Embedded Subgraph Isomorphism and Related Problems

Graph isomorphism, subgraph isomorphism, and maximum common subgraph can be solved in polynomial time when constrained by geometrical information, in particular by the circular ordering in which the edges appear around vertices in a combinatorial embedding.

An embedded graph is just a graph with a combinatorial embedding of the edges around each vertex, that is, a graph in which the circular ordering of the edges around each vertex is fixed.

All embedded graphs are assumed to be connected.

Two embedded graphs are isomorphic if the underlying graphs are isomorphic and the isomorphism preserves and reflects not only the structure of the graphs but also their combinatorial embeddings.

The idea of embedded subgraph isomorphism arises as a natural generalization of the subgraph isomorphism problem for triconnected planar graphs, which have a unique combinatorial embedding in the plane.

Embedded Subgraph Isomorphism and Related Problems

- L. Weinberg. A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. *IEEE Trans. on Circuit Theory*, 13(2):142–148, 1966.
- X. Y. Jiang and H. Bunke. Including geometry in graph representations: A quadratic-time graph isomorphism algorithm and its applications. In *Advances in Structural and Syntactical Pattern Recognition*, volume 1121 of *Lecture Notes in Computer Science*, pages 110–119. Springer-Verlag, 1996.
- X. Y. Jiang and H. Bunke. Optimal quadratic-time isomorphism of ordered graphs. *Pattern Recogn.*, 32(7):1273–1283, 1999.
- G. Valiente. A general method for graph isomorphism. Submitted for publication, 2001.

- X. Y. Jiang and H. Bunke. Marked subgraph isomorphism of ordered graphs. In *Advances in Pattern Recognition*, volume 1451 of *Lecture Notes in Computer Science*, pages 122–131. Springer-Verlag, 1998.

- C. Schlieder. Diagrammatic transformation processes on two-dimensional relational maps. *Journal of Visual Languages and Computing*, 9(1):45–59, 1998.

Embedded Graphs

An embedded graph is a graph with a combinatorial embedding of the edges around each vertex.

Definition. *An embedded graph* $G = (V, E, L)$ *is a graph* (V, E) *together with a set* $L = \{L(v)\}$ *of ordered, circular lists of edges incident to each vertex* $v \in V$*.*

Two embedded graphs are isomorphic if the underlying graphs are isomorphic and the isomorphism preserves and reflects not only the structure of the graphs but also their combinatorial embeddings.

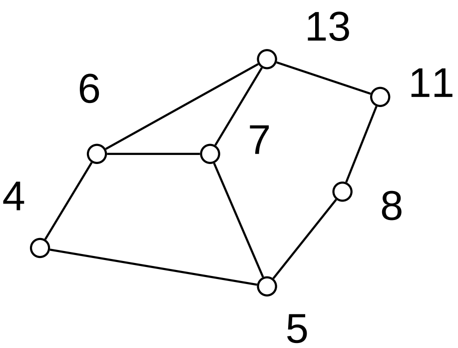
Definition. *An embedded graph isomorphism between two embedded graphs* $G_1 = (V_1, E_1, L_1)$ *and* $G_2 = (V_2, E_2, L_2)$ *is a graph isomorphism* $f : V_1 \to V_2$ *between* (V_1, E_1) *and* (V_2, E_2) *such that* $L_2(f(v))$ *is a cyclic rotation of* $f(L_1(v))$*, for all vertices* $v \in V_1$*.*

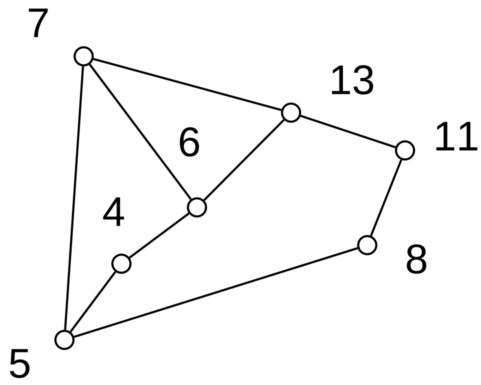
Definition. *An embedded subgraph isomorphism of an embedded graph* $G_1 = (V_1, E_1, L_1)$ *into an embedded graph* $G_2 = (V_2, E_2, L_2)$ *is a subgraph isomorphism* $f : V_1 \to V_2$ *of* (V_1, E_1) *into* (V_2, E_2) *such that* $L_2(f(v))$ *is a cyclic rotation of* $f(L_1(v))$*, for all vertices* $v \in V_1$*.*

