Measuring the distance of generalized maps

Camille Combier, Christine Solnon, Guillaume Damiand

Measuring the distance of generalized maps *

Camille Combier[1,2], Guillaume Damiand[1,2], and Christine Solnon[1,2]

[1] Université de Lyon, CNRS
[2] Université Lyon 1, LIRIS, UMR5205, F-69622, France

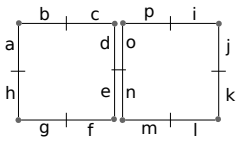
Abstract. Generalized maps are widely used to model the topology of nD objects (such as images) by means of incidence and adjacency relationships between cells (vertices, edges, faces, volumes, ...). In this paper, we define a first error-tolerant distance measure for comparing generalized maps, which is an important issue for image processing and analysis. This distance measure is defined by means of the size of a largest common submap, in a similar way as a graph distance measure may be defined by means of the size of a largest common subgraph. We show that this distance measure is a metric, and we introduce a greedy randomized algorithm which allows us to efficiently compute an upper bound of it.

1 Introduction

Generalized maps are widely used to model the topology of nD objects subdivided in cells (*e.g.*, vertices, edges, faces, volumes, ...) by means of incidence and adjacency relationships between these cells. In 2D, they are an extension of planar graphs, and a generalization for higher dimensions. In particular, generalized maps are very well suited to model 2D and 3D images, and there exist efficient algorithms for extracting maps from images [DBF04,Dam08]. In [DDLHJ+09], we have defined two basic comparison tools, *i.e.*, map isomorphism (which involves deciding if two generalized maps are equivalent) and submap isomorphism (which involves deciding if a copy of a pattern generalized map may be found in a target generalized map), and we have proposed efficient polynomial time algorithms for solving these two problems. However, these decision algorithms cannot be used to quantify the distance between two generalized maps as soon as there is no inclusion relation between them.

In this paper, we define a first error-tolerant distance measure for comparing generalized maps, which is an important issue for image processing and analysis. This distance measure is defined by means of the size of a largest common submap, in a similar way as a graph distance measure is defined by means of the size of a largest common subgraph in [BS98]. In Section 2, we briefly recall definitions related to generalized maps. In Section 3, we define the distance measure and we show that it is a metric distance. In Section 4, we describe a greedy

* This article has been published in Graph-Based Representations in Pattern Recognition 8th IAPR-TC-15 International Workshop, GbRPR 2011, Munster, Germany, May 18-20, 2011. Proceedings, DOI: 10.1007/978-3-642-20844-7



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
α_0	h	c	b	e	d	g	f	a	p	k	j	m	l	o	n	i
α_1	b	a	d	c	f	e	h	g	j	i	l	k	n	m	p	o
α_2	a	b	c	o	n	f	g	h	i	j	k	l	m	e	d	p

Fig. 1. Example of 2G-map: consecutive darts separated with a little segment are 0-sewn (*e.g.*, b and c); consecutive darts separated with a dot are 1-sewn (*e.g.*, a and b); parallel darts are 2-sewn (*e.g.*, d and o); a is 2-free because $\alpha_2(a) = a$.

randomized algorithm which allows us to efficiently compute an approximation of this distance measure. In Section 5, we give first experimental results showing that our algorithm is able to compute good approximations of the distance.

2 Generalized maps and (sub)map isomorphism

Let us first recall basic definitions on generalized maps, which are a generalization of combinatorial maps. We refer the reader to [Lie94] for more details.

Definition 1 *(nG-map) Let $n \geq 0$. An n-dimensional generalized map (or nG-map) is defined by a tuple $M = (D, \alpha_0, \ldots, \alpha_n)$ such that (i) D is a finite set of darts; (ii) $\forall i \in [0, n]$, α_i is an involution[3] on D; and (iii) $\forall i, j \in [0, n]$ such that $i + 2 \leq j$, $\alpha_i \circ \alpha_j$ is an involution.*

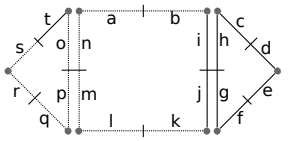
We will say that a dart $d \in D$ is i-free whenever $d = \alpha_i(d)$, and that it is i-sewn to $d' \in D$ whenever $d = \alpha_i(d')$ and $d \neq d'$. Fig. 1 displays an example of 2G-map which describes an object composed of two adjacent faces.

