UNIVERSITÀ CA' FOSCARI DI VENEZIA
Dipartimento di Informatica

Technical Report Series in Computer Science

Technical Report CS-99-21

26 October 1999

R. Focardi and F. L. Luccio

Minimum Feedback Vertex Set in k-Dimensional Hypercubes

Dipartimento di Informatica, Università Ca' Foscari di Venezia
Via Torino, 155 – 30172 – Mestre-Venezia, Italy

Minimum Feedback Vertex Set in k-Dimensional Hypercubes *

Riccardo Focardi
Dipartimento di Informatica,
Università Ca' Foscari, Venezia, Italy
focardi@dsi.unive.it

Flaminia L. Luccio
Dipartimento di Scienze Matematiche,
Università degli Studi di Trieste, Italy
luccio@mathsun1.univ.trieste.it

26 October 1999

Abstract

In this paper we find upper and lower bounds to the size of the feedback vertex set for k-dimensional hypercubes. Given a graph, the minimum feedback vertex set problem consists of finding a subset of vertices of minimum size whose removal induces an acyclic subgraph. The problem is $\mathcal{NP}$-hard for general networks [2], but interesting polynomial solutions have been found for particular graphs [5, 6, 7, 8, 9, 12, 14].

keywords: feedback vertex set, bounds, hypercube, combinatorial problems, algorithms.

1 Introduction

The feedback vertex set of a graph $G = (V, E)$ is a subset of vertices $\overline{V} \subseteq V$ whose removal from G, induces an acyclic subgraph $G' = (V', E')$ where $V' = V \setminus \overline{V}$ and $E' = \{(x, y) \in E : x \in V' \text{ and } y \in V'\}$ (i.e., G' is the restriction of G to V'). If the cardinality of $\overline{V}$ is the minimum possible, the set is called a *minimum feedback vertex set* for G.

The problem is $\mathcal{NP}$-hard for general networks [2], although there exist polynomial time algorithms for particular topologies, e.g., reducible flow graphs [12], cocomparability graphs and convex bipartite graphs [6], cyclically reducible graphs [14], and other [5, 7, 8, 9, 13].

The problem has been widely studied since it has interesting applications. As an example, consider the typical operating system problem of allocating resources to processors while preventing deadlocks. It can be solved by considering the graph where vertices represent processors and each edge (i, j) represents the request of processor i for a resource already allocated to a processor j. Now, if the graph contains a cycle then a deadlock has occurred and therefore processors will indefinitely wait for the requested resources. The problem can be solved by moving into a waiting queue the minimum number of processors that makes the graph acyclic (and so deadlock free) [14].

Actually the first application of the problem was in the area of combinatorial circuit design. In order to let the system stabilize before new inputs are received, registers are inserted in each cycle of the system. This obviously increases the delay in the circuit speed, therefore the number of such registers have to be minimized [4].

Other applications of the problem are e.g., in artificial intelligence for the constraint satisfaction problem and for the reduction of the computational complexity of Bayesian inference [1], and in synchronous distributed systems [9] for the study of "monopolies" ([10, 11]).

In this paper, we consider the problem for particular undirected graphs of degree k that are widely studied in literature, namely k-dimensional hypercubes. For this topology, we find close lower and upper bounds to the size of the feedback vertex set.

*Research partially supported by MURST.

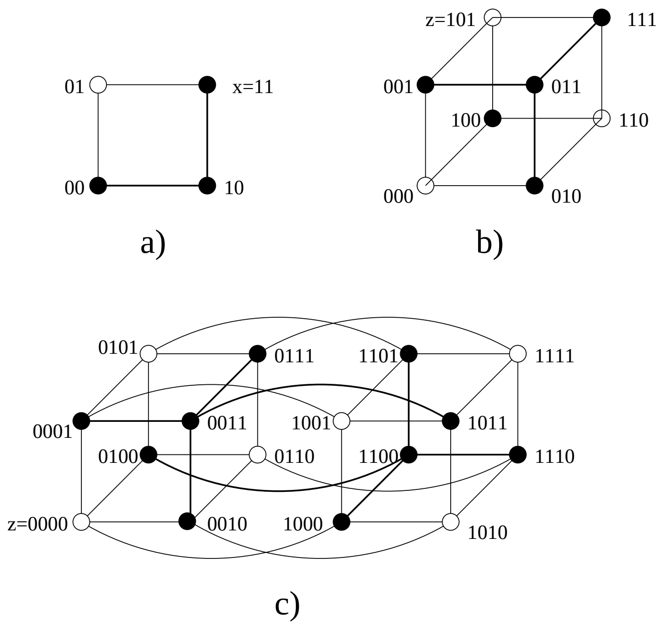

Figure 1: A representation of H_2, H_3, H_4, respectively in figure a, b, c. White vertices will be removed. The thicker line in every H_k represents the new acyclic graph G.

