An efficient algorithm for computing the distance between close partitions

Daniel Cosmin Porumbel, Jin-Kao Hao, Pascale Kuntz

An Efficient Algorithm for Computing the Distance Between Close Partitions

Daniel Cosmin Porumbel [a,b] Jin Kao Hao [a] Pascale Kuntz [b]

[a] *LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers, France*

[b] *LINA, Polytech'Nantes, rue Christian Pauc, 44306 Nantes, France*

Abstract

A $K-$ partition of a set S is a splitting of S into K non-overlapping classes that cover all elements of S. Numerous practical applications dealing with data partitioning or clustering require computing the distance between two partitions. Previous articles proved that one can compute it in polynomial time—minimum $O(|S| + K^2)$ and maximum $O(|S| + K^3)$—using a reduction to the linear assignment problem. We propose several conditions for which the partition distance can be computed in $O(|S|)$ time. In practical terms, this computation can be done in $O(|S|)$ time for any two relatively resembling partitions (i.e. with distance less than $\frac{|S|}{5}$) except specially constructed cases. Finally, we prove that, even if there is a bounded number of classes for which the proposed conditions are not satisfied, one can still preserve the linear complexity by exploiting decomposition properties of the similarity matrix.

Key words: partition distance, partition metric, similarity between partitions, clustering comparison, similarity measure

1 Introduction

A $K-$ partition of a set S is a splitting of S into K non-overlapping classes (clusters, parts, blocks or cells), that cover all elements of S. We assume no restriction on the cardinality of a class in this paper; it can be empty and it can also be equal to S. Given two $K-$partitions P_1 and P_2 (even with different numbers of non-empty classes), the partition distance between P_1 and P_2 is defined as the minimum number of elements that need to be moved between classes of P_1 so that the resulting partition equals P_2. The similarity is defined as the maximum number of elements that do not require to be moved in order to obtain two equal partitions.

This definition of partition distance was first stated in 1965 by Régnier [15] and the currently used computation methodology was presented by Day in 1981 [6]. The method transforms the distance problem into the linear assignment problem on a $K \times K$ matrix. Solving the linear assignment problem by classical algorithms is known to take $O(K^2)$ time in the best case and up to $O(K^3)$ time in the worst case (i.e. with the Hungarian algorithm [13]). A complete description of this methodology is available [11] and it is used by all recent studies ([4,5,3,12,8]) dealing with partition distances.

In this paper, we introduce an algorithm that computes the distance in $O(|S|)$ steps if certain conditions are satisfied, e.g. if each class in P_1 shares with a class of P_2 at least half of the elements in each class (see Theorem 1 and Corollary 2). Furthermore, we prove a more general result: one can use strong decomposition properties on the similarity matrix so as to preserve the linear theoretical complexity even if the Theorem 1 conditions do not hold for a bounded number of classes (i.e. at maximum $\sqrt[3]{|S|}$, see Theorem 3). In addition, we show that the approach can be very useful in practice to compute any small distance. For illustration, we give an application example in the context of a graph coloring algorithm that needs to compute billions of small distances (less than $\frac{|S|}{5}$); for the vast majority of small distances, the conditions are indeed satisfied.

The proposed algorithm can be used in numerous applications concerned with close partitions, as for example those comparing a reference partition (a "gold standard") with a partition determined by an algorithm. A classical example is image segmentation, where a segmentation partition can be evaluated according to its distance from a correct/ideal segmentation [3]. In biology, the distance is used to appreciate the difference (error) between a known partition of a population (a family structure) and a reconstruction based on genetic data [12,1]. In clustering, one often obtains different partitions with different clustering algorithms and needs to find a consensus between them. To do this, one determines a central partition, i.e. a partition that minimizes the average distance to all other partitions [5,2]. If there is no much disagreement between the clustering algorithms, the partitions they produce will be sufficiently similar to be handled by the algorithm proposed in this paper.

From a theoretical viewpoint, the partition distance satisfies several interesting properties. For instance, it is known that the partition distance constitutes a metric in the space of partitions [3]. More in-depth studies [4] show that, although the distance ranges from 0 to $|S|$, it can never reach $|S|-1$ and more precise upper bounds are provided (e.g. $|S| - \left\lceil \frac{|S|}{K} \right\rceil$). A comparison between the distance function and other similar measures for partitions (e.g. the Rand index commonly used for comparing two data clusterings) is available [8], showing the distribution of several indexes between close partitions. The distribution of the distances between random partitions is also studied, showing how one can interpret the value of a distance [7]. Finally, generalizations of

this distance measure are available [2,1].

