

# Optimal Implementations of UPGMA and Other Common Clustering Algorithms

Ilan Gronau      Shlomo Moran

May 2, 2007

## Abstract

In this work we consider hierarchical clustering algorithms, such as UPGMA, which follow the closest-pair joining scheme. We study optimal  $O(n^2)$ -time implementations of such algorithms which use a ‘locally closest’ joining scheme, and specify conditions under which this relaxed joining scheme is equivalent to the original one (i.e. ‘globally closest’).

**Key Words:** Hierarchical clustering, UPGMA, design of algorithms, input-output specification, computational complexity

## 1 Introduction

Hierarchical clustering algorithms were already developed in the 1960’s. They are used in numerous applications such as pattern recognition, computational biology and data mining. These algorithms deal today with a vast amount of input, as introduced for instance by high throughput experiments in biology [10] and by data mining of the world wide web [12]. Throughout the years, extensive research has been done dealing with efficient implementations for such algorithms [19, 15, 4, 9, 7, 17, 6, 1]. In this paper we focus on a common class of hierarchical clustering algorithms, which we call *Globally Closest Pair* (or GCP) clustering algorithms. Specifically, we discuss several known techniques for implementing such algorithms in asymptotically optimal time.

A general scheme for GCP clustering algorithms is described in Table 1. Such algorithms receive as input a set of elements and a matrix containing all dissimilarities between element-pairs, and return a *hierarchy* of clusters over this set. Initially, a singleton-cluster is defined for every element in the set. Clustering then proceeds with the main loop, in which at each iteration two clusters of **minimal dissimilarity** are selected and replaced by their union. The following iteration then continues with the smaller cluster-set. In each such iteration the dissimilarities between the new cluster and all other clusters need to be determined (Step 3). This is typically done using some *reduction formula* to compute new dissimilarities using old ones.

Different variants of GCP clustering algorithms are determined by the specification of the reduction in Step 3. UPGMA (Unweighted Pair Grouping Method

---

**Input:** A dissimilarity matrix  $D$  over a set of elements  $S$ .

**Output:** A cluster-hierarchy  $\mathcal{H}$  over  $S$ .

**Initialization:** Initialize the cluster-set  $\mathcal{C}$  by defining a singleton cluster  $C_i = \{i\}$  for every element  $i \in S$ . Initialize output hierarchy  $\mathcal{H} \leftarrow \mathcal{C}$ .

**Loop:** While  $|\mathcal{C}| > 1$  do:

1. **Cluster-pair selection:** Select a pair of distinct clusters  $\{C_i, C_j\} \subseteq \mathcal{C}$  of minimal dissimilarity under  $D$  (ties are broken arbitrarily).
2. **Cluster-pair joining:** Remove  $C_i, C_j$  from the cluster set  $\mathcal{C}$  and replace them with  $C_i \cup C_j$ . Add  $C_i \cup C_j$  to the hierarchy  $\mathcal{H}$ .
3. **Reduction:** Calculate the dissimilarity  $D(C_k, (C_i \cup C_j))$  for every  $C_k \in \mathcal{C}' \setminus \{C_i \cup C_j\}$ .

**Finalize:** Return the hierarchy  $\mathcal{H}$ .

---

Table 1: A general scheme for ‘globally closest pair’ clustering algorithms.

with Arithmetic-mean) is one of the most commonly used variants. It defines the dissimilarity between clusters as their average dissimilarity (hence its name). This is achieved by using the following reduction formula:

$$D(C_k, (C_i \cup C_j)) \leftarrow \frac{|C_i|}{|C_i| + |C_j|} D(C_k, C_i) + \frac{|C_j|}{|C_i| + |C_j|} D(C_k, C_j) \quad (1)$$

Other common GCP clustering algorithms, such as WPGMA [20] and the single linkage algorithm [2, 19], are determined by different reduction formulae:

$$\textbf{WPGMA:} \quad D(C_k, (C_i \cup C_j)) \leftarrow \frac{1}{2} (D(C_k, C_i) + D(C_k, C_j)) \quad (2)$$

$$\textbf{Single-Linkage:} \quad D(C_k, (C_i \cup C_j)) \leftarrow \min \{D(C_k, C_i), D(C_k, C_j)\} \quad (3)$$

