Efficient Search of Combinatorial Maps using Signatures

Stéphane Gosselin, Christine Solnon, Guillaume Damiand

Efficient Search of Combinatorial Maps using Signatures

Stéphane Gosselin, Guillaume Damiand, Christine Solnon

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

Abstract

In this paper, we address the problem of computing canonical representations of n-dimensional combinatorial maps and of using them for efficiently searching for a map in a database. We define two combinatorial map signatures: the first one has a quadratic space complexity and may be used to decide of isomorphism with a new map in linear time whereas the second one has a linear space complexity and may be used to decide of isomorphism in quadratic time. We show that these signatures can be used to efficiently search for a map in a database.

Keywords:
Combinatorial map, signature, map isomorphism

1. Motivations

Combinatorial maps are nice data structures for modelling the subdivision of a space into cells. First defined in 2D [7, 15, 9, 3], they have been extended to nD [2, 11, 12] and model the subdivision of an object in cells, and all the adjacency and incidence relations in any dimension. Hence, combinatorial maps are often used to model the partition of an image in regions and to describe the topology of this partition (e.g., [1] for $2D$ images and [4] for $3D$ images). There exist efficient image processing algorithms using this topological information.

Email addresses: `stephane.gosselin@liris.cnrs.fr` (Stéphane Gosselin), `guillaume.damiand@liris.cnrs.fr` (Guillaume Damiand), `christine.solnon@liris.cnrs.fr` (Christine Solnon)

Article published in Theoretical Computer Science, Volume 412, Pages 1392-1405, DOI: 10.1016/j.tcs.2010.10.029.

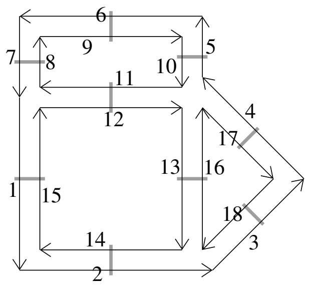
Our goal is to define new algorithms for classifying images modelled by combinatorial maps. More precisely, we propose to characterize image classes by extracting patterns (submaps) which occur frequently in these classes. Finding frequent patterns in large databases is a classical data mining problem, the tractability of which highly depends on the existency of efficient algorithms for deciding if two patterns are actually different or if they are two occurrences of a same object. Hence, if finding frequent subgraphs is intractable in the general case, it may be solved in incremental polynomial time when considering classes of graphs for which subgraph isomorphism may be solved in polynomial time, such as trees or outerplanar graphs [8].

In this paper, we address the problem of computing canonical representations of combinatorial maps which may be used to efficiently search for a map in a database. This work is related to [10], which introduces a polynomial algorithm for deciding of the isomorphism of ordered graphs (i.e., graphs such that the set of nodes adjacent to every node is ordered), based on a vertex labelling. Recently, this work has been extended to combinatorial maps by proposing a polynomial algorithm for map and submap isomorphism based on a traversal of the map [6].

Contribution. In this paper, we define canonical representations of combinatorial maps which are based on these traversal and labelling principles. More precisely, we define two map signatures. Both signatures may be computed in quadratic time with respect to the size of the map. The first signature (called Set Signature) is a set of words and is modelled by a lexicographical tree. It has a quadratic space complexity and it allows us to decide if a new map is isomorphic to a map modelled by this signature in linear time. The second signature (called Word Signature) is a word and has a linear space complexity. As a counterpart, isomorphism with a new map is quadratic.

We also show that these signatures can be used to efficiently search for a map into a database of maps. More precisely, each map of the database is modelled by its signature and the different signatures are merged into a tree. Space and time complexities depend on the considered signature (Set or Word Signature): the first one is faster but needs more space.

