

Routing Permutations in the Hypercube^{*}

Olivier Baudon¹, Guillaume Fertin¹, and Ivan Havel^{2**}

¹ LaBRI U.M.R. C.N.R.S. 5800, Université Bordeaux I
351 Cours de la Libération, F33405 Talence Cedex
`{fertin,baudon}@labri.u-bordeaux.fr`

² Faculty of Mathematics and Physics, Charles University
Malostranské nám. 25, 118 00 Praha, Czech Republic
`haveli@math.cas.cz`

Abstract. We study an n -dimensional directed symmetric hypercube H_n , in which every pair of adjacent vertices is connected by two arcs of opposite directions. Using the computer, we show that for H_4 and for any permutation on its vertices, there exists a system of pairwise arc-disjoint directed paths from each vertex to its target in the permutation. This gives the answer to Szymanski's conjecture [Szy89] for dimension 4. In addition to this study, we consider in H_n the so-called *2-1 routing requests*, that is routing requests where any vertex of H_n can be used twice as a source, but only once as a target. We give two such routing requests which cannot be routed in H_3 . Moreover, we show that for any dimension $n \geq 3$, it is possible to find a 2-1 routing request g_n such that g_n cannot be routed in H_n : in other words, for any $n \geq 3$, H_n is not (2-1)-rearrangeable.

Keywords : Hypercubes, routing permutations, Szymanski's conjecture, 2-1 routing requests.

1 Introduction

The directed symmetric hypercube H_n of dimension $n \geq 1$ has the set of vertices V_n with $|V_n| = 2^n$ and the set of arcs A_n with $|A_n| = n2^n$. From several possible equivalent definitions we are choosing the following : V_n consists of all the integers i such that $0 \leq i \leq 2^n - 1$ and for $i, j \in V_n$, (i, j) is an arc of H_n iff the binary representation of i and j differ in only one coordinate. If this coordinate is in ν -th position in the binary string, we will say that (i, j) is in dimension ν . For every ν , there are 2^n arcs in dimension ν in H_n . Observe that H_n is symmetric, i.e. for each $i, j \in V_n$, $(i, j) \in A_n$ iff $(j, i) \in A_n$.

We also define below a $h - k$ routing request, definition that can be found in [GT97].

^{*} This work has been supported by the Barrande Cooperative Research Grant #97137

^{**} Supported by Grant #201/98/1451 of GACR

Definition 1.1 (Routing Request). A routing request on a directed graph G is a multi-set R of ordered pairs of vertices of G . For each pair $\{s, t\}$ in a routing request, s is called the source and t is called the target of the pair. A routing request R is said to be $h - k$ if each vertex appears in R at most h times as a source and at most k times as a target. A routing request on G is called a partial permutation if it is 1-1, and a permutation if it is 1-1 and has exactly $|V(G)|$ pairs.

Szymanski [Szy89] considered the following problem : given a hypercube H_n and a permutation R on the vertices of H_n , is it possible to realize these source-target pairs by arc-disjoint paths ? That is, is it possible to find for each $\{s_i, t_i\} \in R$ a directed path P_i from s_i to t_i such that the paths P_i are pairwise arc-disjoint ? Or, in the terminology of interconnection networks : is the hypercube *rearrangeable* ? Szymanski [Szy89] conjectured that the answer is yes for any $n \geq 1$; he proved it for any $n \leq 3$, with the stronger property that all the requests are satisfied with shortest paths. In the following, we will refer to Szymanski's conjecture with the shortest paths property as the *strong Szymanski's conjecture*.

Lubiw [Lub90] gave a counterexample of the strong Szymanski's conjecture, concerning the shortest paths assumption. She gave an example of a permutation in H_5 which cannot be realized by arc-disjoint and shortest paths. Moreover, Darnet [Dar92] also gave a counterexample of the strong Szymanski's conjecture concerning the shortest paths, but in dimension 4. It is presented in Fig. 1, which gives a 1-1 routing request π on the vertices of H_4 . Note that π is a partial permutation, but, still, π cannot be routed by arc-disjoint and shortest paths. Any permutation realizing at least the requests from π would fail to be routed by arc-disjoint and shortest paths as well.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(x)$	10	2	14	8	3	15	11						9	1	13	

Fig. 1. A 1-1 routing request π in H_4 which cannot be routed with shortest paths

Note that all the source-target pairs which are given in the table are such that $dist(x, \pi(x)) = 2$ in H_4 . This is the base of the proof : suppose we want to find a shortest path for the source-target pair $\{6, 15\}$. Then we can either route via 14, or via 7. Suppose we route via 14, that is $6 \rightarrow 15 : 6, 14, 15$. Hence the only solution to route the pair $\{2, 14\}$ is $2 \rightarrow 14 : 2, 10, 14$, which means that $0 \rightarrow 10 : 0, 8, 10$, hence $4 \rightarrow 8 : 4, 12, 8$, and consequently $12 \rightarrow 9 : 12, 13, 9$. In that case, the arcs $(14, 15)$ and $(12, 13)$ have both been used. Hence it is impossible to route from 14 to 13 with a shortest path in an arc-disjoint fashion. Similarly, if we decide to route from 6 to 15 via 7, we end up with a contradiction of the same sort.

