# Scheduling-Queues

Finančni praktikum

Špela Bernardič, Ioann Stanković

10. 1. 2023

## 1 Navodilo naloge

Naloga se ukvarja z algoritmi 'Scheduling-Queues', ki uporabljajo podatke napovedane s strojnim učenjem. Potrebno je napisati simulacijo procesov, ki uporabi napovedane čase trajanja procesov in nato narediti analizo časa čakanja procesov. Preveriti je potrebno, kako različene porazdelitve trajanja procesov in kvaliteta napovedi vplivata na povprečen čas čakanja v vrsti. Napovedi časov trajanja procesov se lahko določi z dodajanjem šumov pravim vrednostim, lahko pa se uporabi naučen model.

## 2 Opis problema

S problemom razvrščanja opravil v vrsti se srečamo na različnih področjih. Za opravila nevemo nujno njihove dolžino trajanja, velikokrat pa jo laho ocenimo oziroma napovemo. Zato je poleg optimalnosti algoritmov, ki za razvrščanje uporabljajo dejanske čase trajanja opravila, smiselno analizirati tudi primere, ko se uporabljajo napovedani časi trajanja opravila. Opazovati je torej treba (povprečen) čas čakanja opravila v vrsti. Želimo, da je ta čim krajši.

## 3 Algoritmi razvrščanja procesov

Za reševanje problema razvrščanja procesov v vrsti se uporabljaj različni algoritmi. V nalogi sva uporabila naslednje osnovne algoritme:

- FCFS (First Come First Serve) je algoritem, ki izvaja procese po vrstnem redu njihovega prihoda.
- SJF (Non-Preemptive Shortest Job First) je algoritem, pri katerem se za naslednjo izvedbo izbere proces z najkrajšim časom trajanja. Ko se določi kateri proces se naj izvede naslednji, se ta izvede do konca.
- PSJF (Preemptive Shortest Job First) je algoritem, pri katerem se proces z najkrajšim časom trajanja začne izvajati prvi. Ob prihodu novega procesa se le ta postavi v čakalno vrsto. Če pa pride proces s krajšim trajanjem od procesa, ki se trenutno izvaja, se trenutni proces ustavi in vrne v vrsto. Začne se izvajati proces s krajšim trajanjem.

- SRPT (Shortest remaining processing time) je algoritem podoben PSJF, vendar ta upošteva preostanke trajanj procesov.

Poleg osnovnih algoritmov sva napisala še variacje z napovedmi:

- SPJF (Non-Preemptive Shortest Predicted Job First) za naslednjo izvedbo izbere proces z najkrajšim **napovedanim** časom trajanja.
- PSPJF (Preemptive Shortest Predicted Job First) razvršča procese glede na najkrajši **napovedan** čas trajanja.
- SPRPT (Shortest Predicted remaining processing time) je algoritem podoben PSPJF, vendar ta upošteva preostanek **napovedanega** trajanj procesov.

Primer enega od algoritmov:

```
def PSPJF(seznam):
  opravila = sorted((Opravilo(*t) for t in seznam), key=lambda item: item.arrival)
  opravila = + [Opravilo(None, float('inf'), float('inf'), float('inf'))]
  vrsta = []
  t = 0
  cakanje = 0
  for naslednje in opravila:
      while vrsta:

          prekinitev = naslednje.arrival - t
          if prekinitev <= 0:
              break

          predvideno, cas, opravilo = vrsta[0]
          preostanek = cas - prekinitev

          if preostanek > 0:
              if predvideno > naslednje.predicted:
                  vrsta[0] = (predvideno, preostanek, opravilo)
                  t = naslednje.arrival
                  break

              else:
                  break

          else:
              heappop(vrsta)
              t += cas
              cakanje += t - opravilo.length - opravilo.arrival
      else:
          t = naslednje.arrival

      heappush(vrsta, (naslednje.predicted, naslednje.length, naslednje))
  return cakanje
```

# 4 Generiranje podatkov

Za algoritme je bilo potrebno zgenerirat podatke za čase prihodov opravil, dolžine teh opravil, in šum na opravilih. Za generiranje podatkov sva uporabila programski jezik R. Čase prihodov opravil sva zgenerirala z eksponentno porazdelitvijo s parametrom $\lambda = 0.25$, torej $N \sim exp(0.25)$. Dolžino opravil sva generirala z beta in normalno porazdelitvijo. Šum pa sva dodala z normalno porazdelitvijo $Z \sim N(0, 0.01)$. Želela sva analizirati, kako se čas čakanja spremeni glede na kvaliteto napovedi. Zato sva šum zgenerirala za $Z \sim N(0, \sigma)$ pri različnh $\sigma$. Da bi se izognila dolžinam opravil in šumu velikosti 0 sva za minimalno dolžino opravila vzela $10^{-7}$. Da bi bili končni podatki bolj relevantni, sva poskuse izvajala po 100-krat in na koncu pri vzela povprečje razltatov.