Outline. Basic definitions on combinatorial maps are recalled in section 2. The two map signatures are defined in section 3. We show how to use these signatures to model a database of maps in section 4. In sections 3 and 4, we only consider connected maps. We show how to extend this work to non



d	1	2	3	4	5	6	7	8	9
$\beta_1(d)$	2	3	4	5	6	7	1	9	10
$\beta_2(d)$	15	14	18	17	10	9	8	7	6

d	10	11	12	13	14	15	16	17	18
$\beta_1(d)$	11	8	13	14	15	12	17	18	16
$\beta_2(d)$	5	12	11	16	2	1	13	4	3

Figure 1: $2D$ combinatorial map example. Darts are represented by numbered black arrows. Two darts 1-sewn are drawn consecutively, and two darts 2-sewn are concurrently drawn and in reverse orientation, with a little grey segment between the two darts.

connected maps in section 5. Finally, we experimentally evaluate our work in section 6.

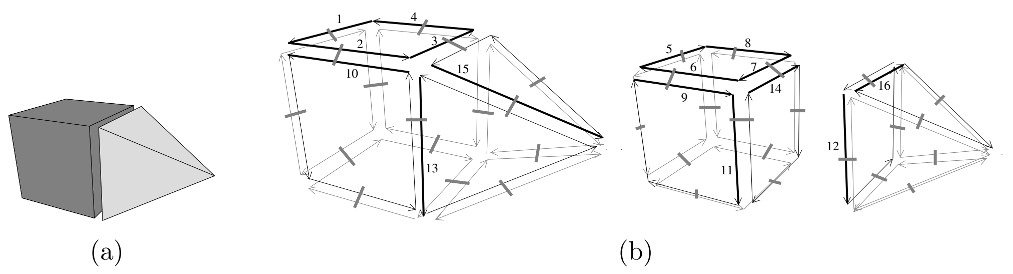
2. Recalls on Combinatorial Maps

Definition 1 (Combinatorial map [12]). An nD combinatorial map (or n-map) is defined by a tuple $M = (D, \beta_1, \dots, \beta_n)$ where

- D is a finite set of darts;
- β_1 is a *permutation* on D, *i.e.*, a one-to-one mapping from D to D;
- $\forall\, 2 \le i \le n$, β_i is an *involution* on D, *i.e.*, a one-to-one mapping from D to D such that $\beta_i = \beta_i^{-1}$;
- $\forall\, 1 \le i \le n-2$, $\forall\, i+2 \le j \le n$, $\beta_i \circ \beta_j$ is an *involution* on D.

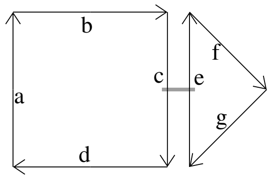
A dart d is said to be i-sewn with another dart d' if $d = \beta_i(d')$. β_1 is a permutation which models edge successions when turning around $2D$ cells (i.e. faces) with respect to some given order. We note β_0 for β_1^{-1} so that β_0 models edge successions when turning around $2D$ cells with respect to the opposite order. Fig. 1 and 2 give examples of $2D$ and $3D$ combinatorial maps.

In some cases, it may be useful to allow some β_i to be partially defined, thus leading to open combinatorial maps. The basic idea is to add a new element ϵ to the set of darts, and to allow darts to be i-sewn with ϵ. By definition, $\forall 0 \le i \le n$, $\beta_i(\epsilon) = \epsilon$. Fig. 3 gives an example of open map



d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
$\beta_1(d)$	2	3	4	1	8	5	6	7	11	...	...	...	...	...	...	...
$\beta_2(d)$	...	10	...	...	...	9	14	...	6	2	...	...	...	7	...	...
$\beta_3(d)$	5	6	7	8	1	2	3	4	10	9	...	...	...	...	...	...

Figure 2: An example of a 3D combinatorial map. (a) A 3D object. (b) The corresponding 3D combinatorial map (external volume on the left; interior on the middle and the right). The graphical convention is the same as in 2D. β_3 is not drawn, but (partially) given in the array.



	a	b	c	d	e	f	g
β_1	b	c	d	a	f	g	e
β_2	ϵ	ϵ	e	ϵ	c	ϵ	ϵ

Figure 3: Open combinatorial map example. Darts a, b, d, f and g are not 2-sewn.