Note that we will see in Theorem 2.1 that all the permutations in H_4 that realize at least π can be routed with non shortest paths.

Since we have managed to find a counterexample in dimension 4 concerning the strong Szymanski's conjecture, it is easy to see that for any dimension

$n \geq 4$, we can find a permutation for which it is not possible to route with shortest paths, since in any hypercube H_n with $n \geq 4$, H_4 is a subgraph of H_n . For this, we take any subgraph isomorphic to H_4 in H_n and we apply π on this subgraph ; then we “complete” π to get a permutation in H_4 .

However, until now, it has not been proved whether Szymanski’s conjecture (i.e. the rearrangeability of H_n , with $n \geq 4$, without the shortest paths condition) holds. Note that many authors [GT97,CS93,ST94,BR90] have solved part of the problem, whether by proving that some families of permutations could be routed in any H_n , or by proving that by doubling some of the arcs in one or several dimensions in the hypercube, any permutation could be routed in H_n . For a good survey of the results concerning the latter, we refer to [GT97].

In this paper, we first show in Section 2 that Szymanski’s conjecture holds for $n = 4$; for this, we use a computer program. We will see that this proof implies the use of 2-1 routing requests. Hence, in Section 3, we will focus on the (2-1)-rearrangeability of H_n , that is the rearrangeability of H_n with respect to the 2-1 routing requests. In H_3 , we show that at least two such (2-1) routing requests cannot be routed by arc-disjoint paths ; this can be generalized, and we show in Section 3.2 that for any $n \geq 3$, H_n is not (2-1)-rearrangeable.

2 On the rearrangeability of H_4

Here, we prove Szymanski’s conjecture for $n = 4$. This is done using a computer program, whose steps are detailed below. First, we observe that deleting all the arcs in dimension i ($0 \leq i \leq n - 1$) in H_n results in a disconnected graph, each of the two connected components being a copy of H_{n-1} . Hence, we will call a *cut* in dimension i ($0 \leq i \leq n - 1$) the deletion of all the directed edges of H_n in dimension i . The two copies of H_{n-1} that we get that way are called *subcubes*. For example, the *cut in dimension 0* of H_4 gives us two subcubes of dimension 3, say $H_{3,0}$ and $H_{3,1}$, where $H_{3,0}$ (resp. $H_{3,1}$) is the subgraph of H_4 induced by the vertices p with $0 \leq p \leq 7$ (resp. with $8 \leq p \leq 15$).

H_4 has 16 vertices, and therefore there are $16!$ permutations on the vertices of H_4 . Hence, thanks to a brute force method, we could have tried to route each and every permutation on the vertices of H_4 . However, a deeper study of H_3 will save us many unnecessary computations.

2.1 Converting the problem from H_4 to H_3

The main idea here is to wonder whether any 2-1 routing request can be routed with arc-disjoint paths in H_3 . Indeed, if we manage to prove this, then it is not difficult to see that any permutation can be routed by arc-disjoint paths in H_4 : suppose we have a permutation π in H_4 , and let us cut H_4 in dimension 0. Let $V_{3,0} = \{v \in V_4 \mid 0 \leq v \leq 7\}$ and $V_{3,1} = \{v \in V_4 \mid 8 \leq v \leq 15\}$. Now let us route π in two steps. The first step is the following : for each $v \in V_{3,0}$ (resp. $v \in V_{3,1}$) , if $\pi(v) \in V_{3,1}$ (resp. $\pi(v) \in V_{3,0}$) then route v to $(v + 8)$ (resp. to

$(v - 8)$) using the arc $(v, v + 8)$ (resp. $(v, v - 8)$) first. For all the pairs $\{v, \pi(v)\}$ such that v and $\pi(v)$ are in the same $V_{3,i}$ ($0 \leq i \leq 1$), do nothing. This first routing step clearly is pairwise arc-disjoint. Now, look at the subgraphs of H_4 induced respectively by $V_{3,0}$ and $V_{3,1}$: each of them is a hypercube of dimension 3, and π induces on each of them a 2-1 routing request by our method. Hence, if any 2-1 routing request can be routed by arc-disjoint paths in H_3 , then H_4 is rearrangeable.

To know whether any 2-1 routing request on H_3 can be routed by arc-disjoint paths, we use the computer. For a better understanding, we give an overview of the algorithm used : first, for each request $\{s_i, t_i\}$, we are allowed certain paths depending on the distance from s_i to t_i in H_3 . These paths are the following.