Most GCP algorithms mentioned in the literature (see [15, 16]) use a *convex* reduction formulae in step 3, meaning that the value set for  $D(C_k, (C_i \cup C_j))$  lies between  $D(C_k, C_i)$  and  $D(C_k, C_j)$ . We denote such algorithms as *convex GCP algorithms*. Some works (e.g. [15, 4, 5]) consider a weaker property of the reduction step, called the *reducibility property*: if  $D(C_i, C_j) \leq \min\{D(C_k, C_i), D(C_k, C_j)\}$ , then  $\min\{D(C_k, C_i), D(C_k, C_j)\} \leq D(C_k, (C_i \cup C_j))$ . It is easy to see that convex GCP algorithms satisfy the reducibility property. Whenever convexity is assumed in this paper, it is actually sufficient to assume reducibility.

Due to their wide applicability, several attempts were made to implement GCP clustering algorithms in  $O(n^2)$  time, which is the obvious lower bound. Eppstein [9] achieves this goal by using a data structure called *quadtrees*, which

enables performing the following operations in  $O(n)$  time: insertion/deletion of an element and computation of a closest pair. Other implementations of GCP clustering algorithms, which are more common in practice, use simpler data structures that allow  $O(n^2)$  implementations only in some special cases [19, 4, 7].

In this paper we focus on an approach which reduces running time by relaxing the selection criterion used in step 1: instead of selecting a *globally closest* cluster-pair in each iteration, the chosen pair is only required to be *locally closest*, meaning that each of the two clusters is closest to the other, but their dissimilarity is not necessarily minimal among all pairwise dissimilarities. We term this relaxed selection scheme the *locally closest pair* (LCP) scheme. A technique which implements convex LCP clustering algorithms in  $O(n^2)$  time was presented in [3, 13, 15]. Recently we used a similar technique in a different context in [11]. While this optimal implementation is less general than the one in [9], it is considerably simpler and uses only elementary data structures, which makes it particularly appealing.

A natural question raised by this relaxation in the clustering scheme is whether the resulted LCP algorithms are *equivalent* to the original GCP algorithms, in the sense that they have the same input-output relation. This question was already addressed in [15, 16]. It is informally argued there that both schemes are equivalent when a convex reduction formula is used. However, as we demonstrate in Section 3, convexity alone is insufficient for implying this equivalence. The main contribution of this paper is a formulation of conditions under which an LCP clustering algorithm is equivalent to the GCP clustering algorithm which uses the same reduction. We also show that whereas these conditions hold for the reduction formulae used by UPGMA (1) and WPGMA (2), a seemingly minor change (which preserves convexity) in the reduction formula can violate this equivalence.

The rest of this paper is organized as follows. In Section 2 we review relevant implementations of GCP and LCP clustering algorithms, including the simple  $O(n^2)$  implementation of LCP algorithms from [15]. In Section 3 we discuss the equivalence of LCP and GCP clustering algorithms and present conditions on the reduction formula under which this equivalence holds.

## 2 Efficient Implementations of Clustering Algorithms Using Nearest Neighbors

In this section we review several known implementations of GCP and LCP clustering algorithms. These algorithms receive an input dissimilarity matrix  $D$  over a set  $S$  of  $n$  elements, and perform  $n - 1$  iterations as outlined by Table 1. Each such iteration involves joining a cluster-pair and reducing the input matrix. For all commonly used reduction formulae (see [15]), the reduction step is easily implemented in  $O(n)$  time. Thus, the running time of the algorithm is typically dominated by the time required for selecting the cluster-pairs. A

naive approach, which requires  $\theta(n^2)$  time in **each iteration** (and a total time complexity of  $\theta(n^3)$ ), scans all possible cluster-pairs to find a globally closest pair. Faster implementations are obtained by maintaining *nearest-neighbors* for all clusters in the current set.

Cluster  $C_j$  is said to be a nearest-neighbor (NN) of  $C_i$  if it is closest to it. Given the dissimilarity matrix  $D$ , finding such a cluster takes  $O(n)$  time. Once a NN is kept for each cluster, a globally closest cluster-pair is found in  $O(n)$  time by selecting a pair  $(C_i, C_j = NN(C_i))$  of minimal dissimilarity. Thus when using nearest-neighbors, the complexity of the algorithm is actually determined by the time it takes to recompute NNs over the reduced cluster set. In particular, NNs should be recalculated for every cluster  $C_k$  previously having one of the joined clusters  $(C_i, C_j)$  as its NN. In general, this may result in  $\Omega(n)$  NN updates during each iteration (see discussion in [15, 16]). By maintaining the entries of each row of  $D$  in a heap, these updates can be made in  $O(n \log(n))$  time, resulting in a general  $O(n^2 \log(n))$  implementation.

