Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 61 (2007) 563–578
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

pPOP: Fast yet accurate parallel hierarchical clustering using partitioning

Manoranjan Dash [a,*], Simona Petrutiu [b], Peter Scheuermann [b]

[a] School of Computer Engineering, Nanyang Technological University, Blk N4, #2c-85, Nanyang Avenue, Singapore 639798, Singapore
[b] Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208, United States

Received 11 January 2006; received in revised form 5 July 2006; accepted 6 July 2006
Available online 8 August 2006

Abstract

Hierarchical agglomerative clustering (HAC) is very useful but due to high CPU time and memory complexity its practical use is limited. Earlier, we proposed an efficient partitioning – partially overlapping partitioning (POP) – based on the fact that in HAC small and closely placed clusters are agglomerated initially, and only towards the end larger and distant clusters are agglomerated. Here, we present the parallel version of POP, pPOP. Theoretical analysis shows that, compared to the existing algorithms, pPOP achieves CPU time speed-up and memory scale-down of O(c) without compromising accuracy where c is the number of cells in the partition. A shared memory implementation shows that pPOP outperforms existing algorithms significantly.

Keywords: Hierarchical agglomerative clustering; Partitioning; Parallel algorithm; Shared memory multiprocessor architecture

1. Introduction

Clustering or grouping of similar objects is one of the most widely used techniques in data mining. It is an important data exploration task used in diversified areas including market segmentation in business, gene categorization in biology, spatial discovery, and document classification on the WWW. Some surveys and books on clustering are found in [14,16,17]. A popular method of clustering is hierarchical agglomerative clustering (HAC). It starts with each point in separate clusters and iteratively agglomerates the closest pair of clusters in each iteration until all points belong to a single cluster. The final hierarchical cluster structure is called a *dendrogram* (see Fig. 1), a tree like structure that shows which clusters are agglomerated at each level. To obtain the best level and its corresponding clusters, each level of the dendrogram can be evaluated by a cluster validation method [21].

HAC algorithms are non-parametric,[1] natural and simple in grouping objects, and capable of finding clusters of different shapes by using different similarity measures. However, they have several drawbacks. First,

* Corresponding author. Tel.: +65 6790 6167; fax: +65 6792 6559.
E-mail address: asmdash@ntu.edu.sg (M. Dash).
[1] It does not assume any statistical distribution.

0169-023X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2006.07.004

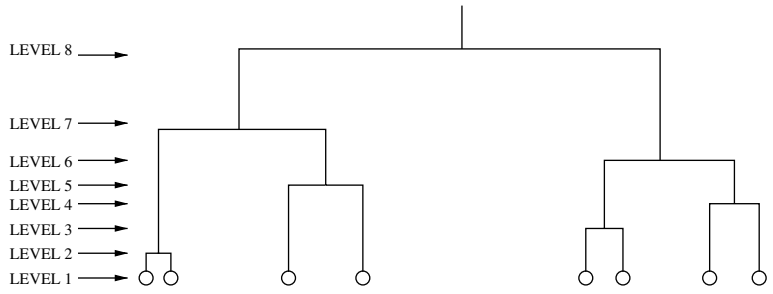

Fig. 1. Dendrogram: output of hierarchical clustering.

HAC algorithms have high CPU time and memory complexities. For example, an efficient algorithm that uses a stored matrix representation based on priority queues and a geometric metric for distances based on the centroid method has time complexity $O(N^2 \log N)$ [10]. We observe here that a number of techniques were proposed to expand HAC to large datasets, such as BIRCH [30] and CURE [13]. However, *they do not make the HAC algorithm itself faster*. Instead, they use approximations (e.g., summarized (BIRCH) or sampled (CURE) points) so that HAC can operate on a smaller dataset.

In [6,7,9] we have shown that the complexities of the existing sequential HAC algorithms can be reduced significantly by an efficient partitioning scheme without compromising accuracy. The proposed methods are based on the observation that in HAC most iterations agglomerate very small clusters separated by a very small dissimilarity. Only a small number of iterations towards the end agglomerate the large clusters. Using this observation, a structure called *p*artially *o*verlapping *p*artitioning (POP) divides the data into a number of overlapping cells. Analyses and experiments showed that POP-based sequential HAC algorithms reduce the existing time and memory complexities by a factor close to the number of cells c – appropriately defined later.

To cluster increasingly massive datasets that are common in today's applications, a number of parallel algorithms have been proposed. For example see [11,19,22–24,29]. These algorithms either parallelize other types of clustering methods such as *K*-means [11], or perform subspace clustering [22]. The parallelizations of hierarchical clustering are limited to the divisive algorithm reported in [24] or to HAC algorithms that are not very efficient due to the lack of performance enhancing partitioning [19,20,23,29].

In this paper we present parallel versions of POP, called pPOP that are geared towards a shared memory multiprocessor architecture.[2] We present the pPOP version in detail for stored matrix using priority queue and compare it with the traditional parallel implementation. Since our implementation is done in OpenMP we give the details about scheduling, data dependencies and race conditions, as appropriate. We then proceed to show theoretically that pPOP, just like its sequential counterpart POP, reduces the time and memory complexities of the existing parallel algorithms by a factor close to the number of cells c for stored matrix versions of the algorithms. Next, we give results of our experimental evaluation over synthetic and benchmark datasets of pPOP versions for both stored matrix and stored data. The results show that for large datasets the pPOP algorithms obtain near linear speedup. In addition, we show that for stored matrix implementations, pPOP results in multiple orders of magnitude improvement in computation time and memory requirements over the existing parallel HAC algorithms.

In summary, parallel HAC algorithms can be improved by using an efficient partitioning scheme without compromising their accuracy. Although POP partitioning is independent of the parallel architecture (be it shared memory or distributed memory as in MPI – Message Passing Interface), we foresee that distributed memory architecture with its numerous communication overhead will not be as advantageous as the shared memory architecture. Due to fine grain parallelism the shared memory implementation should perform better than the distributed memory.

The paper is organized as follows. The next section gives background information that includes a discussion about an important property of HAC, a brief description of existing HAC algorithms, and POP partitioning. In Section 3 we introduce parallel versions of POP and give theoretical results for time and memory complexities

[2] A shorter version of this paper appeared in EuroPar 2004 proceedings [8].