- if $dist(s_i, t_i)=1$, we are only allowed the shortest path, that is the arc (s_i, t_i) ;
- if $dist(s_i, t_i)=2$, we can use the 2 shortest paths or the 6 paths of length 4 ;
- if $dist(s_i, t_i)=3$, we can use the 6 shortest paths or the 6 paths of length 5.

Note that, for each request, we order the possible paths by priority (in that case, the shortest paths will be placed first, then the non shortest ones).

The algorithm is the following : for each request, take the allowed path with higher priority. If at least one arc of this path is already used by a previous request, then try the second path, etc., till one path is such that no arc has been used before. If it is not possible, then backtrack to the previous request, and do the same thing recursively till we can find a path P with no arc already used. In that case, use the path P , and try to route the next request. If no path P is found, the routing request is said to be failing. If a path is found for each of the requests, then the given 2-1 routing request can be routed in H_3 by arc-disjoint paths.

Thanks to the computer, we are able to show that most of the 2-1 routings on H_3 can be routed by arc-disjoint paths. In fact, only 72 of them did not get through our algorithm. Let us call them the *72 failing routing requests*. Thanks to the numerous automorphisms of H_3 , we can show that only two of them are non equivalent : these are the 2-1 routing requests f_3 and g_3 defined in Section 3.1.

Our algorithm does not try each and every possible path for a given request. However, we show in Section 3.1 that f_3 and g_3 *cannot* be routed in H_3 .

2.2 Getting back to H_4

Now let us consider one of those 72 2-1 failing routing requests, say ρ . The aim is to consider ρ as the “projection” on H_3 of a permutation π of the vertices of H_4 and to retrieve all the possible permutations π corresponding. Depending on which of the 4 dimensions we decide to cut H_4 , and, having done so, on which of the 2 subcubes of dimension 3 we consider, there are $4 \cdot 2 = 8$ possibilities. Once we have decided this, we have to “rebuild” π from the informations given by ρ . In each ρ among the 72 failing routing requests, three of the eight vertices are used twice as a source ; hence, only five distinct vertices are used as sources. Consider the following example (Figure 2), where we consider ρ in the subgraph $H_{3,0}$ induced by a cut in dimension 0.

x	0	1	2	3	4	5	6	7
$\rho(x)$			5	4	3	6	1	
			7		2		0	

Fig. 2. A 2-1 routing request ρ in H_3

In that case, we see that 2 is taken twice as a source. Hence the corresponding permutations in H_4 will either have $\{2, 5\}$ and $\{10, 7\}$, or $\{2, 7\}$ and $\{10, 5\}$, as source-target pairs. The same goes for a vertex which is only taken once as a source. Take, for instance, vertex 3. The corresponding permutations in H_4 either could have $\{3, 4\}$ as a source-target pair, or $\{11, 4\}$. As we have five vertices which are sources at least once, this gives us $2^5 = 32$ possible different sets of 8 requests in H_4 . This fixes only 8 requests ; hence, there are $8! = 40320$ possibilities for the 8 remaining requests in H_4 .

Consequently, for each of the 8 considered subcubes of dimension 3, and for each 2-1 routing request ρ in this subcube, we need to test $32 \cdot 40320 = 1290240$ permutations π in H_4 . Thanks to the computer, it is very easy and fast to verify that those permutations in H_4 can be routed by arc-disjoint paths. Indeed, suppose we have cut H_4 in dimension 0, and that we are looking at $H_{3,0}$, i.e. the hypercube of dimension 3 induced by the vertices $0 \leq p \leq 7$. In that case, for each of the 72 2-1 failing routing requests ρ , we have to test the rearrangeability of H_4 on the permutations $\pi_{i,\rho}$ ($1 \leq i \leq 32 \cdot 40320$). For a given $\pi_{i,\rho}$, let us cut H_4 in a different dimension (say 1) and see, in each of the two subcubes induced by the cutting, if the 2-1 routing requests induced by this cut is among the 72 failing ones. If this is the case, let us try by cutting in another dimension (say 2), etc.

It appears that, for each of the 72 2-1 routing requests ρ , no $\pi_{i,\rho}$ is such that, by cutting H_4 in one of the three other dimensions, the new 2-1 routing requests given in each of the two subcubes are among the 72 failing ones. Consequently, if a permutation π is such that a cut in dimension $0 \leq d \leq 3$ induces one of the 72 failing routing requests in at least one of its two subcubes of dimension 3, then there exists $0 \leq d' \neq d \leq 3$ such that a cut in dimension d' does not imply this situation. Hence the following theorem.

Theorem 2.1. *Any permutation π on the vertices of H_4 can be routed by arc-disjoint paths, that is H_4 is rearrangeable.*

3 2-1 routing requests in H_n

In Section 3.1, we study two examples of 2-1 routing requests, f_3 and g_3 (cf. Section 2.1), and prove that they cannot be routed in H_3 with arc-disjoint paths. Starting from g_3 , we show in Section 3.2 a recursively constructed 2-1 routing request on H_n , for which no arc-disjoint routing can be found ; this proves Theorem 3.1.