$O(n^2)$  implementations of GCP algorithms using nearest-neighbors are known in some special cases. Day and Edelsbrunner [4] show (using geometrical considerations) that when the input set  $S$  consists of points in a bounded-dimension space, the number of NN updates required in each iteration is bounded by a constant. This naturally leads to time complexity of  $O(n^2)$  (for all common reduction formulae). In the special case of the single-linkage algorithm, there is a well known  $O(n^2)$  implementation [19], which uses the following observation: if either  $C_i$  or  $C_j$  was a NN of  $C_k$  before their joining, then  $C_i \cup C_j$  is a NN of  $C_k$  after the reduction. The same idea is also used in the fast variant of UPGMA implemented in MUSCLE [7]. However, since the above observation does not hold for the reduction formula of UPGMA, the implied algorithm is not equivalent to UPGMA, as it is guaranteed to yield the same output as UPGMA only on a **limited** set of inputs (e.g. *ultrametrics* [14]). In some sense, the same approach is also taken in the recent fast version of Saitou and Nei's Neighbor Joining algorithm [18, 8].

The techniques used in [4, 7] do not imply, therefore, an  $O(n^2)$  algorithm which is equivalent to UPGMA. However, as shown in [15, 11], a reduction in time complexity can be obtained by relaxing the selection criterion of the algorithm to join in each stage a **locally** (rather than globally) closest cluster-pair. Such cluster pairs are referred to in [15, 16] as *reciprocal* (or *mutual*) *nearest-neighbors* (RNNs). Finding RNNs can be easily done by maintaining a *complete nearest-neighbor chain*<sup>1</sup>. A sequence of distinct clusters  $P = (C_{i_1}, C_{i_2}, \dots, C_{i_l})$  is a *nearest-neighbor chain* (with respect to  $D$ ) if  $C_{i_{r+1}} = NN(C_{i_r})$  for all  $1 \leq r < l$ . A NN-chain is said to be *complete* if it contains at least two clusters, and the last two clusters are RNNs. We conclude this section by reviewing the nearest-neighbor-chain technique [15] which leads to optimal  $O(n^2)$  time complexity.

The algorithm starts with an arbitrary cluster and extends a NN-chain starting from it (by computing NNs) until this chain is complete. Given a complete

---

<sup>1</sup>Similar chains are referred to in [11] as *complete ascending paths*.

NN-chain  $P = (C_{i_1}, \dots, C_{i_l})$ , the algorithm joins  $C_{i_{l-1}}$  and  $C_{i_l}$  (as they are ‘locally closest’), performs a reduction of the dissimilarity matrix, and removes  $C_{i_{l-1}}, C_{i_l}$  from the NN-chain. If the chain was consequently emptied, it initializes the chain with some arbitrary cluster. The important observation to be made here is that if the reduction is **convex**, the remaining chain is a valid NN-chain corresponding to the reduced dissimilarity matrix. Thus a complete NN-chain  $P'$  can be obtained by iteratively extending this chain. Extending the NN-chain by a single cluster and checking whether it is complete takes  $O(n)$  time. This operation is invoked no more than  $2(n-1)$  times overall, since every cluster added to the chain (except one) is removed from it at some point, and only 2 clusters are removed during each iteration. Thus the resulting total time-complexity is  $O(n^2)$ .

### 3 Equivalence of the LCP and GCP Schemes

In the previous section we described the NN-chain technique for implementing convex LCP clustering algorithms in  $O(n^2)$  time. In this section we study the properties of the reduction formula under which an LCP clustering algorithm is guaranteed to be equivalent to the corresponding GCP algorithm. In order to demonstrate equivalence for a given reduction formula, we need to show that every clustering obtained by the corresponding LCP algorithm can also be obtained on the same input using the corresponding GCP algorithm. To simplify the presentation in our analysis below, the set  $S$ , over which the input dissimilarities are defined, is viewed as a set of singleton clusters ( $\mathcal{C}$ ).