2 Feedback Vertex Set in k-Dimensional Hypercubes

In this section we find upper and lower bounds to the size of the feedback vertex set for k-dimensional hypercubes.

We first recall some definitions:

Definition 1 *A (binary)* k-dimensional hypercube *is a graph $H_k = (V, E)$ with 2^k vertices labelled with k distinct bits. A vertex $v_i \in V$, $0 \leq i \leq 2^k - 1$ is connected to the k vertices whose labels differ exactly for one bit from the label of v_i.*

Examples of 2, 3 and 4-Dimensional Hypercubes are shown in figures 1.a, 1.b and 1.c, respectively.

Definition 2 *Given two binary strings x and y of length k, the* Hamming distance *between x and y, written $HD(x, y)$, is the number of bits in which the two strings differ.*

In the following, we will denote with $\overline{V}$ the feedback vertex set.

A lower bound to the size of $\overline{V}$ is given by the following theorem that states a general solution related to the number of components of the resulting acyclic subgraph.

Theorem 1 *For $H_k = (V, E)$ we have $|\overline{V}| \geq \frac{(k-2)2^{k-1}+c}{(k-1)}$, where c is the number of distinct components in which the new acyclic graph $G' = (V', E')$, with $V' = V \setminus \overline{V}$, is decomposed.*

Proof To prove this theorem we need the following result:

Lemma 1 *Given a hypercube $H_k = (V, E)$, the number of edges that can remain in any acyclic graph $G' = (V', E')$ of c distinct components, obtained from H_k is:*

$$|E'| \leq \frac{k(2^{k-1} - c)}{(k-1)}.$$

PROOF We call c_i, with $1 \leq i \leq c$, the c acyclic components in G. We denote with a_i ($a_i \geq 0$) the number of edges of component c_i. Of course, we have that $\sum_{i=1}^{c} a_i = |E'|$. Since every component is a tree, we also have that c_i has exactly $a_i + 1$ vertices.

Now, the number of all the out-going edges from the vertices of a component c_i inside c_i is clearly $2a_i$. As a matter of fact, every edge is out-going from two of the component vertices. We also have that the number of total out-going edges (included the edges outside the components, i.e., the removed edges) is $k(a_i + 1)$, since every vertex has exactly k out-going edges in the hypercube.

So, we can calculate the number of removed edges, for c_i, as follows:

$$k(a_i + 1) - 2a_i = (k - 2)a_i + k$$

Observe that every edge inside the component has been considered twice, but twice has also been considered in the removed edges. Moreover the removed edges are not in common neither inside each component (if an edge has the two vertices in the same component then it is also in the component), nor between two components (otherwise the two distinct components would be merged into a single one). Thus, the total amount of removed edges is at least:

$$\sum_{i=1}^{c} [(k-2)a_i + k] = (k-2)(\sum_{i=1}^{c} a_i) + ck = (k-2)|E'| + ck$$

We know that in a k-dimensional hypercube the number of edges is $k2^{k-1}$. Hence, the number of removed edges plus the edges inside the components must be at most $k2^{k-1}$. Formally:

$$(k-2)|E'| + ck + |E'| = (k-1)|E'| + ck \leq k2^{k-1}$$

and so:

$$|E'| \leq \frac{k(2^{k-1} - c)}{(k-1)}$$

which proves the lemma ■

The proof of the theorem is now based on lemma 1. By a trivial argument we can argue that the number of vertices $|R|$ that remain in G is at most the number of remaining edges, plus 1, on each component c, therefore: $|\overline{V}| \geq n - |R| \geq 2^k - \sum_{i=1}^{c} (a_i + 1) = 2^k - c - |E'| \geq 2^k - c - \frac{k(2^{k-1}-c)}{(k-1)} = \frac{k2^k - 2^k - k2^{k-1} + c}{k-1} = \frac{(k-2)2^{k-1}+c}{(k-1)}$ ■

The bound given by this theorem can be easily applied to the case where $c = 1$, obtaining the following corollary. Indeed, we are presently trying to apply more efficiently the result above, in order to obtain tighter results.