3.1 H_3 is not (2-1)-rearrangeable

We have seen in Section 2.1 that among the 72 failing 2-1 routing requests, only two of them are non equivalent by automorphism of H_3 . We denote those two routing requests f_3 and g_3 , which are as follows :

x	0	1	2	3	4	5	6	7	x	0	1	2	3	4	5	6	7
$f_3(x)$	3		5	4		0	1		$g_3(x)$			5	4	7	2		0
	6		7		2							6		3			1

Fig. 3. f_3 (left) and g_3 (right)

We are going to show that none of them can be routed in H_3 with arc-disjoint paths. First, we introduce some auxiliary notions and notational conventions. We call an arc (u, v) of H_n a d -arc (downwards going arc) if $u > v$. Since we consider H_n , we have : (u, v) is a d -arc iff $u - v = 2^i$, with $0 \leq i \leq n - 1$. An arc (u, v) is called a u -arc (upwards going arc) if it is not a d -arc. Note that if H_n is drawn using a "level" representation in such a way that 0 is the lowest and $2^n - 1$ the highest vertex in the drawing, then u -arcs are really directed upwards and d -arcs downwards (cf. for instance Fig. 4) ; in that case, we say that we *arrange* H_n with respect to the pair $(2^n - 1, 0)$.

Assume $s, t \in V_n$. Then obviously all shortest directed paths from s to t in H_n use the same number of d -arcs (resp. u -arcs) ; let us denote it $d(s, t)$ (resp. $u(s, t)$). Hence, for a routing request $R = \{s_1, t_1\}, \dots, \{s_r, t_r\}$ in H_n , we define $d(R)$ and $u(R)$ as follows : $d(R) = \sum_{i=1}^r d(s_i, t_i)$ and $u(R) = \sum_{i=1}^r u(s_i, t_i)$. Observe that in R , identical ordered pairs are allowed and also that any directed path from s to t uses at least $d(s, t)$ d -arcs and $u(s, t)$ u -arcs. Moreover, for any directed path from s to t , the difference between the number of d -arcs and u -arcs, that is $d(s, t) - u(s, t)$, remains constant. Finally we define, for $v \in V_n$: $v^{in} = \{(u, v); u \in V_n \text{ and } (u, v) \in A_n\}$ and $v^{out} = \{(v, u); u \in V_n \text{ and } (v, u) \in A_n\}$. Now we are ready to prove the following.

Proposition 3.1. *Neither f_3 nor g_3 can be routed by arc-disjoint paths in H_3 .*

Proof. We have $V_3 = \{0, 1, \dots, 7\}$ and $A_3 = \{(0, 1), (1, 0), \dots, (7, 6)\}$ with $|A_3| = 24$. We also verify easily that $d(f_3) = 9$, $u(f_3) = 11$, $d(g_3) = 12$ and $u(g_3) = 8$.

We first assume that there is a routing by arc-disjoint paths in H_3 for f_3 , let us denote it ρ . Analyzing the sources and targets of f_3 we conclude that ρ must use the arcs $(0, 4)$, $(1, 5)$, $(2, 6)$ and $(3, 7)$, i.e. all the arcs leading from the subcube induced in H_3 by the vertices $\{0, 1, 2, 3\}$ to the subcube induced by the vertices $\{4, 5, 6, 7\}$. (The reason is that 4 pairs of f_3 have their sources in the first subcube and targets in the second one.) It follows quite similarly that ρ must also use the arcs $(0, 1)$, $(2, 3)$, $(4, 5)$ and $(6, 7)$. Further, we conclude that ρ uses not more than 2 arcs from each of the following sets: 1^{out} , 4^{out} and 5^{in} (since, e.g. 1 is a target but not a source, hence exactly one path from ρ ends in 1 and no path from ρ begins there). Since ρ uses $(1, 5)$ and $(4, 5)$, it does not use $(7, 5)$.

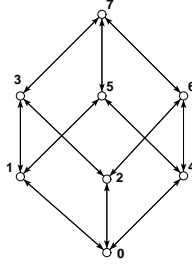


Fig. 4. The hypercube H_3 drawn in “level representation”

Let us look at the set 1^{out} : at least one of its arcs is not used by ρ ; since we showed above that $(1, 5)$ must be used, there are 2 cases to be considered:

