Computing the Overlaps of Two Maps

Jean-Christophe Janodet[1] and Colin de la Higuera[2⋆]

[1] IBISC Lab, University of Evry, 23 Bd de France, F-91037 Evry, France,
janodet@ibisc.univ-evry.fr
[2] LINA Lab, University of Nantes, 2 R de la Houssinière, F-44322 Nantes, France,
cdlh@univ-nantes.fr

Abstract. Two combinatorial maps M_1 and M_2 overlap if they share a sub-map, called an overlapping pattern, which can be extended without conflicting neither with M_1 nor with M_2. Isomorphism and subisomorphism are two particular cases of map overlaps which have been studied in the literature. In this paper, we show that finding the largest connected overlap between two combinatorial maps is tractable in polynomial time. On the other hand, without the connectivity constraint, the problem is $\mathcal{NP}$-hard. To obtain the positive results we exploit the properties of a product map.
Keywords. 2D semi-open combinatorial maps, overlaps, overlapping patterns, product map.

1 Introduction

2D-combinatorial maps are algebraic structures which allow to describe and work with plane graphs, that is, embeddings of planar graphs. Using such structures has allowed to establish several algorithmic properties. *E.g.*, it is possible to decide whether two drawings of planar graphs, or two maps, are isomorphic or not in quadratic time [1–3]. Moreover, deciding whether a *pattern* (*i.e.*, a drawing made of a connected subset of faces) appears in a map is also tractable in quadratic time; this property is interesting since determining whether a connected graph is a sub-graph of a planar graph is known to be an $\mathcal{NP}$-complete problem [4, 5, 3]. So focusing on a particular drawing of a planar graph (among possibly an exponential number of possibilities) is very helpful from an algorithmic point of view. It has also been shown that searching for a *disconnected*-pattern, built from several unconnected patterns, is an $\mathcal{NP}$-hard problem [3].

A related problem consists in finding large common patterns in two maps. In order to get a common pattern, one must eliminate subsets of faces from both maps and obtain the same pattern up to an isomorphism. *E.g.*, in Fig. 1, map (c) is the maximum common pattern of maps (a) and (b); the eliminated faces are shown with dotted lines. It has been proved in [6] that computing such large common patterns is an $\mathcal{NP}$-hard problem, even when patterns are connected.

In this paper, we constrain the definition of common patterns: we now require the pattern to be *extendable* to both maps when adding *independent* groups of

⋆ The second author wishes to acknowledge the support of University of Kyoto

faces. Independence means that if any new face is added in a connected way to the common pattern, then the result ceases to be a pattern of one of the two maps; moreover, if any face is added to the pattern in order to get one of the maps, then no face can be added at the same place in the pattern to get the second map. See Fig. 1 (d) for an example. Every common pattern that has this property results of an overlap between both maps, where pairs of faces of both maps were merged together: such an overlap defines an *overlapping pattern*.

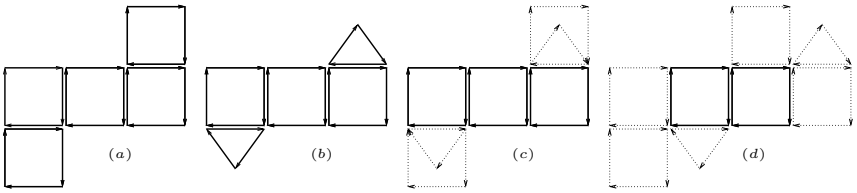


Fig. 1. Maps (a) and (b) have map (c) as maximal common pattern, and map (d) as maximal overlapping pattern. Dotted lines are construction features, and do not belong to the maps.

Notice that the overlapping patterns can be smaller than the maximal common patterns. On the other hand, while the latter are not tractable in polynomial time [6], we show in this paper that computing any connected overlap is tractable in linear time, and enumerating all of the connected overlap is a quadratic-time problem. It follows that finding the largest connected overlap can also be done in polynomial time and space. In contrast, we prove that finding large possibly disconnected overlaps is $\mathcal{NP}$-hard.

Finally, in terms of applications, every maximal overlapping pattern O yields a distance defined by: $d(M_1, M_2) = \text{size}(M_1) + \text{size}(M_2) - 2.\text{size}(O)$. If one can find any maximal overlap in polynomial time, distance d is efficiently computable, and may then be used as an efficient rough approximation to tighter $\mathcal{NP}$-hard graph edit distances.

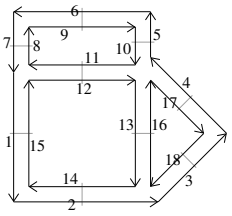
In Sect. 2, we recall the definitions of full and semi-open maps. The overlaps are introduced in Sect. 3, and the overlapping patterns in Sect. 4. The polynomial problems, related to the existence and the enumeration of the connected overlaps, are investigated in Sect. 5. The correctness of both algorithms is proved in Appendix, due to the lack of space. The case of disconnected overlaps is discussed in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Combinatorial Maps

Definition 1. *Let D be a finite set of* darts. *A* 2D full combinatorial map *is a triple $M = (D, \alpha, \beta)$ such that (1) $\alpha : D \to D$ is a 1-to-1 mapping (*i.e.*, a* permutation *over D), and (2) $\beta : D \to D$ is a 1-to-1 mapping such that for all*

$d \in D$, $\beta(\beta(d)) = d$ *(*i.e.*, an* involution *over D). Two darts d and d' such that $d' = \alpha(d)$ or $d' = \beta(d)$ are respectively said α-sewn or β-sewn.*

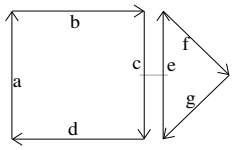
Figure 2 shows an example of a full map. Notice that, given a dart d, the *face* which is incident to d is obtained by iterating α. *E.g.*, in Fig. 2, the face incident to dart 12 is described with set $\{12, 13, 14, 15\}$ and we have $\alpha(12) = 13$, $\alpha(13) = 14$, $\alpha(14) = 15$ and $\alpha(15) = 12$. Similarly, the *edges* and the *vertices* of a full map are respectively introduced as the orbits of permutations β and $\beta \circ \alpha$.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
α	2	3	4	5	6	7	1	9	10	11	8	13	14	15	12	17	18	16
β	15	14	18	17	10	9	8	7	6	5	12	11	16	2	1	13	4	3

Fig. 2. An example of full map. The darts are represented by numbered black segments. Two α-sewn darts are drawn consecutively, and two β-sewn darts are drawn concurrently and in reverse orientation, with little gray segment between them.

