Chapter 5

A related problem

5.1 Introduction

Each planar graph has an embedding in the plane with line segments as edges. But it seems to be interesting to ask whether this is possible under various constraints. In [EW90] this is studied for the case where we restrict the length of each edge to be a fixed value. On the other hand we can assign each vertex a set of points in the plane and ask whether it is possible to embed the graph with line segments as edges with the additional constraint that each vertex must lie in its point set. Here we show that we can achieve NP-hardness if the point sets are closed disks.

From now we only consider oriented planar triangulated graphs. For each triangle in the graph an orientation consists of an enumeration of its vertices modulo an even permutation. We only consider positively oriented embeddings. These have the additional property that all the vertices of a triangle are positive oriented (i.e. are in counter clockwise order when embedded). The results hold for the non oriented case as well because oriented instances can be converted into non oriented instances by forcing all embeddings to be oriented by adding one big triangle around the disks assigning each vertex of this triangle a disk of radii zero and connecting the outer vertices of the graph with the triangle in some obvious way.

Thus let us redefine the problem as follows and refer to it as

constrained embedding

> We are given a labeled triangulated oriented planar graph, each vertex of which is assigned to a closed disk in the plane given by a center point of rational coordinates and a rational radius. Decide whether it is possible to embed the graph positively oriented in the plane with line segments as edges such that each vertex lies in its disk.

In this chapter we will see that constrained embedding is NP-hard. We can even restrict the graphs to be tri connected, see subsection 5.2.8. Note that this chapter is actually an extended (i.e. the full) version of [God95].

5.2 The Reduction

In order to prove that constrained embedding is NP-hard we reduce grid3sat to it. Unless you already have done it, read 3.5 on page 21–25 first.

Let I be an instance of grid3sat. Let us try to construct an instance J of constrained embedding which is solvable if and only if the formula described by I is satisfiable. It is mainly a question of finding a triangulated oriented planar graph which can be embedded under certain constraints if and only if the formula in I is satisfiable. Obviously we should search for a correspondence between an embedding of the graph and a truth assignment for the variables in the formula.

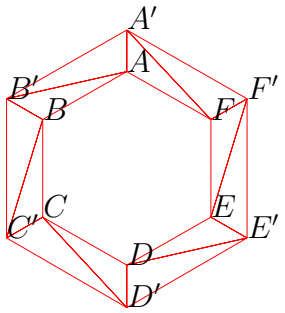
5.2.1 Notions

We will indicate instances of constrained embedding by drawing some graph, and this drawing also defines the orientation of the graph. We will also have to define the labels of the vertices of the graph, which will be subsequently called the ***orbits*** of the vertices.

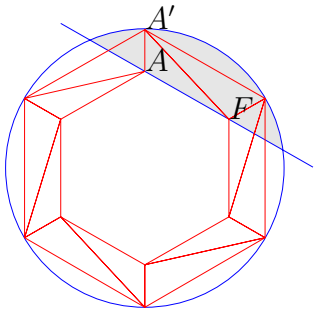
The orbit of a vertex is a disk anyway. If the radius of the disk of is zero we say that the vertex is ***fixed***. Vertices are fixed unless stated otherwise. In most cases the drawn embedding will be a correct embedding.

5.2.2 Honeycombs

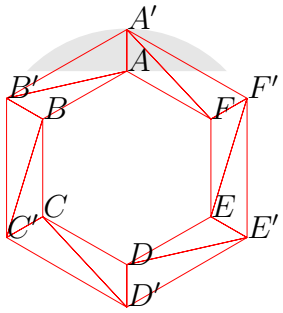
Let us consider first a single variable. A variable can have exactly two states. Let us try to give an oriented graph which can be embedded under certain constraints in only two different ways. Consider the following oriented graph.



Let us consider now embeddings for which the positions of $A, ..., F$ are fixed and the positions of $A', ..., F'$ have to lie in a circle Z centered at the center of the circumcircle of the hexagon $A...F$ and only a little bigger than this circumcircle. What does this mean for instance for A'?