1. $(1, 3)$ is not used by ρ : then all the remaining 11 u -arcs have to be used by ρ and ρ is necessarily a shortest path routing. It follows that $0 \rightarrow 3 : 0, 2, 3$, $0 \rightarrow 6 : 0, 4, 6$, $2 \rightarrow 7 : 2, 6, 7$, $2 \rightarrow 5 : 2, 0, 1, 5$. Now consider $6 \rightarrow 1$: it can start neither with $(6, 7)$ (used already) nor with $(6, 2)$ (there is no way out from 2), hence $6 \rightarrow 1 : 6, 4, 5, 1$; finally, $5 \rightarrow 0 : 5, 4, 0$. This is a contradiction, since all the 3 arcs from 4^{in} are already used and therefore $3 \rightarrow 4$ cannot be done.
2. $(1, 0)$ is not used by ρ : because of symmetry and the fact that at most 2 arcs from 4^{out} may be used by ρ , we conclude that $(4, 0)$ is not used by ρ either. Since $d(f_3) = 9$ and the d -arcs $(1, 0)$, $(4, 0)$, $(7, 5)$ are not used by ρ , ρ has to be a shortest path routing. This is a contradiction, because, obviously, $5 \rightarrow 0$ cannot be routed by a shortest path. This contradiction completes the analysis of the second case and we are done with f_3 .

Now we consider function g_3 : arrange H_3 with respect to the pair $(7, 0)$. We have seen that $d(g_3) = 12$ and $u(g_3) = 8$. But there are 12 d -arcs altogether, hence all the d -arcs must be used and ρ consists of shortest paths. Consider the subset $S = \{3, 5, 7\}$ of V_3 . Observe that, in ρ , there are 5 paths with targets in $V_3 \setminus S$, which start in S . On the other hand, there are 5 arcs going from a vertex in S to a vertex in $V_3 \setminus S$. We conclude that $5 \rightarrow 3 : 5, 7, 3$. Further, ρ uses not more than 2 arcs from 6^{out} , since it is a target and not a source in g_3 . However, no d -arc can be unused, because $d(g_3) = 12$; hence ρ does not use $(6, 7)$, and we conclude that $4 \rightarrow 7$ cannot be managed by a shortest path. This contradiction accomplishes the whole proof. \square

3.2 2-1 routing requests in H_n

In this Section, we are going to define recursively a 2-1 routing request g_n in the hypercube of dimension n , H_n , and show in Theorem 3.1 that for any $n \geq 3$, g_n cannot be routed in H_n with arc-disjoint paths. This shows that for any $n \geq 3$, H_n is not (2-1)-rearrangeable. The idea here is to find an “equivalent” of function g_3 of Section 3.1 for any dimension $n \geq 3$, and to generalize the arguments that

lead us to show that no routing could achieve g_3 with arc-disjoint paths.

First, let us define recursively such a 2-1 routing request, g_n .

Definition 3.1 (2-1 routing request g_n). Let g_3 be the 2-1 routing request defined for H_3 in Section 3.1. For any $n \geq 3$, let g_{n+1} be defined as follows. For any $i \in [0; 2^n - 1]$, if $g_n(i) \in \emptyset$, then $g_{n+1}(i) \in \emptyset$; otherwise :

- $g_{n+1}(i) = g_n(i) + 2^n$;
- $g_{n+1}(i + 2^n) = g_n(i)$.

Note that for any $n \geq 3$, any vertex of V_n is used exactly once as a target.

In order to show that g_n cannot be routed by arc-disjoint paths in H_n for any $n \geq 3$, we first show that if a routing ρ_n could achieve this, then it must be by shortest paths. For this, we arrange H_n with respect to the pair $(2^n - 1, 0)$, and we show by induction that $d(g_n)$, the minimum number of d -arcs used to route g_n , satisfies : $d(g_n) = n \cdot 2^{n-1}$. We have seen that this is true for $n = 3$, since $d(g_3) = 12$. Now suppose it is true for a fixed n , and let us show that it holds for $n + 1$. For this, we refer to the recursive definition of g_{n+1} , and we discuss the number of d -arcs ; for any $i \in [0; 2^n - 1]$, we have :

- $g_{n+1}(i) = g_n(i) + 2^n$. This means that, for any $i \rightarrow g_n(i)$ in g_n , we have $i \rightarrow g_n(i) + 2^n$ in g_{n+1} . This means that for any request of this type in g_{n+1} , we will need at least 1 more u -arc, but as many d -arc as in g_n .
- $g_{n+1}(i + 2^n) = g_n(i)$. This means that, for any $i \rightarrow g_n(i)$ in g_n , we have $i + 2^n \rightarrow g_n(i)$ in g_{n+1} . This means that for any request of this type in g_{n+1} , we will need at least 1 more d -arc, and as many d -arc as in g_n . Since we know that there are 2^n targets in g_n , this means that at least 2^n more d -arcs are needed to route g_{n+1} .

Consequently, we see that $d(g_{n+1}) = d(g_n) + (d(g_n) + 2^n)$, that is $d(g_{n+1}) = (n + 1) \cdot 2^n$. This proves, by induction, that for any $n \geq 3$, $d(g_n) = n \cdot 2^{n-1}$, and this shows that any routing ρ_n which satisfies the 2-1 routing request g_n will be of shortest paths, since we have exactly $n \cdot 2^n$ d -arcs.

