

Univerza v Ljubljani  
Fakulteta za *matematiko in fiziko*



# Linearno programiranje

2. naloga pri Modelske analizi 1

**Avtor:** Marko Urbanč (28232019)  
**Predavatelj:** prof. dr. Simon Širca  
**Asistent:** doc. dr. Miha Mihovilovič

17.10.2023

# Kazalo

|   |           |
|---|-----------|
| <b>1 Uvod</b>   | <b>2</b>  |
| <b>2 Naloga</b>   | <b>2</b>  |
| <b>3 Opis reševanja</b>   | <b>3</b>  |
| 3.1 PuLP: basic cookbook . . . . .  | 3         |
| <b>4 Rezultati</b>  | <b>5</b>  |
| 4.1 Osnovni model . . . . .   | 5         |
| 4.2 Osnovni model brez omejitve mase . . . . .                                  | 5         |
| 4.2.1 Osnovni model z omejitvijo mase . . . . .                                 | 6         |
| 4.2.2 Osnovni model z omejitvijo mase in dodatnimi omejitvami                   | 7         |
| 4.3 Dieta z manj maščobami . . . . .  | 9         |
| 4.3.1 Dieta z manj maščobami in brez omejitvije mase . . . . .                  | 9         |
| 4.3.2 Dieta z manj maščobami z dodatnimi omejitvami . . . . .                   | 10        |
| 4.3.3 Dieta z manj maščobami in omejitvijo mase . . . . .                       | 10        |
| 4.3.4 Dieta z manj maščobami, omejitvijo mase in dodatnimi omejitvami . . . . . | 11        |
| 4.4 Optimizacija cene . . . . .   | 12        |
| 4.4.1 Čim cenejša dieta, ki zadosti osnovnim potrebam . . . . .                 | 12        |
| 4.4.2 Čim cenejša dieta, z dodatnimi zahtevami . . . . .                        | 13        |
| 4.4.3 Čim cenejša dieta, z dodatnimi zahtevami in omejitvijo kalorij . . . . .  | 13        |
| 4.4.4 Namenski bon za 15 EUR . . . . .  | 14        |
| 4.4.5 Namenski bon za 15 EUR z dodatnimi zahtevami in omejitvijo mase . . . . . | 15        |
| <b>5 Expanded dataset</b>   | <b>16</b> |
| 5.1 Predprocesiranje podatkov . . . . .   | 16        |
| 5.2 Rezultati . . . . .   | 19        |
| <b>6 Komentarji in izboljšave</b>   | <b>20</b> |
| <b>Literatura</b>   | <b>21</b> |

## 1 Uvod

Linearna optimizacija je zadnje čase zelo vroča tema, tako da me veseli, da sem se je lahko lotil tudi sam. Linearno programiranje oz. linearna optimizacija je metoda za iskanje maksimuma ali minimuma linearnega izraza, ki je podvržen linearnim omejitvam. Omenjen linearni izraz je funkcija več spremenljivk, ki najpogosteje ovrednoti oz. oceni neko količino, ki jo želimo optimizirati. To je lahko npr. dobiček, strošek, količina proizvodnje, itd. zato to funkcijo imenujemo **cost function** oz. **objective function**. Linearno programiranje je torej metoda, ki nam omogoča, da z linearno funkcijo in linearimi omejitvami poiščemo optimalno rešitev. Uporablja v različnih panogah, kot so ekonomija, logistika, telekomunikacije, transport, kot pa tudi v znanosti, kot je npr. fizika.

Torej če si to pogledamo v matematični notaciji imamo našo cost funkcijo  $f(x_1, x_2, \dots, x_n)$ , definirano kot neko linearno kombinacijo spremenljivk:

$$f(x_1, x_2, \dots, x_n) = c_1x_1 + c_2x_2 + \dots + c_nx_n . \quad (1)$$

in set vezi, ki so pogosto izražene kot neenačbe:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &\leq b_1 , \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &\leq b_2 , \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &\leq b_m . \end{aligned} \quad (2)$$

To je pravzaprav to kar se tiče matematičnega opisa osnovnega problema. Seveda so potem izvedenke postopkov oz. algoritmov malo bolj zapletene, ampak to žal ni namen te naloge.

Dotično pri tej nalogi si bomo pogledali problem optimizacije diete. Posebna omemba ni potrebna, kako pomembno je to, da se prehranjujemo zdravo in uravnoteženo. Vendar pa je to v današnjem času vse težje, saj je na voljo ogromno različnih živil, ki so vse prej kot zdrava. Zato je toliko bolj pomembno, da se zavedamo, kaj jemo in da se prehranjujemo zdravo. Vendar pa je to včasih težko, saj je veliko ljudi prezaposlenih in nimajo časa, da bi se ukvarjali s tem. Skratka ideja je taka, da bomo za dane želje kar se tiče hranilnih snovi, poskusili najti optimalno dieto. Optimalno pa je tu mišljeno v tem smislu, kot ga uporabnik določi. Zagotovo bo razlika med tem, ali se minimizira količina hrane, ki jo je potrebno pojesti, cena hrane, količina določene hranilne snovi ali pa kalorična vrednost.

## 2 Naloga

Naloga torej zahteva, da za dano tabelo živil in njihovih hranilnih vrednosti, najdemo optimalno dieto, kjer minimiziramo kalorije, če zahtevamo sledeče:

1.
  - Vsaj 70 g maščob
  - Vsaj 310 g ogljikovih hidratov

- Vsaj 50 g proteinov
- Vsaj 1000 mg kalcija
- Vsaj 18 mg železa
- Dnevni obroki naj ne presežejo 2000 g

Ta osnovni model lahko potem nadgradimo s tem da dodamo še:

- Vsaj 60 mg Vitamina C
  - Vsaj 3500 mg Kalija
  - Med 500 mg in 2400 mg Natrija
2. Potem pa poglejmo kako se rezultat razlikuje še zahtevamo vsaj 2000 kcal in minimiziramo vnos maščob.
  3. Minimizirajmo zdaj ceno diete
  4. Dodatne izvedbe omejitev za izboljšave uravnoteženosti diete

### 3 Opis reševanja

Reševanja sem se lotil v klasično v Pythonu. Pri tej nalogi je močno prišel v upoštev pythonian princip, da za vse že obstaja neka knjižnica. In res je tako, za reševanje linearne optimizacije obstaja knjižnica PuLP [1], ki je zelo enostavna za uporabo in ima še kar solidno dokumentacijo. Pravzaprav je njihov prvi demonstracijski zgled zelo podoben našemu, z razliko da oni mešajo meso v mačji hrani. Poleg tega pa seveda pridejo zraven še standardni `numpy`, `pandas` in takrat zaradi malo alterative izbire vizualizacije še `holoviews`.