Because of the oriented triangle $A'AF$ the vertex A' has to lie in the half-space above the line AF. Aside from this the vertex A' has to lie in its circle, namely Z. Thus A' has to lie in the circular segment defined to be the intersection of this half-space and this disk (see figure above). We will refer to this as ***fact (1)***. The other vertices $B', ..., F'$ also have to lie in certain circular segments. Since the vertices A' and B' are connected they have to be visible to each other in a correct embedding. To achieve this either A' or B' has to lie within the shaded region shown below. We will refer to this as ***fact (2)***.



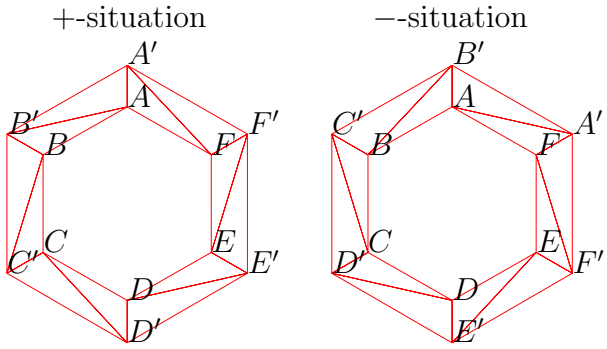
Proof of fact (2). Assume A' and B' would lie both below A. Then by fact (1) B' has to be left of A and A' has to be to the right of A. By fact (1) and since Z is comparatively small A' and B' has to be above D. Then A' and B' can not see each other, because the hexagon $A...F$ would lie in between. □

In the following we want to say that a point lies ***close*** to A if it is in this region. Around the vertices $B, ..., F$ there are similar regions. Together with the region for A these regions will be disjoint if Z sufficiently small. In the following we assume Z to be nearly of the same size as the circumcircle of $A...F$. So near that there will be no noticeable difference in the drawings. Then we do have disjoint regions. By fact (2) in each region at least one of the vertices $A', ..., F'$ has to lie. Since there are six points and six disjoint regions each point has to lie in *exactly one* region.

Due to fact (1) for each point only two regions come into question. The following table shows these possibilities.

	close to					
	A	B	C	D	E	F
Position of A'	*					*
B'	*	*				
C'		*	*			
D'			*	*		
E'				*	*	
F'					*	*

The reader can verify that in principle there are only two possibilities: either A' is close to A, B' is close to B etc. as indicated in the drawings above, or A' is close to F, B' is close to A, etc. We can imagine that $A', ..., F'$ form an outer hexagon which can be in two situations differing by a rotation of 60 degrees. The situation twisted in clockwise orientation we denote ***−-situation*** and the other one ***+-situation***.



We are not able to simulate the behavior of a variable in the sense that there exist exactly two embeddings which correspond to the two truth assignments of the variable but we are able to do it in an approximate sense such that there are two distinguishable classes of embeddings.

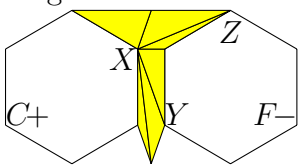
In the following we will indicate these key parts of the variables only by the outer hexagon and talk only about embeddings of the outer vertices and pretend that there exists indeed only two embeddings, namely the − and +-situation.

These key parts of the variables will occur on many places and will be used for different purposes. Therefore we want to give them a name, let us call such a labeled graph consisting of the vertices $A, ..., F, A', ..., F'$ a ***honeycomb***.

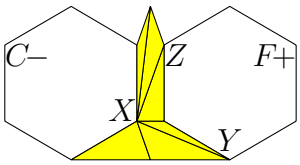
5.2.3 Negations

The following example shows how to combine two honeycombs such that one of them is in +-situation if the other one is in −-situation and vice versa. All vertices not belonging to the honeycombs are assumed to be fixed.

Here the left one is in +-situation
and the right one is in −-situation.



... and this is the reverse case.



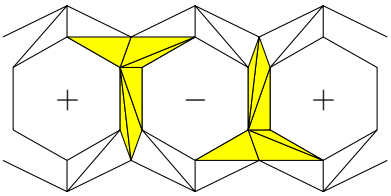
To see why $C{+}F{+}$ is not embeddable note that in this case X and Y would not see each other. To see why $C{-}F{-}$ is not embeddable note that in this case X and Z would not see each other.