Now let us define in H_n the set S_n of vertices as follows : S_n is the set of vertices which have 2 targets in g_n . Note that from now on, we will denote the vertices $u \in [0; 2^n - 1]$ of H_n by their binary representation $B(u)$. More precisely, any vertex $u \in [0; 2^n - 1]$ in H_n will be noted as follows $B(u) = mx_2x_1x_0$, where m consists of the $(n-3)$ leading bits of $B(u)$, and x_i is the bit of weight i in $B(u)$. An arc from u to v , (u, v) , will then be denoted by $(B(u), B(v))$. Thanks to this representation, we will be able to prove Properties 3.1 and 3.2, which will help us to prove the main result, i.e. Theorem 3.1.

Property 3.1. For any $n \geq 3$ and for any m a $(n-3)$ -bits string, every arc of the form $(m101, m111)$ is used in ρ_n to route a particular request $s \rightarrow t$, where $s, t \in S_n$.

Proof. We first characterize the vertices of S_n , thanks to their binary representation. Indeed, we can show by induction that $S_n = \{m011, m101, m111 \mid$

$m \in \{0, 1\}^{n-3}$. This is true by definition for $n = 3$ (i.e., $S_3 = \{011, 101, 111\} = \{3, 5, 7\}$); moreover, since by definition of g_{n+1} , only the vertices $0m011$, $0m101$, $0m111$ and $1m011$, $1m101$ and $1m111$ have two targets by g_{n+1} , we get the result.

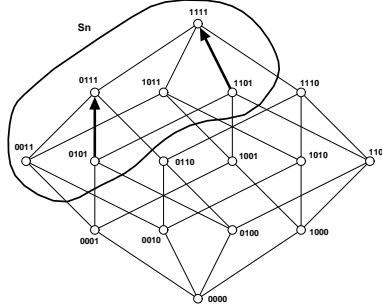


Fig. 5. Property 3.1 illustrated in H_4

The aim now is to show that every routing request of the form $s \rightarrow t$ with $s, t \in S_n$ must take place within S_n , that is no directed path P used to route such a request could be such that one of its vertices $u \notin S_n$. For this, let us first count the number of arcs going from a vertex in S_n to a vertex in $V_n \setminus S_n$, and let us then count the number of requests $s \rightarrow t$ with $s \in S_n$ and $t \in V_n \setminus S_n$.

We see that there are exactly $5 \cdot 2^{n-3}$ arcs going from a vertex of S_n to a vertex of $V_n \setminus S_n$. This is straightforward since, for a vertex $u \in S_n$, any change of a single bit among the $n-3$ leading ones will give another vertex v of S_n . Hence, the only way to get an arc from a vertex $u \in S_n$ to a vertex $v \notin S_n$ is to change one of the last 3 bits. This leads to 9 possibilities for any fixed m (3 possibilities for each vertex $m011, m101$ and $m111$), but we see that $(m011, m111)$, $(m101, m111)$, $(m111, m011)$ and $(m111, m101)$ are arcs whose both end vertices are in S_n . Hence, for a fixed m , there are 5 arcs going from a vertex $u \in S_n$ to a vertex $v \notin S_n$. Since m is a $(n-3)$ -bits string, we have on the whole $5 \cdot 2^{n-3}$ such arcs.

Now, let us count the number of requests of the form $s \rightarrow t$ with $s, t \in S_n$. We show the following by induction : every request $s \rightarrow t$ with $s, t \in S_n$ is for every $s = m101$ and $t = \bar{m}011$, where m is a $(n-3)$ -bits string and \bar{m} is the string obtained from m by changing every 0-bit (resp. 1-bit) into a 1-bit (resp. a 0-bit). This is true for $n = 3$, since the only such request is $5 \rightarrow 3$, that is $101 \rightarrow 011$. If this is true for a fixed n , then it is easy to see, thanks to Definition 3.1, that this is true for $n + 1$ too (that is $g_{n+1}(0m101) = 1\bar{m}011$ and $g_{n+1}(1m101) = 0\bar{m}011$ for every m a $(n-3)$ -bits string). Hence we see that the number of requests of the form $s \rightarrow t$, with $s, t \in S_n$ is equal to 2^{n-3} . Since $|S_n| = 3 \cdot 2^{n-3}$ and since each vertex $u \in S_n$ has two targets, this means that we need at least $5 \cdot 2^{n-3}$ directed paths going from vertices of S_n to vertices of $V_n \setminus S_n$. However, we have exactly $5 \cdot 2^{n-3}$ arcs going from a vertex in S_n to a vertex in $V_n \setminus S_n$. From

this, we conclude that every path P used to route a request of the form $s \rightarrow t$ with $s, t \in S_n$ remains in S_n , that is no intermediate vertex in P can be in $V_n \setminus S_n$.

