

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ**

«На правах рукопису»
УДК 519.2

«До захисту допущено»

В.о. завідувача кафедрою

_____ М.М.Савчук
(підпис) (ініціали, прізвище)

“15” травня 2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 113 «Прикладна математика»

на тему: Послідовний статистичний аналіз в методах виявлення розладки випадкових дискретних процесів

Виконав (-ла): студент (-ка) 2 курсу, групи ФІ -62м

Воробйов Валерій Олександрович

_____ (прізвище, ім'я, по батькові)

_____ (підпис)

Савчук Михайло Миколайович д.ф.-м.н. доцент

Керівник _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Консультант _____

(назва розділу)

(науковий ступінь, вчене звання, прізвище, ініціали)

_____ (підпис)

Хом'як О.М. к.ф.-м.н., н.с. інституту кібернетики ім. В.М.Глушкова

Рецензент _____

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2018року

РЕФЕРАТ

Обсяг роботи 59 сторінок, 5 ілюстрацій, 11 таблиць, 3 додатки, 12 джерел літератури.

Об'єкт дослідження – дискретні випадкові процеси, в яких в деякий момент часу відбувається зміна імовірностних характеристик – розладка.

Предмет дослідження – математичні моделі і алгоритми знаходження розладки.

Методи дослідження: критерії перевірки простих гіпотез; для створення програмної реалізації фреймворку застосовано засіб автоматизації наукових досліджень Python; оцінка побудованого алгоритму виконана за допомогою методу Монте – Карло; поставлені експерименти на реальних та штучно згенерованих даних.

Побудована дискретна модель і реалізовано параметричний алгоритм знаходження розладки. Проведена оцінка алгоритму на синтетичних і реальних даних з наявною розладкою в них і без розладки.

Наукова новизна одержаних результатів: модифіковано вибраний алгоритм та побудовано його для пошуку розладки в дискретних стохастичних процесах.

Практичне застосування. Побудована модель, яка здатна виявити розладку в дискретних стохастичних процесах в таких областях як криптографія, аналіз аномалій у мережевому трафіку, а також в дискретних процесах, де спостерігаються цілочисельні поточні характеристики. Вона дозволяє проводити неперервний моніторинг процесу із інформуванням його стану.

МЕТОДИ ЗНАХОДЖЕННЯ РОЗЛАДКИ, ПАРАМЕТРИЧНІ МЕТОДИ,
ДИСКРЕТНИЙ СТОХАСТИЧНИЙ ПРОЦЕС, КОМП'ЮТЕРНА
СИМУЛЯЦІЯ

ABSTRACT

59 pages, 5 illustrations, 11 tables, 3 applications, 12 sources of literature.

Object of study – discrete random processes in which at a certain moment of time there is a change in probabilistic characteristics – changepoint.

Purpose of study – mathematical models and algorithms of changepoint detection.

Methods: simple hypotheses testing criteria; for software implementation of model means of automation research Python was used; evaluation of the implemented algorithm was carried out using the Monte Carlo method; experiments were conducted on the evaluation of real and synthetically generated data.

A discrete model is constructed and a parametric method for changepoint detection is realized. The evaluation of the algorithm on synthetic and real data with the existing changepoint and without changepoint was carried out.

The scientific novelty of the obtained results: the chosen algorithm was modified and built to search for changepoint in discrete stochastic processes.

Practical implementation. The model has been constructed, which can be used for changepoint detection in discrete stochastic processes for such fields of study as cryptography, analysis of anomalies in network traffic; also it can be applied for changepoint detection in processes with integer flow characteristics; it allows continuously monitor the process and gives the information about its condition on the fly.

CHANGEPOINT DETECTION, PARAMETRIC METHODS, DISCRETE
RANDOM PROCESS, COMPUTER SIMULATION

РЕФЕРАТ

Объем работы 59 страниц, 5 иллюстраций, 11 таблиц, 3 приложения, 12 источников литературы.

Объект исследования – дискретные случайные процессы, в которых в некоторый момент времени происходит изменение вероятностных характеристик – разладка.

Предмет исследования – математические модели и алгоритмы поиска разладки.

Методы исследования: критерии проверки простых гипотез; для создания программной реализации модели применено средство автоматизации научных исследований Python; оценка реализованного алгоритма осуществлялась с использованием метода Монте – Карло; проведены эксперименты по оценке на реальных и искусственно сгенерированных данных.

Построена дискретная модель и реализовано параметрический метод нахождения разладки. Проведена оценка алгоритма на синтетических и реальных данных с имеющейся разладкой в них и без разладки.

Научная новизна полученных результатов: модифицирован выбранный алгоритм и построен для поиска разладки в дискретных стохастических процессах.

Практическое применение. Построена модель, которая способна выявить разладку в дискретных стохастических процессах в таких областях как криптография, анализ сетевого трафика, а также в дискретных процессах, где наблюдаются целочисленные текущие характеристики. Она позволяет проводить непрерывный мониторинг процесса с информированием его состояния.

МОДЕЛЬ ПОИСКА РАЗЛАДКИ, ПАРАМЕТРИЧЕСКИЕ МЕТОДЫ, ДИСКРЕТНЫЙ СТОХАСТИЧЕСКИЙ ПРОЦЕСС, КОМПЬЮТЕРНАЯ СИМУЛЯЦИЯ

ЗМІСТ

Вступ.....	8
1 Огляд літератури	10
1.1 Мотивація та передумови виникнення задачі	10
1.2.1 Класифікація та вимоги до алгоритмів	11
1.2.2 Компроміс між швидкістю знаходження розладки і помилковими тривогами	12
1.3 Огляд прикладних задач.....	15
1.4 Ймовірно-статистичні моделі.	17
1.5 Формальна постановка задачі	20
Висновки до розділу 1	22
2 Побудова параметричної моделі	23
2.1 Загальна модель.....	23
2.2 Модель з загальним розподілом Пуасона.....	27
Висновки до розділу 2	29
3 Модифікація алгоритму та оцінка.....	30
3.1 Опис модифікацій алгоритму	30
3.2 Опис алгоритму	30
3.3 Опис експериментів та критерії оцінки	33
3.4 Реалізація експериментів.....	35
3.4.1 Випадкова послідовність без розладки.....	35
3.4.2 Випадкова послідовність без розладки з шумом	36
3.4.3 Випадкова послідовність з розладкою.....	38
3.4.4 Послідовність осмисленого тексту та випадкова послідовність.....	41
3.4.5 Послідовність шифртексту та випадкова послідовність.....	45
Висновки до розділу 3	48
Висновки	49
Перелік посилань.....	50
Додаток А.....	52
Додаток Б	56
Додаток В.....	58

ВСТУП

Актуальність роботи. Поток даних можуть бути визначені як потенційно нескінченні послідовності упорядкованих спостережень, де кожне спостереження може бути прочитане рівно один раз [1]. Така модель описує ситуацію, коли дані постійно надходять, а також їх кількість може бути занадто великою для безпосереднього зберігання в пам'яті та подальшого аналізу збереженої послідовності. Також невідкладне прийняття рішень може бути необхідним, і тому своєчасний аналіз результатів дає конкурентну перевагу алгоритмам, що працюють з даними «на льоту». Поєднання цих трьох характеристик дає перевагу послідовним або онлайн статистичним методам. Більше того, дана поведінка притаманна багатьом програмам – потоки даних не можуть бути зупинені під час коректування алгоритму і дані будуть продовжувати надходити, незалежно від того, які рішення було прийнято.

