Proceedings of the 8th WSEAS International Conference on APPLIED MATHEMATICS, Tenerife, Spain, December 16-18, 2005 (pp150-152)

A Note on the Complexity of Rooted Trees and Hierarchies with Possible Applications to Organization Design and System Architectures

Peter P. Chen
Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana, 70803
USA

http://www.csc.lsu.edu/~chen

Guoli Ding
Department of Mathematics
Louisiana State University
Baton Rouge, Louisiana, 70803
USA

http://www.math.lsu.edu/~ding

Abstract: We define a complexity function over all rooted trees (i.e., Tree Hierarchies). The purpose of this note is to minimize this function, over all rooted trees on a fixed number of vertices. The results may be useful in the study of organization design and system architectures.

Key-Words: tree complexity, hierarchy complexity, rooted tree, k-ary tree, complete k-ary tree, near-complete k-ary tree, organization design, and system architecture.

1. Introduction

The purpose of this note is to introduce and study a new concept, the *complexity* of a hierarchy.

Many real world structures can be modeled by hierarchies. These examples range from business and/or government organizations to various databases. What we are interested in is how to measure the efficiency of such structures.

In this note, we will limit ourselves to tree hierarchies. That is, we only consider hierarchies that can be represented by rooted tress. This restriction does not lose too much generality since tree hierarchies do represent a big percentage of hierarchies used in real world. On the other hand, this restriction does allow us to present a closed form solution.

Before we formally define the complexity of a rooted tree (a tree hierarchy), we point out that there are two factors that increase the complexity. If the tree has a vertex very far away from the root, then the complexity should be high. This is understandable since passing information from the root to this vertex will take a long time, which means the hierarchy is not very efficient. Similarly, if a vertex has too many immediate descendants, communicating with these descendants would easily overwhelm this vertex, which also reduces the efficiency of the hierarchy.

In the next section, we introduce a definition of the complexity of a rooted tree that takes both factors into consideration. Our main result, Theorem 1, says that "near-complete k-ary trees" (will be defined in the next section) are the most efficient rooted trees. Moreover, our proof implies that every rooted tree can be transformed into an efficient tree by repeatedly applying some local minor modifications.

2. Problem Formulation

Let T be a rooted tree with root r. A vertex v of T is at level ℓ if the unique path from r to v has ℓ edges. The *height* of T, denoted by $h(T)$, is the largest ℓ so that T has a vertex at level ℓ. If $X \subseteq V(T)$ contains r and the restriction of T to X is connected, then we call this restriction a *root subtree* of T, where r is also the root of this subtree.

For each vertex v of T, we denote by $c(v)$ the number of children of v. Let $k \geq 2$ be a fixed positive integer. We call T a k-ary tree if $c(v) \leq k$, for all vertices v of T. A k-ary tree of

height h is complete if $c(v)=k$, for every vertex at a level less than h, and $c(v)=0$, for every vertex v at level h. We denote this tree by $T_{k,h}$. A k-ary tree of height h is near-complete if it is obtained from $T_{k,h}$ by deleting some, could be zero, but not all, vertices at level h.

Let $p_k(x)=x-k$, when $x \geq k$ and let $p_k(x)=0$ when $x \leq k$.

Let $f(T)=h(T)+\sum_{v \in V(T)} p_k(c(v))$, which we call the *complexity function* of T.

Observation 1.
If T is a near-complete k-ary tree on n vertices, then it is straightforward to verify that

$$h(T)=\lceil \log_k (1+n(k-1)) \rceil -1.$$

Theorem 1.
If T is a rooted tree on n vertices, then $f(T) \geq \lceil \log_k (1+n(k-1)) \rceil -1$.

To prove this theorem, we first prove a lemma. Let T be a rooted tree. For i=0,1, let $L_i(T)$ be the set of vertices v with $c(v)=i$.

Lemma 1.
$2|L_0(T)|+|L_1(T)| > |V(T)|$.

Proof.
Let $L_2(T)=V(T)-L_0(T)-L_1(T)$. Since every non-root vertex is a child of a unique vertex,

$$|V(T)|-1=\sum_{v \in V(T)} c(v).$$

Consequently,
$|L_0(T)|+|L_1(T)|+|L_2(T)| > 2|L_2(T)|+|L_1(T)|$, which can be simplified as $|L_0(T)| > |L_2(T)|$, and thus the Lemma follows. ■

Let T be a rooted tree. For each vertex v of T, it is clear that deleting v from T results in exactly $c(v)$ components that contain descendants of v. These components are called *branches* at v. We also define

$$g(T)=\sum\{c(v): v \in V(T) \text{ and } c(v)>k\}.$$

Proof of Theorem 1.
We choose a rooted tree T with the following properties:

(1) f(T) is minimized;
(2) subject to (1), $g(T)$ is minimized;
(3) subject to both (1) and (2), the largest rooted subtree T' of T, such that T' is a near-complete k-ary tree, is maximized.