Map isomorphism checks for the equivalence of nG-maps. It has been defined in [Lie94]. Submap isomorphism checks for the inclusion of two nG-maps. It has been defined in [DDLHJ+09] for combinatorial maps. Definition 2 extends this definition to generalized maps (see Fig. 2 for an example).

Definition 2 *(nG-map subisomorphism) Let $M = (D, \alpha_0, \ldots, \alpha_n)$ and $M' = (D', \alpha'_0, \ldots, \alpha'_n)$ be two nG-maps. M is subisomorphic to M', denoted $M \sqsubseteq M'$ if there exist an injective function $f : D \to D'$ such that $\forall d \in D$, and $\forall i \in [0, n]$:*

- *if d is i-sewn, then $f(\alpha_i(d)) = \alpha'_i(f(d))$;*
- *if d is i-free, then either $f(d)$ is i-free, or $f(d)$ is i-sewn with a dart which is not matched by f to another dart of D,* i.e., *$\forall d_k \in D, f(d_k) \neq \alpha'_i(f(d))$.*

[3] An involution f on D is a bijective mapping from D to D such that $f = f^{-1}$.



(a) M (b) M'

$1 \to b$ $2 \to a$ $3 \to n$
$4 \to m$ $5 \to l$ $6 \to k$
$7 \to p$ $8 \to o$ $9 \to s$
$10 \to r$ $11 \to q$

(c) Subisomorphism function f

Fig. 2. Submap isomorphism example: f is a subisomorphism function from M' to M.

3 Definition of a distance measure for generalized maps

Submap isomorphism may be used to decide of generalized map inclusion, but it cannot be used to quantify the distance between two nG-maps which are not related by an inclusion relationships. In this section, we propose to quantify this distance by means of the largest generalized map included into the two generalized maps to be compared. This definition may be viewed as an extension of the graph distance measure based on maximum common subgraphs defined in [BS98].

Let us first define generalized map sizes and maximum common submaps.

Definition 3 *(size of an nG-map) The size of an nG-map $M = (D, \alpha_0, \ldots, \alpha_n)$ is denoted $|M|$ and is defined by its number of darts,* i.e.*, $|M| = |D|$.*

Definition 4 *(maximum common submap) Let M and M' be two nG-maps. A maximum common submap of M and M', denoted $mcs(M, M')$, is an nG-map such that $mcs(M, M') \sqsubseteq M$, $mcs(M, M') \sqsubseteq M'$; and $|mcs(M, M')|$ is maximal.*

Note that $mcs(M, M')$ is not necessarily unique as there may exist several common submaps which have a same size. A maximum common submap basically corresponds to the intersection of two nG-maps, and its size may be used to quantify their similarity: the larger a maximum common submap, the more similar the two nG-maps. To define a distance, we normalize this size with respect to the size of the largest of the two generalized maps, as proposed in [BS98] for graphs.

Definition 5 *(distance between two nG-maps) Let M_1 and M_2 be two nG-maps. The distance between M_1 and M_2 is defined by:*

$$d(M_1, M_2) = 0 \quad \text{if } |M_1| = |M_2| = 0;$$
$$d(M_1, M_2) = 1 - \frac{|mcs(M_1, M_2)|}{max(|M_1|, |M_2|)} \quad \text{otherwise.}$$

Theorem 1 *Let $n \geq 1$. The distance d is a metric on the set $\mathcal{M}$ of all nG-maps so that the following properties hold:*

1. *Non-negativity: $\forall M_1, M_2 \in \mathcal{M}, d(M_1, M_2) \geq 0$;*

2. *Isomorphism of indiscernibles:*
   $\forall M_1, M_2 \in \mathcal{M}, d(M_1, M_2) = 0$ *iff* M_1 *and* M_2 *are isomorphic;*
3. *Symmetry:* $\forall M_1, M_2 \in \mathcal{M}, d(M_1, M_2) = d(M_2, M_1)$*;*
4. *Triangle inequality:* $\forall M_1, M_2, M_3 \in \mathcal{M}, d(M_1, M_3) \leq d(M_1, M_2)+d(M_2, M_3)$.

