Frequent Submap Discovery

Stéphane Gosselin, Guillaume Damiand, and Christine Solnon*

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

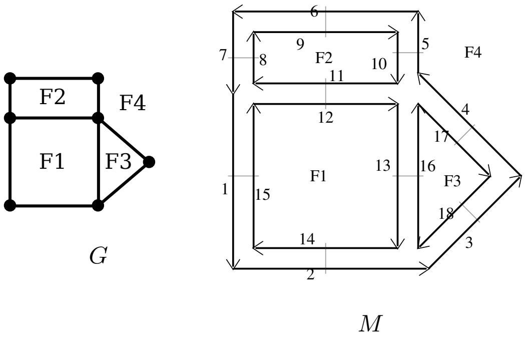
Abstract. Combinatorial maps are nice data structures for modeling the topology of nD objects subdivided in cells (*e.g.*, vertices, edges, faces, volumes, ...) by means of incidence and adjacency relationships between these cells. In particular, they can be used to model the topology of plane graphs. In this paper, we describe an algorithm, called mSpan, for extracting patterns which occur frequently in a database of maps. We experimentally compare mSpan with gSpan on a synthetic database of randomly generated 2D and 3D maps. We show that gSpan does not extract the same patterns, as it only considers adjacency relationships between cells. We also show that mSpan exhibits nicer scale-up properties when increasing map sizes or when decreasing frequency.

1 Introduction

Combinatorial maps are nice data structures for modeling the topology of nD objects subdivided in cells (*e.g.*, vertices, edges, faces, volumes, ...) by means of incidence and adjacency relationships between these cells. First defined in 2D [7, 16, 10, 3], they have been extended to nD [2, 12, 13]. 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. However, there exist few algorithms for analyzing or comparing combinatorial maps, which are key issues in image processing.

In this paper, we describe an algorithm for extracting patterns which occur frequently in a database of maps. This algorithm is a first step for analyzing and characterizing sets of maps. Finding frequent patterns in 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 [9]. We have introduced efficient polynomial-time algorithms to decide of submap isomorphism in [5], and to search for a map into a database of maps in [8]. These algorithms allow us to design an incremental polynomial time algorithm for extracting frequent patterns from a database of maps.

* The authors acknowledge an ANR grant BLANC 07-1_184534: this work was done in the context of project SATTIC.



Face	F4							F2	
Dart	1	2	3	4	5	6	7	8	9
β_1	2	3	4	5	6	7	1	9	10
β_2	15	14	18	17	10	9	8	7	6

Face	F2		F1				F3		
Dart	10	11	12	13	14	15	16	17	18
β_1	11	8	13	14	15	12	17	18	16
β_2	5	12	11	16	2	1	13	4	3

Fig. 1. The map M describes the topology of the plane graph G. Darts are represented by numbered arrows. 1-sewn darts are drawn consecutively, and 2-sewn darts are concurrently drawn and in reverse orientation, with a little grey segment between the two darts. Darts 1 to 7 correspond to face F4, darts 8 to 11 to face F2 and so on.

Outline. Basic definitions on combinatorial maps are recalled in section 2. The algorithm for extracting frequent submaps from $2D$ maps is described in section 3, and its extension to nD maps is described in section 4. First experimental results on a synthetic database of randomly generated 2D and 3D maps are reported in section 5.

2 Recalls on Combinatorial Maps

Combinatorial maps describe the subdivision of nD objects into cells of dimensions lower or equal to n ($0D$ vertices, $1D$ edges, $2D$ faces, $3D$ volumes, ...), and describe the topology of these cells by means of incidence and adjacency relationships between these cells. For sake of simplicity, we first introduce maps in $2D$ and describe our algorithm within this $2D$ context. The extension to nD maps is rather straightforward and is described in section 4.

In $2D$, a combinatorial map models a plane graph *i.e.*, the embedding of a planar graph into a plane, as illustrated in Fig. 1. It is defined by a set of darts and two functions β_1 and β_2 as follows.

