OPTIMAL EMBEDDINGS OF THE FFT GRAPH IN THE HYPERCUBE

David S. Greenberg*
Lenwood S. Heath**
Arnold L. Rosenberg

COINS Technical Report 88-23

*Current address: Department of Computer Science,
Yale University, New Haven, CT 06520

**Current address: Department of Computer Science,
Virginia Polytechnic Institute, Blacksburg, VA 24061

OPTIMAL EMBEDDINGS OF THE FFT GRAPH IN THE HYPERCUBE

David S. Greenberg
Department of Computer Science
Yale University
New Haven, CT 06520

Lenwood S. Heath
Department of Computer Science
Virginia Polytechnic Institute
Blacksburg, VA 24061

Arnold L. Rosenberg
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

March 21, 1988

Abstract

We present two linear-time algorithms for embedding the n-level FFT graph, which has $(n+1)2^n$ vertices, in the $(n + \lceil \log(n+1) \rceil)$-dimensional Hypercube, with unit dilation. Thus, the FFT graph is a subgraph of the smallest Boolean Hypercube that is big enough to hold it. The simpler of our algorithms uses Gray codes to perform the embedding; the more complicated has the advantage of being modular. Either embedding yields a mapping of the FFT algorithm onto the Hypercube architecture, with unit (hence, optimal) dilation and optimal expansion.

1. INTRODUCTION

1.1. The Main Result and Motivation

The main result of this paper is:

> The FFT graph is a subgraph of the smallest Boolean Hypercube that is big enough to hold it.

We present two proofs of the result, each by means of a linear-time algorithm that embeds the n-level FFT graph $F(n)$, which has $(n+1)2^n$ vertices, in the[1] $(n + \lceil \log(n+1) \rceil)$-dimensional Hypercube $Q(n + \lceil \log(n+1) \rceil)$, with unit dilation[2]. The simpler of our algorithms uses Gray codes to perform the embedding; the more complicated has the advantage of being *modular*, in the sense that it obtains its optimal embedding of $F(n+1)$ in $Q(n+1+\lceil \log(n+2) \rceil)$ by extending its optimal embedding of $F(n)$ in $Q(n + \lceil \log(n+1) \rceil)$.

Motivating our study is the *logical mapping problem for parallel arrays of processors*, i.e. the problem of mapping a parallel algorithm onto a processor array: On the one hand, the algorithm has some natural subtask-interdependence structure, defined by either data dependencies or control dependencies; on the other hand, the array has a fixed processor-intercommunication network. The *mapping problem* is the problem of accommodating the algorithm's interdependence structure to the array's intercommunication structure. One typically studies the mapping problem by viewing both of the structures of interest as simple undirected graphs and viewing the mapping problem as one of finding an efficient embedding of the algorithm-graph in the array-graph [2, 3, 5, 6]. The major notions of the efficiency of an embedding are enunciated in [10]; they are the *dilation* of the embedding, which measures the maximum delay engendered by the accommodation, and the *expansion* of the embedding, which is one measure of the efficiency of utilizing the processors of the array.

In this paper, we study the mapping problem for the important *Fast Fourier Transform* (*FFT*) algorithm [1, Ch. 7], which is paradigmatic for convolution-based algorithms, in the popular (Boolean) Hypercube architecture [3, 5, 8], versions of which have been built by Intel, N-cube, BBN, and Thinking Machines. The embeddings that comprise our main results can be interpreted as mappings of the FFT algorithm onto the Hypercube architecture, with unit (hence, optimal) dilation and optimal expansion.

[1] All logarithms are to the base 2.

[2] Technical terms are defined in Section 1.2.

This mapping provides yet another example of the efficiency of the Hypercube as an interconnection structure, to supplement earlier work that has shown the Hypercube to be an efficient host for divide-and-conquer algorithms [3] and for grid-based algorithms [9]. A result superficially similar to ours appears in [7]; in that paper, it is shown that each single level of the FFT graph is a spanning subgraph of the Hypercube, whence one can run the FFT algorithm on the Hypercube as fast as one can run it on the FFT graph. Our result is materially harder than that of [7], in that we embed the entire FFT graph into the Hypercube at once.

1.2. The Formal Framework

The technical vehicle for our investigation is the following notion of graph embedding. Let G and H be simple undirected graphs, having $|G|$ vertices and $|H|$ vertices, respectively. An *embedding* of G in H is a one-to-one association of the vertices of G with the vertices of H. The *dilation* of the embedding is the maximum distance (in H) between vertices of H that are the images of adjacent vertices of G. The *expansion* of the embedding is the ratio $|H|/|G|$. Clearly, no embedding can have better than unit dilation, and such dilation is achievable only if G is a subgraph of H.

> The graph G represents our algorithm: the vertices of G represent the tasks of the algorithm; the edges of G represent the intertask dependencies. The graph H represents our processor array: the vertices of H represent the processors of the array; its edges represent the interprocessor communication links. Thus, the dilation of an embedding can be viewed as a measure of the delay incurred by executing "algorithm" G on "processor array" H, and the expansion of the embedding can be viewed as a measure of how efficiently G utilizes the processors of H.

Our specific focus here is on embeddings between two given finite families of graphs $\mathcal{G}$ and $\mathcal{H}$. We seek the best possible embeddings – relative to dilation – of each $G \in \mathcal{G}$ in the smallest $H \in \mathcal{H}$ that will hold it, i.e., for which $|H|/|G| \geq 1$. Thus, we optimize expansion-cost and then try to optimize dilation-cost. We are able here to achieve unit (hence, optimal) dilation, even while optimizing expansion.

The target graphs for our investigation are FFT graphs (which play the role of our G's) and Boolean hypercubes (which play the role of our H's).