(see [14] for precise definitions). In this paper, we always consider open combinatorial maps.

A map is connected if there exists a path of sewn darts between every pair of darts.

Definition 2 (Connected map). An n-map $M = (D, \beta_1, \ldots, \beta_n)$ is connected if $\forall d \in D, \forall d' \in D$, there exists a path $(d_1, \ldots, d_k)$ such that $d_1 = d$, $d_k = d'$ and $\forall i \in \{1, \ldots, k\}, \exists j_i \in \{0, \ldots, n\}, d_{i+1} = \beta_{j_i}(d_i)$.

Lienhardt has defined isomorphism between two combinatorial maps as follows.

Definition 3 (Map isomorphism [13]). Two n-maps $M = (D, \beta_1, \ldots, \beta_n)$ and $M' = (D', \beta'_1, \ldots, \beta'_n)$ are isomorphic if there exists a one-to-one mapping $f : D \rightarrow D'$, called *isomorphism function*, such that $\forall d \in D, \forall i \in \{1, \ldots, n\}$ $f(\beta_i(d)) = \beta'_i(f(d))$.

This definition has been extended to open maps in [6] by adding that $f(\epsilon) = \epsilon$, thus enforcing that, when a dart is i-sewn with ϵ, then the dart matched to it by f is i-sewn with ϵ.

3. Signatures of connected maps

In this section, we introduce two different canonical representations of maps, called signatures. We only consider connected maps; the extension of this work to non connected maps is discussed in Section 5.

3.1. Labelling of a connected map

Our signatures are based on map labellings, which associate a different label with every different dart. By definition, the label associated with ϵ is 0.

Definition 4 (Labelling). Given an n-map $M = (D, \beta_1, \ldots, \beta_n)$ a labelling of M is a bijective function $l : D \cup \{\epsilon\} \rightarrow \{0, \ldots, |D|\}$ such that $l(\epsilon) = 0$.

Example 1. $l = \{\epsilon : 0,\ a : 3,\ b : 1,\ c : 5\ , d : 7,\ e : 2,\ f : 6,\ g : 4\}$ *is a labelling of the map displayed in Fig. 3.*

One may compute a labelling of a map by performing a map traversal and labelling darts with respect to the order in which they are discovered. Different labellings may be computed, depending on (i) the initial dart from which the traversal is started, (ii) the strategy used to memorize the darts that have been discovered but that have not yet been treated (e.g., FIFO or LIFO), and (iii) the order in which the β_i functions are used to discover new darts.

We define below the labelling corresponding to a breadth first traversal of a map where β_i functions are used in increasing order.

Definition 5 (Breadth first labelling (BFL)). Given a connected n-map $M = (D, \beta_1, \ldots, \beta_n)$ and a dart $d \in D$ the breadth first labelling associated with (M, d) is the labelling returned by the function $BFL(M, d)$ described in algorithm 1.

Algorithm 1: $BFL(M, d)$

```
Input: a connected $n$-map $M = (D, \beta_1, \dots, \beta_n)$, and a dart $d \in D$
Output: a labelling $l : D \cup \{\epsilon\} \to \{0, \dots, |D|\}$
1  for each $d' \in D$ do $l(d') \leftarrow -1$
2  $l(\epsilon) \leftarrow 0$
3  let $Q$ be an empty queue
4  add $d$ at the end of $Q$
5  $l(d) \leftarrow 1$
6  $nextLabel \leftarrow 2$
7  while $Q$ is not empty do
8      remove $d'$ from the head of $Q$
9      for $i$ in $0 \dots n$ do
10         if $l(\beta_i(d')) = -1$ then
11             $l(\beta_i(d')) \leftarrow nextLabel$
12             $nextLabel \leftarrow nextLabel + 1$
13             add $\beta_i(d')$ at the end of $Q$
14 return $l$
```