Definition 1 (Combinatorial map [13]). *A $2D$ combinatorial map (or map) is defined by a tuple $M = (D, \beta_1, \beta_2)$ 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*); and β_2 is an* involution *on* D *(*i.e.*, a one-to-one mapping from* D *to* D *such that $\beta_2 = \beta_2^{-1}$).*

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 dart successions when turning around faces with respect to some given order. β_2 models adjacency relations between faces.

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,

$\beta_1(\epsilon) = \beta_2(\epsilon) = \epsilon$. Fig. 2 gives an example of open map (see [15] for precise definitions).

In this paper, we extract patterns from maps, where patterns are maps which are isomorphic to submaps of these maps. More precisely, map isomorphism has been defined in [14] as follows.

Definition 2 (Map isomorphism). *Two maps $M = (D, \beta_1, \beta_2)$ and $M' = (D', \beta'_1, \beta'_2)$ are isomorphic if there exists a bijection $f : D \to D'$ such that $\forall d \in D, f(\beta_1(d)) = \beta'_1(f(d))$ and $f(\beta_2(d)) = \beta'_2(f(d))$.*

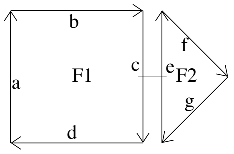
This definition has been extended to open maps in [5] by adding that $f(\epsilon) = \epsilon$, thus enforcing that, when a dart is i-sewn with ϵ, then the dart matched to it by f is i-sewn with ϵ. Submap isomorphism simply derives from the definition of map isomorphism: there is a submap isomorphism from a map M to a map M' if there exists a submap of M' which is isomorphic to M', where a submap is basically obtained by removing some darts (and free-ing darts that were i-sewn with the removed darts). For example, there is a submap isomorphism from the map of Fig. 2 to the map of Fig. 1 as it is isomorphic to the submap of Fig. 1 obtained by removing darts 1 to 11.

In [5], we have described an algorithm which decides of submap isomorphism from a map $M = (D, \beta_1, \beta_2)$ to a map $M' = (D', \beta'_1, \beta'_2)$ in $\mathcal{O}(|D| \cdot |D'|)$, provided that M is connected, *i.e.*, there must exist a path of sewn darts between every pair of darts of M.

In [8], we have introduced a signature which allows us to efficiently search for a map M in a database B containing k maps such that the largest map has t darts: the time complexity for building the signature of the database is $\mathcal{O}(k \cdot t^2)$; the space complexity of this signature is $\mathcal{O}(k \cdot t)$, and the time complexity of searching for all maps of B which are isomorphic to M is $\mathcal{O}(n \cdot t^2)$.

3 Frequent submap discovery

When considering $2D$ maps, the basic cell is the face. Therefore, a pattern is a connected set of faces. We can then define the problem of frequent submap discovery in a similar way as [11] has defined the problem of frequent subgraph discovery: given a set of maps S and a parameter σ such that $0 < \sigma \leq 1$, the goal is to find all patterns M such that $freq(M, S) \geq \sigma \cdot |S|$, where $freq(M, S)$



Face	F1				F2		
Dart	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

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

Algorithm 1: $mSpan(S, \sigma)$

```
   Input: a set of maps $S$ and a real number $\sigma \in ]0;1]$
   Output: the set $F$ of all maps which are submaps of at least $\sigma \cdot |S|$ maps of $S$
1  $F_1 \leftarrow$ all patterns composed of 1 face and occurring in at least $\sigma \cdot |S|$ maps of $S$
2  $F \leftarrow F_1$
3  while $F_1 \neq \emptyset$ do
4      choose a pattern $f$ in $F_1$
5      $Cand \leftarrow \{f\}$
6      while $Cand \neq \emptyset$ do
7          remove a pattern $p$ from $Cand$
8          $F_p \leftarrow grow(p, F_1)$
9          $Cand \leftarrow Cand \cup F_p$
10         $F \leftarrow F \cup F_p$
       /* All frequent patterns which contain face $f$ belong to $F$ */
11     remove $f$ from $F_1$
12 return $L$
```

is the frequency of M in S, *i.e.*, the number of maps $M' \in S$ such that there is (at least) one submap isomorphism from M to M'.