5.2.4 Chains

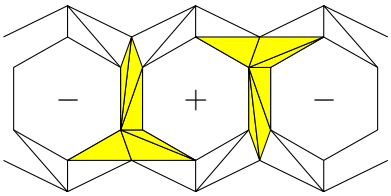
Using the negations we can build long chains of coupled honeycombs. These chains will be subsequently called ***chains*** for short.

Later on all parts of our construction will be ***encapsulated*** in the sense that the boundary of the construction will consist of fixed vertices and edges in between. The following picture shows a part of an encapsulated chain. Again, any vertex not belonging to the honeycombs is fixed.

case $+-+$

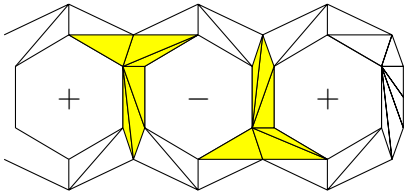


case $-+-$

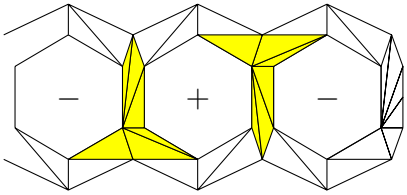


The following pictures indicate how chains end blindly, if necessary. Again, any vertex not belonging to the honeycombs is fixed.

case $+-+$



case $-+-$

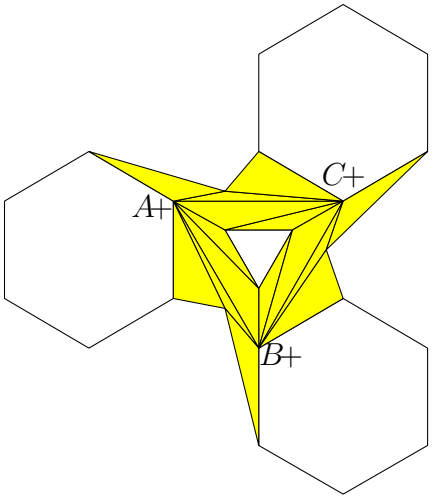


Similar to what we have done in subsection 3.6.3 on page 28 notice that in this construction of a chain there is a little bit tolerance. There was no need to draw it precisely to establish the argument. Thus we are able to stretch, shift or bend chains almost arbitrarily provided that they are long enough.

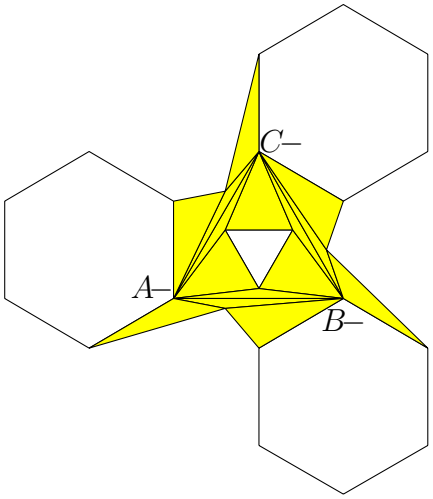
5.2.5 Branches

The following example shows how to combine three honeycombs such that they all have to be situated in the same way (i.e. either all in −-situation or all in +-situation). Again, any vertex not belonging to the honeycombs is fixed.

Here is the +-situation.



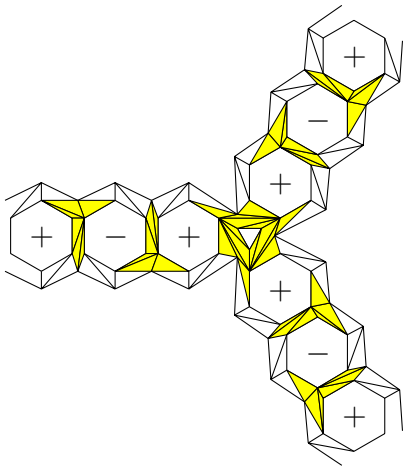
And here is the −-situation.