In the next section, we give a set of basic definitions. Section 3 discusses the conditions required for determining the distance in $O(|S|)$ steps. Section 4 extends the application of the proposed algorithm when some of these conditions are only partially satisfied. Section 5 shows an application of the algorithm to compute distances between graph colorings, which is followed by conclusions.

2 Distance Definition

A $K-$partition P of a finite set $S = \{1, 2, \ldots, |S|\}$ is a function $P : S \to \{1, 2, \ldots, K\}$. It can also be defined as a set of classes $\{P^1, P^2, \ldots, P^K\}$ such that $\bigcup_{1 \leq i \leq K} P^i = S$ and $P^i \cap P^j = \emptyset$, for all $i, j \in \{1, 2, \ldots, K\}$,$i \neq j$. The two definitions are equivalent since $P^i = \{x \in S | P(x) = i\}$; $P(x)$ identifies the number of the class of element x so that $x \in P^{P(x)}$ for all $x \in S$. If the P function is not surjective, some classes of the partition need to be empty.

Given two $K-$partitions P_1 and P_2 of S, we denote by $Dist(P_1, P_2)$ the distance between P_1 and P_2, i.e. the minimum number of elements that need to be moved between classes of P_1 so that the resulting partition becomes equal with P_2. The similarity $Sim(P_1, P_2)$ is a complementary measure of the distance denoting the maximum number of elements of P_1 that *do not need* to be moved in order to obtain equal partitions. The two measures satisfy the following equation:

$$Sim(P_1, P_2) + Dist(P_1, P_2) = |S|. \tag{1}$$

Alternatively, the distance can also be interpreted as the minimum number of elements one needs to erase from S such that the two partitions restricted to the set of remaining elements of S (denoted by S') are equal [11]. S' represents a set of elements that are shared by the two partitions and thus $|S'| = Sim(P_1, P_2)$.

To calculate the similarity $|S'|$, one needs to find the one-to-one correspondence $\sigma : \{1, 2, \ldots, K\} \to \{1, 2, \ldots, K\}$ (*assignment*) maximizing the sum:

$$Sim(P_1, P_2) = MAX_\sigma(\sum_{1 \leq i \leq K} T_{i,\sigma(i)}), \tag{2}$$

where T is the $K \times K$ similarity matrix $T(P_1, P_2)$ with elements:

$$T_{ij} = |P_1^i \cap P_2^j| \tag{3}$$

To determine the maximum of the sum in formula (2), one can solve a classical

assignment problem: determine the maximum weighted matching of the complete bipartite graph having vertices $\{P_1^1, P_1^2, \ldots, P_1^K\} \cup \{P_2^1, P_2^2, \ldots, P_2^K\}$ and edges (P_1^i, P_2^j) weighted by T_{ij} for all $i, j \in \{1, 2, \ldots, K\}$. Most papers [6,11,4,12] just suggest to determine the maximal assignment $\bar{\sigma}$ by applying a classical Hungarian algorithm of time complexity between $O(K^2)$ and $O(K^3)$ [13].

Normalized similarity and normalized distance Very often, it is useful to use the normalized values of the similarity and the distance: $s_{P_1,P_2} = \frac{Sim(P_1,P_2)}{|S|}$ and $d_{P_1,P_2} = \frac{Dist(P_1,P_2)}{|S|}$. These values represent a better indicator of the proportion of elements shared by two partitions, s_{P_1,P_2}, or that require to be moved, d_{P_1,P_2}. Clearly, the following formula holds for any S, P_1 and P_2: $d_{P_1,P_2} + s_{P_1,P_2} = 1$.

Note that since the distance is always strictly less than $|S|$, one can also normalize it with respect to other values, for example the maximum distance [4]. However, the simple normalization above suffices for the purpose of this paper.