Example 2. *The breadth first labellings associated with the map of Fig. 3 for darts a and e respectively are*

$$\begin{aligned} BFL(M,a) &= \{\epsilon : 0,\ a : 1,\ b : 3,\ c : 4,\ d : 2,\ e : 5,\ f : 7,\ g : 6\} \\ BFL(M,e) &= \{\epsilon : 0,\ a : 7,\ b : 5,\ c : 4,\ d : 6,\ e : 1,\ f : 3,\ g : 2\} \end{aligned}$$

Proposition 1. Algorithm 1 returns a labelling.

Proof.

- $l(\epsilon)$ is set to 0 in line 2.
- $\forall d, d' \in D, d \neq d' \Rightarrow l(d) \neq l(d')$. Indeed, each time a label is assigned to a dart (line 11), *nextLabel* is incremented (line 12).
- $\forall d \in D, 1 \leq l(d) \leq |D|$. Indeed, each dart enters exactly once in the queue because (i) the map is connected and (ii) a dart enters the queue only if it has not yet been labelled, and it is labelled just before entering it. □

Proposition 2. The time complexity of algorithm 1 is $\mathcal{O}(n \cdot |D|)$

PROOF. The while loop (lines 7-13) is iterated $|D|$ times as (i) exactly one dart d is removed from the queue at each iteration; and (ii) each dart $d \in D$ enters the queue exactly once. The for loop (lines 9-13) is iterated $n+1$ times. □

Note that the for loop (lines 9-13) iterates for every $i \in \{0, \dots, n\}$, including 0. Indeed, as we consider open maps, some darts may not be 1-sewn. In this case, some darts may not be reachable from the initial dart d without using β_0. Let us consider for example the open map displayed in Fig. 3, and let us suppose that dart b has been removed. If BFL is started from the initial dart a, then no dart will be discovered if we only use β_1 and β_2 to discover new darts (as $\beta_1(a) = \beta_2(a) = \epsilon$ in this case). However, if we use β_0, β_1, and β_2, we can actually discover all darts.

Given a map M and a labelling l, one may describe M (i.e., its functions β_1 to β_n) by a sequence of labels of l. The idea is to first list the n labels of the n darts which are i-sewn with the dart labelled by 1 (*i.e.*, $l(\beta_1(1)), \dots, l(\beta_n(1))$), and then by 2 (*i.e.*, $l(\beta_1(2)), \dots, l(\beta_n(2))$), etc. More formally, we define the word associated with a map and a labelling as follows.

Definition 6 (Word). Given a connected n-map $M = (D, \beta_1, \dots, \beta_n)$ and a labelling $l : D \cup \{\epsilon\} \to \{0, \dots, |D|\}$ the word associated with (M, l) is the sequence

$$W(M,l) = < w_1, \dots, w_{n \cdot |D|} >$$

such that $\forall i \in \{1, \dots, n\}$, $\forall k \in \{1, \dots, |D|\}$, $w_{i \cdot k} = l(\beta_i(d_k))$ where d_k is the dart labelled with k, i.e., $d_k = l^{-1}(k)$.

Notation. The word associated with the breadth first labelling of a map M, starting from a dart d, is denoted $W_{BFL}(M, d)$, i.e.,

$$W_{BFL}(M,d) = W(M, BFL(M,d))$$

Example 3. *The words associated with the map of Fig. 3 for the two labellings of example 2 respectively are*

$$\begin{aligned} W_{BFL}(M,a) &= < 3,0,1,0,4,0,2,5,7,4,5,0,6,0 > \\ W_{BFL}(M,e) &= < 3,4,1,0,2,0,6,1,4,0,7,0,5,0 > \end{aligned}$$

Algorithm for building $W_{BFL}(M,l)$. Given an n-map $M = (D, \beta_1, \ldots, \beta_n)$, the word $W_{BFL}(M,l)$ is computed by calling $BFL(M,l)$ and considering every dart of D in increasing label order and enumerating the labels of its n i-sewn darts. Note that we do not have to sort darts with respect to their labels as we can save this order during the run of BFL. Hence, the time complexity of the construction of the word $W_{BFL}(M,l)$ is $\mathcal{O}(n \cdot |D|)$.