Starting from this point, let us show the property. For any request of the form $s \rightarrow t$ with $s, t \in S_n$ (that is $s = m101 \rightarrow t = \overline{m}011$), since we need to route by shortest paths, we only have two options. They are as follows :

$$m101 \rightarrow \dots \rightarrow \underline{p101} \rightarrow p001 \rightarrow \dots \rightarrow q001 \rightarrow q011 \rightarrow \dots \rightarrow \overline{m}011 \text{ or}$$

$$m101 \rightarrow \dots \rightarrow p'101 \rightarrow p'111 \rightarrow \dots \rightarrow q'111 \rightarrow q'011 \rightarrow \dots \rightarrow \overline{m}011$$

where p, q, p' and q' are some $(n - 3)$ -bits strings.

However, the first option cannot occur : indeed, the underlined step implies the use of an arc going from vertex $p101 \in S_n$ to vertex $p001 \notin S_n$, which contradicts the above arguments. Consequently, only the second option is valid, which shows that the arc $(p'101, p'111)$ is used to route the request $m101 \rightarrow \overline{m}011$ for any fixed m . Since there are 2^{n-3} such requests, since we use arc-disjoint paths and since there are exactly 2^{n-3} different possibilities for p' , we conclude that every arc of the form $(m101, m111)$, for any m a $(n - 3)$ -bits string, is used for a particular routing request $s \rightarrow t$, with $s, t \in S_n$. This proves the property. \square

Property 3.2. For any $n \geq 3$, every arc of the form $(m110, m111)$ is necessarily unused in ρ_n , for every m a $(n - 3)$ -bits string.

Proof. It is possible to prove by induction that, for every $n \geq 3$, every vertex of the form $m110$ (for m a $(n - 3)$ -bits string) has no target in g_n . This is true by definition for any $n = 3$, and if this is true for a fixed n , we see that it is still true for $n + 1$ since $g_{n+1}(0m110)$ and $g_{n+1}(1m110)$ are defined thanks to $g_n(m110)$, which belongs to the empty set.

Since no vertex of the form $m110$ has an image by g_n , we know that there is at least one arc from $m110^{out}$ which is unused by ρ_n . The aim here is to show that this unused arc is necessarily $(m110, m111)$. For this, let us detail all the possible cases for the $(n - 3)$ -bits string m , with decreasing $|m|_1$, where $|m|_1$ is the number of 1-bits in m .

If $|m|_1 = n - 3$, that is $m = 1111 \dots 111$, we know we need to use all the d -arcs (because $d(g_n) = n2^{n-1}$ for any $n \geq 3$) ; thus the only possible unused arc is the only existing u -arc going out of $m110$, that is $(m110, m111)$. Now let $|m|_1 = n - 4$. Then m has exactly one bit equal to 0. Since the number of bits equal to 0 correspond to the number of u -arcs going out of $m110$, we have two options to choose the unused arc in ρ_n : either it is $(m110, m'110)$ (where $m' = 111 \dots 111$), or it is $(m110, m111)$. But if we suppose $(m110, m'110)$ unused, this means that there is one more arc from $m'110^{out}$ which is unused by ρ_n . But we have seen that there could only be one unused arc going out of $m'110$, otherwise a d -arc would be unused, which is not possible. Hence the unused arc must be $(m110, m111)$. This argument is illustrated in Fig. 6, where $n = 4$. The

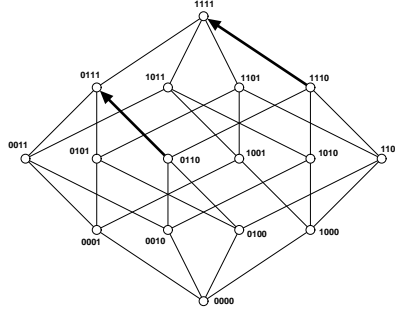


Fig. 6. Property 3.2 illustrated in H_4

same argument goes on for every m a $(n-3)$ -bits string, with decreasing $|m|_1$: indeed, step by step, we see that any m with $|m|_1 = p$ has $n-p-2$ u -arcs. These arcs are either arcs of the form $(m110, m'110)$ (where m' is obtained from m by changing exactly one 0-bit into a 1-bit), or the arc $(m110, m111)$. But since $|m'|_1 = p+1$, this case has already been considered before, and we have seen that no u -arc other than $(m'110, m'111)$ can be unused (otherwise, this would mean that a new u -arc from $m'110^{out}$ is unused by ρ_n , a contradiction). Hence the only u -arc which can be unused by ρ_n is $(m110, m111)$.

Consequently, for any m a $(n-3)$ -bits string, the arc $(m110, m111)$ is unused by ρ_n . \square

Theorem 3.1. g_n cannot be routed in H_n , for any $n \geq 3$, and thus H_n is not $(2-1)$ -rearrangeable for any $n \geq 3$.