To exemplify these notions, consider $S = \{1, 2, \ldots, 10\}$ and the 2−partitions P_1 determined by $P_1^1 = \{1, 2, 3, 4, 5\}$ and $P_1^2 = \{6, 7, 8, 9, 10\}$ and P_2 determined by $P_2^1 = \{1, 2, 3, 4, 6, 7\}$ and $P_2^2 = \{5, 8, 9, 10\}$. Computing the matrix T with formula (3) yields $T = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$. The best assignment $\bar{\sigma}$, that maximizes the sum in formula (2), is $\bar{\sigma}(1) = 1$ and $\bar{\sigma}(2) = 2$. We obtain $Sim(P_1, P_2) = T_{1\bar{\sigma}(1)} + T_{2\bar{\sigma}(2)} = 7$ and $s_{P_1,P_2} = \frac{Sim(P_1,P_2)}{|S|} = 0.7$. The distance is thus $Dist(P_1, P_2) = |S| - Sim(P_1, P_2) = 3$ and the normalized value is $d_{P_1,P_2} = 0.3$. Indeed, one needs to change the class of 3 elements of S (elements 5, 6 and 7) to transform P_1 into P_2.

3 Distance Computation

In this section, we describe the new $O(|S|)$ time algorithm and the necessary conditions for calculating the similarity—and implicitly the distance via (1)—of two given partitions P_1 and P_2 of a set S. The algorithm has two major steps: (i) construct the similarity matrix $T(P_1, P_2)$, and (ii) find the best $\bar{\sigma}$ in formula (2). The space complexity of our algorithm is of order $O(|S| + K^2)$.

3.1 Similarity matrix T in $O(|S|)$ time

Our algorithm works on a $K \times K$ similarity matrix T, but only uses $|S|$ elements (at maximum): $T_{ij} = T_{P_1(x),P_2(x)}$, where $x \in S$. This construction step of T can be done in $O(|S|)$ time. More precisely, one first performs the memory allocation for T (without any initialization, using an appropriate memory allocator [1]) and goes through each $x \in S$ by setting $T_{P_1(x),P_2(x)} = 0$ (in $O(|S|)$ time). Then one goes through again each $x \in S$ by incrementing $T_{P_1(x),P_2(x)} := T_{P_1(x),P_2(x)} + 1$ (in $O(|S|)$ time). In fact, the matrix structure is only used for indexing reasons, to quickly address the positions $T_{P_1(x),P_2(x)}$ with $x \in S$.

From now on, the values at positions $T_{P_1(x),P_2(x)}$ with $x \in S$ will be called *relevant*. The rest of the elements of T are considered irrelevant because the algorithm never needs them, neither for reading nor for writing.

3.2 Maximal assignment in $O(|S|)$ steps

This section discusses the conditions in which $O(|S|)$ steps are enough to determine the partition distance. We use the similarity matrix T from the previous section and the objective is to find a maximal assignment $\bar{\sigma}$, i.e. a bijective function $\bar{\sigma}$ maximizing the sum $\sum_{1 \le i \le K} T_{i,\sigma(i)}$ in formula (2).

Theorem 1 *If for all $i \in \{1, 2, \ldots, K\}$, there exists $j \in \{1, 2, \ldots, K\}$ such that $T_{ij} > T_{ij_1}$ and $T_{ij} > T_{i_1 j}$, for all $j_1 \neq j, i_1 \neq i$, then the partition distance can be determined in $O(|S|)$ time.*

Proof We consider that the input consists of set S and partitions P_1 and P_2, where S is $\{1, 2, \ldots |S|\}$ and P_1 and P_2 are two vectors denoting $P_1(x)$ and $P_2(x)$ for all $x \in S$. In the first step, the algorithm computes in $O(|S|)$ time the relevant elements of T (see Section 3.1) and also $T_{i\bar{\sigma}(i)}$, the unique maximum element of each row i. To determine these row-maximums, one goes only through the $O(|S|)$ relevant elements T and performs the following instruction: if T_{ij} is greater than the current maximum on row i (initially, this maximum is zero), the current maximum is updated to T_{ij}.

