Optimal Arrangement of Data in a Tree Directory

M.J. Luczak [1]

Mathematical Institute, 24–29 St. Giles, Oxford, OX1 3LB, UK

S.D. Noble

Department of Mathematical Sciences, Brunel University, Kingston Lane, Uxbridge, UB8 3PH, UK

Abstract

We define the decision problem DATA ARRANGEMENT, which involves arranging the vertices of a graph G at the leaves of a d-ary tree so that a weighted sum of the distances between pairs of vertices measured with respect to the tree topology is at most a given value. We show that DATA ARRANGEMENT is strongly NP-complete for any fixed $d \geq 2$ and explain the connection between DATA ARRANGEMENT and arranging data in a particular form of distributed directory.

Key words: Complexity, Graph Embedding, Data Arrangement.

1 Introduction

This paper is concerned with arranging data between nodes in a form of distributed directory described in [10] for use in communication systems. The directory has the form of a rooted tree with data for users in a particular cell being stored at each of the leaves of the tree. Upon a call connection request, a signal must pass through the tree from the node containing data about the user making the call to the node containing data about the user receiving the call, in order to find the receiver's current address and enable the call.

For any pair of cells i and j, we define a traffic intensity λ_{ij} between these cells. A natural problem to consider is how to map cells to leaves of a tree directory

[1] The author is supported by a British Telecommunications studentship

to minimise the average cost of lookups weighted by the traffic intensities. This motivates the study of the problem DATA ARRANGEMENT.

In Section 2 we introduce a very general problem, involving graph embedding, which is a generalisation of the OPTIMAL LINEAR ARRANGEMENT problem. We show that it contains various classical graph problems as specialisations. This enables us to prove that it is NP-complete even in very restricted cases. However the main result of this paper concerns the problem DATA ARRANGEMENT which we describe in Section 3. One version of DATA ARRANGEMENT is a very special case of the embedding problem. We prove in Section 4 that DATA ARRANGEMENT is strongly NP-complete.

2 A General Embedding Problem

Problems involving embedding a graph into another so as to minimise a particular cost function have received much attention, due to their importance in VLSI design. For a general survey see for example [3]. We will be concerned with a particular embedding problem, namely OPTIMAL LINEAR ARRANGEMENT.

All our notation is fairly standard. A graph G with vertex set V and edge set E is denoted by $G = (V, E)$. Our graphs are assumed to have no loops or multiple edges. For any vertices u and v, $d_G(u, v)$ denotes the shortest distance between them in G.

Given a graph $G = (V, E)$, a *labelling* of G is an injective function $f : V \to \{1, \ldots, |V|\}$. The *total length* $L_f(G)$ of a labelling f is given by

$$L_f(G) = \sum_{\{i,j\} \in E} |f(i) - f(j)|.$$

The *total length* $L(G)$ of a graph G is the minimum over all labellings of $L_f(G)$. Note that although the total length has been well-studied [2,8] in connection with the optimal linear arrangement problem we describe below, there does not seem to be any popular name for what we call the total length.

This motivates the following decision problem.

OPTIMAL LINEAR ARRANGEMENT
Instance: A graph $G = (V, E)$ and an integer B.
Question: Is $L(G) \leq B$?

This problem was first investigated by Harper [8] in connection with error correcting codes. It is known to be NP-complete [6] but there is a polynomial

time algorithm when the input is restricted to being a tree [2,7,11]. This contrasts with the bandwidth problem, in which the total sum of a labelling is replaced by $\max_{\{i,j\}\in E} |f(i) - f(j)|$, since this problem is NP-complete even when the input is restricted to being a tree with no vertex having degree exceeding three [4].

We now consider ways of generalising the total length of a labelling. One way to do this would be to take a function $g : \mathbb{N}^2 \to \mathbb{N}$ satisfying $g(x, y) = g(y, x)$ for all x and y and define the *g-total length* $L_f^g(G)$ of a labelling f to be

$$L_f^g(G) = \sum_{\{i,j\}\in E} g(f(i), f(j)).$$

However we will restrict our attention to the case when $g(x, y)$ is the distance between vertices x and y in a fixed graph H, so rather than labelling the vertices of a graph G we will be embedding it into a graph H. From now on we will refer to G as the *guest* graph and H as the *host* graph. Given $G = (V_1, E_1)$ and $H = (V_2, E_2)$, an injective function $f : V_1 \to V_2$ is said to be an *embedding*. The *total length* of such an embedding is $\sum_{\{i,j\}\in E_1} d_H(f(i), f(j))$.