The key point which allows us to use words for building signatures is that two maps are isomorphic if and only if they share a word for a breadth first labelling, as stated in theorem 1.

Theorem 1. *Two connected n-maps $M = (D, \beta_1, \ldots, \beta_n)$ and $M' = (D', \beta'_1, \ldots, \beta'_n)$ are isomorphic iff there exist $d \in D$ and $d' \in D'$ such that $W_{BFL}(M,d) = W_{BFL}(M',d')$*

Proof. $\Rightarrow$ Let us first consider two isomorphic n-maps $M = (D, \beta_1, \ldots, \beta_n)$ and $M' = (D', \beta'_1, \ldots, \beta'_n)$, and let us show that there exist two darts d and d' such that $W_{BFL}(M,d) = W_{BFL}(M',d')$. If M and M' are isomorphic then there exists $f : D \to D'$ such that $\forall d \in D, \forall i \in \{1, \ldots, n\}, f(\beta_i(d)) = \beta'_i(f(d))$ (Def. 3). Let d_1 be a dart of D, and let us note l (resp. l') the labellings returned by $BFL(M, d_1)$ (resp. $BFL(M', f(d_1))$). Claim 1: l and l' are such that $\forall d_i \in D, l(d_i) = l'(f(d_i))$. This is true for the initial dart d_1 as both d_1 and $f(d_1)$ are labelled with 1 at the beginning of each traversal. This is true for every other dart $d_i \in D$ as the traversals of M and M' performed by BFL are completely determined by the fact that (i) they consider the same FIFO strategy to select the next labelled dart which will be used to discover new darts and (ii) they use the β_i functions in the same order to discover new darts from a selected labelled dart. Claim 2: $\forall k \in \{1, \ldots, |D|\}, f(l^{-1}(k)) = l'^{-1}(k)$. This is a direct consequence of Claim 1. Conclusion: $\forall i \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, |D|\}$, the $i.k^{th}$ element of $W_{BFL}(M, d_1)$ is equal to the $i.k^{th}$ element of $W'_{BFL}(M', f(d_1))$, i.e., $l(\beta_i(l^{-1}(k))) = l'(\beta'_i(l'{-}1(k)))$. Indeed,

$$\begin{aligned} l(\beta_i(l^{-1}(k))) &= l'(f(\beta_i(l^{-1}(k)))) && \text{(because of Claim 1)} \\ &= l'(\beta'_i(f(l^{-1}(k)))) && \text{(because } f \text{ is an isomorphism function)} \\ &= l'(\beta'_i(l'^{-1}(k))) && \text{(because of Claim 2)} \end{aligned}$$

$\Leftarrow$ Let us now consider two n-maps $M = (D, \beta_1, \ldots, \beta_n)$ and $M' = (D', \beta'_1, \ldots, \beta'_n)$ and two darts d and d' such that $W_{BFL}(M,d) = W_{BFL}(M',d')$,

and let us show that M and M' are isomorphic. Let us note l (resp. l') the labellings returned by $BFL(M, d)$ (resp. $BFL(M', d')$), and let us define the function $f : D \to D'$ which matches darts with same labels, i.e., $\forall d_j \in D, f(d_j) = l'^{-1}(l(d_k))$. Note that this implies as well that $l(d_j) = l'(f(d_j))$.
<u>Claim 3</u>: $\forall i \in \{1, \dots, n\}, \forall k \in \{1, \dots, |D|\}, l(\beta_i(l^{-1}(k))) = l'(\beta'_i(l'-1(k)))$. This comes from the fact that $W_{BFL}(M, d) = W_{BFL}(M', d')$ so that the $i.k^{th}$ element of $W_{BFL}(M, d_1)$ is equal to the $i.k^{th}$ element of $W'_{BFL}(M', f(d_1))$.
<u>Conclusion</u>: $\forall i \in \{1, \dots, n\}, \forall d_j \in D$,

