Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Отчет

по лабораторной работе №4 Статистическое моделирование случайных процессов и систем

по дисциплине «Введение в машинное обучение»

Выполнила
Студентка гр. 5130904/10101 Никифорова Е. А.

Руководитель Чуркин В. В.

Санкт-Петербург
2024

Оглавление

Цель работы

Рассматривается автоматизированная система, структурно-надежностная схема которой и ЛФРС известна. Система состоит из m различных по типу элементов, по каждому типу в схеме ni одинаковых элементов, и Li запасных частей (ЗЧ). Поток отказов элементов системы простейший, это означает, что время наработки до отказа подчиняется экспоненциальному закону с параметром λi. Система функционирует в режиме непрерывного длительного применения и в случае отказа, элемент заменяется на работоспособную запасную часть, если количество оставшихся ЗЧ больше нуля. Считаем, что замена происходит быстро и то время, за которое меняется элемент, не влияет на работоспособность системы.

Требуется определить, используя метод статистического моделирования, какое минимальное количество ЗЧ необходимо, чтобы вероятность безотказной работы (ВБР) системы за время Т была не менее Р0. ЛФРС определяется из структурно-надежностной схемы системы, которая также как и остальные исходные данные для решения задачи приведены ниже повариантно. Всего 10 вариантов заданий, номер варианта задания определяется номером студента в списке группы по модулю 10 плюс 1.

Вариант № 8.

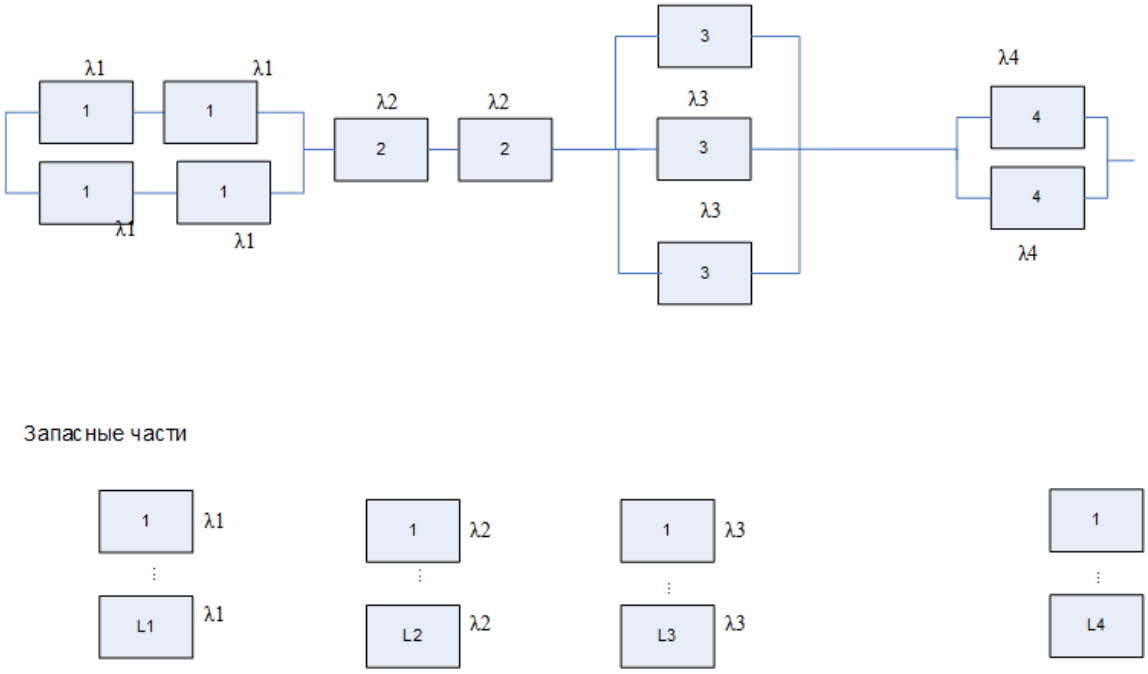

Рисунок 8. Структурно-надежностная схема системы.

$m = 4$; $\lambda_1 = 40 \cdot 10^{-6}, \lambda_2 = 10 \cdot 10^{-6}, \lambda_3 = 80 \cdot 10^{-6}, \lambda_4 = 30 \cdot 10^{-6}$ 1/ч; $P^0 = 0{,}995$, $T =$ 8760 ч.

