ELSEVIER

Information Processing Letters 63 (1997) 91–95

Information Processing Letters

Recognizing Hamming graphs in linear time and space

Wilfried Imrich [a], Sandi Klavžar [b,*]

[a] Department of Mathematics and Applied Geometry, Montanuniversität Leoben, A-8700 Leoben, Austria
[b] Department of Mathematics, PEF, University of Maribor, Koroška cesta 160, 2000 Maribor, Slovenia

Received 10 March 1995; revised 5 March 1997
Communicated by S. Zaks

Abstract

Hamming graphs are, by definition, the Cartesian product of complete graphs. In the bipartite case these graphs are hypercubes. We present an algorithm recognizing Hamming graphs in linear time and space. This improves a previous algorithm which was linear in time but not in space. This also favorably compares to the general decomposition algorithms of graphs with respect to the Cartesian product, none of which is linear. © 1997 Elsevier Science B.V.

Keywords: Hamming graphs; Design of algorithms; Analysis of algorithms

1. Introduction

Hamming graphs are a relevant class of graphs in Computer Science. By definition they are the Cartesian product of complete graphs and the problem of effectively recognizing whether a graph is a Hamming graph could be solved by prime factorization algorithms with respect to the Cartesian product. The fastest known algorithm for such a decomposition is due to Aurenhammer, Hagauer and Imrich [1] and is of time complexity $O(m \log n)$, where m is the number of edges and n the number of vertices of the graph considered. In [4] it was demonstrated how to reduce this complexity to $O(m)$ for the special case of hypercubes and in [8] this was generalized to all Hamming graphs. However, the space complexity of the latter algorithm is $O(n^2)$. In this paper we present an algorithm for recognizing Hamming graphs which is linear in both, time and space. We hope that this algorithm will help to further improve the general decomposition algorithms of graphs with respect to the Cartesian product.

All graphs considered in this note are finite undirected graphs without loops or multiple edges. Throughout the paper, for a given graph G, let n and m stand for the number of its vertices and edges, respectively. For a graph G and a vertex set $X \subset V(G)$ let $\langle X \rangle$ denote the subgraph of G induced by X.

The *Cartesian product* $G \square H$ of graphs G and H is the graph with vertex set $V(G) \times V(H)$ and $(a,x)(b,y) \in E(G \square H)$ whenever $ab \in E(G)$ and $x = y$, or $a = b$ and $xy \in E(H)$. The Cartesian product is commutative and associative in an obvious way and has the one-vertex graph K_1 as a unit. We also recall that, up to isomorphism, each connected graph can be uniquely written as a Cartesian product of prime graphs [11,12].

* Corresponding author. This work was supported in part by the Ministry of Science and Technology of Slovenia under the grant J1-7036.

PII S0020-0190(97)00095-1

As already mentioned, a *Hamming graph* is the Cartesian product of complete graphs. In the special case when all the factors are isomorphic to the graph K_2 we obtain hypercubes. Many characterizations of Hamming graphs are known, we refer to [2,3] and references there. In addition, several other classes of graphs closely related to Hamming graphs were also studied, cf. [13–15] and references therein.

As we work with graphs on n vertices and m edges our input consists of $n + m$ integers and to encode them in the binary representation we need $O(n \log n) + O(m \log m) = O((n + m) \log n)$ bits. In addition, all operations will be performed on such integers, i.e. on words of length $O(\log n)$. Following the customary scheme for graph algorithms we will henceforth omit the factor $\log n$, cf. [9]. In other words, we analyse algorithms in the arithmetic model, cf. [7]. We will return to the binary representation in the last section where it will be observed that in the case of hypercubes the so-called compression procedure is not essential for the space complexity.

2. The algorithm

For our purposes the following alternative definition of Hamming graphs will be convenient.

Let $r_1, r_2, \ldots, r_t$ be given integers $\geqslant 2$ and let V be the set of t-tuples $a_1a_2 \ldots a_t$ with $0 \leqslant a_i \leqslant r_i - 1$. These t-tuples will be the set of vertices of our Hamming graph. We note that there are $n = \prod_{i=1}^{t} r_i$ such t-tuples. We connect any two t-tuples $a_1a_2 \ldots a_t$ and $b_1b_2 \ldots b_t$ by an edge if they differ in exactly one place, i.e. if there is a j such that $a_j \neq b_j$ but $a_i = b_i$ for $i \neq j$. Let E be the set of such edges. It is straightforward to see that the graph $H = (V, E)$ is a Hamming graph. The corresponding labelling of the vertices of H is called a *Hamming labelling*.

