UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

INTERAKTÍVNA APLIKÁCIA PRE RIEŠENIE ÚLOH TEÓRIE HROMADNEJ OBSLUHY

DIPLOMOVÁ PRÁCA

2015 Bc. Andrej ŠIŠKA

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

INTERAKTÍVNA APLIKÁCIA PRE RIEŠENIE ÚLOH TEÓRIE HROMADNEJ OBSLUHY

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomická a finančná matematika
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	Mgr. Soňa Kilianová, PhD.

Bratislava 2015 **Bc. Andrej ŠIŠKA**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Andrej Šiška
Študijný program:	ekonomická a finančná matematika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	9.1.9. aplikovaná matematika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Interaktívna aplikácia pre riešenie úloh teórie hromadnej obsluhy
Interactive application for solving queueing theory problems

Cieľ: Naprogramovať interaktívne prostredie pre riešenie úloh teórie hromadnej obsluhy

Vedúci:	Mgr. Soňa Kilianová, PhD.
Katedra:	FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry:	prof. RNDr. Daniel Ševčovič, CSc.
Dátum zadania:	29.01.2014

Dátum schválenia: 10.02.2014 prof. RNDr. Daniel Ševčovič, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné vyhlásenie

Čestne vyhlasujem, že som predloženú diplomovú prácu spracoval samostatne s použitím uvedenej literatúry a zdrojov.

.....

Bc. Andrej Šiška

Poďakovanie Touto cestou sa chcem poďakovať svojej vedúcej diplomovej práce Mgr. Soni Kilianovej, PhD. za trpezlivosť a odborné usmernenia, ktoré mi pomohli pri písaní tejto práce. Takisto ďakujem priateľom a rodine za morálnu podporu a povzbudenie, keď som to najviac potreboval.

Abstrakt

ŠIŠKA, Andrej: Interaktívna aplikácia pre riešenie úloh teórie hromadnej obsluhy [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: Mgr. Soňa Kilianová, PhD., Bratislava, 2015, 55 s.

Cieľom práce bolo navrhnúť a naprogramovať webovú aplikáciu na výpočet charakteristík a optimalizačných úloh pre modely teórie hromadnej obsluhy. Aplikácia je postavená na platforme Google app engine, s využitím HTML, CSS a programovacích jazykov Python a Javascript.

Práca je rozdelená na niekoľko kapitol. V prvej sa zameriavame na základné vlastnosti teórie hromadnej obsluhy spolu s popisom použitých modelov, kde uvádzame ich charakteristiky spolu s odvodením. Druhá kapitola predstavuje návrh webovej aplikácie, vrátane dizajnu a funkcionality, ako aj popis použitých technológií. V tretej kapitole opíšeme postup pri programovaní spolu s relevantnými ukážkami zdrojového kódu. Posledná, piata, kapitola slúži na predvedenie aplikácie na niekoľkých príkladoch.

Kľúčové slová: Teória hromadnej obsluhy, Teória frontov, Google app engine, Python

Abstract

ŠIŠKA, Andrej: Interactive application for solving queueing theory problems [Masters thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; supervisor: Mgr. Soňa Kilianová, PhD., Bratislava, 2015, 55 p.

The goal of the thesis is to design and program an online application for computing characteristics and optimization problems for queuing theory models. The application is based on the Google app engine platform, using HTML, CSS, and programming languages such as Python and Javascript.

The thesis is divided into several chapters. In the first one, we focus on basic properties of queuing theory together with description of models we use in the application together with their derivation. Second chapter presents the draft for the application, including design and functionality, with short description of applied technologies. In third chapter we describe the procedures in programming the application with relevant samples of code. Last, fifth, chapter serves as a demonstration of the application on several queuing theory problems.

Keywords: Queuing theory, Google app engine, Python

OBSAH

Obsah

OBSAH

ZOZNAM OBRÁZKOV

Zoznam obrázkov

Zoznam použitých symbolov

Zoznam použitých symbolov

$\boldsymbol{\beta}$	intenzita prevádzky pre viac obslužných liniek
$\boldsymbol{E(R)}$	stredná doba čakania v sytéme
$\boldsymbol{E(W)}$	stredná doba čakania vo fronte
$\boldsymbol{\gamma}$	priemerný počet zákazníkov vo fronte
$\boldsymbol{\kappa}$	priemerný počet zákazníkov v systéme
$\boldsymbol{\mu}$	intenzita výstupu
$\boldsymbol{\lambda}$	intenzita vstupu
$\boldsymbol{\rho}$	intenzita prevádzky
$\boldsymbol{\nu}$	priemerný počet obsadených liniek

Zoznam použitých skratiek

Zoznam použitých skratiek

CSS	Cascading Style Sheets
GAE	Google App Engine
HTML	Hypertext markup language
FIFO	First in, first out
LIFO	Last in, first out
SIRO	Service in random order
PRI	Priority
WWW	World wide web

ÚVOD

Úvod

Na predmete stochastické modely operačnej analýzy som sa prvýkrát zoznámil s teóriou hromadnej obsluhy. Napriek tomu, že tento názov mi zo začiatku nič nehovoril, veľmi rýchlo som si uvedomil, že modely hromadnej obsluhy sa nachádzajú všade okolo nás a každodenne sme ich súčasťou v bežnom živote.

Objektom záujmu tejto matematickej disciplíny sú systémy, do ktorých vstupujú jednotky(napríklad zákazníci, tovar, dáta...) a čakajú na obsluhu v obslužných linkách(pokladne, okienka na pošte, autoservis...).

Teória hromadnej obsluhy sa zaoberá sformulovaním matematických modelov pre takéto systémy a ich následným popisom. Dokáže nám teda podať informácie napríklad o priemernom počte zákazníkov v systéme alebo vo fronte, o predpokladanej dĺžke čakania, alebo pravdepodobnosti, že sa v systéme nachádza určitý počet zákazníkov.

Z pohľadu prevádzkovateľa systému je možné sledovať ďalšie charakteristiky, ako je napríklad priemerná vyťaženosť obslužných liniek, priemerný počet zastavení linky a podobne. Takisto je možné optimalizovať celkové náklady vzhľadom na náklady na prevádzku jednotlivých liniek, alebo stratu vzniknutím odmietnutím zákazníka kvôli nedostatočnej dĺžke frontu. Tieto straty môže prevádzkovateľ minimalizovať napríklad zmenou počtu liniek alebo počtu miest vo fronte, v niektorých prípadoch dokáže aj nastaviť rýchlosť alebo výkon linky.

Pre výpočet takýchto charakteristík sa využíva buď simulačný(Monte Carlo), alebo stochastický prístup. V tejto práci sa budeme zaoberať práve druhou spomínanou možnosťou.

Pre zjednodušenie týchto výpočtov je dobré použiť nejaké používateľské rozhranie, ktoré interaktívnou formou na základe zadaných vstupov zobrazí charakteristiky modelu. Avšak verejne dostupných takýchto programov je veľmi málo. Tie ktoré sa mi podarilo nájsť, boli buď nepraktické na použitie, nedostupné, alebo dokonca obsahovali závažné chyby. Žiadna z nájdených nerieši optimalizačné úlohy.

Aj preto bude cieľom mojej práce bude navrhnúť a naprogramovať webovú aplikáciu, ktorá bude schopná vypočítať charakteristiky pre viacero modelov hromadnej obsluhy, spolu s ponúknutím rozhrania pre optimalizáciu systému pre prevádzkovateľa. Pre čo najširšiu dostupnosť bude používateľské rozhranie v angličtine. S programovaním we-

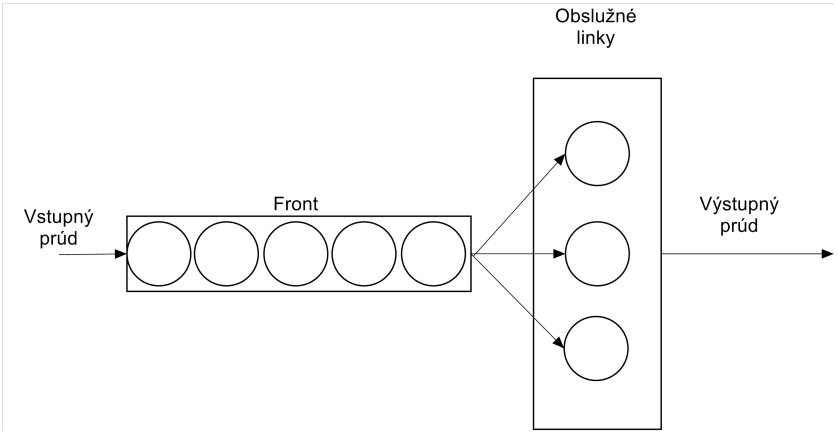
ÚVOD

bovej aplikácie som sa už stretol, preto túto diplomovú prácu beriem aj ako príležitosť rozšíriť svoje schopnosti a skúsenosti, a tiež využiť niektoré pokročilejšie technológie.

1 Teória hromadnej obsluhy

1.1 Úvod

Základným prvkom systému hromadnej obsluhy sú takzvané **obslužné linky**. Ide o miesta, kam prichádzajú zákazníci a dochádza k ich obsluhe. K týmto linkám vedie **front**, v ktorom zákazníci čakajú na obsluhu. Obslužné linky spolu s frontom nazývame **systémom**. Môže ísť napríklad o rady v školskej jedálni, kaderníctvo s čakárňou, autoservis a iné. Základný model je naznačený na Obr. 1.



Obr. 1: Schéma teórie hromadnej obsluhy

Teória hromadnej obsluhy pomáha odpovedať na otázky o priemernej obsadenosti liniek, dĺžke frontu, očakávanom čase strávenom v systéme, pomere zákazníkov odmietnutých pre nízku kapacitu frontu a podobne. Pre prevádzkovateľa systému je užitočné poznať riešenia niektorých optimalizačných úloh, ako napríklad optimálny počet obslužných liniek, optimálna kapacita frontu a ďalšie.

Príchody zákazníkov do systému, ako aj doba ich obsluhy, sú vo všeobecnosti modelované ako náhodné procesy. Táto práca predpokladá základnú znalosť Markovovych procesov, ako aj Poissonovho procesu. V prípade potreby možno čitateľovi pre doplnenie vedomostí odporučiť [3] a [1]. Naša práca sa teoretickému pozadiu náhodných procesov venovať nebude a zameriava sa najmä na praktickú časť.

