Online tree node assignment with resource augmentation

Joseph Wun-Tat Chan · Francis Y.L. Chin · Hing-Fung Ting · Yong Zhang

Published online: 5 February 2010
© Springer Science+Business Media, LLC 2010

Abstract Given a complete binary tree of height $h$, the online tree node assignment problem is to serve a sequence of assignment/release requests, where an assignment request, with an integer parameter $0 \leq i \leq h$, is served by assigning a (tree) node of level (or height) $i$ and a release request is served by releasing a specified assigned node. The node assignments have to guarantee that no node is assigned to two assignment requests unreleased, and every leaf-to-root path of the tree contains at most one assigned node. With assigned node reassignments allowed, the target of the problem is to minimize the number of assignments/reassignments, i.e., the cost, to serve the whole sequence of requests. This online tree node assignment problem is fundamental to many applications, including OVSF code assignment in WCDMA networks, buddy memory allocation and hypercube subcube allocation.

Most of the previous results focus on how to achieve good performance when the same amount of resource is given to both the online and the optimal offline algorithms, i.e., one tree. In this paper, we focus on resource augmentation, where the online algorithm is allowed to use more trees than the optimal offline algorithm. By
using different approaches, we give (1) a 1-competitive online algorithm, which uses \((h + 1)/2\) trees and is optimal because \((h + 1)/2\) trees are required by any online algorithm to match the cost of the optimal offline algorithm with one tree; (2) a 2-competitive algorithm with \(3h/8 + 2\) trees; (3) an amortized \(8/3\)-competitive algorithm with \(11/4\) trees; (4) a general amortized \((4/3 + \alpha)\)-competitive algorithm with \((11/4 + 4/(3\alpha))\) trees, for \(0 < \alpha \leq 4/3\).

**Keywords** Online algorithms · Tree node assignment · Resource augmentation

## 1 Introduction

The tree node assignment problem is defined as follows. Given a complete binary tree of height \(h\), the target is to serve a sequence of requests. Every request is classified as either an assignment request or a release request. To serve an assignment request, which is associated with an integer parameter \(0 \leq i \leq h\), we have to assign it a (tree) node at level (or height) \(i\). To serve a release request, we just need to mark a specified assigned node free. There are two constraints for the node assignments, which are (1) any node can be assigned to at most one unreleased assignment request, (without ambiguity, all assigned requests are assumed unreleased), and (2) there is at most one assigned node in every leaf-to-root path. Figure 1 gives a valid tree node assignment.

The tree node assignment problem can be considered as a general resource allocation problem, which can model the specific problems, such as the Orthogonal Variable Spreading Factor (OVSF) code assignment problem (Chin et al. 2007, 2008; Erlebach et al. 2007; Li and Wan 2005; Minn 2000; Miyazaki and Okamoto 2008; Rouskas and Skoutas 2002; Wan et al. 2007), the buddy memory allocation problem (Brodal et al. 2005; Defoe et al. 2005; Knowlton 1965; Knuth 1975), and the hypercube subcube allocation problem (Dutt 1991). The main difference between these problems is how the resource, the nodes at level \(i\), for \(0 \leq i \leq h\), are interpreted. In the OVSF code assignment problem, the resource consists of codes of frequency bandwidth \(2^i\); in the buddy memory allocation problem, the resource consists of memory blocks of size \(2^i\); in the hypercube subcube allocation problem, the resource consists of subcubes of \(2^i\) processors.

Similarly to the memory allocation problem, algorithms for the tree node assignment problem also face the fragmentation problem. For example, in Fig. 1, there is no node of level 2 that can be assigned (without violating constraint (2) of node assignments). In fact, we can “defragment” the tree by reassigning the assigned node \(c\) to free node \(f\). Then, node \(a\) is free to be assigned to an assignment request of level 2. In this paper, we consider the tree node assignment problem where reassignments of nodes are allowed. In addition, we design algorithms that serve all requests in the request sequence, and we assume that all requests in the request sequence can be served by some algorithm using only one tree of height \(h\).

The performance of an algorithm for the tree node assignment problem is measured by the number of assignments/reassignments, which is called the cost, carried out by the algorithm. The release operations take no cost, as in the applications, the operation to release a resource is usually negligible when compared with the overhead of assigning/reassigning a resource.