The algorithm we are going to present consists of three parts. First, we check some basic properties of a given graph G to be a Hamming graph and prepare data structures for the next parts. Then, in the procedure Labelling, we label vertices of G with strings of length t, where t is the expected number of factors. Finally, in the procedure Compression we shorten these labels to only two coordinates and verify whether G is indeed a Hamming graph.

Clearly, $H = K_{r_1} \square K_{r_2} \square \cdots \square K_{r_t}$ is an $(r_1+r_2+\cdots+r_t-t)$-regular graph on $r_1r_2\cdots r_t$ vertices. Thus H has $\frac{1}{2} r_1 r_2 \cdots r_t (r_1+r_2+\cdots+r_t-t)$ edges. Note also that the neighborhood of a vertex of H induces a disjoint union of complete graphs. With these observations we can define the following procedure. We assume that the input graph G is connected and given in its adjacency list representation.

Procedure Initialization

1. Choose an arbitrary vertex v_0 of G.
2. Rename the vertices of G and adjust the adjacency list according to the BFS order with respect to v_0.
3. Arrange the vertices in levels $L_0, L_1, \ldots, L_k$ such that L_i contains all vertices of distance i from v_0.
4. Find the connected components of the subgraph of G spanned by the vertices of L_1 and sort them by their sizes. Let these components be $C_1, C_2, \ldots, C_t$ with $r_1 - 1, r_2 - 2, \ldots, r_t - 1$ vertices, respectively.
5. If any of the subgraphs induced by the C_i is not complete then reject.
6. If $n \neq \prod_{i=1}^{t} r_i$ then reject.
7. If $m \neq \frac{1}{2} \sum_{i=1}^{t} (r_i(r_i - 1) \prod_{j=1, j\neq i}^{t} r_j)$ then reject.

Suppose for a moment that G is a Hamming graph. Then we can say that L_k consists of all those t-tuples $a_1a_2 \ldots a_t$ in which exactly k of the a_i are $\neq 0$. In particular, L_0 consists only of v_0 and L_1 of all neighbors of v_0.

For the labelling procedure we state the next three straightforward lemmas, cf. [8].

Lemma 1. *Let π be a permutation of $\{0, 1, \ldots, r_i - 1\}$. If*

$h : v \mapsto a_1a_2 \ldots a_i \ldots a_t$

is a Hamming labelling of H, then

$\pi h : v \mapsto a_1a_2 \ldots \pi a_i \ldots a_t$

is also a Hamming labelling.

Lemma 2. *Let $1 \leqslant i < j \leqslant t$ and h be given as in Lemma 1. Then*

$h_{ij} : v \mapsto a_1a_2 \ldots a_{i-1}a_ja_{i+1} \ldots a_{j-1}a_ia_{j+1} \ldots a_t$

is also a Hamming labelling.

Lemma 3. *Let $u = a_1a_2 \dots a_t \in L_k$, $k \geqslant 1$. Then every neighbor v of u in L_{k-1} has exactly one more vanishing component than u.*

Also, if $k \geqslant 2$ the vertex u has at least two neighbors v, w in L_{k-1} and they differ in exactly two coordinates.

Moreover, if $v = b_1b_2 \dots b_t$ and $w = c_1c_2 \dots c_t$ then $a_i = \max\{b_i, c_i\}$ for $i = 1, \dots, t$.

According to the above lemmas, we can assign labels to the vertices of G by the following procedure.

Procedure Labelling

1. Label v_0 with a vector of length t containing only zeros.
2. Label the vertices of C_i with vectors of the form $0 \dots 0a_i0 \dots 0$, i.e. vectors of length t in which only the i-th coordinate a_i is different from zero, but where a_i assumes all values between 1 and $r_i - 1$.
3. For each vertex u in levels L_i, $2 \leqslant i \leqslant k$, select any two vertices adjacent to u, say u^1 and u^2, in level L_{i-1}. If there are no such vertices then reject.
4. Suppose all vertices in L_j, $1 \leqslant j < k$, have already been labelled. Choose an unlabelled vertex u in L_{j+1}. Let the labels of u^1 and u^2 be $b_1b_2 \dots b_t$ and $c_1c_2 \dots c_t$, respectively. Setting $a_i = \max\{b_i, c_i\}$ we obtain a label $a_1a_2 \dots a_t$ for u.

