Univerza v Ljubljani
Fakulteta za matematiko in fiziko
Univerzitetni študijski progra Finančna matematika

Pareto fronte v več dimenzijah

Klara Penko, Nejc Jenko

Januar 2022

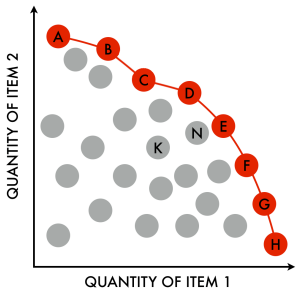
1 Uvod

V projektni nalogi si bova ogledala Pareto fronte na d-dimenzionalnih množicah. To pomeni, da bova na podani množici poiskala točke, ki so nedominirane. Celotna množica nedominiranih točk je Pareto fronta. Za opazovane množice točk bova vzela poljubne točke v d-dimenzionalnih kroglah in hiperkockah. Analizirala bova, število točk v Pareto fronti, glede na dimenzijo prostora d. Poleg prve Pareto fronte, bova želela izračunati še drugo, tretjo in tako naprej.

1.1 Definicija Pareto fronte

Definicija 1 Naj bo S prostor definiran z množico n dimenzij $\{d_1, d_2 \ldots, d_n\}$ in D podmnožica množice S. Naj bo $p \in D$, podana z $p = (p_1, p_2, \ldots, p_n)$, kjer je p_i vrednost v dimenziji d_i. Točka $p \in D$ **dominira** točko $q \in D$ na podprostoru $S' \subseteq S$, če v vsaki dimenziji $d_i \in S'$ velja $p_i \leq q_i$ in v vsaj eni dimenziji $d_j \in S'$, $p_j < q_j$.
Pareto fronta(oz. 1. Pareto fronta) prostora $S' \subseteq S$ je množica točk $D' \subseteq D$, ki niso dominirane z nobeno točko v prostoru S'. Če točke, ki so v 1. Pareto fornti odstranimo iz mnozice dobimo novo množico. Pareto fronto te množice imenujemo **2. Pareto fronta**. Ta postopek lahko n-krat ponovimo, da dobimo **n-to Pareto fronto**. (Od tu naprej uporablajmo izraz ıgloblja ko primerjamo dve Pareto fronti na isti množici. Primer: i-ta Pareto fronta je globlja od j-te Pareto fronta, če velja $i > j$.)

Primer Pareto fronte v dveh dimenzijah



1.2 Hipoteza

1. Predvidevamo, da, če povečujemo dimenzijo, se bo s tem zmanjševalo število točk v globljih Pareto frontah.

2. Predvidevamo, da bodo imele množice, ki imajo možnost za večje odstopanje v pozitivni smeri vseh oz. več koordinat (npr. kocka napram krogli), v 1. Pareto fronti manjše število točk.

2 Potek dela

Za generiranje najinega problema sva uporabila programski jezik Python. Glavni vhodni podatek problema je poljubna množica točk, ki jo lahko ročno izberemo sami, ali pa jo dobimo s pomočjo ene od funkcij, ki sva jih zapisala za generiranje le teh. Zapisala sva funkcijo za izračun Pareto fronte, za katero sva dodatno definirala še eno krajšo funkcijo, kot bomo videli v nadaljevanju. Poleg tega sva zapisala funkciji za generiranje točk v d- dimenzionalni krogli in hiperkocki.

2.1 Funkcije za izračun Pareto fronte

Glavna funkcija najine projektne naloge je funkcija **izracun_pareto_fronte**, ki sprejme množico točki in vrne pareto fronto, ter množico dominiranih točk. Vmes pa se v kodi te funkcije pojavi še funkcija **dominira**. Koda obeh je zapisana spodaj.

```
def dominira(a, b):
    return sum([a[x] >= b[x] for x in range(len(b))]) == len(b)
```

Ta funkcija sprejme dve točki(poljubne iste dimenzije) in vrne **True**, če točka a dominira točko b. Funkcijo **dominira** uporabimo v naslednji kodi.

```
def izracun_pareto_fronte(mnozica):
    pareto_fronta = set()
    dominirane_tocke = set()

    while len(mnozica) != 0:
        kandidat = list(mnozica)[0]
        mnozica.remove(kandidat)
        i = 0
        ni_dominirana = True

        while i < len(mnozica):
            tocka = list(mnozica)[i]
            if dominira(kandidat, tocka):
                mnozica.remove(tocka)
                dominirane_tocke.add(tuple(tocka))
            elif dominira(tocka, kandidat):
                dominirane_tocke.add(tuple(kandidat))
                ni_dominirana = False
                break
            else:
                i += 1

        if ni_dominirana:
```

```
            pareto_fronta.add(tuple(kandidat))

    return pareto_fronta, dominirane_tocke
```