V nasledovných kapitolách budeme čerpať najmä z [1] a [5].

1.2 Kendallova notácia

Z dôvodu veľkej typovej rozmanitosti úloh teórie hromadnej obsluhy je zaužívané používať symboliku, ktorú ako prvý zaviedol **D. G. Kendall**. Úlohy sú označované nasledovne:

$$X/Y/n/m/N/D,$$

kde

X: značí typ náhodného procesu príchodu zákazníkov,

Y: značí rozdelenie pre dĺžku doby obsluhy,

n: značí počet obslužných liniek,

m: značí maximálny počet zákazníkov v systéme (vrátane práve obsluhovaných),

N: značí veľkosť populácie, t.j. koľko zákazníkov môže najviac systém používať,

D: značí typ frontu (FIFO, LIFO, SIRO, PRI).

Za X, Y pripúšťame nasledovné symboly:

M: Poissonov proces/exponenciálne rozloženie doby obsluhy,

D: deterministické príchody/konštantná doba obsluhy,

E: Erlangovo rozloženie,

K: χ^2 rozloženie,

G: všeobecné rozloženie.

V praxi, ak $D = FIFO$ a $N = \infty$, sa tieto symboly často vypúšťajú. Takisto pre prípad $m = \infty$, my však budeme toto označenie používať aj v tomto prípade. Tiež je dobré poznamenať, že v našej aplikácii sa budeme pre rozloženie príchodov do systému a doby obsluhy zaoberať iba Poissonovym procesom, resp. exponenciálnym rozdelením a deterministickými príchodmi, resp. konštantnou dobou obsluhy.

1.3 Prehľad modelov

V tejto časti si rozoberieme jednotlivé systémy z pohľadu ich parametrov a odvodíme vzorce potrebné k naprogramovaniu aplikácie. Postupy sú prebrané z [1] a [5].

1.3.1 Systém M/M/1/∞

M/M/1/∞ je najjednoduchší prípad netriviálneho systému, kde uvažujeme nasledovné predpoklady:

- príchody zákazníkov sú opísané Poissonovym procesom s parametrom λ,
- dĺžka doby obsluhy je exponenciálne rozložená s priemernou dobou obsluhy $1/\mu$,
- v systéme je jedna obslužná linka,
- dĺžka frontu je neobmedzená.

Tento model nepredpokladá prioritu, predbiehanie, ani váhavosť zákazníkov.

Označme $X(t)$ náhodný počet zákazníkov v systéme. Potom môžeme definovať pravdepodobnosti $p_k(t) = P(X(t) = k)$, $p_{ik} = P(X(t+h) = k | X(t) = i)$. Potom platí:

$$\begin{aligned} \frac{dp_k}{dt}(t) &= \frac{d}{dh} p_k(t+h)|_{h=0} \\ &= \frac{d}{dh} \sum_{i=0}^{\infty} p_i(t) p_{ik}(h) = \sum_{i=0}^{\infty} p_i(t) \dot{p_{ik}}(0). \end{aligned} \tag{1}$$

Pre odvodenie diferenciálnych rovníc pre $p_k(t)$ ďalej využijeme Lemu z [1, str. 13]:

Lema 1.1. *Pravdepodobnosť výskytu viac ako jednej udalosti typu príchod/odchod na intervale dĺžky h je $o(h)$.*

Z lemy 1.1 a vlastností Poissonovho procesu ďalej vyplýva:

$$\begin{aligned} &\dot{p}_{i,i+1}(0) = \lambda, \dot{p}_{ik}(0) = 0 \text{ pre všetky } k > i+1, \\ &\dot{p}_{ii}(0) = \frac{d}{dh}(e^{-\lambda h} e^{-\mu h})|_{h=0} = -\lambda - \mu, \dot{p}_{i,i-1}(0) = \mu, \dot{p}_{ik}(0) = 0 \text{ pre } k < i-1 \text{ a } i > 0, \\ &\dot{p}_{00}(0) = -\lambda, \dot{p}_{0k}(0) = 0 \text{ pre } k > 0. \end{aligned} \tag{2}$$

Dostávame diferenciálne rovnice:

$$\begin{aligned} \dot{p_0} &= -\lambda p_0 + \mu p_1, \\ \dot{p_k} &= \lambda p_{k-1} - (\mu + \lambda) p_k + \mu p_{k+1} \text{ pre } k > 0. \end{aligned} \tag{3}$$

Teória hromadnej obsluhy vychádza z predpokladu, že ide o ustálený systém. Budeme teda sústavu rovníc (3) riešiť ako stacionárny systém, čiže taký, kde pravdepodobnosti nezávisia na čase t. Hľadáme pravdepodobnosti p_k, čo sú pravdepodobnosti, že v systéme sa v ktoromkoľvek čase nachádza k zákazníkov.

Položíme teda derivácie rovné nule a dostávame:

$$\begin{aligned} 0 &= -\lambda p_0 + \mu p_1, \\ 0 &= \lambda p_{k-1} - (\mu + \lambda) p_k + \mu p_{k+1} \text{ pre } k > 0. \end{aligned} \tag{4}$$

Použitím substitúcie $z_k = -\lambda p_k + \mu p_{k+1}$ získavame novú sústavu

$$z_0 = 0,$$

$$z_k - z_{k-1} = 0 \text{ pre } k > 0.$$

Z toho jednoznačne $z_k = 0 \forall k$ a použitím spätnej substitúcie

$$\begin{aligned} p_{k+1} &= \frac{\lambda}{\mu} p_k = \rho p_k, \\ p_k &= \rho^k p_0, \end{aligned}$$

kde $\rho = \frac{\lambda}{\mu}$ nazývame *intenzitou prevádzky*. Je zrejmé, že $\rho < 1$, v opačnom prípade bude dĺžka frontu divergovať.

Hodnotu p_0 určíme zo vzťahu pre súčet disjunktných pravdepodobností

$$1 = \sum_{k=0}^{\infty} p_k = p_0 \sum_{k=0}^{\infty} \rho^k = p_0 \frac{1}{1-\rho}, \tag{5}$$

z čoho vyjadríme vzťahy pre p_0 a p_k:

$$\begin{aligned} p_0 &= 1 - \rho, \\ p_k &= (1 - \rho)\rho^k. \end{aligned} \tag{6}$$

S využitím odvodených pravdepodobností dokážeme vypočítať ďalšie charakteristiky systému, ktoré neskôr použijeme vo webovej aplikácii.

Priemerný počet zákazníkov v systéme dostaneme vypočítaním strednej hodnoty diskrétnej náhodnej premennej:

$$\kappa = \sum_{k=0}^{\infty} k p_k = (1 - \rho) \sum_{k=0}^{\infty} k\rho^k = \frac{\rho}{1 - \rho}, \tag{7}$$

s varianciou:

$$\sum_{k=0}^{\infty} k^2 p_k - \frac{\rho^2}{(1 - \rho^2)} = \frac{\rho}{(1 - \rho^2)}. \tag{8}$$

Rovnakým spôsobom dostaneme aj priemerný počet zákazníkov vo fronte. Tu však neuvažujeme práve obsluhovaného zákazníka.

$$\gamma = \sum_{k=0}^{\infty} k p_{k+1} = p_0 \sum_{k=0}^{\infty} k\rho^{k+1} = \frac{\rho^2}{1 - \rho}. \tag{9}$$

Pravdepodobnosť, že k-ty zákazník čaká viac než w možno získať zo vzťahu pre pravdepodobnosť toho, že na intervale dĺžky t nastane najviac k udalostí [1, strana 12]:

$$r_k(t) = e^{-\lambda t} \sum_{i=0}^{\infty} \frac{(\lambda t)^i}{i!}.$$

K-ty zákazník čaká dlhšie než w práve vtedy, ak po dobu w bolo obslúžených najvýš $k - 1$ zákazníkov. Túto situáciu možno ľahko aplikovať na predošlý všeobecný vzorec:

$$P(W_k > w) = e^{-\mu w} \sum_{j=0}^{k-1} \frac{(\mu w)^j}{j!}, \tag{10}$$

kde j je počet obslúžených zákazníkov za dobu w. Ďalej môžeme rozloženie doby čakania zovšeobecniť pre ľubovoľného zákazníka:

$$\begin{aligned} P(W > w) &= \sum_{k=1}^{\infty} p_k P(W_k > w) = \sum_{k=1}^{\infty} (1-\rho)\rho^k e^{-\mu w} \sum_{j=0}^{k-1} \frac{(\mu w)^j}{j!} \\ &= (1-\rho)e^{-\mu w} \sum_{j=0}^{\infty} \frac{(\mu w)^j}{j!} \sum_{k=j+1}^{\infty} \rho^k \\ &= (1-\rho)e^{-\mu w} \sum_{j=0}^{\infty} \frac{(\mu w)^j}{j!} \rho^{k+1} \frac{1}{1-\rho} \\ &= e^{-\mu w} \rho \sum_{j=0}^{\infty} (\mu\rho)^j \frac{(\lambda w)^j}{j!} \\ &= e^{-\mu w} \rho \sum_{j=0}^{\infty} \frac{w^j}{j!} \\ &= \rho e^{-(\mu-\lambda)w}. \end{aligned} \tag{11}$$

Strednú dobu čakania dostaneme z definície strednej hodnoty:

$$\begin{aligned} E(W) &= \int_0^{\infty} w \frac{d}{dw}[1 - \rho e^{-(\mu-\lambda)w}]dw \\ &= -\rho \int_0^{\infty} w \frac{d}{dw}[e^{-(\mu-\lambda)w}]dw \\ &= -\rho \left[w e^{-(\mu-\lambda)w} \right]_0^{\infty} + \rho \int_0^{\infty} e^{-(\mu-\lambda)w} dw \\ &= \frac{\rho}{\mu - \rho}. \end{aligned} \tag{12}$$

Celková doba strávená v systéme je väčšia o dobu obsluhy:

$$\begin{aligned} P(R > w) &= \sum_{k=1}^{\infty} p_k P(W_{k+1} > w) = e^{-(\mu-\lambda)w} \\ &= \frac{P(W > w)}{\rho}. \end{aligned} \tag{13}$$