Given such a cost function, it is natural to consider the complexity of finding an embedding that minimises it. This motivates the following decision problem.

GRAPH EMBEDDING
Instance: Graphs G, H, a subset A of $V(H)$ and an integer B.
Question: Is there a 1-1 function $f : V(G) \to A$ which satisfies

$$\sum_{\{i,j\}\in E(G)} d_H(f(i), f(j)) \leq B.$$

Placing various simple restrictions on the input turns the problem into one of the classical graph problems. For instance,

(1) if G is a circuit with $|V(H)|$ vertices, $A = V(H)$ and $B = |V(H)|$ then the problem becomes HAMILTONIAN CIRCUIT;
(2) similarly, if G is a path with $|V(H)|$ vertices, $A = V(H)$ and $B = |V(H)| - 1$ then the problem becomes HAMILTONIAN PATH;
(3) if $|V(G)| = |V(H)|$, $|E(G)| = |E(H)|$, $A = V(H)$ and $B = |E(G)|$ then the problem becomes GRAPH ISOMORPHISM;
(4) if H is a path with $|V(G)|$ vertices and $A = V(H)$ then the problem becomes OPTIMAL LINEAR ARRANGEMENT.

There are likely to be many similar examples. In the next section we consider a restriction of this problem that requires A to be a proper subset of V_2; it would be nice if there were other natural problems with this requirement.

The first two examples above imply the following.

Proposition 1 *The problem* GRAPH EMBEDDING *is NP-complete even when the guest graph is a path or a circuit.*

The final example implies the following.

Proposition 2 *The problem* GRAPH EMBEDDING *is NP-complete even when the host graph is a path.*

Clearly these two propostions rule out the possibility of a polynomial time algorithm for the case when one of the input graphs has bounded tree-width. It is obviously possible to consider the complexity of many other restrictions. The answer to the following question seems far from obvious.

Problem 3 *What is the complexity of* GRAPH EMBEDDING *when both input graphs are restricted to being trees?*

3 Data Arrangement

We now move on to the main problem in this paper, namely, DATA ARRANGEMENT. First we need to introduce some notation. We define a d-ary tree as a rooted tree in which each node has at most d children. In a $(0, d)$ d-ary tree, each node has exactly 0 or d children. A complete d-ary tree of height h is a $(0, d)$ d-ary tree in which there are exactly d^r nodes at a distance r from the root, for $r = 0, \ldots, h$.

In DATA ARRANGEMENT, the question is to decide whether there exists an arrangement of the data from the nodes of a communication graph $G = (V, E)$ at the leaves of a complete d-ary tree of height $\lceil \log_d |V| \rceil$ such that the sum over all i and j in the range $\{1, \ldots, |V|\}$, weighted by the traffic intensities λ_{ij}, of the lengths of the paths along the tree between cells i and j, is less than or equal to a given value. This corresponds to assigning geographical cells to the leaves of a tree directory so as to minimise communication costs.

We now give a precise formulation of the problem.

DATA ARRANGEMENT
Instance: Graph $G = (V, E)$ integer weights $\lambda_{ij} \geq 0$ for each pair of vertices i and j, symmetric in i and j, a nonnegative integer B; numbers λ_{ij} and B given in binary.
Question: Is there an injective mapping f from V to the leaves of a complete

d-ary tree, T, of height $\lceil \log_d |V| \rceil$, such that

$$\sum_{i=1}^{|V|} \sum_{j=1, j \neq i}^{|V|} \lambda_{ij} d_T(f(i), f(j)) \leq B.$$

We shall be mainly interested in the restricted case where λ_{ij} depends only on whether there is an edge between i and j. That is

$$\lambda_{ij} := \begin{cases} 1, \text{ if } \{i, j\} \in E, \\ 0, \text{ otherwise.} \end{cases}$$

We call the restricted problem SIMPLE DATA ARRANGEMENT. The formal definition is as follows.