**Lemma 3.1.** *Let  $D$  be a dissimilarity matrix over a cluster set  $\mathcal{C}$ , s.t.  $|\mathcal{C}| > 1$ . Then, when executed on  $(\mathcal{C}, D)$ , a convex LCP clustering algorithm joins (at some point) a cluster-pair  $\{C_i, C_j\} \subseteq \mathcal{C}$ , which is globally closest under  $D$ .*

*Proof.* The lemma is proved by induction on  $|\mathcal{C}|$ . It holds trivially when  $|\mathcal{C}| = 2$ . Assume that  $|\mathcal{C}| > 2$ , and let  $d_{\min}$  denote the minimal pairwise dissimilarity in  $D$ . Consider the first cluster-pair  $\{C_k, C_l\}$  joined by the algorithm. If  $D(C_k, C_l) = d_{\min}$ , the lemma follows immediately. Otherwise,  $D(C_k, C_l) > d_{\min}$ , and since  $C_k$  and  $C_l$  are RNNs, all pairwise dissimilarities in  $D$  involving either  $C_k$  or  $C_l$  are strictly greater than  $d_{\min}$ . Now let  $D'$  denote the reduced dissimilarity matrix over the reduced cluster-set  $\mathcal{C}' = \mathcal{C} \setminus \{C_k, C_l\} \cup \{(C_k \cup C_l)\}$ . Since the reduction is convex, all pairwise dissimilarities in  $D'$  involving  $C_k \cup C_l$  are also strictly greater than  $d_{\min}$ . Furthermore,  $d_{\min}$  is still the minimal pairwise dissimilarity in  $D'$ . Therefore, a cluster pair  $\{C_i, C_j\} \subseteq \mathcal{C}'$  is globally closest under  $D'$  iff it is globally closest under  $D$ . The induction hypothesis on  $(\mathcal{C}', D')$  implies that at some point in the execution, a globally closest cluster-pair  $\{C_i, C_j\} \subseteq \mathcal{C}'$  must be joined. By the previous observation,  $\{C_i, C_j\}$  is also globally closest under  $D$ .  $\square$

Informally, in order to complete the proof of equivalence, we need to show that the joining of the globally closest pair guaranteed by Lemma 3.1 can be

moved to the beginning of the execution without affecting the output. This can be achieved when the reduction formula satisfies an additional property – *commutativity* – which is defined below.

**Definition 3.2** (Commutativity of reduction formula). *A reduction formula  $F$  is said to be commutative, if the following holds for every dissimilarity matrix  $D$  over a cluster set  $\mathcal{C}$  (s.t.  $|\mathcal{C}| \geq 4$ ), when using reduction formula  $F$ : Given four arbitrary clusters  $\{C_{i_1}, C_{j_1}, C_{i_2}, C_{j_2}\} \subseteq \mathcal{C}$ , the dissimilarity matrix obtained by first joining  $\{C_{i_1}, C_{j_1}\}$  and then joining  $\{C_{i_2}, C_{j_2}\}$  (according to steps 2,3 in Table 1) is equal to the dissimilarity matrix obtained by first joining  $\{C_{i_2}, C_{j_2}\}$  and then joining  $\{C_{i_1}, C_{j_1}\}$ .*

**Discussion:** The commutativity property holds trivially when the reduction formula  $F$  induces a dissimilarity function  $D_F$  over the set of all clusters, s.t.  $D_F(C_1, C_2)$  depends only on the dissimilarities between elements in  $C_1 \cup C_2$ . Such are for instance the reduction formulae for UPGMA (1) and the single linkage algorithm (3):

$$D_{UPGMA}(C, C') = \frac{1}{|C||C'|} \sum_{i \in C, j \in C'} D(i, j) \quad (4)$$

$$D_{single-linkage}(C, C') = \min_{i \in C, j \in C'} \{D(i, j)\} \quad (5)$$

Some common reduction formulae, such as the one used by WPGMA (2), do not induce such a function. In fact, it is possible for two executions of WPGMA on the **same input** to lead to different reduced dissimilarities (between the same clusters). Nevertheless, it is not hard to see that the reduction formula of WPGMA is in fact commutative.

