КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра моделювання складних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня магістра
за спеціальністю 113 "Прикладна математика"

на тему:

ОПТИМАЛЬНЕ РЕЗЕРВУВАННЯ СИСТЕМ ТЕОРІЇ НАДІЙНОСТІ ПРИ КІЛЬКОХ ОБМЕЖЕННЯХ

студента 2-го курсу магістратури
Галушкіна Владислава Борисовича

_____

Науковий керівник:

професор, доктор фізико-математичних наук
Мацак Іван Каленикович

_____

Роботу заслухано на засіданні кафедри моделювання складних систем
та рекомендовано до захисту в ДЕК.

Протокол № 10 від 08 травня 2020 року

Завідувач кафедри МСС, канд. фіз.-мат. наук,

доц. Черній Д.І. _____
(підпис)

Київ – 2020

РЕФЕРАТ

Обсяг роботи 50 сторінок, 14 ілюстрацій, 9 таблиць, 9 джерел посилань, 2 додатки.

ГЕНЕРУВАННЯ ВИПАДКОВИХ ВЕЛИЧИН, ДИНАМІЧНЕ ПРОГРАМУВАННЯ, ДУБЛЬОВАНА СИСТЕМА, ЕКСПОНЕНЦІЙНИЙ РОЗПОДІЛ, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ІНТЕНСИВНІСТЬ ВІДМОВ, ІНТЕНСИВНІСТЬ ВІДНОВЛЕННЯ, МЕТОД МОНТЕ-КАРЛО, ОПТИМАЛЬНЕ РЕЗЕРВУВАННЯ, РЕЗЕРВОВАНА СИСТЕМА З ВІДНОВЛЕННЯМ, РЕЗЕРВУВАННЯ ПРИ КІЛЬКОХ ОБМЕЖЕННЯХ.

Об'єктом роботи є моделювання і пошук розв'язку задачі оптимального резервування при кількох обмеженнях за допомогою методу Монте-Карло та модифікації методу динамічного програмування. Предметом роботи є програмний засіб для розв'язання певних задач оптимального резервування.

Метою роботи є дослідження нової модифікації методу динамічного програмування для задач оптимального резервування з кількома обмеженнями та створення програмного засобу для розв'язку відповідних задач.

Методи дослідження і розроблення: розробка нового алгоритму динамічного програмування, імітаційне моделювання резервованих систем з відновленням та без, розробка кінцевого програмного продукту моделювання і розв'язку систем. Інструменти розроблення: комерційне, крос-платформне інтегроване середовище розробки веб-сторінок JetBrains PhpStorm 2018.3.3, мова програмування JavaScript, мова розмітки HTML.

Результати роботи: реалізовано і протестовано модифікацію методу динамічного програмування, створено веб-додаток для розв'язку певних задач оптимального резервування систем з кількома обмеженнями.

Створений програмний продукт може застосовуватися для розв'язку певних задач оптимального резервування систем з кількома обмеженнями; також початковий код програми може бути використано для подальшої модифікації розробленого алгоритму динамічного програмування.

ЗМІСТ

ВСТУП

Як було справедливо відмічено в роботі [1], у зв'язку з розвитком сучасної техніки особливу важливість придбали численні питання підвищення ефективності різного роду пристроїв. Комплексна автоматизація виробничих процесів ставить перед керуючими пристроями виключно відповідальні завдання, які повинні виконуватися бездоганно протягом усього періоду роботи автоматичної лінії, автоматизованого цеху або підприємства. Перерва в роботі керуючого пристрою може призвести не тільки до погіршення якості виробленої продукції або до повного припинення виробничого процесу, а й до дуже серйозних аварій, які виходять за локальні рамки підприємства. Питання підвищення надійності систем з моменту становлення такої науки, як теорії надійності, завжди було дуже важливим, і хоча в наш час ситуація в багатьох аспектах істотно змінилася, тим не менш зрозуміло, що в стратегічному плані питання надійності складних систем залишаються актуальними.

Одним з найефективніших методів підвищення надійності складних систем є резервування. І для того, щоб якомога раціональніше розподілити ресурси підвищення надійності між підсистемами, використовуються методи імітаційного моделювання, адже реальні системи схильні до випадкових впливів, і поведінка систем в цьому випадку найбільш ефективно досліджується саме при імітаційному моделюванні.

Однак, дуже часто на резерв можуть накладатися відразу декілька обмежень, будь то його ціна, вага, розміри, тощо. Це значно ускладнює процес пошуку розв'язку задачі. У науковій літературі дуже рідко розглядають подібні питання, а прикладів ефективних й достатньо точних алгоритмів у зв'язці з імітаційним моделюванням і зовсім нема. Саме тому дослідження в цій роботі оптимального резервування систем теорії надійності з кількома обмеженнями має цінність.

1 ОСНОВНІ ПОНЯТТЯ ТЕОРІЇ НАДІЙНОСТІ

В цьому розділі ми введемо основні поняття, які будуть зустрічатись протягом усієї роботи і без яких розуміння матеріалу неможливе.

1.1 Основні показники надійності

Надійність елемента (системи) – це його здатність в заданих умовах і на протязі заданого часу виконувати свої функції. Під елементом будемо розуміти будь-який пристрій, надійність якого вивчається незалежно від надійності його складових частин. Тоді система – це пристрій, надійність якого оцінюється за його структурою та надійністю його частин.

Нехай в момент $t = 0$ елемент починає роботу, а в момент $t = \tau$ відбувається його відмова. Тоді кажуть, що τ – час життя елемента (або час безвідмовної роботи). В теорії надійності припускається, що τ – випадкова величина, $F(t) = P\{\tau < t\}$ – її функція розподілу, $f(t) = F'(t)$– щільність розподілу. Однак слід зауважити, що щоб на практиці це припущення виконувалося, потрібно, щоб елемент був представником великої однорідної групи виробів.

Одна з основних характеристик теорії надійності – це ймовірність безвідмовної роботи елемента за час t:

$$P(t) = 1 - F(t) = P\{\tau > t\}.$$

Функцію $P(t)$ також називають *функцією надійності*.

При цьому, під функцією $F(t)$ слід розуміти *ймовірність відмови елемента (системи)*.

Звернемося до роботи [2], щоб розглянути наступні поняття. *Інтенсивність відмов* – це характеристика надійності $\lambda(t)$.

Ймовірнісне визначення цієї характеристики можна записати як

$$\lambda(t) = \frac{1}{1 - F(t)} \frac{d}{dt} F(t) = \frac{f(t)}{P(t)},$$

тобто $\lambda(t)$ – умовна щільність ймовірності відмови об'єкта до моменту часу t при умові, що до цього моменту відмова елементу не відбулася.

Перед тим, як ввести статистичне визначення, введемо наступні допоміжні позначення:

$n(t)$ – число відмовивших об'єктів на момент часу t;

$N(t)$ – число працездатних об'єктів на момент часу t;

$\Delta n(t, t')$ – число об'єктів, що відмовили в інтервалі часу $[t, t']$.

Отже, статистичне визначення виглядатиме наступним чином:

$$\bar{\lambda}(t) = \frac{n(t + \Delta t) - n(t)}{N(t) \cdot \Delta t} = \frac{N(t + \Delta t) - N(t)}{N(t) \cdot \Delta t} = \frac{\Delta n(t, t + \Delta t)}{N(t) \cdot \Delta t},$$

тобто $\bar{\lambda}(t)$ – відношення кількості відмов на проміжку часу $[t, t + \Delta t]$ до добутку кількості справних об'єктів у момент часу t на тривалість проміжку часу Δt.

Задля опису такого поняття, як *інтенсивність відновлення*, ми введемо такі позначення:

$t = \xi$ – момент часу відновлення об'єкта;

$G(t) = P\{\xi < t\}$ – розподіл часу відновлення;

$g(t) = G'(t)$ – щільність розподілу;

$n_B(t)$ – число об'єктів, відновлення котрих тривало менше за час t;

$N_B(t)$ – число об'єктів, відновлення котрих тривало більше за час t;

$\Delta n_B(t, t')$ – число об'єктів, відновлення котрих тривало більше за час t, але менше, ніж за час t'.

Отже, ймовірнісне визначення інтенсивності відновлення виглядатиме як

$$\mu(t) = \frac{g(t)}{1 - G(t)},$$

тобто $\mu(t)$ – умовна щільність ймовірності відновлення об'єкта в момент часу t, який починається від моменту початку відновлення, за умов, що до моменту часу t відновлення об'єкта не відбулося.

Статистичне визначення інтенсивності відновлення:

$$\bar{\mu}(t) = \frac{n_B(t+\Delta t) - n_B(t)}{N_B(t) \cdot \Delta t} = \frac{N_B(t+\Delta t) - N_B(t)}{N_B(t) \cdot \Delta t} = \frac{\Delta n_B(t, t+\Delta t)}{N_B(t) \cdot \Delta t},$$

тобто $\bar{\mu}(t)$ – відношення кількості відновлень на проміжку часу $[t, t+\Delta t]$ до добутку кількості об'єктів, ще не відновлених на момент часу t, на тривалість проміжку часу Δt.

1.2 Експоненційний закон розподілу теорії надійності

Експоненційний розподіл задається функцією розподілу

$$F(t) = 1 - e^{-\lambda t},$$

$t \geq 0, \lambda > 0,$

щільність розподілу –

$$f(t) = \frac{d}{dt}\left[1 - e^{-\lambda t}\right] = \lambda e^{-\lambda t},$$

$t \geq 0.$

Середній час безвідмовної роботи легко знаходиться шляхом прямого інтегрування

$$M_1 = \int_0^\infty t\lambda e^{-\lambda t} dt = \frac{1}{\lambda} = a.$$

Другий початковий момент знаходиться як

$$M_2 = \int_0^{\infty} t^2 \lambda e^{-\lambda t} dt = \frac{2}{\lambda^2}$$

Отримаємо дисперсію експоненційного розподілу

$$\sigma^2 = M_2 - M_1^2 = \frac{2}{\lambda^2} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2}.$$

Ймовірність безвідмовної роботи визначається наступним чином

$$P(t) = e^{-\lambda t} = e^{\frac{t}{a}},$$

$t \geq 0$.

У своїй роботі [3] Ушаков І. О. відмічає, що починаючи з 50-х років минулого століття, експоненційний закон розподілу займає абсолютно унікальне місце в задачах теорії надійності. Більшість задач теорії надійності для випадку експоненційного розподілу значно простіші, ніж для довільного розподілу. Основна причина цього полягає в тому, що експоненційний розподіл має таку важливу властивість: умовна ймовірність безвідмовної роботи на інтервалі $(t, t+s)$, $P(t, t\ +\ s)$, не залежить від t, а залежить тільки від s (його майбутнє не залежить від минулого).

Дійсно

$$P(t, t\ +\ s) = \frac{P(t+s)}{P(t)} = \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} = e^{-\lambda s}.$$

Інакше кажучи, якщо деякий об'єкт характеризується експоненційним часом роботи до відмови, то об'єкт, який пропрацював довільний час, але не відмовив до даного моменту, за своїми характеристиками надійності буде не відрізнятися від абсолютно нового об'єкта.

2 ФОРМУЛЮВАННЯ ПРОБЛЕМИ ОПТИМАЛЬНОГО РЕЗЕРВУВАННЯ

Як вже відмічалося у вступі, одним з найефективніших методів підвищення надійності складних систем є методи оптимізації, зокрема оптимальне резервування. Мета методу оптимального резервування – якнайбільш раціональніше розподіляти ресурси підвищення надійності між підсистемами. Звернемося до праць Ушакова І. О. [3] і [4], щоб сформулювати наступну проблему оптимального резервування.

Як об'єкт дослідження будемо розглядати послідовне з'єднання n-кількості незалежних ділянок резервування, кожна з яких має свої (незалежні від інших ділянок) резервні елементи підвищення надійності в кількості m_1 , m_2 ,..., m_n одиниць (рисунок 2.1). Для стислості, ділянки резервування будемо називати вже використаним попередньо терміном *підсистемами*. Досліджувати будемо *обернену задачу* оптимального резервування (про це – трохи далі), а в самій задачі використовуватиметься *ненавантажене резервування* (холодний резерв), тобто коли на робочому місці знаходиться елемент і в разі його відмови він миттєво замінюється на новий, справний елемент. Звичайно, слід розуміти, що подібна схема миттєвої заміни є ідеалізованою, бо практичне підключення нового елемента вимагає певної підготовки, розігріву.

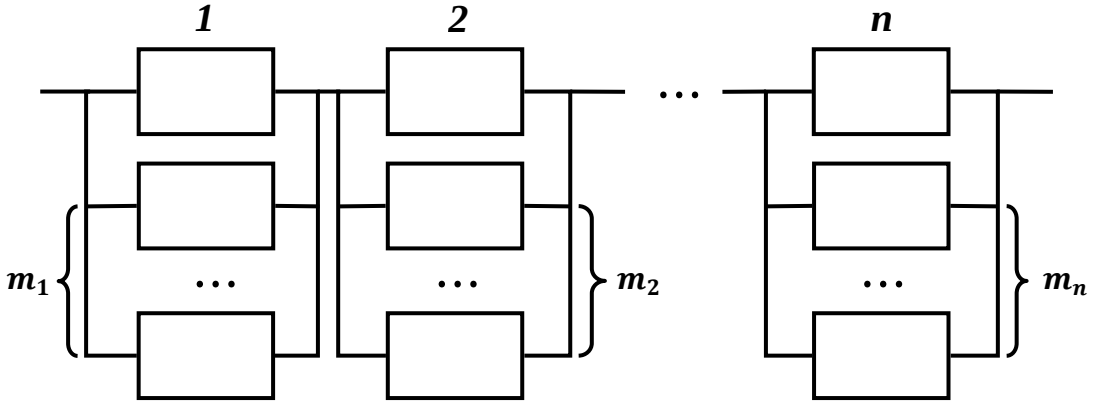


Рисунок 2.1 – Схема послідовного з'єднання

Можливі дві основні постановки задачі оптимального резервування.

