

The Linear Arrangement Problem on Recursively Constructed Graphs

S. B. Horton

Department of Mathematical Sciences, United States Military Academy, West Point, New York, 10996

T. Easton

Department of Industrial and Manufacturing Systems Engineering, Kansas State University, Manhattan, Kansas 66506-5101

R. Gary Parker

School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0205

The linear arrangement problem on graphs is a relatively old and quite well-known problem. Hard in general, it remains open on general recursive graphs (i.e., partial k -trees, etc.), which is somewhat frustrating since most hard graph problems are easily solved on recursive graphs. In this paper, we examine the linear arrangement problem with respect to these structures. Included are some negative (complexity) results as well as a solvable case. © 2003 Wiley Periodicals, Inc.

Keywords: optimal linear arrangement; complexity; Halin graphs

1. INTRODUCTION

1.1. Linear Arrangement

Given a simple, finite graph $G = (V, E)$ of order n , the *optimal linear arrangement* problem (OLA) seeks a one-to-one function $f : V \rightarrow \{1, 2, \dots, n\}$ such that $L(G) = \sum_{(u,v) \in E} |f(u) - f(v)|$ is minimum over all such labelings. For ease, we shall denote optimal labelings by f^* and their values by L^* .

OLA is well known to be \mathcal{NP} -complete in general [9], but has been solved on trees following algorithms of Adolphson and Hu [1], Shiloach [13], and Chung [5] and on *outerplanar* graphs following an algorithm of Frederickson and Hambrusch [7]. The latter are graphs having no subgraphs homeomorphic to either K_4 or $K_{2,3}$ (topologically,

these are structures embeddable in the plane in such a way that all vertices lie on the outer face).

Of interest to us here is that both trees and outerplanar graphs are recursively constructible; however, none of the above algorithms appears to be extendable to general recursive graphs. Furthermore, no complexity result is known for OLA on such graphs. The open status of OLA on recursive graphs is particularly frustrating, because the status of the majority of \mathcal{NP} -complete graph problems is known on these structures, that is, most are polynomially solvable. This paper focuses on adding greater resolution to some of these issues.

1.2. Recursive Graph Classes

Our descriptions and notation for this section follow Borie et al. [3]. Informally, a *recursive graph class* is one in which any sufficiently large member in the class can be formed by successively joining smaller members in the class at specific vertices called *terminals*. In letting the maximum allowable number of terminals be k , we sometimes refer to these generically as *k -terminal graphs*. Common classes that are describable as *k -terminal graphs* include trees, series-parallel graphs, Halin graphs, bandwidth- k graphs, partial k -trees, and others. More formally, a *k -terminal graph* $G = (V, T, E)$ has a vertex set V , edge set E , and a (possibly ordered) set of distinguished vertices or terminals $T \subseteq V$ specified such that $T = \{t_1, t_2, \dots, t_{t(G)}\}$, where $t(G) = |T| \leq k$. For some k , we let U be the set of all *k -terminal graphs*. Then, a *recursively constructed graph family*, $F = (B, R)$ in U , has base elements (graphs) $B \subseteq U$ and a finite set of recursive composition operations $R = \{h_1, h_2, \dots, h_r\}$, where each $h_i : U^{p_i} \rightarrow U$. Here, p_i denotes the *arity* of the operation h_i .

Received September 2001; accepted July 2003

Correspondence to: S. B. Horton

Published online 21 August 2003 in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/net.10093

© 2003 Wiley Periodicals, Inc.

Generally, we consider only base elements in which all vertices are terminals. But, in this case, it follows that all such structures decompose trivially into edges, so we can simply take B to be a singleton consisting of K_2 .

A *decomposition tree* of a k -terminal graph G is a rooted tree with vertex labels g and h such that

- $g_v = G$ if v is the root,
- $h_v \in R$ if v is an interior node,
- $g_v = h_v(g_{v_1}, g_{v_2}, \dots, g_{v_m})$ if interior node v has children v_1, v_2, \dots, v_m , and
- $g_v = B$ if v is a leaf.

Decomposition trees are central to the general problem-solving approach on k -terminal recursive graphs. Early results demonstrating this were given in Bern et al. [2] and Wimer and Hedetniemi [14]. In the former, a general model was proposed for the construction of linear-time algorithms on recursive graph classes. Loosely stated, the tactic was based on a homomorphism that mapped graph-subgraph pairs to a finite number of equivalence classes; the authors called a subgraph property or predicate *regular* if such a finite homomorphism existed for the predicate P . Important in this is that if P (e.g., vertex-edge incidence, membership, independence) is regular, and if the decomposition tree for a graph is available, then there exists a linear-time algorithm for finding optimal values of the variables satisfying P using straightforward dynamic programming.