- Let m be a positive integer. The 2^m-*input FFT graph* $F(m)$ (so named because it reflects the data-dependency structure of the 2^m-input FFT algorithm) is defined

as follows. $F(m)$ has vertex-set[3]

$$V_m = \{0, 1, \cdots, m\} \times \{0,1\}^m$$

For each vertex $v = \langle \ell, \vec{\delta} \rangle \in V_m$, we call ℓ the *level* of v and $\vec{\delta}$ the *position-within-level (PWL) string* of v. Vertices at level 0 of $F(m)$ are called *inputs*, and vertices at level m of $F(m)$ are called *outputs* (in deference to the algorithmic origins of the graph). The edges of $F(m)$ are of two types: For each $\ell \in \{0, \cdots, m-1\}$ and each $\delta_0\delta_1 \cdots \delta_{m-1} \in \{0,1\}^m$, the vertex

$\langle \ell, \delta_0\delta_1 \cdots \delta_{m-1} \rangle$, on level ℓ of $F(m)$,

is connected by a *straight-edge* with vertex

$\langle \ell + 1, \delta_0\delta_1 \cdots \delta_{m-1} \rangle$, on level $\ell + 1$ of $F(m)$;

and it is connected by a *cross-edge* with vertex[4]

$\langle \ell + 1, \delta_0\delta_1 \cdots \delta_{\ell-1}(\delta_\ell \oplus 1)\delta_{\ell+1} \cdots \delta_{m-1} \rangle$, on level $\ell + 1$ of $F(m)$.

(See Fig. 1.) It is often useful to view $F(m)$, $m \geq 2$ ($F(1) = K_{2,2}$ being given), as being constructed inductively, by taking two copies of $F(m-1)$, and 2^m new output vertices, and constructing *butterflies* (or, copies of the complete bipartite graph $K_{2,2}$) connecting the k^{th} outputs of each copy of $F(m-1)$, on the one side, to the k^{th} and $(k + 2^{m-1})^{\text{th}}$ new outputs, on the other side. Thus, $F(m)$ has $(m+1)2^m$ vertices and $m2^{m+1}$ edges.

- Let d be a nonnegative integer. The *d-dimensional Boolean Hypercube* $Q(d)$ is the graph whose vertices are all binary strings of length d and whose edges connect each string-vertex x with the d strings that differ from x in precisely one bit position. (See Fig. 2.) Thus, $Q(d)$ has 2^d vertices and $d2^{d-1}$ edges.

It is not hard to find a unit-dilation embedding of $F(m)$ in $Q(2m)$, by assigning two new dimensions for each level of $F(m)$; but this embedding has expansion $\Omega(N/\log N)$. Likewise, it is not hard to find a dilation-2 (expansion-optimal) embedding of $F(m)$ in $Q(m + \lceil \log(m+1) \rceil)$, which is the smallest Hypercube that holds $F(m)$, using embedding techniques analogous to those used in [5]. What we accomplish here is to optimize both cost measures simultaneously, and to do so via linear-time algorithms (which specify

[3] For any set S and positive integer k, we denote by S^k the set of all length-k strings of elements of S.

[4] $\oplus$ denotes addition modulo 2.

the vertex-mappings). In fact, we present two such algorithms: The first algorithm, which is based on Gray codes, is simple in both specification and verification; the second algorithm is a bit more complicated in both regards, but it has the advantage of being modular, in the sense mentioned in the Introduction. Stated formally, we prove

Theorem. *Every FFT graph is embeddable with unit dilation and optimal expansion in a Hypercube. Thus, for each m, $F(m)$ is a subgraph of $Q(d_m)$, where $d_m =_{\text{def}} m + \lceil \log(m+1) \rceil$; moreover, there is a linear-time algorithm that finds this optimal embedding.*

Both of the embeddings of $F(m)$ in $Q(d_m)$ that we use to prove the Theorem are specified via two labelling schemes:

- We assign each vertex v of $F(m)$ a unique d-bit label $L(v)$, which is its image vertex in $Q(d_m)$.
- We assign each edge (u, v) of $F(m)$ a *bit-position* label $B(u, v) \in \{1, 2, \cdots, d_m\}$ such that $L(u)$ and $L(v)$ differ exactly in bit-position $B(u, v)$.

We simplify our embeddings by using a single pair (s_i, c_i) of bit-positions, called a *bp-pair* (*bit-position pair*), for label assignments to edges between levels $i-1$ and i of $F(m)$, $1 \leq i \leq m$; in particular, all straight-edges between these levels flip[5] bit-position s_i, and all cross-edges between these levels flip bit-position c_i.

> Note that edge (u, v) of $F(m)$ is mapped by our embeddings onto the edge crossing dimension $B(u, v)$ of $Q(d_m)$, between vertex $L(u)$ and vertex $L(v)$ (whence the unit dilation of our embeddings). Note also that flipping bit-position b corresponds to crossing dimension b of $Q(d_m)$.

Thus, our embedding is specified by means of a *levelled bp-pair sequence* (*LBPS*, for short)

$$S = (s_1, c_1), (s_2, c_2), \ldots, (s_m, c_m).$$

The reader can verify easily that the LBPS gives us almost all the information we need to specify the embedding completely: When we assign a d-bit label $L(v)$ to any single vertex v of $F(m)$, the labels of all remaining vertices are completely determined by the LBPS. We can, and shall, therefore, specify our embeddings by labelling input vertex $v_0 =_{\text{def}} \langle 0, 00\cdots0 \rangle$ of $F(m)$ with the length-d_m string of 0's (thereby assigning it to vertex $00\cdots0$ of $Q(d_m)$ in the embedding) and using an appropriate LBPS to

[5] Edge (u, v) of $F(m)$ *flips* bit-position p if $L(u)$ and $L(v)$ differ precisely in bit-position p.

specify inductively the labelling of all remaining vertices. This strategy reduces the problem of specifying an embedding to the problem of specifying an LBPS $S(m)$ for each FFT graph $F(m)$; and, it reduces the problem of validating a given labelling – i.e., verifying that it actually specifies an embedding – to the problem of proving that the label-assignment is one-to-one. This last assertion (about the reduction) is true since any mapping produced by the strategy is *well-defined*, in the sense that the label inductively assigned to each vertex of $F(m)$ is independent of the order of inductively assigning labels. Well-definition is verified as follows.

Proposition 1 *Any mapping of the vertices of the FFT graph to the vertices of the Hypercube that is induced by an LBPS is well-defined.*

Proof. Assume that vertex v is assigned label $L(v)$ when the labelling is induced by the path P from vertex v_0 to v and that it is assigned label $L'(v)$ when the labelling is induced by the path P' from vertex v_0 to v. We shall show that $L(v) = L'(v)$.

We begin with three basic facts about cycles in $F(m)$.

1. For each level $\ell \in \{0, 1, \cdots, m\}$, the number of level-$\ell$ edges in any cycle in $F(m)$ is even.

2. For each level $\ell \in \{0, 1, \cdots, m\}$, the number of level-$\ell$ cross-edges in any cycle in $F(m)$ is even.

3. For each level $\ell \in \{0, 1, \cdots, m\}$, the number of level-$\ell$ straight-edges in any cycle in $F(m)$ is even.