(а) *Пряма задача*.

Шляхом резервування досягти необхідного значення показника надійності (або не перевищити певне допустиме значення ймовірності відмови) системи при мінімально можливих витратах ресурсів на резервні елементи.

(б) *Обернена задача*.

Шляхом резервування домогтися максимально можливого значення обраного показника надійності (або ж мінімального значення ймовірності відмови) системи при заданих обмеженнях, пов'язаних із введенням резервних елементів.

Позначимо надійність i-ї підсистеми як $P_i(m_i)$, де m_i – загальна кількість резервних елементів i-ї резервної групи. Функція $P_i(m_i)$ залежить від того, що вдає із себе підсистема, яка ступінь навантаженості резерву та від інших факторів. Так як ми розглядаємо послідовну систему, то надійність усієї системи буде дорівнювати добутку значень надійності усіх підсистем, тобто

$$P_{\text{Н}}(\overline{m}) = \prod_{i=1}^{n} P_i(m_i).$$

Нехай також відоме D - група з k-кількості функцій-обмежень резервних елементів за кількома параметрами: ціна, вага, об'єм, будь-що інше. Зазвичай вважається, що ці функції в усіх прикладних задачах лінійно залежить від кількості елементів:

$$D = \begin{cases} \sum_{i=1}^{n} c_{1i} m_i \\ \cdots \\ \sum_{i=1}^{n} c_{ki} m_i \end{cases},$$

де c_{ji} – вартість одного резервного елемента i-ї резервної групи по j-му обмеженню, причому $j = \overline{1,k}$.

Формально задача (а) може бути записана наступним чином:

$$\min_{1\le j\le k}\left\{\sum_{i=1}^{n} c_{ji}m_i \;\middle|\; \prod_{i=1}^{n} P_i(m_i) \ge P_{\text{вим}}\right\},$$

де $P_{\text{вим}}$ – допустиме значення надійності системи.

Задача (б) – обернена – формулюється у вигляді

$$P_{\text{Н}}(\overline{m}) = \prod_{i=1}^{n} P_i(m_i) \to \max_{D},$$

$$D = \begin{cases} \sum_{i=1}^{n} c_{1i}m_i \le C_1^{max} \\ \cdots \\ \sum_{i=1}^{n} c_{ki}m_i \le C_k^{max} \end{cases},$$

де C_j^{max} – максимально допустиме (задане) значення по j-му обмеженню, причому $c_{ji},\ C_j^{max} \in \mathbb{N},\quad j = \overline{1,k}.$

3 ПОШУК РОЗВ'ЯЗКУ ЗАДАЧІ ОПТИМАЛЬНОГО РЕЗЕРВУВАННЯ

Відтепер, розібравшись у загальній теорії в двох попередніх розділах, ми нарешті можемо перейти до основної теми цієї роботи.

У своїй книзі [4] Ушаков І. О. описує декілька способів розв'язку задач оптимального резервування з кількома обмеженнями, але усі наведені методи - евристичні, тобто вони не дають гарантію точності знайденого розв'язку. Ще один метод можна знайти в роботі [5] Р. Барлоу і Ф. Прошан, і цей метод можна назвати доволі точним та ефективним, але через його надмірну заплутаність програмну реалізацію цього алгоритму потрібно розглядати в окремій роботі. Отже, не маючі жодних варіантів, лишається тільки вивести новий алгоритм, за основу якого взято метод динамічного програмування. Але перш ніж перейти до нового алгоритму, треба пояснити, що з себе уявляє тільки що згаданий метод, і навести приклад розв'язку більш простої задачі.

3.1 Метод динамічного програмування

Сучасні обчислювальні машини дозволяють обраховувати дані з дуже великою швидкістю, що дозволяє не замислюватися над оптимізацією швидкодії програмного забезпечення. Але якщо наявна дуже велика кількість даних, а час на їхню обробку обмежений? Звернемося до праці Р. Беллмана і С. Дрейфуса [6], щоб дослідити це питання.

Для початку розглянемо просту ситуацію, коли кожна з підсистем нашої задачі може отримувати десять різних значень ймовірностей. Тоді процес максимізації надійності з N-кількістю підсистем приводить до 10^N різних множин вибору. Але в цьому процесі загальна кількість можливостей значно менша, так як вибір значення в одній з підсистем відразу обмежує можливий вибір в інших підсистемах.

Для випадку з десятьма підсистемами ми повинні обстежити 10^{10} випадків. Таке число може здатись не таким великим, але, як то мовиться, все

пізнається в порівнянні. Наприклад, якщо кожну мілісекунду обстежувалась би одна сотня множин з усіх можливих випадків, тоді загалом знадобилося б 10^5 секунд, або 27 годин, 46 хвилин і 40 секунд. Більше однієї доби – це занадто велика кількість часу, отже є сенс у використанні інших методів.

Але чому все ж таки метод динамічного програмування дає можливість швидко розв'язувати завдання з великою кількістю підсистем? Ключем є принцип оптимальності. Згідно з цим принципом, обравши деяку початкову підсистему N, ми потім відмовляємося від обстеження всіх варіантів, що включають цей приватний вибір, а розглядаємо тільки ті варіанти, які є оптимальними для інших $(N-1)$-підсистем. Іншими словами, сутність методу динамічного програмування полягає в покроковій процедурі відшукання умовних оптимальних рішень, яка приводить на останньому кроці до знаходження абсолютного оптимуму.

3.2 Приклад задачі оптимального резервування з одним обмеженням

Розглянемо приклад резервованої системи, яка складається з трьох послідовно з'єднаних підсистем з резервними елементами та з відновлювального (ремонтного) блоку, який містить в собі певну кількість відновлювальних (ремонтних) одиниць (рисунок 3.1).

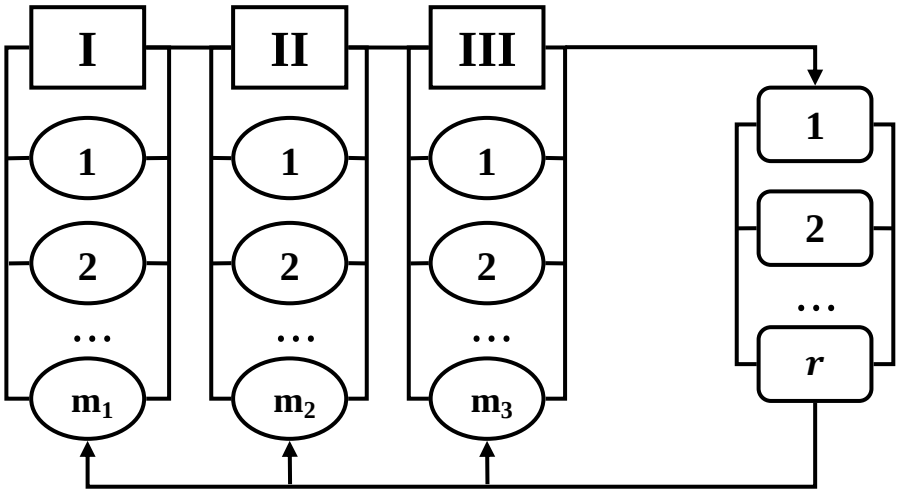


Рисунок 3.1 – Резервована система з відновленням

Отже, кожна з трьох підсистем містить у собі по одному основному елементу та деякої кількості резерву до цих основних елементів (m_1, m_2, m_3 штук резервних елементів для першої, другої та третьої підсистеми відповідно). Також в системі є ремонтний блок з ремонтними одиницями у кількості r штук. Коли один з основних елементів виходить з ладу, він прямує до ремонтного блоку і займає одну з ремонтних одиниць, у той же час на місце зламаного стає резервний елемент. Коли елемент відновився (відремонтувався), він повертається до своєї підсистеми і стає резервним елементом. Система в нас послідовна, тож якщо в деякий момент часу один з основних елементів виходить з ладу, а резервного для нього вже нема, тоді уся система виходить з ладу.

Задача полягає у наступному: треба знаходити таку оптимальну кількість резервних елементів, щоб значення надійності системи на деякому відрізку часу $[0, T]$ було максимальним. Це все враховуючи обмеження на витрати на резервні елементи.

Формально задача виглядає наступним чином:

$$P_{\text{Н}}(\bar{m}) = P(\text{система не відмовить до моменту часу } T) = \\ = P_1(m_1) \cdot P_2(m_2) \cdot P_3(m_3) \to max,$$

$$c_1 m_1 + c_2 m_2 + c_3 m_3 \le C_{\text{доп}},$$

де $P_i(m_i)$ – значення надійності i-підсистеми, $i = \overline{1,3}$.

Час безвідмовної роботи елемента τ, як і час ремонту ξ, підпорядкований експоненційному розподілу теорії надійності:

$$\tau \to F(t) = 1 - e^{-\lambda_i t}, \;\; t \ge 0, \;\; i = \overline{1,3},$$
$$\xi \to G(t) = 1 - e^{-\mu t}, \;\; t \ge 0,$$

де λ_i – інтенсивність відмови елементу i-типу;

μ – інтенсивність відновлення елементу.

Для того, щоб полегшити дослідження цієї задачі, будемо вважати, що кількість ремонтних одиниць більше або дорівнює загальній кількості резервних елементів, тобто

$$m_1 + m_2 + m_3 \leq r.$$

Ця умова не тільки позбавляє систему від такого стану, як *черга на ремонт*, а ще й дозволяє розв'язувати задачу оптимізації покроково, знаходячи спочатку значення надійності першої, потім другої, а потім вже третьої підсистеми. Для пошуку значень надійності підсистем будемо застосовувати імітаційне моделювання методом Монте-Карло. Що з себе уявляє цей метод, чому був обран саме цей і як його застосувати до нашої задачі – опишемо вже в одному з наступних розділів. Важливо відмітити, що моделювання буде проходити шляхом прогонки по алгоритму систем зі зростаючою кількістю резервних елементів. Тобто, спочатку по алгоритму буде проганятися n-кількість разів система без резервних елементів ($m_i = 0, i = \overline{1,3}$). Далі буде проведено тестування системи з однією резервною і однією відновлювальною одиницею ($m_i = r = 1$). Потім – система з $m_i = r = 2$. І так далі, поки не дійдемо до максимально можливого значення m_i, обмеженого ціною елемента i-типу і допустимим обмеженням на сумарну вартість резервних елементів. Тобто, обчислення будуть проводитись для усіх $m_i \leq {}^{C_{\text{доп}}}/_{c_i}$, $i = \overline{1,3}$.

Отримавши результати прогонки по алгоритму імітаційного моделювання у вигляді пар значень *кількість резервних елементів в підсистемі – надійність підсистеми*, ми будемо далі шукати оптимальну кількість резервних елементів методом динамічного програмування.

Формально метод динамічного програмування для цієї задачі вигляду

$$L(\overline{m}) = \sum_{i=1}^{n} \varphi_i(m_i) \to \max_D ,$$

$$D = \begin{cases} \sum_{i=1}^{n} c_i m_i \le C_{\text{доп}} \\ m_i \in \mathbb{Z}_+, \quad i = \overline{1,n} \\ c_i \in \mathbb{N}, \quad i = \overline{1,n} \\ C_{\text{доп}} \in \mathbb{N} \end{cases}$$

можна записати як

$$L_i(s) = L_i\big(m_i(s)\big) = \max_{m_i \le \frac{s}{c_i}} \big(\varphi_i(m_i) + L_{i+1}(s - c_i m_i)\big),$$

де $m_i(s)$ – точка екстремуму, $s = \overline{0, C_{\text{доп}}}$, $s \in \mathbb{Z}$.

В багатьох джерелах, де описан метод динамічного програмування, наприклад, в роботі [3], задача розв'язується з початку, тобто припускається, що $L_1(s) = \varphi_1(s)$, потім знаходиться $L_2(s)$, $L_3(s)$,..., $L_{n-1}(s)$, поки не знайдемо $L_n\big(C_{\text{доп}}\big)$. Але ми зробимо навпаки, будемо починати рух по алгоритму з кінця:

$$\boxed{L_n(s) = \varphi_n(s)} \to \boxed{L_{n-1}(s)} \to \cdots \to \boxed{L_2(s)} \to \boxed{L_1^*\big(C_{\text{доп}}\big)},$$

причому $L_1^*\big(C_{\text{доп}}\big) = \max_{\overline{m} \in D} L(\overline{m})$.

