Carolin Zöbelein

Table of contents

	pages
Overwiew	1 - 3
Eulerian cycles (and helper vertices)	4 - 6
Eulerian Sequences S_E	7 - 9
Maximum number of 1s of coupling gaps	10 - 14
Representation matrix $\underline{M}$ (and $\underline{M}'$)	15 - 18
Analysing of $\underline{M}'$	19 - 20
Higher degree case (degree > 2) for substrings	21 - 23
Construction of $\underline{M}^{2n}$	24 - 25
Summary and possible approaches	26 - 28
Comparison with Tarjan & Trojanowski	29 - 35
Comparison with Tarjan's Decomposition by Clique Separators	36 - 43
Two examples	43 - 45
Résumé	46 - 47
Additional References	48 - 49

Identification of (maximum) independent sets with the help of Eulerian cycles

Given an arbitrary undirected graph $G=(V,E)$.

→ We want to identify the set of maximum independent sets with the help of Eulerian cycles.

We start with a very raw sketch of what we will do.

Sketch:

0. Given an undirected graph $G=(V,E)$ with adjacence matrix A
1. We check if G is connected with breadth-first search with time complexity $\mathcal{O}(|V|+|E|)$. ⊛

   If G has pairwise disconnected subgraphs $G_0, G_1, G_2, \ldots$, we have to run the algorithm for each of them separately and finally unite the maximum solutions.

   To keep things simple, we will assume that we will always have connected graphs G at the beginning and also in each substep of this algorithm, for the moment.

   ⊛ For the identification of strongly components of G, we can use depth-first search with $\mathcal{O}(|V|+|E|)$ or a variation of Tarjan's algorithm. [Tarjan].

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

---

2. We check if G satisfies the characteristics for being Eulerian.

   → If yes: well, nothing to do for this.

   → If no: we will make G Eulerian, by introducing so called "helper vertices", and will discuss how this vertices have to be handled in further algorithm steps. (see later.)

3. Assume G is Eulerian.

We determine the Eulerian cycle of G with the help of Hierholzer's algorithm [Hierholzer] with $\mathcal{O}(|E|)$ [Graphs].

[Hierholzer] 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

[Graphs] Herbert Fleischner, "Eulerian graphs and related topics", vol. 1, Elsevier, 1990

---

4. From Hierholzer's algorithm we get a vertices sequences S_E which represents an Eulerian cycle of G.

→ We identify the vertices V of G in S_E with their belonging degrees $deg(v_i)$, $v_i \in V$.

→ We identify all vertices of V_0 with the smallest degree deg_{min} within our Eulerian sequence S_E and the number of direct consecutive vertices each of them following the same degree the order of S_E (see later). for this smallest degree.

→ For a given degree deg_{min} of A and the set of number of direct consecutive vertices of the same degree, we can derive a boolean function $f_{deg_{min}}$ which describes a maximization problem depending on the set of choosen vertices for getting maximum independent sets (see later.)

5. We will see that the problem statement of the maximization of $f_{deg_{min}}$, leads again to a maximum independent set problem of a new graph G^1 reduced by the vertices of smallest degree of the original graph G. (see later.)

6. ... We do this steps as long until we reach the case of getting a graph G^n which vertices have all the same degree.

We will see, that we can then derive the solution of the last final step, by an equation. (see later.)

7. We can now take the maximization solutions of G^n, to get the maximum solutions of G^{n-1}. This solutions gives us the maximum solutions of G^{n-2} and so on... until we finally get the maximum independent set solutions of G (see later.)

→ After this short sketch, we want to give a more detailed description, now.

→ We will allways assume that G' is connected without mention it anymore.

→ We start with a condsideration of Eulerian characteristic and the introduction of helper vertices.

Eulerian cycles (and helper vertices)

At first, what are Eulerian cycles and when is G Eulerian, at all?

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:

1. G is Eulerian
2. Every vertex $v_i \in V$ of G have an even degree $\deg(v_i)$
3. The set E of G is the set union of all edges of pairwise disjunct cycles.

This was proofed by Hierholzer [Hierholzer].

We will use item 2 to check if G is Eulerian and if not, to map G to an Eulerian one G'.

→ So we check for the degrees of each vertex of G.

This can be done with $\mathcal{O}(|V|^2)$ complexity. by counting all '1's in the row of A_I representing a specific vertex of G, for each vertex $v_i \in V$.

(*) → For further steps we will still need this degree information, so we have to save it, for the moment.

→ If all $\deg(v)$ are even, G is Eulerian and we don't have to do anything anymore.

→ If not all $\deg(v)$ are even, then we have to change G in such a way that it get Eulerian.

→ 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 G to an Eulerian graph G' by connecting always two of our odd degree vertices $\{u, v\}$, $u, v \in V$, by one new introduced helper vertex $h \in H$ (be H the set of helper vertices) and hence we get two new edges (u, h) and (h, v).

→ We do (*) (it's more correct to do it just here and not already at the point above)

The nature of helper vertices

1. Helper vertices h always have $\deg(h) = 2$
2. Possible consequences of adding an helper vertex to our graph, regarding the independent sets question?

We have two possible cases here:

a) 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. So, we will get our original independent set S unchanged and any other second independent set with $S' \cup \{h\}$.

b) The helper vertex h connects two vertices u and v which belong to two different independent sets S_i, S_j with $S_i \notin S_j$, 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 another, 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 $S_k \cup \{h\}$.

3. It's not hard to see that the maximum number of helper vertices which we have to add for getting an Eulerian graph is given by $\lfloor \frac{|V|}{2} \rfloor$.

→ Later, we will see how helper vertices have to be handled during our algorithm

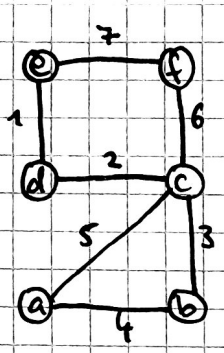
Eulerian Sequences S_E

→ We assume that G is Eulerian.

→ Using Herholzer's algorithm, we get an Eulerian sequence S_E which looks like the following example:

Example 1 (Eulerian Sequence S_E)

Figure 1.



	Couplings C									
	e	d	c	b	a	c	f	e	← S_E	1.
deg(v_i)	2	2	4	2	2	4	2	2		2.
S_E^{N-1}	0	1	0	0	1	0	1	0	{a, d, f}	3.
S_E^{N-1}	0	1	0	1	0	0	1	0	{b, d, f}	4.

→ Line 1 gives us the Eulerian Sequence S_E of G in Figure 1.

~~The~~ We define our S_E in such a way, that the most left vertex (starting point of S_E) and the most right vertex (ending point of S_E), always have to be the same vertex (here: e)

→ In S_E, we know that there exists an edge between directly ~~consec~~ consecutive vertices.

→ If we identify a vertex which is element of a set by '1' and a vertex which doesn't belong to a set by '0', we can represent ~~one~~ an independent set of G by ~~an~~ a bit string (*) of '0's and '1's in the following way:

1. Direct consecutive bits $b_i b_{i-1}$ are allowed to have the bit patterns 00, 01 and 10. The bit pattern 11 is forbidden, since this would mean that ~~two~~ two vertices with $u, v \in V$ ~~for~~ $(u,v) \in E$ would be in the same ~~set~~ set and this is not what we want for independent sets.

(*) $S_E^{N-1} = b_{N-1} b_{N-2} \dots b_2 b_1 b_0$

2. A coupling c of bits of S_E^{N-1} be given by

$c_{\alpha i=1,\dots,j} = \{(b_i,\dots,b_j) \mid b_i = \dots = b_j, \forall i,j : i \neq j \wedge j \neq i+1\}$ and $|c_{\alpha i=1,\dots,j}| \geq 2$.

For all couplings c_α and c_β we have $c_\alpha \cap c_\beta = \emptyset$.

We will denote all bits b which belong to the same coupling c by b^c. S_E^{N-1} consists of at least one coupling of bits which is $c_{N-1,0} = (b_{N-1}, b_0)$.

→ If we choose our bit values in the right way, which means so that we get the maximum number of 1s if we add the value of each vertex, then we will have found the maximum independent sets. In line 3 and 4, we added the solutions of ~~maximum independent sets~~ a maximized bit string S_E^{N-1} which represents the maximum independent sets $\{a,d,f\}$ and $\{b,d,f\}$, for our given example.

Degrees and Coupling gaps in S_E

→ Assume we saved the degrees of all $v_i \in V$, after adding of helper vertices, in a list D with $v_i \to \deg(v_i)$

→ If we have S_E^{N-1} given, we need to know the particular bits with their belonging degrees.

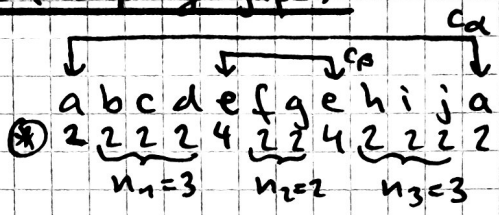
For this, we go through S_E^{N-1} from left (b_{N-1}) to right (b_0). For each bit b_i, we look in $D(v_i)$ for $\deg(v_i)$ with $b_i \to v_i$ and hence generate a list of degrees according to the Eulerian sequence S_E (see line 2 in example 1).

The time complexity for this is given by the number of edges of G, hence $\mathcal{O}(|E|)$.