All the faces of a full map are defined, which is irrelevant if this map is expected to overlap with others; some faces must be invisible, so function β must be partially defined. This leads us to introduce *semi-open maps*, simply called *maps* throughout the rest of this paper [7]. The idea is to implicitly add a new element ε to the set of darts, and allow any dart to be β-linked with ε whenever such a dart has no adjacent face. Figure 3 shows an example.



	a	b	c	d	e	f	g
α	b	c	d	a	f	g	e
β	ε	ε	e	ε	c	ε	ε

Fig. 3. An example of (semi-open) map. Darts a, b, d, f and g are not β-sewn.

Definition 2. *Let D be a finite set of darts and $\varepsilon \notin D$ a fresh implicit dart. A* semi-open map*, or simply* map*, is a triple $M = (D, \alpha, \beta)$ such that*

- *$\alpha : D \cup \{\varepsilon\} \to D \cup \{\varepsilon\}$ is a 1-to-1 mapping with $\alpha(\varepsilon) = \varepsilon$;*

– $\beta : D \cup \{\varepsilon\} \to D \cup \{\varepsilon\}$ *is a* partial involution *[8, Definition 4], that is, a mapping such that (1)* $\beta(\varepsilon) = \varepsilon$ *and (2) for all* $d \in D$*, if* $\beta(d) \neq \varepsilon$*, then* $\beta(\beta(d)) = d$*.*

The complexity of the problems that we address on the maps is often related to connectivity. For instance, searching for a connected pattern (subset of contiguous faces) in a map is a quadratic problem, whereas searching for a disconnected pattern (subset of independent patterns) is $\mathcal{NP}$-complete [8].

Definition 3. *Let* $M = (D, \alpha, \beta)$ *be a semi-open map. Any subset* $U \subseteq D$ *is* connected *in map* M *if for all* $d, d' \in U$*, there exists a sequence* $d_0, d_1, \ldots d_n \in U$ *such that (1)* $d_0 = d$ *and (2)* $d_n = d'$ *and (3) for all* $0 \leq k < n$*, we have* $d_{k+1} = \gamma_k(d_k)$ *with* $\gamma_k = \alpha$ *or* $\gamma_k = \beta$*. We say that map* M *is* connected *if set* D *itself is connected in map* M*.*

3 The Overlaps of Two Maps

An overlap is a maximal one-to-one *matching*. Let $M_1 = \langle D_1, \alpha_1, \beta_1 \rangle$ and $M_2 = \langle D_2, \alpha_2, \beta_2 \rangle$ be two fixed semi-open maps.

Definition 4. *A (one-to-one)* matching *is a function* $h : U_1 \to U_2$ *such that*

1. $U_1 \subseteq D_1$ *and* $U_2 \subseteq D_2$*,*
2. h *is bijective,*
3. *for all* $d_1 \in U_1$ *and* $d_2 = h(d_1)$*,*
   - $\alpha_1(d_1) \in U_1$ *if and only if* $\alpha_2(d_2) \in U_2$*, and* $h(\alpha_1(d_1)) = \alpha_2(d_2)$*;*
   - $\beta_1(d_1) \in U_1$ *if and only if* $\beta_2(d_2) \in U_2$*, and* $h(\beta_1(d_1)) = \beta_2(d_2)$*.*

An example is given in Fig. 4. Notice that $\varepsilon \notin U_1$ and $\varepsilon \notin U_2$ (since $\varepsilon \notin D_1$ and $\varepsilon \notin D_2$); nevertheless, for convenience reasons, we shall implicitly suppose that $h(\varepsilon) = \varepsilon$.

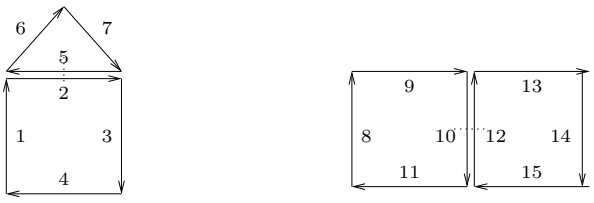


Fig. 4. Consider the maps M_1 and M_2 above. Mapping $h = \{1 \mapsto 8, 2 \mapsto 9, 6 \mapsto 13\}$ is a matching, whereas mapping $h' = \{1 \mapsto 8, 2 \mapsto 9, 6 \mapsto 10\}$ is not; indeed, we have $\alpha_2(9) = 10$, but $\alpha_1(2) \neq 6$. Any matching must preserve the seams of *both* maps.

Definition 5. *Let $h : U_1 \to U_2$ be a one-to-one matching as in Definition 4. We say that h is an* overlap *if:*

1. *for all $d_1 \in U_1$, we have $\alpha_1(d_1) \in U_1$, and*
2. *for all $d_1 \in U_1$ and $d_2 = h(d_1)$, if $\beta_1(d_1) \neq \varepsilon$ and $\beta_2(d_2) \neq \varepsilon$, then $\beta_1(d_1) \in U_1$ and $\beta_2(d_2) \in U_2$.*

We say that h is a connected overlap *if subset U_1 is connected in map M_1. We say that h is a* disconnected overlap *otherwise.*

The first condition above implies that if a dart d_1 of M_1 matches a dart d_2 of M_2, then the whole face incident to d_1 must match the whole face incident to d_2. The second condition means that if darts d_1 and d_2 match together and both have adjacent faces, then opposite β-sewn darts, and thus adjacent faces must also match together. In consequence, each pieces of both maps M_1 and M_2 must be as large as possible, that is, the matching must be "maximal". See Fig. 5 and 6 for examples.

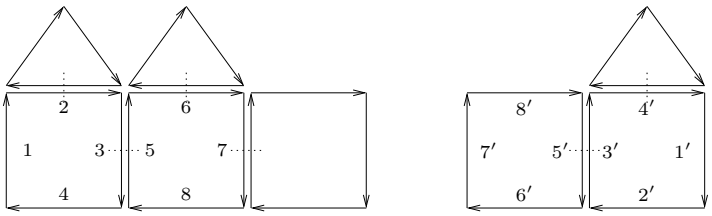


Fig. 5. An example of connected overlap $h = \{1 \mapsto 1', 2 \mapsto 2', \ldots 8 \mapsto 8'\}$. Notice that no overlap can be built with darts 5 and 4' matched together.