Отже для того, щоб можна було скористуватися методом динамічного програмування, треба рівняння максимізації звести до суми. Для цього прологарифмуємо обидві частини рівняння:

$$\ln\big(P_{\text{Н}}(\overline{m})\big) = \ln\big(P_1(m_1) \cdot P_2(m_2) \cdot P_3(m_3)\big) = \\ = \ln\big(P_1(m_1)\big) + \ln\big(P_2(m_2)\big) + \ln\big(P_3(m_3)\big) \to max.$$

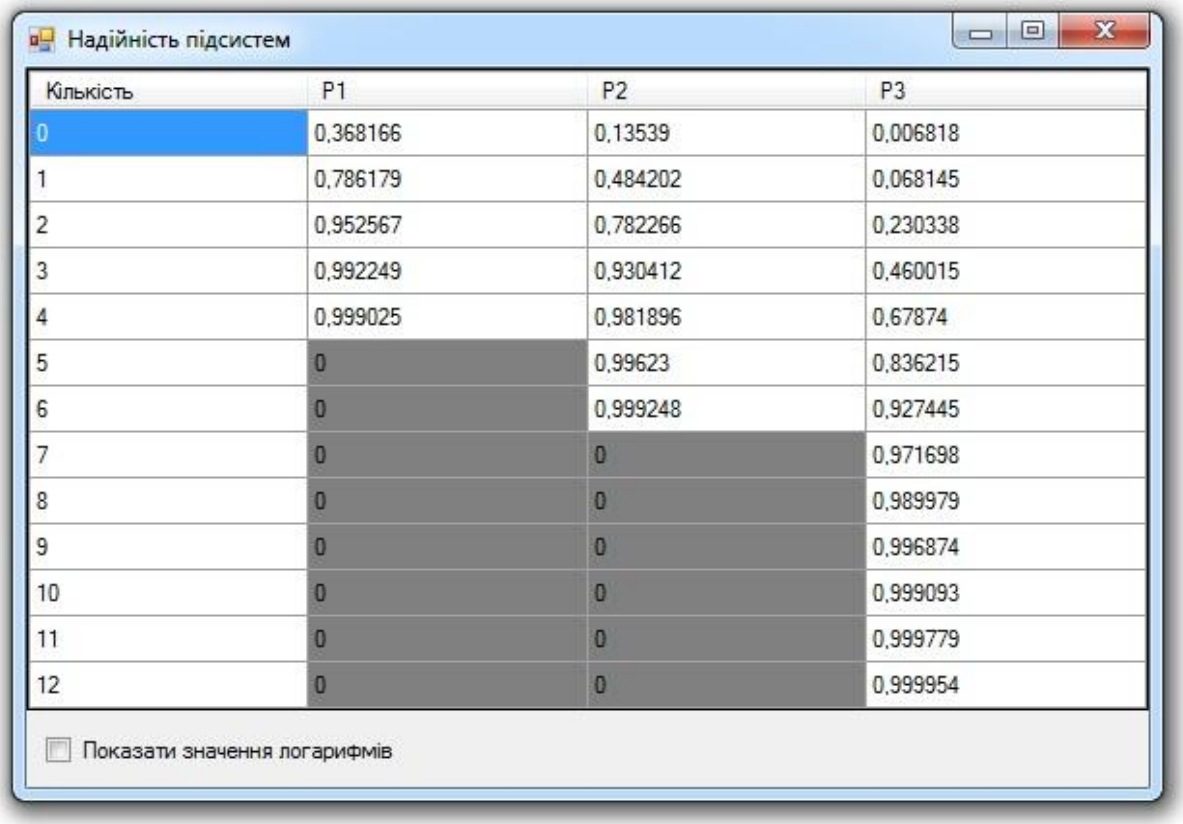
Перевизначивши логарифми як $\ln\big(P_{\text{Н}}(\overline{m})\big) = L(\overline{m})$ і $\ln\big(P_i(m_i)\big) = \varphi_i(m_i)$, отримаємо

$$L(\overline{m}) = \varphi_1(m_1) + \varphi_2(m_2) + \varphi_3(m_3) \to max.$$

Для вхідних даних

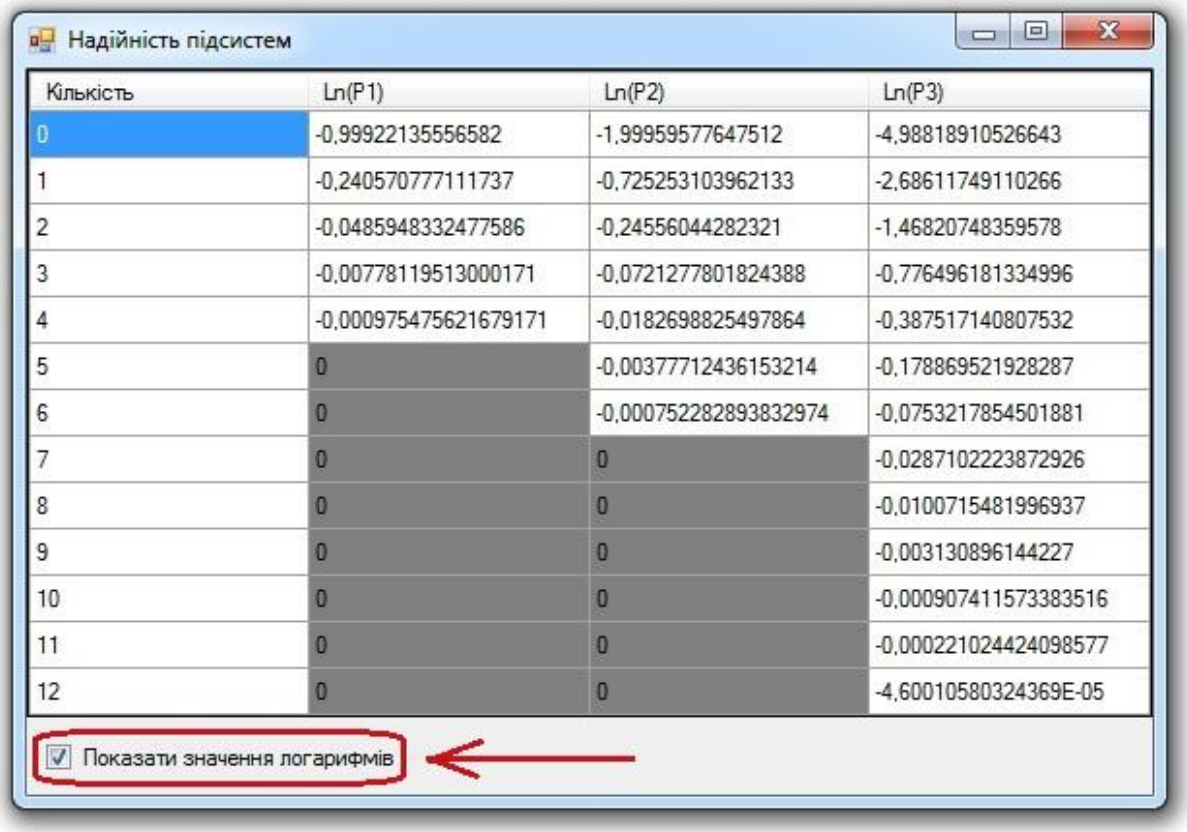
$$\lambda_1 = 1,\ \lambda_2 = 2,\ \lambda_3 = 5, \mu = 1, T = 1,$$
$$c_1 = 3,\ c_2 = 2,\ c_3 = 1,\ C_{\text{доп}} = 12,$$

за допомогою створеного раніше прототипу програми імітаційного моделювання для задач з трьома підсистемами, отримаємо наступні результати:



Кількість	P1	P2	P3
0	0,368166	0,13539	0,006818
1	0,786179	0,484202	0,068145
2	0,952567	0,782266	0,230338
3	0,992249	0,930412	0,460015
4	0,999025	0,981896	0,67874
5	0	0,99623	0,836215
6	0	0,999248	0,927445
7	0	0	0,971698
8	0	0	0,989979
9	0	0	0,996874
10	0	0	0,999093
11	0	0	0,999779
12	0	0	0,999954

Рисунок 3.2 – Отримані значення надійності підсистем



Кількість	Ln(P1)	Ln(P2)	Ln(P3)
0	-0,99922135556582	-1,99959577647512	-4,98818910526643
1	-0,240570777111737	-0,725253103962133	-2,68611749110266
2	-0,0485948332477586	-0,24556044282321	-1,46820748359578
3	-0,00778119513000171	-0,0721277801824388	-0,776496181334996
4	-0,000975475621679171	-0,0182698825497864	-0,387517140807532
5	0	-0,00377712436153214	-0,178869521928287
6	0	-0,000752282893832974	-0,0753217854501881
7	0	0	-0,0287102223872926
8	0	0	-0,0100715481996937
9	0	0	-0,003130896144227
10	0	0	-0,000907411573383516
11	0	0	-0,000221024424098577
12	0	0	-4,60010580324369E-05

Рисунок 3.3 – Логарифми від значень надійності підсистем

Таблицею з рисунка 3.3 ми будемо надалі користуватись для розв'язку даної задачі оптимального резервування. За процесом розв'язку наведеним методом динамічного програмування можна прослідкувати у додатку А.

Розв'язавши задачу динамічного програмування, ми отримали наступні результати:

$m_1 = 1$ – кількість резервних елементів першого типу,

$m_2 = 2$ – кількість резервних елементів другого типу,

$m_3 = 5$ – кількість резервних елементів третього типу,

$L(\overline{m}) = \ln\left(P_{\text{H}}(\overline{m})\right) = -0.664999.$

Отримане значення надійності системи:

$$P_{\text{H}}(\overline{m}) = e^{\ln\left(P_{\text{H}}(\overline{m})\right)} = 0.514274.$$

4 МЕТОД ДИНАМІЧНОГО ПРОГРАМУВАННЯ ДЛЯ ЗАДАЧ З КІЛЬКОМА ОБМЕЖЕННЯМИ

Розглянувши в попередньому розділі приклад розв'язку задачі з одним обмеженням і маючи уявлення про метод динамічного програмування в цілому, можемо перейти до опису модифікації методу динамічного програмування для задач оптимального резервування з кількома обмеженнями.

4.1 Алгоритм

Формально задача для k-кількості обмежень виглядає наступним чином:

$$L(\overline{m}) = \sum_{i=1}^{n} \varphi_i(m_i) \to \max_{D},$$

$$D = \begin{cases} \sum_{i=1}^{n} c_{ji} m_i \le C_j^{max} \\ m_i \in \mathbb{Z}_+ \\ c_{ji} \in \mathbb{N} \\ C_j^{max} \in \mathbb{N} \end{cases},$$

де $i = \overline{1,n}, \quad j = \overline{1,k}$.

Крок n:

Як і в прикладі з одним обмеженням, рух по алгоритму починаємо з кінця. Припускаємо, що

$$L_n^*(s_1, \dots, s_k) = \varphi_n\left(m_n^*(s_1, \dots, s_k)\right),$$
$$m_n^*(s_1, \dots, s_k) = \left[\min\left(\frac{s_1}{c_{1n}}, \dots, \frac{s_k}{c_{kn}}\right)\right],$$

причому шукаються ці значення для усіх можливих комбінацій

$$s_1 = 0, 1, 2, \dots, C_1^{max},$$

$$\dots$$

$$s_k = 0, 1, 2, \dots, C_k^{max}.$$

Далі будується таблиця співвідношень *комбінація* $s_1, \dots, s_k$ *– значення* L_n^* *та* m_n^* для цих комбінацій:

$\boldsymbol{s_1}$	$\boldsymbol{s_2}$	$\cdots$	$\boldsymbol{s_k}$	$\boldsymbol{m_n^*(s_1, s_2, \dots, s_k)}$	$\boldsymbol{L_n^*(s_1, s_2, \dots, s_k)}$
0	0	$\cdots$	0	$m_n^*(0, 0, \dots, 0)$	$L_n^*(0, 0, \dots, 0)$
1	0	$\cdots$	0	$m_n^*(1, 0, \dots, 0)$	$L_n^*(1, 0, \dots, 0)$
0	1	$\cdots$	0	$m_n^*(0, 1, \dots, 0)$	$L_n^*(0, 1, \dots, 0)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
0	0	$\cdots$	1	$m_n^*(0, 0, \dots, 1)$	$L_n^*(0, 0, \dots, 1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
C_1^{max}	C_2^{max}	$\cdots$	C_k^{max}	$m_n^*(C_1^{max}, C_2^{max}, \dots, C_k^{max})$	$L_n^*(C_1^{max}, C_2^{max}, \dots, C_k^{max})$

Ця таблиця знадобиться на наступному кроці і при пошуку розв'язку.

Крок $n-1$:

$$L_{n-1}^*(s_1, \dots, s_k) = \max_{D_{n-1}} \left(\varphi_{n-1}(m_{n-1}) + L_n^*(s_1 - c_{1n-1}m_{n-1}, \dots, s_k - c_{kn-1}m_{n-1})\right),$$

$$D_{n-1}: \ 0 \le m_{n-1} \le \min\left(\frac{s_1}{c_{1n-1}}, \dots, \frac{s_k}{c_{kn-1}}\right),$$

$$m_{n-1}^*(s_1, \dots, s_k) = m_{n-1},$$

де m_{n-1} таке, при якому значення $L_{n-1}^*(s_1, \dots, s_k)$ досягло свого максимуму.