Corollary 1 *For H_k we have $|\overline{V}| \geq 2^{k-1} - \frac{2^{k-1}-1}{k-1}$.*

Observe that a very close bound could also be obtained in the following simple way. First observe that trivially a graph $G = (V, E)$ with n vertices has a cycle if $|E| \geq n$. Moreover H_k has $n = 2^k$ vertices and $k\,2^{k-1}$ edges (each vertex has k out-going edges and H_k is an undirected graph). This implies that no matter which are the edges removed from the graph in order to obtain an acyclic subgraph, at least $k\,2^{k-1} - 2^k + 1 = 2^{k-1}(k-1) + 1$ edges have to be removed. This implies that in the worst case k removed edges are in common with a single vertex, i.e., at least $\frac{2^{k-1}(k-1)+1}{k} = 2^{k-1} - \frac{2^{k-1}-1}{k}$ vertices have to be removed.

We now give an upper bound to the problem. The algorithm is based on constructive steps. For H_2 and H_3 see figure 1. Consider now $k > 3$. In this algorithm A, R and T are sets; R represents vertices removed from G and T the ones taken; x and v are vertex labels.

ALGORITHM 1

/* **STEP 1** */
$R, T, A := \emptyset$;
/* Take all alternating vertices */
$A := V$;
Extract vertex $v = 00\ldots0$ from A and insert it in a set R;
while $A \neq \emptyset$ **do**
    x : random extraction from A;
    if $HD(x, v)$ is odd **then** insert x in T
        else in R;

/* **STEP 2** */
extract $v = 00\ldots0$ from R and insert it in T;
$A := R$;
$R := \emptyset$;
while $A \neq \emptyset$ **do**
    $x :=$ random extraction from A;
    subdivide x into blocks of consecutive 0's and 1's
    if (k is even **and** each block has an even number of bits **and**
      the total number of 1's is divisible by 4)
      or (k is odd **and** x ends with a 0
      and each block has an even number of bits but the last one
      and the total number of 1's is divisible by 4)
        then insert x in T
        else in R

/* **STEP 3** */
Build the subgraph $H'_k = (V', E')$ of H_k,
    with $V' := T$
    and $E' = \{(x, y) \in E : x \in V' \text{ and } y \in V'\}$.

Theorem 2 *The set $\overline{V} = V \setminus V'$ derived from algorithm 1 is a feedback vertex set for H_k, with $|\overline{V}| \leq 2^{k-1} - 2^{\lfloor \frac{k}{2} \rfloor - 1}$.*

Proof We first prove that algorithm 1 correctly defines a vertex feedback set for H_k.

In Step 1 all vertices with Hamming distance odd from vertex $v = 00\ldots0$ are inserted in a set T (taken), the others in R (removed). We now prove the following two things: 1) all vertices in T (and all vertices in R) have even Hamming distance; 2) $|T| = |R| = 2^{k-1}$. Let us first prove 1). W.l.o.g. consider two vertices x and y in R. By construction $h_1 = HD(v, x)$ and $h_2 = HD(v, y)$ are both even. Compare x and y and set to 0 all the j bits which are equal to 1 and which are in the same position in x and y, obtaining two new strings x' and y' (such that where one has a 1, the other has a 0 or vice versa). x' has $h_1 - j$ 1's and y' $h_2 - j$. Since the 1's are placed in different positions $HD(x', y') = h_1 - j + h_2 - j = h_1 + h_2 - 2j$ which is an even number. Then, trivially, $HD(x, y)$ is also even. A similar proof holds for the set T.