V zgornji fukncij najprej določimo prazne množice za iskano množico Pareto fronta in iskano množico dominiranih točk. Najprej si iz množice, ki jo podamo funkciji izberemo prvega kandidata in ga odstranimi iz prvotne množice. Za tega preverimo ali ta dominira točke, ki so še ostale v množici oziroma če katera od točk iz množice dominira izbranega kandidata. V primeru, da kandidat dominira točko, jo dodamo v množico, ki nam bo vrnila dominirane točke. Ta postopek ponavljamo dokler, ne pregledamo vseh točk iz podane množice (tj. podana množica je prazna). Tako nam zapisana funkcija vrne iskano Pareto fronto, ter množico dominiranih točk.

2.2 Funkcija za izračun n pareto front

Poleg računanja prve Pareto fronte, sva se lotila pisanja kode za funkcijo, ki bo vrnila n Pareto front. Spodnja funkcija, kot argument sprejme poljubno množico točk, ter število n. S številom n povemo koliko Pareto front, bi želeli izračunati. Pareto fronte, tukaj izračunamo rekurzivno s pomočjo zgoraj definirane fukncije izracun_pareto_fronte. Množico zmanjšujemo, dokler ne pridemo do željene Pareto fronte oziroma dokler ne izpraznimo podane možice točk.

```
def izracun_n_pareto_front(mnozica, n):
    if len(mnozica) == 0:
        return set()
    elif n < 1:
        return set()
    else:
        pareto, dom = izracun_pareto_fronte(mnozica)
        print(len(pareto))
        return pareto, izracun_n_pareto_front(dom, n-1)
```

2.3 Funkcija za generiranje točk na d-dimenzionalni enotski krogli

Funkciji **random_krogla** podamo število točk, ki jih želimo zgenerirati in dimenzijo prostora, dodatno lahko spremenimo tudi parameter radij, če bi gledali neenotske krogle (tj. radij krogle različen od 1). V kodi najprej zgeneriramo naključni vektor dimenzije d in ga normiramo. Dobljeni vektor pomnožimo z naključno vrednostjo med 0 in 1, da dobimo točko v notranjosti d dimenzionalne krogle. Na ta način zgeneriramo toliko točk, kot smo podali z atributom stevilo_tock.

```
def random_krogla(stevilo_tock, d, radij=1):
    mnozica_tock = set()
    for i in range(stevilo_tock):
        rand_smer = random.uniform(low=-1, high=1, size=d)
        rand_smer /= linalg.norm(rand_smer, axis=0)
        rand_dolzina = random.uniform(low=0, high=1)
        tocka = tuple(radij * (rand_smer * rand_dolzina))
        mnozica_tock.add(tocka)
    return mnozica_tock
```

2.4 Funkcija za generiranje točk na hiperkocki s stranico a

Podobno, kot smo zgenerirali točke znotraj enotske d-dimenzionalne krogle, zdaj zgeneriramo točke znotraj hiperkocke. Funkciji podamo število točk, dimenzijo prostori in stranico hiperkocke a. Če stranice ne podamo, funkcija privzeto zgenerira točke v enotski hiperkocki. V kodi zgeneriramo naključen vektor velikosti d, z vrednostmi med 0 in a, kar nam predstavlja eno točko. In tako zgeneriramo toliko točk kot smo podali.

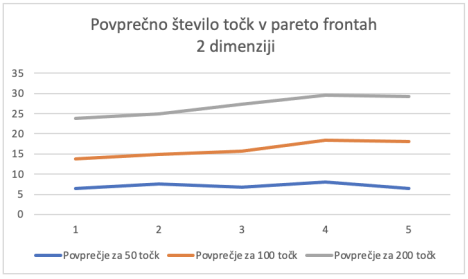
```
def random_kocka(stevilo_tock,d,a=1):
    mnozica_tock = set()
    for i in range(stevilo_tock):
        rand_smer = random.uniform(low=0, high=a, size=d)
        tocka = tuple(rand_smer)
        mnozica_tock.add(tocka)
    return mnozica_tock
```