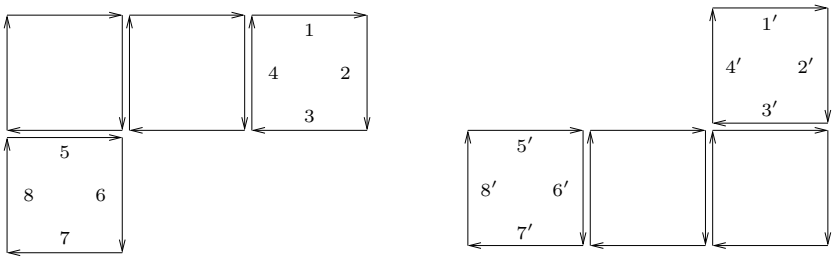


Fig. 6. An example of disconnected overlap $h = \{1 \mapsto 1', 2 \mapsto 2', \ldots 8 \mapsto 8'\}$.

4 Properties of the Overlapping Patterns

Given two maps M_1 and M_2 and an overlap $h : U_1 \to U_2$, the elimination of all the darts, but those from sets U_1 and U_2, in maps M_1 and M_2 respectively, defines two isomorphic sub-maps of M_1 and M_2. In other words, an overlap defines an *overlapping common pattern*:

Definition 6. *Map $P = \langle D, \alpha, \beta \rangle$ is a* pattern *of map $M = \langle D', \alpha', \beta' \rangle$ if there exists a one-to-one matching $\varphi : D \to V$ (with $V \subseteq D'$). Moreover, any map P is a* common pattern *of maps M_1 and M_2 if P is a pattern of both M_1 and M_2.*

As direct consequences of Definitions 4 and 5, the overlapping patterns have the following properties:

- Every overlapping pattern of M_1 and M_2 is a pattern of both M_1 and M_2;
- Given every pair of darts (d_1,d_2), there exists at most one connected overlap, and thus at most one connected overlapping pattern, in which d_1 and d_2 are matched together;
- An overlapping pattern is *maximal* in the following sense: if we add to this pattern something and the result is a semi-open map, then this map is no longer a sub-map of M_1 or of M_2.

Concerning computational issues, we have provided in [2] an algorithm in $\mathcal{O}(|D_1| \times |D_2|)$ time to decide whether two (possibly non-connected) maps were isomorphic or not. An extension of this algorithm can be used to prove that a connected map is a pattern of another map. In both cases, due to the technical necessity for the matching h to commute with bijections α and β, there are no more than $|D_1|$ possible matching functions, and the algorithms actually enumerate all of them in $\mathcal{O}(|D_1| \times |D_2|)$ time. Concerning the second problem, the fact that the first map is connected is crucial, since this problem is $\mathcal{NP}$-complete for disconnected patterns [6].

With respect to finding large common connected patterns, it has unfortunately been shown in [6] that this problem is $\mathcal{NP}$-hard too. But looking for large overlapping patterns is simpler: whereas in general the pattern can be placed anywhere in the maps, we now insist on the pattern to somehow be placed on the border of both maps, corresponding to the part where they overlap.

In order to illustrate and discuss this point, we turn to strings. In terms of strings, common patterns would be common sub-strings of two given strings: given $u, v \in \Sigma^*$, a pattern is a string $w \in \Sigma^+$ such that $u = lwr$ and $v = l'wr'$ for some $l, l', r, r' \in \Sigma^*$.

On the other hand, an overlap defines a string w as above, such that $u = lwr$ and $v = l'wr'$ but with the added condition that $l = \epsilon$ or $l' = \epsilon$ on one hand, $r = \epsilon$ or $r' = \epsilon$ on the other. This means that exactly 4 cases are possible:

1. $u = lw$ and $v = wr'$, w is a suffix of u and a prefix of v,
2. $u = wr'$ and $v = l'w$, w is a prefix of u and a suffix of v,
3. $u = lwr$ and $v = w$, v is a sub-string of u,
4. $u = w$ and $v = l'wr'$, u is a sub-string of v.

So the problem is simpler.

5 Finding Connected Overlaps Efficiently

In this section, we show that it is possible to efficiently find the largest connected overlap of two maps.

5.1 A Linear Time and Space Algorithm to Check whether a Connected Overlap Exists

Let $M_1 = \langle D, \alpha_1, \beta_1 \rangle$ and $M_2 = \langle D_2, \alpha_2, \beta_2 \rangle$ be two semi-open maps. The first problem we tackle consists in determining whether two darts $d_1 \in D_1$ and $d_2 \in D_2$ can match together in an overlap. This is the purpose of Algorithm 1.

This procedure performs a parallel traversal of both maps M_1 and M_2, starting from the darts d_1 and d_2, which are grouped together into a couple (d_1, d_2). The procedure uses the α- and β-functions of both maps to discover new couples of darts from the couples that have been discovered so far. It more precisely builds a candidate overlap h such that $h[d_1] = d_2$. So initially, $h[d_1]$ is set to d_2, whereas $h[d]$ are set to *nil* for all other darts d.

Each time a couple (a, a') is discovered from another couple (d, d') by using the α-functions, we check whether both darts a and a' have ever been met. If either a or a' have already been visited through the traversal, we carefully check basic conditions which ensure us to get a valid matching h at the end of the algorithm. Otherwise, $h[a]$ is set to a' and the couple (a, a') is further used to discover new darts. With respect to β-functions, the same principle holds, but more cases of failure can occur, as the darts d or d' may have no adjacent face.

Note that Algorithm 1 returns a single Boolean value. Nevertheless, one can easily modify the procedure and get the connected overlap h as a certificate, in case of success. The following theorem (which is proved in Appendix) claims that this algorithm is correct.

Theorem 1. *Algorithm 1 is correct, that is:*

- *If* checkConnectedOverlap(M_1, M_2, d_1, d_2) *returns* `true`*, then the array h that is built by the procedure encodes a connected overlap $h : U_1 \to U_2$ such that $h(d_1) = d_2$.*
- *If* checkConnectedOverlap(M_1, M_2, d_1, d_2) *returns* `false`*, then no overlap $h : U_1 \to U_2$ exists such that $h(d_1) = d_2$.*

With respect to the complexity of the algorithm, we have:

Theorem 2. *Algo. 1 runs in $\mathcal{O}(\min(|D_1|, |D_2|))$ time and $\mathcal{O}(\max(|D_1|, |D_2|))$ space.*

Proof. Suppose that $|D_1| \leq |D_2|$ without loss of generality. Then the while loop of Algorithm 1 is iterated at most $|D_1|$ times. Indeed, (1) at each iteration, exactly one dart $d \in D_1$ is removed from the stack S, within a couple (d, x) for some $x \in D_2$, and (2) each dart $d \in D_1$ enters S at most once, within a couple (d, x) for any $x \in D_2 : d$ enters S only if $h[d] = nil$, and before entering S, $h[d]$ is set to dart x. So the algorithm runs in $\mathcal{O}(\min(|D_1|, |D_2|))$ time. As for space issues, notice that the arrays h and g respectively have $|D_1|$ and $|D_2|$ entries, while stack S never contains more than $\min(|D_1|, |D_2|)$ couples of darts. □