The first fact follows since there is no "wraparound" in $F(m)$, so any cycle must re-cross levels. The second fact follows since every level-ℓ cross-edge flips bit-position ℓ of the current vertex's PWL string, and no straight-edge flips a bit; therefore, in order to regain a previous vertex, one must restore every flipped bit-position by re-crossing level ℓ via a cross-edge. The third fact follows from the first two via arithmetic.

Next, note that since we are dealing here only with labelling schemes that are induced by LBPS's, crossing a level of $F(m)$ twice using the same type of edge – a cross-edge or a straight-edge – flips the same bit-position of the Hypercube label twice, hence leaves the label unchanged.

Finally, consider the cycle formed by tracing path P from vertex v_0 to vertex v, followed by tracing the reverse of path P' from vertex v to vertex v_0. By the foregoing facts, each Hypercube dimension appears an even number of times around the cycle;

hence, the parity of the number of appearances of each dimension on P must be the same as the corresponding parity on P'. It follows that $L(v) = L'(v)$. □

The next two sections are devoted to specifying and validating our embeddings within this simplified framework.

2. A SIMPLE EMBEDDING

2.1. The Embedding

Our first embedding is constructed using Gray codes to define the LBPS. The n^{th} *Gray code sequence* $\mathcal{GC}_n$ is defined inductively as follows.

$$\begin{aligned} \mathcal{GC}_1 &= 0 \\ \mathcal{GC}_{n+1} &= \mathcal{GC}_n, n, \mathcal{GC}_n \end{aligned}$$

We denote by $\mathcal{GC}_n(i)$ the i^{th} element of $\mathcal{GC}_n$.

To specify the LBPS $S(m)$, and thereby the embedding of $F(m)$, let $\lambda = \lceil \log(m+1) \rceil$, and let 0^k denote a string of k 0's. Define the LBPS

$$S_1(m) = (s_1, c_1), (s_2, c_2), \ldots, (s_m, c_m)$$

as follows:

- $s_\ell = \ell + \lambda$
- $c_\ell = \mathcal{GC}_\lambda(\ell)$

for all $\ell \in \{1, 2, \cdots, m\}$. Note in particular that, since $\mathcal{GC}_\lambda$ uses no integer greater than $\lambda - 1$, the labels we assign to the cross-edges of $F(m)$ are disjoint from the labels we assign to the straight-edges.

2.2. Validation

Our proof that the LBPS $S_1(m)$ labels $F(m)$ injectively, hence specifies an embedding, depends on two easily verified facts. The first exposes a well-known property of Gray code sequences.

Lemma 1 *Every contiguous subsequence of $\mathcal{GC}_n$ contains at least one element an odd number of times.*

Proof Sketch. Every two occurrences of an integer k in the sequence $\mathcal{GC}_n$ are separated by an occurrence of some integer $\geq k+1$. It follows that the largest element in a subsequence of $\mathcal{GC}_n$ occurs with odd multiplicity. □

The second is a property of FFT graphs.

Lemma 2 *For all PWL strings $\vec{\gamma}, \vec{\delta} \in \{0,1\}^m$, there is a unique length-m path in $F(m)$ connecting vertex $\langle 0, \vec{\gamma} \rangle$ and vertex $\langle m, \vec{\delta} \rangle$.*

Proof. Level-ℓ edges of $F(m)$ are the only ones that affect bit-position ℓ of vertices' PWL strings. A length-m path that connects a vertex at level 0 with a vertex of level m traverses each level of $F(m)$ precisely once. Therefore, any path that connects the vertices in the Lemma must traverse the straight-edge between levels i and $i+1$ for every bit-position i in which the PWL strings $\vec{\gamma}$ and $\vec{\delta}$ agree, and it must traverse the cross-edge between levels i and $i+1$ for every bit-position i in which the PWL strings $\vec{\gamma}$ and $\vec{\delta}$ differ. □

Now, on to the proof of injectiveness of the labelling produced by $S_1(m)$.

Let us assume, for contradiction, that there are two vertices of $F(m)$, say $u = \langle \ell_u, \vec{\delta}_u \rangle$ and $v = \langle \ell_v, \vec{\delta}_v \rangle$ for which $L(u) = L(v)$. Without loss of generality, assume that $\ell_u \geq \ell_v$. Let $u' = \langle m, \vec{\delta}_u \rangle$ be the vertex of $F(m)$ that is in the *bottom* level of the same "column" as u, and let $v' = \langle 0, \vec{\delta}_v \rangle$ be the vertex of $F(m)$ that is in the *top* level of the same "column" as v. Consider the path P in $F(m)$ that starts at u, traverses straight-edges until it reaches u', thence follows the unique length-m path from u' to v' (cf. Lemma 2), and finally traverses straight-edges to end up at v. Let u'' and v'' be, respectively, the vertices at levels ℓ_u and ℓ_v along the subpath of P that connects u' and v'. See Fig. 3.

Since $L(u) = L(v)$, the path P must cross every Hypercube dimension – i.e., flip every bit-position – an even number of times. Consider what this means for the path P:

For each level $k \geq \ell_u$ and each level $k < \ell_v$, the path P traverses two edges connecting level k with level $k+1$ – and one of these edges is a straight-edge. It follows that the other edge must be a straight-edge also, since the LBPS $S_1(m)$ assigns each level-k straight-edge a bit-position that is shared by no cross-edge, and by no straight-edge at any other level of $F(m)$. Thus, if the second level-k edge of P were not also a straight-edge, the net effect of traversing the two edges would be to flip some bit-position of $L(u)$

that is flipped nowhere else, thereby preventing $L(v)$ from being identical to $L(u)$. We conclude that the subpath of P that connects u to u' must coincide with the subpath that connects u' to u'', which means that $u = u''$. By similar reasoning, $v = v''$.

For every remaining level k of $F(m)$, the path P traverses only one edge connecting level k to level $k+1$. If this edge were a straight-edge, then – as above – it would flip a unique bit-position, thereby preventing the coincidence of $L(u)$ and $L(v)$. Therefore, all the edges on these levels must be cross-edges. However, cross-edges on a consecutive sequence of levels of $F(m)$ flip bit-positions that are specified by a contiguous subsequence of a Gray code sequence. By Lemma 1, some element occurs an odd number of times in this subsequence; hence, some bit-position is flipped an odd number of times along the indicated portion of path P. This fact prevents the coincidence of $L(u)$ and $L(v)$.