We will denote this generated list by $D_{S_E}(i)$ (gives degree at position i of bit string)

→ Later, we will also need the number of direct consecutive vertices n_i of the same degree (without the special coupling (b_{N-1}, b_0)) according S_E. For example:

Example 2 (coupling gaps)



$D(v_i) = \deg(v_i)$
$b_i \to v_i$
$\Rightarrow D_{S_E}(i)$ ⊛

If we have $D_{S_E}(i)$, we get all n_i for a fixed choosen minimum degree (in this example $\deg_{min} = 2$) by going from left (b_{N-1}) to right (b_0) through $D_{S_E}(i)$. We keep the first degree value of coupling bit in mind and go to right until we get a next coupling bit. Because of the bit index difference we know then n_i between two coupling gap bits.

In this example: Coupling gaps: (a, e) $n_1 = 3$ (coupling bits)
(e, e) $n_2 = 2$ (the bits between two coupling bits)
(e, a) $n_3 = 3$

The time complexity for this, is given by the number of edges of G, so $\mathcal{O}(|E|)$.

Maximum number of 1s of coupling gaps

→ Like we already mentioned, we are particularly interested in coupling gaps (the number of direct consecutive vertices with the same degree in S_E between two coupling bits) (See also example 2.).

→ Under consideration of our definition of bit strings S_E^{N-1} (see example 1, line 3 and 4) with the allowed bit patterns 00, 01 and 10, but the prohibit pattern 11, we now asking for the maximum number of 1s between two coupling bits, means n_i for coupling gaps.

→ We know that G is Eulerian, so we will always have even degrees.

→ For the moment, we will ignore that some of the vertices could be helper vertices. We will come back to this later.

→ So, we will now analyse the maximum number of ones, denoted by N_1, for a particular degree d.

Maximum number of 1s for $d_{min}=2$

→ we want to start with analysing the maximum number of 1s for $d_{min}=2$ (d_{min} is the smallest degree of vertices $v \in V$ of G)

1. Case: Coupling gap bits (0,0)

n even

n=0 0|0 0

n=2 0|1 0|0 1
0|0 1|0 1

n=4 0|1 0 1 0|0 2
0|0 1 0 1|0 2

n=6 0|1 0 1 0 1 0|0 3
0|0 1 0 1 0 1|0 3

⋮

⇒ $\frac{1}{2}n$

n odd

n=1 0|1|0 1

n=3 0|1 0 1|0 2

n=5 0|1 0 1 0 1|0 3

n=7 0|1 0 1 0 1 0 1|0 4

⋮

⇒ $\frac{1}{2}(n+1)$

2. Case: Coupling gap bits (0,1)

n even

n=0 0|1 0

n=2 0|1 0|1 1

n=4 0|1 0 1 0|1 2

n=6 0|1 0 1 0 1 0|1 3

⋮

⇒ $\frac{1}{2}n$

n odd

n=1 0|0|1 0

n=3 0|0 1 0|1 1
0|1 0 0|1 1

n=5 0|0 1 0 1 0|1 2
0|1 0 1 0 0|1 2

n=7 0|0 1 0 1 0 1 0|1 3
0|1 0 1 0 1 0 0|1 3

⋮

⇒ $\frac{1}{2}(n-1)$

3. Case: Coupling gap bits (1,0)

→ This case is symmetric to the case (0,1)

n even

⇒ $\frac{1}{2}n$

n odd

⇒ $\frac{1}{2}(n-1)$

4. Case: Coupling g&b bits (1,1)

n even

$n=0$ 1|1

Exception because of pattern prob prohibition of 11

$n=2$ 1|0|0|1 0

$n=4$ 1|0 1 0|0|1 1
1|0 0 1 0|1 1

$n=6$ 1|0 1 0 1 0|0|1 2
1|0 0 1 0 1 0|1 2

⋮

$\Rightarrow \boxed{\frac{1}{2}(n-2)}$

n odd

$n=1$ 1|0|1 0

$n=3$ 1|0 1 0|1 1

$n=5$ 1|0 1 0 1 0|1 2

$n=7$ 1|0 1 0 1 0 1 0|1 3

⋮

$\Rightarrow \boxed{\frac{1}{2}(n-1)}$

Summary for a given bit string (Eulertan) $S_E^{N-1} := b_{N-1} b_{N-2} \cdots b_2 b_1 b_0$

$\Rightarrow$ **If n even:** $\frac{1}{2}n - b_{q+(n-1)} \wedge b_q$ (eq.1)

$\Rightarrow$ **If n odd:** $\frac{1}{2}(n-1) + \bar{b}_{q+(n-1)} \wedge \bar{b}_q$ (eq.2)

The number 1s of a bit string S_E^{N-1} is given by (for considering $d_{min}=2$)

$$N_1 := \sum_{\forall n_i \text{ even}} \frac{1}{2}n_i - b_{q_i+(n_i-1)} \wedge b_{q_i} + \sum_{\forall n_j \text{ odd}} \frac{1}{2}(n_j-1) + \bar{b}_{q_j+(n_j-1)} \wedge \bar{b}_{q_j} + \sum_{\forall c_\alpha \in \mathcal{C}} V(C_\alpha) \quad \text{(eq.3)}$$

with the couplings (eq.4)

$$C_\alpha := \{ b_i = \cdots = b_j \mid \forall (b_1, \ldots, b_j) \in c_\alpha \}, \ \forall c_\alpha \in \mathcal{C}$$

and $b_i \wedge b_{i+1} \neq 1 \wedge 1$ $, \forall b_i, b_{i+1} \in S_E^{N-1}$

$V(C_\alpha)$ gives us the value of C_α which is $b^{C_\alpha} := b_i = \cdots = b_j$.

→ Since $n_i = \text{const}$, we can also write:

$$N_1^* := N_1 - \sum_{\forall n_i \text{even}} \frac{1}{2} n_i - \sum_{\forall n_i \text{odd}} \frac{1}{2}(n_j - 1)$$

$$= \underbrace{-\sum_{\forall n_i \text{even}} b_{q_i + (n_i - 1)} \wedge b_{q_i}}_{(*^1)} + \underbrace{\sum_{\forall n_i \text{odd}} b_{q_j + (n_j - 1)} \wedge b_{q_j}}_{(*^2)} + \underbrace{\sum_{\forall c_\alpha \in \mathcal{C}} V(c_\alpha)}_{(*^3)} \quad \text{(eq 5.)}$$

coupling bits

→ For given coupling bits the values of coupling gap bits is unique given, so we don't consider this coupling gap bits anymore.

Discussion of equation 5:

→ If we set all coupling bits to 0, we already get a relatively good result. We call this case the default case, and looking for possible improvements regarding this case.

→ Consider the case of having at first all coupling bits are 0, and the changing one c_α from 0 to 1, in such a way that coupling bit move from (1,0) to (0,1) or (1,0)

⇒ we stay with the same number of coupling gap bits, if all all coupling bits which are influenced by this changing are even, plus the one 1 we introduced by the coupling bit itself (equation part $*^3$)

⇒ The number of coupling gap bits of value 1, gets reduced by one if we have exactly one odd case, plus the one 1 we introduced by the coupling bit itself, so we stay with the same number of 1s like before.

⇒ If we have more than one odd case, the number of 1s of coupling gap bits get reduced by

a number larger than one, so even plus the one 1 we introduced by the coupling bits, we still get a decrease of the number of 1s.

→ Consider the case of having the case (0,1) or (1,0) for coupling gaps and the change to (-1,1).

⇒ We stay with the same number of bits within coupling gaps if all coupling gaps are odd, plus the 1 we introduced by the coupling bits itself, so we get an improvement by 1.

⇒ The number of coupling gap bits of 1 get reduced by one if we have exactly one even case, plus the one 1 we introduced.

⇒ If we have more than one even case, the number of 1s of the coupling gaps bits get reduced by a number larger than one, so even plus the one 1 we introduced by the coupling bits, we still get a decrease of the number of 1s.

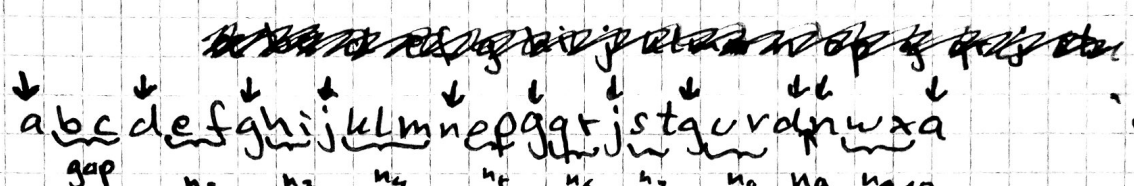
→ We know the concatenation of coupling bits is commutative and associative, since it's trivial to see that it doesn't matter in which order we add coupling bits for the final result.

⇒ Because of commutativity and associativity of coupling bits, we can conclude that if we want to find this coupling bit concatenation which maximized the number of 1s in a bit string S_E^{N-1}, (coupling bits) we have to look for all couplings C_k which improve the number of 1s by coupling to each other pairwise.

→ We can now, put this together to a new matrix.