Originally, this outcome was somewhat limited in that the set of regular predicates was only known to be closed under logical negation, conjunction, and disjunction. However, in Borie et al. [4], it was shown that this attribute could be extended to include the important operations of existential and universal quantification (there are some restrictions to the use of these quantifications, which can also be found in Borie et al. [4]). This important extension formed the basis for the creation of a powerful predicate calculus which allowed concise descriptions of many well-known hard graph problems. This essentially provided a new formal model of linear-time computation on general recursive graphs (other models exist as well; details are given in [4]). Hereafter, we shall refer to this predicate calculus as the BPT predicate calculus. Important is that the majority of \mathcal{NP} -complete graph problems (independent set, colorings, Hamiltonian cycle, etc.) are expressible in the BPT predicate calculus and are thus solvable in linear time on recursive graphs.

The BPT predicate calculus is rarely applied in practice due to the amount of detail required. Its role is typically existential in that if a problem is expressible accordingly then its complexity status is settled. Typically, one can obtain the anticipated linear-time algorithm by constructing a small multiplication table h' for each composition operation h . If $G = h(G_1, G_2, \dots, G_m)$, then the table exhibits the outcome for G that corresponds to each m -tuple of compatible subgraph property-tuples for G_1, G_2, \dots, G_m . It is then straightforward to construct the recurrence rela-

tionship and obtain the linear time algorithm by dynamic programming. The interested reader is directed to Richey and Parker [12] and Horton [10] for illustrations of this strategy.

The paper is organized in the following way: The next section establishes that OLA cannot be expressed in the BPT calculus unless $\mathcal{P} = \mathcal{NP}$. Section 3 provides an algorithm that solves OLA on a particular subclass of *Halin* graphs. We end the paper with some concluding remarks and some directions for further research.

2. NONEXPRESSIBILITY OF OLA

Again, so far as we know, the status of OLA on arbitrary recursive graphs remains open. Obviously, if OLA were expressible in a formal way such as in the BPT calculus, this open status would be resolved (in a positive sense). But this seems difficult to do. To establish this, we begin by examining a restricted version of OLA.

An interesting modification to OLA results if we assume that some (possibly empty) subset $\bar{V} \subseteq V$ is pre-labeled from integers in $\alpha \subseteq \{1, 2, \dots, n\}$ and the notion now is to label the remaining vertices in $V \setminus \bar{V}$ with the other, "unused" labels and to do so in an optimal way overall, given the fixed label value imposed by the initial labeling. Calling this version the *partial optimal linear arrangement (POLA)*, it is not at all clear what its status is, even for graph classes where it is trivial to solve OLA. For example, if G is a simple path P_n , we do not know how to solve POLA, but neither do we have an \mathcal{NP} -hardness outcome, even though OLA is trivial on paths. On the other hand, if we are allowed to slightly relax the instances, then we can produce some negative complexity results.

Theorem 1. *POLA is \mathcal{NP} -complete on multipaths, caterpillar trees, and a forest of paths.*

The proof follows a reduction from 3-PARTITION (cf. Garey and Johnson [8]), the details of which appear in Horton et al. [11]. For this note, it suffices to observe that POLA is \mathcal{NP} -complete even if G is restricted to recursive graphs (each of the graphs in Theorem 1 is a partial 2-tree). We can now proceed to prove the main result of this section.

Theorem 2. *OLA is not expressible in the BPT predicate calculus unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Suppose to the contrary that there is a valid BPT expression for OLA and that $\mathcal{P} \neq \mathcal{NP}$. Now, this formal expression for OLA must allow a report of an assignment of integer labels to vertices, but this implies that the fixed portion of an instance of POLA is also expressible in the BPT predicate calculus. Moreover, a valid expression for the restricted OLA instance that creates the POLA instance is easily formed as a conjunction of the expression for OLA and that used to state the fixed portion. Since the set of regular predicates is closed under conjunction in the BPT

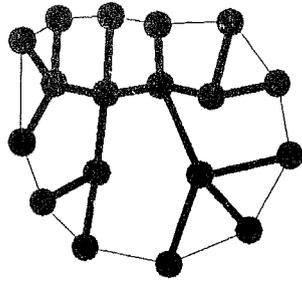


FIG. 1. A Halin graph.

calculus, it must be that POLA is BPT-expressible and, hence, regular, which then means that it is linear-time solvable on recursive graph classes. But, from the result in Theorem 1, this would deny that \mathcal{P} and $\mathcal{N}\mathcal{P}$ are different and we have the desired contradiction. ■

With a few additional restrictions, the result of Theorem 2 can be strengthened to include any such predicate calculus that formalizes polynomial-time solutions on recursive graph instances. The extension is straightforward and left to the reader.