Потоки даних змінюються залежно від часу; коли стан системи змінюється, тоді і вимірювані величини будуть трансформуватися. Проте ми можемо очікувати, що існують періоди стабільності, коли спостереження генеруються деяким базовим стохастичним механізмом. Просте формулювання такого процесу полягає в тому, що потік генерується підгрупою розподілів ймовірності, які нерегулярно змінюються з одного режиму на інший. Потік даних може бути вихідною послідовністю з такої моделі. З огляду на це формулювання, проблема, що представляє особливий інтерес, полягає у виявленні точок розладки у потоці даних – точок, при яких змінюється поточний розподіл потоку від одного режиму до іншого. Слід зауважити, що дослідження, наведені в цій роботі, зосереджується на послідовностях з дискретним часом.

В останні роки проблема виявлення змін в потоках даних виникла в таких галузях як астрономія [12], аналіз трафіку комп'ютерних мереж [4,8,10,11] та фінанси [9].

Об'єкт дослідження – дискретні випадкові процеси, в яких в деякий момент часу відбувається зміна імовірностних характеристик – розладка.

Предмет дослідження – математичні моделі і алгоритми знаходження розладки.

Мета дослідження.

Дослідження дискретних процесів, в яких відбувається розладка та огляд алгоритмів, за допомогою яких визначається момент розладки. Реалізація алгоритму пошуку розладки та оцінка.

Для досягнення мети необхідно виконати такі завдання:

- 1) Зробити огляд релевантної літератури для пошуку алгоритмів та їх застосування.
- 2) Реалізувати алгоритм пошуку розладки.
- 3) Запропонувати покращення
- 4) Дослідити точність алгоритму на синтетичних даних.
- 5) Дослідити точність алгоритму для розпізнання шифрованого тексту у випадково згенерованій послідовності.
- 6) Оцінити реалізований алгоритм.

Методи дослідження: критерії перевірки простих гіпотез; для створення програмної реалізації алгоритму застосовано засіб автоматизації наукових досліджень Python; оцінка побудованого алгоритму виконана за допомогою методу Монте – Карло; поставлені експерименти на реальних та штучно згенерованих даних.

Наукова новизна одержаних результатів: створено алгоритм для пошуку розладки в дискретних стохастичних процесах, який з високою ймовірністю розпізнає розладку та дозволяє неперервно слідкувати за процесом.

Практичне застосування. Описаний в роботі алгоритм реалізовано на мові програмування Python і може бути застосовано для аналізу невід'ємних цілочисельних послідовностей. Зокрема при дослідженні шифрів на початковому етапі.

1 ОГЛЯД ЛІТЕРАТУРИ

1.1 Мотивація та передумови виникнення задачі

Механізмами знаходження розладки часто цікавляться, з уваги практичних інтересів, наприклад виявлення несправностей, а саме – з раптових або поступових (початкових) модифікацій, які впливають на процес, не зупиняючи його. Такі механізми необхідні для того, щоб запобігти подальшому виникненню більш катастрофічних подій.

Причини, що призводять до появи фреймворків та методологій для виявлення розладки, можна підсумувати наступним чином:

- з теоретичної точки зору, такі підходи дозволяють нам обробляти непередбачені зміни, і це є природним аналогом адаптивного фреймворку і сучасною технологією, яка в основному може мати справу лише з явищами, що змінюються повільно. Виникає можливість наблизитись до аналізу нестационарних явищ;
- з практичної точки зору, інструменти прийняття рішення на основі статистики для виявлення та оцінки розладки викликають потенційний інтерес до таких видів проблем:
 - контроль якості та виявлення несправностей в вимірювальних системах та промислових процесах в контексті покращення характеристик обладнання та технічного обслуговування;
 - автоматична сегментація сигналів як перший етап обробки сигналів, орієнтованих на розпізнавання;
 - отримання оновлення в адаптивних алгоритмах ідентифікації для підвищення їх точності відстеження.

Як правило, реалізуються методи виявлення розладки у цих типах ситуацій за допомогою різних підходів і обмежень (наприклад, шляхом вибору

різних моделей і критеріїв; шляхом налаштування параметрів детекторів), але в основному застосовуються однакові методологія та інструменти у більшості ситуацій[1].

У деяких практичних програмах всі три типи задач можуть бути вирішені разом.

1.2 Огляд теоретичного порівняння алгоритмів

Теоретичні дослідження показують, що найбільш швидкі розв'язки задачі виявлення розладки було знайдено у двох різних напрямках: Байєсівський та мінімакс. У випадку Байєса передбачається, що розладка є випадкова змінна, незалежна від спостережень з відомим розподілом. Навпаки, мінімакс випадок передбачає, що розладка є невідомим не випадковим числом.

Тзе Льонг Лай (Tze Leung Lai) розробив обмежений проміжком алгоритм з узагальненим коефіцієнтом правдоподібності з обмеженим обсягом складності для он-лайн обробки, яка асимптотично досягає нижньої межі.

1.2.1 Класифікація та вимоги до алгоритмів

Алгоритми виявлення точки розладки традиційно класифікуються як "онлайн" та "офлайн". Автономні ("офлайн") алгоритми розглядають весь набір даних одночасно, потім переглядають хронологічні дані, щоб визначити, де відбулася розладка. Метою даного сценарію є загальне визначення всіх

точок розладки послідовності в пакетному режимі. На відміну від "офлайн" алгоритмів, алгоритми в режимі реального часу ("онлайн") виконуються одночасно з процесом, який вони контролюють, обробляючи кожну одиницю даних, коли вона стає доступною, з метою виявлення точки розладки якнайшвидше після її появи, в ідеалі перед обробкою наступної одиниці даних. [3]

На практиці, алгоритм виявлення точок розладки не працює в ідеальному реальному часі, оскільки він повинен перевіряти нові дані, перш ніж визначити, чи відбулася розладка між старими та новими одиницями даних. Проте різні "онлайн" алгоритми вимагають різної кількості нових даних до виявлення точок розладки. На основі цього спостереження введемо нові терміни. Будемо позначати як ϵ -реальний часовий алгоритм - "онлайн" алгоритм, який потребує щонайменше ϵ зразків даних у новій партії даних, щоб мати можливість знаходити точки розладки. "Офлайн" алгоритм може розглядатися як ∞ -реальний часовий алгоритм, а повністю-"онлайн" алгоритм - 1-реальний часовий алгоритм, тому що для кожної одиниці даних він може передбачити, чи відбувається точка розладки перед новою одиницею даних. Отже, мінімізація значення ϵ призводить до створення стійкіших, але чутливіших до виявлення точок розладки алгоритмів.

Підкреслимо, що офф-лайн підхід може бути корисним для розробки рішення та/або оцінки алгоритмів, які можуть стати підґрунтям впровадженням он-лайн підходу.