Example 3 (Representation matrix, reduced by degree 2)

Given is the Eulerian bit string S_E^{N-1}:



→ We count for each coupling bit the number of even and the number of odd couplings $\mu := (\mu^+, \mu^-)$

↑ number for even ↑ odd couplings

$$\underline{\mu} := \begin{array}{c} \begin{matrix} a & d & j & g & n \end{matrix} \\ \begin{pmatrix} (0,0) & (1,0) & (0,1) & (0,1) & (1,0) \\ (1,0) & (0,0) & (0,1) & (2,0) & (1,0) \\ (0,1) & (0,1) & (0,0) & (3,0) & (0,1) \\ (0,1) & (2,0) & (3,0) & (0,0) & (1,0) \\ (1,0) & (1,0) & (0,1) & (1,0) & (0,0) \end{pmatrix} \end{array} \begin{matrix} a \\ d \\ j \\ g \\ n \end{matrix}$$

→ We consider the case: $(0,1) \to (0,1)$ or $(1,0)$

Means: We change one bit to 1 and stay the other on 0

$$\underline{X} := \begin{array}{c|ccccc} & a & d & j & g & n \\ \hline a & 0 & 0 & 0 & 0 & 0 \\ d & 0 & 0 & 0 & 0 & 0 \\ j & 0 & 0 & 0 & 0 & -1 \\ g & 0 & 0 & 0 & 0 & 0 \\ n & 0 & 0 & -1 & 0 & 0 \end{array}$$

for explanation, see discussion of equation 5 before

Derived from μ^+ values

→ We consider the case: $(0,1)$ or $(1,0) \to (1,1)$

Means: We already set one bit to 1, now we set an additional second bit to 1

$$\underline{Y} := \begin{array}{c|ccccc} & a & d & j & g & n \\ \hline a & 0 & -1 & 0 & 0 & -1 \\ d & -1 & 0 & 0 & -2 & 0 \\ j & 0 & 0 & 0 & -3 & 0 \\ g & 0 & -2 & -3 & 0 & -1 \\ n & -1 & 0 & 0 & -1 & 0 \end{array}$$

Exception case, since $n_g = 0$ and $(1,1)$ (d, n)

for explanation, see discussion of equation 5 before

Derived from μ^+ values

→ With this, if we combine $\underline{X}$ and $\underline{Y}$, we get the result of the concatenation of two bits. 16

$M_i =$

	a	d	j	g	n	
	0	-1	$\frac{1}{m}(-1)$	0	$-1+\frac{1}{m}(-1)$	a
	-1	0	$\frac{1}{m}(-1)$	-2	$-1+\frac{1}{m}(-1)$	d
	$\frac{1}{m}(-1)$	$\frac{1}{m}(-1)$	$\frac{1}{m}(-1)$	$-3+\frac{1}{m}(-1)$	0	j
	0	-2	$-3+\frac{1}{m}(-1)$	0	$-1+\frac{1}{m}(-1)$	g
	$-1+\frac{1}{m}(-1)$	$-1+\frac{1}{m}(-1)$	0	$-1+\frac{1}{m}(-1)$	$\frac{1}{m}(-1)$	n

(*) Since, we have

d_j the case $n_g = 0$

$d = 1 = n$ is prohibit.

→ Because of $n_g = 0$, we would need 0 instead of -1, since our degree = 2 equation doesn't work in this case

→ But since '11' pattern is prohibit, even so we take -1, to make the prohibit case in this way (maybe a number < -1 would be even better, we stay with this, for the moment)

Example entry calculations:

$$M_{jg} = Y_{jg} + \frac{1}{m}\{X_{jn} + X_{ij} + X_{jd} + X_{ja} + X_{gn} + X_{gg} + X_{gd} + X_{ga}\}$$

$$= -3 + \frac{1}{m}\{-1 + 0 + 0 + 0 + 0 + 0 + 0 + 0\}$$

$$= -3 + \frac{1}{m}(-1) = M_{gj}$$

$$M_{an} = Y_{an} + \frac{1}{m}\{X_{ag} + X_{aj} + X_{ad} + X_{aa} + X_{nn} + X_{ng} + X_{nj} + X_{nd}\}$$

$$= -1 + \frac{1}{m}(-1) = M_{na}$$

Discussion of matrix M_i:

→ We define the case with $a = d = j = g = n = 0$ as our default case

a b c d e f g h i j k l m n o p g q r j s t g u v d n w x a

0 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 0

⇒ Max. number of 1s: 10

→ Matrix M, gives us now information how the ~~maximum number~~ maximum number of 1's change regarding the defect case, if we add addition vertices.
(here: 10) (0→1)

Some examples:

add	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	
a&d	1	0	0	1	0	1	0	1	0	0	1	0	1	0	1	0	0	1	0	0	1	0	1	0	9+2=11
j&n	0	1	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	9+2=11
a&d&g	1	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0	7+3=10
a&g	(x¹) 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	1	10+2=12
	(x²) 1	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	

a&d: $M_{aa} + M_{ad} + M_{dd} + 2 = 0 - 1 + 0 + 2 = \underline{\underline{1}}$ (m=2)

j&n: $M_{jn} + M_{jj} + M_{nn} + 2 = 0 + \frac{1}{m}(-1) + \frac{1}{m}(-1) + 2$ (m=2)

$= 0 + \frac{1}{2}(-1) + \frac{1}{2}(-1) + 2$

$= -1 + 2 = \underline{\underline{1}}$

a&d&g: ~~$M_{aa} + M_{ad} + M_{ag} + M_{dd} + M_{dg} + M_{gg} + 3$~~ (m=3)

$= 0 + (-1) + 0 + 0 + (-2) + 0 + 3$

$= -3 + 3 = \underline{\underline{0}}$

a&g: $M_{aa} + M_{ag} + M_{gg} + 2 = 0 + 0 + 0 + 2 = \underline{\underline{2}}$ (m=2)

→ The parameter m equals the number of vertices which we want to add. The factor $\frac{1}{m}$ is necessary since else elements of X are added m times if we consider m vertices.

→ Since we are looking for the maximum solution, we are interested in the '0' entries of $\underline{M}$.

⇒ We get the maximum independent sets of G for the maximum independent set(s) of $\underline{M}$. (with '0' diagonal entries)

→ In our example the maximum independent set is $\{a,g\}$ of $\underline{M}$

⇒ The maximum independent set of G is:

1. $\{a,g\} \cup$ (*1)

(*1): see example previous page: all the vertizes of the bit gap represented by 1

and 2. $\{a,g\} \cup$ (*2)

→ We also see that we get solutions with the same number of 1s like the default case if we add exactly one vertex with entry sum −1, regarding this additional vertex. For example, see (*) left.

(*)

$\underline{M}$:=	a	b
a	−1	a
b	a	−1

If all entries of $\underline{M}$ are ≠0 ⇒ the default case is already the maximum solution

→ then we could still look for vertizes with −1 diagonal entries for getting additional maximum solution of the same size.

e.g.: ⇒ max. indisets: $\{a,x\}, \{b,x\}$ $\{a,b\}$

→ For finding the maximum independent sets of $\underline{M}$ for getting the maximum solution of G, we have to change $\underline{M}$ in the following way: 1. set all entris <0 to 1.
2. set all diagonal entries ≠0 to 0

⇒ $\underline{M'}$:=

	a	d	j	g	n	deg
a	0	1	1	0	1	3
d	1	0	1	1	1	4
j	1	1	0	1	0	3
g	0	1	1	0	1	3
n	1	1	0	1	0	3

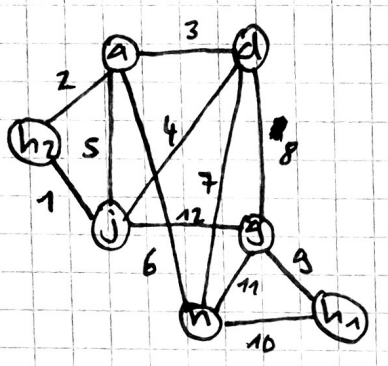
1. We search for the maximum independent sets in $\underline{M'}$
2. Then we check for each of this sets if all their diagonal entries are 0 in the original matrix $\underline{M}$, to determine the valid final solutions.

→ We now analyse M'. We see not all vertices have even degree, so that the belonging graph G' can be Eulerian

⇒ Hence, we add two helper vertices h_1 and h_2

⇒ The new matrix consists of 7 vertices. Five with degree 4 and two with degree 2.

New graph G'



	a	d	j	g	n	h_1	h_2		deg
a	0	1	1	0	1	0	1	a	4
d	1	0	1	1	1	0	0	d	4
j	1	1	0	1	0	0	1	j	4
g	0	1	1	0	1	1	0	g	4
n	1	1	0	1	0	1	0	n	4
h_1	0	0	0	1	1	0	0	h_1	2
h_2	1	0	1	0	0	0	0	h_2	2

⇒ We use Hierholzer's algorithm again to get the Eulerian cycle:

$$j\,h_2\,a\,d\,j\,a\,n\,d\,g\,h_1\,n\,g\,j$$

→ Since all the original vertices {a, d, j, g, n} has the same degree now, we reached our final step of our algorithm for the determination of maximum independent sets.

→ In our example we have helper vertices. Since helper vertices are never part of the maximum independent set solutions, we set their values in the belonging bit string of our new Eulerian cycle to 0.