Так само, як і на попередньому кроці, будується нова таблиця, вже для кроку $n-1$:

$\boldsymbol{s_1}$	$\boldsymbol{s_2}$	$\cdots$	$\boldsymbol{s_k}$	$\boldsymbol{m_{n-1}^*(s_1, \dots, s_k)}$	$\boldsymbol{L_{n-1}^*(s_1, \dots, s_k)}$
…	…	…	…	…	…

Крок i $(i = n-1 \dots 2)$:

$$L_i^*(s_1, \dots, s_k) = \max_{D_i}\left(\varphi_i(m_i) + L_{i+1}^*(s_1 - c_{1i}m_i, \dots, s_k - c_{ki}m_i)\right),$$

$$D_i:\ 0 \le m_i \le \min\left(\frac{s_1}{c_{1i}}, \dots, \frac{s_k}{c_{ki}}\right),$$

$$m_i^*(s_1, \dots, s_k) = m_i\ .$$

Крок 1:

На відміну від усіх попередніх кроків, на цьому таблицю будувати не треба, адже на першому кроці

$$(s_1,\ s_2, \dots,\ s_k) = \ (C_1^{max},\ C_2^{max}, \dots,\ C_k^{max}).$$

Тому

$$L_1^*(C_1^{max}, \dots,\ C_k^{max})$$

$$= \max_{D_1}\left(\varphi_1(m_1) + L_2^*(C_1^{max} - c_{11}m_1, \dots,\ C_k^{max} - c_{k1}m_1)\right),$$

$$D_1:\ 0 \le m_1 \le \min\left(\frac{C_1^{max}}{c_{11}}, \dots, \frac{C_k^{max}}{c_{k1}}\right),$$

$$m_1^*(C_1^{max}, \dots, C_k^{max}) = m_1.$$

Розв'язок:

Вектор оптимальних значень $\bar{m} = (m_1^*,\ m_2^*, \dots,\ m_n^*)$ шукається так:

$m_1^* = m_1^*(C_1^{max}, \dots, C_k^{max})$,

$m_2^* = m_2^*(C_1^{max} - c_{11}m_1^*, \dots, C_k^{max} - c_{k1}m_1^*)$,

…

$m_n^* = m_n^*\left(C_1^{max} - \sum_{i=1}^{n-1} c_{1i}m_i^*\ , \dots, C_k^{max} - \sum_{i=1}^{n-1} c_{ki}m_i^*\right)$.

Як раз тут нам і знадобились побудовані попередньо таблиці ще раз.

Знайшовши

$$L(\bar{m}) = \sum_{i=1}^{n} \varphi_i(m_i^*)\ ,$$

можна визначити і значення надійності системи за оптимальною кількістю резервних елементів.

4.2 Перевірка алгоритму

Для перевірки створеного алгоритму звернемося по роботи Барлоу Р. і Прошан Ф. [5], де наведено приклад розв'язання задачі оптимального резервування з двома обмеженнями.

Система характеризується наступними показниками:

Номер підсистеми i	*1*	*2*	*3*	*4*	*Обмеження*
Вартість c_{1i}	12	23	34	45	470
Вага c_{2i}	1	1	1	1	20
Ненадійність q_i	0.20	0.30	0.25	0.15	

Обмеження задані у вигляді

$$\begin{cases} 12m_1 + 23m_2 + 34m_3 + 45m_4 \le 470 \\ m_1 + m_2 + m_3 + m_4 \le 20 \end{cases},$$

а надійність системи визначається так:

$$P(\bar{m}) = \prod_{i=1}^{4}(1 - {q_i}^{m_i}) = (1 - 0.2^{m_1})(1 - 0.3^{m_2})(1 - 0.25^{m_3})(1 - 0.15^{m_4}).$$

Розв'язком цієї задачі, за наведеним у книзі алгоритмом, стає вектор $\bar{m} = (5, 6, 4, 3)$. Надійність системи при цьому дорівнює

$$P(5, 6, 4, 3) = (1 - 0.2^5)(1 - 0.3^6)(1 - 0.25^4)(1 - 0.15^3) = 0.9917.$$

Отже, такі результати ми повинні отримати, прогнавши цю задачу по нашому алгоритму. Але перед цим потрібно визначити усі відсутні початкові дані, а саме значення логарифмів надійності кожної з підсистем для різної кількості резервних елементів. Для цього скористуємося наведеною вище формулою надійності системи.

Таблиця 4.1 – Початкові дані

m	$\varphi_1(m)$	$\varphi_2(m)$	$\varphi_3(m)$	$\varphi_4(m)$
0	-∞	-∞	-∞	-∞
1	-0.223143551	-0.356674944	-0.287682072	-0.162518929
2	-0.040821995	-0.094310679	-0.064538521	-0.022756987
3	-0.008032172	-0.027371197	-0.015748357	-0.003380708
4	-0.001601281	-0.008132983	-0.003913899	-0.000506378
5	-0.000320051	-0.002432957	-0.000977040	-0.000075940
6	-0.000064002	-0.000729266	-0.000244170	-0.000011391
7	-0.000012800	-0.000218724	-0.000061037	-0.000001709
8	-0.000002560	-0.000065612	-0.000015259	$-2.563 \cdot 10^{-7}$
9	$-5.12 \cdot 10^{-7}$	-0.000019683	-0.000003815	$-3.844 \cdot 10^{-8}$
10	$-1.024 \cdot 10^{-7}$	-0.000005905	$-9.537 \cdot 10^{-7}$	$-5.767 \cdot 10^{-9}$
11	$-2.048 \cdot 10^{-8}$	-0.000001771	$-2.384 \cdot 10^{-7}$	0
12	$-4.096 \cdot 10^{-9}$	$-5.314 \cdot 10^{-7}$	$-5.96 \cdot 10^{-9}$	0
13	$-8.192 \cdot 10^{-10}$	$-1.594 \cdot 10^{-7}$	$-1.49 \cdot 10^{-9}$	0
14	$-1.638 \cdot 10^{-10}$	$-4.783 \cdot 10^{-8}$	0	0
15	$-3.277 \cdot 10^{-11}$	$-1.435 \cdot 10^{-8}$	0	0
16	$-6.55 \cdot 10^{-12}$	$-4.305 \cdot 10^{-9}$	0	0
17	$-1.31 \cdot 10^{-12}$	$-1.291 \cdot 10^{-9}$	0	0
18	$-2.6 \cdot 10^{-13}$	$-3.874 \cdot 10^{-10}$	0	0
19	$-5 \cdot 10^{-14}$	$-1.162 \cdot 10^{-10}$	0	0
20	$-1 \cdot 10^{-14}$	$-3.487 \cdot 10^{-11}$	0	0

Підставивши усі необхідні дані у програму, отримаємо результат (рисунок 4.2), повністю співпадаючий із розв'язком цієї задачі з використанням іншого алгоритму. А це свідчить про те, що створений алгоритм виявився вдалим.

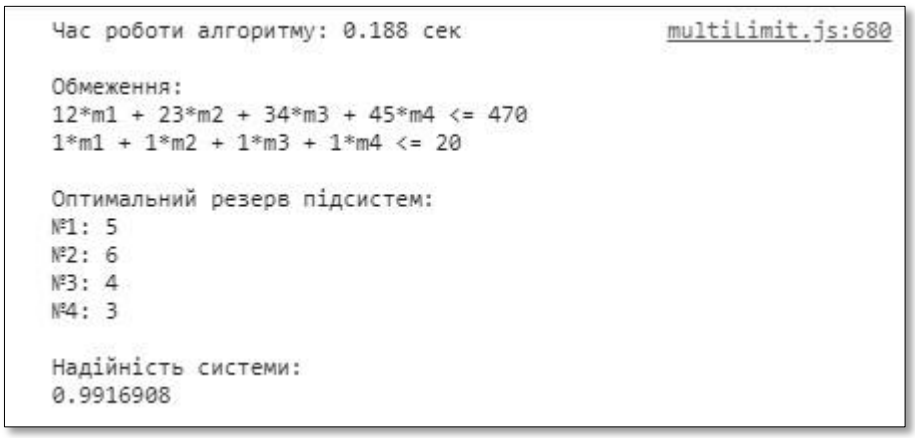


Рисунок 4.2 – Результати розв'язку задачі

4.3 Тестування швидкодії алгоритму

Слід відразу зазначити, що алгоритм було написано на інтерпретованій, не найшвидшій мові програмування, і загалом можна досягти більш кращих результатів, тому основна роль цього підрозділу – показати, як і від чого залежить час виконання розрахунків алгоритмом загалом.

Отже перше, від чого залежить час виконання - це кількість підсистем у системі. Більше підсистем - більше кроків потрібно пройти алгоритму, а найоб'ємніші розрахунки виконуються на кроках $n-1 \dots 2$. Другий і третій фактор, що впливають на час – це кількість обмежень і їх коефіцієнти C_j^{max}. Чим більше ці коефіцієнти, тим більше комбінацій розподілення ресурсів може бути утворено і тим більше розрахунків доведеться зробити.

В таблиці 4.3 наведено результати тестування декількох задач з різними вхідними параметрами. Значення коефіцієнту C_j^{max} у рамках одного випадку для кожного обмеження бралось однакове. Заміри проводились у двох браузерах з різними інтерпретаторами мови JavaScript – Mozilla v75 та Chrome v80. Результати виконання розрахунків записані в останньому стовпчику, для Mozilla зліва і Chrome справа. В деяких випадках визначити час виявилось неможливим – виконання програми переривалось браузером через нестачу пам'яті, або не завершалось протягом години.

Таблиця 4.3 – Залежність часу виконання розрахунків від параметрів

Кількість підсистем	Кількість обмежень	Значення коефіцієнту C_j^{max}	Час виконання розрахунків (секунд)
4	2	40	0.023 / 0.045
7	2	40	0.035 / 0.115
10	2	40	0.046 / 0.129
4	3	40	0.353 / 0.644
7	3	40	0.711 / 1.175
10	3	40	1.052 / 1.726
4	4	40	14.797 / 20.522
7	4	40	31.633 / 42.903
10	4	40	48.809 / 66.993
4	2	100	0.077 / 0.184
7	2	100	0.146 / 0.274
10	2	100	0.204 / 0.358
4	3	100	5.245 / 7.834
7	3	100	12.165 / 16.521
10	3	100	18.343 / 26.474
4	4	100	– / –
7	4	100	– / –
10	4	100	– / –
4	2	200	0.193 / 0.419
7	2	200	0.388 / 0.765
10	2	200	0.566 / 1.114
4	3	200	32.708 / 63.366
7	3	200	72.093 / 143.820
10	3	200	161.789 / –
4	2	1000	3.691 / 7.533
7	2	1000	8.294 / 16.056
10	2	1000	13.762 / 24.351

5 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ЗАДАЧІ

5.1 Сутність імітаційного моделювання. Метод Монте-Карло

Імітаційне моделювання полягає в тому, що процес функціонування складної системи представляється у вигляді певного алгоритму, тобто логічних дій, які і реалізуються на комп'ютері. За результатами реалізації можуть бути зроблені ті чи інші висновки щодо вихідного процесу.

Як зазначено в роботі В. Кельтона і А. Лоу [7], якщо співвідношення, які утворюють модель, досить прості для отримання точної інформації щодо цікавих для нас питань, то можна використовувати математичні методи. Такого роду рішення називається *аналітичним*. Однак більшість існуючих систем є дуже складними, і для них неможливо створити реальну модель, описану аналітично. Такі моделі слід досліджувати за допомогою моделювання. При моделюванні комп'ютер використовується для чисельної оцінки моделі, а за допомогою отриманих даних розраховуються її реальні характеристики.

Моделювання може, наприклад, використовуватися при розгляді виробничою компанією можливості побудови великих додаткових приміщень для одного з її заводів, якщо керівництво компанії не впевнене, що потенційне зростання продуктивності зможе виправдати витрати на будівництво. Неможливо спорудити приміщення, а потім прибрати їх в разі нерентабельності, в той час як моделювання роботи заводу в його поточному стані і з *нібито* створеними додатковими приміщеннями допомагає у вирішенні цієї проблеми.

Метод імітаційного моделювання Монте-Карло отримав свою назву від міста Монте-Карло, Монако, відомого своїми гральними будинками. Успіх в таких іграх як рулетка, кістки і слот-машини цілком залежить від удачі, а результатами ходів є розподіл випадкових величин. Моделювання за методом Монте-Карло визначається як процедура, у якій використовуються випадкові числа, тобто випадкові величини $U(0,1)$.

5.2 Алгоритм імітаційного моделювання резервованої системи з відновленням

Як було відмічено у третьому розділі, кількість ремонтних одиниць, яка більша або дорівнює кількості резервних елементів, гарантує, що зламані елементи однієї з підсистем будуть відновлюватися незалежно від елементів інших підсистем. Це і дозволяє моделювати підсистеми окремо (рисунок 5.1), що значно легше з точки зору реалізації, ніж моделювання усієї системи одночасно. Крім того, такий спосіб спрощує й перевірку вірності складення алгоритму та його програмної реалізації.

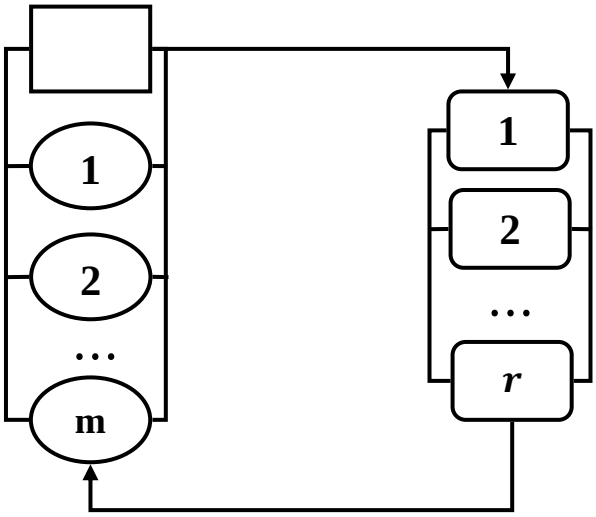


Рисунок 5.1 – Модель окремої підсистеми

Перейдемо тепер до безпосереднього моделювання алгоритму для отриманої на рисунку 5.1 системи. Слід нагадати, що розглядається випадок холодного резервування, в системі один основний елемент, m -кількість резервних і r-кількість ремонтних одиниць, причому $r = m$.

