

# Skupina 18: Kemijski grafi

Avtorja: David Planinšek Šilc, Lenart Žerdin

Datum: 20. 12. 2024

## Opis problema

Najina naloga temelji na raziskovanju kemijskih grafov in njihovem  $\sigma_t^{f(n)}(G)$  indeksu. Zanima naju, kako se indeks v odvisnosti od različnih  $f(n)$  spreminja in katera stopenjska zaporedja grafa pridelajo njegov maksimum. Omejila sva se na funkcije  $f(n) = \frac{1}{n}$  in  $f(n) = c$  za  $c \in (0, 1)$ , kjer sva podrobneje gledala tiste  $c$ , ki so blizu 0 in 1.

## Definicije

1. Graf je kemijski, če je neusmerjen, povezan in so vsa njegova vozlišča stopnje največ 4. Če ima kemijski graf  $a_i$  vozlišč stopnje  $i$ ,  $1 \leq i \leq 4$ , potem njegovo stopenjsko zaporedje označimo kot  $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$ .
2. Definiramo totalni  $\sigma$ -indeks iregularnosti, v angleščini 'Total  $\sigma$ -irregularity',  $\sigma_t^{f(n)}(G)$  kot:

$$\sigma_t^{f(n)}(G) = \sum_{\{u,v\} \subseteq V(G)} |d_G(u) - d_G(v)|^{f(n)},$$

kjer je  $n = |V(G)|$  in je  $f(n)$  funkcija, definirana za  $n \geq 4$ .

## Izrek

Naj bo  $n \geq 7$ ,  $f(n) \leq \log_3 \left( \frac{3n^2}{3n^2-8} \right)$ , in naj bo  $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$  stopenjsko zaporedje kemijskega grafa  $G$  z maksimalno vrednostjo  $\sigma_t^{f(n)}(G)$ . Potem velja:

1. Če  $n = 4k - 1$ , potem  $a_1 = a_3 = a_4 = k$  in  $a_2 = k - 1$ .
2. Če  $n = 4k$ , potem  $a_1 = a_2 = a_3 = a_4 = k$ .
3. Če  $n = 4k + 1$ , potem  $a_1 = a_2 = a_3 = k$  in  $a_4 = k + 1$ .
4. Če  $n = 4k + 2$ , potem velja bodisi  $a_1 = a_3 = k$  in  $a_2 = a_4 = k + 1$ , bodisi  $a_1 = a_3 = k + 1$  in  $a_2 = a_4 = k$ .

Ker je za kemijske grafe razlika med stopnjami vozlišč omejena, domnevamo naslednje:

- **Domneva 1:** Isti grafi, kot v Izreku, imajo maksimalno vrednost za  $\sigma_t^{f(n)}$ , če je  $f(n) = \frac{1}{n}$ .
- **Domneva 2:** Isti grafi, kot v Izreku, imajo maksimalno vrednost za  $\sigma_t^{f(n)}$ , če je  $f(n) = c$ , kjer je  $c$  konstanta v intervalu  $(0, 1)$ .

## Algoritmi in psevdokode

Za preverjanje domnev sva napisala algoritme, ki generirajo kemijske grafe in izračunajo njihov  $\sigma_t^{f(n)}$  indeks. Najprej sva se lotila sistematičnega iskanja za grafe z  $n$  vozlišči, kjer je  $n \in [7, 14]$ ,  $n \in \mathbb{N}$ . Za večje grafe sva zaradi predolgega trajanja iskanja uporabila algoritma Hill climbing in Simulated annealing. Natančneje za grafe velikosti  $n \in [15, 20] \cup [50, 53] \cup [100, 103]$ .

## Sistematično iskanje

Pri sistematičnem iskanju sva generirala vsa možna stopenjska zaporedja z uporabo knjižnice *nauty\_geng*, za različne  $n$  in izračunala  $\sigma_t^{f(n)}(G)$  za različne  $f(n)$ . Nato sva izbrala tiste grafe, kjer je bila pri določenih  $f(n)$  vrednostjo  $\sigma_t^{f(n)}(G)$  največja.