A map may have an exponential number of different submaps so that mining frequent submaps has an exponential time complexity in the worst case. To reduce the set of candidate patterns to be explored, we exploit the fact that the frequency constraint is anti-monotone with respect to the submap isomorphism partial order relation: if a pattern p is not frequent, then any pattern p' such that p is subisomorphic to p' cannot be frequent.

Algorithm 1 describes our frequent submap mining algorithm, called mSpan for Map-based Substructure Pattern mining. mSpan follows the same basic principle as gSpan [17] which extracts frequent subgraphs: it constructs patterns in depth-first and exploits the frequency constraint to prune parts of the search space which do not contain frequent patterns.

More precisely, we first compute the set F_1 of all frequent patterns composed of a single face, and we initialize the set F of all frequent patterns with F_1. Then, for each face f of F_1, we build all frequent patterns which contain f plus some faces of F_1 and we add these frequent patterns to F (lines 4-10). Finally, we remove f from F_1 (line 11) in order to prevent us from re-building frequent patterns containing f in the next iterations of the while loop of lines 3-11. The set of all frequent patterns which contain f plus some faces of F_1 is built iteratively by using a set $Cand$ of frequent patterns which are candidate to be extended by sewing to them one face of F_1: at each iteration (lines 6-10), we remove a pattern p from $Cand$ (line 7) and the *grow* function computes all frequent patterns that may be built by sewing a face of F_1 to p (line 8); these frequent patterns are added to the set $Cand$ (line 9) in order to further build new patterns which contain them.

Algorithm 2: $grow(p, F_1)$

```
   Input: a frequent pattern $p$ and a set of frequent 1-face patterns $F_1$
   Output: a set $F_p$ of all frequent patterns built by adding a face of $F_1$ to $p$
1  $L_p \leftarrow \emptyset$
2  for each occurrence $o$ of the pattern $p$ in a map of $S$ do
3      for each dart $d$ which belongs to the boundary of this occurrence $o$ of $p$ do
4          if $\beta_2(d) \neq \epsilon$ so that there exists a face which is 2-sewn with $d$ then
5              let $f$ be the face 2-sewn with $d$
6              if $f \in F_1$ then
7                  let $p_f$ be the pattern obtained by 2-sewing face $f$ to dart $d$ of $o$
8                  if $p_f \notin L_p$ then add $p_f$ to $L_p$ and initialize $freq(p_f)$ to 1
9                  else update $freq(p_f)$
10 return $\{p_i \in L_p \mid freq(p_i) \geq \sigma \cdot |S|\}$
```

The *grow* function is described in algorithm 2. Given a frequent pattern p and a set of frequent 1-face patterns F_1, it returns all frequent patterns obtained by sewing one face of F_1 to p. This is done by traversing the boundary of every occurrence o of p in a map of S: for each dart d of this boundary, if the face which is 2-sewn to d belongs to F_1 then the pattern p_f obtained by 2-sewing this face to dart d of o is a candidate frequent pattern which is added to L_p if it does not already belong to it (line 8). Once all candidate patterns have been computed in L_p, we return all patterns of L_p which are frequent (line 10).

Data structure used to memorize pattern occurrences (line 2 of Algo. 2). When trying to grow pattern p by adding to it a new face, we do not compute all occurrences of p in a map of S. This information is incrementally stored: each time an occurrence of a pattern p_f is found (line 7), we keep track of it in an occurrence list $occ(p_f)$ which contains one dart for every pattern occurrence of p_f as illustrated in Fig. 3.

Traversing the boundary of a pattern occurrence (line 3 of Algo. 2). The darts which belong to the boundary of an occurrence o of a pattern p are found by performing a traversal of o, guided by the pattern p, starting in parallel from dart 1 of p and from the initial dart associated with o in the occurrence list $occ(p)$, as illustrated in Fig. 3 (see [6] for more details). This is done in linear time with respect to the number of darts of the pattern p.

Data structure used to decide if a pattern p_f belongs to L_p (line 8 of Algo. 2). Each time a new pattern p_f is found (line 7), we compute its signature and we add this signature to a signature tree. If the pattern p_f has k darts, then the space complexity of the signature of p_f is $\mathcal{O}(k)$ and the time complexity to compute the signature and to add it to the signature tree is $\mathcal{O}(k^2)$ in the worst case. Using this tree signature allows us to check if p_f already belongs to L_p (line 8) in $\mathcal{O}(k)$, whatever the size of L_p is (see [8] for more details).