Since the existence of conflicting vertices u and v guarantees the existence of an "even-flipping" path P, and we have just shown that no such path can exist, we conclude that the mapping induced by the LBPS $S_1(m)$ is injective. We thus have our first proof of the Theorem. □

3. A MODULAR EMBEDDING

3.1. The Embedding

Call an LBPS $S = (s_1, c_1), (s_2, c_2), \ldots, (s_m, c_m)$ *proper* if, for each i, at least one of s_i or c_i does not occur at an earlier level, i.e., in the set $\{s_j, c_j : j < i\}$; call such a virgin s_i or c_i *new*. We construct the LBPS $S_2(m)$ that specifies our second embedding by proceeding in stages, concentrating on ensuring propriety at every stage.

Partition the levels $\{0, 1, \cdots, m\}$ of $F(m)$ into *tiers*: tier k is the set of levels

$$\{i : 2^k \leq i \leq 2^{k+1} - 1\} \cap \{0, 1, \cdots, m\}.$$

Let the singleton $\{0\}$ constitute tier -1. If the level of vertex v in $F(m)$ is in tier k, then we say that vertex v *is in* tier k.

Our second embedding is incremental, hence modular, in that we obtain the LBPS $S_2(m+1)$ from the LBPS $S_2(m)$. Clearly, when we specify $S_2(m+1)$ and its induced labelling, a *new* bit-position can never lead to duplicated labels. We shall, therefore, always use any new bit-position as one becomes available (which happens when we must add a new dimension to the host Hypercube in order to accommodate the next

bigger FFT graph). As we proceed from $F(m)$ to $F(m+1)$, the number of dimensions in the smallest Hypercube that holds the FFT graph increases by at least one; hence, there is always at least one new bit-position to use in $S_2(m+1)$. Let us always use this new bit-position to label the straight-edge at level i, i.e., to be bit-position s_i. Whenever $m = 2^k$ is a power of 2, two new bit-positions are available for the expanded labelling when we proceed to $F(m+1)$. In this case, we call the pair (s_m, c_m) of new bit-positions *shield positions* for tier k of all $F(n)$, $n \geq m$. Given this strategy, we can specify explicitly

- $s_\ell = \ell + \lceil \log \ell \rceil$ for all $\ell \in \{1, 2, \cdots, m\}$;
- $c_{2^k} = 2^k + k + 1$ for all $k \in \{1, 2, \cdots, \lfloor \log m \rfloor\}$.[6]

We complete our specification of $S_2(m+1)$ by specifying the c_i's that are not shield positions. We proceed inductively, based on the tier number k, the case $k = 1$ being trivial. Having chosen the c_i's for tier $k-1$, we choose the $2^k - 1$ c_i's for tier k as follows.

- $c_{2^k+i} = \begin{cases} s_{2^{k-1}+i} & \text{if } 1 \leq i < 2^{k-1} \\ s_{2^{k-1}} & \text{if } i = 2^{k-1} \\ c_{2^{k-1}+i} & \text{if } 2^{k-1} < i < 2^k \end{cases}$

3.2. Validation

We simplify the task of verifying that $S_2(m)$ induces a unique label for each vertex of $F(m)$, by showing that we need consider $S_2(m)$'s behavior only on the m-level complete binary tree rooted at vertex v_0 (which is clearly a subgraph of $F(m)$).

For any vertex $v \in V_m$, let $T(v)$ be the complete binary tree rooted at v and extending monotonically downward (i.e., to increasing levels) so that the leaves of $T(v)$ are outputs of $F(m)$. If vertices $u, v \in V_m$ are at the same level of $F(m)$, then there is a unique isomorphism $\iota_{u,v} : T(u) \rightarrow T(v)$ that preserves the bit-positions assigned to edges.

Lemma 3 *Let S be a proper LBPS for $F(m)$. If the labelling L of V_m induced by S is injective on the vertices of $T(v_0)$, then L is injective on the vertices of $F(m)$.*

[6] To see the consistency with our specification of the s_i, note that $2^k + k + 1 = \ell + \lceil \log \ell \rceil + 1$ when $\ell = 2^k$.

Proof. Suppose, for contradiction, that the Lemma is false. Then the labelling L is injective on the vertices of the tree $T(v_0)$, but there are distinct vertices $u, v \in V_m$ for which $L(u) = L(v)$.

Note first that there is no input $z \in V_m$ of $F(m)$ such that both u and v reside in the tree $T(z)$. If there were such a z, then one sees easily that the vertices $\iota_{z,v_0}(u)$ and $\iota_{z,v_0}(v)$ of $T(v_0)$ must also be assigned the same label, since the labelling of $T(v_0)$ is dictated by the same LBPS S as is the labelling of $T(z)$. This would contradict the assumed uniqueness of labels in $T(v_0)$.

Let u reside at level ℓ_u in $F(m)$, and let v reside at level ℓ_v. Without loss of generality, say that $\ell_v \leq \ell_u$. Since u and v do not reside in the same $T(z)$, it follows that there exists an input z of $F(m)$ such that u is not in $T(z)$, but v is. Say we have chosen such a z.

Consider the graph $G(z,u) =_{\text{def}} T(z) \cup T(u)$. Choose a path P from u to v in $G(z,u)$, of the following form: P proceeds monotonically down $T(u)$ until a vertex p of $T(z)$ is reached; P then proceeds monotonically up $T(z)$ to an ancestor (not necessarily proper) of v; P finally goes down to v. The path P is guaranteed to exist, since the graph $G(z,u)$ is connected: the leaves of $T(z)$ comprise *all* of the outputs of $F(m)$ (z being an input of $F(m)$) while the leaves of $T(u)$ comprise some of the outputs of $F(m)$. Since $L(u) = L(v)$, it follows that every bit-position flips on edges of P an *even* number of times. Now, let $q \in V_m$ be the vertex that precedes p in P, and let $r \in V_m$ be the vertex that succeeds p in P; moreover, let k be the level of vertex p in $F(m)$ (easily, $\ell_v \leq \ell_u < k$). The edges (q,p) and (p,r) are the only edges in P that connect vertices on levels $k-1$ and k. Since these edges are distinct (r being in $T(z)$ while q is not) and share an endpoint, it follows that bit-position s_k flips on one of these edges, and bit-position c_k flips on the other. Since the LBPS S is proper, at least one of the bit-positions s_k, c_k must be new, hence flip only once in P, contradicting the even-flipping requirement. This contradiction establishes the Lemma. □

We complete our verification by showing that the labelling induced by the proper LBPS $S_2(m)$ is injective on $T(v_0)$.