Перш за все відмітимо, що система складається з трьох блоків: блоку основного елементу (позначимо цю множину A_1), блоку резервних елементів (A_2) і ремонтного блоку (відповідно, A_3), який складається з m-ремонтних одиниць.

Основний елемент:

$$A_1 = \{z_1, t_1\},$$

де $z_1 = \overline{0,1}$ – кількість елементів в множині A_1;

t_1 – залишковий час роботи основного елемента.

Резервні елементи:

$$A_2 = \{z_2\},$$

де $z_2 = \overline{0,m}$ – кількість резервних елементів в множині A_2.

Ремонтний блок:

$$A_3 = \{z_{3.1}, z_{3.2}, \dots, z_{3.m}, t_{3.1}, t_{3.2}, \dots, t_{3.m}\},$$

перепозначимо як $A_{3.i} = \{z_{3.i}, t_{3.i}\}, \ \ i = \overline{1,m},$

де $z_{3.i} = \overline{0,1}$ – наявність елементу в i-ремонтній одиниці (тобто, якщо $z_{3.i} = 1$, то ремонтна одиниця зайнята елементом, що відновлюється, якщо $z_{3.i} = 0$ – тоді ремонтна одиниця вільна);

$t_{3.i}$ – залишковий час відновлення елемента в i-ремонтній одиниці.

Існує два типи подій, коли в системі буде щось змінюватись – це коли відмовляє основний елемент (позначимо цю подію як E_1) і коли відновлюється один з елементів (відповідно, E_3). Коли відбувається одна з цих подій, вона позначається k-кроком моделювання системи. Момент часу, коли ця подія виникає, позначимо як T_k. Час до наступного кроку (або *довжину кроку*) позначимо через Θ – тобто, $T_{k+1} = T_k + \Theta$.

Довжина кроку рахується наступним чином:

$$\Theta = \min\{t_1^k, t_{3.i}^k\}, \ \ i = \overline{1,m}$$

Інакше кажучи, час до $(k+1)$-кроку – це мінімум серед усіх значень залишкового часу роботи або відновлення елементів на k- кроці моделювання системи.

Подія E_1 відбувається, коли $\Theta = t_1^k < t_{3.i}^k$, тобто коли значення залишкового часу роботи основного елемента виявилось мінімальним. Коли мінімальним виявляється одне зі значень залишкового часу відновлення елементу (тобто, $\Theta = t_{3.i}^k < t_1^k$), тоді виникає подія E_3.

Нехай на k-кроці моделювання системи, в момент часу T_k, система має такий стан:

$$A_1^k = \{z_1^k, t_1^k\}$$
$$A_2^k = \{z_2^k\}$$
$$A_{3.i}^k = \{z_{3.i}^k, t_{3.i}^k\}$$

Тоді далі, на $(k+1)$-кроці, залежно від типу події, відбуваються наступні зміни в системі.

<u>Подія E_1</u>:

$$A_1^{k+1} = \{z_1^{k+1}, t_1^{k+1}\} = \begin{cases} (1, \tau_k), & z_2^k \geq 1 \\ (0, M), & z_2^k = 0 \end{cases}, \quad (M = \infty),$$

тобто, якщо на попередньому кроці в множині A_2^k є хоча б один резервний елемент, тоді зламаний елемент замінюється новим основним, для якого обчислюється час його роботи τ_k; якщо ж резервного елементу нема, тоді вся система припиняє своє функціонування (тобто, вона виходить з ладу);

$$A_2^{k+1} = \{z_2^{k+1}\} = \{z_2^k - 1\},$$

тобто, кількість елементів в резервному блоці зменшується на один;

$$A_{3.i}^{k+1} = \{z_{3.i}^{k+1}, t_{3.i}^{k+1}\} = \begin{cases} (z_{3.i}^k, t_{3.i}^k - \Theta), z_{3.i}^k = 1 \\ (1, \xi_k), \qquad\quad z_{3.i}^k = 0 \end{cases}, \quad i = \overline{1, m},$$

тобто, якщо i-ремонтна одиниця на попередньому кроці була зайнята, тоді для неї просто перераховується залишковий час ремонту, а перша ж незайнята ремонтна одиниця займається зламаним елементом і для неї шукається час відновлення елемента ξ_k.

Подія E_3:

$$A_1^{k+1} = \{z_1^{k+1}, t_1^{k+1}\} = \{z_1^k, t_1^k - \Theta\},$$

тобто, просто перераховується залишковий час роботи елемента;

$$A_2^{k+1} = \{z_2^{k+1}\} = \{z_2^k + 1\},$$

тобто, кількість резервних одиниць збільшується на одну;

$$A_{3.i}^{k+1} = \{z_{3.i}^{k+1}, t_{3.i}^{k+1}\} = \begin{cases} (z_{3.i}^k, t_{3.i}^k - \Theta), & t_{3.i}^k - \Theta > 0 \\ (0, M), & t_{3.i}^k - \Theta = 0 \end{cases} \quad (M = \infty),$$

тобто, для тієї ремонтної одиниці, що звільнилася, встановлюється час відновлення рівний нескінченності, а для інших ремонтних одиниць просто перераховується залишковий час відновлення елемента.

Отже, так виглядає алгоритм імітаційного моделювання. Але як знаходиться час роботи основного елемента τ_k і час відновлення ξ_k?

При виникненні в процесі моделювання будь-яких випадкових факторів слід звернутися до вибірки, або генерування, випадкових величин з розподілів ймовірностей. Для нашої задачі розподіл вже відомий – це експоненційний розподіл. Залишається лише зрозуміти, як будуть генеруватись величини τ_k і ξ_k.

Існує безліч методів генерування випадкових величин, і один з них, який описан в роботі [7], – метод *зворотного перетворення*. Припустимо, нам необхідно генерувати випадкову величину X, яка є неперервною і має функцію розподілу F – неперервну і строго зростаючу, коли $0 < F(t) < 1$. Візьмемо, що

F^{-1} – це зворотна функція F. Тоді алгоритм для генерування випадкової величини X, яка має функцію розподілу F, буде наступним:

1. Генеруємо $U \sim \mathrm{U}(0,1)$.

2. Повертаємо $X = F^{-1}(U)$.

На прикладі експоненційного розподілу $F(t) = 1 - e^{-\lambda t}$, задаємо $U = F(t)$, щоб знайти F^{-1}:

$$F^{-1}(U) = -\frac{1}{\lambda} ln(1 - U).$$

Слід зазначити, що в цьому випадку можна замінити $1 - U$ на U заради й незначного, але виграшу в швидкодії, адже $1 - U$ і U мають один і той самий розподіл $\mathrm{U}(0,1)$.

Таким чином ми отримуємо формули для генерування часу роботи і відновлення елемента:

$$\tau_k = -\frac{1}{\lambda} ln(u'),$$
$$\xi_k = -\frac{1}{\mu} ln(u''),$$

де u', u'' – випадкові величини $\mathrm{U}(0,1)$. Важливо зазначити, що u' і u'' повинні генеруватись незалежно один від одного, різними функціями.

Перевіряти коректність генерування випадкових величин $\mathrm{U}(0,1)$ немає сенсу, адже при моделюванні систем будуть використані вже готові функції, інтегровані в бібліотеку мови програмування і перевірені часом й багатьма користувачами. А от перевірка реалізації самого алгоритму імітаційного моделювання ще й як потрібна, чому будуть присвячені наступні підрозділі.

5.3 Перевірка алгоритму на прикладі дубльованої системи ($m = 1$)

Об'єктом нашої перевірки стане дубльована система, тобто система з одним резервним елементом ($m = 1$) та однією відновлювальною одиницею ($r = 1$), а результати, отримані моделюванням, будемо порівнювати з аналітичним розв'язком *дубльованої системи* з відновленням (рисунок 5.2) для ненавантаженого резерву, яка описана в роботі Гнеденка Б. і Коваленка І. [8].

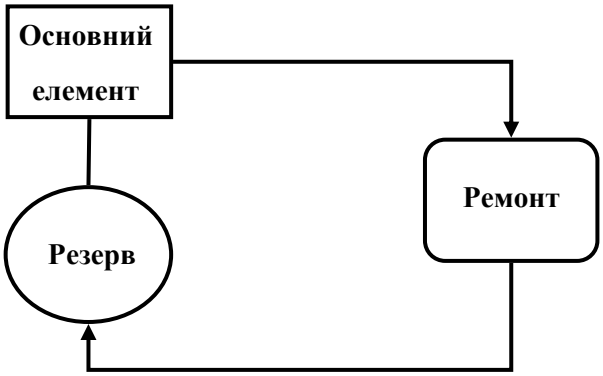


Рисунок 5.2 – Дубльована система з відновленням

Дубльована система з відновленням у випадку ненавантаженого резерву – це один з прикладів процесу загибелі та народження. Уявімо собі, що система в кожен момент часу може знаходиться в одному зі станів E_0, E_1, E_2,..., множина яких злічена. Стани системи з часом змінюються, причому за проміжок часу довжини h система зі стану E_n у момент часу t з ймовірністю $\lambda_n h + o(h)$ переходить у стан E_{n+1} і з ймовірністю $\mu_n h + o(h)$ – у стан E_{n-1}.

Позначимо через $P_k(t)$ ймовірність того, що система в момент часу t знаходиться у стані E_k. Тоді система буде описуватись наступною системою диференційних рівнянь:

$$P_0'(t) = -\lambda_0 P_0(t) + \mu_1 P_1(t) \tag{5.1}$$

і при $k \geq 1$

$$P_k'(t) = -(\lambda_k + \mu_k)P_k(t) + \lambda_{k-1}P_{k-1}(t) + \mu_{k+1}P_{k+1}(t). \tag{5.2}$$

Ми можемо зазначити три стани нашої дубльованої системи: E_0, E_1 і E_2 – тобто, коли в системі 0, 1 або 2 елементи, що відмовили. Тоді ймовірності переходів з одного стану в інший за проміжок часу h виглядатимуть наступним чином:

$$P\{E_0(t) \to E_1(t+h)\} = \lambda_0 h + o(h),$$
$$P\{E_1(t) \to E_0(t+h)\} = \mu_1 h + o(h),$$
$$P\{E_1(t) \to E_2(t+h)\} = \lambda_1 h + o(h),$$
$$P\{E_2(t) \to E_1(t+h)\} = \mu_2 h + o(h).$$

В даному процесі загибелі та розмноження $\lambda_0 = \lambda_1 = \lambda$, $\mu_1 = \mu$ і $\mu_2 = 0$. Тоді рівняння (5.1) – (5.2) для цієї задачі прийматимуть такий вигляд:

$$P_0'(t) = -\lambda P_0(t) + \mu P_1(t), \tag{5.3}$$
$$P_1'(t) = -(\lambda + \mu) P_1(t) + \lambda P_0(t), \tag{5.4}$$
$$P_2'(t) = \lambda P_1(t). \tag{5.5}$$

Початкові умови задачі: $P_0(0) = 1$, $P_1(0) = 0$, $P_2(0) = 0$.

Розв'язавши систему диференціальних рівнянь (5.3) – (5.5), врешті-решт отримаємо формулу для знаходження ймовірності безвідмовної роботи дубльованої системи, яка має вигляд

$$P(t) = P_0(t) + P_1(t) =$$
$$= e^{-\left(\lambda+\frac{\mu}{2}\right)t}\left[\cosh\frac{t}{2}\sqrt{4\lambda\mu+\mu^2} + \frac{2\lambda+\mu}{\sqrt{4\lambda\mu+\mu^2}}\sinh\frac{t}{2}\sqrt{4\lambda\mu+\mu^2}\right]. \tag{5.6}$$

Тепер, отримавши аналітичний результат у вигляді формули (5.6), можемо перейти до перевірки алгоритму метода Монте-Карло. У таблиці 5.3 в перших трьох колонках представлені значення інтенсивності відмов λ, інтенсивності відновлень μ і часу T, для яких ми шукаємо значення ймовірності безвідмовної роботи. В наступних двох колонках – значення ймовірностей, отримані аналітичною формулою та запрограмованим алгоритмом метода Монте-Карло. В останній колонці запишемо абсолютну похибку, різницю між

отриманими двома різними способами значеннями ймовірностей безвідмовної роботи. Слід зазначити, що для кожної трійки значень λ, μ і T, задля отримання якнайбільш точних результатів, система симулюється алгоритмом імітаційного моделювання $n = 10^6$ разів.

Таблиця 5.3 – Порівняння отриманих значень ймовірностей

			Результати, отримані за допомогою		Абсолютна похибка
λ	μ	T	аналітичної формули	метода Монте-Карло	
1	5	5	0.492641	0.491890	0.000751
1	5	0.1	0.995991	0.995989	0.000002
1	5	0.01	0.999951	0.999957	0.000006
1	10	3	0.782944	0.782786	0.000158
2	10	3	0.425763	0.426093	0.000330
5	10	3	0.019356	0.019383	0.000027
10	1	0.2	0.423524	0.423999	0.000475
10	3	0.2	0.455654	0.455042	0.000612
10	6	0.2	0.497663	0.497584	0.000079

Як можемо бачити, абсолютна похибка, тобто різниця значень, отриманих формулою (5.6) та методом Монте-Карло, не перевищує значення $\varepsilon = 10^{-3}$, навіть при наближенні значень ймовірностей безвідмовної роботи до 0.5, що є непоганим показником. Але проведення тестів лише на одній системі недостатньо, адже є ймовірність того, що алгоритм коректно працює лише для дубльованої системи. Тож є сенс провести ще декілька перевірок, але вже на прикладі більш складних систем.