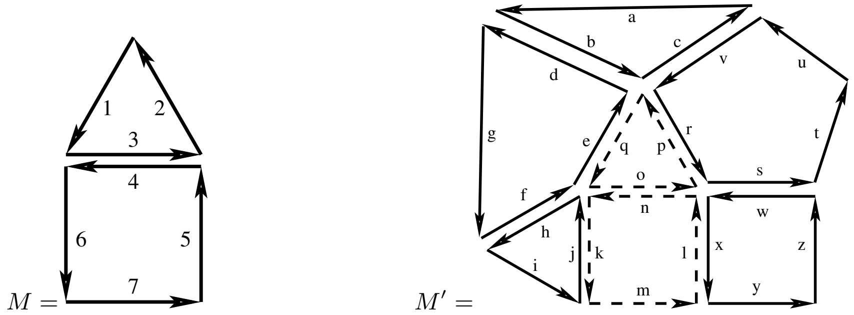


Fig. 3. Example of pattern occurrence list. Pattern M occurs 5 times in map M'. For each occurrence o, we memorize the dart of M' which corresponds to dart 1 of pattern M. Hence, the occurrence list associated with pattern M in map M' is $occ(M) =< q, a, j, p, i >$. To find the boundary of an occurrence of M, we search for the darts of M' which correspond to the 2-free darts of M (*i.e.*, 1, 6, 7, 5, 2). For example, the boundary of the occurrence of M which starts at dart q contains darts q, k, m, l, p.

4 Generalization to *nD* combinatorial maps

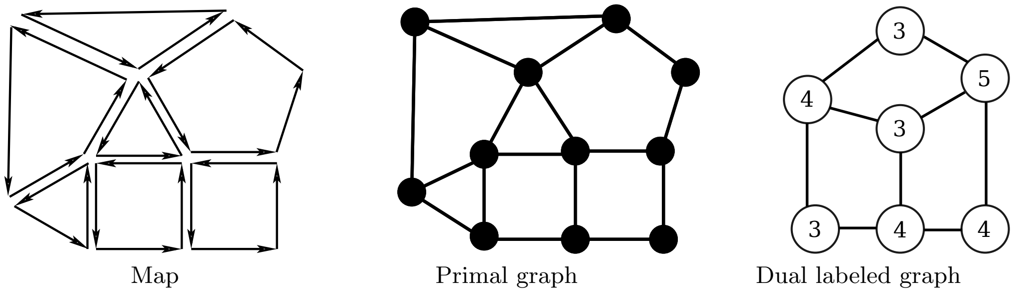
For sake of simplicity, we have described our frequent submap mining algorithm for $2D$ combinatorial maps. However, it can be extended to nD maps in a very straightforward way. Actually, we have implemented it for the nD case and we report experimental results on 2D and 3D maps in the next section.

If $2D$ maps are described by two functions β_1 and β_2 which respectively describe adjacency relations between edges and faces, nD maps are described by n functions, β_1 to β_n, such that each β_i function describes adjacency relations between cells of dimension i, called i-cells (1-cells are edges, 2-cells faces, 3-cells volumes, ...). We have extended submap isomorphism to nD maps in [6].

In nD, mined patterns are connected n-cells (*i.e.*, connected faces in 2D, connected volumes in 3D, ...). Algorithms 1 and 2 are extended to the nD case by replacing faces with n-cells: we first search for all frequent patterns composed of one n-cell and, for each of these patterns, we iteratively compute all frequent patterns which contain it. The *grow* function builds new frequent patterns by n-sewing a frequent n-cell with a frequent pattern.

5 Experimental results

In this section, we report preliminary experimental results on synthetic databases of randomly generated 2D and 3D maps. All experiments have been performed on an Intel Core 2 Duo with 4GB RAM. Using synthetic databases allows us to evaluate scale-up properties when decreasing the frequency threshold σ, and when increasing the size of the maps, *i.e.*, the number of faces in 2D and the number of volumes in 3D.



Map

Primal graph

Dual labeled graph

Fig. 4. Example of primal graph and dual labeled graph associated with a map.