Proof. Properties 1, 2, and 3 are direct consequences of Def. 5.
Let us denote $m_{ij} = mcs(M_i, M_j)$, and $S_{ij} = max(|M_i|, |M_j|)$, and let us show Property 4 by considering separately the two following cases.

(Case 1): $d(M_1, M_2) + d(M_2, M_3) \geq 1$.
In this case, the triangle inequality trivially holds as $d(M_1, M_3) \leq 1$.

(Case 2): $d(M_1, M_2) + d(M_2, M_3) < 1$.
In this case, let us first show that there exists at least one dart d of M_2 which belongs both to m_{12} and m_{23} or, in other words, that the sum of the number of darts of m_{12} and m_{23} is strictly greater than the number of darts of M_2 so that at least one dart of M_2 belongs to the two common submaps, *i.e.*,

$$|M_2| < |m_{12}| + |m_{23}| \tag{1}$$

This inequation can be proven by considering all possible order relations between nG-map sizes. For example, if $|M_1| \geq |M_3| \geq |M_2|$, then:
(Case 2) $\Leftrightarrow 1-\frac{|m_{12}|}{|M_1|}+1-\frac{|m_{23}|}{|M_3|} < 1$ (by Def. 5, and as $S_{12} = |M_1|$ and $S_{23} = |M_3|$)
$\Leftrightarrow |M_3| < \frac{|M_3|}{|M_1|}|m_{12}| + |m_{23}|$ (by multiplying by $|M_3|$)
$\Rightarrow |M_3| < |m_{12}| + |m_{23}|$ (as $\frac{|M_3|}{|M_1|} < 1$)
$\Rightarrow |M_2| < |m_{12}| + |m_{23}|$ (as $|M_3| \geq |M_2|$).
Ineq. (1) can be proven in a very similar way for the five other possible order relations between nG-map sizes.

Ineq. (1) shows that the sum of the sizes of the two common submaps m_{12} and m_{23} is always strictly greater than the size of M_2 so that there are at least $|m_{12}| + |m_{23}| - |M_2|$ darts that both belong to m_{12} and m_{23}. Therefore, the nG-map $mcs(m_{12}, m_{23})$ is a common submap of M_1, M_2, and M_3 which has at least $|m_{12}| + |m_{23}| - |M_2|$ darts. This nG-map gives a lower bound on the size of the maximum common submap of M_1 and M_3, *i.e.*,

$$|m_{13}| \geq |m_{12}| + |m_{23}| - |M_2| \tag{2}$$

Let us use this lower bound to show that the triangle inequality holds. When developing the triangle inequality w.r.t. Def. 5, it becomes:

$$|m_{13}| \geq \frac{S_{13}}{S_{12}}|m_{12}| + \frac{S_{13}}{S_{23}}|m_{23}| - S_{13} \tag{3}$$

Let us prove (3) by considering all order relations between nG-map sizes:

(Case 2.1): $|M_1| \geq |M_2| \geq |M_3|$ so that $S_{13} = |M_1|, S_{12} = |M_1|, S_{23} = |M_2|$.
Ineq. (3) becomes $|m_{13}| \geq |m_{12}|+\frac{|M_1|}{|M_2|}|m_{23}|-|M_1|$. As $|m_{13}| \geq |m_{12}|+|m_{23}|-|M_2|$ (Ineq. (2)), we have to show that $|m_{23}| - |M_2| \geq \frac{|M_1|}{|M_2|}|m_{23}| - |M_1|$, *i.e.*, $|m_{23}| \leq |M_2|$ (as $|M_2| - |M_1| < 0$). This inequality trivially holds by Def. 4.