#### 3.1 PuLP: basic cookbook

Kot sem že omenil, je knjižnica PuLP zelo enostavna za uporabo. Lahko na hitro povzamem postopek ker tako ali tako trenutno čakam, da se mi predprocesirajo podatki za (hopefully uspe) zadnji del naloge. Torej začnemo s tem da uvozimo knjižnico in definiramo problem.

```
from pulp import *
import pandas as pd

# Create the "prob" variable to contain the problem data
prob = LpProblem("Diet Problem", LpMinimize)
```

Sledi nalaganje podatkov iz tabele živil in ustvarjanje spremenljivk. Spremenljivke so v tem primeru količina živila, ki ga bomo pojedli.

```

df = pd.read_table('Data/table.dat', sep=',', skiprows=2, index_col=0)

# Create a list of the food items
food_items = list(df.index)

# Create variables. These variables are the amounts of each food item to
# buy
food_vars = LpVariable.dicts("Food", food_items, lowBound=0,
                             cat='Continuous')

```

Okay sedaj pa definiramo še našo cost funkcijo. V tem osnovnem primeru bo to količina kalorij, ki jih bomo zaužili.

```

# Define objective function and add it to the problem
prob += lpSum([df.loc[i, 'Energija[kcal]'] * food_vars[i] for i in
              food_items]), "Total energy intake per person"

```

Zelo praktično je, da v bistvu kar prištevamo izraze k našemu problemu. Tako lahko zelo enostavno dodajamo vezi. Storimo to.

```

# And now we can add the constraints
prob += lpSum([df.loc[i, 'Mascobe[g]'] * food_vars[i] for i in
              food_items]) >= 70, "FatRequirement"
prob += lpSum([df.loc[i, 'Ogljikovi_Hidrati[g]'] * food_vars[i] for i in
              food_items]) >= 310, "CarbohydrateRequirement"
prob += lpSum([df.loc[i, 'Proteini[g]'] * food_vars[i] for i in
              food_items]) >= 50, "ProteinRequirement"
prob += lpSum([df.loc[i, 'Ca[mg]'] * food_vars[i] for i in food_items])
            >= 1000, "CalciumRequirement"
prob += lpSum([df.loc[i, 'Fe[mg]'] * food_vars[i] for i in food_items])
            >= 18, "IronRequirement"
prob += lpSum([100 * food_vars[i] for i in food_items]) <= 2000,
            "MassLimit"

```

Zdaj ko smo model tako lepo definirali je smiselno, da ga tudi spravimo v datoteko. Nato pa kličemo reševanje. PuLP bo sam poskrbel za izbiro algoritma, ki bo najbolj primeren za naš problem, večina algoritmov je implementiranih v COIN-OR [2] knjižnici.

```

model_name = "diet-model_no-weight-con.lp"
prob.writeLP(f"Models/{model_name}")

# Slove the problem
prob.solve()

```

In na koncu še spravimo rešitev v neko obliko, ki jo lahko potem uporabimo za vizualizacijo.

```

# Save the solution to a file with numpy
var_names = np.array([v.name for v in prob.variables()])
var_values = np.array([v.varValue for v in prob.variables()])
solution = np.column_stack((var_names, var_values))

np.savetxt(f"Solutions/{model_name}-sol.dat", solution, delimiter=",",
→ fmt="\%s", header="Item,Value")

```

Če bi nas zanimalo še takojšnje ovrednotenje lahko z spodnjim blokom kode izpišemo nekaj uporabnih informacij.

```

# Print the status of the solution
print("Status:", LpStatus[prob.status])

# Print the optimal solution
for v in prob.variables():
    print(v.name, "=", v.varValue)
print("Total energy intake per person = ", value(prob.objective))

```

Od tod naprej pa lahko poljubno zapletemo vezi (dokler ostanejo linearne seveda) ali pa spremenimo vhodne podatke.

## 4 Rezultati

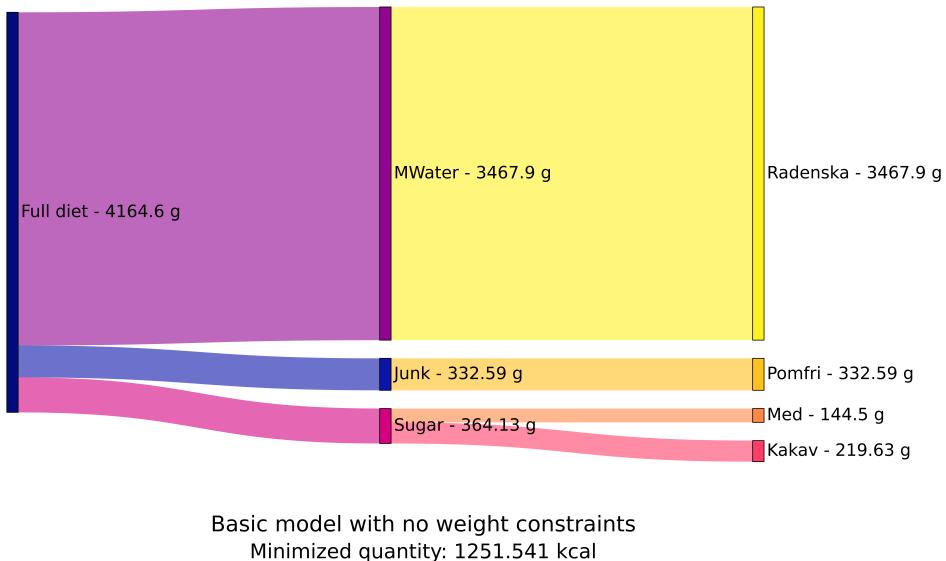
Tole poročilo bo res med krajšimi, kar sem jih napisal, kar se tiče surove vsebine, ampak dejstvo je, da je narava naloge taka, da gremo lahko takoj na zabavne rezultate.

### 4.1 Osnovni model

### 4.2 Osnovni model brez omejitve mase

Najprej si poglejmo rezultate osnovnega modela, kjer ne omejujemo mase. Torej lahko pojemo kolikor želimo. Rezultati so prikazani na sliki (1).

Radenska really does give you three hearts!