Convex reduction formulae are not necessarily commutative, as demonstrated by the following reduction formula:

$$D(C_k, (C_i \cup C_j)) \leftarrow \left[ \frac{|C_i|}{|C_i| + |C_j|} D(C_k, C_i) + \frac{|C_j|}{|C_i| + |C_j|} D(C_k, C_j) \right] \quad (6)$$

This formula is identical to the one used by UPGMA, apart from the fact that it rounds up the result. It is clearly convex, but as illustrated in Figure 1, the LCP algorithm which uses this formula is not equivalent to its GCP counterpart. The example described in Figure 1 details a specific execution of the LCP algorithm which leads to different clustering than the one produced by the (unique) execution of the GCP algorithm on the same input. The different output is a direct result of the non-commutativity of the above reduction formula.

Finally, we show that convexity and commutativity imply the desired equivalence between the LCP and GCP clustering schemes:

**Theorem 3.3.** *Let  $D$  be an arbitrary dissimilarity matrix over a cluster-set  $\mathcal{C}$ , and let  $F$  be a convex and commutative reduction formula. Then every execution (on  $(\mathcal{C}, D)$ ) of the LCP clustering algorithm which uses  $F$  has an equivalent execution of the corresponding GCP algorithm.*

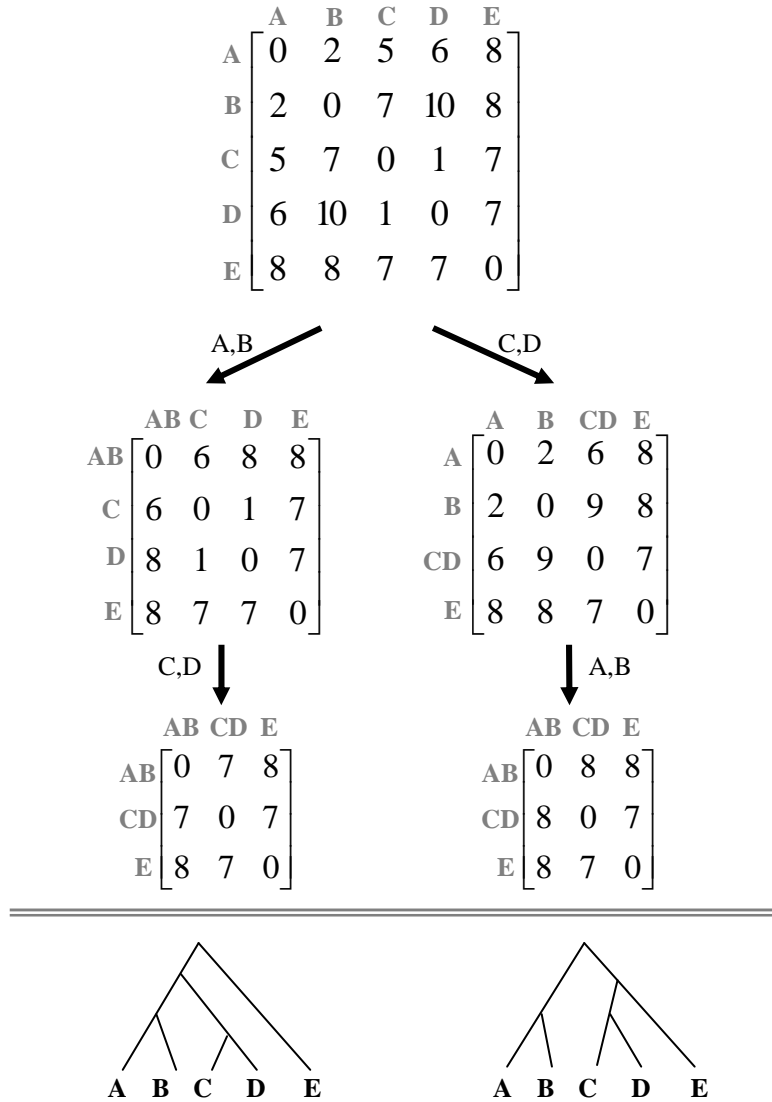


Figure 1: **Convexity does not guarantee equivalence:** when using the convex reduction formula from (6), the unique GCP execution first joins C and D and then joins A and B (right path), while an LCP execution invokes the same joining events in reverse order (left path). The resulting dissimilarity matrices are different and may lead to different clustering.