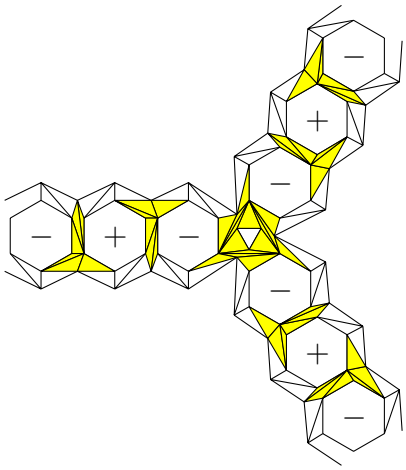
The other combinations are indeed not embeddable. To see this first note that otherwise we find a pair (X, Y) of letters out of $\{(A, B), (B, C), (C, A)\}$, where the honeycomb X is in +-situation and the honeycomb Y is in −-situation. For symmetry reasons we only have to consider the case $A+B-$. Then the line segment between $A+$ and $B-$ would collide with the central fixed triangle in the construction.

The following pictures show how to connect such a branch with chains thereby encapsulating the construction. Again, any vertex not belonging to the honeycombs is fixed.

case +

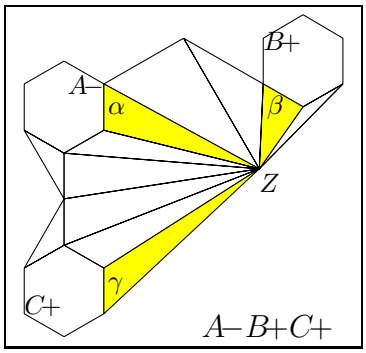


case −



5.2.6 Clauses

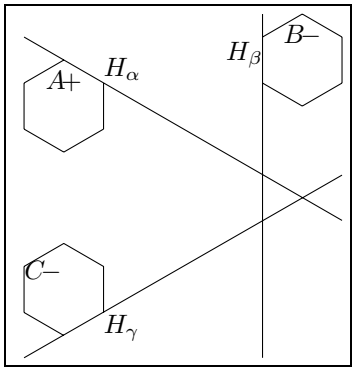
Consider the following construction.



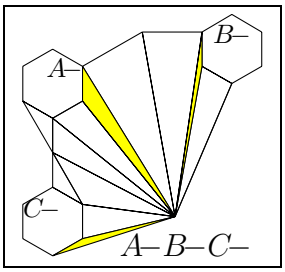
The three hexagons are honeycombs and all other vertices are fixed with one exception namely Z which orbit is a disk large enough to contain the whole picture.

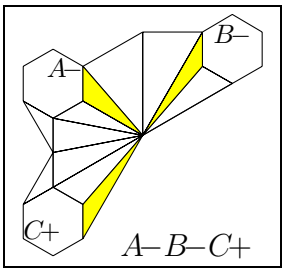
The crucial fact is that the complementary embedding (A+B–C–) is not possible no matter where Z is situated. This is because of the triangles α, β and γ. Consider for example α. If the honeycomb A were in the +-situation the vertex Z would have to be in the half-space H_α (see below).

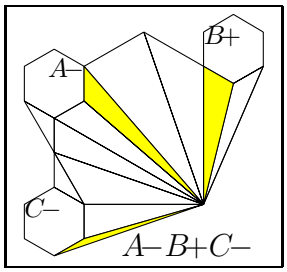
Analogously one can observe that if B were in $-$-situation the vertex Z would have to lie in the half-space H_β because of β. If then C were in $-$-situation the vertex Z would have to lie in the half-space H_γ, too. But $H_\alpha \cap H_\beta \cap H_\gamma$ is empty.