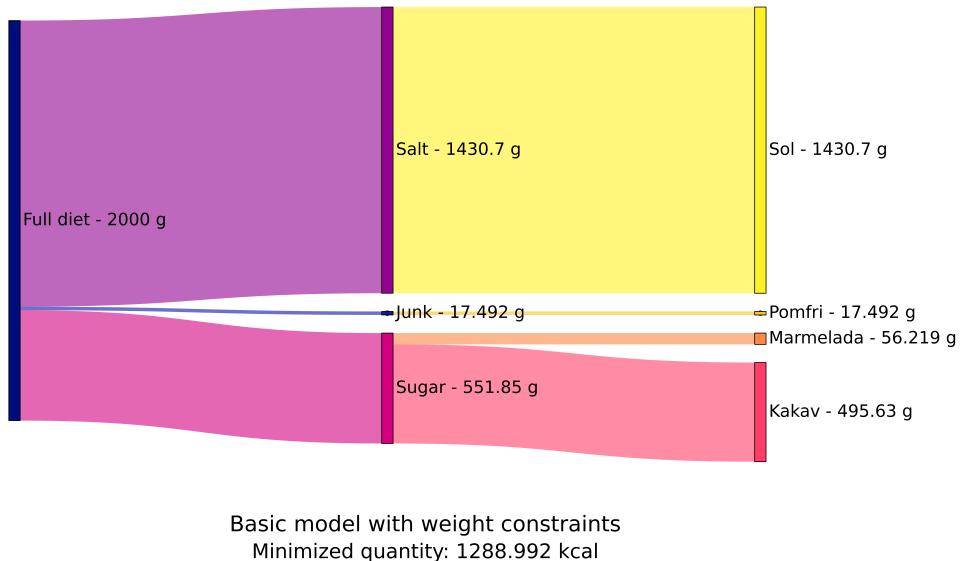
Slika 1: Osnovni model brez omejitve mase

Tu smo dobili znanstveni dokaz, da nam radenska res da tri srčke. Presenetljivo številke niso tako nerealistično velike kot bi lahko bile in kot bomo videli, da bodo, tako da bi dieta, ki bi slonela na radenski, pomfriu, medu in kakavu lahko bila še kar obetavna. Radenska je fenomenalna izbira, ker optimiziramo kalorije, ki jih zaužijemo. Voda ie. radenska pa ima zelo malo kalorij, tako da je to zelo dobra izbira, hkrati pa ima neko količino mineralov, ki jih potrebujemo in tako prevlada v izboru.

#### 4.2.1 Osnovni model z omejitvijo mase

Sedaj pa si poglejmo še rezultate, kjer omejujemo maso na 2000 g. Rezultati so prikazani na sliki (2).

Salt supremacists unite. For rock and stone!



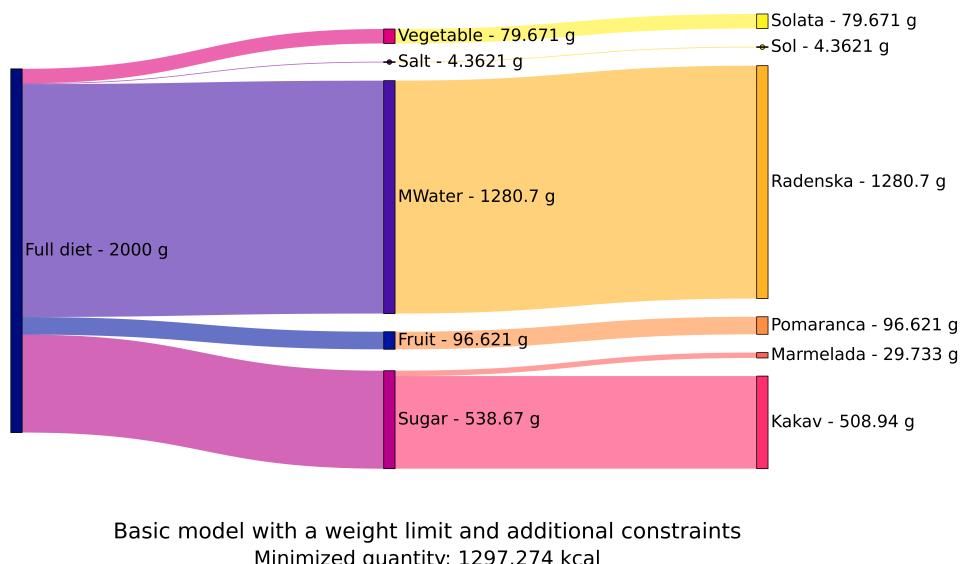
Slika 2: Osnovni model z omejitve mase

Meni se zdi neznosno amusing rezultat, da je glede na naše preproste vezi, optimalna dieta takšna, da se pravzaprav ubiješ s soljo. Dodatno pa ješ kakav za mineralni vnos. To bi lahko bil jedilnik astronautov v distopični prihodnosti, kjer smo nekako ugotovili, kako varno lahko jemo ogromne količine soli. Simpatično je, da imamo tudi za okras in okus malo pomfrija. Mogoče so pa to your average *Mc D's* pomfriji, kjer je verjetno brez šale na 17 g pomfrija 1.4 kg soli. Sicer pa verjetno je tu argument podoben kot pri radenski. Kar naenkrat rabimo nekaj kar je mineralno bolj gosto in kaj je boljše kot mineralna voda? Mineral sam. Tudi sol ima precej nizko kalorično vrednost, tako da je to zelo dobra izbira. O smrtnih posledicah hipernatriemije pa ne bi.

#### 4.2.2 Osnovni model z omejitvijo mase in dodatnimi omejitvami

Zdaj pa izključimo to zoprno sol in dodajmo še tiste dodatne omejitve, kjer si postavimo nek smiselen interval za količino natrija v naši hrani. Ob tem pa lahko še dodamo par vezi, ker lahko. Rezultati so prikazani na sliki (3).

