← Traveling salesman Knight in a War Grid →

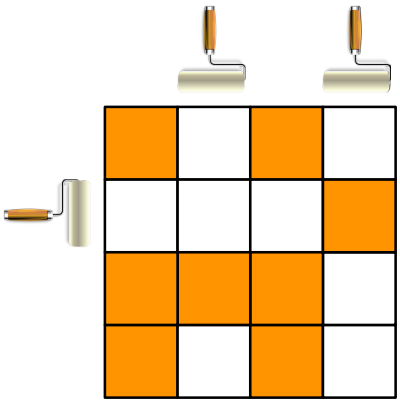
Bipartite minimum vertex cover

Aug 5, 2016 • matching

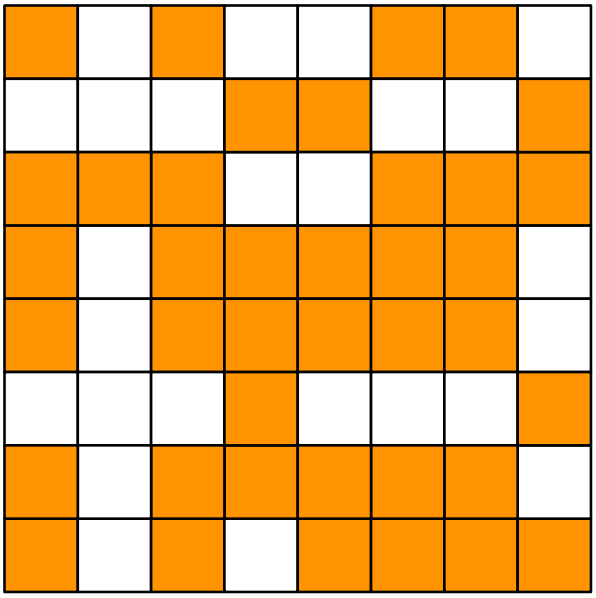
Given a bipartite graph $G(U, V, E)$ find a vertex set $S \subseteq U \cup V$ of minimum size that covers all edges, i.e. every edge has at least one endpoint in S.

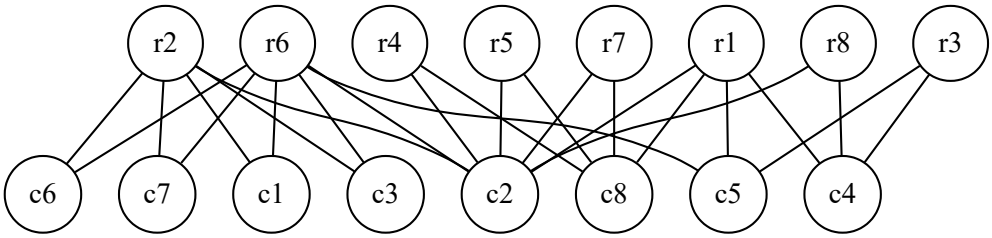
A motivation

Suppose we are given a grid with black and white cells. (In the illustrations above, black is replaced by orange). We want to find a minimum cardinality set of lines (which is either a row or a column) such that if we color all those lines in black then we obtain an all black grid. In other words every white cell belongs to a selected line.



This problem can be viewed as finding a *minimum cardinality vertex cover* of a *bipartite graph*. Here is the explanation. Interpret the given grid as the adjacency matrix of a bipartite graph. There will be a vertex for every row and a vertex for every column. For every white cell there is an edge between the corresponding row and column vertices. Such a graph is called *bipartite* because there are two types of vertices and edges exist only between vertices of different types.



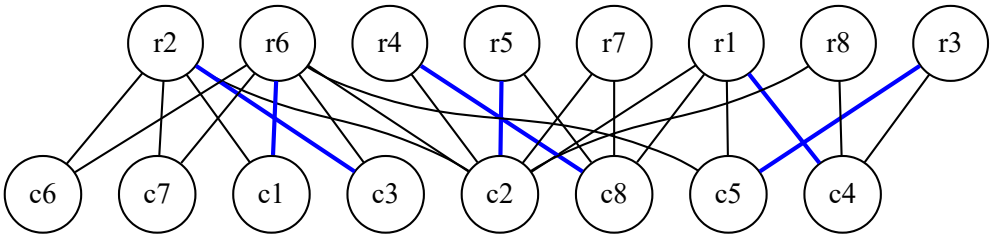


A vertex cover is a set of vertices such that every edge has at least one endpoint in the set. In that sense the set is covering all edges. Finding the minimum cardinality vertex cover is a hard problem, but for bipartite graphs it is easy to solve as we will see now.