So, we have to analyse the following case:

$$j\,h_2\,a\,d\,j\,a\,n\,d\,g\,h_1\,n\,g\,j$$

⇒ $\underbrace{j}_{1}\,0\,\underbrace{a\,d\,j\,a\,n\,d\,g}_{2}\,0\,\underbrace{n\,g\,j}_{3}$

→ We can rewrite this to:

⇒ $0|adjandg|0 \quad 0|ngj|0$

≙ $0|x_1x_2x_3x_1x_4x_2x_5|0 \quad 0|x_4x_5x_3|0$

with $x_1 := a$, $x_2 := d$, $x_3 := j$, $x_4 := n$, $x_5 := g$

and consider the possible allowed cases:

$0|x_1x_2x_3x_4x_2x_5|0 \quad 0|x_4x_5x_3|0$

$0|1\ 0\ 0\ 1\ 0$

$0|x_1x_2x_3x_1x_4x_2x_5|0 \quad 0|x_4x_5x_3|0$ — 1s in the gaps

1.	$0\|1\ 0\ 0\ 1\ 0\ 0\ 1\|0 \quad 0\|0\ 1\ 0\|0$	$4 \Longleftrightarrow$ 2 vertices
2.	$0\|0\ 1\ 0\ 0\ 0\ 1\ 0\|0 \quad 0\|0\ 0\ 0\|0$	$2 \Rightarrow$ 1 vertex
3.	$0\|0\ 0\ 1\ 0\ 1\ 0\ 0\|0 \quad 0\|1\ 0\ 1\|0$	$4 \Rightarrow$ 2 vertices
4.	$0\|0\ 0\ 0\ 0\ 1\ 0\ 0\|0 \quad 0\|1\ 0\ 0\|0$	$2 \Rightarrow$ 1 vertex
5.	$0\|0\ 0\ 0\ 0\ 0\ 0\ 1\|0 \quad 0\|0\ 1\ 0\|0$	$2 \Rightarrow$ 1 vertex

Problem Characteristic:

→ Case 1 belongs to solution $\{a, g\}$

→ Case 3 belongs to solution $\{j, n\}$

all x_i have the same degree!

↳ this isn't a maximum solution, like we already mentioned, because of $M_{ij} = \frac{1}{m}(-1)$ and $M_{\substack{m \\ n}} = \frac{1}{m}(-1)$

→ In this example we had helper vertices,

We saw that this vertices 'split' our bit string

→ In the case of not having any helper vertices, we simple get one string like e.g. $x_1x_2x_3x_1x_4x_5x_2$ —— with $\deg(x_1) = \deg(x_2) = \dots = \deg(x_5)$.

→ we introduce false bits $b_1 = b_2 = 0$, so that we finally get: $0|x_1x_2x_3x_1x_4x_5x_2 - - 10$

for which we have to find the maximum configuration

$\forall x_i \quad \deg(x_i) + 1$

If full rank of M' ⇒ we have a fully connected graph G' ⇒ we reached last recursion step of the algorithm

→ Until now, we started with the trivial case of $d_{min} = 2$, and looked a bit at the case of degree > 2 of our example.

→ Now, we want to discuss $d_{min} > 2$, a bit more.

→ We see, we can write for ...

→ $d_{min} = 2$: $\sqcup b_1 | x_1 x_2 \dots x_m | b_2 \sqcup b_3 | x_{m+1} \dots x_n | b_4 \dots \sqcup$

↑ Placeholder for the rest of the bits of S_E^{N-1}

gap g_1 gap g_2 ...

- Each x_i appears exactly ones
- All gaps g_j are independently from each other
- We can interpret the set S of all x_i as an own subgraph G_S of G.
- For $d=2$ it's trivial to see how the number of 1s maximize for $b_1 = b_2 = b_3 = b_4 = \dots = 0$
- It's als easy to see under which conditions the default case changes and how (stay equal, increment, decrement) if we change a bit b_i from 0 to 1.
- We can represent this changings for the whole bit string S_E^{N-1} in a matrix M.

→ $d_{min} = 4$:

e.g. $b_1 | x_1 x_2 x_1 | b_2 \quad b_3 | x_2 | b_4$

- Each x_i appears exactly two times.
- $x_i x_i$ is prohibit (would be an edge on themselve)
- $x_i x_j = x_j x_i$ because of undirected graph
- $x_i x_j$ can only appears ones (because of Eulerian cycle, we pass each edge only once)
- gaps are not longer independently.

- We can interpret the set S of all x_i as an own subgraph G_S of G.
- For $d=4$ it is no longer trivial to see how the number of 1s maximize for $b_1 = b_2 = b_3 < b_4 = \ldots = 0$
- It's also no longer super easy to see under which conditions the default case changes and how (stay equal, increment, decrement) if we change a bit b_i from 0 to 1.
- We have to represent this changings for the whole bit string S_E^{N-1} in a 4dimensional [redacted] M^{4d}

Example:

	b_1	x_1	x_2	x_3	x_4	b_2	b_3	x_2	x_4	x_1	x_3	b_4	
default case all b_i are 0 (*1)	0	0	1	0	0	0	0	1	0	0	0	0	1 vertex
	0	0	0	1	0	0	0	0	0	1	0	0	
	0	0	0	0	1	0	0	1	0	0	0	0	
	0	1	0	0	0	0	0	0	1	0	0	0	
three b_i are 1, one b_i is 0 (*2)	1	0	0	1	0	1	1	0	0	0	1	0	$b_1 b_2 b_3$
	1	0	1	0	0	1	0	1	0	0	0	1	$b_1 b_2 b_4$ 4 vertices
	1	0	0	0	1	0	1	0	1	0	0	1	$b_1 b_3 b_4$
	0	1	0	0	0	1	1	0	0	1	0	1	$b_2 b_3 b_4$
all four b_i are 1 (*3)	1	0	0	0	0	1	1	0	0	0	1	1	4 vertices $b_1 b_2 b_3 b_4$

default 1 $\xrightarrow{+3}$ 4

If we have (*2) solutions: we take three b_i → [redacted] for each b_i we have 0 entries in M

If we have (*3) solution: We take four b_i → we have 0 entries for three b_i entry and one −1 [redacted] like $\sum$ entry (b_1, b_2, b_3, b_4) = −1

⇒ For the general case of $d_{min} = 2n$, $n \in \mathbb{N}$, follows: 23

- Each x_i appears exactly n times.
- $x_i x_i$ is prohibit
- $x_i x_j = x_j x_i$, for undirected graphs
- $x_i x_j$ can only appears ones (because of Eulerian cycle)
- gaps are not independently (apart from case $d_{min} = 2$)
- We can interpret the set S of all x_i as an own subgraph G_S of G.
- It's not trivial to see how the number of 1s maximize for $b_2 = b_1 = b_3 = \dots = 0$. (apart from case $d_{min} = 2$)
- It's not easy to see under which conditions the default case changes and how, if we change a bit b_i from $0 \to 1$ (apart from case $d_{min} = 2$)
- We have to represent this changings for the whole bit string S_E^{N-1} in a 2n-dimensional M^{2n}.

Construction of M^{2n}:

- We assume, we understand the maximization of 1s of the default case.
- We also assume the conditions when the maximal number of 1s of the default case stay the same, get increased or get decreased if we change a bit b_i from 0 to 1.

- For $d_{min} = 2n$, $n \in \mathbb{N}$, and n_c (the number of) x_i of degree d_{min}, we have until: $n_c \cdot n \cdot 2$ bits b_i (coupling gap bits)

- We take an example again: ($d_{min} = 4$)

S_E^{n-1}: $c_1 c_2 b_{11} | x_1 x_2 x_3 x_4 | b_2 c_3 c_4 c_5 b_3 | x_2 x_4 x_1 x_3 | b_4 c_6 c_1$

with $g_1^2, g_2^2, g_1^4, g_3^2, g_4^2, g_5^2, g_6^2, g_7^2, g_2^4, g_8^2, g_9^2$

→ x_i are all the deg=4 vertices of S_E^{n-1} : $deg(x_i) = 4$

→ c_1, c_2, c_3, c_4, c_6 and $b_{11}, b_{21}, b_{31}, b_4$ have all any degree $deg(c_i) > 4$ respectively $deg(b_i) > 4$.

→ We now determine the changings of all changing steps:

→ Step 1: We change one b_i or c_i from 0 to 1 (and assume the other b_i are still 0)

1. We look for g_1^4 and g_2^4 and determine if the changing change the number of 1s

   (have attention, that apart from the $d_{min} = 2$ case, the gap changing behaviour can not be considered as symmetrical, in generally)
   (means $b_1 = 0$ and $b_2 = 1$ can have a different influence like $b_1 = 1$ and $b_2 = 0$ e.g.)

   → Note this relative changing to the default case

2. We interpret the connections of all other bits as $d = 2$ direct connections like we already did it in the example for generating M at first.

We go through the bit string S_E^{N-1} and look for the direct neighbour bits c and b of b_i, we interpret this changing then like the simple case with gap size $|g_i^2|=0$.

A changing for $[0,0] \to (1,0]$ or $(1,0)$ means a matrix of 0 for the overall solution of our contribution

step 1 case.

→ Step 2: We change one additional bit (b_i or c_i) from 0 to 1 and do the same constructions like for step 1. For our g_i^2 gaps we get a -1 for better an number < -1, since this would be the prohibit '11' pattern which we want to mark special) if changing two bits leads to a '11' case. For each appearing of '11' we count -1, so we can interpret this like $(0,1)$ or $(1,0) \to (1,1)$ for g_i^2 cases

→ Step 3 and step 4: Analog to step 2.