It follows immediately from Lemmas 1–3 that the labelling algorithm, applied to a Hamming graph G, yields a Hamming labelling of G.

We next describe how to compress the labels obtained in the previous procedure to only two coordinates. Consider the graph

$$H = K_{r_1} \,\Box\, K_{r_2} \,\Box \cdots \Box\, K_{r_t}$$

and set

$$H_1 = K_{r_1} \,\Box\, K_{r_3} \,\Box \cdots$$

and

$$H_2 = K_{r_2} \,\Box\, K_{r_4} \,\Box \cdots.$$

In other words, H_1 and H_2 are the subproducts of H with odd indexed and even indexed factors, respectively. As the Cartesian product is associative and commutative we have $H = H_1 \,\Box\, H_2$.

Procedure Compression

1. Let G_1 be the subgraph of G induced by the vertices with all even coordinates equal to zero and let G_2 be the subgraph of G induced by the vertices with all odd coordinates equal to zero.
2. If G_1 is not isomorphic to H_1, or G_2 is not isomorphic to H_2, then reject.
3. Label the vertices of G_1 with $\{1, 2, \dots, |G_1|\}$, the vertices of G_2 with $\{1, 2, \dots, |G_2|\}$ and represent G_1 and G_2 by their adjacency matrices.
4. Label each vertex v of G be two coordinates as follows. If v labelled $a_1a_2 \cdots a_t$, let its first coordinate be the vertex of G_1 corresponding to label $a_10a_30 \dots$ and let its second coordinate be the vertex of G_2 corresponding to the label $0a_20a_4 \dots$.
5. For each edge uv of G check whether it is in product. More precisely, if the label of u is ij and the label of v is $i'j'$, then check if either $i = i'$ and j is adjacent (in G_2) to j', or $j = j'$ and i is adjacent (in G_1) to i'.

We can thus summarize our algorithm as follows.

The Hamming Graph Algorithm

Input: A connected graph G in its adjacency list representation.

Output: A Hamming labelling of G if it exists, rejection otherwise.

1. Initialization
2. Labelling
3. Compression

Correctness of the algorithm follows from the above discussion.

We next consider the time and the space complexity of the algorithm. Clearly all the steps of the procedure Initialization as well as Steps 1 and 2 of the procedure Labelling can be done in $O(m)$ time and space.

For Step 3 of Labelling we first remember for each vertex its level and then we go through adjacency list and find two appropriate vertices. Therefore the label for a vertex u in Step 4 can be computed in time $O(t)$. It follows that the complexity of this step is $O(nt)$. Since every vertex of G has at least t neighbors we infer $nt \leqslant 2m$. Hence, $O(nt) = O(m)$. We conclude that the procedure Labelling can also be performed in $O(m)$ time and space.

Finally to the compression procedure. The selection of vertices for G_1 and G_2 takes $O(nt) = O(m)$ time. Suppose now that $t = 2s$. Then since $r_1 \leqslant r_2 \leqslant \cdots \leqslant r_{2s}$ we have

$$\begin{aligned} & r_1^2 \cdot r_3^2 \cdot \ldots \cdot r_{2s-3}^2 \cdot r_{2s-1}^2 \\ & \quad \leqslant r_1 \cdot r_2 \cdot r_3 \cdot \ldots \cdot r_{2s-1} \cdot r_{2s} \\ & \quad \leqslant 2m \end{aligned}$$

and similarly

$$\begin{aligned} & r_2^2 \cdot r_4^2 \cdot \ldots \cdot r_{2s-2}^2 \cdot r_{2s}^2 \\ & \quad \leqslant r_2 \cdot r_3 \cdot r_4 \cdot \ldots \cdot r_{2s-1} \\ & \qquad \cdot r_{2s}(r_1 + \cdots + r_{2s} - 2s) \\ & \quad \leqslant 2m. \end{aligned}$$

Analogously we proceed if t is odd. Thus, in all cases we have $|V(G_1)| = O(\sqrt{m})$ and $|V(G_2)| = O(\sqrt{m})$. For the adjacency matrices of G_1 and G_2 we thus need $O(m)$ space. This means that for Step 2 of the procedure Compression we can use the recognition algorithm for Hamming graphs from [8] which is linear in time but uses an adjacency matrix as an input.