Proof. It is not difficult to see that, for any $n \geq 3$, the request $4 \rightarrow 2^n - 1$ always exists in g_n . Indeed, we see that $4 \rightarrow 7$ exists in H_3 . Moreover, if we have $4 \rightarrow 2^n - 1$ in H_n for a fixed n , then the request $4 \rightarrow 2^{n+1} - 1$ exists in H_{n+1} , since by definition $g_{n+1}(4) = g_n(4) + 2^n$, that is $g_{n+1}(4) = 2^n - 1 + 2^n = 2^{n+1} - 1$ for any $n \geq 3$.

Now let us show that the request $4 \rightarrow 2^n - 1$ cannot be routed in H_n for any $n \geq 3$. For this, let us consider the binary representation of the source and target in this request. In that case $B(4) = 000\dots00100$, and $B(2^n - 1) = 11111\dots11111$. Let $m_0 = 000\dots0000$ and $\overline{m_0} = 111\dots1111$ be such that $|\overline{m_0}|_1 = n - 3$. In that case, $4 \rightarrow 2^n - 1$ becomes $m_0100 \rightarrow \overline{m_0}111$. Since we know we need to use arc-disjoint and shortest paths, there are only two routing schemes for this request. They are as follows :

$$m_0100 \rightarrow \dots \rightarrow p_0100 \rightarrow p_0110 \rightarrow \dots \rightarrow \underline{q_0110} \rightarrow \underline{q_0111} \rightarrow \dots \rightarrow \overline{m_0}111 \text{ or}$$

$$m_0100 \rightarrow \dots \rightarrow p'_0100 \rightarrow p'_0101 \rightarrow \dots \rightarrow \underline{q'_0101} \rightarrow \underline{q'_0111} \rightarrow \dots \rightarrow \overline{m_0}111$$

where p_0, q_0, p'_0 and q'_0 are some $(n-3)$ -bits strings.

However, in each of those two cases, there is one step which cannot be achieved, which we have underlined. Indeed, in the first case, we know by Property 3.1 that any arc $(m101, m111)$ is already used to route a request of the form $s \rightarrow t$, with $s, t \in S_n$, for every m a $(n-3)$ -bits string. Since $4 \notin S_n$ for all $n \geq 3$, we conclude that we cannot use the first scheme. Similarly, Property 3.2 yields that any arc of the form $(m110, m111)$ cannot be used for any routing request, for every m a $(n-3)$ -bits string. This shows that the request $4 \rightarrow 2^n - 1$ of g_n cannot be routed in H_n for any $n \geq 3$, which proves the theorem. \square

Remark 3.1. Let f_4 be the 2-1 routing request on H_4 obtained by applying to f_3 the same operation which gave g_4 from g_3 . We note that f_4 can be routed in H_4 with arc-disjoint paths.

4 Conclusion

In the first part of this paper, we use the computer to prove the rearrangeability of H_4 . This proof relies on the “splitting” of any permutation in H_4 into two 2-1 routing requests (one in each of the H_3 obtained by cutting H_4 in dimension 0), which we try to route by arc-disjoint paths in their respective subcube H_3 . We note that this method is the one employed by Szymanski [Szy89] to prove the rearrangeability of H_3 .

Starting from what was a study of 1-1 routing requests in H_n and the rearrangeability of H_n , we have mainly studied and proved the non-rearrangeability of H_n with respect to 2-1 routing requests. Though we have answered the question of the (2-1)-rearrangeability of H_n , the question of the (1-1)-rearrangeability of H_n for any $n \geq 5$ remains open.

References

- [BR90] R. Boppana and C.S. Raghavendra. Optimal self-routing of linear-complement permutations in hypercubes. In *DISTMEMCC: 5th Distributed Memory Computing Conference*, pages 800–808. IEEE Computer Society Press, 1990.
- [CS93] S.B. Choi and A.K. Somani. Rearrangeable circuit-switched hypercube architectures for routing permutations. *Journal of Parallel and Distributed Computing*, 19:125–130, 1993.
- [Dar92] A. Darmet. Private communication, 1992.
- [GT97] Q-P Gu and H. Tamaki. Routing a permutation in the hypercube by two sets of edge disjoint paths. *Journal of Parallel and Distributed Computing*, 44(2):147–152, 1997.
- [Lub90] A. Lubiw. Counterexample to a conjecture of Szymanski on hypercube routing. *Information Processing Letters*, 35:57–61, 1990.
- [ST94] A.P. Sprague and H. Tamaki. Routing for involutions of a hypercube. *Discrete Applied Mathematics*, 48:175–186, 1994.
- [Szy89] T. Szymanski. On the permutation capability of a circuit-switched hypercube. In *Proc. Internat. Conf. on Parallel Processing*, pages I–103 – I–110, 1989.
- [ZS90] K. Zemoudeh and A. Sengupta. Routing frequently used bijections on hypercube. In *DISTMEMCC: 5th Distributed Memory Computing Conference*, pages 824–832. IEEE Computer Society Press, 1990.