Kőnig's theorem

Given a bipartite graph $G(U, V, E)$ find a vertex set $S \subseteq U \cup V$ of minimum size that covers all edges, i.e. every edge (u,v) has at least one endpoint in S, that is $u \in S$ or $v \in S$ or both.

Let M be a maximum cardinality matching in G. A *matching* is an edge set $M \subseteq E$ such that no two edges of M have the same endpoint.

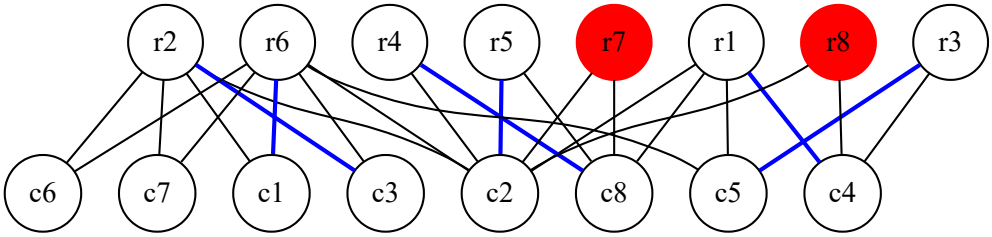


Since every edge of a matching must have at least one endpoint on S, the size of a maximum matching M is a lower bound on the size of the minimum cardinality vertex

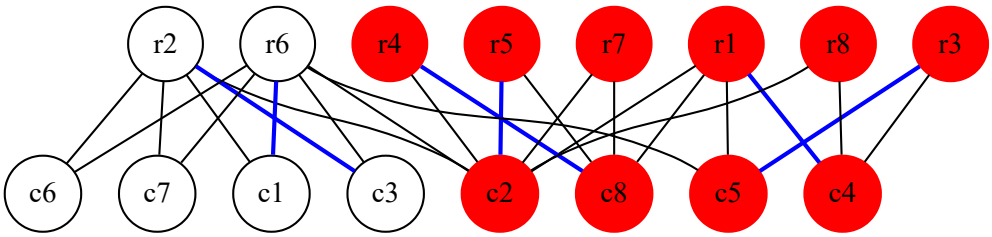
cover. Kőnig's theorem states that these optimal cardinalities are in fact equal.

The proof of the theorem is constructive and provides an algorithm to compute the minimum vertex cover given a maximum cardinality matching M for the graph $G(U, V, E)$.

Initially let Z be the set of non matched vertices in U. In the figures the row vertices represent U and the column vertices represent V.

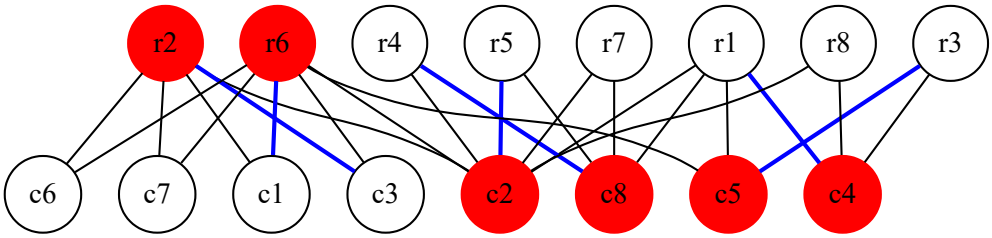


Now add to Z all vertices reachable by some vertex from Z via an alternating path. A path is called *alternating* if it alternates between edges from the matching and edges which are not in the matching.