Next, projections from Step 4 can clearly be computed in $O(nt)$ time. Finally, since we have only two coordinates and G_1 and G_2 are represented by their adjacency matrices, checking for an edge in Step 5 can be done in $O(1)$ time. We have thus proved:

Theorem 4. *For a given graph G on n vertices and m edges one can decide in $O(m)$ time and $O(m)$ space whether G is a Hamming graph. Both complexities are optimal.*

3. Concluding remarks

In this paper we have given a linear time and space algorithm for recognizing Hamming graphs. A simpler linear time algorithm is given in [8], but its space complexity is $O(n^2)$. This space is needed if we wish to find out in constant time whether two vertices labelled with strings of length t are connected by an edge. Without the adjacency matrix this would yield to time complexity $O(mt)$ which is in general $O(m \log n)$ – the complexity of the algorithm from [1] for the prime factor decomposition with respect to the Cartesian product. Thus the main insight of the present algorithm is the compression procedure, which reduces the number of coordinates.

In the case of hypercubes, though, the compression procedure is not needed, as pointed by one of the referees. Indeed, by our assumption comparing two $O(log n)$ bit integers takes constant time. But then we can also test in constant time whether two labels obtained in the labelling procedure differ in exactly one bit. (We first transform the $\log n$ bit labels to $\log n$ bit integers and then use the XOR and AND boolean operations.) We cannot simplify our algorithm in the general case, because we need to compare strings consisting of $O(\log n)$ integers, each of size $O(\log n)$. The above approach would then add a factor $\log n$ to the time complexity.

Finally, we wish to add that there is a general related result which asserts that if a given graph algorithm runs in linear time and uses $O(n^2)$ space, (i.e. the matrix representation), then the algorithm can be simulated in linear space and *expected* linear time. The expected linear time follows from reference [5], while reference [6] gives the same result but also ensures high probability. The main insight of this note is thus that in the case of Hamming graphs we are able to find an algorithm which runs in deterministic linear time.

Acknowledgement

We wish to thank the referees for helpful remarks.

References

[1] F. Aurenhammer, J. Hagauer and W. Imrich, Cartesian graph factorization at logarithmic cost per edge, *Comput. Complexity* 2 (1992) 331–349.

[2] H.-J. Bandelt, Characterization of Hamming graphs, Manuscript, 1992.

[3] H.-J. Bandelt, H.M. Mulder and E. Wilkeit, Quasi-median graphs and algebras, *J. Graph Theory* 18 (1994) 681–703.

[4] K.V.S. Bhat, On the complexity of testing a graph for n-cube, *Inform. Process. Lett.* 11 (1980) 16–19.

[5] J. Carter and M. Wegman, Universal classes of hash functions, *J. Comput. System Sci.* 18 (1979) 143–154.

[6] M. Dietzfelbinger, A. Karlin, K. Melhorn, F. Meyer auf der Heide, H. Rohnert and R.E. Tarjan, Dynamic perfect hashing: Upper and lower bounds, *SIAM J. Comput.* 23 (1994) 738–761.

[7] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Algorithms and Combinatorics, Vol. 2 (Springer, Berlin, 2nd corr. ed., 1994).

[8] W. Imrich and S. Klavžar, On the complexity of recognizing Hamming graphs and related classes of graphs, *European J. Combin.* 17 (1996) 209–221.

[9] J. van Leeuwen, Graph Algorithms, in: *Handbook of Theoretical Computer Science* (Elsevier, Amsterdam, 1990) Chapter 10, 525–631.

[10] H.M. Mulder, The interval function of a graph, *Math. Centre Tracts (Amsterdam)* 132 (1980).

[11] G. Sabidussi, Graph multiplication, *Math. Z.* 72 (1960) 446–457.

[12] V.G. Vizing, The Cartesian product of graphs, *Comp. El. Syst.* 2 (1966) 352–365.

[13] E. Wilkeit, Isometric embeddings in Hamming graphs, *J. Combin. Theory Ser. B* 50 (1990) 179–197.

[14] E. Wilkeit, The retracts of Hamming graphs, *Discrete Math.* 102 (1992) 197–218.

[15] P. Winkler, Isometric embeddings in products of complete graphs, *Discrete Appl. Math.* 7 (1984) 221–225.