## 4.1 Generiranje dolžin opravil Beta porazdelitve

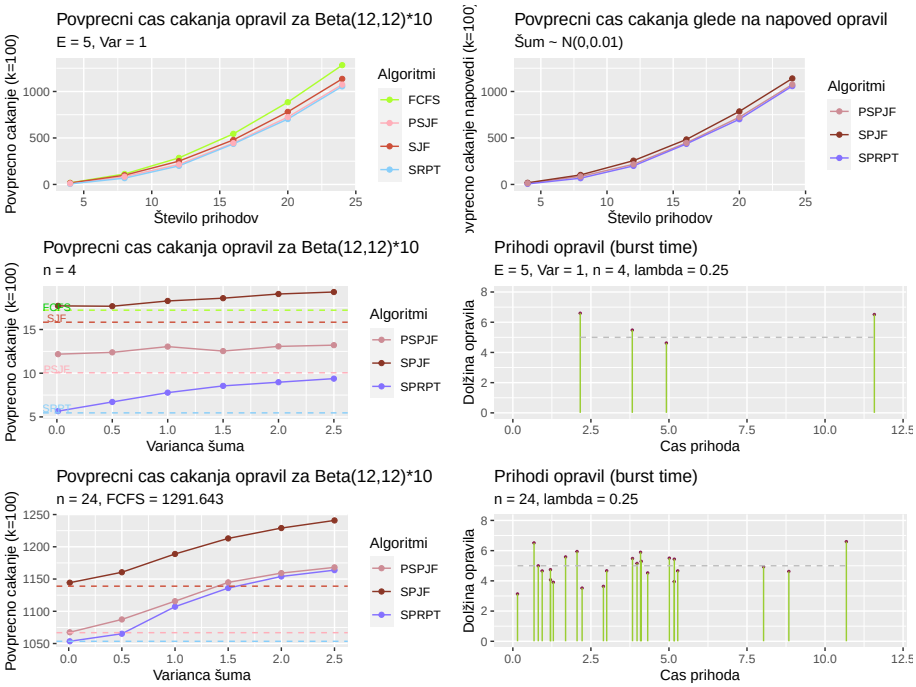
Za generiranje dolžin opravil sva v enem primeru uporabila beta porazdelitev skalirano za 10, torej $X \sim Beta(\alpha, \beta) \times 10$, s pričakovano vrednostjo $E[X] = 5$ in varianco $Var[X] = 1$. Od tod sva izračunala parametre za beta porazdelitev $\alpha = \beta = 12$.
Zanimalo naju je tudi, če se oblika grafa čakanja spremeni, ko izberemo majhne dolžine opravil. To sva opazovala le za Beta porazdelitev. Izbrala sva si vrednosti za porazdelitev dolžine opravil $E[X] = 0.5$ in $Var[X] = 0.01$ (torej $X \sim Beta(12, 12)$) ter določila šum z $Var[Z] = 0.01$.