## A 'passable' diet



Slika 3: Osnovni model z omejitve mase in dodatnimi omejitvami

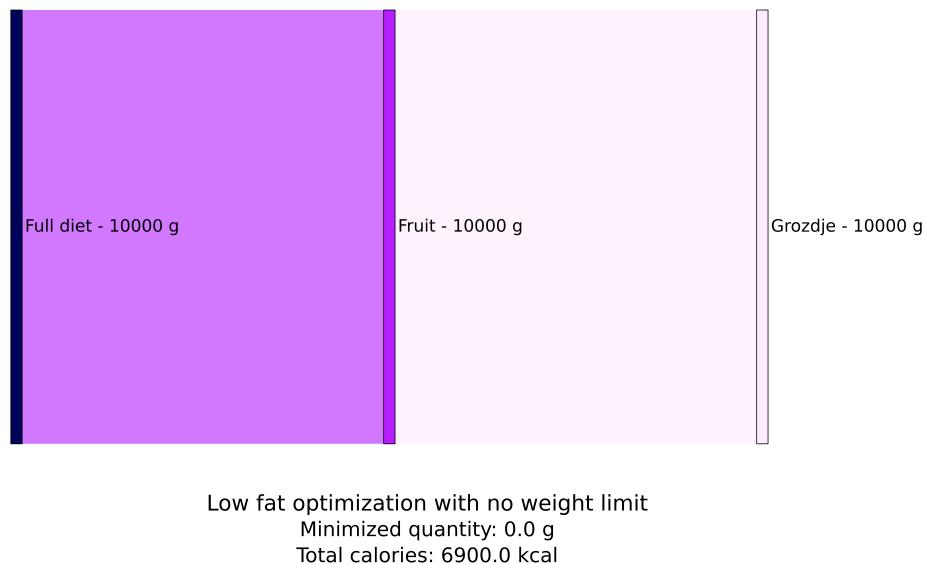
Opa, to pa je že nekaj kar bi lahko človek rekel, da je popolnoma passable. Dober liter radenske se ne zdi pretiravanje, če te mehurčki ne motijo. Potem pa malo solate, malo sadja in seveda (tukaj odpove heh..) pol kile kakava. Očitno sta kakav in radenska še vedno zelo kalorično učinkovita.

## 4.3 Dieta z manj maščobami

### 4.3.1 Dieta z manj maščobami in brez omejitvije mase

Zamaščena jetra? Ni problema! Naši eksperti (opice za računalnikom) so pravili dieto ki bo vaša jetra očistila v trenutku. Rezultati so prikazani na sliki (4).

Fatless with boring grapes



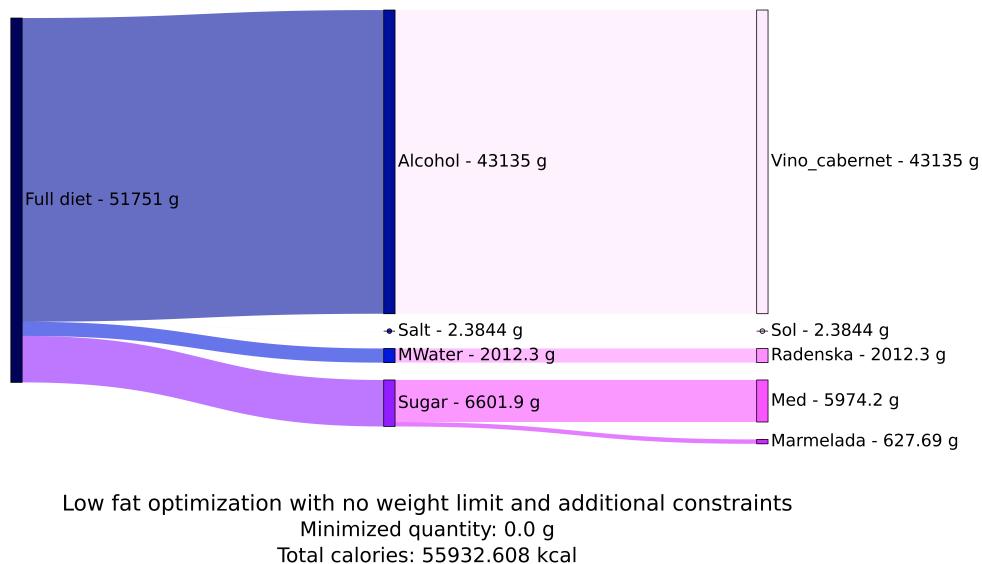
Slika 4: Dieta z manj maščobami in brez omejitvije mase

Enkratno. Ena gajba grozdja na dan in boste kot novi. Šalo na stran je zanimivo, da je za naše osnovne omejitve očitno grozdje dovolj hranilno v vseh kategorijah, če ga poješ dovolj. To je torej prava space-age hrana. Or rather nek koncentrat grozdja.

#### 4.3.2 Dieta z manj maščobami z dodatnimi omejitvami

Ojoj zdravnik vam je rekel, da imate zamaščena jetra zaradi prekomernega pijača alkohola. Pa kolega, res ni problem. Naši pridni eksperti imajo tudi za to popolno dieto. Rezultati so prikazani na sliki (5).

Fatless with lots o' fun grapes: An alcoholics paradise



Slika 5: Dieta z manj maščobami z dodatnimi omejitvami

Saj vete kako pravijo eh? *Klin se s klinom zbij, kaj ti škodi še liter vina?*. You heard correctly, efekte svoje crippling zasvojenosti z alkoholom lahko po naši briljantni dieti odpravite s tem, da pijete še več alkohola. Pri tej dieti nikoli ne bote lačni saj je kalorično zelo učinkovita in garantiramo vam, da se vam bo zdel svet takoj lepši. Zabavno mi je, da se spet pojavi grozdje, le v drgačni obliki. Očitno je, da je grozdje res superhrana.

#### 4.3.3 Dieta z manj maščobami in omejitvijo mase

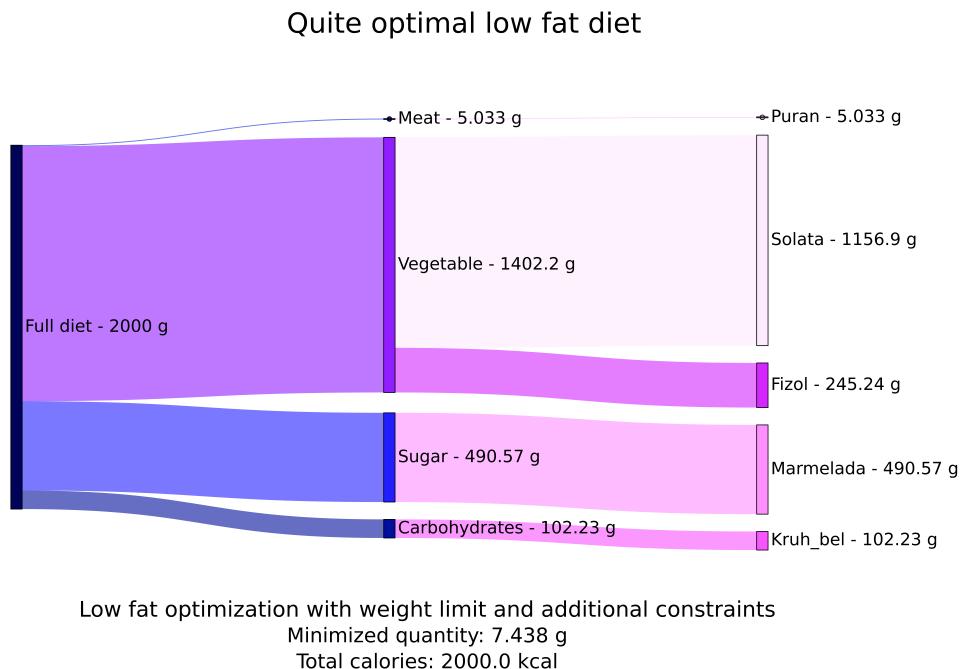
Zaradi več pritožb strank, da se počutijo, kot da se prenajedajo (in morebitno kakšne smrti zaradi zastrupitve z alkoholom) smo našim opicam.. ah mislim ekspertom razložili približno koliko banan lahko tipičen človek poje na dan. Skovali so dieto, ki je prikazana na sliki (??).