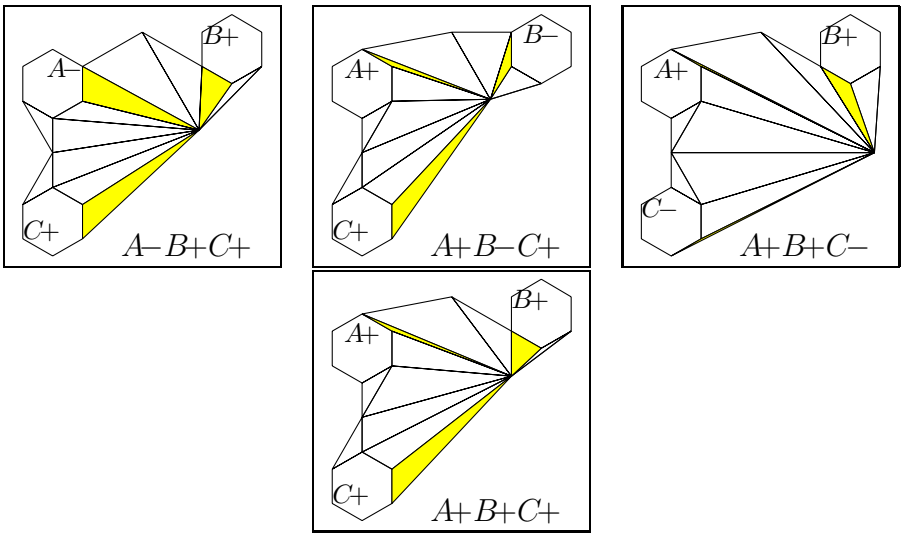


The following pictures demonstrate that all other combinations are embeddable.



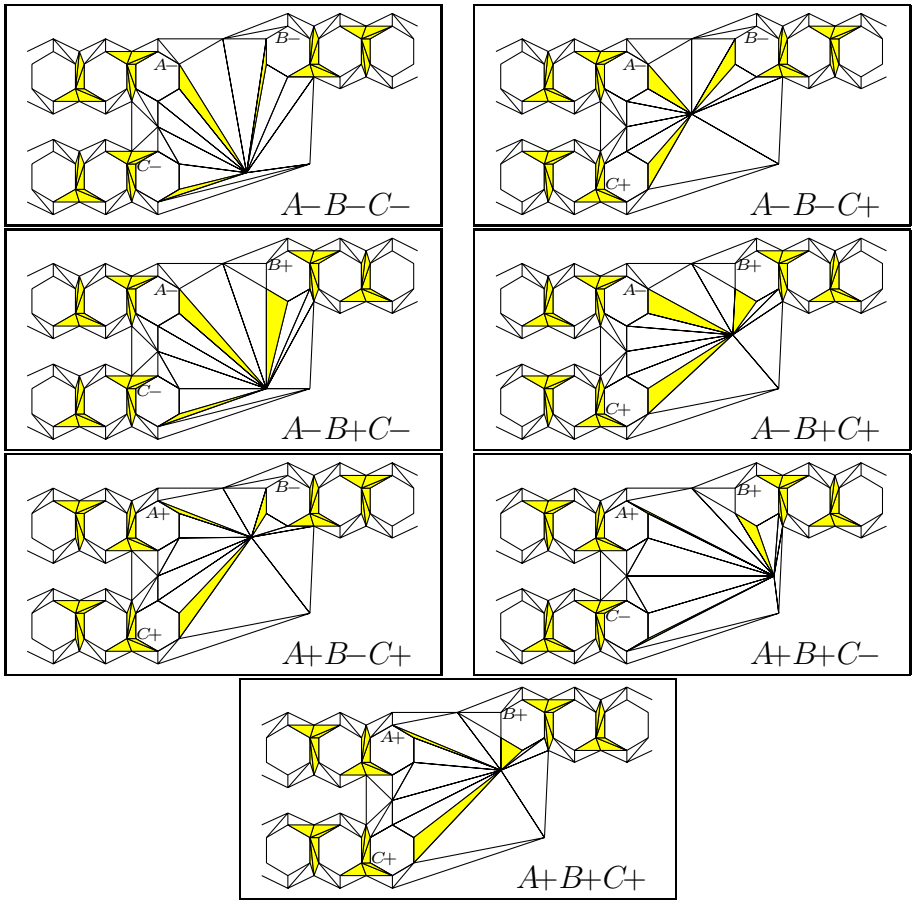






Thus we have three honeycombs connected so that all combinations of situations are embeddable except one (namely $A{+}B{-}C{-}$). This corresponds to a clause of the form $\neg x \vee y \vee z$. With one additional negation this can be converted into a clause of the form $x \vee y \vee z$.