(Case 2.2): $|M_2| \geq |M_1| \geq |M_3|$ so that $S_{13} = |M_1|, S_{12} = |M_2|, S_{23} = |M_2|$. Ineq. (3) becomes $|m_{13}| \geq \frac{|M_1|}{|M_2|}|m_{12}| + \frac{|M_1|}{|M_2|}|m_{23}| - |M_1|$. As $\frac{|M_1|}{|M_2|} \leq 1$, Ineq. (2) implies that $|m_{13}| \geq \frac{|M_1|}{|M_2|}(|m_{12}| + |m_{23}| - |M_2|$. Therefore, Ineq. (3) holds.

(Case 2.3): $|M_1| \geq |M_3| \geq |M_2|$ so that $S_{13} = |M_1|, S_{12} = |M_1|, S_{23} = |M_3|$. Ineq. (3) becomes $|m_{13}| \geq |m_{12}| + \frac{|M_1|}{|M_3|}|m_{23}| - |M_1|$. As $|m_{13}| \geq |m_{12}| + |m_{23}| - |M_2|$ (Ineq. (2)), we have to show that $|m_{23}| - |M_2| \geq \frac{|M_1|}{|M_3|}|m_{23}| - |M_1|$, *i.e.*, $|m_{23}| \leq |M_3|\frac{|M_2|-|M_1|}{|M_3|-|M_1|}$ (as $|M_3| - |M_1| < 0$). This inequality trivially holds by Def. 4 because $\frac{|M_2|-|M_1|}{|M_3|-|M_1|} \geq 1$.

The 3 others cases can be obtained by inverting M_1 and M_3. □

4 Algorithm for approximating the nG-map distance

Computing the distance between two nG-maps basically involves computing their maximum common submap. We have given in [DDLHJ+09] a polynomial time algorithm for deciding if there exists a submap isomorphism between two connected generalized maps (such that there exists a path of i-sewn darts between every pair of darts). This problem becomes $\mathcal{NP}$-complete as soon as the pattern nG-map is not connected. As the maximum common submap of two nG-maps is not necessarily connected, computing a maximum common submap is at least as hard as deciding of submap isomorphism between two non connected nG-maps and, therefore, the maximum common submap problem is $\mathcal{NP}$-hard.

In this section, we describe a polynomial-time algorithm which efficiently compute an upper bound of $d(M, M')$. The idea is to build a submap which is common to M and M' and which is (hopefully) as large as possible. To compute a common submap, we actually build a dart matching which induces a submap, in a rather similar way as we compute a common subgraph by building a vertex matching which induces a subgraph. More precisely, a dart matching is a function $m : D \to D' \cup \{\epsilon\}$ such that, for each dart $d \in D$, if $m(d) = \epsilon$ then d is not matched, otherwise d is matched to $m(d) \in D'$. Symmetrically, a dart $d' \in D'$ is said to be not matched if $\forall d_k \in D, m(d_k) \neq d'$, otherwise d' is said to be matched with the dart $d \in D$ such that $m(d) = d'$. A matching induces a submap which is obtained by removing from M all non matched darts. Obviously, a matching must preserves involutions so that two darts i-sewn must be matched to two darts which are also i-sewn. A matching must also induce a consistent submap which satisfies constraint (iii) of Def. 1, as illustrated in Fig. 3. More precisely, Def. 6 defines consistent matchings.

Definition 6 *(consistent dart matching) Let $M = (D, \alpha_0, \ldots, \alpha_n)$ and $M' = (D', \alpha'_0, \ldots, \alpha'_n)$ be two nG-maps. A matching $m : D \to D' \cup \{\epsilon\}$ is consistent if*

1. *it matches different darts of M to different darts of M',* i.e.*,*
   $\forall d_1, d_2 \in D, d_1 \neq d_2 \Rightarrow m(d_1) = \epsilon \vee m(d_2) = \epsilon \vee m(d_1) \neq m(d_2)$*;*
2. *it preserves all involutions between matched darts* i.e.*,*
   $\forall d \in D$ *such that* $m(d) \neq \epsilon$*,* $\forall i \in [0, n]$*:*