$$\begin{array}{rcll} f(\beta_i(d_j)) = & & l'^{-1}(l(\beta_i(d_j))) & \text{(by definition of } f\text{)} \\ & = & l'^{-1}(l'(\beta'_i(l'^{-1}(l(d_j))))) & \text{(because of Claim 3)} \\ & = & \beta'_i(l'^{-1}(l(d_j))) & \text{(by simplification)} \\ & = & \beta'_i(l'^{-1}(l'(f(d_j)))) & \text{(by definition of } f\text{)} \\ & = & \beta'_i(f(d_j)) & \text{(by simplification)} \end{array}$$

Hence, f is an isomorphism function and M and M' are isomorphic. □

3.2. Set Signature of a connected map

A map is characterized by the set of words associated with all possible breadth first labellings. This set defines a signature.

Definition 7 (Set Signature). Given an n-map $M = (D, \beta_1, \dots, \beta_n)$, the Set Signature associated with M is $SS(M) = \{W_{BFL}(M, d) | d \in D\}$

Fig. 4 shows the Set Signature of the map of Fig. 3. Note that a Set Signature may contain less than $|D|$ words as there may exist different darts d and d' such that $W_{BFL}(M, d) = W_{BFL}(M, d')$ (in case of automorphisms).

Theorem 2. *$SS(M)$ is a signature, i.e., two connected maps M and M' are isomorphic if and only if $SS(M) = SS(M')$*

Proof. $\Rightarrow$ Let us consider two isomorphic maps $M = (D, \beta_1, \dots, \beta_n)$ and $M' = (D', \beta'_1, \dots, \beta'_n)$, and let us show that $SS(M) = SS(M')$. This is a direct consequence of theorem 1, which ensures that given an isomorphism function f between M and M' we have, for every dart $d \in D$, $W_{BFL}(M, d) = W_{BFL}(M', f(d))$. Hence, every word of $SS(M)$, computed from any dart of D, necessarily belongs to $SS(M')$ (and conversely).

$\Leftarrow$ Let us consider two maps $M = (D, \beta_1, \ldots, \beta_n)$ and $M' = (D', \beta'_1, \ldots, \beta'_n)$ such that $SS(M) = SS(M')$, and let us show that M and M' are isomorphic. Indeed, there exist two words $W \in SS(M)$ and $W' \in SS(M')$ such that $W = W'$, thus M and M' are isomorphic due to theorem 1. □

Note that a direct consequence of theorem 1 and theorem 2 is that for two non isomorphic maps M and M', $SS(M) \cap SS(M') = \emptyset$.

The Set Signature of a map M may be represented by a lexicographical tree which groups common prefixes of words.

Definition 8 (Set Signature Tree of a map). Given an n-map $M = (D, \beta_1, \ldots, \beta_n)$, the tree associated with the Set Signature of M is the tree $T_{SS}(M)$ such that

- every node u except the root has a label $l(u)$ which is an integer ranging between 0 and $|D|$; we note $w(u)$ the word obtained by concatening all these labels along the path from the root to u;
- for every node u, all the children of u have different labels;
- there are $|SS(M)|$ leaves and for every leaf u, we have $w(u) \in SS(M)$.

For example, the Set Signature Tree of the map displayed in Fig. 3 is displayed in Fig. 4.

Property 1. The space complexity of the Set Signature Tree $T_{SS}(M)$ of a map M is $\mathcal{O}(n \cdot |D|^2)$.