5.4 Перевірка алгоритму на прикладі систем з $m = 2$ та $m = 3$

Перейдемо тепер до моделювання й тестування систем з двома ($m = 2$) та трьома ($m = 3$) резервними елементами (рисунок 5.4).

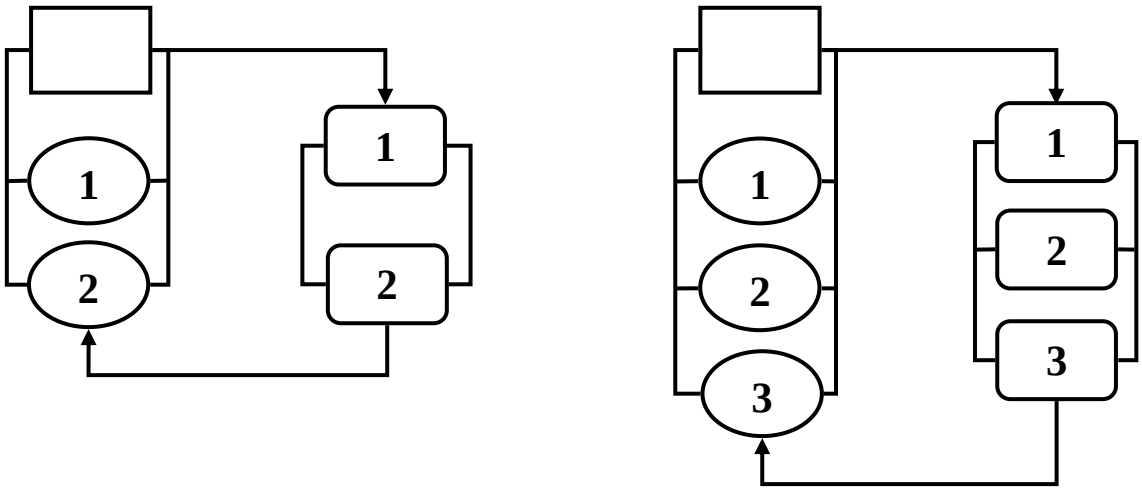


Рисунок 5.4 – Системи з $m = 2$ (зліва) та з $m = 3$ (справа)

Ми можемо зазначити чотири стани системи $m = 2$, коли відмовило нуль, один, два або три елементи (стани E_0, E_1, E_2, E_3), та п'ять станів системи $m = 3$, коли відмовило від нуля до чотирьох елементів (E_0, E_1, E_2, E_3, E_4). Нагадаємо, як виглядають ймовірності переходів з одного стану в інший:

$$P\{E_k(t) \to E_{k+1}(t+h)\} = \lambda_k h + o(h),$$
$$P\{E_{k+1}(t) \to E_k(t+h)\} = \mu_{k+1} h + o(h).$$

В системі з двома резервними елементами, $\lambda_0 = \lambda_1 = \lambda_2 = \lambda$, $\mu_1 = \mu$, $\mu_2 = 2\mu$ і $\mu_3 = 0$; усі інші λ_k і μ_k дорівнюють нулю. З (5.1) – (5.2) отримуємо таку систему диференційних рівнянь

$$P_0'(t) = -\lambda P_0(t) + \mu P_1(t), \tag{5.7}$$
$$P_1'(t) = -(\lambda + \mu) P_1(t) + \lambda P_0(t) + 2\mu P_2(t), \tag{5.8}$$
$$P_2'(t) = -(\lambda + 2\mu) P_2(t) + \lambda P_1(t), \tag{5.9}$$
$$P_3'(t) = \lambda P_2(t). \tag{5.10}$$

Для системи з трьома резервними елементами, $\lambda_0 = \lambda_1 = \lambda_2 = \lambda_3 = \lambda$, $\mu_1 = \mu$, $\mu_2 = 2\mu$, $\mu_3 = 3\mu$ і $\mu_4 = 0$; усі інші λ_k і μ_k так само дорівнюють нулю. З (5.1) – (5.2) для $m = 3$ виходить така система диференційних рівнянь

$$P_0'(t) = -\lambda P_0(t) + \mu P_1(t), \tag{5.11}$$

$$P_1'(t) = -(\lambda + \mu)P_1(t) + \lambda P_0(t) + 2\mu P_2(t), \tag{5.12}$$

$$P_2'(t) = -(\lambda + 2\mu)P_2(t) + \lambda P_1(t) + 3\mu P_3(t), \tag{5.13}$$

$$P_3'(t) = -(\lambda + 3\mu)P_3(t) + \lambda P_2(t), \tag{5.14}$$

$$P_4'(t) = \lambda P_3(t). \tag{5.15}$$

Початкові умови для задачі (5.7) – (5.10) наступні: $P_0(0) = 1$, $P_1(0) = 0$, $P_2(0) = 0$, $P_3(0) = 0$; для задачі (5.11) – (5.15) такі: $P_0(0) = 1$, $P_1(0) = 0$, $P_2(0) = 0$, $P_3(0) = 0$, $P_4(0) = 0$.

Вивести загальні формули знаходження ймовірності безвідмовної роботи з отриманих систем диференційних рівнянь майже неможливо, але сучасне програмне забезпечення дозволяє розв'язувати складні системи і, залежно від параметрів, отримувати значення функцій. Звернемося за допомогою до програмного пакету Maple. В додатку Б можна прослідкувати, як розв'язуються системи диференційних рівнянь і знаходяться ймовірності безвідмовної роботи систем.

Перейдемо тепер до перевірки алгоритму метода Монте-Карло. У таблицях 5.5 та 5.6 можна побачити результати, отримані після розв'язку систем диференційних рівнянь у Maple та отримані методом імітаційного моделювання, для систем з $m = 2$ та $m = 3$ відповідно. Як і до цього, система симулюється алгоритмом імітаційного моделювання $n = 10^6$ разів.

Таблиця 5.5 – Порівняння отриманих значень (система з $m = 2$)

			Результати, отримані за допомогою		Абсолютна похибка
λ	μ	T	Maple	метода Монте-Карло	
1	10	3	0.987800	0.987865	0.000065
2	10	3	0.918827	0.918782	0.000045
5	10	3	0.415553	0.415725	0.000172
10	1	0.2	0.702357	0.701513	0.000844
10	3	0.2	0.746336	0.746305	0.000031
10	6	0.2	0.797704	0.797545	0.000159

Таблиця 5.6 – Порівняння отриманих значень (система з $m = 3$)

			Результати, отримані за допомогою		Абсолютна похибка
λ	μ	T	Maple	метода Монте-Карло	
1	10	3	0.999589	0.999589	0.000000
2	10	3	0.994269	0.994216	0.000053
5	10	3	0.861254	0.861051	0.000203
10	1	0.2	0.877015	0.877047	0.000032
10	3	0.2	0.907969	0.907986	0.000017
10	6	0.2	0.938928	0.939324	0.000396

Проаналізувавши усі отримані результати, можна з упевненістю сказати, що запрограмований алгоритм метода Монте-Карло працює як треба. Абсолютна похибка значень ймовірностей для систем з $m = 2$ і $m = 3$, як і в випадку дубльованої системи, не перевищує значення $\varepsilon = 10^{-3}$.

6 ОГЛЯД РЕАЛІЗОВАНОЇ ПРОГРАМИ

Мовою JavaScript було запрограмовано алгоритм імітаційного моделювання для розглянутої задачі оптимального резервування, а також нову модифікацію методу динамічного програмування пошуку розв'язку цієї задачі. Все це було поєднано в один веб-додаток, доступний за посиланням [9].

Розглянемо інтерфейс програми (рисунок 6.1), що за певних вхідних даних шукає розв'язок задачі оптимального резервування.

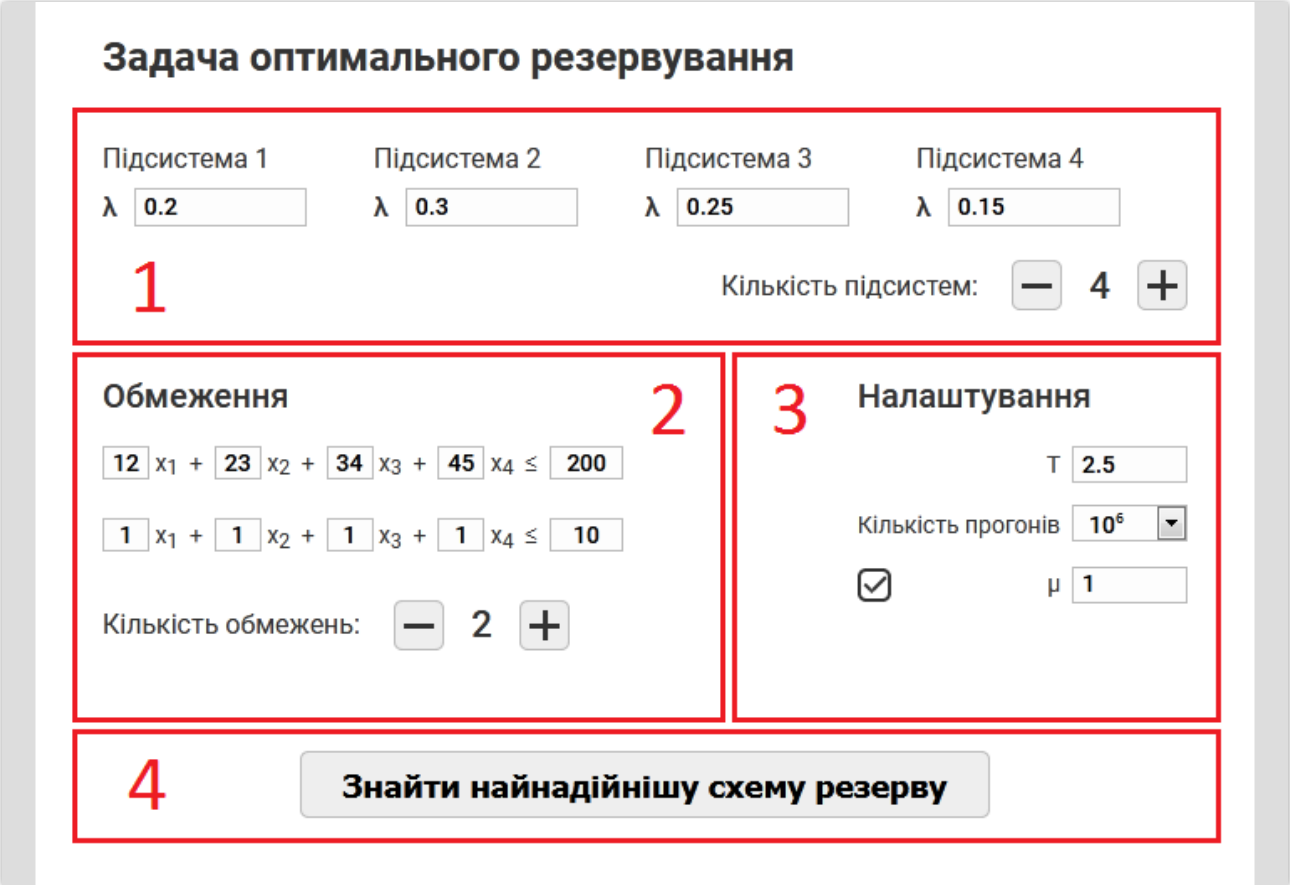


Рисунок 6.1 – Інтерфейс програми

В першому блоці управління задаються значення інтенсивності відмов λ і кількість підсистем. Користувач може обрати від однієї до п'ятнадцяти підсистем, а коефіцієнтом інтенсивності відмов обрати будь-яке невід'ємне число. В другому блоці користувач обирає кількість обмежень і їх вагові й обмежуючі коефіцієнти. У якості коефіцієнтів можливо обрати тільки ціле невід'ємне число, а кількість обмежень – від одного до чотирьох. В третьому

блоці вводяться такі дані: момент часу T, до якого буде проводитися симуляція системи, кількість ітерацій ("прогонів") симуляції алгоритму імітаційного моделювання та інтенсивність відновлення μ. Коефіцієнтами часу й відновлення можна обрати будь-яке невід'ємне число, а для кількості прогонів доступні лише дві опції: 10^5, для більш швидкого моделювання системи, і 10^6, для більш точних значень надійності підсистем. Також користувач може зняти галочку навпроти коефіцієнту μ – це прибере з системи ремонтний блок, тобто задача буде розв'язуватися без можливості відновлення елементів. Натискаючи на кнопку "4", користувач запускає програму. Процес моделювання задачі і пошук її розв'язку може займати досить тривалий час, тому користувачу виводиться відповідне повідомлення (рисунок 6.2).