Lemma 4 *The LBPS $S_2(m)$ is proper. Its induced vertex-labelling is injective on $T(v_0)$.*

Proof. $S_2(m)$ being proper by construction, we concentrate only on the injectiveness of its induced labelling. Assume, for the sake of contradiction, that the distinct vertices u and v of $T(v_0)$ are assigned the same label by the LBPS $S_2(m)$: $L(u) = L(v)$.

Claim. *u and v reside in the same tier.*

Let u reside at level ℓ_u of $T(v_0)$, and let v reside at level ℓ_v. Say that ℓ_u is in tier k; then ℓ_v must also be in tier k, since the shield bit-position that $L(u)$ has *on*[7] from tier k must also be *on* in $L(v)$. Without loss of generality, assume that ℓ_u (hence k) is smallest possible, and that $\ell_v \leq \ell_u$.

One verifies by inspection that the like-labelled vertices u and v cannot exist if $k < 2$; therefore, assume henceforth that $k \geq 2$.

Let $P(u)$ (resp., $P(v)$) be the path from v_0 to u (resp., to v) in $T(v_0)$. With no loss of generality, we may assume that the path $P(v)$ has a special form: Note that $P(u)$ (resp., $P(v)$) can be viewed as *choosing* either bit-position s_i or c_i at each level i, $1 \leq i \leq \ell_u$ (resp., $1 \leq i \leq \ell_v$). We say that a path *chooses the s-alternative* (resp., the *c-alternative*) at level i if it chooses bit-position s_i (resp., bit-position c_i). Say that, at a particular level i, we force path $P(u)$ (resp., path $P(v)$) to make the opposite choice of bit-position. Then the label of the vertex u' (resp., v') at level ℓ_u (resp., ℓ_v) that the new path leads to is $L(u') = L(u) \oplus 2^{s_i - 1} \oplus 2^{c_i - 1}$ (resp., $L(v') = L(v) \oplus 2^{s_i - 1} \oplus 2^{c_i - 1}$); hence, we obtain another pair of vertices with identical labels. It follows that by switching bit-position choices at precisely those levels i where $P(v)$ makes the choice c_i, and by making the corresponding switches in $P(u)$, we obtain two vertices $u+$ and $v+$ with identical labels and with the property that the path $P(v+)$, which was obtained by switching all c_i's to s_i's in $P(v)$, always chooses the s-alternative. Thus, when we look at the like-labelled vertices u and v, *we lose no generality by assuming that* $P(v)$ *chooses the s-alternative at every level.*

We can now determine enough about the labels $L(u)$ and $L(v)$ to complete the proof. Consider the choices that the paths $P(u)$ and $P(v)$ make as they traverse tier k. Since bit-positions $s_{2^k}, \ldots, s_{\ell_v}$ are new in tier k and are *on* in $L(v)$ (because $P(v)$ always chooses the s-alternative), they must be *on* in $L(u)$; hence, $P(u)$ makes the same choices as $P(v)$ at levels $2^k, \ldots, \ell_v$. Also, since bit-positions $s_{\ell_v+1}, \ldots, s_{\ell_u}$ are new in tier k and are *off* in $L(v)$ (since $P(v)$ terminates at v), path $P(u)$ must choose the c-alternative at levels $\ell_v + 1, \ldots, \ell_u$. We now know all the choices $P(u)$ and $P(v)$ make in tier k, so we turn our attention to tier $k-1$.

Our analysis breaks into two cases, depending on which *half-tier* of tier k the vertices u and v reside in. Say the *first half-tier* of tier k comprises levels $2^k, 2^k+1, \ldots, 2^k + 2^{k-1} - 1$, and the *second half-tier* comprises levels $2^k + 2^{k-1}, 2^k + 2^{k-1} + 1, \cdots, 2^{k+1} - 1$.

Claim. *u and v reside in the same half-tier.*

If this were not true, then, by assumption, v would reside in the first half-tier, while u would reside in the second half-tier. Now, bit-position $s_{2^{k-1}}$ is *on*, and bit-position

[7] Bit-position i is *on* in a label if it contains a "1"; otherwise it is *off*.

$c_{2^{k-1}}$ is *off* in $L(v)$, since $P(v)$ always chooses the s-alternative, which is always new; hence, the same configuration must appear in $L(u)$. However, we have seen that $P(u)$ must always choose the c-alternative at levels below ℓ_u, so in particular, $P(u)$ must choose bit-position $c_{2^k+2^{k-1}} = s_{2^{k-1}}$. But now consider the choice that $P(u)$ makes at level 2^{k-1}. If it chooses the c-alternative at that level, then since that bit-position is new there and has not recurred, $L(u)$ would have *both* bit-positions $s_{2^{k-1}}$ and $c_{2^{k-1}}$ *on*. Alternatively, if $P(u)$ chooses the s-alternative at that level, then this choice would "cancel" the choice of $c_{2^k+2^{k-1}} = s_{2^{k-1}}$, so $L(u)$ would have *both* bit-positions $s_{2^{k-1}}$ and $c_{2^{k-1}}$ *off*. Either contingency would contradict the assumption that $L(u) = L(v)$. We conclude that u and v are in the same half-tier.

We are now ready to complete the proof.

Claim. *u and v cannot be in the first half-tier.*

Suppose that u and v were both in the first half-tier. Since bit-positions $s_{\ell_u - 2^{k-1}+1}, \ldots, s_{2^{k-1}}$

- are new in tier $k-1$ (by definition),
- are *on* in $L(v)$ (since $P(v)$ always chooses the s-alternative),
- do not recur until levels $> \ell_u$,

these positions must be *on* in $L(u)$ also, so $P(u)$ must make the same choices as $P(v)$ at levels $\ell_u - 2^{k-1}+1, \ldots, 2^k - 1$. By construction of $S_2(m)$, bit-positions $c_{\ell_v+1}, \ldots, c_{\ell_u}$ are identical to bit-positions $s_{\ell_v-2^{k-1}+1}, \ldots, s_{\ell_u-2^{k-1}}$. These bit-positions are *on* in $L(v)$ by virtue of tier $k-1$ (where they are new); they must, therefore, be *on* in $L(u)$, since $L(u) = L(v)$. However, if they are *on* in $L(u)$, it must be by virtue of the c_i choices that we have already noted that $P(u)$ must make in tier k. It follows that $P(u)$ must choose the c-alternative at levels $\ell_v - 2^{k-1}+1, \ldots, \ell_u - 2^{k-1}$: if $P(u)$ were to choose the s-alternative at any of those levels, that choice would combine with the matching c_i choice in tier k to turn off a bit-position that is *on* in $L(v)$.