1.2.2 Компромiс між швидкістю знаходження розладки і помилковими тривогами

Задача виявлення розладки перш за все стосується виявлення змін у стані процесу. У послідовному налаштуванні доки поведінка спостережень

узгоджується з початковим або цільовим станом, то це одна з підстав щоб продовжити процес. Якщо стан зміниться, тоді ми зацікавлені у виявленні цієї зміни, як правило, якомога швидше після її виникнення. Будь-яка методика визначення може призвести до помилкового визначення моменту виникнення розладки – «помилкова тривога». Прагнення виявляти зміни швидко призводить до того, що алгоритм буде спрацьовувати та найменших змінах процесу і буде видавати велику кількість помилкових тривог, навіть якщо не сталося змін. З іншого боку, намагання уникнути помилкових тривог надто ретельно призводить до тривалої затримки між часом виникнення реальної розладки і її виявленням. Суть задачі виявлення розладки полягає в розробці методу виявлення, який мінімізує середню затримку виявлення, пов'язану з середньою частотою помилкового визначення моменту виникнення розладки.

Іншими критеріями, що можна скористатися для розробки та оцінки алгоритмів виявлення розладки є:

1. середній час між помилковими тривогами;
2. імовірність помилкового виявлення;
3. середня затримка для виявлення;
4. ймовірність невиявлення;
5. точність оцінок часу появи розладки.

Інша властивість алгоритмів виявлення розладки, що має велике практичне значення, – це стійкість. Алгоритми, які є стійкими щодо умов шуму та помилок моделювання і які легко налаштовуються на новий сигнал, очевидно, користуються перевагою на практиці. Але надзвичайно складно формально, в загальному вигляді описати випадки та умови стійкості алгоритмів, тому для кожної конкретної задачі доводиться обирати специфічні параметри та обмеження. В цьому і полягає одна з найголовніших складностей розробки стійких алгоритмів виявлення розладки.

Іншим важливим питанням при розробці алгоритмів виявлення розладки є використання попередньої інформації про розладку. Коли вже вибрано структуру та параметризацію моделі, корисно, якщо не обов'язково вивчити,

що відомо про можливі значення параметрів до і після розладки, і як використовувати цю попередню інформацію.

Важливим критерієм застосовності алгоритму є властивість масштабованості, тому що часові ряди реальних даних від таких джерел, як діяльність людини або супутники дистанційного зондування, стають дедалі більшими як у кількості точок (одиниць) даних, так і в числі вимірів. Методи виявлення розладки повинні бути спроектовані обчислювально ефективним способом, щоб вони мали можливість масштабування для великих розмірів даних. Добре порівнювати обчислювальні витрати альтернативних алгоритмів CPD, щоб визначити, який з них може досягти оптимального (або задовільного) рішення якнайшвидше. Один із способів порівняння обчислювальних витрат алгоритмів полягає в тому, щоб визначити чи являється алгоритм параметричним або непараметричним. Відмінність між параметричними та непараметричними підходами важлива, оскільки непараметричні підходи продемонстрували більший успіх для великих наборів даних.

Параметричний підхід визначає конкретний функціональну форму розподілу даних, яку слід досліджувати за допомогою моделі, а потім оцінює невідомі параметри на основі навчальної вибірки даних. Коли модель буде навчена, навчальна вибірка може бути відкинута. На противагу цьому, непараметричні методи не роблять жодних припущень щодо розподілу основної функції. Відповідний недолік, полягає в тому, що всі доступні дані повинні зберігатися при обробці прийняття рішення .

Успішний алгоритм повинен запропонувати баланс між якістю рішення та обчислювальними витратами. Одним із перспективних підходів є використання anytime алгоритмів, які дозволяють будь-коли переривати виконання, і виводити найкраще рішення, отримане до цього часу. Аналогічним способом є контрактний алгоритм, який також обчислює час розрахунку рішення якості, але заздалегідь дає допустимий час виконання як тип контракту. На відміну від anytime алгоритму, контрактний алгоритм

отримує допустимий час виконання як заданий параметр. Якщо контрактний алгоритм переривається до закінчення виділеного часу, це може не дати ніяких корисних результатів. Алгоритм з прериванням (наприклад, anytime алгоритм) - це той, час виконання якого не виділяється заздалегідь, і тому повинен бути готовим до переривання в будь-який момент часу, але він використовує доступний час для постійного підвищення якості свого рішення. Взагалі, кожен алгоритм з прериванням тривіально є контрактним алгоритмом, але зворотне твердження не є вірним.

При дослідженні алгоритмів необхідно брати до уваги всі обмеження, що накладаються на дані, що аналізуються та реалізацію самого алгоритму.

Підходи до CPD також можна виділити на основі вимог, які накладаються на вхідні дані та алгоритм. Ці обмеження важливі для вибору відповідної техніки для виявлення точок розладки в певній послідовності даних. Обмеження, пов'язані з характером даних часових рядів, можуть виходити з стаціонарності, розмірності або неперервності даних.

Деякі з алгоритмів вимагають інформації про дані, наприклад, кількість точок розладки даних, кількість станів у системі та особливості системних станів. Іншим важливим питанням в параметричних методах є те, наскільки алгоритм чутливий до вибору початкових значень параметрів [3].

1.3 Огляд прикладних задач

Проблема знаходження розладок виникає у різних галузях науки і техніки та мають величезний спектр важливих застосувань, включаючи спостереження та моніторинг навколишнього середовища, біомедичний сигнал та обробка зображень, інженерія контролю якості, виявлення несправностей зв'язку в мережах зв'язку, виявлення вторгнення в

комп'ютерних мережах і системи безпеки, виявлення та відстеження таємних ворожих дій, хімічних та біологічних систем виявлення агресора як засоби захисту від терористичних нападів, виявлення епідемії, виявлення несправностей у системах виробництва та великих машинах, виявлення цілей в системах нагляду, економетрика, фінансові ринки, виявлення сигналів з невідомим часом прибуття в сейсмологію, навігацією, обробкою радіолокаційних і сонарних сигналів, сегментацією мовлення та аналізом історичних текстів. У всіх цих додатках датчики приймають спостереження, які зазнають змін у розподілі у відповідь на зміни та аномалії у середовищі або зміни у моделях певної поведінки. Спостереження отримують послідовно, і до тих пір, поки їх поведінка відповідає нормальному стану. Якщо стан змінюється, тоді з'являється потреба у виявленні цього якнайшвидше. Протягом останніх років виникла низка нових сфер застосування: структурний моніторинг мостів, вітрових турбін та літаків, моніторинг динаміки залізничного транспорту, виявлення інцидентів дорожнього руху або зміни в умовах дорожнього руху, моніторинг компонентів низького споживання автотранспортних засобів, діагностика автомобільних антиблокувальних систем, спостереження за щоденними оцінками захворювань, біологічне спостереження

Зокрема, за допомогою послідовного аналізу вирішено низку комп'ютерних і мережевих проблем: виявлення аномалій в IP-мережах, захищена IP-телефонії, виявлення вторгнень, вірусів та іншої відмови в обслуговуванні (DoS), в тому числі сканування мережі з подальшим вторгненням виявлення біотероризму та інші аспекти глобальної безпеки, характеристика моделей доступу до Інтернету, відстежування вподобань користувачів у рекомендаційних системах[2].

В [4] було розроблено двомірний параметричний механізм виявлення (bPDM), який може виявити аномалії та низькочастотні атаки протягом декількох секунд. Цей підхід дозволяє оцінювати параметри моделі в режимі реального часу, і для тренувань потрібно лише 2-3 секунди фонового трафіку.