→ Final construction of M^{2n}:

→ The necessary information can be easily derived from the Eulerian cycle in $O(|E|^2)$ time.

We make a concatenation of our part results (analog to of step 1 to 4, our $deg_{max}=2$ example) to get M^{2n}, a 2n dimensional representation

Summary:

⇒ In each algorithm step, we can choose all vertices with degree n.

⇒ If we understand the default case and its changing behaviour for $d_{min}=2n$, we can define $\underline{M}^{2n}$

⇒ Making $\underline{M}'^{2n}$ Eulerian, to identify the maximum independent sets of $\underline{M}'^{2n}$.

⇒ Identification of Eulerian cycles of $\underline{M}'^{2n}$

⇒ Taking again d_{min} of $\underline{M}'^{2n}$ and identify for this new d_{min} the next $\underline{M}^{2n}$ representation

⇒ ... and so on.

⇒ Abboard condition: 1. Getting a fully connected graph by $\underline{M}^{2n}$

or 2. All vertices of $\underline{M}^{2n'}$ have the same degree.

⇒ Finaly determine our searched solution by taking the maximum independent set of the last step, to identify the one of the direct step before, and so on ...

⇒ Recursive Solution

Problems: 1. Understanding of default case and its changing if we take allway all vertices with d_{min} in each step.

2. We have to work with d-dimensional representations $\underline{M}^{2n}$ in each step.

We have two possible approaches for this problem:

Approach 1:

- We are not forced to take all degree-min vertices. We could e.g. only choose one d-min vertex, and handle all the other vertices, including the other deg-min vertices, like we did in our step explanation as 'd=2' connections.

  Advantage: It's trivial to understand when a single vertex of d-min change and it's maximum number of 1's in the default case — Additionally we get a 2-dimensional again

  e.g. $c_4 b_1 | x_1 | b_2 c_2 .. c_3 b_3 | x_2 | b_4 c_4 .. b_5 | x_3 | b_5 ..$

  have attention, that now b_i can also be deg-min vertices

  → In the default case, the maximum number of 1's is always 1.

  → The maximum number of 1s decrease by 1, to 0, if we get on b_i with 1.

  → If we have additional b_i with 1, nothing change anymore, so we still have the same.

  (→ we could of course also choose any other set of vertices whose connection nature we do understand)

  Disadvantage: The deeper understanding of the characteristic / nature of maximum independent sets get lost!

(*) Have attention that choosen vertices connection which don't change anything can lead to the original 1^{2n} of this step again.

→ we have to think about the right kinds of vertices connections, not to end in loops

(*) (*)

⇒ This approach mainly! (not complete!) equals the algorithm of Tarjan and Trojanowski. (we will come back to this later)

[Tarjan & Trojanowski]

Tarjan, R. E, & Trojanowski, A.E. (1977). 'Finding a maximum independent set'. SIAM Journal on Computing, 6(3), 537-546.

Approach2:

- We take all degmin vertices,

Advantage: - Better understandings of the nature of maximum independent sets.

- If we understand all same degree bit string parts in generall, we can derive the solutions of maximum independent sets for any arbitrary graph G by putting the same degree cases together.

→ We get immadiatly a deeper understanding of their structure nature.

Disadvantage: - We have to handle Λ^{2n} representations on 2n-dimensions

- To get the properties of the same degree substrings isn't trivial.

(*) Additonally, we can derive a kind of 'mapping' from the Λ-2dim to the Λ-2n-dim case!!!?

But:

- We can analyse the same degree substrings with the same method which we described in approach 1 for the full bit string, in generally.

- Trying to figure out a good system of choosing bit by bit within the substring for a given substring pattern (we know the degree, the number of this bits, the order of this bits and how much gaps (respectively b_i) they build from the Eulerian cycle!)

(*) - So, we can derive a generall 'approch' for same degree substrings!?

Comparison with Tarjan & Trojanowski

[Tarjan&Trojanowski]

Tarjan, R.E., & Trojanowski, A.E. (1977). 'Finding a maximum independent set'. SIAM Journal on Computing, 6(3), 537-546.

Link to pdf: https://infolab.stanford.edu/pub/cstr/reports/cs/tr/76/550/CS-TR-76-550.pdf

---

[Robson2001]

Robson, J.M. (2001), 'Finding a maximum independent set in time $O(2^{n/4})$'.

Link: https://www.labri.fr/perso/robson/mis/techrep.html

→ We want compare our approach with already existing algorithms.

- Over the years several ones were published. Since the most of them are using the same basic concept, just with different kinds of improvements mostly regarding case studies, we only want to focus on the orignal one by Tarjan & Trojanowski [Tarjan & Trojanowski] with $O(2^{n/3}) = O(1.2599^n)$.
The most efficient currently know version was given by Robson [Robson2001] with $O(2^{0.249n}) = O(1.1888^n)$.

- Tarjan & Trojanowski give at first a few definitions/theorems then their algorithm until degree 8 and finally a complexity analysis.
We want compare their pre work and the considerations until degree 2 with our idea, to understand the similarities and the differences.

→ [Tarjan & Trojanowski]'s introduction

- page 4: 'The algorithm uses a recursive, or backtracking scheme'

  → We use a recursive algorithm, but no backtracking and with try-and-error scheme!

- page 4: 'Let $v \in V$. Let $A(v)$ be the set of vertices adjacent to v. Then any maximum independent set either contains v or it does not. Thus any maximum independent set of G is either $\{v\}$ combined with a maximum independent set in $G[V-\{v\}-A(v)]$, or it is a maximum independent set in $G(V-\{v\})$.'

- page 5: 'We extend this idea. For any $S \subseteq V$, let $A(S) = \bigcup_{v \in S} A(v)$. If $S \subseteq V$, then any maximum independent set in I in G consists of an independent set $I \cap S$ in $G(S)$ and a maximum independent set $I-S$ in $G(V-S-A(I))$.'

  → This is the equivalent to our substring of S_E^{N-1} considerations, where we look at the maximum number of 1s in our gap ($\exists$ set S here), and the not gap bits ($\exists$ $V-S$). We try to maximize all bits outside of the gap ($\exists$ here: maximum independent set $I-S$ in $G(V-S-A(I))$), like it is done in [Tarjan & Trojanowski], and combine this with the independent set of substring vertices.

- page 5: 'Our algorithm selects a subset $S \subset V$, finds each independent set J in $G(S)$, and for each such J, recursively finds a maximum independent set in $G(V-S-A(J))$.'

→ we not ~~am~~ look for each J in our substring for a maximum independent set in $G(V-S-A(S))$.

~~Instead we already define our~~ Instead our M is already defined in such a way, that we get for the ~~largest~~ maximum independent set in G, the ideal independent set combination of the substring S plus from the other bits $G(V-S-A(S))$.

- page 5: '... by introducing the concept of dominance. If $S \subseteq V$ and I, J are independent in $G(S)$, we say I dominates J if, for any $J' \subseteq V-S$ such that $J \cup J'$ is independent, there is a set $I' \subseteq V-S$ such that $I \cup I'$ is independent and $|I \cup I'| \geq |J \cup J'|$.'

→ This is literally nothing more like 'choose the two subsolutions of the substring plus the other bits in such a way that the solution maximize' which is represented in our algorithm by solving the maximum independent set problem for M.

- page 5: 'We give two examples which are used extensively in the algorithm. let $v \in V$. Let $S = \{v\} \cup A(v)$. If $w \in A(v)$, then $\{v\}$ dominates $\{w\}$ in S, since $I \subseteq V-S$ and $I \cup \{w\}$ independent implies $I \cup \{v\}$ independent'. Similarly, $\{v\}$ dominates $\emptyset$ in S.'

→ Consider: $b_1 | x_1 | b_2 \cdots$ ($V-S$) and $I \subseteq V-S$

$\supseteq$ $A(v) | v | A(v)$ → If $w = b_1$ or $w = b_2$,

→ $I \cup \{w\}$ independent and $I \subseteq V-S \Rightarrow I \cup \{v\}$ independent

$\{v\}$ dominates $\emptyset$ in S: $\hat{=}$ $b_1 = b_2 = 0$ case.

- page 5: 'Let $S \subseteq V$. Let I and $J = I \cup \{v\}$ be independent in $G(S)$. Suppose $(V-S) \cap A(v) = \{w_1, w_2\}$. Then $S \cup \{w_1, w_2\}$, J dominates both $I \cup \{w_1\}$ and $I \cup \{w_2\}$.'

→ Assume: $b_1 | x_1 x_2 x_3 | b_2 \dots$ ($x_1 x_2 x_3$: g)

$\Rightarrow w_1 | x_1 x_2 \; v \; | w_2$ ($x_1 x_2$: I)

$\Rightarrow I \cup \{v\} = J$

This equals the case that taking w_1 or w_2 to our solution reduces the number of 1s in the gap. g.

- page 5: 'We distinguish three possibilities.

(i) $(w_1, w_2) \in E$ or $I \cap A(\{w_1, w_2\}) \neq \emptyset$. Then J dominates I in S: if $I' \subseteq V-S$ and $I' \cup I$ is independent, then $|I' \cap \{w_1, w_2\}| \leq 1$. Thus $J' = I' - \{w_1, w_2\}$ satisfies $|J \cup J'| \geq |I' \cup I|$ and $J' \cup J$ is independent.'