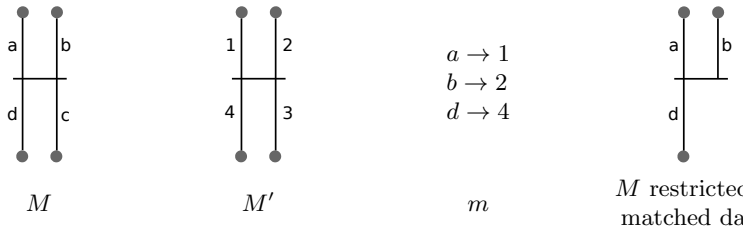


Fig. 3. Example of matching $m : M \to M'$ which does not induce a consistent submap. Constraint $\alpha_i \circ \alpha_j(d) = \alpha_j \circ \alpha_i(d)$ of Def. 1 is not satisfied in M restricted to $\{a, b, d\}$ as $\alpha_2(\alpha_0(a)) = d$ and $\alpha_0(\alpha_2(a)) = b$.

- *if d is i-free and $m(d)$ is i-sewn then $\alpha'_i(m(d))$ is not matched;*
- *if d is i-sewn and $m(d)$ is i-free then $\alpha_i(d)$ is not matched;*
- *if both d and $m(d)$ are i-sewn then $m(\alpha_i(d)) = \alpha'_i(m(d))$ or both $\alpha_i(d)$ and $\alpha'_i(m(d))$ are not matched.*

3. *matched darts satisfy constraint (iii) of Def. 1,* i.e.,
$\forall d \in D$ *such that* $m(d) \neq \epsilon$, $\forall i, j \in [0, n]$ *such that* $i + 2 \leq j$, *we have* $(\alpha_i(d) \neq d \wedge m(\alpha_i(d)) \neq \epsilon \wedge \alpha_j(d) \neq d \wedge m(\alpha_j(d)) \neq \epsilon) \Rightarrow m(\alpha_i(\alpha_j(d))) \neq \epsilon$.

Definition 7 *(submap induced by a matching) Let $M = (D, \alpha_0, \ldots, \alpha_n)$ and $M' = (D', \alpha'_0, \ldots, \alpha'_n)$ be two nG-maps. The submap induced by a consistent matching $m : D \to D' \cup \{\epsilon\}$ is $M_{\downarrow m} = (D'', \alpha''_0, \ldots, \alpha''_n)$ such that*

- $D'' = \{d \in D \mid m(d) \neq \epsilon\}$;
- $\forall d \in D''$, $\forall i \in [0, n]$, *if $\alpha_i(d) \in D''$ then $\alpha''_i(d) = \alpha_i(d)$; otherwise $\alpha''_i(d) = d$.*

Proposition 1 *Let M and M' be two nG-maps and m be a consistent matching from M to M'. $M_{\downarrow m}$ is an nG-map such that $M_{\downarrow m} \sqsubseteq M$ and $M_{\downarrow m} \sqsubseteq M'$.*

Proof. Cond. 3 of Def. 6 ensures that $M_{\downarrow m}$ is a consistent nG-map which satisfies constraints of Def. 1. $M_{\downarrow m}$ is a submap of M by Def. 7. $M_{\downarrow m}$ is a submap of M' because Cond. 1 of Def. 6 ensures that darts of M are matched to different darts of M' and because Cond. 2 and 3 of Def. 6 ensure that all involutions are preserved by m. □

Algorithm 1 describes a polynomial-time algorithm which computes a consistent matching m in an incremental way. It starts from an empty matching. At each iteration, it chooses a couple (d, d') which is candidate to be added to m (line 5) according to a choice procedure described in Section 4.1. When adding this couple of darts to m, some other new couples must also be added in order to ensure its consistency. This propagation step is done line 6 and is described in Section 4.2. If propagation detects an inconsistency, then another couple of darts must be chosen (repeat loop lines 4-7); otherwise, m is updated with respect to the results of the propagation step (line 9) and the set of candidate couples is

Algorithm 1: ApproxD(M, M')

Input: two nG-maps $M = (D, \alpha_0, \ldots, \alpha_n)$ and $M' = (D', \alpha'_0, \ldots, \alpha'_n)$
Output: returns an upper bound of $d(M, M')$

```
1  Let m be the set of matched darts; initialize m to ∅
2  Cand← D × D'
3  while Cand≠ ∅ do
4      repeat
5          choose (d, d') ∈Cand and remove (d, d') from Cand
6          m' ← propagate(m, d, d')
7      until m' is consistent or Cand= ∅ ;
8      if m' is consistent then
9          m ← m'
10         update Cand with respect to m'
11 return 1 − |m| / max(|M|,|M'|)
```

updated (line 10) as described in Section 4.3. When no more couple of darts can be consistently added to m, an upper bound of the distance is computed with respect to m (line 11). Note that the size of $M_{\downarrow m}$ is actually equal to the number of couples of darts in m.