Algorithm 1: checkConnectedOverlap(M_1, M_2, d_1, d_2)

Input: Two semi-open maps $M_1 = \langle D_1, \alpha_1, \beta_1 \rangle$ and $M_2 = \langle D_2, \alpha_2, \beta_2 \rangle$, and an initial couple of darts $(d_1, d_2) \in D_1 \times D_2$
Output: `true` if a connected overlap h exists such that $h(d_1) = d_2$, `false` otherwise
Variables: Two arrays $h : D_1 \to D_2$ and $g : D_2 \to D_1$ (where $g = h^{-1}$) both initialized with nil, and a stack S which is initially empty

```
1  h[d1] ← d2 ; g[d2] ← d1 ; push (d1, d2) in S ;
2  while S is not empty do
3      pop a couple of darts (d, d') from S ;
4      a ← α1(d) ; a' ← α2(d') ;
5      if h[a] = nil and g[a'] = nil then
6          h[a] ← a' ; g[a'] ← a ; push (a, a') in S ;
7      else if h[a] ≠ a' or g[a'] ≠ a then
8          return false ;
9      b ← β1(d) ; b' ← β2(d') ;
10     if b ≠ ε and b' ≠ ε then
11         if h[b] = nil and g[b'] = nil then
12             h[b] ← b' ; g[b'] ← b ; push (b, b') in S ;
13         else if h[b] ≠ b' or g[b'] ≠ b then
14             return false ;
15     else if b = ε and b' ≠ ε and g[b'] ≠ nil then
16         return false ;
17     else if b ≠ ε and b' = ε and h[b] ≠ nil then
18         return false ;
19 return true ; // Array h may be returned as a certificate
```

Notice that Algorithm 1 exploits the same key idea as the algorithms developed in [2] to solve the map isomorphism and sub-isomorphism problems. We nevertheless improve them as the failure cases are detected during the traversal of the maps, and no further verification stage is needed after the traversal.

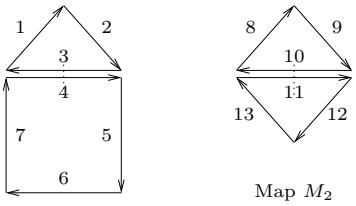
5.2 A Quadratic Time and Space Algorithm to Get all the Connected Overlaps

Let $M_1 = \langle D, \alpha_1, \beta_1 \rangle$ and $M_2 = \langle D_2, \alpha_2, \beta_2 \rangle$ be two semi-open maps. In Sect. 5.1, we have given a procedure that checks whether two darts $d_1 \in D_1$ and $d_2 \in D_2$ can match together in a connected overlap. To achieve this goal, Algorithm 1 performs a parallel traversal of both maps M_1 and M_2, starting from couple (d_1, d_2), and using the α- and β-functions of both maps to investigate new couples of darts from the couples that have been visited so far. Clearly, this procedure may visit any couple of $D_1 \times D_2$. So we use this set to define a new map, denoted $M_1 \otimes M_2$, and call the *product map* of maps M_1 and M_2:

Definition 7. *The* product *of two maps* M_1 *and* M_2 *is* $M_1 \otimes M_2 = \langle D_1 \times D_2, \alpha, \beta \rangle$ *where, for all* $(d_1, d_2) \in D_1 \times D_2$*:*

$$\alpha(d_1, d_2) = (\alpha_1(d_1), \alpha_2(d_2)), \text{ and}$$
$$\beta(d_1, d_2) = \begin{cases} (\beta_1(d_1), \beta_2(d_2)) & \text{if } \beta_1(d_1) \neq \varepsilon \text{ and } \beta_2(d_2) \neq \varepsilon, \\ \varepsilon & \text{otherwise.} \end{cases}$$

For instance, consider the maps of Fig. 7, respectively made of 7 and 6 darts. The product map $M_1 \otimes M_2$, which has 42 darts, contains 6 connected components. Clearly, two kinds of connected components appear. The four small components will be called *real*, and the two large ones, *imaginary*:



Map M_1 Map M_2

	α	β
$(1,8)$	$(2,9)$	ε
$(2,9)$	$(3,10)$	ε
$(3,10)$	$(1,8)$	$(4,11)$
$(4,11)$	$(5,12)$	$(3,10)$
$(5,12)$	$(6,13)$	ε
$(6,13)$	$(7,11)$	ε
$(7,11)$	$(4,12)$	ε
$(4,12)$	$(5,13)$	ε
$(5,13)$	$(6,11)$	ε
$(6,11)$	$(7,12)$	ε
$(7,12)$	$(4,13)$	ε
$(4,13)$	$(5,11)$	ε
$(5,11)$	$(6,12)$	ε
$(6,12)$	$(7,13)$	ε
$(7,13)$	$(4,11)$	ε

	α	β
$(1,12)$	$(2,13)$	ε
$(2,13)$	$(3,11)$	ε
$(3,11)$	$(1,12)$	$(4,10)$
$(4,10)$	$(5,8)$	$(3,11)$
$(5,8)$	$(6,9)$	ε
$(6,9)$	$(7,10)$	ε
$(7,10)$	$(4,8)$	ε
$(4,8)$	$(5,9)$	ε
$(5,9)$	$(6,10)$	ε
$(6,10)$	$(7,8)$	ε
$(7,8)$	$(4,9)$	ε
$(4,9)$	$(5,10)$	ε
$(5,10)$	$(6,8)$	ε
$(6,8)$	$(7,9)$	ε
$(7,9)$	$(4,10)$	ε

	α	β
$(1,9)$	$(2,10)$	ε
$(2,10)$	$(3,8)$	ε
$(3,8)$	$(1,9)$	ε

	α	β
$(1,10)$	$(2,8)$	ε
$(2,8)$	$(3,9)$	ε
$(3,9)$	$(1,10)$	ε

	α	β
$(1,11)$	$(2,12)$	ε
$(2,12)$	$(3,13)$	ε
$(3,13)$	$(1,11)$	ε

	α	β
$(1,13)$	$(2,11)$	ε
$(2,11)$	$(3,12)$	ε
$(3,12)$	$(1,13)$	ε

Fig. 7. Given the maps M_1 and M_2, we build the product map $M_1 \otimes M_2$ and display its connected components. The two big components are imaginary, and the four small ones are real. Couple $(1,8)$ belongs to an imaginary component, so following Theorem 3, no overlap exists such that darts 1 and 8 match. Conversely, couple $(1,9)$ is in a real component and mapping $\{1 \mapsto 9, 2 \mapsto 10, 3 \mapsto 8\}$ is a connected overlap.