→ Assume:

$b_1 | x_1 x_2 x_3 x_4 x_1 | b_2 \dots b_1 b_2 \dots$

$w_1 | v \; x_2 x_3 x_4 \; v | w_1 \dots w_1 w_2 \dots$

($x_2 x_3 x_4$: I; $J := I \cup \{v\}$; I')

$|I' \cap \{w_1, w_2\}| \leq 1$

$= \{0, 1\}$

If $I' \cup I$ independent, only w_1 or only w_2 or none of them is element of I'.

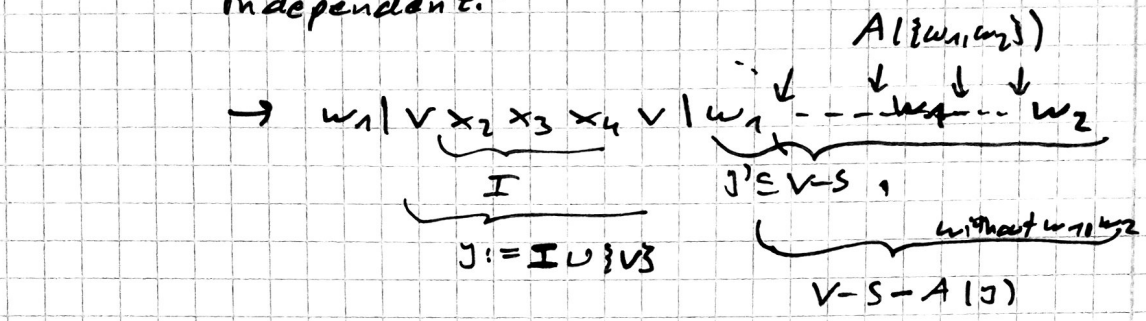
$\Rightarrow J' = I' - \{w_1, w_2\}$ (because of $J := I \cup \{v\}$)

$\Rightarrow |J' \cup J| \geq |I' \cup I| \Rightarrow J$ dominates I

$\Rightarrow$ This means: it is "preferred" to take v instead of w_1 or w_2, which happens if w_1 or w_2 would reduce the number of 1s in substring.

- page 6: '$(w_1, w_2) \notin E$, $I \cap A(\{w_1, w_2\}) = \emptyset$, and ~~[illegible]~~ $|(V - S - A(J)) \cap A(\{w_1, w_2\})| \leq 1$.

Then I dominates J in S (and $I \cup \{w_1, w_2\}$ dominates J in $S \cup \{w_1, w_2\}$: if $J' \subseteq V - S$ and $J' \cup J$ is independent, then $I' = (J' \cup \{w_1, w_2\}) - A(\{w_1, w_2\})$ ~~[illegible]~~ satisfies $|I' \cup I| \geq |J' \cup J|$ and $I' \cup I$ is independent.'



$\Rightarrow |V - S - A(J) \cap A(\{w_1, w_2\})|$

~~[illegible]~~ The rest set ~~[illegible]~~ of bits which doesn't contain S, V, w_1, w_2 ~~[illegible]~~, contains maximal one vertex which is adjacent to w_1 or w_2

$\Rightarrow$ Then we choose I which dominates J

- Since $I \cap A(\{w_1, w_2\}) = \emptyset \Rightarrow I \cup \{w_1, w_2\}$ ✔ that's clear.
- If $J' \cup J$ is independant $\Rightarrow$ J' doesn't contain $\{w_1, w_2\}$

$\Rightarrow |I' \cup I| \geq |J' \cup J|$

$\Rightarrow$ This is the equivalent of our case, when we have a reduced maximization of MS in our gap. ('V' has to dissapear) to get a total maximum solution.

- Now, we look at a ~~few~~ cases of the algorithm ~~itself~~ itself.

- page 7: '0: V is connected

---. Let maxset $= \sum_{i=1}^{k}$ maxset (V_i).

V is connected. Let v be a vertex of minimum degree.'

→ That's the case which we mentioned in the early beginning, that the total maximum ~~[illegible]~~ independent set of a graph G is the combination of the maximum independent sets of all its connected components.

- page 7: '1: d(v) = 1.

Let A(v) = {w}

Let ~~[illegible]~~ maxset = 1 + maxset (V − {v, w})'

→ That's trivial to see.

- page 7: '2: d(v) = 2.

2.1: d(w) = 2 for all w∈V.

Note that the vertices of V form a cycle.

Let maxset = $\lfloor \frac{|V|}{2} \rfloor$.'

→ That's trivial to ~~see~~ see.

- page 8: 'There exist v, w such that d(v) = 2, d(w₁) ≥ 3, and (v, w₁) ∈ E. Let A(v) = {w₁, w₂}.

2.2. (w₁, w₂) ∈ E.

Let maxset = 1 + maxset (V − {v, w₁, w₂}).'

→ $w_1 | V | w_2 - w_1 w_2$

M ⇒ maximum sets {v}, {w₁}, {w₂}

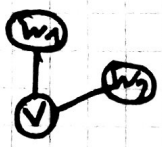
M :=

	w_1	w_2	
	-1(½)	-1	w_1
	-1	-1(½)	w_2

(if we ignore all the other vertices of G)

→ page 8: ' 2.3 $(w_1, w_2) \notin E$.

Let maxset = max{1 + maxset $(V - \{v, w_1, w_2\})$, 2 + maxset $(V - A(w_1) - A(w_2))$}.'



→ If we ignore all the other vertices of G, we would have the M matrix:

→ $w_1 \mid v_a \mid w_2$ ----

M_1:

	w_1	w_2	
	$-1(\infty)$	0	w_1
	0	$-1(\infty)$	w_2

M_1: ⇒ maximum sets: $\{w_1, w_2\}$

→ If we have not other vertices in G like v_a, w_1, w_2 then, this would be our final solution.

→ If we assume that we also consider the other vertices of G, then M can look differently, so that the choice of v could be prefered instead.

We would have e.g.:

M_2:

	w_1	w_2	
	$-1(\infty)$	-2	w_1
	-2	$-1(\infty)$	w_2
	⋮	⋮	⋮ further vertices

→ Tarjan & Trojanowski ignore the other vertices, hence they have to constaer both cases by a backtracking scheme.

We don't ignore the other vertices, so that we get one unique M which maximum independent set solution leads to the total maximum independent set.

Our way of definition of M leads to a representation, which represent both cases, so that we don't need branches with "try-and-error".

(*) Higher degree cases of their algorithm works analog but only more complex, so that we left their discussion out. It's analog to this argumentations which already are given

[TarjanDeco]

Tarjan, R.E. (1985). 'Decomposition by clique separators'. Discrete mathematics, 55(2), 221-232.

- We want to make a sketchy comparison with Tarjan's description of Maximum independent set determination by 'Decomposition by clique separators' given in [TarjanDeco].

- In [TarjanDeco], page 8:

Quote: - Problem: Finding a maximum-weight independent set (each vertex has a real-valued weight)

- If maximum-weight independent sets on certain subgraphs of the atoms can be found, then we can find a maximum-weight independent set for the entire graph.

- We use the following recursive method:

Step 0: Let A, B, C be a vertex partition such that C is clique, no edge joins a vertex in A and a vertex in B, and $G(A \cup C)$ is an atom.
We denote by $wt(I)$ the total weight of vertex set I.

step 1: For each $v \in C$, determine a maximum-weight independent set $I(v)$ in $G[A - adj(v)]$.
Determine a maximum-weight set I' in $G(A)$.

step 2: For each vertex $v \in G$, redefine the weight of v to be $wt(\{v\}) + wt(I(v)) - wt(I')$. Find a maximum-weight independent set I'' in $G'' = G(B \cup C)$ with respect to the new weights.

Step 3: Define $I = I_{(v)} \cup I''$ if $v \in I'' \cap C$

$I = I' \cup I''$ if $I'' \cap C = \emptyset$

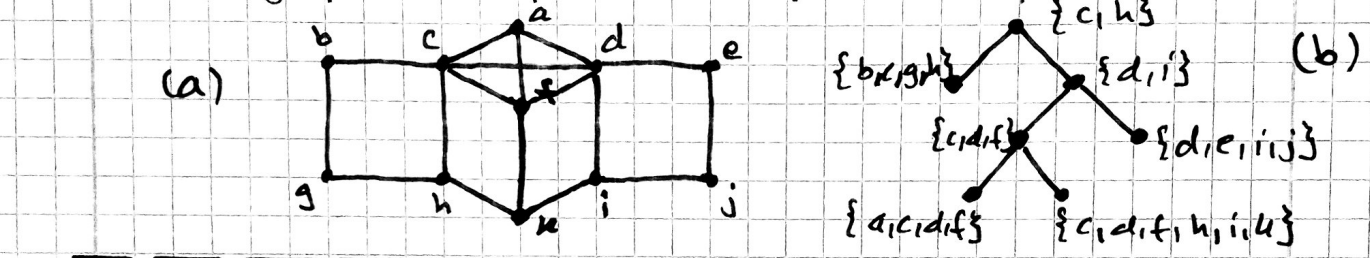
Quote End.

→ **Remark:** I'm not sure regarding the following argumentations!