Teda aj stredná hodnota

$$E(R) = \frac{E(W)}{\rho} = \frac{1}{\mu - \lambda}. \tag{14}$$

1.3.2 Systém M/M/1/n

Na rozdiel od predošlého systému M/M/n/∞, v tomto prípade je maximálny počet zákazníkov v systéme obmedzený(n). Znamená to, že pravdepodobnosť, že v systéme je k zákazníkov, je $p_k = 0$ pre $k > n$. Preto $p_{n,n+1}(h) = 0$ a

$$p_{nn}(h) = \frac{d}{dh}e^{-\mu h} = -\mu h + o(h).$$

Diferenciálne rovnice sa zmenia pre $k = n$ takto:

$$\begin{aligned} \dot{p_0} &= -\lambda p_0 + \mu p_1 \\ \dot{p_k} &= \lambda p_{k-1} - (\mu + \lambda)p_k + \mu p_{k+1} \text{ pre } 0 < k < n \\ \dot{p_n} &= \lambda p_{n-1} - \mu p_n. \end{aligned} \tag{15}$$

Pre $0 \leq k \leq n$ dostávame

$$p_k = \rho^k p_0.$$

Je dôležité poznamenať, že v prípade obmedzenej dĺžky frontu pripúšťame aj hodnoty $\rho > 1$. Potom z podmienky $\sum p_k = 1$ dostávame:

$$p_0 = \frac{1}{\sum_{k=0}^{n} p_k} = \begin{cases} \frac{1-\rho}{1-\rho^{n+1}} & \text{pre } \rho \neq 1, \\ \frac{1}{n+1} & \text{pre } \rho = 1. \end{cases} \tag{16}$$

Oproti systému M/M/n/∞ máme nové charakteristiky pre pravdepodobnosť odmietnutia zákazníka ($p_n = p_0\rho^n$) a priemerný počet odmietnutých zákazníkov za jednotku času ($p_n\lambda$). Teraz odvodíme ďalšie charakteristiky užitočné pre tento model.

Priemerná doba obsadenosti linky:

$$U_s = 1 - p_0.$$

Priemerný počet zákazníkov v systéme [5, str. 33]:

$$\begin{aligned} \kappa &= \sum_{k=1}^{n} k\rho^k p_0 = p_0 \sum_{k=1}^{n} k \\ &= \frac{1}{n+1}\frac{n(n+1)}{2} = \frac{n}{2} \text{ pre } \rho = 1. \end{aligned} \tag{17}$$

Pre intenzitu prevádzky

$$\rho \neq 1 :$$

$$\begin{aligned}
\kappa &= \sum_{k=1}^{n} k\rho^k p_0 = \rho p_0 \sum_{k=1}^{n} k\rho^{k-1} \\
&= \rho p_0 \left(\sum_{k=1}^{n} k\rho^{k-1} \right) = \rho p_0 \left(\rho \frac{1-\rho^n}{1-\rho} \right) = \rho p_0 \left(\frac{\rho - \rho^{n+1}}{1-\rho} \right) \\
&= (1-(n+1)\rho^n)(1-\rho) + \rho - \rho^{n+1} \frac{\rho p_0}{(1-\rho)^2} \\
&= \frac{\rho p_0 (1-(n+1)\rho^n - \rho + (n+1)\rho^{n+1} + \rho - \rho^{n+1})}{(1-\rho)^2} \\
&= \frac{\rho p_0 (1-(n+1)\rho^n + n\rho^{n+1})}{(1-\rho)^2} \\
&= \frac{\rho(1-(n+1)\rho^n + n)}{(1-\rho)(1-\rho^{n+1})}.
\end{aligned} \tag{18}$$

Priemerný počet zákazníkov vo fronte:

$$\gamma = \sum_{k=1}^{n} (k-1)p_k = \sum_{k=1}^{n} k p_k - \sum_{k=1}^{n} p_k = \kappa - U_s. \tag{19}$$

Očakávaná doba strávená v systéme [5, str. 33]:

$$\begin{aligned}
E(R) &= \sum_{k=1}^{n-1} \frac{k+1}{\mu} \frac{p_k}{1-p_n} = \sum_{k=1}^{n-1} \frac{k+1}{\mu} \frac{\rho^k p_0}{1-p_n} \\
&= \frac{1}{\lambda(1-p_n)} \sum_{k=0}^{n-1} (k+1) p_{k+1} = \frac{\kappa}{\lambda(1-p_n)}.
\end{aligned} \tag{20}$$

Očakávaná doba strávená vo fronte (rozdiel tvorí opäť iba doba obsluhy):

$$E(W) = E(R) - \frac{1}{\mu} = \frac{\kappa}{\lambda(1-p_n)} - \frac{1}{\mu}. \tag{21}$$

1.3.3 Systém M/M/n/∞

Tento model je variáciou na systém M/M/1/∞. Rozdiel je len v počte obslužných liniek n. Taktiež treba mať na zreteli, že intenzita prevádzky μ je vyjadrená pre jednu obslužnú linku.

Podobne ako v systéme M/M/1/∞ dostávame rovnice:

$$\begin{gathered}\dot{p}_{i,i+1}(0)=\lambda, \dot{p}_{ik}(0)=0 \text{ pre všetky } k \geq i+1, \\ \dot{p}_{ii}(0)=-\lambda-\min\{i,n\}\mu, \dot{p}_{i,i-1}(0)=\min\{i,n\}\mu, \dot{p}_{ik}(0)=0 \text{ pre } k<i-1 \text{ a } i>0, \\ \dot{p}_{00}(0)=-\lambda, \dot{p}_{0k}(0)=0 \text{ pre } k<0.\end{gathered} \tag{22}$$

Ďalej

$$\begin{aligned}\frac{d}{dt}p_k(t) &= \frac{d}{dh}p_k(t+h)|_{h=0} \\ &= \sum_{i=0}^{\infty} p_i(t)\frac{d}{dh}p_{ik}(h)|_{h=0} = \sum_{i=0}^{\infty} p_i(t)\frac{d}{dh}p_{ik}(h)|_{h=0} \\ &= \begin{cases} -\lambda p_0(t)+\mu p_1(t) & \text{pre } k=0, \\ \lambda p_{k-1}(t)-(\lambda+k\mu)p_k(t)+(k+1)\mu p_{k+1}(t) & \text{pre } 1\leq k\leq n, \\ \lambda p_{k-1}(t)-(\lambda+k\mu)p_k(t)+n\mu p_{k+1}(t) & \text{pre } k\geq n. \end{cases}\end{aligned} \tag{23}$$

Opäť skúmame ustálené stacionárne pravdepobnosti. Položíme teda derivácie rovné nule. Zavedením substitúcie

$$\begin{gathered}z_k=\lambda p_{k-1}-n\mu p_k \text{ pre } k>n, \\ z_k=\lambda p_{k-1}-k\mu p_k \text{ pre } 1\leq k\leq n\end{gathered} \tag{24}$$

dostávame

$$z_0=0,$$

$$z_k-z_{k-1}=0 \text{ pre } k>0,$$

teda $z_k=0\ \forall k$.

Rozlíšením situácie podľa k dostávame pre $0<k\leq n$:

$$p_k=\frac{\lambda}{k\mu}p_{k-1}=\frac{\rho}{k}p_{k-1},\ \rho=\frac{\lambda}{\mu},$$

po úprave

$$p_k = \frac{\rho^k}{k!} p_0.$$

Pre $k > n$

$$p_k = \beta^{k-n} p_n = \frac{\rho^{k-n}}{n^{k-n}} p_n,$$

kde $\beta = \frac{\lambda}{\mu n}$ je intenzita prevádzky. Pre zamedzenie divergencie dĺžky frontu požadujeme $\beta < 1$.

Zo vzťahu $1 = \sum_{k=0}^{\infty} p_k$ opäť odvodíme predpis pre p_0.

$$\begin{aligned} 1 = \sum_{k=0}^{\infty} p_k &= p_0 \left(\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \sum_{k=n}^{\infty} \beta^{k-n} \right) \\ &= p_0 \left(\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \frac{1}{1-\beta} \right), \end{aligned} \tag{25}$$

$$p_0 = \left(\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \frac{1}{1-\beta} \right)^{-1}.$$

Pomocou pravdepodobností p_k vyjadríme ďalšie charakteristiky.

Pravdepodobnosť čakania je pravdepodobnosť, že pri mojom príchode je v systéme aspon toľko zákazníkov, ako je počet obslužných liniek:

$$\Pi = \sum_{k=n}^{\infty} p_k = p_n \frac{1}{1-\beta}. \tag{26}$$

Pravdepodobnosť okamžitej obsluhy je teda doplnok k pravdepodobnosti čakania:

$$\sum_{k=0}^{n-1} p_k = 1 - \Pi. \tag{27}$$

Priemerný počet zákazníkov vo fronte vypočítame cez strednú hodnotu diskrétnej náhodnej premennej:

$$\gamma = \sum_{r=0}^{\infty} r p_{n+r} = p_n \sum_{r=0}^{\infty} r \beta^r = p_n \frac{\beta}{(1-\beta)^2}. \tag{28}$$

Pri výpočte priemerného počtu obsadených liniek cez strednú hodnotu treba osobitne uvažovať prípad, kedy sú obsadené všetky linky a kedy iba niektoré z nich:

$$\begin{aligned}\nu &= \sum_{k=1}^{n} k p_k + n \sum_{k=n+1}^{\infty} p_k = \rho \left[\sum_{k=1}^{n} k \frac{\rho^k}{k!} + n\rho^n \sum_{n+1}^{\infty} \frac{\beta^{n-k}}{n!} \right] \\ &= p_0 \rho \left[\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \sum_{k=n}^{\infty} \beta^{k-n} \right] p_0 = \rho = \lambda/\mu = n\beta.\end{aligned} \tag{29}$$

Priemerný počet zákazníkov v systéme sa oproti počtu zákazníkov vo fronte líši o priemerný počet práve obsluhovaných zákazníkov:

$$\kappa = \sum_{k=0}^{\infty} k p_k = \sum_{k=0}^{n-1} k p_k + \sum_{k=n}^{\infty} (k-n) p_k + \sum_{k=n}^{\infty} n p_k = \nu + \gamma. \tag{30}$$

