Title: Eulerian cycles for combinatorial independent sets determination Slug: adhocindependentsets-euleriancycles Date: 2020-06-12 19:30 Category: Math Tags: graph theory, independent sets, algorithm, complexity, combinatorial, eulerian cycles Author: Carolin Zöbelein Summary: In the last post, we described a naive algorithm to determine independent sets and got a terrible complexity. Now, we want to grab this algorithm to use it as a base for an alternative way of independent set generation.

This post is the second part of a series regarding combinatorial independent sets determination.

- First post of this series: Ad hoc method for independent sets

In the last post, we described a naive algorithm to determine independent sets and got a terrible complexity. Now, we want to grab this algorithm to use it as a base for an alternative way of independent set generation. Instead of putting edges in an arbitrary order into our algorithm, we will consider properties of the set of edges E, given by Eulerian cycles.

Eulerian cycle expansions

For improving our algorithm, we will need Eulerian cycles.

Definition 1 (Eulerian cycle)
An Eulerian cycle in an undirected graph is a cycle that uses each edge exactly once. A graph which owns such a cycle is called Eulerian.

If G is a connected graph, then the following statements are equivalent:

- *G is Eulerian*
- *Every vertex $v \in V$ of G have an even degree* $\deg(v)$
- *The set E of G is the set union of all edges of pairwise disjunct cycles*

This was proofed by Hierholzer [1].

So, what we need is a connected graph which only has vertices with even degree. Because, of course, not all graphs statisfy this conditions by default, we will now think, how we can map a given graph G to a graph G' which statisfies the conditions, but still gives us the searched independent sets of G, without increasing noticeable the full algorithm complexity, finally.

Connectivity

At first, let's check if the graph G is connected. This can be done by breadth-first search with $\mathcal{O}(|V| + |E|)$. If yes, we don't have to do anything anymore regarding connectivity. In the case of no, we have several independent, pairwise disconnected, subgraphs $G_0, G_1, ...$ which we can handle without much work. We determine the independent sets S_{G_i} for each G_i separately. After determining all

independet sets, we finally have simply to combine all of this S_{G_i} with each other. To find all strongly components of a graph we can use Tarjan's algorithm [2] which also works with $\mathcal{O}(|V| + |E|)$. Later, we will see that the final independent sets can be easily derived from our partial results.

Even degrees

The second condition for an Eulerian graph is regarding even degrees. At first, let's check the degree of each vertex in G. This is a trivial thing with linear complexity. If yes, we don't have to do anything anymore regarding even degrees. In the case of no, we have to do some modification of G. From basic graph theory we know that for every graph G we have $2|E| = \sum_{v_i \in V} \deg(v_i)$. Hence, we know that we always have an even number of vertices with odd degree. With this we can simply map our graph G to an Eulerian graph by connecting always two of our odd degree vertices $\{u, v\}$ by one new introduced helper vertex $h \in H$ (be H the set of helper vertices) and hence we get the two new edges (u, h) and (h, v). So, what will be the consequences of adding an helper vertex to our graph, regarding the independent sets? We have two possible cases here:

- The helper vertex h connects two vertices u and v which belong to the same independent set S which means $u \in S \wedge v \in S$ and hence $(u, v) \notin E$.
  Consequence: Since u and v are still not directly connected with each other, they will still belong to the same independent set. Since, each of them have a connection to h, h will become a member of an other, second, independent set. So, we will get our original independent set S unchanged and any other second independent set with $S' \cup \{h\}$.
- The helper vertex h connects two vertices u and v which belong to two different independent sets S_i, S_j with $S_i \not\subseteq S_j$, which means $u \in S_i \wedge v \in S_j$ and hence $(u, v) \in E$.
  Consequence: Since u and v are already directly connected, also with a connection to h they will still both belong to two different sets. Since, each of them have a connection to h, h will become a member of an other, third, independent set. So, we will get our original two independent sets S_i and S_j unchanged and any other third independent set with $S_k \cup \{h\}$.

We will come back to this, later.

Eulerian cycle partitioning

After mapping our graph G to an Eulerian one, we determine the Eulerian cycles with Hierholzer's algorithm [1] and complexity $\mathcal{O}(|E|)$. Like already mentioned, a graph G can be disconnected into several subgraphs $G_0, G_1, ...$ For each of them we have to identify the Euler cycles. The outcome of this can be written in a general way like as the following example:

$$\underbrace{\left(\overbrace{\underset{0}{a_1},\underset{1}{b_1},\underset{2}{c_1}}^{C_0},\overbrace{\underset{3}{a_2},\underset{4}{d_1},\underset{5}{e_1},\underset{}{f_1}}^{C_1},\overbrace{\underset{}{\widehat{b_2}}}^{C_2},\overbrace{\underset{}{\widehat{e_2}}}^{C_3},\overbrace{\underset{}{\widehat{a_3}}}^{C_4}\right)}_{G_0}\underbrace{\left(\overbrace{\underset{0}{g_1},\underset{1}{h_1},\underset{2}{i_1}}^{C_0'},\overbrace{\underset{3}{\widehat{g_2}}}^{C_1'}\right)}_{G_1}\cdots \tag{1}$$

Here, $a_i, b_i, \cdots, h_i, i_i$ are vertices of V, their indices are the numbers of their appeareance and the numbers below them are the numbering of the first appeareance order within the Eulerian graph. In each graph we can find inner cycles C_i which are recognized through an additional appeareance of an already in the sequence existing vertex. For our further steps, we will see, that it is useful to cut a graph sequence exactly at this points.

What we want to do now, is to put this information of one graph G_i in a more helpful data structure. For this we use a list of linked lists for each G_i in which we save for each cycle its size s, its first element and the the position of this first element within the sequence of all first appearences of numbers of the whole Eulerian sequence, (see table 1), at all.

Table 1. Lists L_0 and L_1 of linked lists. On the left generated for graph G_0 and on the right for G_1.

Cycle, Size → First element → First app.
C_0, $s_0 \to$ a $\to 0$
C_1, $s_1 \to$ a $\to 0$
C_2, $s_2 \to$ b $\to 1$
C_3, $s_3 \to$ e $\to 4$
C_4, $s_4 \to$ a $\to 0$

Cycle, Size → First element → First app.
C_0', $s_0' \to$ g $\to 0$
C_1', $s_1' \to$ g $\to 0$

Figure 1: GitHub Logo

We know the maximum number of edges of a graph G is $\frac{1}{2}\left(|V|^2 - |V|\right)$ and that we have this value plus one for entries of G, like in (1). So we have a total complexity of $\mathcal{O}\left(\left(\frac{1}{2}\left(|V|^2 - |V|\right) + 1\right)^2\right) = \mathcal{O}\left(|V|^4\right)$ for generating all lists of linked lists of a given G, like that.

Outlook

In the next post, we will use this information given by the eulerian cycle to define an appropriate representation for a combinatorial approach for determination of independent sets.

References

- [1] Carl Hierholzer and Chr Wiener, *Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren*, Mathematische Annalen **6** (1873), no. 1, 30–32.

- [2] Robert Tarjan, *Depth-first search and linear graph algorithms*, SIAM journal on computing **1** (1972), no. 2, 146–160.