- Assume again, we have a bit string S_E^{N-1} and $\underline{M}$ as well as $\underline{M'}$;

  ... $c_1 b_1 | x_1 x_2 ... | b_2 c_2 ... c_3 b_3 | x_8 x_9 x_8 ... | b_4 c_8 c_9 ....$ etc...

  also, we want to use the example of [TarjanDeco] (page 2; Fig. 1) for a graph (a) and a possible binary decomposition tree (b).



- We go through the given steps of [TarjanDeco] and will compare this with our algorithm approch.
- We start with identifying A of **step 1**, with our substring vertices x_i.

  The set C equals a set of vertices b_i and c_i with $\underline{M}$ entries $\neq 0$, since taking them influence the maximum number of 1s in the substring which forms a clique. in a negative way.

  → Have attention that we talk about 1 entries in $\underline{M'}$ not in the original adjacence matrix $\underline{A}$, hence this vertices don't need to have a 'geographical' real edge connection. But they have a 'influence' connection with each other (like e.g. b_1 and b_2 which influence $x_1 x_2$ ..., but they don't have to have an edge in $\underline{A}$ between them)

- We can follow from this assumptions, that I' of $G(A)$ in step 1, equals the maximum number of 1s default case in our substring.

The independent set $I(v)$ in $G[A-adj(v)]$, can be considered in the following way:

- Since we not specific know, which b_i and c_i build a clique C, we consider the whole set B and C as $B \cup C$ together. This equals the set of all vertices of our M respectively M'.

- So, we consider $A-adj(v)$ for all $v \in (B \cup C)$, instead.
  The set $A-adj(v)$ equals the consideration which we do if adding an additional vertex to our solution would stay the same, increase or decrease the maximum number of 1s in our default case.

Hence, the independent set $I(v)$ for all $v \in (B \cup C)$ is naturally indirectly given by each choice of vertices of M, since assume e.g. we take b_1 and b_2, we know how the x_i string of a for a maximization between b_1 and b_2 have to looks like, which equals $I(v)$ for two different V_i.

(*) by the choosen $V_i \in M$

If we compare this with the example of [TarjanDeco] our substring vertices x_i of A equals to $\{b, g\}$, $\Rightarrow I' = \{g\}$ and $\{b\}$ and $I(v)$ with $G(\{b,g\} - \{b\})$ and $G(\{b,g\} - \{g\})$ leads to $I(v) = \{b\}$ or $\{g\}$ for the (0,0) to (0,1) or (1,0) case and $\{\emptyset\}$ for (0,1) or (1,0) to (1,1) case, which is put together in our matrix M (M') (*)

- If we look at step 2 and the redefinition of weights of v, we recognize that $wt(\{v\})$ (*2) $+ wt(I(v)) - wt(I')$ (*1)

(*1) equals our definition of M entries.

It gives us the difference of number of 1s in the substring between the default case (I') and the case of changing a bit from 0 to 1, respectively changing several bits from 0 to 1 ($I_{(v)}$).

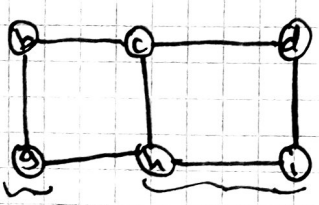
(*2) this equals the addition of the taken bits

So, finding the maximum weight independent set I'' in $G'' = G(B \cup C)$ with respect to the new weights, it's nothing more like finding the maximum independant set of M^* (respectively M').

(Keep in mind/remember that we derived the M entries with the help of considerations which we are able to identify with domination consideration of the algorithm of [Tarjan & Trojanowski].)

We compare this again with the example of [Tarjan Deco].

$B \cup C = \{c, h, d, i\}$ and $A = \{b, g\}$ (with ignoring the rest of the vertices to keep it simple)



$M :=$

	c	h	d	i	
	0	-X	-X	0	c
	-X	0	0	-X	h
	-X	0	0	-X	d
	0	-X	-X	0	i

-X because of prohibit '11' pattern

possible clique in M are: $\{c, h\}$, $\{d, c\}$, $\{h, i\}$, $\{d, i\}$

$\Rightarrow B = \{d\}$ (↑ not usable for clique decomposition)

$I' = \{b\}, \{g\}$ $\Rightarrow |I'| = 1$ default

$I'' = \{d, h\}, \{c, i\}$

$\Rightarrow c|bg|h$

or 01 / 10 possible

Total $\Rightarrow I = \{b, d, h\}, \{g, c, i\}$ $\Rightarrow |I_{max}| = 3$ $\Rightarrow +2$ regarding the default case.

40

- Finally, we look at step 3.

Here, we differ between the two cases $v \in I'' \cap C$ and $I'' \cap C = \emptyset$ to take $I_{(v)} \cup I''$ or $I' \cup I''$.

If you remember that we always can derive clearly our maximum number of 1s and their belonging configuration in our substring from the taken b_i, it becomes clear that if we know I'', we can directly derive from only I'' the set with which we have to merge I'', without the necessity of knowing C and without the necessity of knowing if $v \in I'' \cap C$ or $I'' \cap C = \emptyset$.

We compare this again with the example of [TarjanDeco].

$I'' = \{d,h\}$ and $\{c,i\}$, and $C = \{c,h\} \Rightarrow \{c,h\} \cap \{d,h\} = \{h\}$
$\{c,h\} \cap \{c,i\} = \{c\}$

- Assume, we take $I'' = \{d,h\}$:

$\Rightarrow$

	c	b	g	h
●	1	0	1	

$\Rightarrow I_{(v)}^{v=h} = \{b\}$

$\Rightarrow I = \{b\} \cup \{d,h\}$

for $C = \{c,h\}$, $adj(h) = g$
since $G(\{b,g\} - \{g\})$
$\Rightarrow I_{(h)} = \{b\}$
and $h \in I'' \cap C$

- Assume, we take $I'' = \{c,i\}$

$\Rightarrow$

	c	b	g	h
	1	0	1	0

$\Rightarrow I_{(v)}^{v=c} = \{g\}$

$\Rightarrow I = \{g\} \cup \{c,i\}$

for $C = \{c,h\}$, $adj(c) = b$
since $G(\{b,g\} - \{b\})$
$\Rightarrow I_{(c)} = \{g\}$
and $c \in I'' \cap C$

Since $I'' \cap C \neq \emptyset$ for both I'':

In this example, we don't have a case $I'' \cap C = \emptyset$, at first glance. If we look closer at H again, we see that we get this case, if we take $\{d\}$ or if we take $\{i\}$. ($C = \{c,h\}$: $C \cap \{d\} = \emptyset$, $C \cap \{i\} = \emptyset$)

This would mean: $I = \{b\} \cup \{d\}$, $\{b\} \cup \{i\}$, $\{g\} \cup \{d\}$ and $\{g\} \cup \{i\}$.

But what you can see, that this I, would not be the maximum solution, because of $|I|=2$, (In the other case, we had $|I|=3$)

So, you can see, that our consideration makes sense.

- Until now, we have ignored step 0. We want to catch up on this, now.

- ' Let A, B, C be a vertex partition such that C is clique, no edge joins a vertex in A and a vertex in B '

  → Following our definition from substrings with vertices x_i, we know that all vertices which are adjacent on M to x_i, is the set of b_i and c_i which vertices from b_i and c_i which chances the cardinal number of the maximum independent set of all x_i negative.

  We define A as the set of substring vertices x_i.

  ⇒ If it exist at least one b_i, c_i which influence I_{max} of A negative ⇒ It exist one clique of size $|C| \geq 1$, which decrease I_{max}, maximal

  ~~Furthermore, from our definition of substrings, directions~~

  ⇒ $\forall$ b_i, c_i which are not element of C, there exist no adjacent connection to elements of A in M which are also adjacent to elements of A in the adjacent matrix A of G.

  Since vertices of A which are adjacent to elements of A, can also influence other vertices apart from A, this doesn't mean that their entries in M' have to be 0! It's only mandatory 0 regarding A

- '...and $G(A \cup C)$ is an atom'.

→ After [TarjanDeco], G is an atom if there exists no clique, so that G is decomposable.

Again, let's define A as the set of substring x_i and C clique, which influence I_{max} of A negative.
We know that all $k \in C$ which influence I_{max} of A negative have an $\underline{M}$ entry $\neq 0$, and are adjacent to at least one member of A. So, we can describe this by
$\dots b_1 | x_1 x_2 x_3 \dots x_n | b_2 \dots$ with at least one $b_i \in C$, with $b_i = 1$ in $\underline{M}$).
We define: $\underbrace{x_1 x_2 x_3 \dots x_n | b_2}_{=: G(A \cup C) \text{ (for example).}}$
for all elements of A $x_i \in A$.

Since, for our algorithm, we can choose freely any kind of substring which default case and changing behaviour we understand in each step of our algorithm, we can always choose an atom one.

The most trivial substring which we can choose consists of exactly one x_i. For the case of $\deg(x_i) = 2n$, we have until $2n$ b_i connections. like $b_1 | x_1 | b_2 \; b_3 | x_n | b_4 \dots$

We know for this choice: - $|I_{max}| = 1$ for the default case
- If we add one $b_i = 1$: $|I_i|$ of A changes to 0.
- If we add additional $b_j = 1$: $|I|$ of A don't change anymore, (since it is already 0).
- If we ignore the additional bits of S_E^{N-1}: With each additional $b_i = 1$, we increase the total $|I_{total}|$ of S_E^{N-1} by +1.