### Psevdokoda za sistematično iskanje

```
1: function GENERATEUNIQUECHEMICALGRAPHSCONFIGS( $n$ )
2:    $unique\_configs \leftarrow []$ 
3:   for all  $g \in graphs.nauty\_geng(f''n - c - D4'' )$  do
4:      $config \leftarrow DegreeConfiguration(g)$ 
5:     if  $config \notin unique\_configs$  then
6:       Dodaj  $config$  v  $unique\_configs$ 
7:     end if
8:   end for
9:   return  $unique\_configs$ 
10: end function
11: function SIGMATOTALIRREGULARITYFROMCONFIG( $config, f_n$ )
12:   for all  $config \in degree\_list$  do
13:     compute  $\sigma_t^{f(n)}(G)$ 
14:   end for
15:   return ( $config | \max(\sigma_t^{f(n)}(G))$ )
16: end function
```

### Hill climbing algoritem

Hill climbing algoritem je optimizacijski algoritem, ki iterativno izboljšuje trenutno rešitev tako, da na vsakem koraku poišče sosednjo rešitev z boljšo, v najinem primeru večjo,  $\sigma_t^{f(n)}(G)$  vrednostjo. Algoritem sva ustavila po 100000 iteracijah.

### Psevdokoda za Hill climbing algoritem

```
1: function GENERATEINITIALGRAPH( $n$ )
2:   Ustvarimo drevo na  $n$  vozliščih
3: end function
4: function MUTATEGRAPH( $g, u, v$ )
5:   if  $g$  vsebuje povezavo  $(u, v)$  then
6:     Odstrani povezavo  $(u, v)$  iz  $g$ 
7:     if  $g$  je povezan then
8:       return  $g$ 
9:     else
10:      Dodaj nazaj  $(u, v)$ 
11:      return  $g$ 
12:     end if
13:   else
14:     Dodaj povezavo  $(u, v)$  v  $g$ 
15:     if  $\max(\text{degree}(g)) \leq 4$  then
16:       return  $g$ 
17:     else
18:       Odstrani  $(u, v)$ 
19:       return  $g$ 
20:     end if
21:   end if
22: end function
```

```
1: function HILLCLIMBING( $n, f, iterations$ )
2:    $current\_graph \leftarrow \text{GenerateInitialGraph}(n)$ 
3:    $degree\_counts \leftarrow$  seznam stopenj v  $current\_graph$ 
4:   for  $i$  in  $\text{range}(iterations)$  do
5:      $vertices \leftarrow$  seznam vozlišč v  $current\_graph$ 
```

```

6:       $(u, v) \leftarrow$  naključno izbran par vozlišč iz vertices
7:      original_contribution  $\leftarrow 0$ 
8:      for all  $w \in vertices$  do
9:          if  $w \neq u$  then
10:             org_contribution  $+= |degree\_counts[u] - degree\_counts[w]|^{f_n}$ 
11:          end if
12:          if  $w \neq v$  in  $w \neq u$  then
13:             org_contribution  $+= |degree\_counts[v] - degree\_counts[w]|^{f_n}$ 
14:          end if
15:      end for
16:      new_graph  $\leftarrow$  MutateGraph(current_graph,  $u, v$ )
17:      new_degree_counts  $\leftarrow$  seznam stopenj v new_graph
18:      new_contribution  $\leftarrow 0$ 
19:      for all  $w \in vertices$  do
20:          if  $w \neq u$  then
21:             new_contribution  $+= |degree\_counts[u] - degree\_counts[w]|^{f_n}$ 
22:          end if
23:          if  $w \neq v$  in  $w \neq u$  then
24:             new_contribution  $+= |degree\_counts[v] - degree\_counts[w]|^{f_n}$ 
25:          end if
26:      end for
27:      if new_contribution  $>$  original_contribution then
28:          current_graph  $\leftarrow$  new_graph
29:          degree_counts  $\leftarrow$  new_degree_counts
30:      end if
31:  end for
32:   $\sigma_t^{f(n)}(G) \leftarrow 0$ 
33:  for all  $(x, y) \in$  pari vozlišč v current_graph do
34:       $\sigma_t^{f(n)}(G) += |degree(x) - degree(y)|^{f_n}$ 
35:  end for
36:  return (current_graph,  $\sigma_t^{f(n)}(G)$ )
37: end function

```

### Simulated annealing algoritem

Problem pri Hill climbing algoritmu je, da se lahko zatakne v lokalnem maksimumu. Simulated annealing algoritem je pristop, ki se temu izogne, tako da včasih sprejema tudi slabše rešitve. Algoritem sva ustavila po 100000 iteracijah.

