An Optimal Balanced Partitioning of a Set of 1D Intervals

Chuan-Kai Yang, National Taiwan University of Science and Technology, Taiwan

ABSTRACT

Given a set of 1D intervals and a desired partition number, in this paper, the author examines how to make an optimal partitioning of these intervals, such that the number of intervals between the largest partition and smallest partition is minimal among all possible partitioning schemes. This problem has its difficulty due to the fact that an interval “striding” multiple partitions should be counted multiple times. Previously the author proposed an approximated solution to this problem by employing a simulated annealing approach (Yang & Chiueh, 2006), which could give satisfactory results in most cases; however, there is no theoretical guarantee on its optimality. This paper proposes a method that could both optimally and deterministically partition a given set of 1D intervals into a given number of partitions. The author shows that some load balancing problems could also be formulated as a balanced interval partitioning problem.

Keywords: Computational Geometry, Dynamic Programming, Interval Partitioning, Multiple Partitions, Partitioning Schemes

1. MOTIVATION AND INTRODUCTION

How to partition a set of 1D intervals into balanced partitions is what to be discussed in this paper, and such a problem originally arisen from dealing with efficient volume visualization. Volume visualization, sometimes deemed a sub-field of computer graphics, aims to provide 2D images, or more specifically, 2D visualizations, from a so called 3D volume dataset, which is normally composed of a stack of 2D images. Instead of distributing data values on 2D grid points like a 2D image, a volume dataset generalizes the distribution to 3D grid points. Iso-surface extraction is one of the most popular methods to visualize a volume dataset, where an iso-value is first given, and then the corresponding iso-surface, representing the surface intersecting with the given value, is extracted and represented by a set of interpolated triangles. These triangles could later be sent to a graphics card to be rendered into images. As technology advances, nowadays data acquisition devices tend to collect or generate data with higher and higher precision, thus leading to volume datasets of huge sizes. The visualization of a huge volume dataset usually would cause problems as such a task normally needs to bring in the whole dataset into the

DOI: 10.4018/jalr.2010040106

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
main memory before any further processing could start. As the size of a huge volume dataset could be more than several GBytes, the scenario of an out-of-core processing arises (Cox & Ellsworth, 1997; Cox, 1997; Ueng et al., 1997), which means the required memory for data processing is larger than a system could provide. To address this, data partitioning is one popular approach, such as the method proposed by Chiang et al. (1998). Since volume visualization is often the most important purpose that a dataset was collected or generated in the first place, it should be taken into account when dealing with data partitioning. In Chiang et al.’s (1998) approach, they partition a volume dataset spatially so that each resulting partition could be loaded into the main memory and processed individually without incurring any virtual memory overheads, namely, memory swappings or pagers. The extracted iso-surface is an aggregate result from each partition. Such a paradigm, though simple, has the following drawback: unless some clever pre-processing is employed, all the partitions need to be examined for iso-surface intersection. We argue that a different partitioning scheme may sometimes offer better processing efficiency. That is, instead of partitioning a volume dataset based on spatial information, we could decompose a dataset into layers based on the data values associated with the grid points, and each layer represents a portion of data bearing a particular value range. Such a partitioning has the advantage that for an iso-value query, at most one layer is needed for extracting the iso-surface, thus making it an attractive scheme. Furthermore, the result is a continuous or integrated representation, rather than a segmented one faced in the case of a spatial partitioning, where a post stitching or merging may be needed for smoothing or reducing the rendering traffic sent to graphics cards afterward. However, there are also two issues associated with this kind of partitioning approach. First, it is not suitable for a volume dataset with high data variation, such as some of the datasets obtained from CFD (computational fluid dynamics) simulation, since a value-based partitioning may lead to decompositions that are too fragmented, thus incurring extra representation overheads. Second, how to make a balanced partitioning is in fact more difficult that it seems, as will be explained in detail in later sections. For the ease of ensuing discussion, we will refer to the first type of partitioning a space-based or coordinate-based scheme, while the second type of partitioning a value-based or layer-based scheme. Previously (Yang & Chiueh, 2006) have shown that how a balanced (with a slightly different definition of balance) partitioning could be achieved by a simulated annealing approach. Though efficient, such a soft optimization method can neither guarantee the optimality nor provide a theoretical analysis of the result. In this work, we provide a dynamic programming approach that could find an optimal solution to this problem.

Applications of this balanced interval partitioning problem are not limited to volume visualization only, as some types of load balancing problems could also be formulated as such a problem. For example, considering the following load balancing problem, where a number of activities are to be held in a hall, and each activity has its own specific starting time and duration. Now assuming there are only a fixed number of janitors available to assist these activities, a reasonable arrangement is to partition these activities into groups so that each janitor is responsible for roughly the same number of activities. Some balanced task scheduling problems could also be formulated in a similar fashion, thus making solutions to this problem more important.

The rest of the paper is organized as the following. Section 2 reviews the literature related to this work. Section 3 details our proposed algorithm. Section 4 analyzes the time complexity of our proposed approach. Section 5 concludes this work and points out some potential future directions.

2. RELATED WORK

As explained in the Introduction Section, the problem of balanced 1D interval partitioning
Artificial Higher Order Neural Network Models
www.igi-global.com/chapter/artificial-higher-order-neural-network-models/161032?camid=4v1a

Using Cuckoo Search Algorithm for Hybrid Flow Shop Scheduling Problems Under Makespan Criterion
www.igi-global.com/chapter/using-cuckoo-search-algorithm-for-hybrid-flow-shop-scheduling-problems-under-makespan-criterion/198934?camid=4v1a

GA on IR: Study the Effectiveness of the Developed Fitness Function on IR
www.igi-global.com/article/study-effectiveness-developed-fitness-function/74332?camid=4v1a