Definition 8. *A connected component $C = \langle D, \alpha, \beta \rangle$ of product map $M_1 \otimes M_2$ is said* real *if for all $(d_1, d_2), (d'_1, d'_2) \in D$,*

1. *$d_1 = d'_1$ iff $d_2 = d'_2$, and*
2. *$\beta_1(d_1) = d'_1$ iff $\beta_2(d_2) = d'_2$.*

A connected component which is not real is said imaginary.

Remark 1. The reader may wonder why no condition addressing the α-functions is given in Definition 8. Actually, a consequence of Condition 1 is that for all (d_1, d_2), $(d'_1, d'_2) \in D$, we have $\alpha_1(d_1) = d'_1$ iff $\alpha_2(d_2) = d'_2$. Indeed, let $(d_1, d_2) \in D$ and suppose that $(\alpha_1(d_1), d'_2) \in D$; as component C is connected, we have $\alpha(d_1, d_2) = (\alpha_1(d_1), \alpha_2(d_2)) \in D$; so using Condition 1, we deduce that $d'_2 = \alpha_2(d_2)$. Such a proof cannot be given for Condition 2, due to the fact that d_1 or d_2 can β-free.

Connected overlaps of maps M_1 and M_2 on the one hand, and real connected components of product map $M_1 \otimes M_2$ on the other hand, are strongly related. Indeed, we get the following result (which is proved in Appendix):

Theorem 3. *Let M_1 and M_2 be two semi-open maps.*

- *For each connected overlap $h : U_1 \to U_2$, there exists a real connected component C of product map $M_1 \otimes M_2$ whose set of darts is $\{(d, h(d)) : d \in U_1\}$;*
- *Conversely, for every real connected component $C = \langle D, \alpha, \beta \rangle$ of product map $M_1 \otimes M_2$, set D is the graph of a connected overlap of M_1 and M_2, that is to say, if we fix $h(d_1) = d_2$ for all $(d_1, d_2) \in D$, then h is a connected overlap.*

As a consequence of Theorem 3, we get an efficient algorithm to enumerate all the connected overlaps (see Algorithm 2).

Algorithm 2: getAllConnectedOverlaps(M_1, M_2)

Input: Two semi-open maps $M_1 = \langle D, \alpha_1, \beta_1 \rangle$ and $M_2 = \langle D_2, \alpha_2, \beta_2 \rangle$)
Output: All the connected overlaps of M_1 and M_2

```
1 Compute product map M1 ⊗ M2 ; // see Definition 7
2 Select all the real connected components ; // see Definition 8
3 Return all the connected overlaps; // see Theorem 3
```

Theorem 4. *Algorithm 2 is correct and runs in $\mathcal{O}(|D_1| \cdot |D_2|)$ time and space.*

Proof. The correctness follows from Theorem 3. As for the complexity, the product map $M_1 \otimes M_2$ has $|D_1| \cdot |D_2|$ darts. Obviously, the computation of functions α and β, and the computation of the connected components, and the selection of the real connected components, are in linear time and space with respect to $|D_1| \cdot |D_2|$. □

5.3 Finding the Largest Overlap of Two Maps

It is straightforward to use the results from the previous algorithm to return the largest overlap, provided this one is connected.

Furthermore a direct procedure exists to find the largest possibly disconnected overlap: consider a graph whose nodes are the connected overlaps between maps M_1 and M_2, each with a weight indicating how many darts they concern, and an edge indicates that two overlaps are compatible (do not contain common darts). Finding a clique with maximum sum of weights gives the largest possibly disconnected overlap. This procedure clearly runs in exponential time, and the following section shows that we cannot hope better.

6 Finding Large Disconnected Overlaps Is Intractable

We consider the following problem:

Name LARGE_DISCONNECTED_OVERLAP;
Instance An integer N, and two semi-open maps M_1 and M_2;
Problem Does there exist a disconnected overlap $h : U_1 \to U_2$ s.t. $|U_1| \geq N$?

We get the following result:

Theorem 5. *Problem* LARGE_DISCONNECTED_OVERLAP *is $\mathcal{NP}$-complete.*

Proof. Basically, Problem LARGE_DISCONNECTED_OVERLAP is in class $\mathcal{NP}$ since any certificate h can easily be verified. Now consider following problem:

Name DISCONNECTED_PATTERN;
Instance Two semi-open maps M_1 and M_2;
Problem Is map M_1 a disconnected pattern of map M_2?

One can prove, by reduction from SEPARABLE_PLANAR_3SAT [9, Lem. 1], that this problem is $\mathcal{NP}$-complete. We do not give the details here: the proof is essentially the same as that provided in [6, Sect. 4] [3]. We can finally reduce DISCONNECTED_PATTERN to LARGE_DISCONNECTED_OVERLAP: we simply need to fix $N = |D_1|$. Indeed, map M_1 is a pattern of map M_2 iff an overlap $h : U_1 \to U_2$ exists with $|U_1| \geq |D_1|$ (that is, $U_1 = D_1$). □

7 Conclusion and Future Works

Computing the overlaps has two advantages over computing the common patterns of two maps: on one hand, the optimisation problems are tractable (Sect. 5), and on the other, the overlaps are maximal objects (Sect. 4).

[3] Actually, Problem DISCONNECTED_PATTERN, which is defined for semi-open maps, is an instance of Problem INDUCED_SUBMAP_ISOMORPHISM, which is defined for nG-maps, but proved $\mathcal{NP}$-complete by using 2G-maps, and gadgets that can easily be redefined in terms of semi-open maps. Thus rewriting such a proof is of no relevance.

The overlaps allow us also to consider super-maps, where a super-map of M_1 and M_2 is a map of which both M_1 and M_2 are patterns. Then the smallest common super-map is obtained by adding to the largest overlap the faces which belong to M_1 and M_2 but are not matched.

Super-maps offer interesting possibilities as smallest common super-maps would be constructable in polynomial time whereas their duals, the largest common sub-maps, are not.

Finally, notice that the techniques introduced in this paper can, with no difficulty, be extended to *open* maps[4], nD-maps and n-Gmaps [10].