Відстеження параметрів кількості переданих пакетів та розміру пакетів дозволяє виявляти аномалії в зашифрованому трафіку та уникнути інтенсивного відстеження потоку, оскільки метод не використовує трафік. Об'єднання відстежування цих двох метрик також усуває більшість помилкових спрацьовувань. Було оцінено методи, на основі згенерованих слідів та емуляції атак Iperf, і виявлено, що bPDM може виявити атаки за кілька секунд. Для всіх розглянутих наборів даних, а також основної теоретичної моделі, було виявлено, що час виявлення розладки зменшується, оскільки SNR бітрейту збільшується. Крім того, при збільшенні швидкості нападу час виявлення зменшується; оскільки рівень фонового трафіку зростає, час виявлення зменшується.

1.4 Ймовірно-статистичні моделі.

Математичні моделі випадкових явищ, що вивчаються в теорії ймовірностей і математичній статистиці, ґрунтуються на понятті імовірнісного простору (Ω, \mathcal{A}, P) , де $\Omega = \{\omega\}$ – непорожня множина, яку називають простором елементарних подій (множина всіх можливих результатів досліджуваного випадкового явища), \mathcal{A} – σ -алгебра його підмножин і P – імовірнісна міра на ній.

Випадкова величина – це функція, яка відображає простір елементарних подій в множину дійсних чисел (її можна розуміти як деяку числову характеристику експерименту з випадковим результатом). Випадкову величину ми будемо позначати символом $X = X(\omega)$ (це може бути число або вектор деякої розмірності), а її реалізацію – відповідною буквою нижнього регістру x .

В даній роботі обмежемося розглядом випадкових величин лише дискретного типу, коли X складається з кінцевого або зліченного числа точок $x_1, x_2, \dots, x_n, \dots$.

В імовірнісних задачах щільність досліджуваної випадкової величини повністю відома, але в статистичних – вона відома лише з тим або іншим ступенем невизначеності. Часто при цьому передбачається, що щільність задана з точністю до значень тих чи інших параметрів, від яких залежить функція $f(x)$, – в таких випадках говорять про параметричні статистичні моделі.

На практиці часто зустрічаються ситуації, коли експеримент полягає в проведенні серії повторних незалежних спостережень над деякою випадковою величиною ξ . Якщо проводиться n спостережень, то вибірка $X = (X_1, \dots, X_n)$ являє собою n незалежних копій величини ξ , тобто є n -мірним випадковим вектором з незалежними і однаково розподіленими компонентами. У цьому випадку щільність $f_X(x), x = (x_1, \dots, x_n)$, вибірки $X = (X_1, \dots, X_n)$ має вигляд вид $f_X(x) = f_\xi(x_1) \dots f_\xi(x_n)$, тобто повністю визначається щільністю випадкової величиною, що спостерігається. Кажуть, що $X = (X_1, \dots, X_n)$ – випадкова вибірка об'єму n з розподілу $L(\xi)$.

Випадкова вибірка є математичною моделлю незалежних вимірювань, проведених в однакових умовах, і саме такі моделі найчастіше застосовуються на практиці і будуть в основному розглядатися в подальшому.

Оцінювання. У статистичних задачах розглядаються різні функції $T = T(X)$ від вибірки $X = (X_1, \dots, X_n)$, які самі є випадковими величинами (для яких при всіх t визначені ймовірності з функцією розподілу $F_T(t) = P\{T(X) \leq t\}$). Якщо при цьому функція $T = T(X)$ не залежить від невідомого параметра моделі, то її прийнято називати статистикою. У статистичних задачах мова йде або про оцінювання за спостереженнями $X = (X_1, \dots, X_n)$ тієї чи іншої характеристики випадкової величиною, що спостерігається ξ , яка для параметричної моделі завжди є деякою функцією $\tau(\theta)$ від невідомого

параметра (такі функції і називаються параметричними), або про перевірку тих чи інших статистичних гіпотез про закон розподілу $L(\xi)$ (про параметр θ – в разі параметричної моделі).[6]

Якщо для оцінювання параметричної функції $\tau(\theta)$ використовується деяка статистика $T = T(X)$, то вона називається оцінкою $\tau(\theta)$ (для $\tau(\theta)$).

Послідовний аналіз відноситься до теорії математичної статистики та методів обробки даних, у яких загальна кількість спостережень не зафіксована заздалегідь, але якоюсь мірою залежить від даних, що спостерігаються, як тільки вони стають доступними. Послідовний метод характеризується двома компонентами:

1. правило зупинки, яке вирішує, чи зупинити процес спостереження за допомогою (X_1, \dots, X_n) або отримати додаткове спостереження X_{n+1} для $n \geq 1$;

2. правило прийняття рішення, яке визначає дію, що має бути прийнята щодо розглянутої проблеми (оцінка, виявлення, класифікація тощо) після того, як спостереження припинилося.

Позначивши за допомогою τ змінну часу зупинки та d як кінцеве рішення, пара визначає послідовне правило прийняття рішення (або процедуру прийняття рішення). Така пара не може бути унікальною для заданої проблеми. Мета послідовного аналізу полягає у визначенні оптимального правила прийняття рішення d , що задовольняє деяким критеріям. Варто звернути увагу, що якщо τ фіксується з ймовірністю 1, то процедура має апіорний фіксований розмір вибірки. Ми будемо називати такі процедури процедурами фіксованого розміру вибірки.

Проте в проблемах послідовного виявлення розладки ситуація дещо відрізняється. Процедура виявлення розладки ідентифікується часом зупинки залежно від спостережень і рішення про незмінність є еквівалентом рішення про продовження спостереження. Крім того, зазвичай процес спостереження не припиняється навіть після того, як було прийнято рішення про зміну, і все відновлюється знову і знову, що призводить до мультициклічної процедури виявлення.

У деяких інших застосованих статистичних висновках послідовний аналіз є найбільш економічним рішенням, з точки зору розміру вибірки, вартості чи тривалості експерименту. Це випадок так званого обмеженої процедури вибірки, яка забезпечує таку ж потужність, вимагаючи меншого розміру вибірки, ніж найкраща процедура фіксованого розміру вибірки.

По-перше, класичний послідовний аналіз та задача виявлення розладки оперує випадками незалежних та ідентично розподілених спостережень та двох простих гіпотез. Ці припущення можуть бути досить обмеженими для багатьох сучасних випадків застосування.

Практичні потреби різних прикладних областей ведуть дослідників до вивчення більш складних статистичних моделей, розглядаючи:

- не тотожно розподілені та/або залежні спостереження;
- кілька гіпотез (більше двох);
- композиційні гіпотези.

1.5 Формальна постановка задачі

Нехай x_1, x_2, \dots – реалізація, отримана в ході спостереження за незалежними випадковими величинами X_1, X_2, \dots , що характеризує одну з метрик процесу що спостерігається. Нехай також є деякий невідомий (випадковий або ні) параметр θ , що приймає значення в множині $\{0, 1, \dots, \infty\}$. Випадок, коли $\theta = \infty$, будемо вважати нормальним режимом процесу. В цьому випадку всі спостережувані величини є незалежними однаково розподіленими випадковими величинами із щільністю розподілу $f^\infty(x), x \in R$. Випадок $\theta = 0$ інтерпретується як випадок, коли процес з самого початку йде з розладкою. В цьому випадку вважається, що величини X_1, X_2, \dots також є

незалежними однаково розподіленими випадковими величинами із щільністю розподілу $f^0(x), x \in R$.