Since $T_{i\bar{\sigma}(i)}$ is a strict maximum on row i, any other mapping $\sigma : S \to S$ would lead to a no larger sum $\sum_{1 \le i \le K} T_{i,\sigma(i)}$. Checking that $\bar{\sigma}$ is bijective follows from the fact that if $\bar{\sigma}(i) = \bar{\sigma}(i') = j$, then both T_{ij} and $T_{i'j}$ represent the unique

[1] A good example of memory allocator is TLSF (`rtportal.upv.es/rtmalloc/`), that "performs the allocation/deallocation in constant time maintaining a very low memory fragmentation." [14, p. 150].

maximum of column j, and so, i and i' need to be the same. □

The inequalities in the conditions from this theorem need to be strict, because otherwise one can not determine a bijective $\bar{\sigma}$, e.g. if $T = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$. The case solved by this theorem can be seen as a dual of a specific Hungarian algorithm case in which the first step uncovers K mutually independent zeros (i.e. not lying in the same row or column). However, we compute the partition distance without converting the problem to a minimization problem (the Hungarian algorithm solves minimization problems) and without performing the $O(K \times K)$ row/column reductions needed by the Hungarian algorithm.

A similar condition could be expressed without mentioning the matrix T.

Corollary 2 *If for all $i \in \{1, 2, \ldots, K\}$, there exists $j \in \{1, 2, \ldots, K\}$ such that $|P_1^i \cap P_2^j| > \frac{|P_1^i|}{2}$ and $|P_1^i \cap P_2^j| > \frac{|P_2^j|}{2}$, then the partition distance can be determined in $O(|S|)$ time.*

Proof The given hypothesis conditions represent a particular case of the conditions in Theorem 1. From (3), we have $\sum_{1 \le k \le K} T_{ik} = \sum_{1 \le k \le K} |P_1^i \cap P_2^k|$. Since all P_2^k are disjoint and their union is S, $\sum_{1 \le k \le K} |P_1^i \cap P_2^k| = |P_1^i \cap S| = |P_1^i|$. Therefore, for all $i \in \{1, 2, \ldots K\}$, we have

$$\sum_{1 \le k \le K} T_{ik} = |P_1^i|. \tag{4}$$

By similar reasoning, we can conclude that:

$$\sum_{1 \le k \le K} T_{kj} = |P_2^j|. \tag{5}$$

Using the hypothesis conditions, it follows that $T_{ij} > T_{ij_1}$ and $T_{ij} > T_{i_1 j}$, for all $j_1 \neq j, i_1 \neq i$ and the proof can be finished by using Theorem 1. □

The main practical drawback of the conditions of this Corollary and of Theorem 1 is that, if there is a single row i on which they are *not* satisfied, the rest of the construction can not be used for determining the best assignment. The next theorem overcomes this issue and moreover, it can not be related to a dual of a step of the Hungarian algorithm. We show how one can determine the best assignment $i \xrightarrow{\bar{\sigma}} j$ on a row i by looking only at the elements on row i and column j—recall that the Hungarian algorithm returns only complete solutions and it takes no such intermediate (early) decisions on particular rows or columns.

Theorem 3 *If for a given row $i \in \{1, 2, \ldots, K\}$, there exists column $j \in \{1, 2, \ldots, K\}$ such that $T_{ij} \ge T_{ij_1} + T_{i_1 j}$ for all $j_1 \neq j, i_1 \neq i$, one can construct*

a maximal assignment $\bar{\sigma}$ such that $\bar{\sigma}(i) = j$. If the number of rows i not *satisfying this condition is bounded (i.e. less than $\sqrt[3]{|S|}$), the partition distance can be determined in $O(|S|)$ time.*

Proof Following a very similar algorithm to the one in Theorem 1, one can determine matrix T and also the maximum value on each row and on each column. By going through the $O(|S|)$ relevant elements once again, one marks all maximum elements that are discovered on each row. Since only a marked row-maximum T_{ij} can satisfy $T_{ij} \geq T_{ij_1} + T_{i_1 j}$ for all $j_1 \neq j, i_1 \neq i$, the algorithm just needs to check for each row-maximum T_{ij} that it is greater than $T_{i,-} + T_{-,j}$, where $T_{i,-}$ and $T_{-,j}$ are the second maximum values in row i and column j, respectively. We consider that $T_{i,-} = T_{ij}$ if and only if row i has at least two maximum value elements. Furthermore, determining the second maximum value of a row (or column, respectively) is very similar to determining the first maximum. Thus, the hypothesis condition can be checked for all rows in $O(|S|)$ time.