Let us now prove 2). To prove that $|T| = |R| = 2^{k-1}$ first observe that by construction $T \cap R = \emptyset$ (vertices are exclusively inserted in a set or in the other). Now build a bijective function f. f is such that for every $x = x_1 \ldots x_k \in T$ $f(x_1 x_2 \ldots x_k) = \bar{x_1} x_2 \ldots x_k$, i.e., f complements the first bit of x. If $x \in T$ it has an even number of 0's, therefore $f(x)$ has an odd number, i.e., $f(x) \in R$ and f is an function $f : R \rightarrow T$. Moreover f is injective. If there exists a $z \in T$ such that $f(x) = y = f(z)$ then $x = x_1 \ldots x_k$ and $y = y_1 \ldots y_k$ and $z = z_1 \ldots z_k$ but $\bar{x_1} = y_1 = \bar{z_1}$ and $x_2 \ldots x_k = y_2 \ldots y_k = z_2 \ldots z_k$

therefore $x = z$. With the same argument we can prove that f is surjective, therefore f is bijective. This implies that $|T| = |R|$ and since $T \cap R = \emptyset$ and $T \bigcup R = V$ (since all vertices have been considered) then $|T| + |R| = 2^k$, i.e., $|T| = |R| = 2^{k-1}$.

1) and 2) imply that after step 1) $V = T \bigcup R$ and $|T| = |R|$. Moreover all vertices in T have Hamming distance at least 2. This implies that the resulting partial subgraph is acyclic, has 2^{k-1} vertices and the same number of components.

In Step 2) other vertices are added to T. We want to prove that these new vertices do not form any cycle. We add to T all vertices whose blocks of consecutive 0's and 1's in the label have an even number of bits (but eventually the last block for k odd) and the total number of 1's is divisible by 4. These vertices are at distance 1 from k vertices in T so they could potentially form a cycle. We now prove that all such vertices are at distance at least 4 from each others. This will trivially imply that every vertex does not have another vertex at distance 2. Therefore the resulting subgraph will not have cycles since every hypercube has cycles of length at least 4. Assume this is not true, i.e., a cycle of length 3 exists. Since by definition 1 every neighbouring vertex differs in a single bit a vertex a distance 2 has at least 2 different bits and therefore all its neighbours either 3 or 1 from the initial one. I.e., no cycle of length 3 can exist. The initial vertex that is added, i.e., $v = 00\ldots0$, cannot trivially form a cycle since it does not have vertices at distance 2 in T. We finally prove that all vertices inserted in T in Step 2) have Hamming distance at least 4.

Build the following table. Starting from two strings 00 and 11 that have Hamming distance 2, build 2 new strings of length 2 by changing the last bit of 00 and 11, obtaining 01 and 10. Since 00 and 11 differ in 2 bits and the last bit of both of them has been changed also 01 and 10 differ in two bits from each other but they differ in 1 bit from 00 and 11. Therefore add a bit 0 to 00 and 11 and a 1 to 10 and 01. The new strings 000, 110, 101, 011 differ all in 2 bits. Repeat the same procedure up to when the strings have all length $\lfloor \frac{k}{2} \rfloor$. I.e., at a generic step i consider the actual set of strings, build a set with the same string with the last bit complemented, add a final 0 to the string in the first set and a 1 to the others.

We now prove by induction that at step i in the table there are 2^i distinct strings that have Hamming distance at least 2 among them, and have an even number of 1's. To prove this observe that at step 1 there are 2 distinct strings, 00 and 11 that have Hamming distance 2 and have an even number of 1's. Assume that at step i there are 2^i distinct strings that have Hamming distance at least 2 and have an even number of 1's. The new strings are obtained by complementing the last bit of the others. Since old strings differ in at least 2 bits, the new strings cannot be equal to the old ones (differ in at least 1 bit) and trivially differ from each other in at least 2 bits. If a 0 is added to the last bit of the old strings and a 1 to the new ones distinct strings that differ in at least 2 bits are obtained. The number of these strings is then $2\,2^i = 2^{i+1}$. Moreover, since the parity is changed twice (when changing the last bit and when adding a 1 at the end) all the new strings have an even number of 1's.

Apply this procedure up to step $\lfloor \frac{k}{2} \rfloor - 1$. At the end there are $2^{\lfloor \frac{k}{2} \rfloor - 1}$ strings with Hamming distance at least 2 and with an even number of 1's. Replace in these strings all 0's with 00 and all 1's with 11 and add a 0 to all the strings if k is odd. The resulting strings have length k, trivially have Hamming distance 4, their number of 1's proportional to 4 and have even blocks of 0's and 1's (but eventually the last one for k odd). It is also easy to see, by construction, that in this way we obtain all the strings of this form.