⇒ If we choose $|A|=1$, with $A:=\{x_i\}$, with $2n$, b_i connections, we get for $G(A \cup C)$, a set

$A \cup C = \{x_i, b_i\}$ with $|A \cup C| = 2$.

↑ that's the b_i which decrease $|\Gamma|$ of A in the first place.

Since a set of size 2 is never clique decomposable (see also the following first example)

⇒ $G(A \cup C)$ is an atom.

Example for $\deg(x_i)=4$ with $|A|=1$:

We have: ... $b_1 | x_1 | b_2 \ldots b_3 | x_1 | b_4$ ...

We ignore the other bits of S_ε^{N-1}, to keep it simple.

and also we assume that no connections between the b_i's exist.

From default case $(0,0,0,0) \rightarrow (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)$:

$X :=$

	b_1	b_2	b_3	b_4	
	0	-1 | 0 | 0	0 | -1 | 0	0 | 0 | -1	b_1
	//	0	-1 | 0	0 | -1	b_2
	//	//	0	-1	b_3
	//	//	//	0	b_4

(symmetric)

$Y :=$

b_1	b_2	b_3	b_4	
0	0	0	0	b_1
0	0	0	0	b_2
0	0	0	0	b_3
0	0	0	0	b_4

↔ (1 times 0, 3 times 1) → (1,1,1,1)

(it's the same also for
(3 times 0, 1 times 1) → (2 times 0, 2 times 1)
(2 times 0, 2 times 1) → (1 times 0, 3 times 1))

$\Rightarrow \underline{M} :=$

	b_1	b_2	b_3	b_4	
	$-1(\frac{1}{m})$	0	0	0	b_1
	0	$-1(\frac{1}{m})$	0	0	b_2
	0	0	$-1(\frac{1}{m})$	0	b_3
	0	0	0	$-1(\frac{1}{m})$	b_4

If we also consider the ignored c_i bits, and set the diagonal entries to 0, we get again the original adjacence matrix $\underline{A}$ of G, just reduced by the vertex x_i.
→ (the influence of x_i is encoded into $\underline{M}$ by the diagonal entries $-1(\frac{1}{m})$)

- Choosing only substrings $|A|=1$, leads us to a typical branching algorithm like [Tarjan & Trojanowski]

Example for $\deg(x_i)=4$ with $|A|=2$

b_1	x_1	x_2	b_2	b_3	x_1	b_4	b_5	x_2	b_6
0	1	0	0	0	1	0	0	0	0
0	0	1	0	0	●	0	0	1	0

default case, size 1

$\underline{X} :=$

	b_1	b_2	b_3	b_4	b_5	b_6	
	0	0	0	0	0	0	b_1
	0	0	0	0	0	0	b_2
	0	0	0	0	0	0	b_3
	0	0	0	0	0	0	b_4
	0	0	0	0	0	0	b_5
	0	0	0	0	0	0	b_6

$\underline{Y} :=$

	b_1	b_2	b_3	b_4	b_5	b_6	
	0	-1	0	0	-1	-1	b_1
	-1	0	-1	-1	0	0	b_2
	0	-1	0	0	-1	-1	b_3
	0	-1	0	0	-1	-1	b_4
	-1	0	-1	-1	0	0	b_5
	-1	0	-1	-1	0	0	b_6

$\Rightarrow \underline{M} :=$

	b_1	b_2	b_3	b_4	b_5	b_6	
	0	$-1\binom{A}{m-1}$	0	e	$-1\binom{A}{m-1}$	$-1\binom{A}{m-1}$	b_1
	$-1\binom{A}{m-1}$	0	$-1\binom{A}{m-1}$	$-1\binom{A}{m-1}$	0	0	b_2
	0	$-1\binom{A}{m-1}$	0	0	$-1\binom{A}{m-1}$	$-1\binom{A}{m-1}$	b_3
	0	$-1\binom{A}{m-1}$	0	0	$-1\binom{A}{m-1}$	$-1\binom{A}{m-1}$	b_4
	$-1\binom{A}{m-1}$	0	$-1\binom{A}{m-1}$	$-1\binom{A}{m-1}$	0	0	b_5
	$-1\binom{A}{m-1}$	0	$-1\binom{A}{m-1}$	$-1\binom{A}{m-1}$	0	0	b_6

- The factor $\binom{A}{m-1}$ = number of -1's, has to be introduced, since we are considering a 6-dimensional structure in a 2-dimensional representation.
- We see, that considering substrings with set $|A| > 1$, leads us to higher dimensional $\underline{M}$ representations, and the considerations of finding (maximum) independent sets on higher dimensional structurs.
- Additional, if we want to use substrings with set $|A| > 1$, it makes still sense to use Eulerian cycles, to derive the necessary information, since it is more efficient like to get the same information from adjacence matrices $\underline{A}$ or higher dim-$\underline{M}$, as long Eulerian cycles can be still determined in an efficient way, too. on higher dim-$\underline{M}$ representations.

↳ This leads to further Topological analysis of substring sets A on higher dim-$\underline{M}$ representations.

Résumé

- Given an undirected graph G and we are interested the maximum set(s).
- The most natural, characteristic, understanding way of solving the problem is a combination of [Tarjan & Trojanowski] like ~~translation~~ algorithm which use domination considerations, but branching ~~on~~, with [Tarjan] clique based decomposition based algorithm. But instead of doing Branching like in [Tarjan & Trojanowski] we put the domination considerations in higher dimensional representations $\underline{M}$ based on a splitting like in [Tarjan].
- With this the maximum independent set(s) can be found efficiently, ~~if~~ in the following way: (we assume the graphs of each step are always connected for a simpler explanation, ~~more~~ here)

1. Take G (respectively its adjacent matrix $\underline{A}$), make it Eulerian and determine an Eulerian cycle with Hierholzers algorithm.

2. Take a substring A with size $|A| \geq 2$. Depending on the degree(s) of the ~~so~~ chosen ~~deg~~ vertices, we can get until $2n - bi$ bits (with n = max degree of vertices in set A) $\Rightarrow$ we get a $2n$-dimensional new representation $\underline{M}$ with $|G-A|$ vertices, The entries in $\underline{M}$ are ~~2-dim~~ determined by weight/dominans considerations of vertices $(G-A)$ regarding A. We see that we get the maximum independent set, for a maximum independent set in $\underline{M}$ ~~again~~.
3. Determine ~~again~~ ~~another~~ an Eulerian cycle of $\underline{M}$. Take a substring A_2 with $|A_2| \geq 2$ and ~~again~~

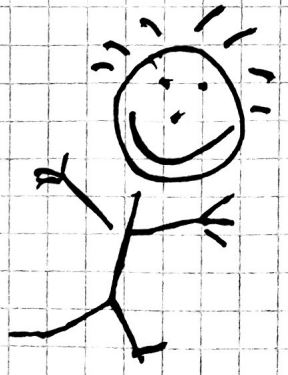
4. We reached the final step n if we get left with the trivial case of $|\overline{A_n^*}| = 2$.

5. we can then take the solution of the final step n, to get the solution of the step $n-1$, Then this one to get the solution of $n-2$ and so on ... until we make it back to the original graph G step from the beginning.

⇒ What we can see is, that the maximum independent set is naturally given by a concatenation of higher-dimensional substructures given by M, defined by sets of size $|A| \geq 2$. It can be assumed that the size of the set A can be considered as 'small' since there is no reason to choose large ones apart from avoiding the case of choosing A in such a way that we generate the original same matrix again.

(*) we assume that 'small' means sets of the size $|A| = 2$ and $|A| = 3$

So the algorithm can solve the problem in an efficient way, if we are able to understand the topological structure of 'small' A substring sets in a general way. and how they build M.



Additional References

[1] Robson, J. M. (1986). 'Algorithms for maximum independent sets'. Journal of Algorithms, 7(3), 425-440.

[2] Xiao, M., & Nagamochi, H. (2017). 'Exact algorithms for maximum independent set'. Information and Computation, 255, 126-146.

[3] Whitesides, S. H. (1981). 'An algorithm for finding clique cut-sets'. Information Processing Letters, vol. 12, nb. 1, 31-32.

[4] Alekseev, V. E. (2003). 'On easy and hard hereditary classes of graphs with respect to the independent set problem'. Discrete Applied Mathematics, 132(1-3), 17-26.

[5] Coudert, D., & Ducoffe, G. (2018). 'Revisting decomposition by clique separators'. SIAM Journal on Discrete Mathematics, 32(1), 682-694.

[6] Brandstädt, A., & Mahfud, S. (2007). 'New applications of clique separator decomposition for the Maximum Weight Stable Set problem'. Theoretical Computer Science, 370(1-3), 229-239.

[7] Brandstädt, A., & Hoàng, C. T. (2007). 'On clique separators, nearly chordal graphs, and the Maximum Weight Stable Set Problem'. Theoretical Computer Science, 389(1-2), 295-306.

[8] Georgakopoulos, A. (2009). 'Topological circles and Euler tours in locally finite graphs'. The electronic journal of combinatorics, R40-R40.

[9] Duval, A., Klivans, C. J., & Martin, J. L. (2016). 'Simplicial and cellular trees'. In Recent trends in combinatorics (pp. 713–752). Springer, Cham.

[10] Spivak, D. I. (2009). 'Higher-dimensional models of networks'. arXiv preprint arXiv:0909.4314.