Considered datasets. We have generated different databases. Each database $D(n, k)$ contains 1000 connected maps such that $n \in \{2, 3\}$ corresponds to the dimension of the map, and k to the number of n-cells (faces for $n = 2$ and volumes for $n = 3$). Maps are randomly generated in such a way that faces (resp. volumes) have degrees varying between 3 and 10 (resp. 4 and 10).

Maps vs graphs. We compare mSpan with gSpan[1], which is a state-of-the-art algorithm for extracting frequent connected subgraphs from a database of graphs [17]. Given a 2D map, we can generate a primal graph in a very straightforward way (see Fig. 4). However, mining the primal graph is not really meaningful and the extracted patterns cannot be compared with those extracted by mSpan. Indeed, mSpan extracts connected sets of faces whereas patterns extracted by gSpan are connected subgraphs which may not correspond to connected sets of faces at all (*e.g.*, trees). For a fair comparison, we consider the dual graph which associates a vertex with every face of the 2D map and which connects two vertices iff the corresponding faces in the map are adjacent. We also label each vertex of the dual graph with the degree of the corresponding face (*i.e.*, its number of edges). Patterns extracted by gSpan from the labeled dual graph are connected subgraphs and, therefore, correspond to connected sets of faces in the 2D map. Labels associated with vertices allow gSpan to discriminate faces which have different degrees and greatly improve performances of gSpan. However, gSpan does not consider the topology of the graph (*i.e.*, the order in which faces are encountered when turning around one face) so that two different submaps may correspond to the same subgraph in the dual labeled graph, as illustrated in Fig. 5. Therefore, mSpan and gSpan do not extract the same frequent patterns.

For 3D maps, we also generate dual labeled graphs: we associate a vertex with every volume of the 3D map; we connect two vertices iff the corresponding volumes are adjacent; and we label each vertex with the degree of the corresponding volume. This way, connected subgraphs of dual labeled graphs correspond to connected sets of volumes. However, like in 2D, different connected sets of volumes may correspond to a same connected graph.

[1] implementation found in http://www.cs.ucsb.edu/∼xyan/software/gSpan.htm

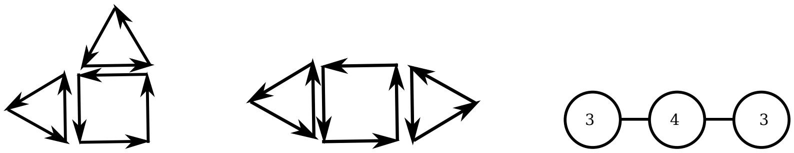


Fig. 5. Two different submaps which correspond to the same dual labeled graph.

Note that labeled dual graphs are much smaller than the corresponding maps: a 2D (resp. 3D) map which has 350 faces (resp. 80 volumes) has 1800 (resp. 1200) darts or so, whereas the corresponding dual graph has 800 (resp. 160) edges or so and 350 (resp. 80) vertices.