PROOF. The tree contains one leaf for each word in the signature, i.e., at most $|D|$ leaves, and the length of each path from the root to a leaf is $n \cdot |D|$. Hence, the tree has $\mathcal{O}(n \cdot |D|^2)$ nodes. At each node u of the tree, we use a list to memorize all its children. The sum of the sizes of all lists is equal to the number of edges of the tree which is equal to the number of its nodes minus one. □

Property 2. The time complexity for building the Set Signature Tree $T_{SS}(M)$ of a map M is $\mathcal{O}(n \cdot |D|^2)$.

PROOF. The tree can be built in an incremental way: starting from the tree which only contains the root, we iteratively add $W_{BFL}(M, d)$ to it, for every dart $d \in D$. The time complexity for computing a word is $\mathcal{O}(n \cdot |D|)$. The

$$\begin{aligned}\{&< 3,0,1,0,4,0,2,5,7,4,5,0,6,0 >,\\ &< 3,0,1,0,4,5,2,0,7,3,5,0,6,0 >,\\ &< 3,0,1,4,2,0,6,2,4,0,7,0,5,0 >,\\ &< 3,4,1,0,2,0,6,1,4,0,7,0,5,0 >,\\ &< 3,4,1,0,5,0,7,1,2,0,4,0,6,0 >,\\ &< 3,0,1,0,2,4,6,3,4,0,7,0,5,0 >,\\ &< 3,0,1,5,4,0,2,0,7,2,5,0,6,0 >\}\end{aligned}$$

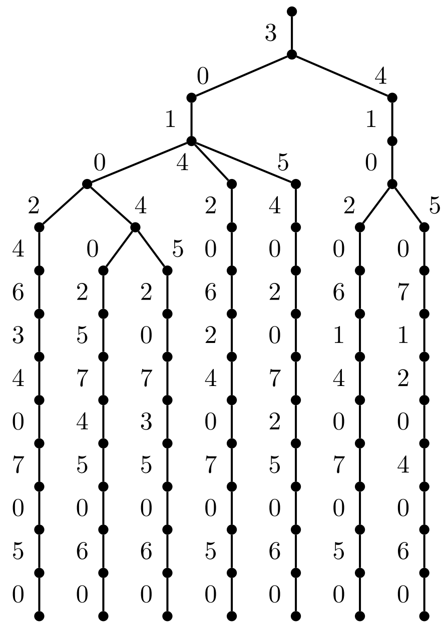


Figure 4: Set Signature of the map of Fig. 3. On the left, the set of words $SS(M)$, on the right, the tree $T_{SS}(M)$ (for each node u, $l(u)$ is displayed above u).

time complexity for adding it to the current tree is also $\mathcal{O}(n \cdot |D|)$. Indeed, the length of a word is $n \cdot |D|$. For each label x of the word, we mainly have to decide if the current node of the tree has a child u such that $l(u) = x$. This is done in linear time with respect to the number of children of the current node as we use lists to memorize node children. A node has at most $|D| + 1$ children. However, the sum of the number of children of all nodes between the root and a leaf is bounded by the length of the path between the root and a leaf plus the number of leaves, i.e., $n \cdot |D| + |D|$. □

Property 3. Given an n-map $M = (D, \beta_1, \ldots, \beta_n)$ and the Set Signature Tree $T_{SS}(M')$ of another map M', the complexity of deciding of the isomorphism between M and M' is $\mathcal{O}(n \cdot |D|)$.

Proof. To decide of the isomorphism, we build a breadth first labelling, starting from any dart $d \in D$, and decide if $W_{BFL}(M, d)$ corresponds to a path from the root to a leaf of the tree. This is done in linear time with respect to the length of the word (again, if a node of the tree may have up to $|D|+1$ children, the sum of the number of children of all nodes between the root and a leaf is bounded by $n \cdot |D| + |D|$). □

Note that we can check that the word corresponds to a path in the tree during the construction of the word so that we can stop the construction as soon as it does not match a branch in the tree. Note also that this algorithm is optimal. Indeed, to decide of the isomorphism between two maps we have to check if $f(\beta_i(d)) = \beta'_i(f(d))$ for every dart $d \in |D|$ and every dimension $i \in \{1, \ldots, n\}$. This cannot be done in less than $\mathcal{O}(n \cdot |D|)$.