Ход работы

Рассматривается автоматизированная система, структурно-надежностная схема которой и ЛФРС известна. Система состоит из m различных по типу элементов, по каждому типу в схеме ni одинаковых элементов, и Li запасных частей (ЗЧ). Поток отказов элементов системы простейший, это означает, что время наработки до отказа подчиняется экспоненциальному закону с параметром λi. Система функционирует в режиме непрерывного длительного применения и в случае отказа, элемент заменяется на работоспособную запасную часть, если количество оставшихся ЗЧ больше нуля. Считаем, что замена происходит быстро и то время, за которое меняется элемент, не влияет на работоспособность системы. Требуется определить, используя метод статистического моделирования, какое минимальное количество ЗЧ необходимо, чтобы вероятность безотказной работы (ВБР) системы за время Т была не менее Р0.

Обобщенный алгоритм вычисления ВБР системы $p\ (T)$ следующий:

Цикл от 1 до N, где N – количество итераций МСМ

Цикл по i от 1 до m

Генерирование ni СВ времени отказа каждого модуля в структуре с экспоненциальным законом распределения $t_l = -\ln(\alpha) / \lambda\ ,\ l = 1..n$, α – СВ с равномерным распределением в интервале [0,1].

Цикл от 0 до Li

Вычисление l, такого что $t_l = \min(t_1, ..., t_n)$.

Генерирование СВ времени отказа элемента, введенного на замену $t_l = t_l - \ln(\alpha) / \lambda$.

Конец цикла по i от 1 до m

Конец цикла от 1 до N

Если ЛФРС структуры равна 0 (вычисляется подстановкой в качестве булевых переменных в ЛФРС - $x_l = \begin{cases} 0, t_l \le T \\ 1, t_l > T \end{cases}$), то увеличение счетчика количества отказов d.

$p\ (T)$ = 1- d/N.

Конец алгоритма.

Пусть α = 0.995 ε = 0.001

Воспользуемся формулой для определения количества реализаций N, где

t_α = 2.576 – квантиль нормального распределения по уровню α.

$$N = {t_\alpha}^2 \frac{p(1-p)}{\varepsilon^2} = 2.576 * 2.576 * \frac{0.995 * (1 - 0.995)}{0{,}001^2} = 33013$$