Pre rozloženie doby čakania uplatníme rovnaký prístup ako v modeli M/M/1/∞. Pravdepodobnosť, že k-ty zákazník čaká dlhšie než w je rovná pravdepodobnosti, že po dobu w nebolo obslúžených viac ako $k-1$ zákazníkov:

$$e^{-n\mu w} \sum_{j=0}^{k-1} \frac{(n\mu w)^j}{j!}. \tag{31}$$

Pomocou tohoto vzťahu dokážeme vyjadriť rozloženie doby čakania bez informácie o poradí zákazníka:

$$\begin{aligned}P(W > w) &= p_n e^{-n\mu w} \sum_{k=0}^{\infty} \beta^k \sum_{j=0}^{k} \frac{(n\mu w)^j}{j!} \\ &= p_n e^{-n\mu w} \sum_{j=0}^{\infty} \frac{(n\mu w)^j}{j!} \sum_{k=j}^{\infty} \beta^k \\ &= p_n e^{-n\mu w} \sum_{j=0}^{\infty} \frac{(n\mu w\beta)^j}{j!} \frac{1}{1-\beta} \\ &= \frac{p_n}{1-\beta} e^{-n\mu w} \sum_{j=0}^{\infty} \frac{(\lambda w)^j}{j!} \\ &= -\pi e^{-(n\mu-\lambda)w}.\end{aligned} \tag{32}$$

Vypočítaním strednej hodnoty získame predpis pre strednú dobu čakania vo fronte:

$$\begin{aligned} E(w) &= \int_0^\infty w \frac{d}{dw}[1 - \pi e^{-(n\mu-\lambda)w}]dw \\ &= -\pi \int_0^\infty w \frac{d}{dw} e^{-(n\mu-\lambda)w} dw \\ &= -\pi \left[w e^{-(n\mu-\lambda)w} \right]_0^\infty + \pi \int_0^\infty e^{-(n\mu-\lambda)w} dw \\ &= \frac{\pi}{n\mu - \lambda}. \end{aligned} \tag{33}$$

Stredná doba čakania v systéme je opäť dlhšia o dobu obsluhy:

$$E(R) = E(w) + \frac{1}{\mu}. \tag{34}$$

1.3.4 Systém M/M/n/m

Tento model je variáciou systému M/M/n/∞ s počtom zákazníkov v systéme obmedzeným na m. Platí pritom, že $n \leq m$. Takisto vzhľadom na konečnú dĺžku frontu pripúšťame intenzitu prevádzky $\beta \geq 1$.

Pravdepodobnosti p_k pre $k > 0$ majú rovnaký predpis ako v prípade M/M/n/∞:

$$p_k = \begin{cases} \frac{\rho^k}{k!} p_0 & \text{pre } 0 < k \leq n, \\ \frac{\rho^k}{n^{k-n} n!} p_0 & \text{pre } n \leq k \leq m. \end{cases} \tag{35}$$

Z rovnice $1 = \sum p_k = \sum_{k=0}^{n-1} p_k + \sum_{k=n}^{m} p_k$ vyjadríme p_0 ako

$$\begin{aligned} p_0 &= \left(\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \sum_{k=n}^{m} \frac{\rho^k}{n^{k-n} n!} \right)^{-1} \\ &= \left(\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \sum_{k=n}^{m} \beta^{k-n} \right)^{-1} \\ &= \begin{cases} \left[\frac{\rho^n}{n!} \frac{1-\beta^{m-n+1}}{1-\beta} + \sum_{k=0}^{n-1} \frac{\rho^k}{k!} \right]^{-1} & \text{pre } \beta \neq 1, \\ \left[\frac{\rho^n}{n!} (m-n+1) \sum_{k=0}^{n-1} \frac{\rho^k}{k!} \right]^{-1} & \text{pre } \beta = 1. \end{cases} \end{aligned} \tag{36}$$

Priemerný počet zákazníkov vo fronte [5, str. 56]:

$$\begin{aligned}\gamma &= \sum_{k=n+1}^{m}(k-n)*p_k = \sum_{k=n+1}^{m}(k-n)\frac{\lambda^k}{n^{k-n}n!\mu^k}p_0 \\ &= \frac{p_0\rho^n}{n!}\sum_{k=n+1}^{m}(k-n)\frac{\rho^{k-n}}{n^{k-n}} = \frac{p_0\rho^n\beta}{n!}\sum_{k=n+1}^{m}(k-n)\beta^{k-n-1} \\ &= \frac{p_0\rho^n\beta}{n!}\sum_{i=1}^{m-n}i\beta^{i-1} = \frac{p_0\rho^n\beta}{n!}\frac{d}{da}\left(\sum_{i=1}^{m-n}\beta^i\right) \\ &= \frac{p_0\rho^n\beta}{n!}\frac{d}{da}\left(\frac{1-\beta^{m-n+1}}{1-\beta}\right).\end{aligned} \tag{37}$$

Po spätnej derivácii sa výraz upraví na

$$\gamma = \frac{p_0\rho^n\beta}{n!\,(1-\beta)^2}\left[1-\beta^{m-n+1}-(1-\beta)(m-n+1)\beta^{m-n}\right]. \tag{38}$$

Označíme skutočnú hodnotu vstupnej intenzity (berúc do úvahy pravdepodobnosť odmietnutia zákazníka) ako $\bar{\lambda} = \lambda(1-p_m)$. Potom priemerný počet zákazníkov v systéme je

$$\kappa = \gamma + \frac{\bar{\lambda}}{\mu} = \gamma + \rho(1-p_n). \tag{39}$$

Využitím Littlovho zákona [4] získavame strednú dobu strávenú v systéme

$$E(R) = \frac{\kappa}{\lambda(1-p_n)} \tag{40}$$

a strednú dobu strávenú vo fronte

$$E(W) = \frac{\gamma}{\lambda(1-p_n)}. \tag{41}$$

1.3.5 Systém M/D/1/∞

Na rozdiel od predchádzajúcich modelov sa tento líši rozložením doby obsluhy. Príchod zákazníkov sa stále riadi Poissonovym procesom s parametrom λ, avšak dobu obsluhy považujeme za konštantnú s dĺžkou $d = 1/\mu$. Môže ísť napríklad o prevádzky, kde obsluhu vykonáva stroj (automaty na lístky, lanovka, ...).

Prechodové pravdepodobnosti odvodené v [1, str. 20]:

$$\begin{aligned} p_{jk} &= e^{-\lambda d}(\lambda d)^k! = e^{-\rho}\frac{\rho^k}{k!} \text{ pre } j = 0, k \geq 0, \\ p_{jk} &= e^{-\lambda d}(\lambda d)^{k-j+1}/(k-j+1)! = e^{-\rho}\rho^{k-j+1}! \text{ pre } j > 0, k \geq j-1, \\ p_{jk} &= 0 \text{ pre } 0 \leq k < j-1. \end{aligned} \tag{42}$$

Intenzita prevádzky je teda $\rho = \lambda/\mu = \lambda d$ a rovnice pravdepodobnosti stavov procesu $X(t)$, kde $X(t)$ je počet zákazníkov v systéme v čase t sú:

$$\begin{aligned} p_0(t+d) &= p_0(t)e^{-\rho} + p_1(t)e^{-\rho}, \\ p_k(t+d) &= e^{-\rho}\left[p_0(t)\frac{\rho^k}{k!} + \sum_{j=1}^{k} p_j(t)\rho^{k-j+1}/(k-j+1)! + p_{k+1}(t)\right] \text{ pre } k > 0. \end{aligned} \tag{43}$$

V ustálenom stave z toho dostaneme:

$$\begin{aligned} p_0 e^{\rho} &= p_0 + p_1, \\ p_k e^{\rho} &= p_0\frac{\rho^k}{k!} + \sum_{j=1}^{k} p_j \rho^{k-j+1}/(k-j+1)! + p_{k+1}. \end{aligned} \tag{44}$$

Pre potreby našej aplikácie vieme explicitne vyjadriť formuly pre stacionárne rozdelenie pravdepodobností p_k a pravdepodobnosti čakacej doby. Úplné odvodenie možno nájsť v [1, str. 21-22].

$$\begin{aligned} p_k &= p_0 \sum_{j=0}^{k-1} (-1)^{k-j} e^{j\rho}\left[\frac{(j\rho)^{k-j}}{(k-j)!} + \frac{(j\rho)^{k-j-1}}{(k-j-1)!}\right] + e^{k\rho}, \\ p_0 &= 1-\rho. \end{aligned} \tag{45}$$

$$F(w) = (1-\rho)\sum_{k=0}^{n} e^{\rho(\mu w - k)}\frac{[-\rho(\mu w - k)]^k}{k!}, \tag{46}$$

kde $n = [\mu w]$.

Stredná hodnota:

$$E(w) = \frac{\rho}{2\mu(1-\rho)}. \tag{47}$$

2 Návrh webovej aplikácie

Táto kapitola slúži na stručný popis použitých technológií a programovacích jazykov, ako aj návrh webovej aplikácie vrátane dizajnu, funkcionality a používateľského rozhrania. Popíšeme ciele práce z programátorského pohľadu a naznačíme postupy, ktorými ich bude možné dosiahnuť.

2.1 Použité technológie

Aplikácia pobeží na serveroch Google na adrese **qtcalculator.appspot.com**. Ako platformu použijeme **Google App Engine(GAE)**, ktorý umožňuje hosting a správu aplikácie. Webová stránka aplikácie využije jazyk **HTML** spolu s **CSS** pre úpravu vizuálnej stránky. Ako programovací jazyk sme zvolili **Python** spolu so svojimi rozšíreniami **Webapp2** a **Jinja**. Pre niektoré dynamické zobrazovacie metódy a výpočty, ktoré musia byť na stránke realizované v reálnom čase používame **Javascript**. V tejto sekcii povieme viac o jednotlivých technológiách a jazykoch a ich použití v aplikácii pre teóriu hromadnej obsluhy.