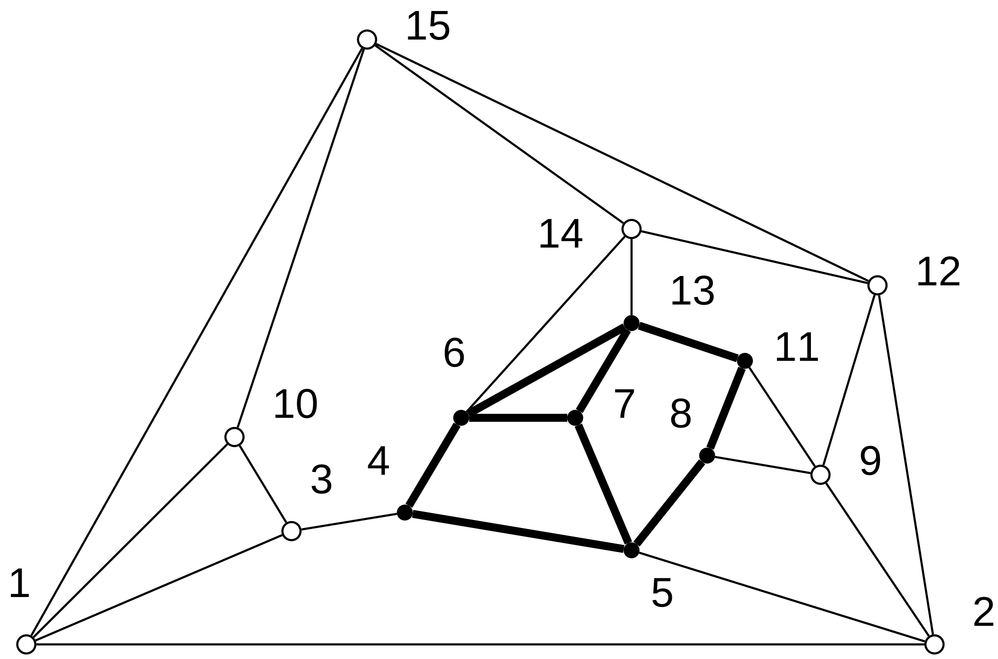
Definition. *An embedded maximum common subgraph between two embedded graphs* $G_1 = (V_1, E_1, L_1)$ *and* $G_2 = (V_2, E_2, L_2)$ *is an embedded graph* $G_3 = (V_3, E_3, L_3)$ *such that* (V_3, E_3) *is a maximum common subgraph of* (V_1, E_1) *and* (V_2, E_2)*, and there exist embedded subgraph isomorphisms of* G_3 *into* G_1 *and into* G_2*.*

Embedded Graphs

Consider the following plane embedding of a sample triconnected planar graph.







Embedded Graph and Subgraph Isomorphism

In a finite, undirected, connected graph it is always possible to construct a cyclic directed path passing through each edge once and only once in each direction.

An Eulerian path through an undirected graph can be constructed by traversing each edge of the corresponding bidirected graph exactly once in each direction, what guarantes that the degree of each vertex is even. Such a traversal is called a leftmost depth-first traversal, since the edges are explored in left-to-right order (if drawn downwards) for any vertex of the graph and, more generally, the whole graph is explored in a left-to-right fashion.

An algorithm was formulated by Trémaux and recalled by Weinberg for finding a way out of a maze, that is, for the leftmost depth-first traversal of an undirected graph. Starting with an edge traversed in one of its directions,

- When a non-visited vertex is reached, take the next (in the counter-clockwise ordering of the edges around the vertex) edge.
- When a visited vertex is reached along a non-visited edge, take the same edge but in the opposite direction.
- When a visited vertex is reached along a visited edge, take the next (in the counter-clockwise ordering of the edges around the vertex) non-visited edge, if any.

Embedded Graph and Subgraph Isomorphism

The following algorithm performs a leftmost depth-first traversal of an embedded graph G starting with edge e.

```
procedure LMDFS(G, e)
    let v be the target of edge e
    let e_r be the reverse of edge e
    if vertex v has been visited then
        if edge e_r has been visited then
            let e' be e_r
            repeat
                let e' be the cyclic successor of edge e'
            until e' = e_r or edge e' has not been visited
            if edge e' has been visited then
                return
        else
            let e' be e_r
    else
        let e' be the cyclic successor of edge e_r
    mark edge e as visited
    mark vertex v as visited
    LMDFS(G, e')
end procedure
```

The following algorithm performs a synchronized LMDFS on two embedded graphs, starting with edges e_1 of G_1 and e_2 of G_2.

