shown in Table 6. From Table 6, if

\[
\rho(u, v, h) = 1, \quad \rho(u^{(k)}, v, h) = \rho(u, v, h) \text{ holds. Hence, from Lemma 2, } \sum_{j=0}^{n} \rho(u^{(k)}, v, j) = 1 \text{ holds. In addition, similarly to Case 1a, } \rho(u^{(k)}, v, j) = \rho(u, v, j) \quad (h-1 \geq j \geq 0) \text{ holds. If } \rho(u, v, h) = 2, \text{ then } \rho(u^{(k)}, v, h) = \rho(u, v, h) - 1 \text{ holds. Hence, from Lemma 2, } \sum_{j=0}^{n} \rho(u^{(k)}, v, j) = \sum_{j=0}^{n} \rho(u, v, j) \text{ holds. In addition, similarly to Case 1c, let } S = \{ j \mid j \leq h-1, u_{2j} = v_{2j} \}. \text{ If } S = \emptyset, \text{ then } \rho(u^{(k)}, v, j) = \rho(u, v, j) \quad (l-1 \geq j \geq 0) \text{ holds. Otherwise, if } S \neq \emptyset, \text{ let } l = \max S. \text{ Then,}
\]

\[
\rho(u^{(k)}, v, l) = \begin{cases} 
\rho(u, v, l) + 1 & (\rho(u, v, l) = 0) \\
\rho(u, v, l) - 1 & (\rho(u, v, l) = 1)
\end{cases}
\]

and \( \rho(u^{(k)}, v, i) = \rho(u, v, i) \quad (k-1 \geq i \neq l \geq 0) \text{ holds. From above discussion,}
\]

\[
\rho(u^{(k)}, v) = \begin{cases} 
\rho(u, v) + 1 & (\rho(u, v, h) = 1) \\
\rho(u, v) & (\rho(u, v, h) = 2, S \neq \emptyset) \\
\rho(u, v) + 1 & (\rho(u, v, h) = 2, S = \emptyset) \\
\rho(u, v) - 1 & (\rho(u, v, h) = 2 \neq \emptyset, \rho(u, v, l) = 1)
\end{cases}
\]

holds. See Figure 10.

From above discussion, Figures 11 shows the algorithm that calculates an array \( d(k) \) \( (0 \leq k \leq n-1) \) with \( d(k) = \rho(u^{(k)}, v) - \rho(u, v) \) for vertices \( u \) and \( v \) in \( CQ_n \). Note that \( f_1 \) and \( f_2 \) represent \( \rho(u^{(k)}, v) - \rho(u, v) \) in Case 1c and Case 2c, respectively. Also, note that because \( \rho(u^{(k)}, v, j) \), it is possible to judge whether \( (u_{2k+1}, v_{2k}) \approx (v_{2k+1}, v_{2k}) \) or not in \( O(1) \) time by checking if \( r[k] \) is odd or even.

For a source vertex \( u_{(0,1,1,0,0,1,1,1,0,1,0,1)} \) and a destination vertex \( v_{(1,0,1,0,1,0,1,0,1,0,1,0)} \) in \( CQ_{12} \), we give an example of classification of neighboring vertices of the source vertex and intermediate vertices for sending a message from \( u \) to \( v \). Our algorithm classifies the neighboring vertices of \( u \), \( N(u) \), into \( N_{-1}(u, v) \) \( = \{ (0,0,1,1,1,0,0,1,1,0,1,0) \} \) \( \neq \emptyset \), and \( N_{0}(u, v) \) \( = \{ (0,0,1,1,1,0,0,1,1,0,1,0) \} \) \( \neq \emptyset \), and \( N_{1}(u, v) \) \( = \{ (0,0,1,1,1,0,0,1,1,0,1,0) \} \) \( \neq \emptyset \). See Figure 12. The six vertices in \( N_{-1}(u, v) \) are on the shortest paths from \( u \) to \( v \). That is, the distance from each vertex in \( N_{-1}(u, v) \) is \( d(u, v) - 1 \). The three vertices in \( N_{0}(u, v) \) are on the sidetrack paths from \( u \) to \( v \). That is, the distance from each vertex in \( N_{0}(u, v) \) is \( d(u, v) \). The three vertices in \( N_{1}(u, v) \) are on the backtrack paths from \( u \) to \( v \). That is, the distance from each vertex in \( N_{1}(u, v) \) is \( d(u, v) + 1 \).