Corollary 3. *If C is a predicate calculus that allows conjunction and any problem that is expressible in C can be solved in polynomial time on recursive graphs, then OLA is not expressible in C unless $\mathcal{P} = \mathcal{N}\mathcal{P}$.*

3. A SOLUTION ON A SUBCLASS OF HALIN GRAPHS

Halin graphs are planar graphs having the property that their edge set E can be partitioned as $E = T \cup C$ with $T \cap C = \emptyset$, where T is a tree and C is a cycle on only and all of the pendant vertices of T . A simple illustration is given in Figure 1 (tree edges are indicated in bold).

The status of OLA on the class of Halin graphs remains open. On the other hand, these structures are contained in the recursive class of *partial 3-trees* (cf. Borie et al. [3]). In this section, we state a procedure for solving the problem on a restricted class of Halin graphs. Consider an easy lemma regarding OLA on arbitrary graphs.

Lemma 4. *Given $G = (V, E)$, let G_1, G_2, \dots, G_m be edge-disjoint subgraphs of G that partition E . Then,*

$$\sum_{i=1}^m L^*(G_i) \leq L^*(G).$$

A *caterpillar* is a tree having the property that if the degree 1 (pendant) vertices are removed the resulting subgraph is a path. Now, suppose that G is a Halin graph where T is a caterpillar. In the following, we state an algorithm for solving OLA on these special graphs (Easton et al. [6]).

Consider the tree component T and label its vertices in the following way: Find a longest path in T , say P , and label the terminal vertices of P by 1 and n . We claim that the edge set $E(T) \setminus E(P)$ is a collection of stars and, accordingly, we then partition the integers $\{2, \dots, n-1\}$ as $\{2, \dots, k_1\}, \{k_1+1, \dots, k_2\}, \dots, \{k_q+1, \dots, n-1\}$ and label each of the $q+1$ stars formed by $E(T) \setminus E(P)$ in an optimal way with the integers in the respective components of the stated partition. Now, with T labeled as described, append $E(C)$. We assume that the vertices on C that extend from those labeled 1 and n are labeled in monotonic fashion. This is a mild assumption since it requires only some care in how the pendant vertices were labeled as part of the labeling for T . In any event, this induced labeling for the vertices of C produces a total labeling for the stated graph G . We will now show that this labeling is optimal.

Let all of the nonpendant vertices of T be collected in a set θ and denote by $n_i = \deg(v_i) - 1$ the order of the i th star formed by $v_i \in \theta$ and its adjacent vertices, other than those in θ . We have

Theorem 5. *Let G be a Halin graph with cycle C and tree component T which is a caterpillar. Then, the labeling produced by the stated algorithm is optimal for G and has value*

$$L^*(G) = 3(n-1) + \sum_{1 \leq i \leq t} \left\lfloor \frac{n_i^2}{4} \right\rfloor.$$

Proof. It is an easy exercise to verify that a correct application of the algorithm yields an arrangement with the stated value. Now, to show that no smaller value exists, let f' be an arbitrary (but admissible) labeling of any graph G in the specified class. Since Halin graphs are minimally three-connected, there are three paths connecting the vertices labeled 1 and n that are vertex disjoint except for the vertices labeled 1 and n and such that one of these paths consists entirely of edges in the tree component of G , while the other two paths have at least one edge that is part of the cycle component of G . Let the subgraph defined by these three paths be G_1 and let $G_2 = G \setminus G_1$. It is trivial to see that $\deg_{G_2}(v) = \deg_G(v) - 2$ unless $f'(v)$ is 1 or n , in which case $\deg_{G_2}(v) = \deg_G(v) - 3$. Observe also that any labeling of G_1 has value $L'(G_1) \geq 3(n-1)$. In the following, we specify two cases:

CASE 1. Assume that $f'(u) = 1$ and $f'(v) = n$, where u or v or both are vertices lying on the spine of the caterpillar tree. For ease, we will consider only vertex u ; the analysis for v is identical. Let us define S_u to be a star in G_2 with u as the hub. Note that S_u may be a single vertex. Now, since $f'(u) = 1$,

$$L'(S_u) \geq \sum_{1 \leq j \leq n_i-2} j = \frac{(n_i-1)(n_i-2)}{2}.$$