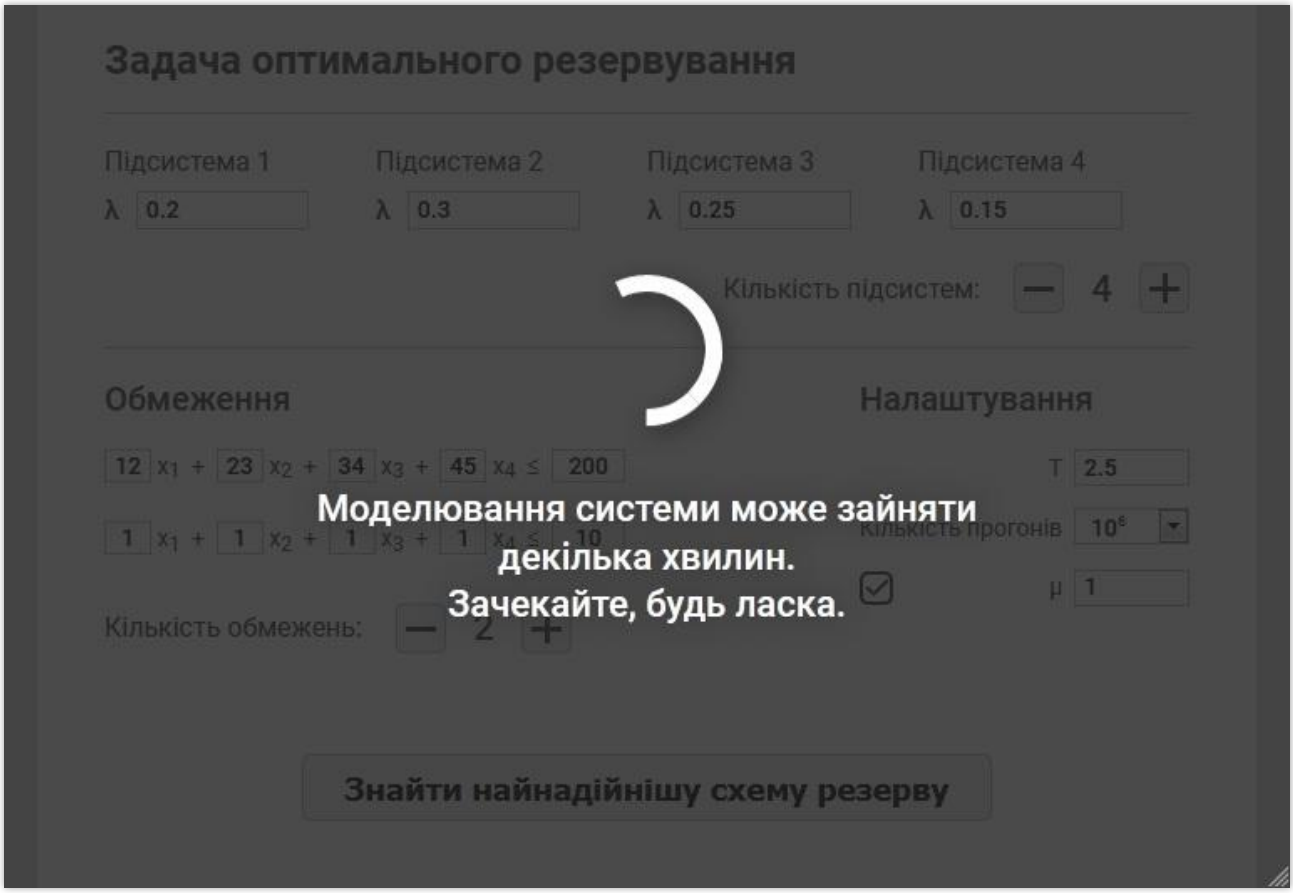


Рисунок 6.2 – Процес пошуку розв'язку розпочато

Коли моделювання системи закінчиться і розв'язок буде знайдено, користувачу буде виведено екран результатів (рисунок 6.3). Тут він побачить розв'язок задачі – оптимальну кількість резерву з урахуванням обмежень і

надійність системи, що досягається з відповідним набором резервних елементів – і результати усіх прогонів системи, а саме значення ймовірностей безвідмовної роботи кожної з підсистем для усіх можливих кількостей резервних елементів. Для тих значень кількості резерву, для котрих обчислення ймовірностей не відбувалось, відповідні клітини зафарбовані сірим кольором. Якщо поставити галочку навпроти "Показати значення логарифмів", то користувач побачить логарифми від значень надійності підсистем. Ці логарифми якраз і є значеннями функцій $\varphi_i(m)$, що використовуються для пошуку розв'язку задачі методом динамічного програмування. Натиснувши на хрестик у правому верхньому кутку вікна програми, користувач згорне результати і потрапить на стартовий екран (рисунок 6.1). Натиснути на цю кнопку ще раз – і результати знов розгорнуться.

Розв'язок

Оптимальний резерв підсистем

№1: 3
№2: 2
№3: 2
№4: 1

Надійність системи

0.9518926

Надійність підсистем

К-ть резерву	P1	P2	P3	P4
0	0.606644	0.472178	0.534131	0.687301
1	0.949072	0.897037	0.925001	0.969814
2	0.99617	0.988534	0.993117	0.998308
3	0.999787	0.999086	0.999538	0.999939
4	0.999995	0.999955	0.999987	0.999995
5	1	0.999996	0.999999	1
6	1	0.999999	1	1
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	1	1	1	1

☐ Показати значення логарифмів

Рисунок 6.3 – Екран результатів

Згадуючи результати тестування алгоритму динамічного програмування і те, що за деяких вхідних даних розв'язок так і не знайдеться через нестачу пам'яті, а браузер і пристрій користувача й зовсім можуть зависнути, було передбачено вивід попередження про це, якщо обрано забагато підсистем і завеликі коефіцієнти обмежень (рисунок 6.4). Користувачу дається вибір, повернутися до стартового екрану чи запустити процес пошуку розв'язку.

Серед інших особливостей веб-додатка:

- перед запуском процесу моделювання, усі поля зі значеннями коефіцієнтів перевіряються, і якщо десь було введено нуль, або зовсім нічого не введено, тоді програма не запускається, а відповідні поля виділяються червоним;
- усі дані щодо останніх обраних коефіцієнтів чи кількості підсистем і обмежень зберігаються у локальному сховищі браузера – перезавантажив сторінку, користувач побачить саме те, що він обрав останнього разу;
- інтерфейс повністю адаптований під мобільні пристрої.

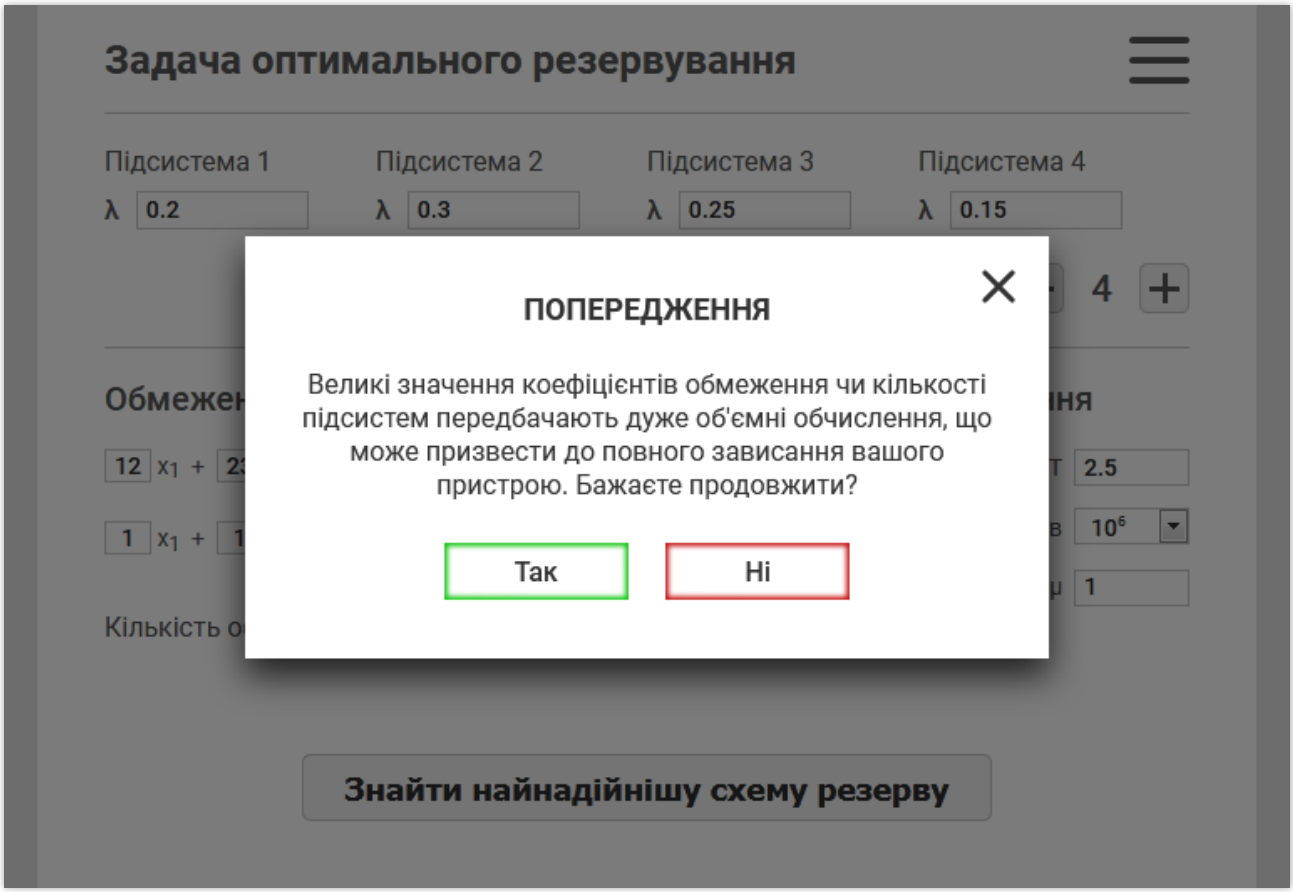


Рисунок 6.4 – Попередження про можливі проблеми

ВИСНОВКИ

Була розглянута нова модифікація методу динамічного програмування для пошуку розв'язку оберненої задачі оптимального резервування з кількома обмеженнями. Перевірки показали, що новий алгоритм знаходить розв'язок досить точно. Проте час, що витрачається на пошук розв'язку, може варіюватися від сотих секунди до кількох хвилин. Це пов'язано із кількістю заданих обмежень і їх обмежуючих коефіцієнтів – чим вони більше, тим більше потрібно алгоритму розглянути можливих комбінацій розподілення резерву, що може призводити до колосальних об'ємів обчислень. Мінусом цього методу також є те, що усі коефіцієнти обмежень повинні бути цілими числами, і якщо в умовах задачі присутні дробові коефіцієнти, усі значення коефіцієнтів відповідного обмеження повинні бути пропорційно перетворені на цілі, що звичайно збільшує обмежуючий коефіцієнт, а значить – збільшує і час роботи алгоритму. Звичайно, можна схитрувати і перетворювати коефіцієнти на цілі непропорційно, щоб максимізувати швидкодію, але за це можна поплатитися точністю розрахунків і врешті-решт отриманий розв'язок буде зовсім неоптимальний.

Однак мова JavaScript, якою було реалізовано алгоритм, не є достатньо швидкою, і за допомогою таких компільованих мов програмування, як C++, Rust чи Go, час виконання обчислень алгоритмом можна скоротити у кілька разів, що зможе відкрити нові можливості для дослідження задач оптимального резервування з трьома і чотирма, а можливо навіть і п'ятьма обмеженнями. Також не виключено, що метод динамічного програмування можливо модифікувати ще таким чином, щоб розрахунки проводилися максимально оптимально, без зайвих обчислень.

Отже, розглянута в цій роботі модифікація методу динамічного програмування є досить вдалою, але простір для дослідження оптимального резервування систем теорії надійності з кількома обмеженнями ще залишається.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Математические методы в теории надежности / Гнеденко Б. В., Беляев Ю. К., Соловьев А. Д. – М. : Наука, 1965. – 524 с. – (Серия "Физико-математическая библиотека инженера")
2. Алексеев А. Е. Диагностика надежности автоматизированных систем: учебное пособие. – А. : ГОУ АГТУ, 2004. – 75 с.
3. Ушаков И. А. Курс теории надежности систем: учеб. пособие для вузов / И. А. Ушаков. – М. : Дрофа, 2008. – 239, [1] с. : ил. – ISBN 978-5-358-01586-9
4. Ушаков И. А. Методы решения простейших задач оптимального резервирования при наличии ограничений. – М. : "Советское радио", 1969. 176 с.
5. Р. Барлоу, Ф. Прошан. Математическая теория надежности. – М. : "Советское радио", 1969. 488 с.
6. Прикладные задачи динамического программирования / Р. Беллман, С. Дрейфус. – М. : Наука, 1965. – 460 с.
7. Имитационное моделирование. Классика CS, 3-е издание / В. Кельтон, А. Лоу К. : Издательская группа BHV, 2004. – 847 с. : ил. – ISBN 966-552-118-7
8. Введение в теорию массового обслуживания, 2-е изд., перераб. и доп. / Гнеденко Б. В., Коваленко И. Н. – М. : Наука, 1987. – 336 с. – (Серия "Физико-математическая библиотека инженера")
9. Програма для розв'язання певних задач оптимального резервування при кількох обмеженнях [веб-сторінка]. URL: https://vladgalafm.github.io/OptimalRedundancy/uk/

ДОДАТОК А

Приклад розв'язку задачі методом динамічного програмування

Занесемо дані для задачі з підрозділу 3.2, отримані в результаті моделювання системи (див. рис. 3.3), у таблицю А.1. Нагадаємо, що задача полягає в максимізації суми логарифмів від отриманих значень ймовірностей за наявності обмеження на загальні витрати, пов'язані із введенням резервних елементів.