Типовим є випадок, коли з самого початку спостережень за процесом має місце нормальний хід процесу (з розподілом $f^\infty(x), x \in R$), а потім, в деякий момент θ відбувається збій, або настає розладка. Вона полягає в тому, що у реалізацій $x_\theta, x_{\theta+1}, \dots$ щільність розподілу ймовірностей стає рівною $f^0(x), x \in R$, в той час як реалізації $x_1, x_2, \dots, x_{\theta-1}$ мають щільність $f^\infty(x), x \in R$,

Опишемо задачу в загальному вигляді. Будемо вважати, що розглядається бінарний експеримент

$$(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n \geq 0}; P^0, P^\infty),$$

де (Ω, \mathcal{F}) – деякий вимірний простір, $(\mathcal{F}_n)_{n \geq 0}$ – потік сігма – алгебр, $\mathcal{F}_0 = \{\emptyset, \Omega\}, \mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \dots \subseteq \mathcal{F}$.

Під мірою P^∞ будемо визначати розподіл ймовірностей на (Ω, \mathcal{F}) , що відповідає гіпотезі H^∞ , а саме те, що розладка не мала місце взагалі, тобто визначаємо що $\theta = \infty$.

Під мірою P^0 будемо визначати розподіл ймовірностей на (Ω, \mathcal{F}) , що відповідає гіпотезі H^0 , а саме те, що розладка мала місце з самого початку спостережень, тобто визначаємо що $\theta = 0$.

Отже, враховуючи заданий бінарно-статистичний експеримент $(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n \geq 0}; P^0, P^\infty)$, побудуємо ймовірнісну модель

$$(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n \geq 0}; P^\theta, \theta \in \Theta),$$

де $\Theta = \{0, 1, 2, \dots, \infty\}$.

Міри повинні бути такими, що

$$P^\theta = \begin{cases} P^0, & \text{якщо } \theta = 0, \\ P^\infty, & \text{якщо } \theta = \infty. \end{cases}$$

Також, в додаток до загальних обмежень на ймовірнісні міри P^θ при $0 < \theta < \infty$ повинна виконуватись наступна властивість:

$$P^\theta(A) = P^\infty(A),$$

якщо $A \in \mathcal{F}_n$ і $n < \theta$. Ця властивість показує відсутність впливу майбутнього на минуле.

Будемо вважати $P_n^0 = P^0 | \mathcal{F}_n$, $P_n^\infty = P^\infty | \mathcal{F}_n$, $P_n = \frac{1}{2}(P_n^0 + P_n^\infty)$. В канонічному випадку густини мір P_n^0 і P_n^∞ будемо позначати $f_n^0(x_0, x_1, \dots, x_n)$ і $f_n^\infty(x_0, x_1, \dots, x_n)$ відповідно.

Із визначення θ як моменту, коли змінюється характер розподілу (з P^∞ на P^0), вважаємо, що умовна щільність $f_n^\theta(x_n | x_0, \dots, x_{n-1})$ задається формулою

$$\begin{aligned} f_n^\theta(x_n | x_0, \dots, x_{n-1}) &= \\ &= I(n < \theta) f_n^\infty(x_n | x_0, x_1, \dots, x_{n-1}) + I(n \geq \theta) f_n^0(x_n | x_0, x_1, \dots, x_{n-1}) \end{aligned}$$

Нехай $0 < a \leq 1$ і $\mathfrak{M}_a = \{0 \leq \tau < \infty : P^\theta(\tau < \theta) \leq a\}$ – клас таких скінченних моментів зупинки τ , для яких ймовірність помилкового спрацювання $P^\theta(\tau < \theta)$ менша чи рівна a [5].

Висновки до розділу 1

У даному розділі було описано першопричини, що призвели до появи різного типу задач про розладку. Описані різні види алгоритмів для знаходження розладки, такі як: онлайн і офлайн, параметричні і непараметричні. Розглянуто проблему компромісу між затримкою у знаходженні алгоритмом розладки і реальним часом розладки. Приведено приклади застосування задачі про розладку в методах та системах захисту інформації. Наведено основні визначення, а також описана задача в загальному вигляді і формальна постановка задачі.

2 ПОБУДОВА ПАРАМЕТРИЧНОЇ МОДЕЛІ

2.1 Загальна модель

Тестування гіпотез використовує попередні знання про данні, їх статистичний опис для того, щоб робити вибір серед набору альтернатив [1].

У нашій постановці задачі є дві гіпотези:

H_0 : аномалія відсутня,

H_1 : аномалія присутня.

Умовна щільність імовірності, коли гіпотеза H_i істинна, позначається $p(x|H_i)$ для $i = 0,1$. Будемо вважати спостереження $\{x_k, k = 1, 2, \dots\}$ незалежними і однаково розподіленими, які належать одному з двох розподілів імовірності. З огляду те, що ми маємо дві гіпотези, постає вибір з двох рішень. Є чотири можливих сценаріїв, з яких ми зосереджені на двох: хибно-позитивний (false positive – FP), або помилкова тривога, оголошується, коли алгоритм обирає H_1 , але насправді істинною є H_0 . Вибір H_0 при насправді правдивій H_1 називається хибно-негативним сценарієм (false negative – FN). Ймовірності цих двох сценаріїв:

$$\begin{aligned} \alpha &= P_{FP} = \Pr[H_1|H_0] \\ \beta &= P_{FN} = \Pr[H_0|H_1] \end{aligned} \quad (2.1)$$

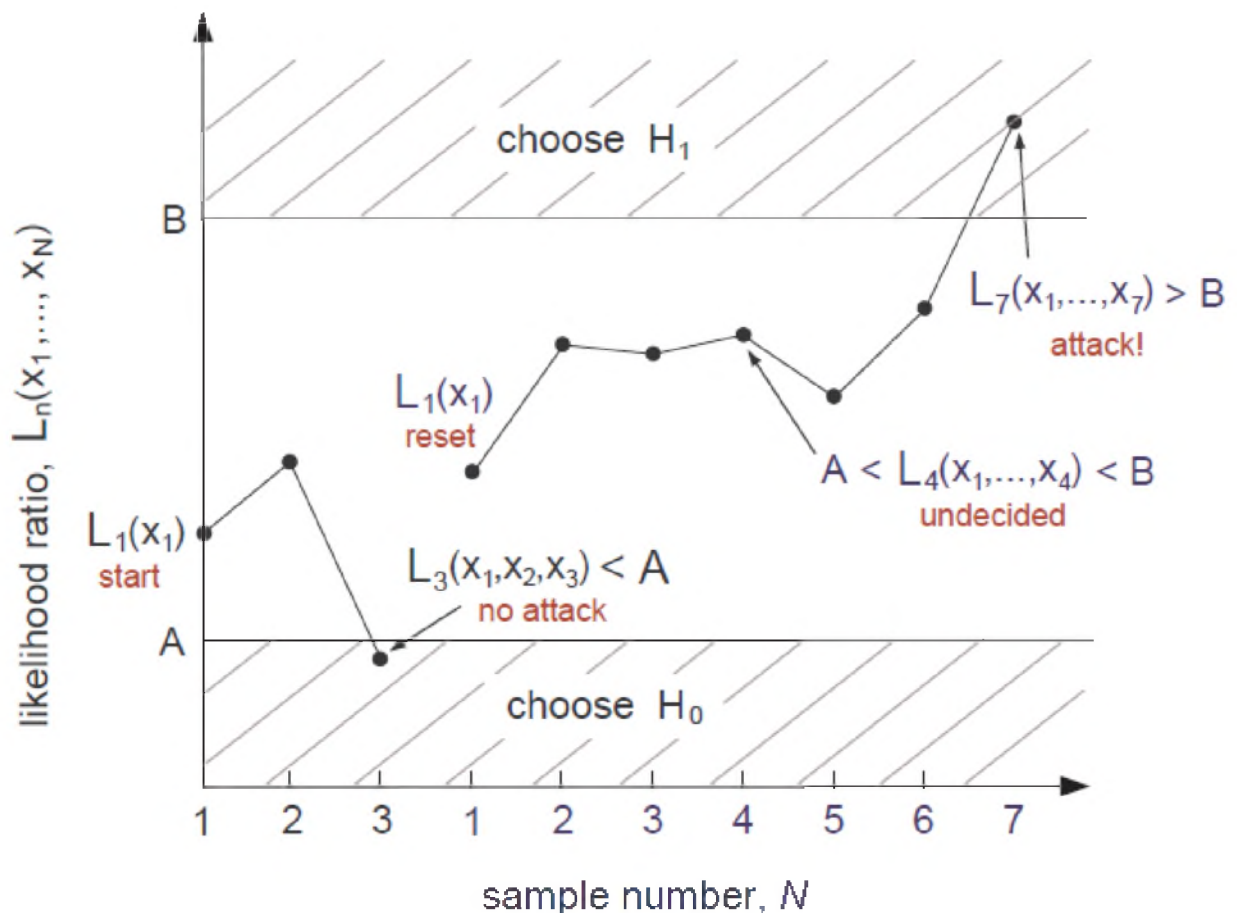
Дані ймовірності використовуються як критерій оцінки якості послідовного тесту вибору гіпотез. Тест виявлення bPDM () використовує послідовний тест співвідношень ймовірностей (sequential probability ratio test) (SPRT) для швидкого виявлення розладки. Співвідношення правдоподібності використовується для реалізації SPRT.