2.1.1 Google App Engine

Google App Engine je takzvaný *Platform As A Service (PaaS)*. Znamená to, že zákazníkom umožňuje vytvárať, prevádzkovať a udržiavať svoju aplikáciu na infraštruktúre Google. GAE podporuje širokú ponuku programovacích jazykov a ich rozšírení, z najpopulárnejších to sú napríklad Java, PHP, Python alebo GO. Okrem toho ponúka ďalšie služby, ako napríklad dátový sklad, správu používateľských účtov a iné. Takisto umožnuje využiť ďalšie rozšírenia. V našom prípade pri použití jazyka Python sú to napríklad *webapp2* a *Jinja*, spomínané neskôr, alebo *Numpy*, ktorý umožnuje vykonávať zložitejšie matematické operácie prakticky na úrovni platených matematických softvérov, ako je Matlab. Práve deklarovaná podpora Numpy bola jedným z hlavných dôvodov, pre ktorý som sa rozhodol využiť GAE spolu s jazykom Python.

Niektoré služby a tiež veľmi veľké aplikácie sú platené. Na našu aplikáciu ale budú stačiť voľne dostupné funkcie.

2.1.2 HTML a CSS

HTML, alebo HyperText Markup Language je najpopulárnejší značkovací jazyk pre tvorbu webových stránok používaný už od roku 1993. Nasledoval vývoj prehliadačov, ktoré dokážu zobrazovať HTML kód do podoby, v akej web čítame dnes. HTML však nie je programovací jazyk, na prácu s dátami, premennými, alebo na dynamickú tvorbu obsahu sa preto používajú ďalšie jazyky ako PHP alebo Javascript.

Aplikácia teórie hromadnej obsluhy využíva funkcie najnovšej verzie HTML 5. Je preto potrebné ju používať na prehliadačoch, ktoré túto verziu podporujú. Avšak najnovšie verzie všetkých najpopulárnejších prehliadačov s HTML5 nemajú problém.

CSS je skratkou pre Cascading Style Sheets. Jeho hlavnou úlohou je oddeliť vzhľad dokumentu od HTML. Umožňuje meniť vzhľad aj štruktúru dokumentu. Spravidla sa pre jednoduchšie editovanie CSS kód uchováva v osobitnom súbore.

2.1.3 Python, webapp2, Jinja

Python je objektovo orientovaný programovací jazyk so všeobecným účelom. Dôraz je kladený na produktivitu programátora. Cieľom je ľahko čitateľný kód, ktorý je navyše priestorovo úsporný. Od väčšiny iných jazykov sa Python odlišuje aj tým, že namiesto zátvoriek, alebo iných znakov slúžiacich na ohraničenie blokov kódu, používa výhradne formátovanie a zarovnanie textu.

Webapp2 je rozšírenie pre Python, ktoré má za úlohu zjednodušiť komunikáciu s webovou aplikáciou. Je možné ho použiť spolu s GAE, ale aj na vlastnej platforme.

Jinja je tzv. „template engine“. Umožňuje používať pythonovský kód a premenné priamo v HTML súboroch (šablónach).

2.1.4 Javascript

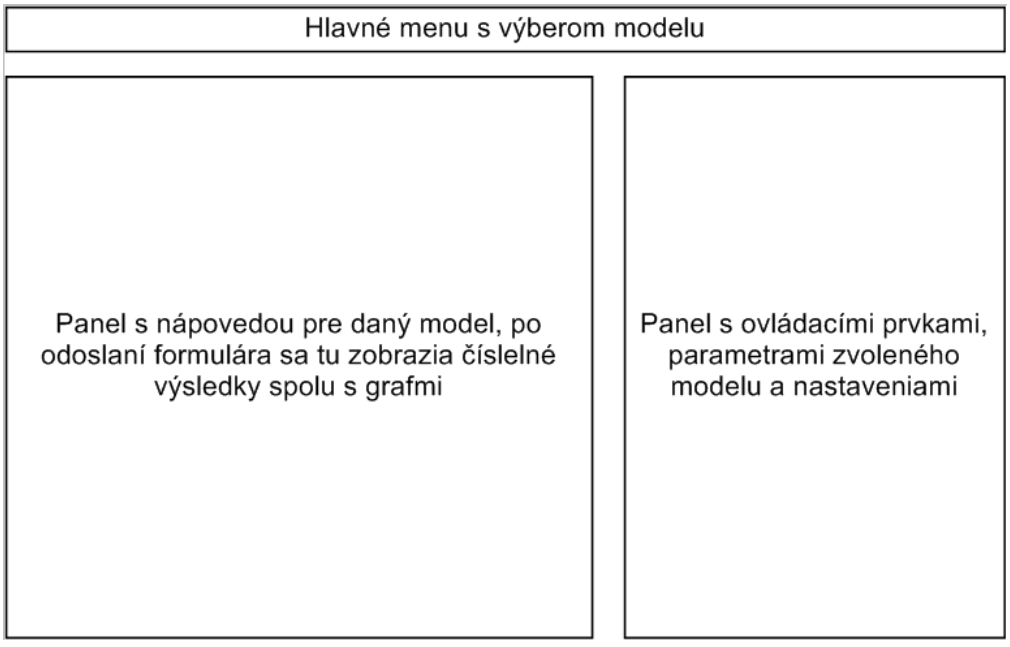
Javascript je skriptovací jazyk používaný najmä pri tvorbe webových stránok. Napriek svojmu názvu nemá nič spoločné s populárnym jazykom Java. Javascript býva často vkladaný priamo do HTML kódu. Používa sa hlavne na ovládanie interaktívnych prvkov na stránke. V našom prípade to bude tvorba grafov, dynamické zobrazovanie obsahu, výpočty zo vstupných hodnôt bez odoslania formulára alebo obnovenia stránky a v neposlednom rade na interpretovanie LaTexového zdrojového kódu na stránke.

Na rozdiel od iných webových programovacích jazykov ako PHP alebo Python sa Javascript kód interpretuje až po načítaní stránky a funguje výhradne na strane klienta. Práve nutnosť nutnosť opätovného obnovovanie stránky pri každej interakcii s používateľom ma motivovala využiť Javascript. Okrem hlavných výpočtov, ktoré stále prebiehajú na strane servera po odoslaní formulára, je možné využívať ostatné spomínané funkcie na strane klienta, čo podľa môjho názoru výrazne spohodlní užívateľovi prácu s aplikáciou.

2.2 Návrh užívateľského rozhrania

Pre čo najširšiu prístupnosť bude aplikácia voľne dostupná na internete na adrese **qt-calculator.appspot.com**. Z rovnakého dôvodu bol ako jazyk pre užívateľské rozhranie zvolená angličtina.

Všetky ovládacie prvky budú jednoducho dostupné. Používateľ si v hlavnej ponuke vyberie model, na ktorý chce aplikovať výpočty. Následne bude mať možnosť pre vybraný model upresniť presné parametre. Okrem toho bude môcť užívateľ nastaviť, na koľko desatinných miest chce zaokrúhliť výsledky. Tiež bude možnosť riešiť niektoré optimalizačné úlohy, ako napríklad nájsť optimálny počet obslužných liniek alebo porovnať straty dvoch rôznych modelov. Toto rozloženie možno vidieť na Obr.2



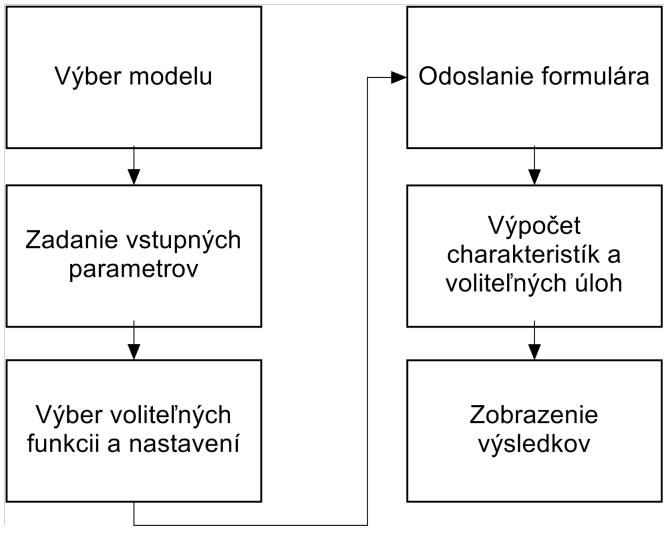
Obr. 2: Rozloženie webovej aplikácie

2.3 Vnútorný návrh aplikácie

Výber modelu v hlavnej ponuke načíta podstránku pre zvolený model. Tu sa zobrazí formulár s prednastavenými hodnotami, ktoré môže používateľ zmeniť. Údaje sa po zadaní do formulára prevedú z HTML na pythonovské premenné a následne vstúpia ako parametre do jednej z výpočtových funkcií. Táto bude jedinečná pre každý model.

Vo funkcii najprv dôjde ku kontrole správnosti dát. Overí sa, či boli zadané všetky potrebné údaje a či tieto nadobúdajú povolené hodnoty. Pokiaľ niektorá z podmienok nebude splnená, aplikácia vráti chybovú hlášku.

Ak kontrola prebehne bez chýb, algoritmus pokračuje výpočtom charakteristík modelu. Základné charakteristiky sa vypočítajú automaticky, niektoré optimalizačné úlohy iba ak o to používateľ požiadal pri zadávaní údajov. Výsledné hodnoty sa zaokrúhlia na požadovaný počet desatinných miest a odošlú sa späť v premennej typu **slovník**. Premenné tohoto typu môžu byť prostredníctvom Jinja2 použité ako parameter v pripravenej HTML šablóne a následne zobrazené pre používateľa. Zjednodušenú podobu tejto schémy možno vidieť na Obr.3



Obr. 3: Schéma funkcionality

2.4 Výstupy z aplikácie

Výstupy budú prevažne vo forme textu. Pre niektoré charakteristiky, ako napríklad pravdepodobnosť určitého počtu zákazníkov v systéme alebo rozloženie doby obsluhy,

bude možné tieto zobraziť graficky. Pre tieto bude tiež možné zobraziť konkrétnu hodnotu (pre zvolené k alebo w) priamym zadaním. Pre túto funkciu budeme na výpočty používať namiesto jazyka Python Javascript.