We now have a total picture of the choices made by paths $P(u)$ and $P(v)$ in tier k and in the bottom portion of tier $k-1$, when u and v reside in the first half-tier. Let the path $P'(u)$ be obtained from $P(u)$ by truncating the latter at level $\ell_u - 2^{k-1}$; let the path $P'(v)$ be obtained from $P(v)$ by truncating the latter at level $\ell_v - 2^{k-1}$. Let $L'(u)$ and $L'(v)$ be the vertex-labels obtained by following $P'(u)$ and $P'(v)$, respectively. Since this truncation removes the same bit-position settings from $P(u)$ and $P(v)$, one sees easily that $L'(u) = L'(v)$, even though the corresponding vertices reside within tier $k-1$ of $F(m)$. This contradicts the assumed minimality of k.

Claim. *u and v cannot be in the second half-tier.*

Finally, suppose that u and v are both in the second half-tier. By assumption, $P(v)$ chooses the s-alternative at levels $\ell_v - 2^k + 1, \ldots, 2^k - 1$. $P(u)$ must also make the same choices, since those bit-position settings in $L(u)$ must agree with $L(v)$; and we have seen that $P(u)$ does not choose those bit-positions when they recur (by definition of $S_2(m)$) in the first half-tier of tier k.

We now have a total picture of the choices made by paths $P(u)$ and $P(v)$ in tier k and in the bottom portion of tier $k-1$, when u and v reside in the second half-tier. Let the path $P'(v)$ be obtained from $P(v)$ by truncating the latter at level $\ell_v \quad 2^{k-1}$; let $P'(u)$ be obtained from $P(u)$ by truncating the latter at level $\ell_u \quad 2^{k-1}$ and by choosing the c-alternative at levels $\ell_v - 2^{k-1} + 1, \ldots, \ell_u \quad 2^{k-1}$. Let $L'(u)$ and $L'(v)$ be the vertex-labels obtained from following paths $P'(u)$ and $P'(v)$, respectively. One verifies as above that $L'(u) = L'(v)$ and that the corresponding vertices reside in tier $k-1$ of $F(m)$. Once again, we have contradicted the assumed minimality of k.

These contradictions establish the Lemma. □

We have verified that the LBPS $S_2(m)$ specifies an embedding of $F(m)$ in $Q(m+\lambda)$. The modularity of the embedding is obvious. □

4. FUTURE DIRECTIONS

Our results leave unresolved a number of significant questions.

1. Is every optimal embedding of the FFT graph into the Hypercube induced by an LBPS?

2. How efficiently can a Butterfly network be embedded in a Hypercube, where the m^{th} Butterfly network is obtained by identifying the input and output vertices of $F(m)$? The same question seems even harder if one substitutes certain Butterfly-derivatives for Butterflies, e.g., the deBruijn graph or (essentially equivalently) the Shuffle-Exchange graph.

3. How robust a subgraph of the Hypercube is $F(m)$? Specifically, we know from [3] and from easy generalizations of the results there that many "popular" networks can be embedded efficiently[8] in the Hypercube, e.g., binary trees, X-trees, and meshes. We know from [4] that complete binary trees can be embedded efficiently

[8] By "efficiently", we mean with simultaneous dilation $O(1)$ and expansion $O(1)$.

in the Butterfly, but that X-trees and meshes cannot be so embedded. We still do not know yet if arbitrary binary trees can be embedded efficiently in the Butterfly, nor if complete binary trees can be embedded efficiently in $F(m)$.[9]

ACKNOWLEDGMENT. The research of D. S. Greenberg was supported in part by NSF Grant MIP-86-01885. The research of L. S. Heath was supported in part by NSF Grant DCI-85-04308. The research of A. L. Rosenberg was supported in part by NSF Grants DCI-85-04308 and DCI-87-96236.

5. REFERENCES

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman (1974): *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA.

2. F. Berman and L. Snyder (1984): On mapping parallel algorithms into parallel architectures. *Intl. Conf. on Parallel Processing.*

3. S.N. Bhatt, F.R.K. Chung, F.T. Leighton, A.L. Rosenberg (1986): Optimal simulations of tree machines. *27th IEEE Symp. on Foundations of Computer Science,* 274-282.

4. S.N. Bhatt, F.R.K. Chung, J.-W. Hong, F.T. Leighton, A.L. Rosenberg (1988): Optimal simulations by Butterfly networks. *20th ACM Symp. on Theory of Computing,* to appear.

5. S.N. Bhatt and I. Ipsen (1985): Embedding trees in the hypercube. Yale Univ. Rpt. RR-443.

6. S.H. Bokhari (1981): On the mapping problem. *IEEE Trans. Comp., C-30.*

7. T.F. Chan (1986): On Gray code mapping for mesh-FFTs on binary N-cubes. Tech. Rpt. RIACS-86.17, NASA Ames Research Center.

8. T.C. Chen, M.D.F. Schlag, C.K. Wong (1983): The hypercube connection network. IBM Report RC-10219.

9. L. Johnsson (1985): Basic linear algebra computations on hypercube architectures. Tech. Rpt., Yale Univ.

[9] The embedding in [4] uses the wraparound feature of the Butterfly heavily.

10. A.L. Rosenberg (1981): Issues in the study of graph embeddings. In *Graph-Theoretic Concepts in Computer Science: Proceedings of the International Workshop WG80*, Bad Honnef, Germany (H. Noltemeier, ed.) *Lecture Notes in Computer Science 100*, Springer-Verlag, New York 150-176.