### Psevdokoda za Simulated annealing algoritem

```

1: function SIMULATED_ANNEALING( $n, f_n, iterations, T, \alpha$ )
2:      $T = 1$ 
3:      $\alpha = 0.99$ 
4:      $\Delta \leftarrow new\_contribution - original\_contribution$ 
5:     if  $\Delta > 0$  or random()  $<$   $\exp(\Delta/T)$  then
6:         current_graph  $\leftarrow$  new_graph
7:         degree_counts  $\leftarrow$  new_degree_counts
8:     end if
9:      $T \leftarrow T \cdot \alpha$ 
10: end function

```

## Tabele in grafi

Tabela za sistematično iskanje

$n$	$\frac{1}{n}$	0.0001	0.1	0.2	0.45	0.55	0.8	0.9	0.9995
7	(2, 1, 2, 2)	(2, 2, 2, 1)	(2, 1, 2, 2)	(2, 1, 2, 2)	(2, 1, 2, 2)	(3, 1, 1, 2)	(3, 0, 1, 3)	(3, 0, 1, 3)	(3, 0, 1, 3)
8	(2, 2, 2, 2)	(2, 2, 2, 2)	(2, 2, 2, 2)	(2, 2, 2, 2)	(3, 1, 1, 3)	(3, 1, 1, 3)	(3, 1, 1, 3)	(4, 0, 0, 4)	(4, 0, 0, 4)
9	(2, 2, 2, 3)	(2, 3, 2, 2)	(2, 2, 2, 3)	(2, 2, 2, 3)	(3, 2, 1, 3)	(3, 2, 1, 3)	(4, 1, 0, 4)	(4, 1, 0, 4)	(4, 1, 0, 4)
10	(3, 2, 3, 2)	(3, 2, 3, 2)	(3, 2, 3, 2)	(3, 2, 3, 2)	(4, 1, 2, 3)	(4, 1, 2, 3)	(5, 0, 1, 4)	(5, 0, 1, 4)	(5, 0, 1, 4)
11	(3, 2, 3, 3)	(3, 3, 3, 2)	(3, 2, 3, 3)	(3, 2, 3, 3)	(4, 2, 2, 3)	(4, 1, 2, 4)	(5, 0, 1, 5)	(5, 0, 1, 5)	(5, 0, 1, 5)
12	(3, 3, 3, 3)	(3, 3, 3, 3)	(3, 3, 3, 3)	(3, 3, 3, 3)	(4, 2, 2, 4)	(4, 2, 2, 4)	(5, 1, 1, 5)	(5, 1, 1, 5)	(6, 0, 0, 6)
13	(3, 3, 3, 4)	(3, 3, 3, 4)	(3, 3, 3, 4)	(3, 3, 3, 4)	(4, 3, 2, 4)	(4, 2, 2, 5)	(5, 1, 1, 6)	(6, 1, 0, 6)	(6, 1, 0, 6)
14	(4, 3, 4, 3)	(4, 3, 4, 3)	(4, 3, 4, 3)	(4, 3, 4, 3)	(5, 2, 3, 4)	(5, 2, 3, 4)	(6, 1, 2, 5)	(7, 0, 1, 6)	(7, 0, 1, 6)