2.5 Úlohy vhodné na riešenie aplikáciou

Pre rôzne typy modelov vieme vo všeobecnosti riešiť iné druhy úloh. Výstupy spoločné pre všetky modely, pre výpočet ktorých si vystačíme so základnými parametrami modelu (intenzity vstupného a výstupného prúdu, počet obslužných liniek a maximálna dĺžka frontu) sú tieto:

- intenzita prevádzky,
- pravdepodobnosť toho, že v systéme je k zákazníkov,
- priemerný počet zákaznikov v systéme,
- priemerný počet zákaznikov vo fronte,
- čakacia doba k-teho zákazníka vo fronte,
- čakacia doba vo fronte bez ohľadu na poradie zákazníka,
- priemerná doba vo fronte,
- priemerná doba v systéme.

Na riešenie týchto úloh si vystačíme so vzorcami popísanými v 1.

Optimalizačné úlohy sa vo všeobecnosti líšia v závislosti na zvolenom modeli. Pre systém M/M/n/m vieme napríklad minimalizovať stratu vzniknutú odmietnutím zákazníka pomocou zmeny počtu obslužných liniek, miest vo fronte, alebo v oboch súčasne. Voliť optimálny počet liniek môžeme aj v úlohe M/M/n/∞, optimálny počet miest vo fronte zase v systéme M/M/1/n. Minimalizovať náklady vzniknuté chodom, resp. prestojom linky môžeme v modeli M/M/1/∞.

3 Implementácia

3.1 Základ aplikácie

Základ aplikácie bežiacej pod Google App Engine sú súbory **app.yaml** a **main.py**. Prvá z nich obsahuje názov aplikácie spolu s verziami použitých programovacích jazykov a knižníc, ako aj adresu k statickým súborom (obrázky, HTML a CSS súbory), ktoré aplikácia využíva.

```
application: qtcalculator
version: 1
runtime: python27
api_version: 1
threadsafe: false

handlers:
- url: /files
  static_dir: files

- url: /.*
  script: main.application

libraries:
- name: webapp2
  version: "2.5.2"
- name: jinja2
  version: latest
- name: numpy
  version: "1.6.1"
```

Súbor **main.py** slúži ako ústredné ovládacie prostredie aplikácie. Tu napríklad nastavujeme, ktoré metódy majú kedy prebehnúť. Tento kúsok kódu aktivuje metódu podľa zvoleného modelu.

```
application = webapp2.WSGIApplication([
    ('/', MainPage),
    ('/MM1N', MM1N),
    ('/MM1Inf', MM1Inf),
    ('/MMNInf', MMNInf),
    ('/MD1Inf', MD1Inf),
    ('/MMCK', MMCK),
], debug=True)
```

Nasledovný postup si kvôli zachovaniu prehľadnosti ilustrujeme na jedinom modeli - M/M/n/m. Ostatné modely je možné naprogramovať analogicky.

3.1.1 Zadávanie údajov

Po zvolení modelu sa zobrazí HTML formulár so všetkými nastaveniami a parametrami. Jednotlivé vstupné polia sú označené identifikátormi, pomocou ktorých zadané hodnoty vieme načítať po odoslaní formulára ako pythonovské premenné.

```
<form action = "/MMCK" method = "post">
 <div id="parametre">
       <input name = "lambda" type = "number" min = 0 step="any" value =
           "0.5" >Arrival Rate - <span style = "font-size:20px;"
           lang="latex">\lambda</span><br />
       <input name = "mu" type = "number" min = 0 step="any" value = "1"
           >Service Rate - <span style = "font-size:20px;"
           lang="latex">\mu</span><br />
       <input name = "C" type = "number" min = "1" step="1" value = "1"
           >Number of servers<br />
       <input name = "K" type = "number" min = "1" step="1" value = "1" >MAX
           system size<br />
                      Round to <input name = "rounding" type = "number" min =
                          1 step="any" value = 3 >decimals<br /><br />
```

```
 </div>
       ....
 <br />
 <div id = "subButton">
 <input type="submit" class = "btn" value="Calculate">
 </div>
</form>
```

3.1.2 Spracovanie údajov a výpočet charakteristík

Takto zadané parametre sa v **main.py** spracujú pomocou metódy **MMCK**. Po odoslaní formulára sa načítajú pomocou **self.request.get(„id“)** a odošlú do funkcie **MMCK_vals**, ktorú načítavame z externého súboru **metódy**.

```
class MMCK(webapp2.RequestHandler):
    def get(self):
        template = JINJA_ENVIRONMENT.get_template('MMCKentry.html')
        self.response.write(template.render())
    def post(self):
        l = self.request.get("lambda")
        m = self.request.get("mu")
        maxk = self.request.get("maxk")
        w = self.request.get("w")
        K = self.request.get("K")
        C = self.request.get("C")
        r = self.request.get("rounding")
        ......
        values = metody.MMCK(l, m, maxk, K, C, ... , costC, costK)
        template = JINJA_ENVIRONMENT.get_template('MMCKresult.html')
        self.response.write(template.render(values))
```

Tu dôjde k výpočtu charakteristík a optimalizačných úloh. Najskôr ale dáta testujeme pre správnosť vstupných údajov. Pokiaľ niektorý potrebný údaj chýba, alebo je nesprávny (nulové počty servisných liniek, nepovolené hodnoty intenzity prevádzky, ...), funkcia vráti hodnoty v premennej **val** spolu s príznakom pre chybu **err** a chybovým hlásením informujúcim o type problému **errMSG**.

```
try:
        l = float(l)
        ....
        r = int(float(r))
        if alt ==1:
                prof = float(prof)
                cost = float(cost)
                rc = float(rc)
                if altmodel == 1:
                                altK = int(float(altK))
                .......
        err = 0
except:
        err = 1
        val = {
                'err': err,
                'beta': 0,
                'w': w,
                ....
                'errMSG': "Input is missing or in a wrong format",
        }
        return val
```

Pokiaľ nedošlo k chybe, pokračujeme k výpočtu charakteristík. Ako prvé označíme pomocné premenné ako β alebo *rho*. Ďalej počítame pravdepodobnosti p_k. Využívame až tri premenné - dve ďalšie slúžia na neskoršie zobrazenie graficky pre diskrétne a

kumulatívne pravdepodobnosti pomocou Javascript.

```
p = []
Peq = []
Pleq = []
Peq.append(['n', 'probabilities'])
Pleq.append(['n', 'cumulative probabilities'])
sumPom = 0
for i in range(0,C):
       sumPom = sumPom + rho**i/factorial(i)
if beta == 1:
       sumPom = sumPom + ((rho**C)/factorial(C)) * (K-C+1)
else:
       sumPom = sumPom + ((rho**C)/factorial(C)) *
           (1-(beta**(K-C+1)))/(1-beta)
p0 = 1/sumPom
pn = (rho**C*p0)/factorial(C)
p.append(p0)
Peq.append([0, p0])
Pleq.append([0, p0])
for i in range(1, K+1):
       if i>C:
              p.append(rho**i*p0/(factorial(C)*C**(i-C)))
              Peq.append([i, rho**i*p0/(factorial(C)*C**(i-C))])
       else:
              p.append(rho**i*p0/factorial(i))
              Peq.append([i, rho**i*p0/factorial(i)])
       Pleq.append([i, sum(p)])
```

S pomocou pravdepodobností dokážeme vypočítať ostatné základné charakteristiky.

```
Qhat =
    rho**C*beta*p0/(factorial(C)*(1-beta)**2)*(1+(K-C)*beta**(K-C+1)-(K-C+1)*beta**(K-C))
Lhat = l*(1-p[K])
What = Qhat/Lhat
sumPom1 = 0
sumPom2 = 0
sumPom3 = 0
for i in range(0,C):
        sumPom1 = sumPom1 + i*p[i]
        sumPom2 = sumPom2 + p[i]
        sumPom3 = sumPom3 + ((C-i)*rho**i)/factorial(i)
Nhat = Qhat + sumPom1 + C*(1-sumPom2)
That = What + 1/m
PI = 0
for i in range(0,C):
        PI = PI + p[i]
PI = 1 - PI
gamma = Qhat;
kappa = Nhat
PP = PI
EW = What
ER = That
```

Nasledované rozložením doby čakania:

```
# waiting time probabilities based on position
PWk = []
for j in range(1, K+1):
        sumK = 0
        for i in range(0, j):
                        sumK = sumK + ((C*m*w)**i)/(factorial(i))
        PWk.append(round(math.exp(-(C*m*w)) * sumK, r))
```

```
# waiting time probability
PW = 0
sumI = 0
for i in range(0, K+1):
        sum1 = 0
        for j in range(0, i+1):
                sum1 = sum1 + ((C*m*w)**j)/factorial(j)
        sumI = sumI + (beta**i) * sum1
PW = p[C] * math.exp(-C*m*w) * sumI
```

Ako posledné počítame optimalizačné úlohy (uvedené porovnanie strát dvoch modelov):

```
# loss analysis
loss = 0
lossalt = 0
if alt == 1 and altmodel==1:
        # probs for ALT model
        sumPom = 0
        pAlt = []
        for i in range(0,altC):
                sumPom = sumPom + rho**i/factorial(i)
        if betaAlt == 1:
                sumPom = sumPom + ((rho**altC)/factorial(altC)) * (altK-altC+1)
        else:
                sumPom = sumPom + ((rho**altC)/factorial(altC)) *
                    (1-(beta**(altK-altC+1)))/(1-betaAlt)
        p0 = 1/sumPom
        pn = (rho**C*p0)/factorial(altC)
        pAlt.append(round(p0, r))
        for i in range(1, altK + 1):
```

```
            if i>altC:
                    pAlt.append(round(rho**i*p0/(factorial(altC)*altC**(i-altC)),
                        r))
            else:
                    pAlt.append(round(rho**i*p0/factorial(i), r))
    loss = l * p[K] * prof * rc
    lossalt = l * pAlt[altK] * prof * rc + cost * (altC - C)
```

Všetky hodnoty spolu so vstupnými parametrami priradíme do výstupnej premennej **val** (typu **slovník**) a vraciame hodnoty späť metóde **MMCK**.

```
val = {
    #computed vars
    'rho': round(rho, r),
    ......
    'EW': round(EW, r),
    'err': err,
    'loss': loss,
    'lossalt': lossalt,

    #old vars
    'w': w,
    'l': l,
    'C': C,
    'K': K,
    .....
    'KcostC': KcostC,
}
return val
```