Ah ja, fantastično. Solata z marmelado je v nekaterih državah verjetno že tradicionalna jed in lepo je, da naši eksperti pustijo tudi nekaj lean mesa, kot je puran. Opazka tu je, da se grozdje sploh ne pojavi več. Očitno je super le za

neomejeno dieto, a v bolj realističnem primeru je gostota hranih snovi prenizka.

#### 4.3.4 Dieta z manj maščobami, omejitvijo mase in dodatnimi omejitvami

Za naše zveste stranke smo prejšno uspešnico diete z manj maščobami še malo izboljšali. Glavna pritožba je bila, da vsebuje bistveno preveč sladkorja. Zdaj inzulin se menda lahko vedno samo vbrizga v žilo, tako da to ni res problem, ampak vseeno. Rezultati so prikazani na sliki (6).



Slika 6: Dieta z manj maščobami, omejitvijo mase in dodatnimi omejitvami

S to izboljšavo boste jedli še bolj zdravo saj boste lahko zaužili še 75 g fižola več. Žal pa se bo treba odpovedati večini purana. Ampak ker so eksperti milostljivi so ga pustili za eno čajno žličko, da je za okus na koncu. Kakšnih drastičnih sprememb ni, le malo smo prerazporedili količine hrane..

*Tok. Tok. Dum. FBI open up!*



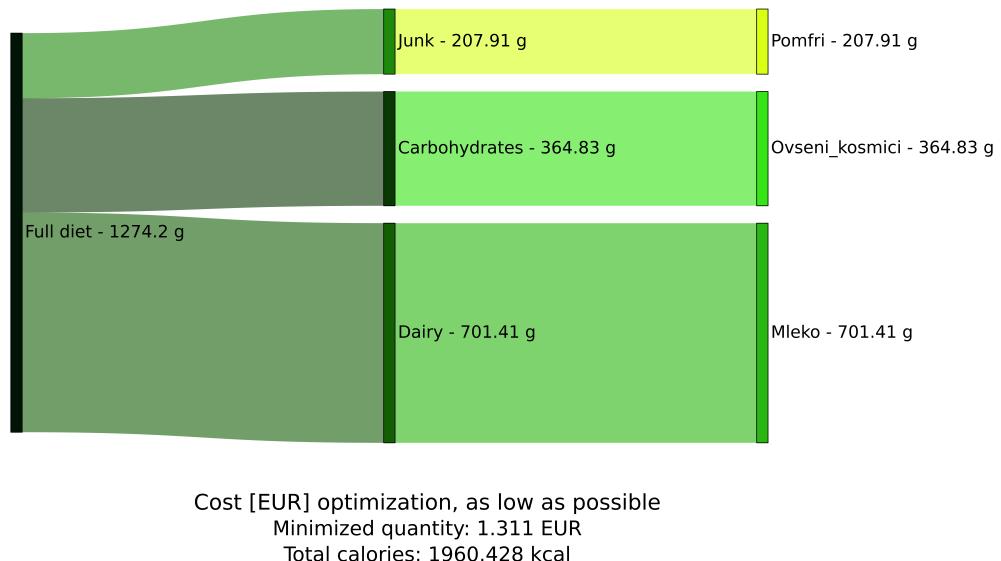
Oh shit aaaa hitro skrij opice. Beživa!

#### 4.4 Optimizacija cene

##### 4.4.1 Čim cenejša dieta, ki zadosti osnovnim potrebam

Poiskimo čim cenejšo dieto, ki zadostuje osnovnim potrebam kar se tiče hraničnih snovi. Rezultat je prikazan na sliki (7).

Mleko in ovseni kosmiči, a kalorije so višje



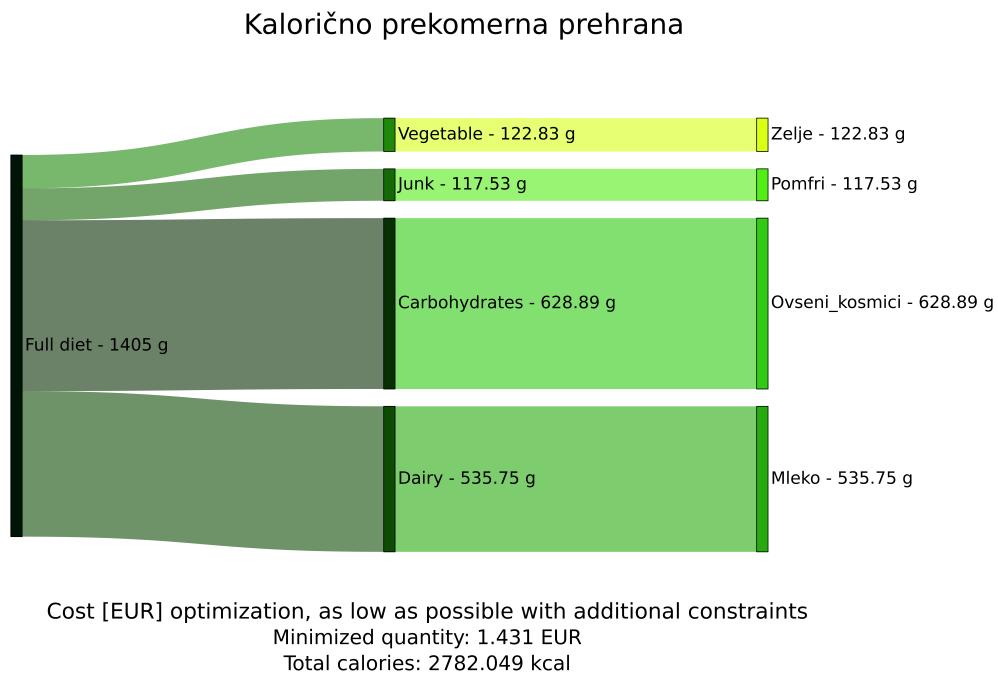
Slika 7: Čim cenejša dieta

Hja.. pravzaprav je kot nekako pričakovano. Ovseni kosmiči in mleko sta res med cenejšima izdelkoma in za zagotovitev soli še ena majhna porcija pomfrija, ki je tudi še kar cenjen. Kar je tudi vredno omembne je, da so kalorije za takšno

dieto takoj višje, kar je znan efekt v dandanašnjem svetu. Dejansko je ceneje jesti nezdravo hrano.