## 4.2 Generiranje dolžin opravil Normalne porazdelitve

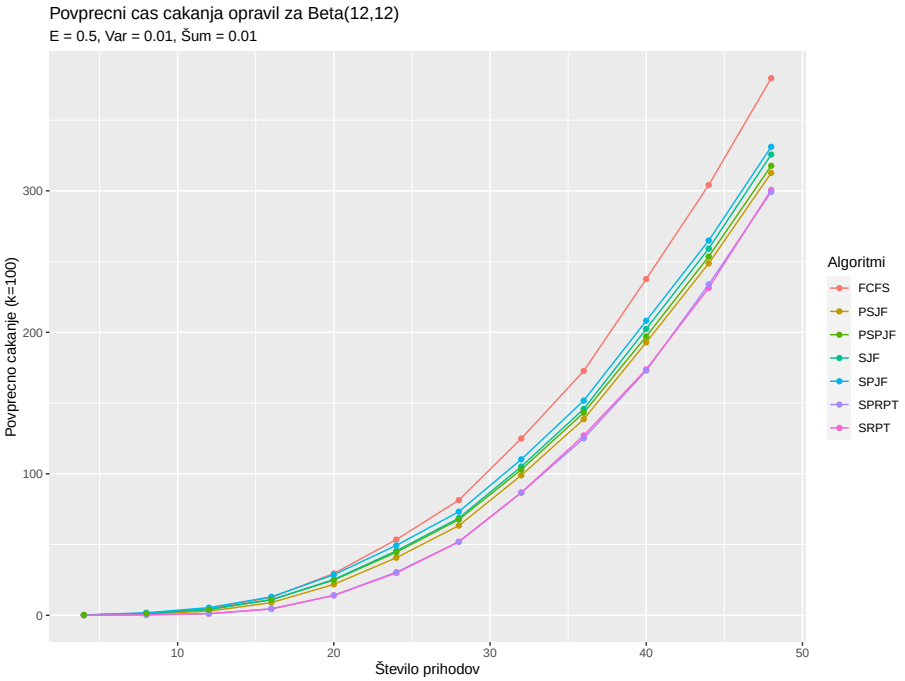
V drugem primeru sva uporabila Normalno porazdelitev $Y \sim |N(5, 5)|$, vzela sva absolutno vrednost normalne porazdelitve, da bi se izognila negativnim vrednostim. DEJANSKA VREDNOST UPANJA IN VARIANCE.

# 5 Analiza

## 5.1 Analiza Beta porazdelitve

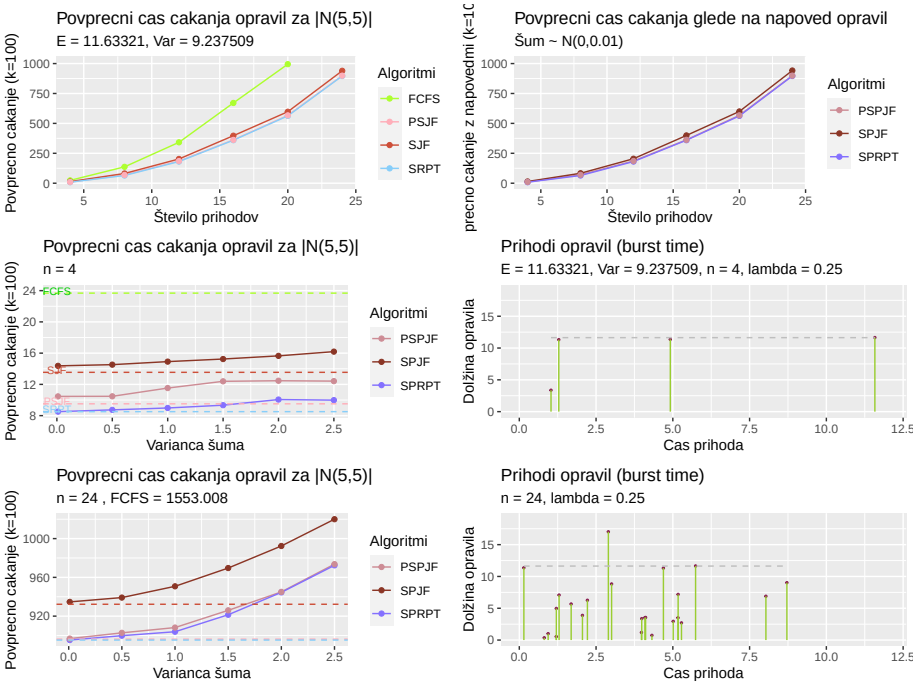



Iz grafov lahko sklepamo, da čas povprečnega čakanja narašča eksponentno glede na število prihodov. Opazimo večje časovno odstopanje za algoritem FCFS, kar nas tudi ne preseneča, saj je ta algoritem že za majhno število prihodov relativno neučinkovit. Mogoče bolj presenetljiv rezultat je, da imamo zelo majhno odstopanje povprečnega časa čakanja za algoritme z napovedmi, tudi če povečamo varianco na 2.5, kar je relativno velik šum, ki znatno spremeni začetne podatke. Iz grafov sklepava, da povprečni čas čakanja narašča linearno s spremembo variance šuma. Nisva pogledala obnašanje algoritmov za večje variance šumov, saj se nam to ni zdelo smiselno.




Tu so zelo presenetljivi rezultati, ki jih nerazumem

## 5.2 Analiza Normalne porazdelitve




Lahko sklepamo podobno kot pri prvi analizi, vendar opazimo še večje odstopanje pri algoritmu FCFS, razlog je da so v tem primeru dolžine opravil v

povprečju daljše kot pri beta porazdelitvi.