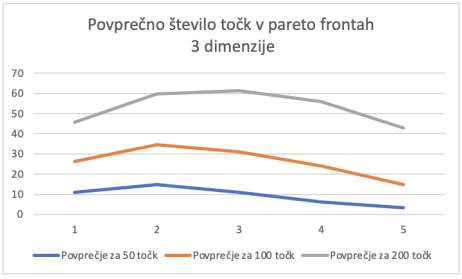
3 Eksperimentalni Del

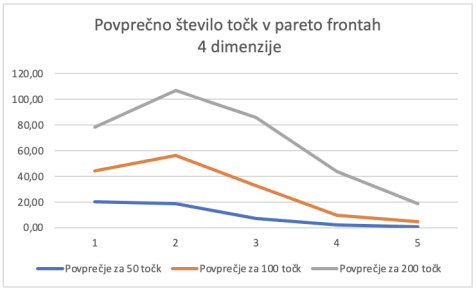
Ko imamo napisane funkcije za generiranje podatkov in njihovo obdelavo, lahko začnemo z eksperimentalnim delom. Naše eksperimente bomo ločili na dva dela. V prvem delu si bomo pogledali vpliv višanja dimenzije prostora. V drugem delu pa bomo v isti dimenziji primerjali rezultate, če testne točke generiramo na različnih območjih (enotska krogla in enotska kocka). Vsi konkretni podatki so dostopni v priloženem .xlsx dokumentu.

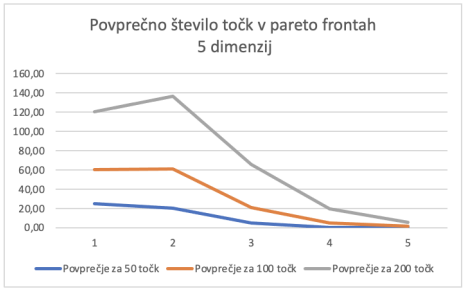
3.1 Prvi Del - Vpliv dimenzije

V prvem delu bomo za svoje podatke generirali 2, 3, 4 in 5 dimenzionalne krogle in primerjali število točk v prvih petih Pareto forntah. Za vsako dimenzijo bomo opazovali množice po 50, 100 in 200 točk. Za vsako od teh kombinacij bomo poskus ponovili desetkrat. V spodnjih grafih so prikazana povprečna števila točk v vsaki Pareto fronti.