Результаты

L = [4, 2, 4, 3], sum = 13, P = 0.9954866264804774

L = [4, 2, 4, 4], sum = 14, P = 0.9960015751370672

L = [4, 2, 5, 2], sum = 13, P = 0.9960924484294066

L = [4, 2, 5, 3], sum = 14, P = 0.9971526368400327

L = [4, 2, 5, 4], sum = 15, P = 0.9968497258655681

L = [4, 3, 4, 2], sum = 13, P = 0.9956986641626027

L = [4, 3, 4, 3], sum = 14, P = 0.996183321721746

L = [4, 3, 4, 4], sum = 15, P = 0.996183321721746

L = [4, 3, 5, 2], sum = 14, P = 0.9970617635476934

L = [4, 3, 5, 3], sum = 15, P = 0.9973343834247115

L = [4, 3, 5, 4], sum = 16, P = 0.9978796231787478

L = [4, 4, 4, 2], sum = 14, P = 0.9954866264804774

L = [4, 4, 4, 3], sum = 15, P = 0.9962439039166389

L = [4, 4, 4, 4], sum = 16, P = 0.9968194347681216

L = [4, 4, 5, 2], sum = 15, P = 0.9971829279374792

L = [4, 4, 5, 3], sum = 16, P = 0.9981219519583194

L = [4, 4, 5, 4], sum = 17, P = 0.9981825341532123

L = [5, 2, 4, 2], sum = 13, P = 0.9966376881834429

L = [5, 2, 4, 3], sum = 14, P = 0.9969405991579074

L = [5, 2, 4, 4], sum = 15, P = 0.9970920546451398

L = [5, 2, 5, 2], sum = 14, P = 0.9969405991579074

L = [5, 2, 5, 3], sum = 15, P = 0.9983642807378912

L = [5, 2, 5, 4], sum = 16, P = 0.9985763184200164

L = [5, 3, 4, 2], sum = 14, P = 0.9973343834247115

L = [5, 3, 4, 3], sum = 15, P = 0.9980310786659801

L = [5, 3, 4, 4], sum = 16, P = 0.997697876594069

L = [5, 3, 5, 2], sum = 15, P = 0.9981522430557659

L = [5, 3, 5, 3], sum = 16, P = 0.999273013661285

L = [5, 3, 5, 4], sum = 17, P = 0.9991215581740527

L = [5, 4, 4, 2], sum = 15, P = 0.9971223457425863

L = [5, 4, 4, 3], sum = 16, P = 0.9979704964710872

L = [5, 4, 4, 4], sum = 17, P = 0.9979704964710872

L = [5, 4, 5, 2], sum = 16, P = 0.9985157362251235

L = [5, 4, 5, 3], sum = 17, P = 0.9991821403689456

L = [5, 4, 5, 4], sum = 18, P = 0.9990306848817133

L = [6, 2, 4, 2], sum = 14, P = 0.9956077908702632

L = [6, 2, 4, 3], sum = 15, P = 0.9976070033017296

L = [6, 2, 4, 4], sum = 16, P = 0.997758458788962

L = [6, 2, 5, 2], sum = 15, P = 0.9975464211068367

L = [6, 2, 5, 3], sum = 16, P = 0.9985460273225699

L = [6, 2, 5, 4], sum = 17, P = 0.9988792293944809

L = [6, 3, 4, 2], sum = 15, P = 0.9973949656196044

L = [6, 3, 4, 3], sum = 16, P = 0.998485445127677

L = [6, 3, 4, 4], sum = 17, P = 0.9983339896404446

L = [6, 3, 5, 2], sum = 16, P = 0.9983642807378912

L = [6, 3, 5, 3], sum = 17, P = 0.9995153424408566

L = [6, 3, 5, 4], sum = 18, P = 0.999606215733196

L = [6, 4, 4, 2], sum = 16, P = 0.9968800169630145

L = [6, 4, 4, 3], sum = 17, P = 0.9984248629327841

L = [6, 4, 4, 4], sum = 18, P = 0.9982734074455517

L = [6, 4, 5, 2], sum = 17, P = 0.9979704964710872

L = [6, 4, 5, 3], sum = 18, P = 0.9994244691485172

L = [6, 4, 5, 4], sum = 19, P = 0.9995153424408566

Оптимальные результаты

L = [4, 2, 4, 3], sum = 13, P = 0.9954866264804774

L = [4, 2, 5, 2], sum = 13, P = 0.9960924484294066

L = [4, 3, 4, 2], sum = 13, P = 0.9956986641626027

L = [5, 2, 4, 2], sum = 13, P = 0.9966376881834429

Вывод

В ходе лабораторной работы мы выяснили, сколько минимально необходимо ЗЧ для обеспечения безотказной работы системы за 8760ч с вероятностью не менее 0,995.
Оптимальные результаты получаются при 13 элементах:
[4, 2, 4, 3], [4, 2, 5, 2], [4, 3, 4, 2], [5, 2, 4, 2]

Приложение

```
import numpy as np
import math


def logic_func(x):
    t_value = 8760
    return (((x[0] > t_value and x[1] > t_value) or (x[2] > t_value and x[3] > t_value))
            and x[4] > t_value and x[5] > t_value
            and (x[6] > t_value or x[7] > t_value or x[8] > t_value)
            and (x[9] > t_value or x[10] > t_value))


def find_P(L):
    N = 33013
    n = [4, 2, 3, 2]
    m = 4
    lambdas_values = [40 * 10 ** (-6), 10 * 10 ** (-6), 80 * 10 ** (-6), 30 * 10 ** (-
6)]
    d_counter = 0

    for k in range(N):
        x = []
        for i in range(m):
            t = []
            for j in range(n[i]):
                alfa = np.random.uniform(0, 1)
                t.append((-math.log(alfa) / lambdas_values[i]))
            for j in range(L[i]):
                l = t.index(min(t))
                t[l] -= math.log(np.random.uniform(0, 1)) / lambdas_values[i]
            for j in range(n[i]):
                x.append(t[j])
        d_counter += not logic_func(x)

    P = 1 - d_counter / N
    return P


if __name__ == '__main__':
    L = [0, 0, 0, 0]
    arr = [4, 2, 3, 2]
    P0 = 0.995
    for i in range(arr[0], arr[0] + 3):
        L[0] = i
        for j in range(arr[1], arr[1] + 3):
            L[1] = j
            for k in range(arr[2], arr[2] + 3):
                L[2] = k
                for l in range(arr[3], arr[3] + 3):
                    L[3] = l
                    P = find_P(L)
                    if P > P0:
                        print(f'L = {L}, sum = {sum(L)}, P = {P}\n')
```