Let T_{ij} be an element marked by the above $O(|S|)$ procedure, such that $T_{ij} \geq T_{ij_1} + T_{i_1 j}$ for all $j_1 \neq j, i_1 \neq i$. We need to show that one can construct an maximum assignment by mapping i to j. Let σ be a maximal assignment. If $\sigma(i) = j$, then σ constitutes the searched assignment. Otherwise, let $j_1 = \sigma(i)$ and $i_1 = \sigma^{-1}(j)$. Using the hypothesis condition, one obtains:

$$T_{ij} + T_{i_1 j_1} \geq T_{i_1 j} + T_{ij_1} = T_{i_1 \sigma(i_1)} + T_{i\sigma(i)} \tag{6}$$

By composing the transposition permutation (i, i_1) with σ, one obtains a new bijective mapping $\bar{\sigma}$ that differs from σ only on positions i and i_1, such that the values on these positions are switched, i.e. $\bar{\sigma}(i) = j$ and $\bar{\sigma}(i_1) = j_1$. The difference of value between assignments $\bar{\sigma}$ and σ (see (2)) is $T_{ij} + T_{i_1 j_1} - (T_{i_1 j} + T_{ij_1}) \geq 0$. Using (6), $\bar{\sigma}$ also needs to be a maximal assignment.

To summarize, an algorithm could establish a partial best assignment on all rows i that satisfy the hypothesis condition, regardless of the rows that do not satisfy this condition. This assignment on these rows is determined by mapping row i to any column j satisfying $T_{ij} \geq T_{ij_1} + T_{i_1 j}$ for all $j_1 \neq j, i_1 \neq i$. If there are two rows i_1 and i_2 pointing to the same j, then $T_{i_1 j}$ and $T_{i_2 j}$ need to be the only non-zero elements on row i_1 and i_2. As such, one can map i_1 to j and i_2 can be mapped to any other value on row i_2.

The rest of the assignment can be constructed by applying the Hungarian algorithm on the remaining elements. Under the given hypothesis condition, the number of unassigned $\bar{\sigma}$ elements is $K' = \left\lfloor \sqrt[3]{|S|} \right\rfloor$ in the worst case. To complete the assignment, one first marks the K' unassigned rows and the K' unassigned columns. A new $K' \times K'$ matrix is also allocated and initialized to zero in less than $O(|S|)$. Then, one goes through the relevant elements

of T and copies into a new $K' \times K'$ matrix all elements situated at the intersection of a marked row and column. Finally, the Hungarian algorithm determines the maximum assignment value on this restricted matrix using maximum $O(K'^3) = O(|S|)$ operations, resulting in a total time complexity of $O(|S|)$ for the whole algorithm. □

A similar condition could be expressed without mentioning the matrix T, in a simpler manner.

Corollary 4 *If for all $i \in \{1, 2, \ldots, K\}$, there exists $j \in \{1, 2, \ldots, K\}$ such that $|P_1^i \cap P_2^j| \geq \frac{|P_1^i \cup P_2^j|}{2}$, then the partition distance can be computed in $O(|S|)$ steps.*

Proof This proposition follows from equations (4) and (5) as it becomes a particular case of Theorem 3. However, this corollary also has the advantage that it is very easy to implement because $|P_1^i \cup P_2^j| = |P_1^i| + |P_2^j| - |P_1^i \cap P_2^j|$ and $|P_1^i|$ and $|P_2^j|$ can be easily determined. □

Regarding the most general proofs of this section, note that Theorem 1 and Theorem 3 do not result one from another. For example, if $T = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$, only Theorem 1 can be used; if $T = \begin{bmatrix} 2 & 2 \\ 0 & 2 \end{bmatrix}$, one should use Theorem 3. In the next section, we show how Theorem 3 and Corollary 4 can also be used in practice to construct only a part of the solution.