| Table 6 | Relationship among \( (u_{2h+1}, v_{2h}), (v_{2h+1}, v_{2h}), \) and \( (u'_{2h+1}, v'_{2h}) \) with \( \rho(u^{(k)}, v, h) \) and \( \rho(u^{(k)}, v, h) \).
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_{2h+1}, v_{2h} )</td>
<td>( u_{2h+1}, v_{2h} )</td>
<td>( u'<em>{2h+1}, v'</em>{2h} )</td>
<td>( \rho(u^{(k)}, v, h) )</td>
<td>( \rho(u^{(k)}, v, h) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (0,0) )</td>
<td>( (0,0) )</td>
<td>( (0,0) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (0,0) )</td>
<td>( (1,0) )</td>
<td>( (0,0) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (0,0) )</td>
<td>( (1,0) )</td>
<td>( (0,0) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (0,1) )</td>
<td>( (0,0) )</td>
<td>( (1,1) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (0,1) )</td>
<td>( (1,1) )</td>
<td>( (0,1) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (1,0) )</td>
<td>( (0,0) )</td>
<td>( (1,1) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (1,0) )</td>
<td>( (1,1) )</td>
<td>( (0,0) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (1,1) )</td>
<td>( (0,0) )</td>
<td>( (1,1) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (1,1) )</td>
<td>( (1,1) )</td>
<td>( (0,0) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (0,0) )</td>
<td>( (0,1) )</td>
<td>( (1,0) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (0,0) )</td>
<td>( (1,0) )</td>
<td>( (1,1) )</td>
<td>( 1 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 9: \( \Delta \rho = \rho(u^{(k)}, v, j) - \rho(u, v, j) \) in Case 2c.

Fig. 10: \( \Delta \rho = \rho(u^{(k)}, v, j) - \rho(u, v, j) \) in Case 3c.
procedure class($u$, $v$, $\rho(u,v,j)$ ($0 \leq j \leq n$))
begin
  $f_1 := 0$;
  $f_2 := -1$;
  $h := \lfloor \max\{k \mid u_k \neq v_k\}/2 \rfloor$;
  $r[n] := \rho(u,v,n)$;
  for $i := n - 1$ to 0 step -1 do
    $r[i] := r[i+1] + \rho(u,v,i+1)$;
  for $k := 0$ to $n - 1$ do begin
    $k := \lfloor k/2 \rfloor$;
    if $k < h$ then begin
      if $\rho(u,v,k) = 0$ then
        $d[k] := 1$
      else if $(u_{2k+1}, u_{2k}) \neq (v_{2k+1}, v_{2k})$ then
        $d[k] := -1$
      else $d[k] := f_1$;
      if $k$ is even and $u_k = v_k = 1$ then
        if $f_1 + 1 \geq 0$ then
          $f_1 := 1$
        else $f_1 := -1$
      else $f_1 := -1$;
      if $k$ is even and $\rho(u,v,k) > 0$ then
        if $u_{k+1} \neq v_{k+1}$ then
          $f_2 := f_1$
        else $f_1 := -1$ end
    else if $k = h$ then begin
      if $\rho(u,v,k) = 2$ then
        $d[k] := -1$
      else if $\rho(u,v,k) = 1$ and $u_k = v_k$ then
        $d[k] := 1$
      else $d[k] := f_2$ end
    else if $\rho(u,v,h) = 1$ then
      $d[k] := 1$
    else $d[k] := f_1$
  end;
return $d$
end

Fig. 11: Algorithm for calculation of $d[k] = \rho(u^{(k)}, v) - \rho(u, v)$.

1, 1), (0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0)$. Among $N_{-1}(u_1, v)$, we pick up the vertex $u_2 = (1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1)$ to forward the message. Then, $N(u_2)$ is classified into three subsets: $N_{-1}(u_2, v) = \{(1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1)\}$, $N_0(u_2, v) = \emptyset$, and $N_{+1}(u_2, v) = N(u_2) \setminus N_{-1}(u_2, v)$. Since $N_{-1}(u_2, v)$ is a singleton set, we select $u_3 = (1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0)$ to forward the message. Then, $N(u_3)$ is classified into three subsets: $N_{-1}(u_3, v) = \{(1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1)\}$, $N_0(u_3, v) = \emptyset$, and $N_{+1}(u_3, v) = N(u_3) \setminus N_{-1}(u_3, v)$. Again, since $N_{-1}(u_3, v)$ is a singleton set, we select $u_4 = (1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1)$ to forward the message. Then, $u_4$ and $v$ are adjacent, and the message is sent to $v$. Figure 13 illustrates the construction of this path.

4. Conclusion and Future Works

In this paper, we have proposed an algorithm for the shortest-path routing in an $n$-dimensional crossed cube $CQ_n$. For a current vertex $c$ and a destination vertex $d$, the algorithm classifies the neighboring vertices of $c$ into three subsets: $N_{-1}(c, d) = \{n \mid n \in N(c), d(n, d) = d(c, d) + 1\}$, $N_0(c, d) = \{n \mid n \in N(c), d(n, d) = d(c, d)\}$, and $N_{+1}(c, d) = \{n \mid n \in N(c), d(n, d) = d(c, d) + 1\}$ in $O(n)$ time. By forwarding the message to one of the elements in the first subset $N_{-1}(c, d)$, fully adaptive minimal routing is attained.

Let us assume that the network based on the crossed cube contains some faulty vertices and/or edges. Then, we can utilize the classification of the neighboring vertices of the current vertex $c$ for message forwarding. That is, to forward the message from $c$ to one of its neighboring vertices, we can select it among the three subsets with the priority $N_{-1}(c, d)$, $N_0(c, d)$, and $N_{+1}(c, d)$ in this order so that the path will be shorter. To evaluate this approach is one of the future works.
Acknowledgements

This study was partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under Grant No. 17K00093.

References