The number of such strings is then $2^{\lfloor \frac{k}{2} \rfloor - 1}$ a total of $2^{k-1} - 2^{\lfloor \frac{k}{2} \rfloor - 1}$ vertices will be removed. ■

We finally give the following:

Corollary 2 *The bounds of theorem 1 and 2 match if we put $c = 2^{k-1} - (k-1)\,2^{\lfloor \frac{k}{2} \rfloor - 1}$ in theorem 1. In fact this is the value of c of the graph built by algorithm 1.*

Proof Observe that in algorithm 1 after Step 1 there are 2^{k-1} components. In Step 2, $2^{\lfloor \frac{k}{2} \rfloor - 1}$ are added, each of which is connected to k distinct components, that are "absorbed" by the new components. Hence the components increase by $2^{\lfloor \frac{k}{2} \rfloor - 1}$ and decrease by $k2^{\lfloor \frac{k}{2} \rfloor - 1}$. Therefore the total number decreases to

$c = 2^{k-1} - k\,2^{\lfloor \frac{k}{2} \rfloor -1} + 2^{\lfloor \frac{k}{2} \rfloor -1} = 2^{k-1} - 2^{\lfloor \frac{k}{2} \rfloor -1}(k-1)$. Observe that the bound given in theorem 2 is $|\overline{V}| \geq \frac{(k-2)2^{k-1}+c}{(k-1)}$, therefore substituting c we obtain $|\overline{V}| \geq \frac{(k-2)2^{k-1}+2^{k-1}-(k-1)\,2^{\lfloor \frac{k}{2} \rfloor -1}}{(k-1)} = 2^{k-1} - 2^{\lfloor \frac{k}{2} \rfloor -1}$. ■

References

[1] R.Bar-Yehuda and D. Geiger and J. Naor and R.M. Roth. Approximation Algorithms for the Feedback Vertex Set Problem with Applications to Constraint Satisfaction and Bayesian Inference. *SIAM Journal of Computing*, **27**, (4), 942-959, 1998.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco 1979.

[3] D.Z. Du and D.F. Hsu *Combinatorial Network Theory.* Kluwer Academic Publisher, Dordrecht, The Netherlands, 1996.

[4] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Science*, **9**, 256-278, 1974.

[5] Y.D. Liang. On the feedback vertex set in permutation graphs. *Information Processing Letters*, **52**, (3), 123-129, 1994.

[6] Y.D. Liang and M.S. Chang. Minimum Feedback Vertex Set in Cocomparability Graphs and Convex Bipartite Graphs. *Acta Informatica*, **34**, 337-346, 1997.

[7] E.L. Llyod and M.L. Soffa. On Locating Minimum Feedback Vertex Sets. *Journal of Computer and System Sciences*, **37**, 292-311, 1988.

[8] C. Lu and C. Tang. A linear-time algorithm for the weighted feedback vertex problem on interval graphs. *Information Processing Letters*, **61**, (2), 107-112, 1997.

[9] F.L. Luccio. Exact Minimum Feedback Vertex Set in Meshes and Butterflies. *Information Processing Letters*, **66**, (2), 59-64, 1998.

[10] D. Peleg. Local majority voting, small coalitions and controlling monopolies in graphs: A review. *Proc. 3rd Colloquium on Structural Information and Communication Complexity*, 152-169, 1996.

[11] D. Peleg. Size bounds for dynamic monopolies. *Proc. 4th Colloquium on Structural Information and Communication Complexity*, Carleton Univ. Press, 165-175, 1997.

[12] A. Shamir. A Linear Time Algorithm for Finding Minimum Cutsets in Reducible Graphs. *SIAM Journal of Computing*, **8**, (4), 645-655, 1979.

[13] G.W. Smith and R.B. Walford. The Identification of a Minimal Feedback Vertex Set of a Directed Graph. *IEEE Transaction on Circuits and Systems*, **CAS-22**, (1), 9-14, 1975.

[14] C. Wang and E.L. Lloyd and M.L. Soffa. Feedback Vertex Sets and Cyclically Reducible Graphs. *JACM* **32**, (2), 296-313, 1985.