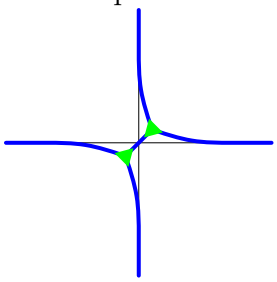
The following pictures show the seven valid combinations for an encapsulated clause with chains attached to it.



5.2.7 Components revisited

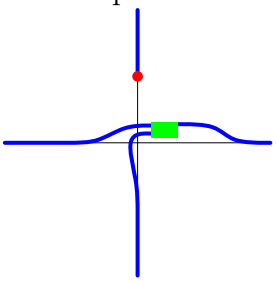
In order to construct the components characterized in subsection 3.5.3 on page 23 we scale the construction such that the honeycombs would be small enough to allow chains to consist of as many honeycombs as necessary to be flexible enough. In the following drawings a fat blue stroke indicates such a chain.

variable components



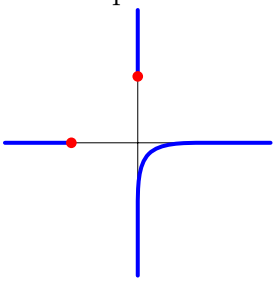
The two green triangles indicate branches (which we have discussed in section 5.2.5 on page 66).

clause components



The green rectangle indicates a clause (which we have discussed in section 5.2.6 on page 67).

connection components



The red disk indicates a blind end of a chain (which we have discussed in 5.2.4 on page 65).

The variable components are made of two branches and we obtain the 16 different types of variable components by stretching the appropriate chains thereby decreasing the number of negations by one. The connection components are made of chains only. In each clause component a clause is used as well.

Note that each component has four incomplete chains pointing outwards. It will be easy to connect them to other incomplete chains just by identifying two vertices of one incomplete chain with two vertices of the other one and by introducing nine additional edges (try to connect two incomplete endings in a picture on page 65 with a pencil to see what we mean).

The reduction works as follows. Given an instance of grid3sat. We substitute each component by the corresponding construction described above. Then we connect all incomplete chains with appropriate other incomplete chains wherever possible. Then there will remain only these incomplete chains which will point outwards of the whole construction (that is outside of the grid area). We will introduce additional blind ends and connect them to the remaining incomplete chains. So far our construction is not fully triangulated. But since all areas which are still not triangulated are bounded by fixed vertices and edges in between we can triangulate them arbitrarily without changing the way everything works.

This completes the construction. It remains to show that the formula given by an instance I of grid3sat is satisfiable if and only if there exists an embedding solving the instance J of constrained embedding.

Assume that the formula is satisfiable. Then we place all honeycombs in the branches in the variable elements in $-$-situation, if the corresponding variable is assigned to be false and in $+$-situation otherwise. The situations of the other honeycombs follow.

Assume conversely there exists an embedding solving the instance J. Then we look at the honeycombs in the branches in the variable components to get a truth assignment which fulfills the formula.

5.2.8 Remarks

If you look carefully on the constructions used in the reduction you will notice that the graph obtained from them will be tri connected. After triangulating it it will be a tri connected triangulated graph which has, by the way, a combinatorial unique embedding therefore. This is, however, not very surprising because we never used the ambiguity between combinatorial different embeddings in our proof (reread subsection 5.2.2 on page 62 to see what we mean).

Hence we can strengthen our result by stating that constrained embedding remains NP-hard even if we restrict the graphs to be tri connected. So we actually have an NP-hardness result for the following problem.

constrained embedding'

> We are given a labeled triangulated tri connected oriented planar graph, each vertex of which is assigned to a closed disk in the plane given by a center point of rational coordinates and a rational radius. Decide whether it is possible to embed the graph positively oriented in the plane with line segments as edges such that each vertex lies in its disk.

5.3 Open problems

It is not known whether constrained embedding is in NP. On the first sight constrained embedding is a special case of quadratic programming which is known to be NP-hard but which is not known to be in NP.

An approach to solve the problem would be to regard it as a quadratic programming problem. Thus we have just proved that quadratic programming is NP-hard. Of course, it is already known that quadratic programming is NP-hard and the proof is much simpler than the one obtained from what we have done so far here. Nevertheless it is still not known whether quadratic programming is in NP, as stated in [GJ79].