Дано N незалежних та ідентично розподілених спостережень $X = \{x_1, \dots, x_N\}$, тоді відношення правдоподібності $L_N(X)$ визначається як

$$L_N(X) = \prod_{k=1}^N \frac{p(x_k|H_1)}{p(x_k|H_0)} = \frac{p(x_k|H_1)}{p(x_k|H_0)} L_{N-1}(x_1, \dots, x_{N-1}) \quad (2.2)$$

де друга рівність показує, що відношення правдоподібності може бути легко оновлено з урахуванням нового спостереження.

З огляду на нове спостереження, відношення правдоподібності порівнюється з двома порогами A і B , які відповідають вибору H_0 або H_1 , відповідно. На малюнку 2.1 зображена реалізація SPRT.



Малюнок 2.1. Схематичне зображення SPRT

З малюнку 2.1 бачимо, що якщо $A < L_N(x_1, \dots, x_N) < B$, то послідовне тестування продовжується, а додаткове спостереження x_{N+1} приймається так, як на спостереженні L_4 на мал.1. Але, якщо $L_N(x_1, \dots, x_N) \geq B$ або $L_N(x_1, \dots, x_N) \leq A$, тоді тест закінчується і ми обираємо гіпотезу H_1 , якщо має місце перше співвідношення, або гіпотезу H_0 , якщо друге.

На малюнку 2.1 видно, що коли $L_3 < A$, і, відповідно, обирається H_0 , тоді, тестування послідовного співвідношення правдоподібності починається спочатку, аномалія не виявлена, а SPRT триває.

Коли коефіцієнт правдоподібності перетинає будь-який поріг, наприклад на об'ємі вибірки m , послідовний тест скидається шляхом обчислення оновленої ймовірності співвідношення $L(x_{m+1})$ замість $L(x_1, \dots, x_m, x_{m+1})$. Такий випадок демонструється на малюнку 2.1 в точці $L_8 > B$. В цьому випадку обирається H_1 , що вказує на виявлену аномалію. Ми можемо або зупинити тестування зараз (як показано на малюнку 2.1), або скинути SPRT і побачити, чи перетинає співвідношення $L_N(X)$ поріг B знову, потенційно підтверджуючи наявність аномалії.

В ідеалі межі A та B повинні бути обрані для мінімізації ймовірності помилки для всіх можливих значень N ; однак таке формулювання проблеми, як правило, важко вирішувати. Будемо використовувати наближення Вальда для визначення A та B :

$$\begin{aligned} A &\cong \beta / (1 - \alpha) \\ B &\cong (1 - \beta) / \alpha \end{aligned} \quad (2.3)$$

які є функціями від параметрів з (2.1).

Зауважимо, що приблизні значення A та B не залежать від $p(x|H_i)$. Кількість спостережень, необхідних для конкретного випробування для прийняття рішення, є випадковим числом. Таким чином, будемо розглядати середнє значення цього випадкового числа, що називається середнім числом вибірки (average sample number) (ASN), для вимірювання ефективності SPRT. Для бінарного тесту гіпотез, функція ASN позначається $\mathbb{E}_i(N)$ для гіпотези H_i , далі дамо її опис.

$$\begin{aligned} \mathbb{E}_0(N) &= \frac{\alpha \ln B + (1 - \alpha) \ln A}{\mathbb{E}(r)} \\ \mathbb{E}_1(N) &= \frac{(1 - \beta) \ln B + \beta \ln A}{\mathbb{E}(r)} \end{aligned}$$

$$\mathbb{E}(r) = \lambda r + (x - r - 1) \ln[\theta + \lambda(x - r)] + \ln[x!] - (x - 1) \ln[\theta + \lambda x] - \ln[(x - r)!]$$

Класичний SPRT передбачає модель з відомими і постійними параметрами. Насправді такі значення параметрів не завжди доступні, і таким чином ми розглянемо тест узагальненого коефіцієнту правдоподібності (GLRT), визначений як:

$$G_N(X) = \prod_{k=1}^N \frac{p(x_k, \hat{\Theta}_1 | H_1)}{p(x_k, \hat{\Theta}_0 | H_0)} \quad (2.4)$$

де ми використовуємо позначення $p(x_k, \hat{\Theta}_i | H_i)$, щоб позначити заміщення істинних значень параметрів моделі Θ_i з умовною щільністю імовірності $p(x_k | H_i)$ їх оцінкою максимальної правдоподібності (maximum likelihood) (ML). Щоб сформувані узагальнений SPRT, оцінювані параметри замінюються у тесті як описано раніше. Зокрема, ми продовжуємо приймати нові спостереження якщо $A < G_N(x_1, \dots, x_N) < B$, і будемо приймати рішення, обравши H_0 або H_1 , якщо $G_N(x_1, \dots, x_N) \leq A$ або $G_N(x_1, \dots, x_N) \geq B$ відповідно.

Реалізуючи GLRT, атрибути параметрів моделі обох щільностей можна оцінити. Позначимо $\hat{\theta}_i = \hat{\theta} | H_i$ як оцінку $\hat{\theta}$ параметра θ коли H_i правдива. Тут, для обох гіпотез (присутності аномалії та гіпотези про відсутність аномалії) відповідні параметри моделі оцінюються за допомогою спостережень у SPRT для обох наших випадків. Зокрема, параметри моделі оновлюються, використовуючи неперекриваючі вікна спостережень. Спочатку ми використовуємо вікна з фіксованим розміром для обох гіпотез.