#### 4.4.2 Čim cenejša dieta, z dodatnimi zahtevami

Poglejmo še kaj bi dobili, če bi zraven še upoštevali še naše dodatne zahteve. Rezultat je prikazan na sliki (8).



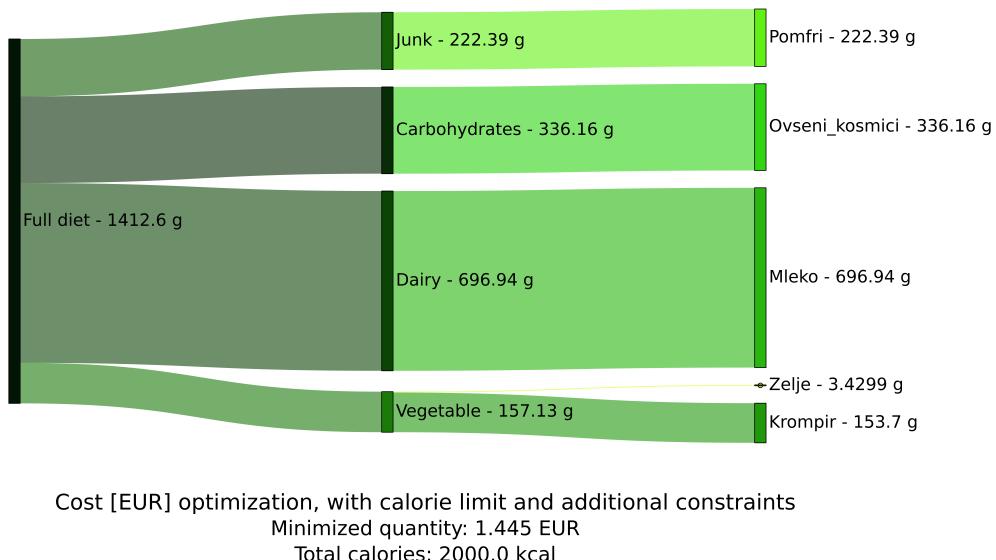
Slika 8: Čim cenejša dieta z dodatnimi zahtevami

Kar smo zdaj dobili je hranilno bolj popolno, a bi bilo na dolgi rok, vsaj za večino ljudi, škodljivo saj vsebuje preveč kalorij.

#### 4.4.3 Čim cenejša dieta, z dodatnimi zahtevami in omejitvijo kalorij

Kot rečeno prejšnja dieta vsebuje preveč kalorij, torej lahko to poskusimo praviti, tako da omejimo kalorije. Rezultat je prikazan na sliki (9).

## Sprejemljiva poceni dieta



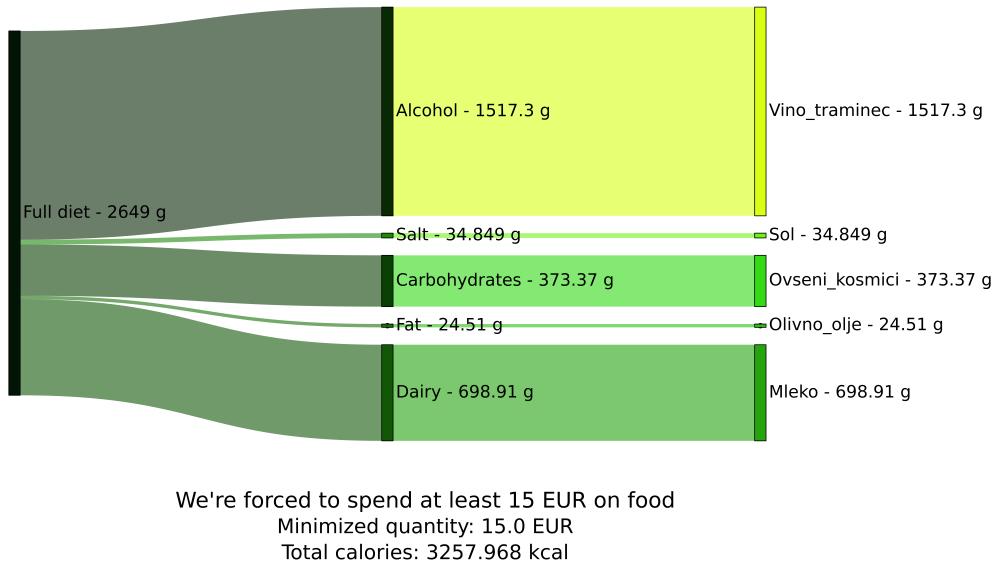
Slika 9: Čim cenejša dieta z dodatnimi zahtevami in omejitvijo kalorij

In tu smo dobili nekaj kar je popolnoma uporabno. Vsaj meni se zdi upgrade no-brainer, ker z to dieto lahko jemo po omejitvi kalorij in to le za dober cent na dan več. Zdaj vprašanje je bolj če bi ta cent razlike se res ohranil, ker se običajno različne količine hrane v resnici prodajajo po različnih cenah, torej če bi npr. kupili manj mleka, bi bilo to verjetno dražje.

### 4.4.4 Namenski bon za 15 EUR

Od šefa v službi smo prejeli namenski bon za 15 EUR v naši najljubši živilski trgovini. Kaj bi bilo najbolj optimalno, da kupimo. Rezultati osnovnih zahtev so prikazani na sliki (10).