Таблиця А.1 – Початкові дані

m	$\varphi_1(m)$	$\varphi_2(m)$	$\varphi_3(m)$
0	-0.999221	-1.999595	-4.988189
1	-0.240570	-0.725253	-2.686117
2	-0.048594	-0.245560	-1.468207
3	-0.007781	-0.072127	-0.776496
4	-0.000975	-0.018269	-0.387517
5	0	-0.003777	-0.178869
6	0	-0.000752	-0.075321
7	0	0	-0.028710
8	0	0	-0.010071
9	0	0	-0.003130
10	0	0	-0.000907
11	0	0	-0.000221
12	0	0	-0.000046

Формалізуємо математичну модель розв'язку задачі.

1. Кількість кроків дорівнює трьом.
2. Нехай s – кількість резервних елементів, що ми маємо перед даним кроком і що характеризує стан системи на кожному кроці.

3. На i-кроці ($i = \overline{1,3}$) m_i – це кількість резервних елементів, яку ми обираємо для i-підсистеми.
4. Значення $\varphi_i(m_i)$ на i-кроці – це значення, отримане при виборі m_i резерву для i-підсистеми.
5. Якщо на i-кроці маємо якесь значення витрат на резерв s і для i-підсистеми обрано m_i резервних елементів, тоді для подальшого розподілення резерву на $(i+1)$-кроці залишається $(s - c_i m_i)$ елементів.
6. На останньому ($i = 3$) кроці $m_3(s) = s$, а $L_3(s) = \varphi_3(s)$.
7. Основне функціональне рівняння має вигляд

$$L_i(s) = \max_{m_i \le \frac{s}{c_i}} \big(\varphi_i(m_i) + L_{i+1}(s - c_i m_i)\big).$$

Проведемо покрокову оптимізацію, за результатами якої заповнимо таблицю А.2.

Таблиця А.2 – Результати покрокової оптимізації

s	$i = 3$		$i = 2$		$i = 1$	
	$m_3(s)$	$L_3(s)$	$m_2(s)$	$L_2(s)$	$m_1(s)$	$L_1(s)$
0	0	-4.988189	0	-6.987784		
1	1	-2.686117	0	-4.685712		
2	2	-1.468207	0	-3.467802		
3	3	-0.776496	0	-2.776091		
4	4	-0.387517	1	-2.193460		
5	**5**	-0.178869	1	-1.501749		
6	6	-0.075321	1	-1.112770		
7	7	-0.028710	1	-0.904122		
8	8	-0.010071	2	-0.633077		
9	9	-0.003130	**2**	-0.424429		
10	10	-0.000907	2	-0.320881		
11	11	-0.000221	3	-0.250996		
12	12	-0.000046	3	-0.147448	**1**	-0.664999

В перша колонка таблиці А.2 заповнюється можливими станами системи, в верхньому рядку – номери кроків і відповідні для них $m_i(s)$ і $L_i(s)$. Так як для останнього кроку $i = 3$ функціональне рівняння має вигляд $m_3(s) = s,\ L_3(s) = \varphi_3(s)$, то дві колонки таблиці А.1, відповідні $i = 3$, заповнюються автоматично за таблицею А.1 початкових даних.

На кроці $i = 2$ основне функціональне рівняння має вигляд

$$L_2(s) = \max_{m_2 \le \frac{s}{2}} \big(\varphi_2(m_2) + L_3(s - 2m_2)\big).$$

Тому для проведення оптимізації на цьому кроці заповнимо таблицю А.3 для різних станів s при кроці $i = 3$.

Таблиця А.3 – Оптимізація на кроці $i = 2$

s	$m_2(s)$	$\varphi_2(m_2)$	$s - 2m_2$	$L_3(s - 2m_2)$	$\varphi_2 + L_3$	$L_2(s)$
12	0	-1.999595	12	-0.000046	-1.999641	-0.147448
	1	-0.725253	10	-0.000907	-0.726160	
	2	-0.245560	8	-0.010071	-0.255631	
	3	-0.072127	6	-0.075321	**-0.147448**	
	4	-0.018269	4	-0.387517	-0.405786	
	5	-0.003777	2	-1.468207	-1.471984	
	6	-0.000752	0	-4.988189	-4.988941	
11	0	-1.999595	11	-0.000221	-1.999816	-0.250996
	1	-0.725253	9	-0.003130	-0.728383	
	2	-0.245560	7	-0.028710	-0.274270	
	3	-0.072127	5	-0.178869	**-0.250996**	
	4	-0.018269	3	-0.776496	-0.794765	
	5	-0.003777	1	-2.686117	-2.689894	

Продовження таблиці А.3

s	$m_2(s)$	$\varphi_2(m_2)$	$s-2m_2$	$L_3(s-2m_2)$	φ_2+L_3	$L_2(s)$
10	0	-1.999595	10	-0.000907	-2.000502	-0.320881
	1	-0.725253	8	-0.010071	-0.735324	
	2	-0.245560	6	-0.075321	**-0.320881**	
	3	-0.072127	4	-0.387517	-0.459644	
	4	-0.018269	2	-1.468207	-1.486476	
	5	-0.003777	0	-4.988189	-4.991966	
9	0	-1.999595	9	-0.003130	-2.002725	-0.424429
	1	-0.725253	7	-0.028710	-0.753963	
	2	-0.245560	5	-0.178869	**-0.424429**	
	3	-0.072127	3	-0.776496	-0.848623	
	4	-0.018269	1	-2.686117	-2.704386	
8	0	-1.999595	8	-0.010071	-2.009666	-0.633077
	1	-0.725253	6	-0.075321	-0.800574	
	2	-0.245560	4	-0.387517	**-0.633077**	
	3	-0.072127	2	-1.468207	-1.540334	
	4	-0.018269	0	-4.988189	-5.006458	
7	0	-1.999595	7	-0.028710	-2.028305	-0.904122
	1	-0.725253	5	-0.178869	**-0.904122**	
	2	-0.245560	3	-0.776496	-1.022056	
	3	-0.072127	1	-2.686117	-2.758244	
6	0	-1.999595	6	-0.075321	-2.074916	-1.112770
	1	-0.725253	4	-0.387517	**-1.112770**	
	2	-0.245560	2	-1.468207	-1.713767	
	3	-0.072127	0	-4.988189	-5.060316	
5	0	-1.999595	5	-0.178869	-2.178464	-1.501749
	1	-0.725253	3	-0.776496	**-1.501749**	
	2	-0.245560	1	-2.686117	-2.931677	

Кінець таблиці А.3

s	$m_2(s)$	$\varphi_2(m_2)$	$s-2m_2$	$L_3(s-2m_2)$	φ_2+L_3	$L_2(s)$
4	0	-1.999595	4	-0.387517	-2.387112	-2.193460
	1	-0.725253	2	-1.468207	**-2.193460**	
	2	-0.245560	0	-4.988189	-5.233749	
3	**0**	-1.999595	3	-0.776496	**-2.776091**	-2.776091
	1	-0.725253	1	-2.686117	-3.411370	
2	**0**	-1.999595	2	-1.468207	**-3.467802**	-3.467802
	1	-0.725253	0	-4.988189	-5.713442	
1	**0**	-1.999595	1	-2.686117	**-4.685712**	-4.685712
0	**0**	-1.999595	0	-4.988189	**-6.987784**	-6.987784

На кроці $i=1$ основне функціональне рівняння має вигляд

$$L_1(s)=\max_{m_1\le\frac{s}{3}}\big(\varphi_1(m_1)+L_2(s-3m_1)\big),$$

а стан системи перед першим кроком $s=12$, тому для проведення оптимізації на цьому кроці заповнимо таблицю А.4.

Таблиця А.4 – Оптимізація на кроці $i=1$

s	$m_1(s)$	$\varphi_1(m_1)$	$s-3m_1$	$L_2(s-3m_1)$	φ_1+L_2	$L_1(s)$
12	0	-0.999221	12	-0.147448	-1.146669	-0.664999
	1	-0.240570	9	-0.424429	**-0.664999**	
	2	-0.048594	6	-1.112770	-1.161364	
	3	-0.007781	3	-2.776091	-2.783872	
	4	-0.000975	0	-6.987784	-6.988759	

Як можемо бачити, $L^*(\overline{m})=L_1(s)=-0.664999$.

$m_1^*=1,\qquad m_2^*=m_2(12-3m_1^*)=m_2(9)=2,$

$m_3^*=m_3(12-3m_1^*-2m_2^*)=m_3(5)=5.$

$\overline{m}^*=(1,2,5)$ – оптимальна кількість резервних елементів.

ДОДАТОК Б

Приклади пошуку значень ймовірності безвідмовної роботи систем з двома та трьома резервними одиницями

Подивимося, як за допомогою програмного пакету Maple можна знаходити значення ймовірностей безвідмовної роботи систем. Розглянемо спочатку на прикладі системи з двома резервними елементами.

```
1. Встановлюємо, за яких λ, μ та T будемо знаходити значення ймовірності безвідмовної роботи
> λ := 1 :
  μ := 10 :
  T := 3 :
2. Визначаємо систему диференційних рівнянь (sys) та початкові умови (cond)
> sys := diff(P[0](t), t) = -λ·P[0](t) + μ·P[1](t),
         diff(P[1](t), t) = λ·P[0](t) - (λ + μ)·P[1](t) + 2·μ·P[2](t),
         diff(P[2](t), t) = λ·P[1](t) - (λ + 2·μ)·P[2](t) :
> cond := P[0](0) = 1, P[1](0) = 0, P[2](0) = 0 :
3. Розв'язуємо систему методом Рунге-Кутти
> result := dsolve({sys, cond}, {P[0](t), P[1](t), P[2](t)}, numeric) :
4. Перевіряємо, яких значень набувають розв'язки за певного часу T
> result(T);
[t = 3., P_0(t) = 0.894493032375599, P_1(t) = 0.0890651986105201, P_2(t)
    = 0.00424204891839047]
5. Підсумовуємо отримані в п.4 значення, отримуємо ймовірність безвідмовної роботи
> R := rhs(result(T)[2]) + rhs(result(T)[3]) + rhs(result(T)[4]);
                          R := 0.987800279904509
```

Рисунок Б.1 – Приклад розв'язку системи в Maple для випадку з $m = 2$

Як можемо бачити, Maple дозволяє без зайвих зусиль отримувати доволі точні значення. Все, що було необхідно – це правильно визначити систему диференційних рівнянь (див. формули (5.7) – (5.10)) і початкові умови.

Для випадку з трьома резервними елементами процедура та сама, треба тільки замінити систему на відповідну для випадку з $m = 3$ (див. формули (5.11) – (5.15)) і визначити відповідні початкові умови.

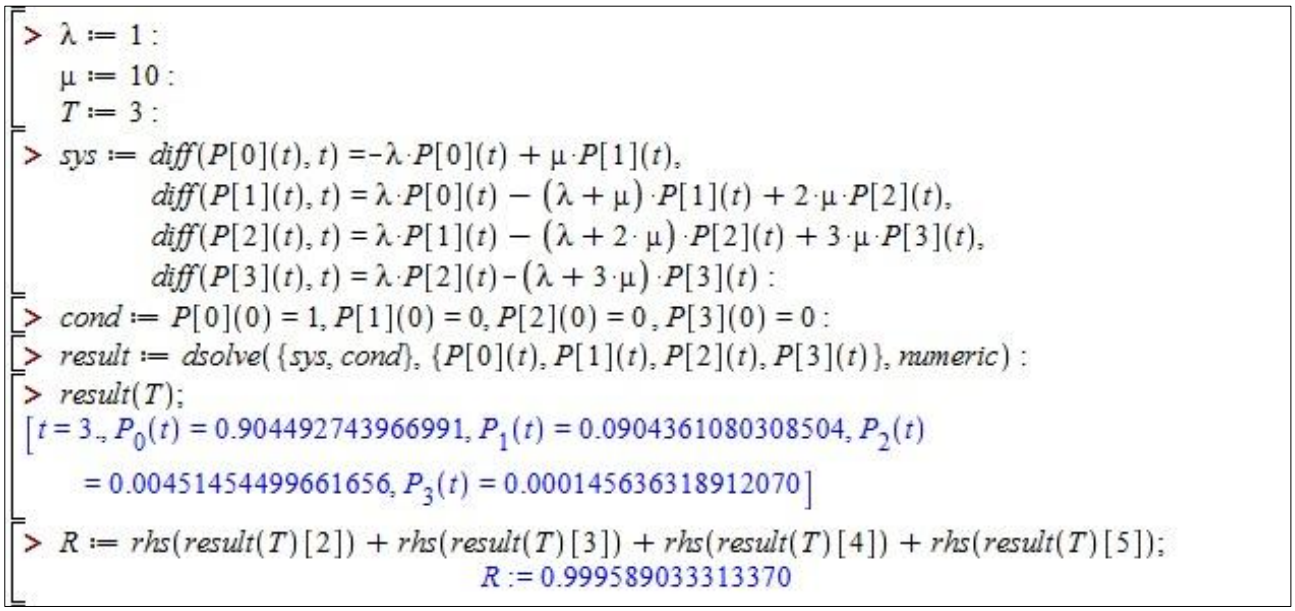


Рисунок Б.2 – Приклад розв'язку системи в Maple для випадку з $m = 3$