Define the set

$$S = (U \setminus Z) \cup (V \cap Z).$$

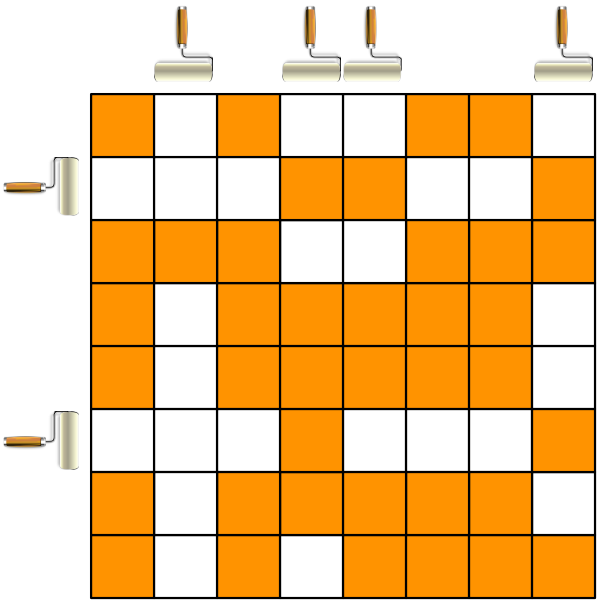


The construction of Z implies that for every edge $(u, v) \in M$, if $v \in Z$ then $u \in Z$ as well. And if $u \in Z$, since u was not initially in Z among the non matched vertices in U, it must be that u was added to Z by following some matched edge, which means that $v \in Z$ as well.

This implies that for every edge $(u, v) \in M$ of the matching either both endpoints are in Z or none. Hence every edge from the matching has exactly one endpoint in S, and

$|S| \geq |M|$.

Now we show that S covers all edges of the graph. Let (u, v) an arbitrary edge. If $u \notin Z$, then the edge is covered. From now on assume If $u \in Z$. If (u, v) is not an edge from the matching then by maximality of Z the vertex v has to be in Z, and hence $v \in S$. But if (u, v) is an edge from the matching, then by the previous observation v belongs to Z and hence to S as well. This shows that S is a vertex cover, which implies by our first observation $|S| \leq |M|$, from which we can conclude $|S| = |M|$.



The algorithm

Build a maximum cardinality matching M. Build maximum alternating forest starting from non matched vertices in U. From this define the vertex cover set S.

Building the matching is the bottleneck in the algorithm, the rest takes only linear time.

The implementation

```
from tryalgo.bipartite_matching import max_bipartite_matching


def alternate(u, bigraph, visitU, visitV, matchV):
    """extend alternating tree from free vertex u.
      visitU, visitV marks all vertices covered by the tree.
    """
    visitU[u] = True
    for v in bigraph[u]:
        if not visitV[v]:
```

```
            visitV[v] = True
            assert matchV[v] is not None   # otherwise match not maximum
            alternate(matchV[v], bigraph, visitU, visitV, matchV)


def koenig(bigraph):
    """Bipartie minimum vertex cover by Koenig's theorem

    :param bigraph: adjacency list, index = vertex in U,
                                    value = neighbor list in V
    :assumption: U = V = {0, 1, 2, ..., n - 1} for n = len(bigraph)
    :returns: boolean table for U, boolean table for V
    :comment: selected vertices form a minimum vertex cover,
              i.e. every edge is adjacent to at least one selected vertex
              and number of selected vertices is minimum
    :complexity: `O(|V|*|E|)`
    """
    V = range(len(bigraph))
    matchV = max_bipartite_matching(bigraph)
    matchU = [None for u in V]
    for v in V:                         # -- build the mapping from U to V
        if matchV[v] is not None:
            matchU[matchV[v]] = v
    visitU = [False for u in V]         # -- build max alternating forest
    visitV = [False for v in V]
    for u in V:
        if matchU[u] is None:           # -- starting with free vertices in U
            alternate(u, bigraph, visitU, visitV, matchV)
    inverse = [not b for b in visitU]
    return (inverse, visitV)
```

2 Comments

Type Comment Here (at least 3 chars)

Name (optional) | E-mail (optional) | Website (optional) | Preview | Submit

Anonymous • 5 years ago

can you help to translate in java?

-39 | **Reply**

Louis • 4 years ago

I just made the link with an old problem I solved:
http://www.acm.uestc.edu.cn/#/problem/show/79

The modelization as a bipartite graph is the same, but the fact that a cell can be made dirty again totally changes the algorithm and the problem becomes linear.

1 | **Reply**