## Vina in mleka naj teče reka



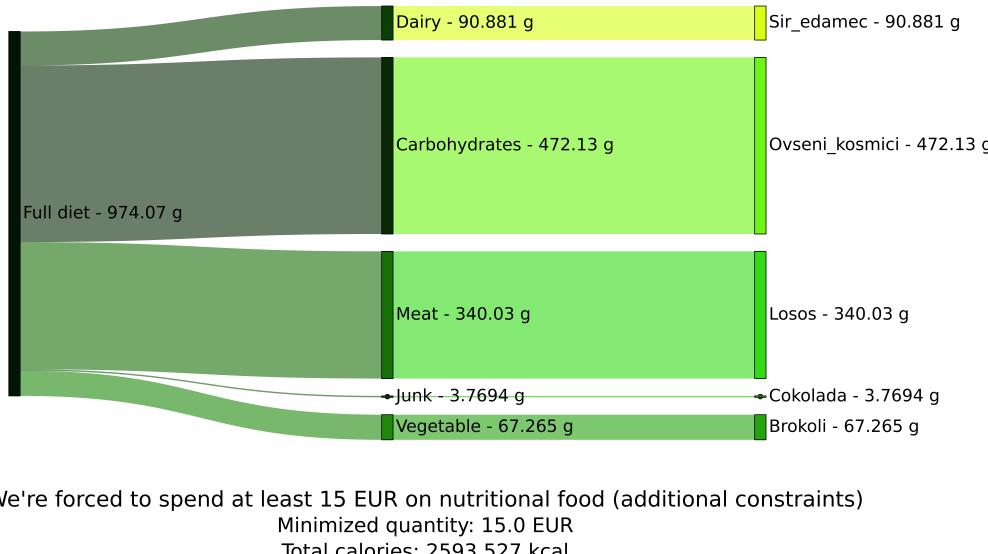
Slika 10: Namenski bon za 15 EUR

Ah ja tu bi se zabavali. Tako dieto bi vam priporočal moj daljni stric iz devete vasi v Ameriki (za katerega sem slišal, da so ga ravno zaprli.. nekaj z nekimi opicami in reklamami). Kakorkoli, tu se spet pojavi grozdje, ker nisem dal omejitve na maso. Overall bi sicer bila to kar slaba dieta, ampak bi bil človek mogoče preveč pijan, da bi opazil.

### 4.4.5 Namenski bon za 15 EUR z dodatnimi zahtevami in omejitvijo mase

Kaj pa če bi zraven še upoštevali naše dodatne zahteve in omejili maso. Rezultati so prikazani na sliki (11).

## Kupil bi losos



Slika 11: Namenski bon za 15 EUR z dodatnimi zahtevami in omejitvijo mase

Ej takšno dieto bi pa tudi jaz z veseljem jedel. Losos je super, brokoli je mega in edamec je tudi fin. Zraven pa skuhaš kašo iz ovsenih kosmičev. To je verjetno najbolj kalorično učinkovita dieta, ki sem jo dobil thus far, saj je hrane vse skupaj manj kot kilogram, pa kljub temu presežemo 2000 kcal. Grozno kako hitro se kalorije naberejo.

## 5 Expanded dataset

Da bi dobili še malo bolj zanimive rezultate, sem se odločil, da bom podatke malo razširil. S tem mislim, da sem na internetu staknil podatkovno bazo od USDA (*United States Department of Agriculture*), ki ima podatke o hranilnih vrednostih za ogromno živil. Vzel sem le živila, ki imajo zraven še znamko. Vse skupaj znaša dobrih 400 tisoč živil.

### 5.1 Predprocesiranje podatkov

The name of the game pri ogromnih podatkovnih setih, še sploh pa pri nesandardnih tabelah je predprocesiranje podatkov. To sem prvotno počel vse lokalno na svojem računalniku, a sem se, ko se je obdelava zapletla, hitro naveličal čakanja. Računalnik je tako ali tako uporabljal samo eno jedro. Čas je bil za MPI in prehod na moj erm.. **ghetto HPC sistem** oz. ljubkovalno mu

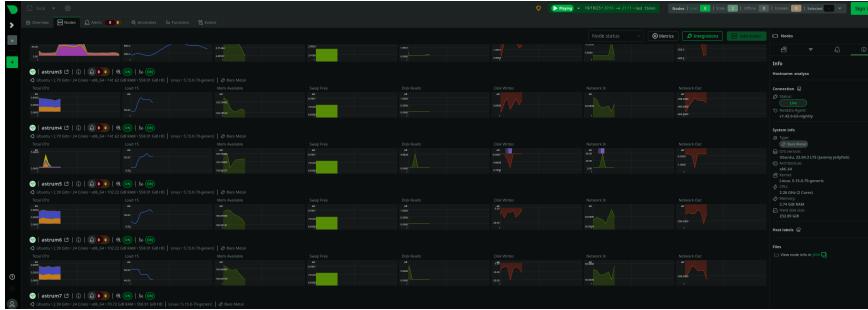
je ime **Astrum**.



Slika 12: **Astrum** server room

Moja domača gruča je sestavljena primarno iz starih IBM SAN Volume Controllerjev, ki sem jih radodarno dobil zastonj. Teh je aktivno povezanih 10 vendar sta v času pisanja tega poročila dva out of service zaradi problemov z RAID kontrolerji in tretja ni compute node, ampak je NFS strežnik in zbiralec statistike. Tako da je na voljo 7 strežnikov. Na komad en strežnik 24 niti, konfiguracija spomina pa je malo različna od strežnika do strežnika, ampak gre od 4 GB do okoli 142 GB. Vse skupaj je povezano z 1 Gbps ethernetom. Imam tudi opremo, da bi vzpostavil fibre channel SAN, ampak absolutno ni bilo dovolj časa za namen te naloge. Strastno si želim, da bi kje dobil priložnost, kjer bi se lahko res

detajlno ukvarjal z parametri in postavitevijo gruče. Pravzaprav trenutnotudi nobenega resource management-a ni na gruči. Resno bi se bilo potrebno lotiti Slurm-a.