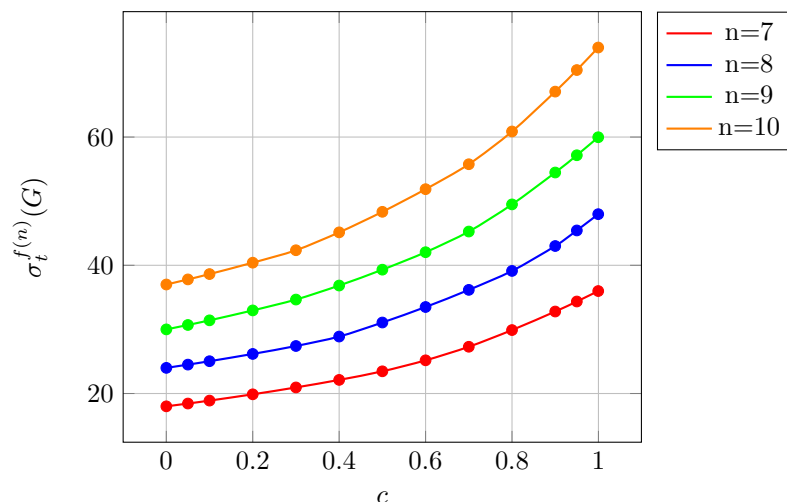
Tabela za Hill climbing algoritem

$n$	$\frac{1}{n}$	0.001	0.1	0.5	0.9	0.995
15	(4, 3, 4, 4)	(4, 3, 4, 4)	(5, 2, 3, 5)	(5, 2, 3, 5)	(6, 1, 2, 6)	(7, 0, 1, 7)
16	(4, 4, 4, 4)	(4, 4, 4, 4)	(5, 3, 3, 5)	(5, 3, 3, 5)	(6, 1, 2, 7)	(6, 1, 2, 7)
17	(4, 4, 4, 5)	(4, 4, 4, 5)	(5, 3, 3, 6)	(6, 3, 2, 6)	(6, 2, 2, 7)	(7, 1, 1, 8)
18	(4, 5, 4, 5)	(5, 4, 5, 4)	(5, 4, 3, 6)	(7, 2, 3, 6)	(7, 2, 1, 8)	(8, 0, 2, 8)
19	(5, 4, 5, 5)	(5, 4, 5, 5)	(6, 3, 4, 6)	(7, 3, 3, 6)	(7, 1, 3, 8)	(9, 0, 1, 9)
20	(5, 5, 5, 5)	(5, 5, 5, 5)	(6, 4, 4, 6)	(7, 3, 3, 7)	(7, 3, 1, 9)	(8, 1, 2, 9)
50	(12, 13, 12, 13)	(12, 13, 12, 13)	(15, 10, 9, 16)	(17, 8, 9, 16)	(21, 4, 3, 22)	(21, 4, 1, 24)
51	(13, 12, 13, 13)	(13, 12, 13, 13)	(16, 9, 10, 16)	(17, 8, 9, 17)	(21, 4, 3, 23)	(24, 1, 0, 26)
52	(13, 13, 13, 13)	(13, 13, 13, 13)	(16, 10, 10, 16)	(17, 9, 9, 17)	(22, 3, 4, 23)	(22, 3, 2, 25)
53	(13, 13, 13, 14)	(13, 13, 13, 14)	(17, 10, 9, 17)	(18, 9, 8, 18)	(23, 3, 3, 24)	(23, 3, 1, 26)
100	(25, 25, 25, 25)	(25, 25, 25, 25)	(31, 19, 19, 31)	(34, 16, 16, 34)	(44, 5, 6, 45)	(47, 3, 1, 49)
101	(25, 25, 25, 26)	(25, 25, 25, 26)	(32, 19, 18, 32)	(34, 17, 16, 34)	(45, 5, 5, 46)	(45, 5, 1, 50)
102	(25, 26, 25, 26)	(25, 26, 25, 26)	(32, 19, 18, 33)	(34, 17, 16, 35)	(46, 4, 6, 46)	(44, 6, 2, 50)
103	(26, 25, 26, 26)	(26, 25, 26, 26)	(27, 24, 25, 27)	(35, 16, 17, 35)	(47, 4, 5, 47)	(49, 2, 1, 51)
1000	(250, 250, 250, 250)	(250, 250, 250, 250)	(264, 236, 236, 264)	(337, 163, 163, 337)	(354, 128, 58, 460)	(353, 112, 37, 498)
1001	(250, 250, 250, 251)	(250, 250, 250, 251)	(264, 236, 236, 265)	(337, 163, 163, 338)	(361, 129, 51, 460)	(361, 112, 31, 497)