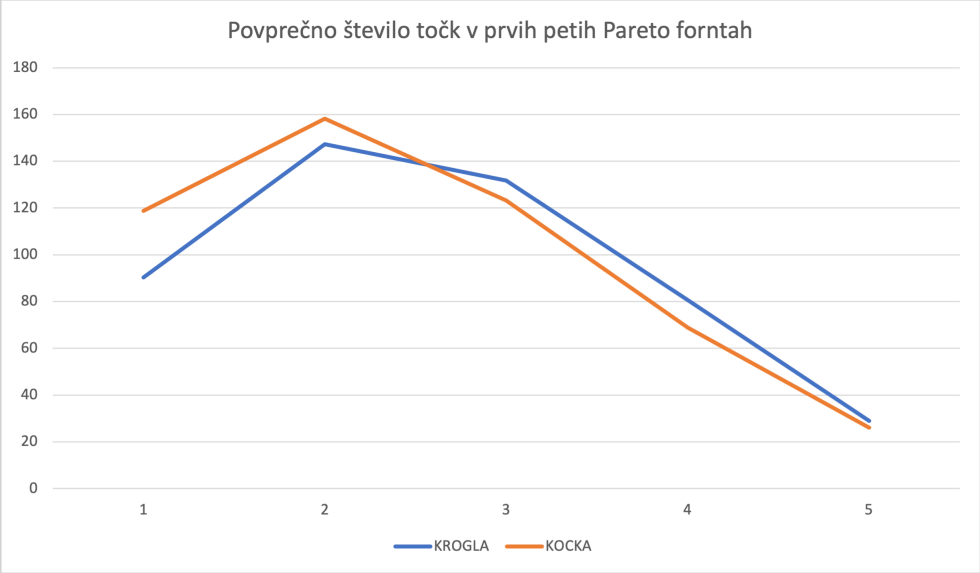


Kot lahko vidimo iz zgornjih grafov, se z višanjem dimencije delež točk v manj globokih Pareto forntah povečuje. Pri dvodimenzionalni krogli imamo v prvih petih Pareto frontah približno enako točk. V naših poskusih jih je v 1. Pareto fronti celo manj točk kot v 5. Pareto fronti. Pričakujemo lahko, da bi z večanjem števila poskusov, ne glede na število generiranih točk, vse krivulje na grafu postajale vedno bolj linearne. Pri povečanju dimenzije na 3 že opazimo trend, da maksimalno število točk dobimo v 2. oz 3. Pareto fronti. To predvidevamo, da je posledica tega, da v 1. Pareto fronti zajamemo najbolj izstopajočje točke, ki dominirajo velik delež ostalih točk. Pri 2. in 3. pareto fronti pa opazujemo že bolj "zglajeno" množico, zato v pareto fronti dobimo več točk. V treh dimenzijah začenjamo opažati tudi vpliv števila generiranih točk. Z več generiranimi točkami se spremembe kažejo bolj intenzivno. Ko dimenzijo povečamo na 4 ali 5 se nam pokaže očiten maksimum števila točk v 2. Pareto fronti. v teh primerih skoraj vse točke množice zajamemo že v prvih treh Pareto frontah. Do tega po našem mnenju pride, ker imamo v več dimenzijah več kriterijov, po katerih mora biti točka dominirana, da izpade iz Pareto fronte.
`Opazimo lahko, da eksperiment potrjuje našo prvo hipotezo.`

3.2 Drugi del - Vpliv oblike množice

V drugem delu si bomo pogledali primerjavo v številu točk v Pareto frontah, če naše testne točke generiramo v enotski krogli ali pa v hiperkocki s stranico dolžine 1. Pri eksperimentu bomo za vsak primer izvedli 20 poskusov. Tokrat bomo poskuse delali z 500 točkami in v 5 dimenzijah. 5 dimenzij smo izbrali, ker se v tem primeru že bolje vidi upad števila točk v globjih Pareto frontah.

Ker pa je ta upad tako velik, smo število generiranih točk povečali na 500. S tem smo želeli preprečiti, da bi že v manj kot 5 Pareto frontah zajeli vse točke iz naše testne množice.



Kot lahko vidimo iz zgornjega grafa, do največjega odstopanja pride v 1. Pareto fronti, kjer je pri množici generirani v hiperkocki v povprečju skoraj 30 točk več, kot pri množici generirani v krogli. To je v nasprotju z našimi pričakovanji. Pričakovali smo, da bodo točke, ki bodo v kotu hiperkocke z največjimi vrednostmi v vseh koordinatah dominirale velik del množice in zato zmanjšale število točk v 1. Pareto fronti.
Prav tako je tudi pri pregledu 2. Pareto fronte še vedno več točk pri hiperkocki. Trend se obrne šele v 3. Pareto fronti, od koder obe krivulji enakomerno padata.
`Opazimo, da se eksperiment ne sklada z našo drugo hipotezo.`

3.3 Časovna zahtevnost

Naj bo n število generiranih točk, d dimenzija prostora in m globina Pareto front.

- Za generiranje točk v krogli uporabljamo `for` zanko z n ponovitvami. V vsaki ponovitvi opravimo d korakov za generiranje vektorja, $2d+1$ korakov za računanje norme, d korakov za deljenje vektorja z normo, 1 korak za generiranje dolžine, $2d$ korakov za množnje vektorja z dolžino in radijem in 1 korak za dodajanje točke v množico.
  Pri generiranju točk v hiperkocki prav tako uporabimo n ponovitev zanke. V tem primeru pa v vsaki ponovitvi izvedemo le d korakov za generiranje točke in 1 korak da točko dodamo v množico.

Torej, pri generiranju podatkov v obeh primerih za izvedbo funkcije potrebujemo $O(nd)$ korakov.

- Pri funkciji za računanje Pareto fronte uporabimo 2 gnezdeni `while` funkciji. Iz ustavitvenih pogojev lahko ugotovimo, da bo maksimalno število ponovitev prve zanke n, maksimalno število ponovitev druge zanke pa $n-1$. V drugi zanki nato dvakrat uporabimo funkcijo `dominira`, za katero potrebujemo d korakov. Skupno število korakov je torej:

  $$n(4 + (n-1)(1+d+d+1) + 1 + 1) + 1 = 2n^2d + 2n^2 - 2nd + 4n + 1.$$

  V skupnem lahko velikostni razred označimo z $O(n^2d)$. V realnosti je število korakov lahko precej manjše, saj lahko v vsaki ponovitvi odstranimo več kot 1 točko.

- Pri računanju m-te Pareto fronte m-krat rekurzivno uporabimo funkcijo. Torej je velikostni razred časovne zahtevnosti $O(n^2dm)$. V realnosti je tudi tu zahtevnost manjša, saj v vsaki ponovitvi rekurzije delamo s precej manjšo množico točk.