4.1 Choice of a couple of darts (line 5)

At each iteration, we choose a couple of darts $(d, d') \in Cand$ to be added to m. This choice is done in a greedy way, by choosing a couple of darts which maximizes the number of preserved involutions, *i.e.*, a couple (d, d') such that $|\{i \in [0, n] | m(\alpha_i(d)) = \alpha'_i(d'))\}|$ is maximal. The goal is to favor the choice of darts which are sewn to already matched darts: the more sewn, the better.

There usually exist several couples which maximize preserved involutions. We have defined two different ways of breaking ties, which are experimentally compared in Section 5. The first way to break ties (denoted *Rand*) consists in randomly choosing a couple within the set of candidate couples which maximize preserved involutions. The second way to break ties (denoted *Deg*) is defined with respect to degrees and co-degrees of incident cells. Indeed, each dart d is incident to $n + 1$ i-cells (one in each dimension): the 0-cell corresponds to the vertex incident to d, the 1-cell to the edge, the 2-cell to the face, the 3-cell to the volume, etc. Each i-cell (with $0 \leq i < n$) has a degree, which is the number of incident $(i+1)$-cells (*e.g.*, the degree of a vertex (resp. edge, face) is the number of incident edges (resp. faces, volumes)). Each i-cell (with $0 < i \leq n$) also has a co-degree, which is the number of incident $(i-1)$-cells (*e.g.*, the co-degree of an edge (resp. face, volume) is the number of incident vertices (resp. edges, faces)). Hence, when several couples of candidate darts preserve the same number of involutions, *Deg* break ties by choosing the couple of darts which maximizes the number of i-cells with the same degrees and co-degrees, thus favoring the choice of darts which have a similar neighborhood. Further ties are randomly broken.

4.2 Propagation of a couple of darts to ensure consistency (line 6)

Adding (d, d') to m may lead to an inconsistent matching which does not satisfies Cond. 3 of Def. 6 (such as the one displayed in Fig. 3). Hence, we don't simply add (d, d') to m but compute the whole set m' of couples of darts that must be added to m so that the matching is consistent according to Def. 6. This propagation is done recursively for each new couple of darts added to m'. It either stops when the matching has been completed to a consistent matching, or when an inconsistency has been detected (when it is not possible to consistently add (d, d') to m). In this latter case, the repeat loop (lines 4-7) must be re-iterated to choose another couple of darts to be added to m.

4.3 Update of *Cand* with respect to m' (line 10)

A consistent matching m must satisfy the 3 conditions of Def. 6. Condition 3 is ensured by the propagation step. Conditions 1 and 2 are ensured by removing from the set *Cand* every couple (d_k, d'_k) which does not satisfy them. More precisely, we remove from *Cand* every couple (d_k, d'_k) such that d_k or d'_k are already matched to another dart in m' (Cond. 1 of Def. 6), or such that there exists a couple of matched darts $(d, d') \in m'$ such that d is i-sewn with d_k whereas d' is not i-sewn with d'_k for some dimension i (Cond. 2 of Def. 6).

5 First experimental results

ApproxD (described in Algorithm 1) greedily computes a consistent matching m for two nG-maps M and M'. This matching induces a common submap $M_{\downarrow m}$ which can be used to compute an upper bound of the distance between M and M', corresponding to $1 - \frac{|m|}{max(|M|,|M'|)}$ (as $|m| = |M_{\downarrow m}|$).