Tabela za Simulated annealing algoritem

$n$	$\frac{1}{n}$	0.001	0.1	0.5	0.9	0.995
15	(4, 3, 4, 4)	(4, 3, 4, 4)	(4, 3, 4, 4)	(5, 2, 3, 5)	(6, 1, 2, 6)	(7, 0, 1, 7)
16	(4, 4, 4, 4)	(4, 4, 4, 4)	(4, 4, 4, 4)	(5, 3, 3, 5)	(7, 1, 1, 7)	(6, 1, 2, 7)
17	(4, 4, 4, 5)	(4, 4, 4, 5)	(4, 4, 4, 5)	(6, 3, 2, 6)	(6, 2, 2, 7)	(7, 1, 1, 8)
18	(4, 5, 4, 5)	(5, 4, 5, 4)	(5, 4, 5, 4)	(7, 2, 3, 6)	(6, 2, 2, 8)	(8, 1, 0, 9)
19	(5, 4, 5, 5)	(5, 4, 5, 5)	(5, 4, 5, 5)	(6, 3, 4, 6)	(8, 1, 2, 8)	(8, 1, 0, 10)
20	(5, 5, 5, 5)	(5, 5, 5, 5)	(5, 5, 5, 5)	(7, 3, 3, 7)	(9, 1, 1, 9)	(8, 1, 2, 9)
50	(12, 13, 12, 13)	(12, 13, 12, 13)	(13, 12, 11, 14)	(17, 8, 9, 16)	(20, 4, 4, 22)	(24, 0, 2, 24)
51	(13, 12, 13, 13)	(13, 12, 13, 13)	(14, 12, 12, 13)	(17, 8, 9, 17)	(21, 4, 3, 23)	(23, 1, 1, 26)
52	(13, 13, 13, 13)	(13, 13, 13, 13)	(14, 12, 12, 14)	(17, 9, 9, 17)	(22, 3, 4, 23)	(24, 1, 2, 25)
53	(13, 13, 13, 14)	(13, 13, 13, 14)	(14, 13, 12, 14)	(18, 9, 8, 18)	(23, 3, 3, 24)	(25, 1, 1, 26)
100	(25, 25, 25, 25)	(25, 25, 25, 25)	(26, 24, 24, 26)	(34, 16, 16, 34)	(42, 7, 6, 45)	(46, 3, 2, 49)
101	(25, 25, 25, 26)	(25, 25, 25, 26)	(26, 24, 24, 27)	(34, 17, 16, 34)	(46, 5, 4, 46)	(44, 6, 0, 51)
102	(25, 26, 25, 26)	(25, 26, 25, 26)	(26, 25, 24, 27)	(34, 17, 16, 35)	(45, 6, 5, 46)	(46, 4, 2, 50)
103	(26, 25, 26, 26)	(26, 25, 26, 26)	(27, 24, 25, 27)	(35, 16, 17, 35)	(47, 4, 5, 47)	(42, 9, 0, 52)
1000	(250, 250, 250, 250)	(250, 250, 250, 250)	(264, 236, 236, 264)	(337, 163, 163, 337)	(360, 129, 52, 459)	(351, 111, 41, 497)
1001	(250, 250, 250, 251)	(250, 250, 250, 251)	(264, 236, 236, 265)	(337, 163, 163, 338)	(358, 124, 58, 461)	(356, 112, 34, 499)

Graf  $\sigma_t^{f(n)}(G)$  indeksa v odvisnosti od števila vozlišč  $n$



## Rezultati in ugotovitve

Ugotovila sva, da so največje vrednosti  $\sigma_t^{f(n)}(G)$  za  $f(n) = \frac{1}{n}$  dosežene pri grafih, ki imajo stopenjsko zaporedje enako kot v izreku. Enako velja za  $f(n) = c$ , kjer so vrednosti za  $c$  blizu 0. Nato se na neki točki začneta notranja člena stopenjskega zaporedja zmanjševati, zunanja pa povečevati. Za  $c$  blizu 1 pa so vrednosti  $\sigma_t^{f(n)}(G)$  maksimizirane takrat, ko sta zunanja člena stopenjskega zaporedja  $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$  čim večja, notranja pa čim manjša. Najini algoritmi teh vrednosti niso dosegli, saj sva uporabila omejeno število iteracij. Misliva, da za grafe drži naslednja trditev.

### Trditev

Naj bo  $n \geq 7$ ,  $f(n) = c$ , za  $c$  'zelo blizu' 1 in naj bo  $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$  stopenjsko zaporedje kemijskega grafa  $G$  z maksimalno vrednostjo  $\sigma_t^{f(n)}(G)$ . Potem velja:

1. Če  $n = 4k - 1$ , potem  $a_1 = 2k - 1$ ,  $a_2 = 1$ ,  $a_3 = 0$  in  $a_4 = 2k - 1$ .
2. Če  $n = 4k$ , potem  $a_1 = a_4 = 2k$  in  $a_2 = a_3 = 0$ .
3. Če  $n = 4k + 1$ , potem  $a_1 = a_4 = 2k$  in  $a_2 = 1$  ( $a_3 = 0$ ) ali  $a_3 = 1$  ( $a_2 = 0$ ).
4. Če  $n = 4k + 2$ , potem  $a_1 = 2k$ ,  $a_2 = 1$ ,  $a_3 = 0$  in  $a_4 = 2k + 1$ .