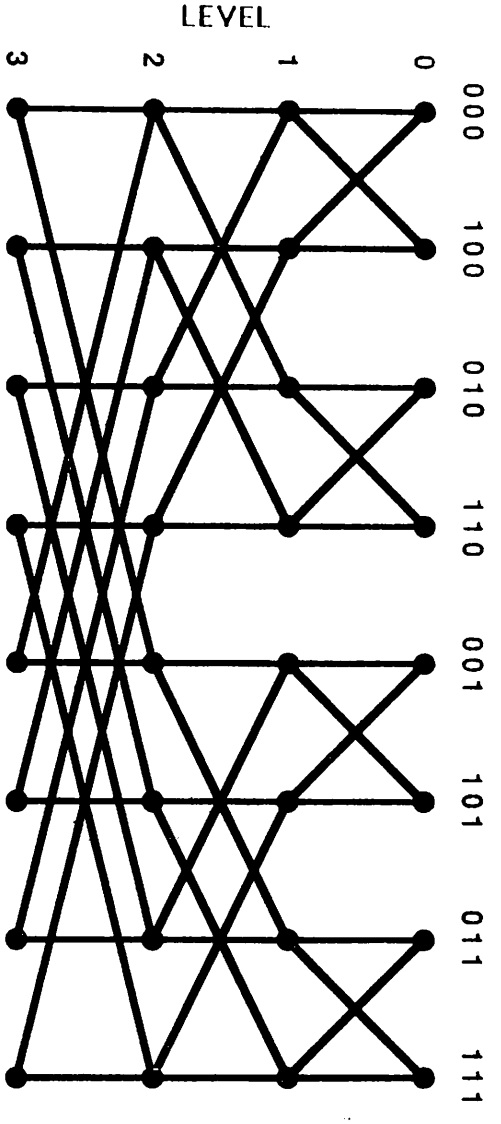

Figure 1: The 8-input FFT graph $F(3)$.

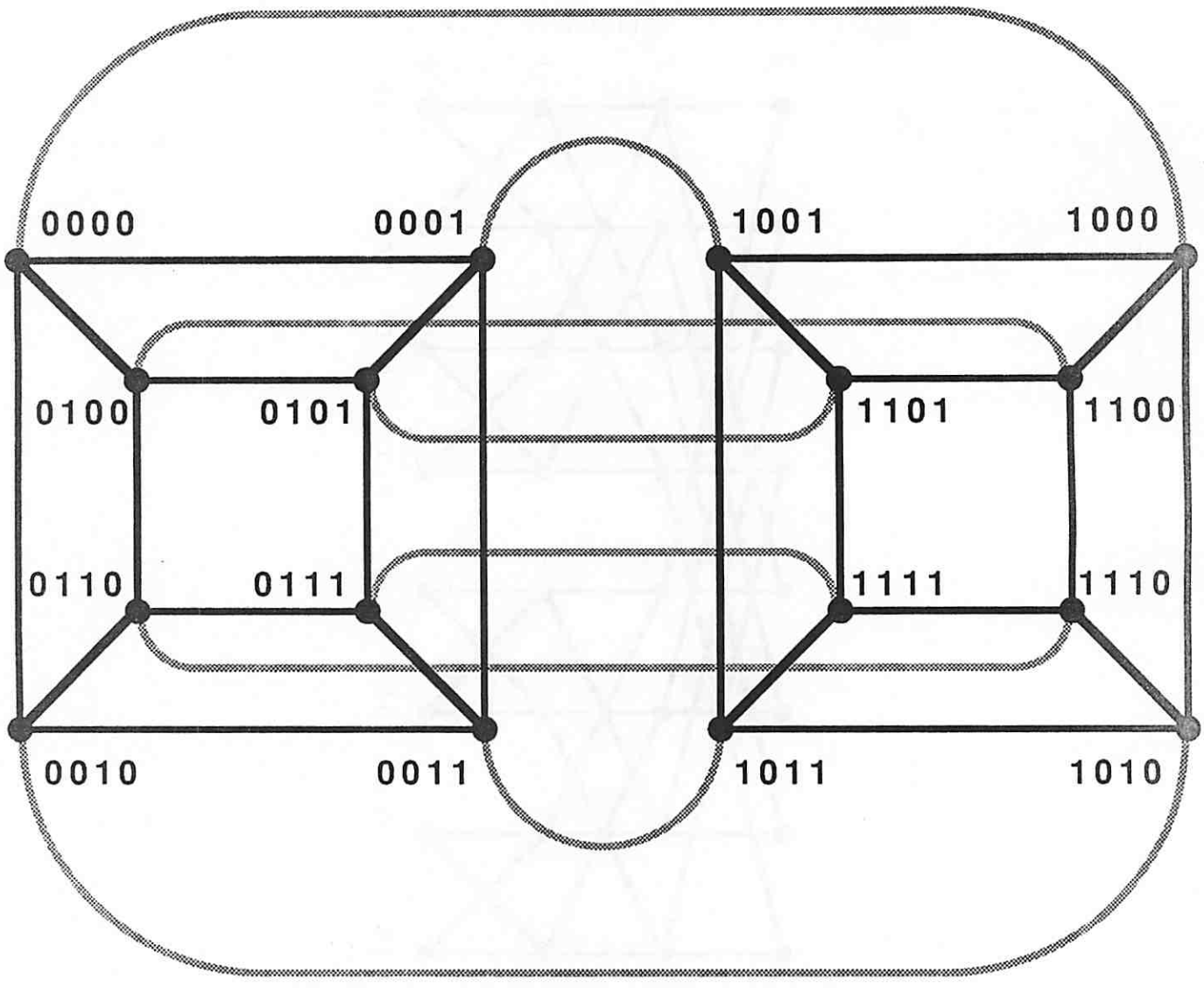

Figure 2: The 4-dimensional Hypercube $Q(4)$.

