# *Grid-peeling* - kratek opis projekta

Gašper Pust, Mitja Mandić

20. november 2020

# 1 Opis problema

V projektu si bomo podrobneje ogledali konveksne ovojnice $m \times n$ mreže (tako kvadratne kot tudi pravokotne). Konveksna ovojnica množice je najmanjša konveksna množica, ki vsebuje dano množico. Najlažje si jo predstavljamo tako, kot da bi okoli elementov množice napeli elastiko - kar elastika obkroži, je konveksna ovojnica. Lupljenje konveksnih ovojnic mreže, oziroma angleško *grid - peeling* je proces, ko iz mreže iterativno odstranjujemo konveksne ovojnice. S simboli lahko to zapišemo takole: $P_0 = G_{n,m} = \{1, \ldots, n\} \times \{1, \ldots, m\}$. Naj bo $C_i = \mathcal{CH}(P_{i-1})$ za $i = 1, \ldots$. $V_i$ naj bo množica vozlišč $C_i$ - kot vozlišče razumemo točko, ki je na vogalu mreže (torej za katero bi zataknili elastiko). Naj bo sedaj $P_i = P_{i-1} \setminus V_i$. Začnemo torej z $n \times m$ mrežo in iterativno lupimo konveksne ovojnice, dokler ne odstranimo vseh točk.

V projektni nalogi bomo s pomočjo simulacij opazovali v literaturi navedene številke za $n \times n$ mrežo - teorija napoveduje $\theta(n^{\frac{4}{3}})$ ovojnic. Za $n \times m$ mrežo v literaturi ni navedenih podatkov, zanimala nas bo morebitna povezava. Simulacije bomo izvedli tudi za točke na neenakomerni mreži.

# 2 Orodja in algoritmi

Za iskanje ovojnic bova v Pythonu implementirala Jarvis-march (gift - wrapping) algoritem ali Graham-scan. Prvi je enostavnejši, vendar nekoliko počasnejši. Jarvisov algoritem na vsakem koraku pregleda vse točke, ki niso v ovojnici in vanjo doda tiste, ki so najbolj levo in najdlje od trenutne točke. Podrobneje ga bomo predstavili v zaključnem poročilu. Njegova časovna zahtevnost je $\mathcal{O}(nh)$, kjer je $h$ število vozlišč v ovojnici. V najslabšem primeru, ko so v ovojnici vsa vozlišča, ima zahtevnost $\mathcal{O}(n^2)$.

Graham-scan deluje v dveh korakih. Najprej določimo izhodiščno točko (minimalni x,y koordinati) in nato glede na njo izračunamo kote do vseh ostalih vozlišč. Ta uredimo v seznam glede na naraščajoč kot. Če imajo vozlišča enak kot, ohranimo najoddaljenejšo. V drugem delu nato v kopico dodamo prva tri vozlišča iz seznama. Če kot med prvima dvema elementoma kopice in prvim elementom seznama naredi ovinek v desno, prvi element iz kopice nadomestimo s prvim elementom seznama. Nadaljujemo, dokler seznam ni prazen. Vozlišča v kopici predstavljajo konveksno ovojnico. Ta algoritem ima časovno zahtevnost $\mathcal{O}(n \log n)$

Po izvedenem eksperimentalnem delu, bomo rezultate analizirali in jih primerjali z rezultati iz literature. Zanimalo nas bo, kako drugačno je število ovojnic na $m \times n$ mreži v primerjavi s simetrično. Vse skupaj bomo potem primerjali s podatki iz poljubne mreže.