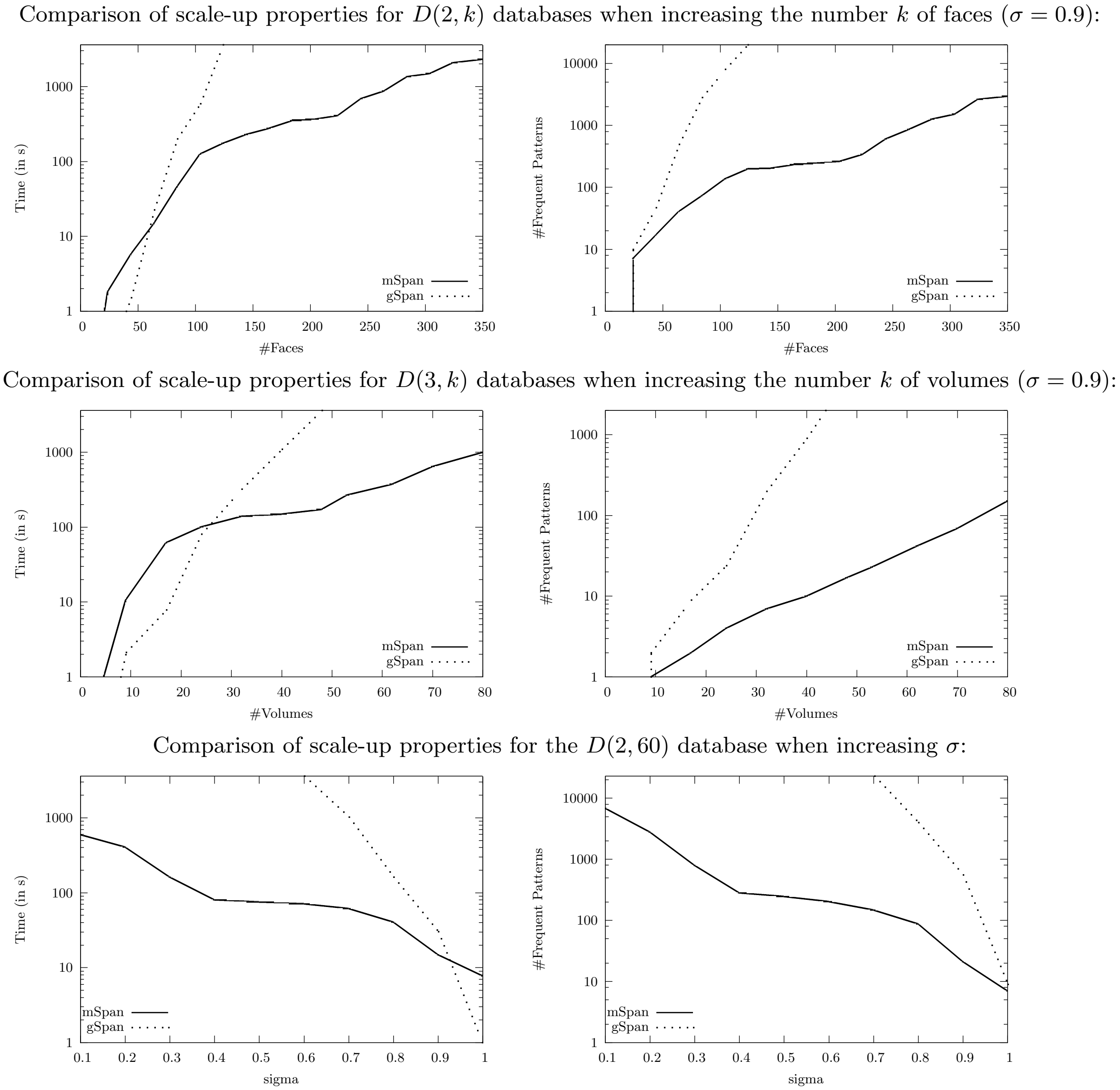
Scale-up properties when increasing map sizes. Top and middle curves of Fig. 6 display results of mSpan and gSpan on $D(n, k)$ databases with $\sigma = 0.9$ when increasing the number of faces k from 4 to 350 for $n = 2$, and when increasing the number of volumes k from 2 to 80 for $n = 3$. Each run has been limited to 3600 seconds of CPU time. mSpan is able to extract all frequent patterns within this time limit, even for the largest values of k. gSpan is faster than mSpan when $k < 50$ in 2D, and when $k < 30$ in 3D. However, for larger values of k it becomes slower, and it is not able to compute all frequent patterns within the CPU time limit of 3600 seconds when $k > 120$ in 2D and when $k > 50$ in 3D. Actually, gSpan extracts much more frequent patterns than mSpan, and the greater k, the larger the difference. This comes from the fact that graphs do not model the topology so that different map patterns (which may not be frequent) correspond to the same graph pattern (which may become frequent).

Scale-up properties when increasing σ. Bottom curves of Fig. 6 displays results of mSpan and gSpan on the $D(2, 60)$ database when increasing the frequency threshold from 0.1 to 1. It shows us that gSpan is faster than mSpan when $\sigma > 0.9$, but for smaller values of σ, mSpan becomes faster and gSpan is not able to compute all frequent patterns within the CPU time limit of 3600 seconds when $\sigma < 0.7$. Actually, the number of extracted patterns grows much quicker for gSpan than for mSpan when decreasing the frequency threshold σ. We have performed similar experiments on 3D maps, and observed very similar results.