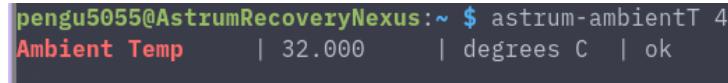


Slika 13: Astrum nodes dashboard

Kakorkoli ideja je uporabiti OpenMPI in Python izvedenko `mpi4py`, da razpršim podatkovno bazo čez niti strežnikov in tako pospešim predprocesiranje. Rekel bi, da je bilo to zabavno, saj je to res eden iz med tistih delov, ki me res veselijo, ampak res lovim rok za oddajo. Pa še T2 je bilo treba klicati, ker so mi resetirali nastavitev modema in of course ne zaupajo svojim strankam dovolj, da bi jim dali dostop do nastavitev.

```
Hi! This is rank 118 on astrum7. Ready to go to work...
Hi! This is rank 0 on astrum3. Ready to go to work...
Hi! This is rank 10 on astrum3. Ready to go to work...
Hi! This is rank 16 on astrum3. Ready to go to work...
Hi! This is rank 18 on astrum3. Ready to go to work...
Hi! This is rank 15 on astrum3. Ready to go to work...
Hi! This is rank 7 on astrum3. Ready to go to work...
Hi! This is rank 43 on astrum4. Ready to go to work...
Hi! This is rank 3 on astrum3. Ready to go to work...
Hi! This is rank 4 on astrum3. Ready to go to work...
Hi! This is rank 19 on astrum3. Ready to go to work...
Hi! This is rank 25 on astrum4. Ready to go to work...
Hi! This is rank 44 on astrum4. Ready to go to work...
Hi! This is rank 33 on astrum4. Ready to go to work...
Hi! This is rank 47 on astrum4. Ready to go to work...
Hi! This is rank 24 on astrum4. Ready to go to work...
Hi! This is rank 9 on astrum3. Ready to go to work...
Hi! This is rank 12 on astrum3. Ready to go to work...
Hi! This is rank 20 on astrum3. Ready to go to work...
Hi! This is rank 23 on astrum3. Ready to go to work...
Hi! This is rank 37 on astrum4. Ready to go to work...
Hi! This is rank 40 on astrum4. Ready to go to work...
Hi! This is rank 14 on astrum3. Ready to go to work...
Hi! This is rank 17 on astrum3. Ready to go to work...
Hi! This is rank 1 on astrum3. Ready to go to work...
Hi! This is rank 2 on astrum3. Ready to go to work...
Hi! This is rank 22 on astrum3. Ready to go to work...
Hi! This is rank 97 on astrum7. Ready to go to work...
Hi! This is rank 21 on astrum3. Ready to go to work...
Hi! This is rank 98 on astrum7. Ready to go to work...
Hi! This is rank 5 on astrum3. Ready to go to work...
Hi! This is rank 102 on astrum7. Ready to go to work...
Hi! This is rank 134 on astrum8. Ready to go to work...
Hi! This is rank 107 on astrum7. Ready to go to work...
Hi! This is rank 139 on astrum8. Ready to go to work...
Hi! This is rank 101 on astrum7. Ready to go to work...
Hi! This is rank 123 on astrum8. Ready to go to work...
Hi! This is rank 117 on astrum7. Ready to go to work...
Hi! This is rank 126 on astrum8. Ready to go to work...
Hi! This is rank 11 on astrum3. Ready to go to work...
Hi! This is rank 127 on astrum8. Ready to go to work...
Hi! This is rank 99 on astrum7. Ready to go to work...
Hi! This is rank 129 on astrum8. Ready to go to work...
Hi! This is rank 35 on astrum4. Ready to go to work...
Hi! This is rank 120 on astrum8. Ready to go to work...
Hi! This is rank 39 on astrum4. Ready to go to work...
```

Slika 14: Astrum MPI pognan na 152 nitih



Slika 15: Hitre posledice kljub zastekljenemu balkonu kjer je  $T_{\text{ambient}} \approx 14^\circ\text{C}$

Slika 16: Na glavnem strežniku se v glavni niti rekombinirajo in shranjujejo podatki

AH! Končno je vse skupaj predprocesirano. Sedaj pa lahko nadaljujemo z reševanjem. Po servisiranju gruče, kar mi je vzelo nekaj časa (in še vedno ni vse v najboljšem stanju), je trajalo kako uro, da se je vse skupaj izračunalo. Če privzamemo, da bi se vse skupaj izračunalo na enem jedru in bi se čas linearno povečeval z manjšanjem števila jeder, bi trajalo torej 152 ur oz. skoraj cel teden. Torej smo z MPI **OGROMNO** pridobili.

## 5.2 Rezultati

Načeloma sem prosil za akademsko cloud licenco za Gurobi optimizer, ampak glede na to, da jaz rabim oddat to nalogo zdaj, ne morem čakati na njihov odgovor. Tako da sem se odločil, da bom uporabil PuLP za modeliranje in poskusil z SCIP reševalnikom [3]. Če mi ne uspe v nekem zglednem času bom oddal nalogo in kot svojo zapuščino imel sprocesirano podatkovno bazo.

Ah ja, tu se ta pot konča. Dependency, ki ga želi je na voljo le za Ubuntu 20.04 in jaz sem na 22.04. Sem pa našel avtomatsko generacijo Gurobi licenc, le da moraš biti priključen na omrežje akademske institucije, da ti pusti generirati licence. K sreči imam remote dostop do mreže na Inštitutu Jožef Stefan, tako da sem lahko to naredil. Med pisanjem tega stavka mi je odpisal gospod iz Gurobi-ja in začel sem se pogajati za licenco. Mogoče je še upanja. Močno si želim, da si bo prof. Miha kljub zamudi lahko takoj ogledal nalogo.

Predlagana rešitev za mojo situacijo je Gurobijev Instant Cloud, ki se ga menda lahko dobi tudi na akademsko licenco. V ozadju skušam namestiti Gurobi Optimizer na gručo, medtem ko čakam da mi gospod Lee odgovori z nadaljnimi navodili. Ah ja, 2 uri kasneje in se mi zdi, da je izginil. To je to. Problem ostane odprt za prihodnje generacije.

```
pengu5055@astrum3:/cloud$ sudo apt install ./SCIP.deb
[sudo] password for pengu5055:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'scipoptsuite' instead of './SCIP.deb'
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 scipoptsuite : Depends: libgsl23 (>= 2.5) but it is not installable
E: Unable to correct problems, you have held broken packages.
pengu5055@astrum3:/cloud$
```

Slika 17: Dead end

## 6 Komentarji in izboljšave

Želim si da bi se prej lotil, da bi tako lahko imel čas se še zafrkavati z reševanjem tega expanded modela. Tako pa sem se moral zadovoljiti z osnovnim modelom in s tem, da sem na kar sprejemljiv način predprocesiral podatke za nekoga drugatega. Načeloma sem imel pri nalogi še zelo velike ambicije. Cela kategorizacija te tabele se dela preko OpenAI API-ja, ampak mogoče zdaj nima več toliko smisla to opisovati, če iz teh podatkov nimam nobene rešitve. Ideja je sicer bila taka, da bi ustvari nekaj jedilnikom, ki imajo constraint ta, da morajo imeti pravilno razmerje med skupinami živil, tako da bi upoštevali prehrambeno piramido. Skratka podatki za to so pripravljeni, samo nekdo mora zdaj zooptimizirat ta circa 400 k dimenzionalen problem.

Man I hope its not too late and my work is still interesting..

## Literatura

- [1] Stuart Mitchell, Michael J. O’Sullivan, and Iain Dunning. Pulp : A linear programming toolkit for python. 2011.
- [2] John et al. Forrest. Coin-or/CBC: Release releases/2.10.10, 2023.
- [3] Stephen Maher, Matthias Miltenberger, João Pedro Pedroso, Daniel Rethfeldt, Robert Schwarz, and Felipe Serrano. PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pages 301–307. Springer International Publishing, 2016.