3.2 Zobrazenie výsledkov

Použitím Jinja vraciame nadobudnuté premenné HTML šablóne.

```
values = metody.MMCK(l, m, maxk, K, C, ... , costC, costK)
template = JINJA_ENVIRONMENT.get_template('MMCKresult.html')
self.response.write(template.render(values))
```

V HTML najprv prebieha kontrola, či nedošlo k chybe. Umožnenie pythonovského kódu a výstupných premenných z metódy MMCK priamo v HTML slúži **webapp** a **Jinja**. V prípade, že došlo k chybe, namiesto výstupného formulára sa zobrazí hlásenie o chybe.

```
{%if err==1%}
                    Wrong or incomplete Input<br />
                    {%if beta >=1%}
                            Beta must be less than 1, but it is equal to
                                {{beta}}<br />
                    {%endif%}
                    <br />
            {{errMSG}}
```

Ak príznak **err** nehlási chybu, pokračujeme k zobrazeniu výstupov.

```
<input type = "number"id = "pID"> probability of <span id = pIDk> 0 </span>
    people being in the system is <span id = "pIDres"></span><br />
<input type = "number"id = "pwID"> probability of waiting for more than
    <span id = pwIDk> 0 </span> is <span id = "pwIDres"></span><br />
<input type = "number"id = "pwkID"> probability of <span id =
    pwkIDk>1</span>. customer waiting for more than <span id = pwkIDw> 0
    </span> is <span id = "pwkIDres"></span><br />
```

```
<input type="checkbox" name="hideprobchart" id="hideprobchart"
    checked="true"><label for = "hideprobchart">display/hide the
    probabilities chart</label><br/>
<input type="checkbox" name="hideprobleqchart" id="hideprobleqchart"
    checked="true"><label for = "hideprobleqchart">display/hide cumulative
    probability chart</label><br/>
 <input type="checkbox" name="hidePWchart" id="hidePWchart"
     checked="true"><label for = "hidePWchart">display/hide chart for waiting
     time probabilities</label><br/>
<div id="PWchart" ></div>
<div id="probchart" ></div>
<div id="probleqchart" ></div>
{%set i=0%}
{%for item in p:%}
       P{{i}} = {{ item }} <br />
       {%set i = i + 1%}
{%endfor%}

<h3>RESULTS FOR M/M/C/K</h3>
<h4>Basic model results</h4>
<div id = "res">
       Beta - service intensity:<br />
       {{beta}}<br /><br />
       Average number of customers in system:<br />
       {{kappa}}<br /><br />
       Average number of customers in queue:<br />
       {{gamma}}<br /><br />
       Average number of busy servers:<br />
       {{rho}}<br /><br />
       Probability of waiting <span lang="latex">\Pi</span>:<br />
       {{PP}}<br /><br />
       Average time spent in queue:<br />
       {{EW}}<br /><br />
```

```
        Average time spent in system:<br />
        {{ER}}<br /><br />
</div>

{%if alt ==1%}
<br />
Loss on the first model: {{loss}}<br />
Loss on alternate model: {{lossalt}}
{%endif%}
```

Tu okrem HTML a Jinja využívame aj Javascript. Jeho hlavným účelom je skrytie a opätovné zobrazenie objektov, grafické zobrazovanie, ale tiež opätovný výpočet pravdepodobností pre okamžité a dynamické zobrazenie. Skripty Javascript načítavame zo súboru **scripts.html**. Tu využijeme výstupné premenné z pythonovských výpočtových funkcií, ktoré slúžia ako zdrojové dáta pre výsledné charakteristiky a zobrazené grafy. Z dôvodu, že používateľ má možnosť nechať zobraziť pravdepodobnosti p_k a najmä pravdepodobnosť čakania pre akýkoľvek zadaný čas, je nutné tieto charakteristiky na rozdiel od ostatných vypočítať v Javascripte. Alternatívou by bola nutnosť odoslania formulára pri akejkoľvek zmene preferencií.

Tu je príklad využitia Javascript pre zobrazenie dát graficky cez **Google Charts** a ich následné skrytie/odkrytie pomocou dynamického nastavenia parametrov HTML objektov.

```
google.setOnLoadCallback(drawChart);

function drawChart() {
        var data1 = google.visualization.arrayToDataTable(dataset1);
        var data2 = google.visualization.arrayToDataTable(dataset2);
        var data3 = google.visualization.arrayToDataTable(dataset3);

        var options1 = {
```

```
            title: 'Probability of N people being in the system',
            curveType: 'function',
            'width': 560,
            'height': 350,
            'chartArea': {'width': '90%', 'height': '80%'},
            'legend': {'position': 'bottom'}
        };
        ..........

        var chart1 = new
            google.visualization.ColumnChart(document.getElementById('probchart'));
        var chart2 = new
            google.visualization.ColumnChart(document.getElementById('probleqchart'));
        var chart3 = new
            google.visualization.LineChart(document.getElementById('PWchart'));

        chart1.draw(data1, options1);
        chart2.draw(data2, options2);
        chart3.draw(data3, options3);
}


$('[name=hideprobchart]').change(function(){
    var c = this.checked ? 'block' : 'none';
        $('#probchart').css('display', c);
});

$('[name=hideprobleqchart]').change(function(){
    var c = this.checked ? 'block' : 'none';
        $('#probleqchart').css('display', c);
});

$('[name=hidePWchart]').change(function(){
```

```
    var c = this.checked ? 'block' : 'none';
        $('#PWchart').css('display', c);
});
```

Vo výsledku má používateľ nad zobrazeným obsahom kontrolu jednak pri zadávaní údajov, kde môže nastaviť voliteľné typy úloh, ako aj želané zaokrúhlenie výsledkov a takisto pomocou vstupov typu **checkbox** môže zobraziť alebo skryť celé bloky výstupu alebo grafu, a takisto si môže nechať zobraziť práve tie pravdepodobnosti ktoré ho zaujímajú.

4 Ilustračné príklady

Funkcionalitu webovej aplikácie si predvedieme na niekoľkých príkladoch. Konkrétne necháme zobraziť všetky charakteristiky modelu textovo aj graficky. V druhom príklade vyriešime pomocou aplikácie optimalizačný príklad [2, Pr. 13].

4.1 Príklad 1

Našim cieľom je získať všetky dostupné charakteristiky pre model M/M/n/m s parametrami:

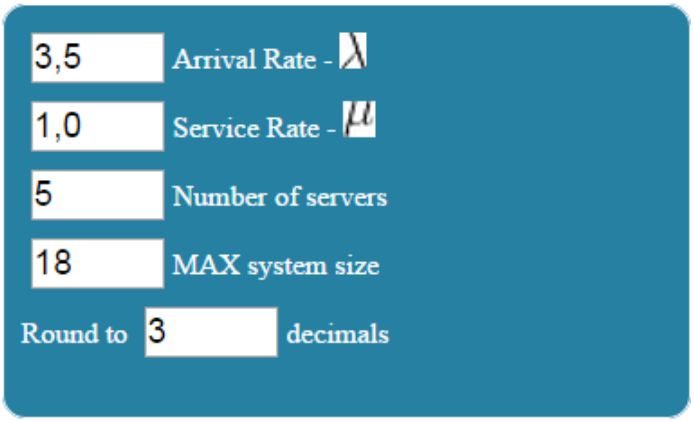
- Intenzita vstupu: $\lambda = 3.5$,
- Intenzita výstupu: $\mu = 1$,
- Počet obslužných liniek: $n = 5$,
- Maximálna kapacita systému: $m = 18$,

V hlavnom menu si volíme model M/M/n/m (Obr. 4) a do zobrazeného vstupného formulára zadáme požadované parametre (Obr. 5). Po stlačení tlačidla **Compute** sa zobrazí tabuľka so základnými charakteristikami(Obr. 6). Pre zobrazenie pravdepodobnosti konkrétneho počtu zákazníkov v systéme a pravdepodobnosti čakania na obsluhu po čas dlhší než w pre k-teho zákazníka vo fronte, ako aj bez poznania poradia, zadáme príslušné hodnoty k, w do textového poľa (Obr. 7). Pre lepšiu predstavu je tieto pravdepodobnosti možné zobraziť graficky pre všetky relevantné hodnoty k, w (Obr. 8, 9, 10).

Obr. 4: Príklad 1 - výber modelu

4.2 Príklad 2

V druhom príklade ilustrujeme užitočnosť aplikácie priamo na príklade zo zbierky cvičení [2, Pr. 13].



Obr. 5: Príklad 1 - zadanie parametrov

Informácie na informačnej telefónnej linke s priemerným trvaním 1 min majú exponenciálne rozloženie, záujem o informácie je Poissonovsky rozložený s priemerným počtom 30 za hodinu. Informácia stojí 50 hal/sec., polovica odmietnutých o informáciu opäť nepožiada. Aké priemerné straty má informačná kancelária v dôsledku odmietnutia zákazníkov? Ak prevádzka ďalšej linky stojí 100 Sk/hod., oplatí sa prevádzkovať ju?

Opäť zvolíme rovnaký model M/M/n/m, kde do vstupného formulára nastavíme parametre (Obr. 11):

- Intenzita vstupu: $\lambda = 0.5$,
- Intenzita výstupu: $\mu = 1$,
- Počet obslužných liniek: $n = 1$,
- Maximálna kapacita systému: $m = 1$,

Ide teda o model M/M/1/1. Úlohou je porovnať straty vzniknuté odmietnutím zákazníka oproti modelu M/M/2/2, pričom (po prepočítaní na koruny a hodiny) prevádzka ďalšej linky stojí 1,66 Sk/hod, zisk je 30 Sk/hod a v priemere polovica odmietnutých zákazníkov sa už nevráti. Zaškrtnutím políčok **Perform loss analysis** a **Analyse alternative model** dostávame možnosť nastaviť parametre alternatívneho modelu spolu s hodnotami ziskov a strát (Obr. 12).