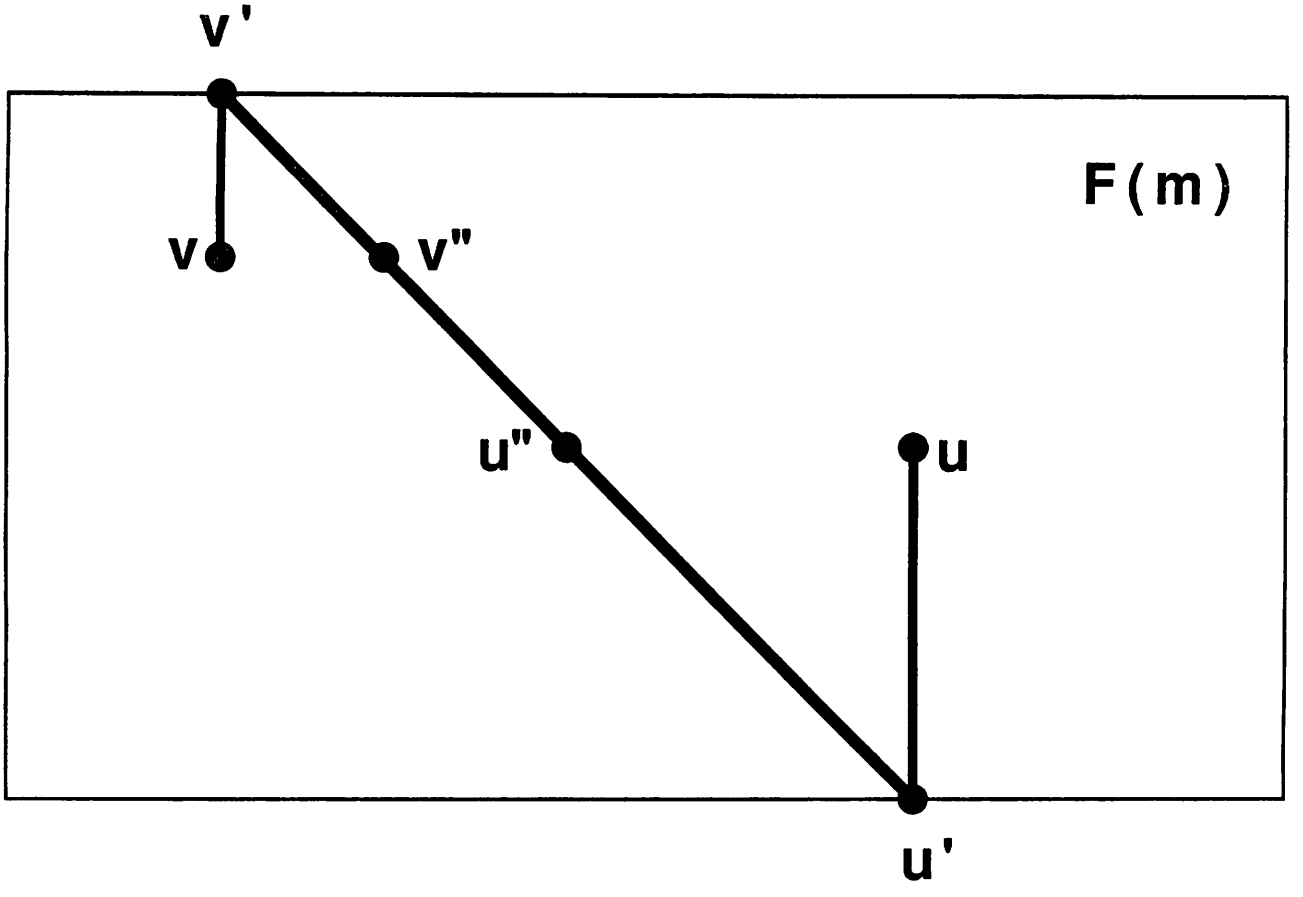

Figure 3: The path connecting u and v in the proof of Lemma 2.

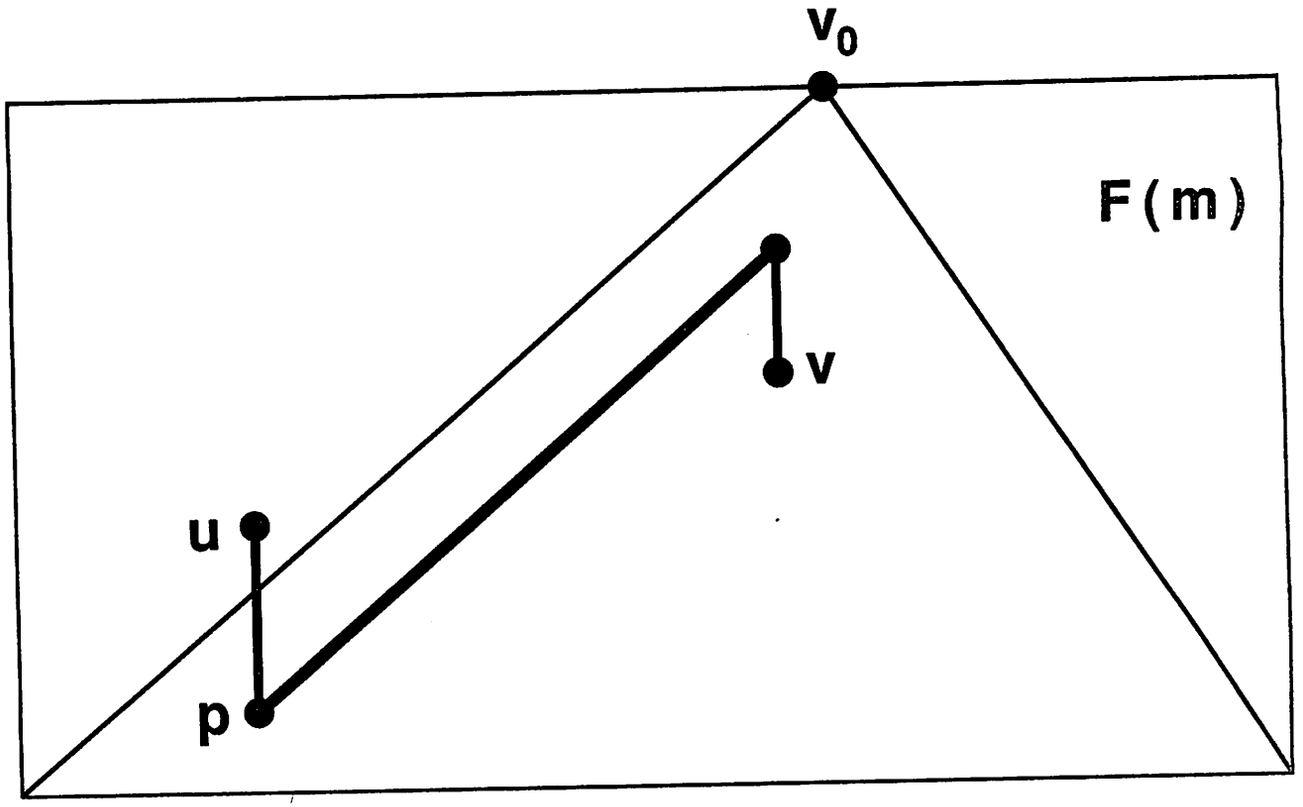

Figure 4: The path connecting u and v in the proof of Lemma 3.