References

1. Cori, R.: Un code pour les graphes planaires et ses applications. PhD thesis, Université Paris 7 (1973)
2. Damiand, G., de la Higuera, C., Janodet, J.C., Samuel, E., Solnon, C.: Polynomial algorithm for submap isomorphism: Application to searching patterns in images. In: Proc. 7th IAPR International Workshop on Graph Based Representation in Pattern Recognition (GbRPR'09)., LNCS 5534 (2009) 102–112
3. de la Higuera, C., Janodet, J.C., Samuel, E., Damiand, G., Solnon, C.: Polynomial algorithms for open plane graph and subgraph isomorphisms. Theoretical Computer Science **498** (2013) 76–99
4. Dorn, F.: Planar subgraph isomorphism revisited. In: Proc. 27th International Symposium on Theoretical Aspects of Computer Science (STACS'10). Volume 5 of LIPIcs., Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010) 263–274
5. Eppstein, D.: Subgraph isomorphism in planar graphs and related problems. Journal of Graph Algorithms and Applications **3**(3) (1999) 1–27
6. Solnon, C., Damiand, G., de la Higuera, C., Janodet, J.C.: On the complexity of submap isomorphism and maximum common submap problems. Pattern Recognition **48**(2) (2015) 302–316
7. Poudret, M., Arnould, A., Bertrand, Y., Lienhardt, P.: Cartes combinatoires ouvertes. Research Notes 2007-1, Laboratoire SIC E.A. 4103, F-86962 Futuroscope Cedex - France (2007)
8. Damiand, G., Solnon, C., de la Higuera, C., Janodet, J.C., Samuel, E.: Polynomial algorithms for subisomorphism of nd open combinatorial maps. Computer Vision and Image Understanding **115**(7) (2011) 996–1010
9. Lichtenstein, D.: Planar formulæ and their uses. SIAM Journal of Computing **11**(2) (1982) 329–343
10. Damiand, G., Lienhardt, P.: Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing. A K Peters/CRC Press (2014)

[4] for which the α-functions may be *partial permutations* [7].

A Proof of Theorem 3

Proposition 1. *Consider a real connected component $C = \langle D, \alpha, \beta \rangle$ of product $M_1 \otimes M_2$. Then the set D is the graph of a connected overlap of M_1 and M_2, that is, if we set $h(d_1) = d_2$ for all $(d_1, d_2) \in D$, then h is a connected overlap.*

Proof. Let $U_1 = \{d_1 : (d_1, d_2) \in D \text{ for any } d_2 \in D_2\}$ and $U_2 = \{d_2 : (d_1, d_2) \in D \text{ for any } d_1 \in D_1\}$. We basically have $U_1 \subseteq D_1$ and $U_2 \subseteq D_2$. Let h be the relation defined be D. As connected component C is real, if $(d_1, d_2) \in D$ and $(d_1, d_2') \in D$, then $d_2 = d_2'$ so relation h is a function from U_1 to U_2. Moreover, for any $d_2 \in U_2$ there exists $d_1 \in D_1$ such that $(d_1, d_2) \in D$, so there exists $d_1 \in U_1$ such that $h(d_1) = d_2$, that is, h is surjective. Finally, let $d_1, d_1' \in U_1$ such that $h(d_1) = h(d_1') = d_2$ for some $d_2 \in D_2$. We have that $(d_1, d_2) \in D$ and $(d_1', d_2) \in D$. Since connected component C is real, we deduce that $d_1 = d_1'$, so function h is injective. In consequence, $h : U_1 \to U_2$ is a bijection.

We claim that h is a matching. Indeed, let $d_1 \in U_1$ and $d_2 = h(d_1)$. Component C is connected and $(d_1, d_2) \in D$, so we deduce that $\alpha(d_1, d_2) \in D$, that is, $(\alpha_1(d_1), \alpha_2(d_2)) \in D$. Therefore we have $\alpha_1(d_1) \in U_1$ and $\alpha_2(d_2) \in U_2$ and $h(\alpha_1(d_1)) = \alpha_2(d_2)$. Now suppose that $\beta_1(d_1) \in U_1$. We have $(d_1, d_2) \in D$, and there exists $d_2' \in U_2$ such that $(\beta_1(d_1), d_2') \in D$. Since component C is real, we deduce that $d_2' = \beta_2(d_2)$, so $\beta_2(d_2) \in U_2$ and $h(\beta_1(d_1)) = \beta_2(d_2)$. Conversely, if $\beta_2(d_2) \in U_2$, then there exists $d_1' \in U_1$ such that $(d_1', \beta_2(d_2)) \in D$, and we have $(d_1, d_2) \in D$. As component C is real, we deduce that $d_1' = \beta_1(d_1)$, thus $\beta_1(d_1) \in U_1$.

We now show that h is an overlap. Let $d_1 \in U_1$ and $d_2 = h(d_1)$, that is, $(d_1, d_2) \in D$. In product map $M_1 \otimes M_2$, we have $\alpha(d_1, d_2) = (\alpha_1(d_1), \alpha_2(d_2))$, so the darts (d_1, d_2) and $(\alpha_1(d_1), \alpha_2(d_2))$ are α-sewn. Therefore $(\alpha_1(d_1), \alpha_2(d_2))$ is a dart of connected component C, thus $\alpha_1(d_1) \in U_1$. Now suppose that $\beta_1(d_1) \neq \varepsilon$ and $\beta_2(d_2) \neq \varepsilon$. Then in product $M_1 \otimes M_2$, we have $\beta(d_1, d_2) = (\beta_1(d_1), \beta_2(d_2))$. The darts (d_1, d_2) and $(\beta_1(d_1), \beta_2(d_2))$ are β-sewn, so $(\beta_1(d_1), \beta_2(d_2))$ is a dart of connected component C. Therefore, $\beta_1(d_1) \in U_1$ and $\beta_2(d_2) \in U_2$.

Finally, let us prove that h is a connected overlap. Let $e, e' \in U_1$. There exist $f, f' \in U_2$ such that $(e, f), (e', f') \in D$. Component C is connected, so in product map $M_1 \otimes M_2$, there is a sequence $(e, f) = (d_0, d_0'), \ldots, (d_n, d_n') = (e', f')$ such that $(d_{k+1}, d_{k+1}') = \delta_k(d_k, d_k')$ with $\delta_k \in \{\alpha, \beta\}$ for all $0 \leq k < n$. By construction of $M_1 \otimes M_2$, we deduce that in M_1, there is a sequence $e = d_0, \ldots, d_n = e'$ such that $d_{k+1} = \gamma_k(d_k)$ with $\gamma_k = \alpha_1$ if $\delta_k = \alpha$, and $\gamma_k = \beta_1$ if $\delta_k = \beta$, for all $0 \leq k < n$. Hence U_1 is connected, thus h is a connected overlap. □

Proposition 2. *For each connected overlap $h : U_1 \to U_2$, there exists a real connected component $C = \langle D, \alpha, \beta \rangle$ of product $M_1 \otimes M_2$ whose set of darts is the graph of h.*

Proof. Let $h : U_1 \to U_2$ be an connected overlap. We define $D = \{(d, h(d)) : d \in U_1\}$. Since h is an instance of matching, we have $U_1 \subseteq D_1$ and $U_2 \subseteq D_2$, so $D \subseteq D_1 \times D_2$, that is, D is a subset of the darts of product map $M_1 \otimes M_2 = \langle D_1 \times D_2, \alpha, \beta \rangle$.