Clearly, by Observation 1, we need only prove that T is a near-complete k-ary tree, which is equivalent to $T=T'$.

We first prove that T is a k-ary tree. Suppose, on the contrary, that T has a vertex v for which $c(v)=c>k$. Let $T_1, T_2, \ldots, T_c$ be the branches at v. Without loss of generality, we may assume that $|V(T_1)| \leq |V(T_i)|$, for all i. Let S be the rooted tree, with v as its root, that consists of $T_2, T_3, \ldots, T_c$ and v. By Lemma 1, and the choice of T_1, we have
$2|L_0(S)|+|L_1(S)| > |V(S)| > |V(T_1)|$.

Now we modify T as follows. First, we delete all edges that are incident with some vertex in $V(T_1)$. Then we add a new edge from each vertex in $V(T_1)$ to a vertex in $L_0(S) \cup L_1(S)$ such that each vertex in $L_0(S)$ is incident with at most two of these new edges and each vertex in $L_1(S)$ is incident with at most one of these new edges. From the above inequality we know that this is possible.

Let T^* be this new tree. Then $h(T^*) \leq h(T)+1$ and v has exactly one child less in T* than it has in T. Moreover, for every other vertex u, either $c(u)$ does not change or u has at most two, with is less than or equal to k, children in T^*. Therefore, $f(T) \geq f(T^*)$ and $g(T)>g(T^*)$, contradicting either (1) or (2). This contradiction proves that T is a k-ary tree.

It remains to prove that $T'=T$. Since T' is a near-complete k-ary tree and it is a rooted subtree of the k-ary tree T, it follows that every vertex of T at level $\ell \leq h(T')$ belongs to T'.

Suppose, on the contrary, that $T' \neq T$. Then T must have a vertex u at level $h(T')+1$. By the maximality of T', adding u to T' does not result in a near-complete k-ary tree. This means that T' is not a complete k-ary tree, which implies that some vertex v of T at level $h(T')-1$ has fewer than k children. Now we

modify T as follows. Take a vertex w at level $\ell > h(T')$ with $c(w)=0$. Delete the only edge incident with w and add a new edge vw. Let R be this new rooted tree. It is clear that $f(T) \geq f(R)$ and $g(T)=g(R)$. However, if we take R' to be the rooted subtree of R that consists of T' and w, then R' is a near-complete k-ary tree, which is bigger than T', contradicting (3). This contradiction proves that $T=T'$, which completes the proof of Theorem 1. ■

3. Possible Applications and Extensions

The definition of "complexity" in this paper is somewhat different than definitions of complexity by other researchers [1-5]. We think our definition will be useful in the study of optimal organization structure and also the costs of migrating from one organization structure to another.

Another possible extension and application is to develop mathematical framework including a set of algebraic operations, which will be useful in studying different alternatives of systems architectures [6-11]

Acknowledgment. This research was partially supported by U.S. NSF grant DMS-9970329, NSF Grant IIS- 0326387, and U.S. AFOSR Grant No. F49620-03-1-0239, F49620-03-1-0238, F49620-03-1-0241, F49620-01-1-0264, and FA9550-05-1-0454.

References:

[1] Codynamics, "Introduction to the Basic Concepts of Complexity Science," in http://www.codynamics.net/intro.htm, 2004.

[2] J. Collier, "Organized Complexity: Properties, Models, and Limits of Understanding," in http://www.nu.ac.za/undphil/collier/papers/cuba-complexity.pdf, 2004.

[3] E. E. Olson, Glenda H. Eoyang, "Facilitating Organization Change: Lessons from Complexity Science," First ed: Jossey-Bass/Pfeiffer, February 7, 2001.

[4] M. Lissack, "Michael Lissack's Publications," in http://lissack.com/writings/, 2004.

[5] T. Petzinger, "Complexity Reading List," in http://www.petzinger.com/complexity.shtml, 2004.

[6] P. Chen, and Guoli Ding, "Unavoidable double-connected large graphs," *Discrete Mathematics*, 2004.

[7] P. Chen, "A Preliminary Framework for Analyziing Critical Issues in Engineering Systems," presented at MIT Symposium on Engineering Systems, http://esd.mit.edu/symposium/pdfs/papers/chen-abst.pdf, Cambridge, MA, 2004.

[8] J. Moses, "Three Design Methodologies, Their Associated Organizational Structures and Their Relationships to Various Fields," presented at MIT Symposium on Engineering Systems, http://esd.mit.edu/symposium/pdfs/papers/moses.pdf, Cambride, MA, 2004.

[9] P. Chen, "The entity-relationship model: Toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, 1976.

[10] P. Chen, and Guoli Ding, "Generating r-regular graphs," *Discrete Applied Mathematics*, vol. 129, pp. 329-343, 2003.

[11] P. Chen, and Guoli Ding, "The Best Expert Versus the Smartest Algorithm," *Theoretical Computer Science*, Vol. 332, No. 1-3, (2005), pp. 63-81.