Початкові параметри розміру вікна фону (відсутності аномалій) та атаки будуть позначатися як M_{init}, N_{init} .

Зсувне вікно використовується для оцінки параметрів при H_1 і використовуються більш недавні спостереження, таким чином можна виявити зміну в параметрах моделі. Кожного разу, коли SPRT перетинає нижчий поріг, підтверджуючи відсутність атаки, функція ASN обчислюється за гіпотезою H_0 , а розмір вікна оновлення змінюється до

$$M = \min\{E_0(N), M_{init}\} \quad (2.5)$$

Аналогічно, коли аномалія виявляється bPDM, довжина вікна оновлення для параметрів H_1 змінюється на

$$N = \min\{\mathbb{E}_1(N), N_{init}\} \quad (2.6)$$

де перший аргумент функції мінімум в (2.5) і (2.6) – ASN функція при вірній гіпотезі H_0 та H_1 відповідно.

2.2 Модель з загальним розподілом Пуассона

Нульова гіпотеза H_0 , яка являє собою відсутність аномалій, моделюється за допомогою узагальненого розподілу Пуассона (generalized Poisson distribution) (GPD), де функція щільності імовірності (probability density function) (PDF) задана наступним співвідношенням:

$$p(x|H_0) = \theta(\theta + \lambda x)^{x-1} e^{-\theta - \lambda x} / x! \quad (2.7)$$

де $x \in \{0, 1, \dots\}$ – це кількість подій в фіксований інтервал часу, $\{\theta, \lambda\}$ – параметри GPD.

В моделі визначаємо джерело аномалій як потік подій з постійною, детермінованою невідомою швидкістю r . Випадкова величина Y , взята з розподілу з аномалією, вказується як

$$Y = r + X \quad (2.8)$$

де X належить розподілу GPD, який моделює потік подій без аномалій, тобто при гіпотезі H_0 . Для гіпотези з аномалією ми припустимо, що функція щільності розподілу має вигляд

$$p(x|H_1) = \theta(\theta + \lambda(x - r))^{x-r-1} e^{-\theta - \lambda(x-r)} / (x - r)! \quad (2.9)$$

Зауважимо, що у випадку, коли присутня аномалія, r – це мінімальна кількість подій за фіксований інтервал часу.

Визначимо узагальнений коефіцієнт правдоподібності:

$$G_N(X) = \prod_{k=1}^N \frac{p(x_k, \hat{\theta}_1, \hat{\lambda}_1, \hat{r} | H_1)}{p(x_k, \hat{\theta}_0, \hat{\lambda}_0 | H_0)} \quad (2.10)$$

Будемо порівнювати до порогу, наведеного в (2.3). Звернемо увагу, що густини, вказані в (2.10), є GPD (7) і sGPD (2.9) з оцінками параметрів, які використовуються замість відомих значень параметрів. Тепер визначимо оцінки для параметрів GPD та sGPD лише для гіпотези без аномалій та наявності аномалій відповідно. Середнє та дисперсійне значення GPD задаються як

$$\mu = \theta(1 - \lambda)^{-1} \quad \text{та} \quad \sigma^2 = \theta(1 - \lambda)^{-3} \quad (2.11)$$

і використовуються для виявлення моментних оцінок параметрів θ та λ при правдивій гіпотезі H_0 , які визначаються як:

$$\begin{aligned} \hat{\theta}_0 &= \sqrt{\frac{\bar{x}^3}{s^2}} \\ \hat{\lambda}_0 &= 1 - \sqrt{\frac{\bar{x}}{s^2}} \end{aligned} \quad (2.12)$$

де \bar{x} та s^2 – вибіркове середнє та вибіркова дисперсія відповідно, з розміром вікна M . Зазначимо, що вибіркове середнє та дисперсія обчислюються за допомогою їх незміщених оцінок:

$$\begin{aligned} \bar{x} &= \frac{1}{M} \sum_{i=1}^M x_i \\ s^2 &= \frac{1}{M-1} \sum_{i=1}^M (x_i - \bar{x})^2 \end{aligned} \quad (2.13)$$

Для sGPD моменти оцінок трьох параметрів моделі (θ_l, λ_l, r) вимагають обчислення моментів третього та четвертого порядку, які, як ми побачили, вимагали на порядок більшої кількості зразків для обчислення, ніж середній час до виявлення. Використаємо альтернативну, обчислювально легку процедуру оцінки параметрів моделі при вірній гіпотезі H_1 .

З побудови моделі аномалії в (2.8) можна очікувати, що можна отримати незміщену оцінку r просто використовуючи різницю в середній кількості

подій в аномальному та нормальному випадках. Крім того, оскільки ми отримуємо моменти оцінки в рамках SPRT, ми використовуємо оцінку

$$\hat{r} = \max \left\{ \left[-\frac{\hat{\theta}_0}{1 - \hat{\lambda}_0} + \bar{x} \right], \min \{x_1, \dots, x_M, x_{M+1}, \dots, x_{M+N}\} \right\} \quad (2.14)$$

де $\hat{\theta}_0$ та $\hat{\lambda}_0$ визначені в (12). Оцінка r обчислюється на даних як із вікна N , так і з вікна M . Оскільки $(Y - r)$ – це випадкова величина, розподілена за узагальненим розподілом Пуассона, інші два параметри sGPD оцінюються за допомогою оцінки GPD структури в (2.12) і даються як:

$$\begin{aligned} \hat{\theta}_1 &= \sqrt{\frac{(\bar{x} - \hat{r})^3}{s^2}} \\ \hat{\lambda}_1 &= 1 - \sqrt{\frac{\bar{x} - \hat{r}}{s^2}} \end{aligned} \quad (2.15)$$

де \hat{r} - оцінка, наведена в (2.14), а також \bar{x} та s^2 обчислюється за допомогою (2.13) з використанням даних вікна N -вибірки.

Застосування тесту гіпотез GPD / sGPD дозволяє нам виявити зміну в середньому потоці подій, але збільшення середнього значення не завжди є вичерпним прикладним показником.

Висновки до розділу 2

В даному розділі описана загальна модель, що буде використовуватися. Визначено модель із загальним розподілом Пуассона, що буде основою для побудови алгоритму. Визначено початковий алгоритм, який в подальшому буде модернізовано у розділі 3, а також буде реалізовано.

В розділі також наведені основні тотожності та вирази для оцінки вибіркового параметрів. Сформульовано основний критерій, що буде використаний алгоритмом для знаходження розладки.

3 МОДИФІКАЦІЯ АЛГОРИТМУ ТА ОЦІНКА

3.1 Опис модифікацій алгоритму

В подальшому будемо використовувати видозмінений алгоритм з [4] на основі моделі описаної вище. Серез змін відмітимо такі модифікації:

- у функціях зміни розміру вікон вибірки $\mathbb{E}_0(N)$ задля зменшення швидкості спадання об'єму вибірки. Визначимо зміни вікон таким чином $\mathbb{E}_0(M) = \max\left(\left[\frac{M}{2}\right], M_{min}\right)$, $\mathbb{E}_1(N) = \max\left(\left[\frac{N}{2}\right], N_{min}\right)$;
- введення параметрів мінімального розміру опорного та експериментального вікон вибірки, для підтримання її репрезентативності;
- введення параметру зсуву експериментального вікна.

Данні до алгоритму надходять як потік натуральних чисел.