4 Extensions

In case the hypothesis conditions of Theorem 3 or Corollary 4 only hold for a restricted set of rows $i \in \{1, 2, \ldots, K\}$, we can still perform important time complexity reductions. First, let us prove the following proposition:

$$|P_1^i \cap P_2^j| \leq \frac{|P_1^i \cup P_2^j|}{2} \text{ for all } i, j \in \{1, 2, \ldots, K\} \Rightarrow Sim(P_1, P_2) \leq \frac{2}{3}|S|. \quad (7)$$

Let $\bar{\sigma}'$ be a maximal assignment and $j = \bar{\sigma}'(i)$, where i is any row. One can write $T_{ij} \leq \frac{|P_1^i \cup P_2^j|}{2}$ as $T_{ij} \leq \frac{|P_1^i| + |P_2^j| - |P_1^i \cap P_2^j|}{2}$, or $3T_{ij} \leq |P_1^i| + |P_2^j|$. Making the sum over all rows i, one obtains $3 \sum_{1 \leq i \leq K} T_{i\bar{\sigma}'(i)} \leq \sum_{1 \leq k \leq K} |P_1^k| + \sum_{1 \leq k \leq K} |P_2^k| = 2|S|$, which proves (7).

We use the conditions of Corollary 4 only for a greater readability, but the

same result could be derived for the conditions of Theorem 3, i.e. if $T_{ij} < T_{ij_1} + T_{i_1 j}$ for all $j_1 \neq j, i_1 \neq i$, then $|P_1^i \cap P_2^j| \leq \frac{|P_1^i \cup P_2^j|}{2}$ is also satisfied—see (4) and (5).

Now, we present the actual reduction of the $Sim(P_1, P_2)$ computation into smaller pieces. We divide S into two subsets A and B such that only the computation on B requires an algorithm of higher complexity. Let us denote by I the set of elements i for which there is $j_i \in \{1, 2, \ldots, K\}$ such that $|P_1^i \cap P_2^{j_i}| > \frac{|P_1^i \cup P_2^{j_i}|}{2}$. We write $J = \{j \in S |$ there exists $i \in I$ s.t.$j = j_i\}$ and let

$$A = \bigcup_{i \in I} P_1^i \cup P_2^{j_i} \tag{8}$$

and $B = S - A$. Using A in Corollary 4 (or Theorem 3), one finds there exists a maximal assignment $\bar{\sigma}$ satisfying $\bar{\sigma}(i) = j_i,$ for all $i \in I$. Since J is the image of I through the bijective $\bar{\sigma}$, then $\{1, 2, \ldots, K\} - J$ is the image of $\{1, 2, \ldots, K\} - I$. The rest of $\bar{\sigma}$ can be constructed only using rows and columns from these two sets, which contain values generated only by classes of B (i.e. subsets $P_1^i, P_2^j \subseteq B$).

Writing $Sim(P_1, P_2)|_X$ the similarity between partitions P_1 and P_2 restricted to set $X \subset S$, we obtain $Sim(P_1, P_2) = Sim(P_1, P_2)|_A + Sim(P_1, P_2)|_B$. Since no confusion arises, we can simply write: $|S| \cdot s_{P_1,P_2} = |A| \cdot s_{P_1,P_2}|_A + |B| \cdot s_{P_1,P_2}|_B$ and we can even omit the index $_{P_1,P_2}$:

$$s_S = \frac{|A|}{|S|} s_A + \frac{|B|}{|S|} s_B,$$

where s_X is the normalized similarity between P_1 and P_2 restricted to set X.

The sets A and B can be directly determined from set I using (8) and I can be determined in $O(|S|)$ time, following the reasoning of Theorem 3. Furthermore, s_A can be determined in $O(|A|) < O(|S|)$ as explained in Section 3.2; s_B can be determined in maximum $O((K - |I|)^3)$ using the Hungarian algorithm. To summarize, the total time complexity of computing the similarity this way is $O(|S|) + O((K - |I|)^3)$ at maximum.

Using (7), we obtain $s_B \leq \frac{2}{3}$. This means that if the total similarity is high (i.e. for example $s_S > 0.9$), S can be split in two parts:

(1) A, on which the normalized similarity s_A is very high (e.g. $s_A > s_S > 0.9$) and can be computed in $O(|S|)$.
(2) B, on which the normalized similarity is *much* lower $s_B \leq \frac{2}{3}$ and can not be computed in $O(|S|)$.

In case the total similarity of P_1 and P_2 is high, even if we cannot always compute it in $O(|S|)$ time, we can always identify the part of S where the

matching is stronger (i.e. A) in $O(|S|)$ time. This could be particularly useful for applications that only need to find the best class matches between two partitions.