ApproxD is not deterministic as ties are randomly broken when choosing a couple of darts (line 5). Therefore, we can run it several times and finally return the smallest computed upper bound. Let us denote *Rand(k)* (resp. *Deg(k)*) the algorithm which runs ApproxD k times and uses the *Rand* (resp. *Deg*) procedure to break ties when choosing a couple of darts (line 5). Note that for each different run of ApproxD, we enforce the algorithm to choose a different couple of darts when choosing the first couple: for *Rand(k)* we randomly order the $|M| \cdot |M'|$ possible couples of darts whereas for *Deg(k)* we order them by decreasing order with respect to the number of i-cells with same (co-)degrees (see Section 4.1)[4].

We compare *Rand*(k) and *Deg*(k) on a synthetic benchmark of randomly generated couples of 3G-maps. As we cannot compute the exact distance, we have generated couples of 3G-maps for which we know an upper bound on the

[4] We have also performed experiments where ApproxD chooses the next couple of darts (line 5) completely randomly (*i.e.*, independently from the number of preserved involutions), or where ApproxD does not perform the propagation step (line 6). In both cases, the computed matchings are much poorer and induce very bad approximations of the distance so that we do not report results of these variants.

p	$Deg(1000)$				$Rand(1000)$			
	Upper bound		#constructions		Upper bound		#constructions	
	avg	(sdv)	avg	(sdv)	avg	(sdv)	avg	(sdv)
10 ($d \leq 0.9$)	0.309	(0.0038)	330	(245)	0.324	(0.0031)	498	(276)
20 ($d \leq 0.8$)	0.274	(0.0030)	406	(281)	0.311	(0.0150)	467	(303)
30 ($d \leq 0.7$)	0.258	(0.0028)	459	(211)	0.293	(0.0204)	435	(279)
40 ($d \leq 0.6$)	0.238	(0.0022)	451	(280)	0.275	(0.0204)	570	(271)
50 ($d \leq 0.5$)	0.209	(0.0025)	469	(194)	0.252	(0.0324)	502	(289)
60 ($d \leq 0.4$)	0.186	(0.0015)	645	(219)	0.229	(0.0388)	545	(261)
70 ($d \leq 0.3$)	0.158	(0.0014)	362	(323)	0.212	(0.0545)	471	(277)
80 ($d \leq 0.2$)	0.107	(0.0009)	271	(304)	0.155	(0.0493)	418	(282)
90 ($d \leq 0.1$)	0.068	(0.0005)	94	(175)	0.127	(0.0749)	581	(251)

Table 1. Comparison of $Deg(1000)$ and $Rand(1000)$ on (p%) instances. Each line first gives the value of p and the upper bound on d corresponding to p (*i.e.*, $1-p/100$). Then, for each algorithm, it successively gives the best computed upper bound within a limit of 1000 matching constructions, and the number of matching constructions needed to reach this upper bound (average and standard deviation on 50 runs).

distance by construction: we randomly generate a first 3G-map M with 2000 darts; we then extract a submap of M which has p% of its 2000 darts; we finally generate a new 3G-map M', starting from the submap of M by randomly adding (and sewing) new darts to it until it has 2000 darts. We have generated different couples of 3G-maps, with different values of p, denoted (p%). By construction, we know that the distance for an instance (p%) is lower or equal to $1 - p/100$. However, this is just a bound as the parts of M and M' which do not belong to the p% common part may also have some common patterns.

Table 1 compares $Deg(1000)$ and $Rand(1000)$ on nine (p%) instances, with p ranging from 10 to 90. For each instance, we report average results on 50 runs. On these instances, $Deg(1000)$ always computes a smaller upper bound than $Rand(1000)$. The difference is more particularly sensible for larger values of p. In particular, the bound computed by $Deg(1000)$ is twice as small as the one computed by $Rand(1000)$ when $p = 90$. Bounds computed by $Deg(1000)$ and $Rand(1000)$ are much smaller than $1 - p/100$ for small values of p, thus showing that the parts of M and M' which do not belong to the p% common part actually also share many sub-patterns. Note also that $Deg(1000)$ always computes upper bounds smaller than $1 - p/100$, whereas $Rand(1000)$ has not been able to compute an upper bound smaller or equal to $1 - p/100$ when $p = 90$. Standard deviations of bounds are also smaller for $Deg(1000)$ than for $Rand(1000)$.