SIMPLE DATA ARRANGEMENT
Instance: Graph $G = (V, E)$ and a nonnegative integer B given in binary.
Question: Is there an injective mapping f from V to the leaves of a complete d-ary tree, T, of height $\lceil \log_d |V| \rceil$, such that

$$\sum_{\{i,j\} \in E} d_T(f(i), f(j)) \leq B?$$

The path between i and j in the tree contributes to the total sum if and only if i and j are joined by an edge in G. Note that SIMPLE DATA ARRANGEMENT is a special case of GRAPH EMBEDDING where $G = (V, E)$ is an arbitrary graph, H is a complete d-ary tree of height $\lceil \log_d |V| \rceil$ and A is the set of leaves of H.

We now state the main result of the paper.

Theorem 4 SIMPLE DATA ARRANGEMENT *is NP-complete for any fixed* $d \geq 2$.

The next section consists of the proof of this theorem. This result implies the following corollary.

Corollary 5 DATA ARRANGEMENT *is strongly NP-complete for any fixed* $d \geq 2$.

The idea of embedding on the leaves of a binary tree is not new. In [1] Bienstock considers this problem in detail but he is concerned with cutwidth and bandwidth. He gives bounds on the cutwidth and bandwidth of the embedding in terms of the tree-width of the graph being embedded. In contrast to [1] we have just been concerned with complexity.

It seems sensible to consider various restrictions on the input graph in the hope that this would lead to a polynomial time algorithm for SIMPLE DATA ARRANGEMENT. The position seems far from clear even when the input is a tree.

Problem 6 *What is the complexity of* SIMPLE DATA ARRANGEMENT *when the input is restricted to being a tree?*

4 Proof of Main Result

We shall first show that the following problem is NP-complete.

SIMPLE MAX DATA ARRANGEMENT
Instance: Graph $G = (V, E)$ and a nonnegative integer B.
Question: Is there an injective mapping f from V to the leaves of a complete d-ary tree, T, of height $\lceil \log_d |V| \rceil$ such that

$$\sum_{\{i,j\} \in E} d_T(f(i), f(j)) \geq B?$$

Before setting out to prove the main result, we need to establish some auxiliary facts.

Definition 7 *Suppose $P = \{V_i : 1 \leq i \leq m\}$ is a partition of the vertex set V of a graph $G = (V, E)$. Then a* cross-edge *of P is an edge of the form $\{i, j\}$, with i and j contained in two distinct elements of P.*

We now make the following observation.

Observation 1 *The total sum of all the path lengths in a complete d-ary tree can be split into the contributions due to separate levels in the tree.*

PROOF. Let us consider the contribution of the edges at level r for any $1 \leq r \leq h$. There are d^{h-r+1} nodes directly below them, and each of these nodes has a subtree with d^{r-1} leaves rooted underneath. These subtrees correspond to a partition of the nodes of G into d^{h-r+1} subsets. The contribution due to level r edges, therefore, is the number of 'cross-edges' of the partition, multiplied by 2. $\square$

We also need to consider the following problem.

d- PARTITION INTO STABLE SETS OF EQUAL SIZE (d-PSSES)

Instance: Graph $G = (V, E)$, with $|V|$ divisible by d, where d is a positive integer.
Question: Can V be partitioned into d stable sets of equal size?

Proposition 8 *d-PSSES is NP-complete for any fixed $d \geq 3$.*

PROOF. It is clear that the problem is in NP. The proof of its NP-completeness is by a transformation from GRAPH d-COLOURABILITY [5], [9]. In the proof, we restrict the input to graphs on d^h vertices, for positive integers h.

Given an instance of d-COLOURABILITY, that is, a graph $G = (V, E)$, we construct a corresponding instance of d-PSSES,

$$V' = V \cup \{x_1, \ldots, x_{(d^h - |V|)}\},$$

where

$$V \cap \{x_1, \ldots, x_{(d^h - |V|)}\} = \emptyset,$$

$$h = \lceil \log_d |V| \rceil + 1,$$

and

$$E' = E.$$

Then $|V'| = d^h$. The transformation is polynomial, since $|V| \geq d^{h-2}$ and $|V'| = O(|V|)$. Suppose G is d-colourable. Then G can be partitioned into stable sets $A_1, \ldots, A_d$ corresponding to colours $1, \ldots, d$. Add vertices to each A_i from $\{x_1, \ldots, x_{(d^h - |V|)}\}$, to obtain $A'_1, \ldots, A'_d$ such that

$$|A'_i| = d^{h-1}$$

for $i = 1, \ldots, d$. Thus G' can be partitioned into d stable sets of equal size.