6 Conclusion

We have introduced an algorithm called mSpan for extracting frequent patterns from combinatorial maps. This algorithm uses efficient polynomial time procedures for deciding of submap isomorphism [5], and for searching for isomorphic occurrences of a given map in the signature of a base of maps [8].

Combinatorial maps 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. We have shown that we can use dual labeled



Comparison of scale-up properties for $D(2,k)$ databases when increasing the number k of faces ($\sigma = 0.9$):

Comparison of scale-up properties for $D(3,k)$ databases when increasing the number k of volumes ($\sigma = 0.9$):

Comparison of scale-up properties for the $D(2,60)$ database when increasing σ:

Fig. 6. Comparison of mSpan (bold lines) and gSpan (dashed lines) scale-up properties: curves on the left (resp. right) plot the evolution of CPU-time in seconds (resp. number of extracted patterns).

graphs to model adjacency relationships between cells, but these graphs do not model the topology of these cells (*i.e.*, the order in which they are encountered when turning around a given cell). Therefore, different map patterns (which may not be frequent) may be modeled by a same dual labeled graph (which may become frequent) so that a graph mining algorithm extracts much more patterns. Of course, the relevancy of extracted patterns depends on the application. We have already applied mSpan to an aperiodic tiling application, the goal of which is to find the largest pattern occurring frequently in a given aperiodic tiling. Clearly, on this kind of application, the topology is of uppermost importance and patterns extracted from dual labeled graphs are not relevant.

Further work will mainly concern the application of mSpan to image recognition and classification tasks. Indeed, 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). We more particularly propose to use mSpan to extract patterns which occur frequently in maps modeling images belonging to a same class, thus allowing us to characterize classes by means of these patterns.

References

1. J.-P. Braquelaire and L. Brun. Image segmentation with topological maps and inter-pixel representation. 9(1):62–79, march 1998.
2. E. Brisson. Representing geometric structures in d dimensions: topology and order. In *SCG*, pages 218–227, Saarbrücken, Germany, 1989.
3. R. Cori. Un code pour les graphes planaires et ses applications. In *Astérisque*, volume 27. Soc. Math. de France, Paris, France, 1975.
4. G. Damiand. Topological model for 3d image representation: Definition and incremental extraction algorithm. *CVIU*, 109(3):260–289, March 2008.
5. 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)*, LNCS, pages 102–112. Springer, May 2009.
6. G. Damiand, C. Solnon, C. De La Higuera, J.-C. Janodet, and E. Samuel. Polynomial Algorithms for Subisomorphism of nD Open Combinatorial Maps. Technical Report RR-LIRIS-2009-044, December 2009.
7. J. Edmonds. A combinatorial representation for polyhedral surfaces. *Notices of the American Mathematical Society*, 7, 1960.
8. S. Gosselin, G. Damiand, and C. Solnon. Efficient search of combinatorial maps using signatures. *Theoretical Computer Science*, In Press, Corrected Proof:–, 2010.
9. T. Horvath, J. Ramon, and S. Wrobel. Frequent subgraph mining in outerplanar graphs. In *KDD 2006*, pages 197–206, 2006.
10. A. Jacques. Constellations et graphes topologiques. In *Combinatorial Theory and Applications*, volume 2, pages 657–673, 1970.
11. M. Kuramochi and G. Karypis. Frequent subgraph discovery. *Data Mining, IEEE International Conference on*, 0:313, 2001.
12. P. Lienhardt. Subdivision of n-dimensional spaces and n-dimensional generalized maps. In *SCG*, pages 228–236, Saarbrücken, Germany, 1989.
13. P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82, 1991.
14. P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *IJCGA*, 4(3):275–324, 1994.
15. M. Poudret, A. Arnould, Y. Bertrand, and P. Lienhardt. Cartes combinatoires ouvertes. Research Notes 2007-1, Laboratoire SIC E.A. 4103, October 2007.
16. W.T. Tutte. A census of planar maps. *Canad. J. Math.*, 15:249–271, 1963.
17. X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 721–, Washington, DC, USA, 2002. IEEE Computer Society.