Table 1 also gives the number of matching constructions needed to reach the best upper bound (within a limit of 1000 constructions). It shows us that Deg usually needs less constructions than $Rand$ (whereas it computes better approximations). Note also that the number of constructions needed by Deg to find the best upper bound decreases for large values of p. In particular, the best approximation is found in a hundred or so constructions when $p = 90$%. Fig. 4 displays the evolution of the quality of the bound with respect to the number k of matching constructions for three different values of p. It shows us that the upper

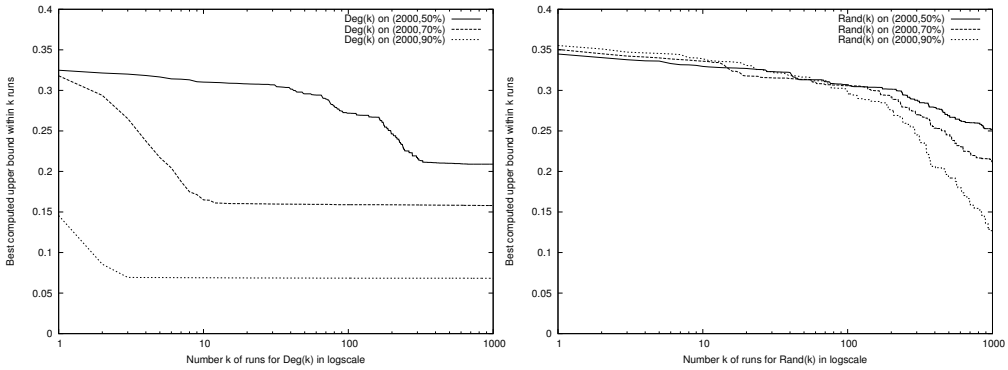


Fig. 4. Evolution of the upper bound w.r.t. the number of matching constructions (average on 50 runs) for instances (50%), (70%) and (90%): $Deg(k)$ on the left and $Rand(k)$ on the right.

bound computed by $Deg(k)$ decreases quicker than the upper bound computed by $Rand(k)$, and that the higher p, the quicker it decreases.

Let us finally point out that matching constructions are rather quickly computed. For instance, on an Intel Xeon E5520 CPU with 16GB RAM, Algorithm 1 runs in less than 1 second (resp. 2s, 11s, 56s) when 3G-maps have 1000 (resp. 2000, 4000, and 8000) darts. Note that CPU times do not really depend on the size of the common part p.

6 Conclusion

We have introduced a first distance measure for comparing nG-maps. This distance is based on the size of the largest common submap and we have shown that it is a metric. We have described a polynomial time algorithm which is able to compute an approximation of this distance and very first experimental results have shown us that this algorithm uses relevant heuristics, which allow it to find better approximations, and that the computed approximations are always better than bounds obtained by construction. Further work will concern the evaluation of our distance measure on a real image retrieval application where images are modeled by 2G-maps. We also plan to extend our work to partial submaps, such that not all involutions are preserved by the matching, and more generally, to a nG-map edit distance which evaluates the distance between two generalized maps by means of the smallest number of edit operations needed to transform one nG-map into the other, in way similar to graph edit distances [Bun97].

References

[BS98] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *PRL*, 19(3-4):255–259, 1998.

[Bun97] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *PRL*, 18:689–694, August 1997.

[Dam08] G. Damiand. Topological model for 3d image representation: Definition and incremental extraction algorithm. *CVIU*, 109(3):260–289, 2008.

[DBF04] G. Damiand, Y. Bertrand, and C. Fiorio. Topological model for two-dimensional image representation: definition and optimal extraction algorithm. *CVIU*, 93(2):111–154, February 2004.

[DDLHJ+09] G. Damiand, C. De La Higuera, J.-C. Janodet, E. Samuel, and C. Solnon. Polynomial algorithm for submap isomorphism: Application to searching patterns in images. In *GBR*, volume 5534 of *LNCS*, pages 102–112, Venice, Italy, May 2009. Springer Berlin/Heidelberg.

[Lie94] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275–324, 1994.