5 An Application Example

In this section we present numerical experiments in the context of a graph coloring heuristic algorithm that needs to compute billions of distances to build up a search space clusterization. The graph $K-$coloring problem is a partition problem: divide the vertex set S of the graph into K disjoint classes such that there is no edge with both end vertices in the same class (i.e. of the same color). To check the distance of two given $K-$colorings, the partition distance is typically used [9] and the Hungarian algorithm is used for the computation([10]).

We have computed one billion small distances by applying the heuristic coloring algorithm on two standard coloring graphs [2] with 1000 vertices (so that $|S| = 1000$) and 20 and respectively 86 colors (so that K is 20 and respectively 86). The considered distances are small as they are computed between close positions in a series of neighboring colorings; two neighboring coloring differ only by the color of a single vertex. However, even if there are also greater distances, we only count in this statistics the pairs (P_1, P_2) satisfying $Dist(P_1, P_2) < \frac{|S|}{5}$. The practical objective was to analyze the structure of the search space of a heuristic algorithm in order to control the exploration diversity. For each distance calculation, we apply our partition distance algorithm following this methodology:

(1) If the condition in Corollary 4 is satisfied, the algorithm simply computes the correct distance in $O(|S|)$ time.
(2) Otherwise, the algorithm detects that the condition is not satisfied (it returns $IMPOSSIBLE$ in $O(|S|)$ time, see algorithm in appendix A.) and the $O(K^3)$ Hungarian algorithm is executed.

The first step was sufficient for more than 99.99% cases. More precisely, in more than 10^9 computed distances, we found less than 10^3 (i.e. 737 when $K = 20$ and 880 for the $K = 86$ case) pairs (P_1, P_2) for which $Dist(P_1, P_2) < \frac{|S|}{5}$ but the hypothesis condition in Corollary 4 is not satisfied. If we consider even smaller distances (more exactly, only pairs (P_1, P_2) such that $Dist(P_1, P_2) < \frac{|S|}{10}$) the $O(|S|)$ time algorithm is sufficient for all practical cases we encountered.

[2] DIMACS Graphs *dsjc*1000.1 and *dsjc*1000.5, available (for example) from http://mat.gsia.cmu.edu/COLOR/instances.html

The explanation of this practical success lies in the fact that the similarity restricted to different classes (subsets) of S presents quite homogeneous values in practice. Thus, if the total similarity is high (i.e. $s_{P_1,P_2} > 0.9$ equivalent to $Dist(P_1, P_2) < \frac{|S|}{10}$ in our example), the s_X values are quite close to 0.9 for most classes X of the partitions. Thus, the cardinal of set B (on which $s_B < \frac{2}{3}$, and the conditions from section 3.2 are not met) is very limited, usually B is empty in practice.

However, theoretically one can still construct a counter-example to this by taking two partitions such that $P_1^1 = P_2^1$ and $|P_1^1| = 0.9|S|$. In this case $s_{P_1,P_2} \geq 0.9$, but the two partitions, which are very similar on $A = P_1^1$, can be totally different on $B = S - A$. The most difficult part is the computation of the similarity restricted to B; only this one may still require between $O(K^2)$ and $O(K^3)$ time.

6 Conclusions

This paper introduces a very fast algorithm for computing the distance between two close partitions P_1 and P_2 of a set S under the conditions specified in Section 3. From a practical perspective, the partition distance between two given partitions can be calculated in $O(|S|)$ time, if they have a distance less than $\frac{|S|}{5}$. This should be contrasted with the conventional methods based on the Hungarian algorithm that requires $O(|S| + K^2)$ time for the same cases considered in this paper.

Moreover, the proposed algorithm can also be useful even if the required conditions are not totally satisfied as it is explained in Section 4. In such a situation, the algorithm can be used to identify the subset of S on which the matching is stronger, i.e. where the normalized similarity is at least $\frac{2}{3}$. Finally, a part of the proposed algorithm can be useful for solving general assignment problems dealing with sparse matrices.

Acknowledgments: This work is partially supported by the CPER project "Pôle Informatique Régional" (2000-2006) and the Régional Project MILES (2007-2009). We are grateful for the comments of two anonymous referees whose ideas have helped us to improve the paper.

References

[1] T.Y. Berger-Wolf, S.I. Sheikh, B. DasGupta, M.V. Ashley, I.C. Caballero, W. Chaovalitwongse, and S.L. Putrevu. Reconstructing sibling relationships

in wild populations. *Bioinformatics*, 23(13):i49, 2007.