*Proof.* By induction on  $|\mathcal{C}|$ . The claim holds for  $|\mathcal{C}| \leq 3$ , since in such a case, a locally-closest cluster-pair is globally-closest as well. Assume, therefore, that  $|\mathcal{C}| > 3$ , and let  $LE$  be an execution of the LCP algorithm on  $(\mathcal{C}, D)$  which uses  $F$ . Let  $\{C_i, C_j\} \subseteq \mathcal{C}$  be a globally closest cluster-pair in  $D$  which is joined during  $LE$ , as guaranteed by Lemma 3.1. Since  $F$  is commutative, we can move up the joining of this cluster-pair to the beginning of  $LE$ , without changing the output of the execution. Thus we can assume that the first cluster-pair joined by the execution  $LE$  is  $\{C_i, C_j\}$ . Observe  $LE'$ , the suffix execution of  $LE$  defined on the reduced cluster set  $\mathcal{C}' = \mathcal{C} \setminus \{C_i, C_j\} \cup \{(C_i \cup C_j)\}$  and the corresponding reduced matrix  $D'$ . The induction hypothesis implies that there is an execution  $GE'$  of the corresponding GCP algorithm on  $(\mathcal{C}', D')$  which is equivalent to  $LE'$ . By appending  $GE'$  to the joining of  $\{C_i, C_j\}$  we get an execution  $GE$  of the GCP algorithm which is equivalent to  $LE$ .  $\square$

In conclusion, Theorem 3.3 implies that every GCP clustering algorithm which uses a reduction formula which is both convex and commutative can be implemented by the NN-chain technique presented in Section 2. This includes many common variants, such as UPGMA, WPGMA and the single linkage algorithm.

## Acknowledgement

We would like to thank David Bryant for drawing our attention to [15].

## References

- [1] S. Akella, J. Davis, and P. Waddell. Accelerating phylogenetics computing on the desktop: experiments with executing UPGMA in programmable logic. In *Engineering in Medicine and Biology Society (IEMBS)*, volume 4, pages 2864–2868, 2004.
- [2] J. Barthélemy and A. Guenoche. *Trees and proximities representations*. Wiley, 1991.
- [3] J. Benzécri. Construction d’une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques. *Les Cahiers de l’Analyse des Données*, VII(2):209–219, 1982.
- [4] W. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, 1984.
- [5] C. Ding and X. He. Cluster aggregate inequality and multi-level hierarchical clustering. In *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 224–231, 2005.
- [6] Z. Du and F. Lin. A novel parallelization approach for hierarchical clustering. *Parallel Comput.*, 31(5):523–527, 2005.

- [7] R. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5).
- [8] I. Elias and J. Lagergren. Fast neighbor joining. In *Proc. of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 1263–1274. Springer-Verlag, July 2005.
- [9] D. Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. *J. Exp. Algorithmics*, 5:1–23, 2000.
- [10] V. Filkov and S. Skiena. Heterogeneous data integration with the consensus clustering formalism. In *International Workshop on Data Integration in the Life Sciences (DILS)*, pages 110–123, 2004.
- [11] I. Gronau and S. Moran. Neighbor joining algorithms for inferring phylogenies via LCA-distances. *Journal of Computational Biology*, 14(1):1–15, 2007.
- [12] A. Have T. Christiansen J. Larsen, L. Hansen and T. Kolenda. Webmining: learning from the world wide web. *Computational Statistics and Data Analysis*, 38(4):517–532, 2002.
- [13] J. Juan. Programme de classification hiérarchique par l’algorithme de la recherche en chaîne des voisins réciproques. *Les Cahiers de l’Analyse des Données*, VII(2):219–225, 1982.
- [14] M. Krivánek. The complexity of ultrametric partitions on graphs. *Inform. Process. Lett.*, 27:265–270, 1988.
- [15] F. Murtagh. Complexities of hierarchic clustering algorithms: state of the art. *Computational Statistic Quarterly*, 1(2):101–113, 1984.
- [16] F. Murtagh. Clustering in massive data sets. In P. Pardalos J. Abello and M. Reisende, editors, *Handbook of massive data sets*, pages 501–543. Kluwer Academic Publishers, Norwell, MA, USA, 2002.
- [17] C. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21(8):1313–1325, 1995.
- [18] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4:406–425, 1987.
- [19] R. Sibson. SLINK: an optimally efficient algorithm for the single link cluster method. *The Computer Journal*, 16:30–34, 1973.
- [20] P. Sneath and R. Sokal. *Numerical Taxonomy : the principles and practice of numerical classification*. W. H. Freeman, San Francisco, 1973.