Let us show that D is stable with respect to functions α and β. Let $(d_1, d_2) \in D$ and $(d'_1, d'_2) = \alpha(d_1, d_2) = (\alpha_1(d_1), \alpha_2(d_2))$. We have $d_1 \in U_1$ and $d_2 = h(d_1)$. Since h is an overlap, we have $\alpha_1(d_1) \in U_1$, and since h is an instance of matching, we deduce that $\alpha_2(d_2) \in U_2$ and $h(\alpha_1(d_1)) = \alpha_2(d_2)$, so $(d'_1, d'_2) \in D$. Now, with respect to β, suppose that $\beta(d_1, d_2) \neq \varepsilon$. Then $\beta_1(d_1) \neq \varepsilon$ and $\beta_2(d_2) \neq \varepsilon$, so let $(d'_1, d'_2) = \beta(d_1, d_2) = (\beta_1(d_1), \beta_2(d_2))$. We have $d_1 \in U_1$ and $d_2 = h(d_1)$. Since h is an overlap, we have $\beta_1(d_1) \in U_1$ and $\beta_2(d_2) \in U_2$, and as h is an instance of matching, we deduce that $h(\beta_1(d_1)) = \beta_2(d_2)$, so $(d'_1, d'_2) \in D$. On the other hand, if $\beta(d_1, d_2) = \varepsilon$, then the property holds if we suppose that ε implicitly belongs to D.

Hence, map $M = \langle D, \alpha, \beta \rangle$ is a sub-map of product map $M_1 \otimes M_2 = \langle D_1 \times D_2, \alpha, \beta \rangle$. We now show that M is a *connected* component of $M_1 \otimes M_2$. Let $(d, f), (d', f') \in D$. We have $d, d' \in U_1$ and $f, f' \in U_2$. Since h is an connected overlap, U_1 is connected, so there exists a sequence $(d = d_0, d_1, \ldots, d_n = d')$ such that $d_{k+1} = \gamma_k(d_k)$ with $\gamma_k \in \{\alpha_1, \beta_1\}$ for all $0 \leq k < n$. We claim that we can dub this sequence with another sequence $(f = f_0, f_1, \ldots, f_n = f')$ in U_2 where (1) $f_k = h(d_k)$ and (2) $f_{k+1} = \delta_k(f_k)$ with $\delta_k = \alpha_2$ if $\gamma_k = \alpha_1$, and $\delta_k = \beta_2$ if $\gamma_k = \beta_1$, for all $0 \leq k < n$. Indeed, we have $h(d) = f$, so $h(d_0) = f_0$. Let $0 \leq k < n$, and suppose that $h(d_k) = f_k$. On the one hand, if $d_{k+1} = \alpha_1(d_k) \in U_1$, then $\alpha_2(f_k) \in U_2$ and $h(\alpha_1(d_k)) = \alpha_2(f_k)$ since h is an instance of matching. So we fix $f_{k+1} = \alpha_2(f_k)$ and the property holds. On the other hand, if $d_{k+1} = \beta_1(d_k) \in U_1$, then $\beta_2(f_k) \in U_2$ and $h(\beta_1(d_k)) = \beta_2(f_k)$ since h is an instance of matching. So we fix $f_{k+1} = \beta_2(f_k)$ and the property holds. Finally, we have $f_n = h(d_n) = h(d') = f'$. Using both sequences, we conclude that in sub-map M, there exists a sequence $((d, f) = (d_0, f_0), (d_1, f_1), \ldots, (d_n, f_n) = (d', f'))$ such that $(d_{k+1}, f_{k+1}) = \zeta_k(d_k, f_k)$ with $\zeta_k = \alpha$ if $\gamma_k = \alpha_1$ (and $\delta_k = \alpha_2$), and $\zeta_k = \beta$ if $\gamma_k = \beta_1$ (and $\delta_k = \beta_2$). In other words, sub-map M is connected.

We finally need to show that connected component M is *real*, and this follows from the fact that h is an instance of matching. Indeed, let $(d_1, d_2), (d'_1, d'_2) \in D$. We have $d_2 = h(d_1)$ and $d'_2 = h(d'_1)$. So if $d_1 = d'_1$ then $d_2 = d'_2$ because h is a function, and conversely, if $d_2 = d'_2$, then $d_1 = d'_1$ because h is a bijection. Now if $d'_1 = \beta_1(d_1)$, then $\beta_2(d_2) \in U_2$ and $h(\beta_1(d_1)) = \beta_2(d_2)$, so $h(d'_1) = \beta_2(d_2)$, that is, $d'_2 = \beta_2(d_2)$, and conversely, if $d'_2 = \beta_2(d_2)$, then the same arguments yield $d'_1 = \beta_1(d_1)$. □

B Proof of Theorem 1 (Sketch)

We here prove the correctness of Algorithm 1, by exploiting Theorem 3. So let $M_1 = \langle D, \alpha_1, \beta_1 \rangle$ and $M_2 = \langle D_2, \alpha_2, \beta_2 \rangle)$ be two semi-open maps, and $d_1 \in D_1$ and $d_2 \in D_2$ two darts.

Consider the product map $M_1 \otimes M_2$ and let $C = \langle D, \alpha, \beta \rangle$ be the connected component such that $(d_1, d_2) \in D$. Following Theorem 3, either component C is real and there exists a connected overlap $\varphi : U_1 \to U_2$ such that $\varphi(d_1) = d_2$, or component C is imaginary and no such connected overlap exists.

Now, consider Algorithm 1. By Theorem 2, this procedure halts after no more than $\min(|D_1|, |D_2|)$ iterations. After this run, we define two sets:

$$V = \{(d, d') \in D_1 \times D_2 \mid (d, d') \text{ was popped from } S \text{ at some point}\}$$
$$W = \{(d, d') \in D_1 \times D_2 \mid (d, d') \text{ was pushed in } S \text{ at some point}\}$$

We clearly have $(d, d') \in W$ iff $h[d] = d'$ iff $g[d'] = d$.