Conversely, suppose G' can be partitioned into d stable sets $A'_1, \ldots, A'_d$ of equal size. Set $A_i = A'_i \backslash \{x_1, \ldots, x_{(d^h - |V|)}\}$ for $i = 1, \ldots, d$, to get a d-colouring for G.

The result follows from the NP-completeness of d-COLOURABILITY for $d \geq 3$ [9]. $\square$

Proof of the main result Firstly, it is clear that both SIMPLE DATA ARRANGEMENT and SIMPLE MAX DATA ARRANGEMENT are in NP.

We now show that SIMPLE MAX DATA ARRANGEMENT is NP-complete. In the proof, we restrict the input to the class of graphs $G = (V, E)$ such that

$|V| = d^h$ for some positive integer h.

Case 1 $d \geq 3$. The proof is by a transformation from d-PSSES.

Given an instance of d-PSSES, such that the input graph $G = (V, E)$ has d^h vertices for some h, we define the corresponding instance of SIMPLE MAX DATA ARRANGEMENT on G with $B := 2h|E|$. Clearly, the transformation is polynomial.

Note that the value of B in the transformation is the maximum possible one in SIMPLE MAX DATA ARRANGEMENT. It can only be achieved by an arrangement of vertices such that every pair of vertices joined by an edge in G are separated by $2h$ links in the tree.

By Observation 1, the contribution due to level r edges is twice the number of cross-edges in the partition of the vertices of G into d^{h-r+1} sets of equal size determined by the subtrees underneath the level r edges. In particular, the contribution of the top level edges is twice the number of cross-edges in the corresponding partition of V into d sets of equal size.

Suppose there exists a partition $\{V_1, \ldots, V_d\}$ of V into stable sets of equal size. We define a corresponding leaf arrangement in the d-ary tree by putting the members of V_i together, in an arbitrary order, as leaves of the ith subtree of the root node.

Then only paths of length $2h$ contribute to the sum and the number of contributions equals the number of edges, $|E|$.

Conversely, suppose that there exists an arrangement such that the weighted sum of distances in the tree equals B. Let U_i be the subset of V which is mapped onto the leaves of the ith subtree of the root for $i = 1, \ldots, d$. Then each of the U_i must be a stable set.

Case 2 $d = 2$. The proof of this part is by a transformation from SIMPLE MAX CUT [5], [6].

Given an instance of SIMPLE MAX CUT, a graph G and a nonnegative integer K, we construct a graph G' as follows. Corresponding to each edge $e = \{i, j\}$, we add nodes e_1 and e_2. We also replace edge $\{i, j\}$ by edges $\{i, e_1\}$, $\{e_1, e_2\}$, and $\{e_2, j\}$. We now add an appropriate number of isolated nodes so that the number of vertices in G' is a power of 2. We want $|V'| = 2^h$, where $h = \lceil \log_2 (|V| + 2|E|) \rceil + 2$. It is clear that this can be done in time polynomial in input size.

Let M be the maximum size of a cut in G; then $2|E| + M$ is the maximum size of a cut in G'. To see this, suppose $(S, V \setminus S)$ is a maximum cut in G. We obtain a maximum cut $(S', V \setminus S')$ as follows. If $e = \{i, j\}$ is an edge such that $i \in S$ and $j \in V \setminus S$, then we put j and e_1 in $V \setminus S'$, and we put i and e_2 in S'. This ensures that edges $\{i, e_1\}$, $\{e_1, e_2\}$, and $\{e_2, j\}$ all contribute to the cut. If i and j are in the same set, say S, then put i and j in S', e_1 and e_2 in $V \setminus S'$. Then edges $\{i, e_1\}$ and $\{e_2, j\}$ contribute to the cut.

Now, we show that G has a cut of size at least K if and only if there is an arrangement of vertices of G' with total weighted sum of path lengths

greater than or equal to

$$2(h-1)(|E|-K)+2h(2|E|+K).$$

First note that the above value is the maximum possible. It can only be achieved by an arrangement of vertices of G' in which each edge of G' contributes at least $2(h-1)$ to the sum. Thus the vertices of G' must be partitioned into 4 stable sets of equal size. Also, a necessary condition is that G' has a cut of size at least $2|E|+K$. This can only happen if G has a cut of size at least K.

