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.

2 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** prostora $S' \subseteq S$ je množica točk $D' \subseteq D$, ki niso dominirane z nobeno točko v prostoru S'.

Primer Pareto fronte v dveh dimenzijah

3 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.

3.1 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 fukniji 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.

3.2 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 defini-

rane 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)
```

3.3 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
```

3.4 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
```