Moreover, for all $(d, d') \in W$, there is a sequence $((e_0, f_0), (e_1, f_1), \ldots, (e_n, f_n))$ such that (1) $(e_0, f_0) = (d_1, d_2)$ and $(e_n, f_n) = (d, d')$ and (2) $(e_k, f_k) \in V$ for all $0 \leq k < n$ and (3) $(d_{k+1}, f_{k+1}) = \gamma_k(d_k, f_k)$ with $\gamma_k \in \{\alpha, \beta\}$ for all $0 \leq k < n$.

As component C is connected, we deduce that $V \subseteq W \subseteq D$.

When the Algorithm Returns `false`:

We suppose that checkConnectedOverlap(M_1, M_2, d_1, d_2) returns `false`. Consider the couple $(d, d') \in V$ that produced this early exit.

First case: let $a = \alpha_1(d)$ and $a' = \alpha_2(d')$, and either $h[a] \neq a'$ or $g[a'] \neq a$. Suppose that $h[a] = f \neq a'$. On the one hand, we have $(a, a') \in D$, since $(d, d') \in V \subseteq W \subseteq D$ and $(a, a') = \alpha(d, d')$. On the other hand, we have $(a, f) \in W$ since $h[a] = f$, so $(a, f) \in D$. As $a' \neq f$, we deduce that component C is imaginary. Same conclusion if $g[a'] \neq a$.

Second case: let $b = \beta_1(d)$ and $b' = \beta_2(d')$ and suppose that $b \neq \varepsilon$ and $b' \neq \varepsilon$ and either $h[b] \neq b'$ or $g[b'] \neq b$. Same case as Case 1.

Third case: let $b = \beta_1(d)$ and $b' = \beta_2(d')$ and suppose that $b = \varepsilon$ and $b' \neq \varepsilon$ and $g[b'] = e \neq nil$. We have $(d, d') \in D$ and $(e, b') \in D$ and $b' = \beta_2(d')$ but $e \neq \beta_1(d)$, so component C is imaginary. Same conclusion if $b \neq \varepsilon$ and $b' = \varepsilon$ and $h[b] = f \neq nil$.

So we conclude that if checkConnectedOverlap(M_1, M_2, d_1, d_2) returns `false`, then component C is imaginary. Following Theorem 3, no connected overlap h such that $h(d_1) = d_2$ exists in this case.

When the Algorithm Returns `true`:

We now suppose that checkConnectedOverlap(M_1, M_2, d_1, d_2) returns `true`. Then we have $V = W$ (since `true` is returned only if stack S is empty) and $V \subseteq D$.

We actually have $D \subseteq V$, thus $D = V$. Indeed, let $(d, d') \in D$. As component C is connected, there exists in C a sequence $((d_1, d_2) = (e_0, f_0), (e_1, f_1), \ldots, (e_n, f_n) = (d, d'))$ such that $(d_{k+1}, f_{k+1}) = \gamma_k(d_k, f_k)$ with $\gamma_k \in \{\alpha, \beta\}$ for all $0 \leq k < n$. By induction, we get $(d_k, f_k) \in V$ for all $0 \leq k \leq n$, so $(d, d') \in V$.

Now let us prove that component C is real, so let $(e_1, f_1), (e_2, f_2) \in D$, that is, $(e_1, f_1) \in V$ and $(e_2, f_2) \in V$. Suppose that $e_1 = e_2$ and $f_1 \neq f_2$, or $e_1 \neq e_2$ and $f_1 = f_2$, thus couples (e_1, f_1) and (e_2, f_2) are distinct. Assume without loss of generality that (e_1, f_1) was pushed in S first, before (e_2, f_2) was pushed in S. When couple (e_2, f_2) is met, we have $h[e_1] = f_1$ and $g[f_1] = e_1$. Moreover couple

(e_2, f_2) is met when a couple (d, d') is popped from S and $(e_2, f_2) = \gamma(d, d')$ with $\gamma \in \{\alpha, \beta\}$. If $e_1 = e_2$, then we have $h[e_2] = h[e_1] = f_1 \neq nil$. If $h[e_2] \neq f_2$, then the procedure returns `false`, which contradicts the assumption, so $h[e_2] = f_2$, that is, $f_1 = f_2$. On the other hand, if $f_1 = f_2$, then we have $g[f_2] = g[f_1] = e_1 \neq nil$. If $g[f_2] \neq e_2$, the procedure returns `false`, so $e_1 = e_2$ likewise.

We now suppose that $e_2 = \beta_1(e_1)$ (or equivalently that $e_1 = \beta_1(e_2)$, since $\beta_1(\beta_1(d)) = d$ for all darts d such that $\beta_1(d) \neq \varepsilon$). We assume again that (e_1, f_1) was pushed in S before (e_2, f_2) was pushed in S. Let us look at the point where (e_2, f_2) was popped from S. We have $h[e_1] = f_1$ and $g[f_1] = e_1$ and $h[e_2] = f_2$ and $g[f_2] = e_2$. Let $b = \beta_1(e_2)$ and $b' = \beta_2(f_2)$. As $e_2 = \beta_1(e_1)$ and $b = \beta_1(e_2)$, we deduce that $b = e_1$, so $b \neq \varepsilon$. If $b' = \varepsilon$, then as $b \neq \varepsilon$ and $b' = \varepsilon$ and $h[b] = h[e_1] = f_1 \neq nil$, the procedure returns `false`, which contradicts the assumption, so $b' \neq \varepsilon$. Therefore, we have $b \neq \varepsilon$ and $b' \neq \varepsilon$ and $h[b] = h[e_1] = f_1 \neq nil$. If $h[b] \neq b'$, then the procedure returns `false`, which contradicts the assumption. So $h[b] = b'$, that is, $h[e_1] = b'$, and since $h[e_1] = f_1$, we deduce that $b' = f_1 = \beta_2(f_2)$, thus $f_2 = \beta_2(f_1)$. If instead of assuming that $e_2 = \beta_1(e_1)$, we suppose that $f_2 = \beta_2(f_1)$, then we can easily adapt the proof to reach the same conclusion.

Hence, if checkConnectedOverlap(M_1, M_2, d_1, d_2) returns `true`, then component C is real. By Theorem 3, we know that the set D, that is the set V, is the graph of a connected overlap of M_1 and M_2. Since $(d, d') \in V$ iff $h[d] = d'$, we deduce that h is a connected overlap such that $h[d_1] = d_2$. □