Basic model results

Beta - service intensity:
0.7

Average number of customers in system:
4.338

Average number of customers in queue:
0.842

Average number of busy servers:
3.5

Probability of waiting Π :
0.376

Average time spent in queue:
0.241

Average time spent in system:
1.241

Obr. 6: Príklad 1 - Základné charakteristiky

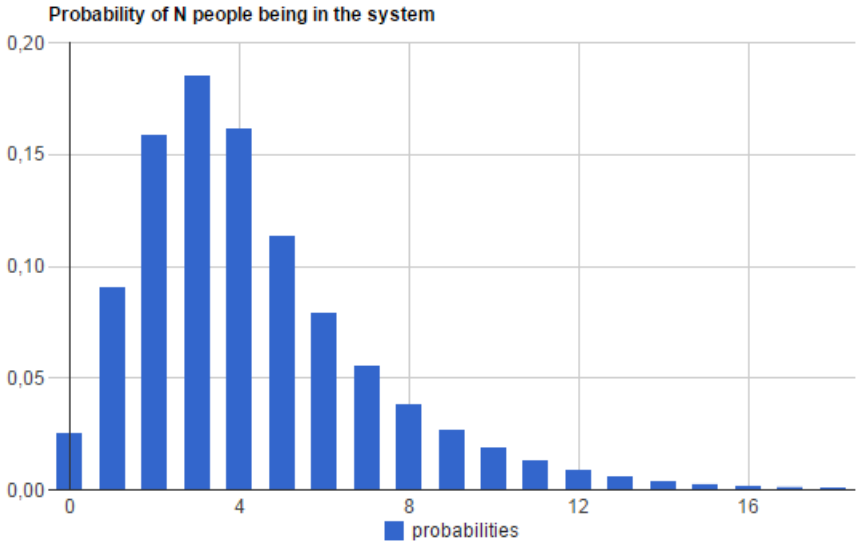
3 probability of 3 people being in the system is 0.18553908109757244

0,5 probability of waiting for more than 0.5 is 0.1785048809194051

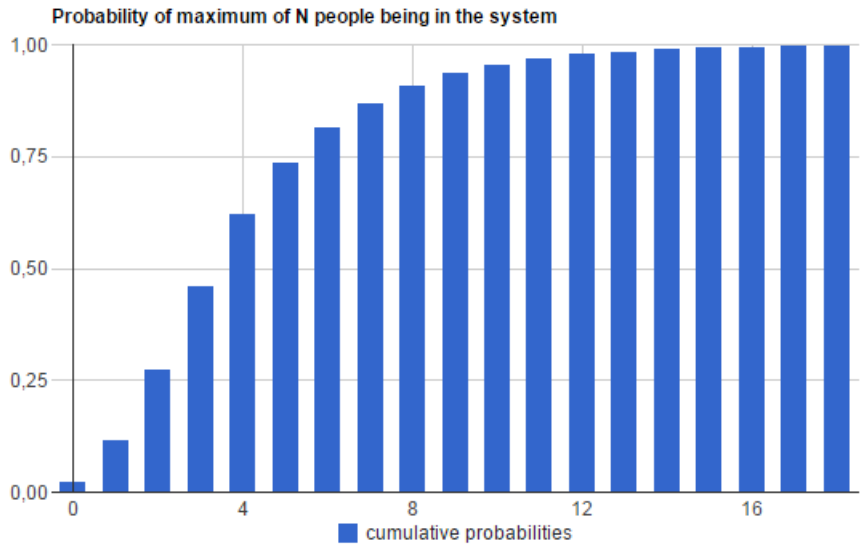
4 probability of 4. customer waiting for more than 0.5 is
0.7575761331330659

Obr. 7: Príklad 1 - Pravdepodobnosti v textovom poli

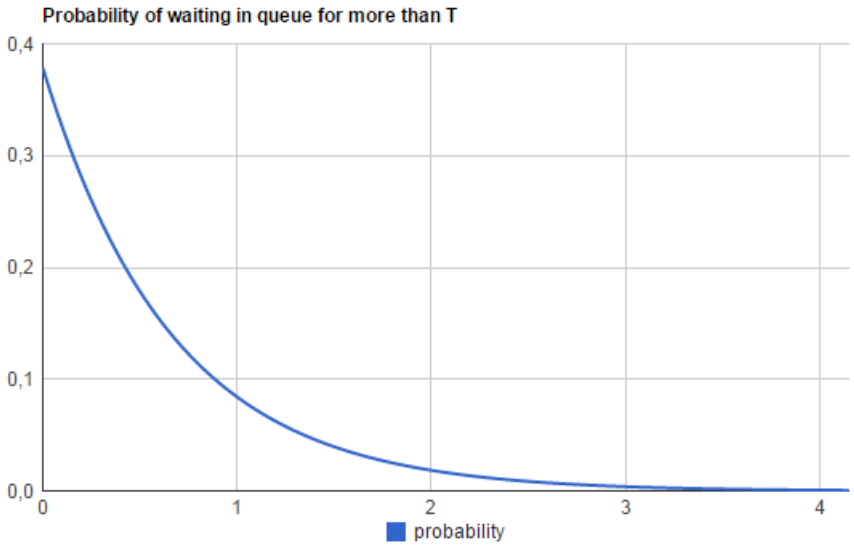
Po odoslaní formulára tlačidlom **Compute** nás zaujíma iba prehľad strát odmietnutím zákazníkov vyčíslené na hodinu (Obr. 13). Ako možno vidieť, alternatívny model s dvoma obslužnými linkami má nižšie straty a je pre prevádzkovateľa výhodnejší.



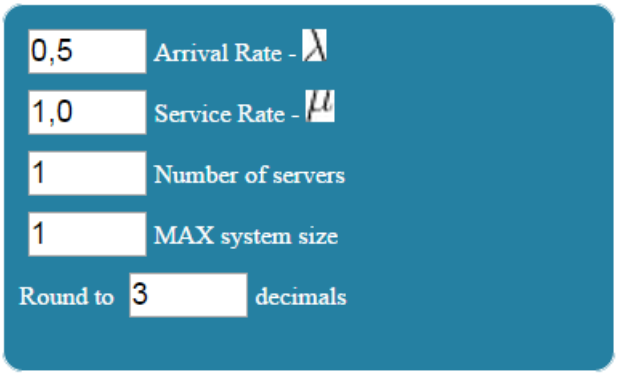
Obr. 8: Príklad 1 - Pravdepodobnosť počtu zákazníkov graficky



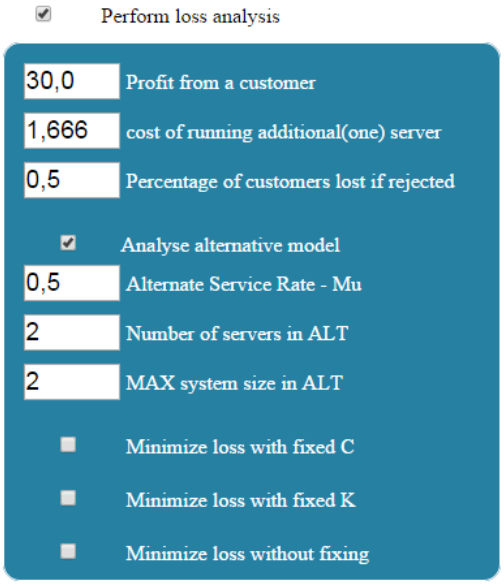
Obr. 9: Príklad 1 - Kumulatívna pravdepodobnosť počtu zákazníkov



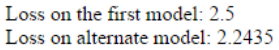
Obr. 10: Príklad 1 - Rozloženie doby čakania



Obr. 11: Príklad 2 - výber úlohy



Obr. 12: Príklad 2 - zadanie



Obr. 13: Príklad 2 - výsledok

ZÁVER

Záver

V diplomovej práci sme sa venovali popisu a aplikácii modelov teórie hromadnej obsluhy. Stručne sme popísali jednotlivé modely a odvodili vzorce potrebné pre pre výpočet charakteristík webovou aplikáciou.

Cieľom práce bolo navrhnúť a naprogramovať túto aplikáciu a urobiť ju čo možno najviac prístupnú a jednoduchú na používanie. Po navrhnutí dizajnu a funkcionality aplikácie sme túto vyvinuli na platforme Google App Engine, s podporou programovacích jazykov a ich rozšírení ako HTML, Python, Javascript, CSS, Webapp a Jinja. Aplikácia bola po dokončení umiestnená na voľne prístupný server Google na adrese *qtcalculator.appspot.com*. Vďaka svojmu umiestneniu a angličtine ako jazyku užívateľského rozhrania je dostupná komukoľvek a kdekoľvek.

Hlavné využitie aplikácie vidím ako pomôcku pri štúdiu na Fakulte matematiky, fyziky a informatiky, ale aj na ďalších školách. Taktiež je možné použiť ju ako nástroj na optimalizovanie behu skutočnej prevádzky, pokiaľ sa reálne vlastnosti prevádzky budú blížiť tým teoretickým, s ktorými teória hromadnej obsluhy pracuje.

V budúcnosti je možné na aplikácii ďalej pracovať, je možné zaviesť nové modely, alebo pridať ďalšie charakteristiky a optimalizačné úlohy k tým aktuálnym. Vďaka textu tejto práce, najmä kapitole 4 a komentárom priamo v zdrojovom kóde by nemal byť problém pokračovať v mojej práci ďalej.

ZOZNAM POUŽITEJ LITERATÚRY

Zoznam použitej literatúry

[1] Brunovský, P.: *Stochastické modely operačnej analýzy*, učebné texty, dostupné na internete (23.4.2015): http://www.iam.fmph.uniba.sk/institute/kilianova/files/smoa_text_08_07.pdf

[2] Brunovský, P.: *Stochastické modely operačnej analýzy*, cvičenia k prednáškam, dostupné na internete (23.4.2015): http://www.iam.fmph.uniba.sk/institute/kilianova/files/smoa_exercises.pdf

[3] Janková, K. a kol.: *Markovove reťazce a ich aplikácie*, EPOS, 2015

[4] Little, J. D. C.: *A Proof for the Queuing Formula:* $L = \lambda W$, Operations Research, 1961

[5] Sztrik, J.: *Basic queuing theory*, učebné texty, dostupné na internete (23.4.2015): http://irh.inf.unideb.hu/ jsztrik/education/16/SOR_Main_Angol.pdf

Príloha A

Príloha A

Elektronická príloha Súbor obsahujúci zdrojový kód aplikácie