Conversely, suppose G has a cut $(S, V \setminus S)$ of size K. We partition the vertices of G' into four stable sets of equal size, S_1, S_2, S_3 and S_4 as follows. Set S_1 consists of vertices of S and an appropriate number of isolated nodes. Set S_4 contains nodes from $V \setminus S$ and some isolated nodes. Using the same notation as above, for each edge $e = \{i, j\}$ such that $i \in S$ and $j \in V \setminus S$, we put e_1 in S_3 and e_2 in S_2. If i and j are both in S or both in $V \setminus S$, then we put e_1 in S_2 and e_2 in S_3 (or the other way round). We 'fill up' S_2 and S_3 with isolated vertices. This is possible because the number of vertices $|V'|$ of G' satisfies $|V| \leq |V'|/4$, and $|E| \leq |V'|/4$. We place S_1 and S_2 together in the left subtree, and S_3 together with S_4 in the right subtree. The resulting vertex arrangement has the value of the path length sum required.

Finally we need to show that SIMPLE DATA ARRANGEMENT is NP-complete. Again, it is enough to consider the restricted case where the number of vertices in the input graph is a power of d. We make a transformation from SIMPLE MAX DATA ARRANGEMENT. Given an instance of this, a graph $G = (V, E)$ on d^h vertices for some positive integer h, and an integer B, we construct an equivalent instance of SIMPLE DATA ARRANGEMENT on $\bar{G}$, the complement of G, setting

$$B' := \sum_{r=1}^{h} 2r \binom{d}{2} d^{2(r-1)} d^{h-r} - B.$$

Then the original instance of SIMPLE MAX DATA ARRANGEMENT has a 'yes' answer if and only if the SIMPLE DATA ARRANGEMENT instance constructed has a 'yes' answer. This is because, for $r = 1, \ldots, h$, there are d^{h-r} subtrees of height r whose leaves are also leaves of the entire tree. Each such subtree contributes up to $\binom{d}{2} d^{2(r-1)}$ pairs of leaves separated by $2r$ links. $\square$

We observe also that the case $d = 2$ of SIMPLE DATA ARRANGEMENT requires a different reduction, since partitioning the vertex set of a graph into two stable sets of equal size is easy.

5 Conclusions

We close by recalling the two problems stated in this paper.

(1) What is the complexity of GRAPH EMBEDDING when both input graphs are restricted to being trees?
(2) What is the complexity of SIMPLE DATA ARRANGEMENT when the input graph is restricted to being a tree?

The existence of a polynomial time algorithm for either of these problems would then suggest studying the case where the input is restricted to graphs with bounded tree-width.

Acknowledgements

We would like to thank Dominic Welsh for helpful suggestions.

References

[1] D. Bienstock, On embedding graphs in trees, Journal of Combinatorial Theory Series B 49 (1990) 103–137.

[2] F. R. K. Chung, On optimal linear arrangements of trees, Computers and Mathematics with Applications 10 (1984) 43–60.

[3] F. R. K. Chung, Labellings of graphs, in: L. W. Beineke and R. J. Wilson, eds., Selected Topics in Graph Theory 3 (Academic Press, 1988), 151–169.

[4] M. R. Garey, R. L. Graham, D. S. Johnson and D. E. Knuth, Complexity results for bandwidth minimization, SIAM Journal on Applied Mathematics 34 (1978) 477–495.

[5] M. R. Garey and D. S. Johnson, Computers and Intractability (W. H. Freeman, New York, 1979).

[6] M. R. Garey, D. S. Johnson and L. Stockmeyer, Some simplified NP-complete graph problems, Theoretical Computer Science 1 (1976) 237–267.

[7] M. A. Goldberg and I. A. Klipker, Minimal placing of trees on a line, Technical report, Physico-Technical Institute of Low Temperatures, Academy of Sciences of Ukranian SSR, USSR (1976).

[8] L. H. Harper, Optimal assignments of numbers to vertices, Journal of the Society for Industrial and Applied Mathematics 12 (1964) 131–135.

[9] R. M. Karp, Reducibility among combinatorial problems, in: J. W. Thatcher and R. E. Miller, eds., Complexity of Computer Computations (Plenum Press, New York, 1972), pages 85–103.

[10] I. Novoa and M. Wilby, Knowledge and location, in: Proceedings of the International Joint Conference on Artificial Intelligence, Montreal (August 1997).

[11] Y. Shiloach, A minimum linear arrangement algorithm for undirected trees, SIAM Journal on Computing 8 (1979) 15–32.