[2] P. Berman, B. DasGupta, M.Y. Kao, and J. Wang. On constructing an optimal consensus clustering from multiple clusterings. *Information Processing Letters*, 104(4):137–145, 2007.

[3] J.S. Cardoso and L. Corte-Real. Toward a Generic Evaluation of Image Segmentation. *IEEE Transactions on Image Processing*, 14(11):1773–1782, 2005.

[4] I. Charon, L. Denoeud, A. Guénoche, and O. Hudry. Maximum Transfer Distance Between Partitions. *Journal of Classification*, 23(1):103–121, 2006.

[5] J.F.P. da Costa and P.R. Rao. Central partition for a partition-distance and strong pattern graphs. *REVSTAT–Statistical Journal*, 2(2):127–143, 2004.

[6] W.H.E. Day. The complexity of computing metric distances between partitions. *Mathematical Social Sciences*, 1:269–287, 1981.

[7] L. Denoeud. Transfer distance between partitions. *Advances in Data Analysis and Classification*, 2(3):279–294, 2008.

[8] L. Denoeud and A. Guénoche. Comparison of distance indices between partitions. In *Data Science and Classification*, pages 21–28. Springer, 2006.

[9] P. Galinier and J.K. Hao. Hybrid Evolutionary Algorithms for Graph Coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

[10] C.A. Glass and A. Pruegel-Bennett. A polynomially searchable exponential neighbourhood for graph colouring. *Journal of the Operational Research Society*, 56(3):324–330, 2005.

[11] D. Gusfield. Partition-distance: A problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82(3):159–164, 2002.

[12] D.A. Konovalov, B. Litow, and N. Bajema. Partition-distance via the assignment problem. *Bioinformatics*, 21(10):2463–2468, 2005.

[13] H.W. Kühn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[14] M. Masmano, I. Ripoll, P. Balbastre, and A. Crespo. A constant-time dynamic storage allocator for real-time systems. *Real-Time Systems*, 40(2):149–179, 2008.

[15] S. Régnier. Sur quelques aspects mathématiques des problèmes de classification automatique. *Mathématiques et Sciences Humaines*, 82:20, 1983. reprint of *ICC Bulletin*, 4, 175- 191, Rome, 1965.

Algorithm 1 Algorithm corresponding to Corollary 4

Inputs:

- $|S|$, so that $S = \{1, 2, \ldots, |S|\}$
- P_1 and P_2 as $|S|$-vectors (i.e. position x in P_1 represent $P_1(x)$, for all $x \in S$)

Return value:

- the distance Σ, if the condition in Corollary 4 is satisfied
- *IMPOSSIBLE*, otherwise.

Begin

(1) init $\Sigma = 0$
(2) init K = maximum value in vectors P_1 and P_2
(3) allocate the $K \times K$ matrix T (without filling any element)
(4) init $K-$vectors M and $\bar{\sigma}$ to 0 (the maximum on each row and the maximal assignment that is constructed)
(5) init $K-$vectors $|P_1|$ and $|P_2|$ to 0 (denoting the cardinal of each class)
(6) FOR $x = 1$ TO $|S|$
    - set $i = P_1(x)$ and $j = P_2(x)$
    - set $T_{ij} = 0$
(7) FOR $x = 1$ TO $|S|$
    - set $i = P_1(x)$ and $j = P_2(x)$
    - increment T_{ij}, $|P_1^i|$, $|P_2^j|$
    - IF $T_{ij} > M_i$, THEN set $M_i = T_{ij}$ and $\bar{\sigma}(i) = j$.
(8) FOR $i = 1$ TO K
    - IF $M_i = 0$, THEN CONTINUE (with next i)
    - IF $3M_i \leq |P_1^i| + |P_2^{\bar{\sigma}(i)}|$, THEN RETURN *IMPOSSIBLE*
    - set $\Sigma = \Sigma + T_{i,\bar{\sigma}(i)}$
(9) RETURN Σ

End

A A complete algorithm example

The algorithm we present in this appendix corresponds to Corollary 4 and as one can see, it is quite simple. By detailing it, one can also implement a similar version corresponding to any other theorem in the paper. The number of classes does not need to be the same for P_1 and P_2 because K is not given in the input; it is computed as the maximum number of classes in any partition.