Embedded Graph and Subgraph Isomorphism

procedure $match(G_1, G_2, e_1, e_2, M)$

let v_1 be the target of edge e_1

let v_2 be the target of edge e_2

if vertex v_1 has been visited **then**

if reversal of edge e_1 has been visited **then**

let e'_1 be the reversal of edge e_1

let e'_2 be the reversal of edge e_2

let e''_1 be e'_1

repeat

let e'_1 be the cyclic successor of edge e'_1

let e'_2 be the cyclic successor of edge e'_2

until $e'_1 = e''_1$ or edge e'_1 has not been visited

if edge e'_1 has been visited **then**

return

else

let e'_1 be the reversal of edge e_1

let e'_2 be the reversal of edge e_2

else

let e'_1 be the cyclic successor of reversal of edge e_1

let e'_2 be the cyclic successor of reversal of edge e_2

add (v_1, v_2) to vertex mapping M

mark edge e_1 and vertex v_1 as visited

$match(G_1, G_2, e'_1, e'_2, M)$

end procedure

Embedded Graph and Subgraph Isomorphism

As the synchronized leftmost depth-first traversal proceeds, procedure *match* extends a vertex mapping $M : V_1 \to V_2$ into the maximal vertex mapping representing an embedded subgraph isomorphism of a subgraph of G_1 into G_2.

Starting with an empty mapping, the following algorithm finds, whenever possible, a vertex mapping $M : V_1 \to V_2$ representing an embedded subgraph isomorphism of an embedded graph G_1 into an embedded graph G_2.

```
function subgraph_isomorphism(G1, G2, M)
   let e1 be an edge of G1
   for all edges e2 of G2 do
      let M be an empty vertex mapping
      match(G1, G2, e1, e2, M)
      let s be the size of M
      if s = n1 then
         return true
   return false
end function
```

Embedded Maximum Common Subgraph

The following algorithm finds a vertex mapping $M : V_1 \to V_2$ representing a common subgraph of largest size of two nonempty embedded graphs G_1 and G_2.

procedure $maximum\_common\_subgraph(G_1, G_2, M)$
    let s' be zero
    for all edges e_1 of G_1 **do**
        for all edges e_2 of G_2 **do**
            let M be an empty vertex mapping
            $match(G_1, G_2, e_1, e_2, M)$
            let s be the size of M
            if $s > s'$ **then**
                let e'_1 be e_1
                let e'_2 be e_2
                let s' be s
    $match(G_1, G_2, e'_1, e'_2, M)$
end procedure

If an edge $e_1 = (u_1, v_1)$ of G_1 corresponds to an edge $e_2 = (u_2, v_2)$ of G_2 in the mapping associated to an embedded maximum common subgraph of G_1 and G_2, then the maximal embedded common subgraph of G_1 and G_2 containing a mapping of vertex u_1 to vertex u_2 and vertex v_1 to vertex v_2 will have been found by the algorithm when performing the synchronized leftmost depth-first traversal starting with e_1 and e_2.

Embedded Subgraph Isomorphism and Related Problems

All embedded graphs are assumed to be connected.

Given two embedded graphs G_1 and G_2 with $n_1 = n_2$, it is clear that G_1 and G_2 are isomorphic if, and only if, there is an embedded subgraph isomorphism of G_1 into G_2. Therefore, algorithm *subgraph_isomorphism* also solves the embedded graph isomorphism problem.

Since the *match* algorithm visits every edge of the embedded graphs at most once in each direction, the worst-case time complexity is $O(m_1 + m_2)$, and the space complexity is $O(m_1 + m_2)$.

The worst-case time complexity of the *subgraph_isomorphism* algorithm is $O((m_1 + m_2)m_2)$, and the space complexity is $O(n_1 n_2 + m_1 + m_2)$.

The worst-case time complexity of the *maximum_common_subgraph* algorithm is $O((m_1 + m_2)m_1 m_2)$, and the space complexity is, again, $O(n_1 n_2 + m_1 + m_2)$.

The algorithms can be readily extended in order to enumerate all embedded subgraph isomorphisms or all maximum common subgraphs between two embedded graphs.