3.3. Word Signature of a connected map

The lexicographical order is a strict total order on the words of a Set Signature, and we have shown that if two Set Signatures share one word, then they are equal. Hence, we may define a map signature by considering the smallest word of the Set Signature.

Definition 9 (Word Signature). Given a map M, the Word Signature of M is, $WS(M) = min(SS(M))$.

Example 4. *The Word Signature of the map displayed in Fig. 3 is*

$$WS(M) =< 3, 0, 1, 0, 2, 4, 6, 3, 4, 0, 7, 0, 5, 0 >$$

Property 4. The space complexity of a Word Signature is $\mathcal{O}(n \cdot |D|)$.

Algorithm for computing $WS(M)$. The Word Signature of a map M is built by calling $BFL(M, d)$ for each dart $d \in D$, and keeping the smallest word with respect to the lexicographical order. The time complexity for computing the Word Signature is $\mathcal{O}(n \cdot |D|^2)$. Note that this process may be improved (without changing the worst case complexity) by incrementally comparing the word in construction with the current smallest word and stopping the construction whenever it becomes greater.

Property 5. Given an n-map $M = (D, \beta_1, \ldots, \beta_n)$ and the Word Signature $WS(M')$ of another map M', we can decide of the isomorphism between M and M' in $\mathcal{O}(n \cdot |D|^2)$.

PROOF. To decide of the isomorphism, we have to build breadth first labellings, starting from every different dart $d \in D$, until either $W_{BFL}(M, d) = WS(M')$ (M is isomorphic to M') or all darts have been tried (M is not isomorphic to M'). In the worst case, we have to build $|D|$ labellings so that the overall time complexity is $\mathcal{O}(n \cdot |D|^2)$. □

4. Signatures of databases of connected maps

Map signatures can be used to decide of the isomorphism of two maps. However, in many cases, we have to compare a map M not only with one other map but with a whole database of maps in order to search for maps isomorphic to M. To this aim, we define the signature of a database of maps. This signature actually merges all the signatures of the maps of the database into a single tree. We can either consider Set or Word Signatures.

In this section, we only consider connected maps; the extension of this work to non connected maps is discussed in Section 5.

4.1. Set Signature of a database of maps

We can model a database $B = \{M^1, \ldots, M^k\}$ of k n-maps by a list of k independent Set Signature Trees. Given a new n-map M, we can search for an isomorphic map in B in $\mathcal{O}(k \cdot n \cdot |D|)$ by iteratively searching for M in each of these trees. This complexity can be improved by merging the k trees into a single tree.

If all maps in the database have the same number of darts, then all branches (between the root and a leaf) have the same length in all trees. In this case, all branches also have the same length in the merged tree and we simply have to memorize, for every leaf u, the set of maps such that $w(u)$ belongs to the Set Signature of the map (the database may contain different maps which are isomorphic so that a same leaf may correspond to several maps).

However, if the number of darts is different from a map to another, then we have to merge trees whose branches have different lengths. In this case, it may happen that the word associated with a leaf in a tree is a prefix of the word associated with another leaf of another tree so that, when merging these two trees, a word may end on a node which is not a leaf. Hence, for each node u of the tree, we memorize the set $m(u)$ of maps such that $w(u)$ belongs to the Set Signature of the map.

Definition 10 (Set Signature Tree of a database of maps). Let $B = \{M^1, \ldots, M^k\}$ be a database of k n-maps and t be the maximum number of darts of the maps of B. The Set Signature Tree of B is the tree $T_{SS}(B)$ such that

- every node u except the root has a label $l(u)$ such that $l(u)$ is an integer ranging between 0 and t; we note $w(u)$ the word obtained by