3.2 Опис алгоритму

Алгоритм складається з чотирьох процедур:

1. процедура ініціалізації;
2. процедура набору початкової вибірки;
3. процедура обчислень оцінок;
4. процедура вибору гіпотез.

Опишемо кожен з них окремо:

1. Процедура ініціалізації

Задамо константи:

- M_{init} – початкова довжина опорного вікна M_{sample} ;
- N_{init} – початкова довжина експериментального вікна N_{sample} ;
- M_{min} – мінімальна довжина M_{sample} ;
- N_{min} – мінімальна довжина N_{sample} ;
- A, B – Нижня та верхня межі відповідно, прийняття рішень алгоритму;
- $Warn_max$ – кількість послідовних підтверджень наявності розладки;
- $Warn$ – ітератор для відстеження кількості послідовних підтверджень наявності розладки.

2. Процедура набору початкової вибірки

- Із потоку даних послідовно набрати вибірку M_{sample} розміру $M = M_{init}$
- $step = step + M$
- Із потоку даних послідовно набрати вибірку N_{sample} розміру $N = N_{init}$
- $step = step + N$

3. Процедура обчислень оцінок

- Для вибірок опорного та експериментального вікна обчислити оцінки статистичних характеристик за формулами (2.13).
- Для вибірки опорного вікна обчислити оцінки за формулами (2.12).
- Обчислити \hat{r} за формулою (2.14).
- Для вибірки експериментального вікна обчислити оцінки за формулами (2.15).
- Обчислити значення функції щільності за формулами (2.7) та (2.9) використовуючи оцінки параметрів визначених у кроках 2 – 4.

- Обчислити узагальнений коефіцієнт правдоподібності $G_N(X)$ за формулою (2.10).

4. Процедура вибору гіпотез

Якщо $G_N(X) > B$:

$$Warn = Warn + 1;$$

Якщо $Warn = Warn_{max}$:

$$step = step - N;$$

Зупинити алгоритм.

$$N = \max\left(\left\lceil \frac{N}{2} \right\rceil, N_{min}\right);$$

Обрати з вибірки експериментального вікна N_{sample} N перших елементів.

Якщо $G_N(X) < A$:

$$Warn = \max(0, Warn - 1)$$

$$M = \max\left(\left\lceil \frac{M}{2} \right\rceil, M_{min}\right)$$

Обрати з вибірки опорного вікна M_{sample} M останніх елементів.

Якщо $A \leq G_N(X) \leq B$:

$$Warn = \max(0, Warn - 1)$$

Обрати M останніх елементів з об'єднання вибірок M_{sample}, N_{sample} ;

Із потоку даних послідовно набрати вибірку N_{sample} розміру N ;

$$step = step + N.$$

Отже, алгоритм має такий вигляд:

1. процедура ініціалізації;
2. процедура набору початкової вибірки;
3. процедура обчислень оцінок;
4. процедура вибору гіпотез;
5. повторити кроки 3 та 4 поки вхідний потік є не пустою множиною або не знайдена розладка.

Зауважимо, що вибірка опорного вікна M_{sample} використовується лише для обчислення статистичних оцінок $\hat{\theta}_0$, $\hat{\lambda}_0$ та \hat{r} . Величини щільності ймовірності, для обчислення коефіцієнту правдоподібності, обчислюються на вибірці експериментального вікна N_{sample} . Таким чином ми користуємося вибіркою M_{sample} як пам'яттю алгоритму. Також визначимо те, що алгоритм закінчує роботу при знаходженні першої розладки незалежно від наявності елементів в потоці даних.

3.3 Опис експериментів та критерії оцінки

Поставимо серію експериментів для оцінки правильності роботи описаного алгоритму. Вхідними даними для вищезазначеного алгоритму є натуральні числа. Змоделюємо різні типи вхідних послідовностей даних, поступово ускладнюючи потік вхідної послідовності. А саме:

1. Згенеровані дані з Пуассонівського розподілу без розладки зі сталим математичним очікуванням $\lambda = const$;
2. Згенеровані дані з Пуассонівського розподілу із математичним очікуванням $\lambda \in (const_0, const_1)$;
3. До часу τ випадкові величину з Пуассонівського розподілу без розладки зі сталим математичним очікуванням $\lambda = const_0$ а в момент часу $\tau + 1$ математичне очікування змінюється на $\lambda = const_1$;
4. До моменту часу τ вхідна послідовність задається осмисленим текстом, а в момент часу $\tau + 1$ потік починає доповнюватися випадковою послідовністю. Послідовність являє собою вибірку з повтореннями, з рівноймовірними реалізаціями випадкових величин;

5. До моменту часу τ вхідна послідовність задається осмисленим текстом, який пропущено через шифратор, а в момент часу $\tau + 1$ потік починає доповнюватися випадковою послідовністю. Випадкова послідовність повністю аналогічна описаній в пункті 4.

Для оцінки якості побудованого алгоритму введемо 3 стани на яких алгоритм може завершити свою роботу. Опишемо ці стани:

- Розладку не знайдено – алгоритм опрацьовує увесь набір експериментальних даних і не виявляє розладку на наборі даних.
- Помилкова тривога – алгоритм зупиняє свою роботу з результатом «розладка» і видає крок, на якому він зупинився. Так як в експериментальних даних момент розладки відомий спостерігачу наперед, то є можливість порівняти крок, на якому зупинився алгоритм, і теоретичний момент розладки. Якщо крок зупинки алгоритму менше або дорівнює, ніж різниця теоретичного моменту розладки і розміру вікна M , класифікуватимемо такий експеримент як «помилкова тривога». Звісно, так як в деяких експериментах будуть згенеровані штучно випадкові величини з розподілом Пуассона і сталим математичним очікуванням, це не виключає можливості наявності розладки до теоретичного значення при аналізі вузьких вікон спостережень.
- Розладку знайдено вірно – алгоритм зупиняє свою роботу з результатом «розладка» і видає крок, на якому він зупинився. Крок зупинки алгоритму більший різниці теоретичного моменту розладки і розміру вікна M та менший, ніж сума теоретичного моменту розладки та початкового розміру опорного та експериментального вікон, тобто використовуючи позначення час зупинки як τ , а розмір вибірки до розладки через D_0 , повинна виконуватися нерівність $D_0 - M < \tau < M + N + D_0$ (3.1).

В літературі [1,7] для оцінки якості алгоритмів також використовують таку метрику як *середня тривалість роботи* (Average Run Length). Будемо використовувати скорочення ARL. Визначимо метрику ARL0 для алгоритму як середній час між помилковими тривогами, визначеними алгоритмом. В ту саму чергу через ARL1 визначимо як середній час затримки між моментом виникнення розладки і часом визначення розладки алгоритмом.

Проведемо вищеописані експерименти.

3.4 Реалізація експериментів

3.4.1 Випадкова послідовність без розладки

Метою даного експерименту є перевірка стійкості алгоритму до варіацій випадкових величин, що опрацьовуються.

Проведемо 1000 симуляцій, в кожній з симуляцій випадково обремо математичне очікування $\lambda \in (10,250)$ і для кожного експерименту випадково згенеруємо 2000 величин, з пуассонівським розподілом та математичним очікуванням λ .

Визначимо початкові параметри:

Параметри зупинки алгоритму: $\beta = 5 * 10^{-5}$ $\alpha = 5 * 10^{-5}$

Початкові розміри вікон: $M_{init} = N_{init} = 150$.

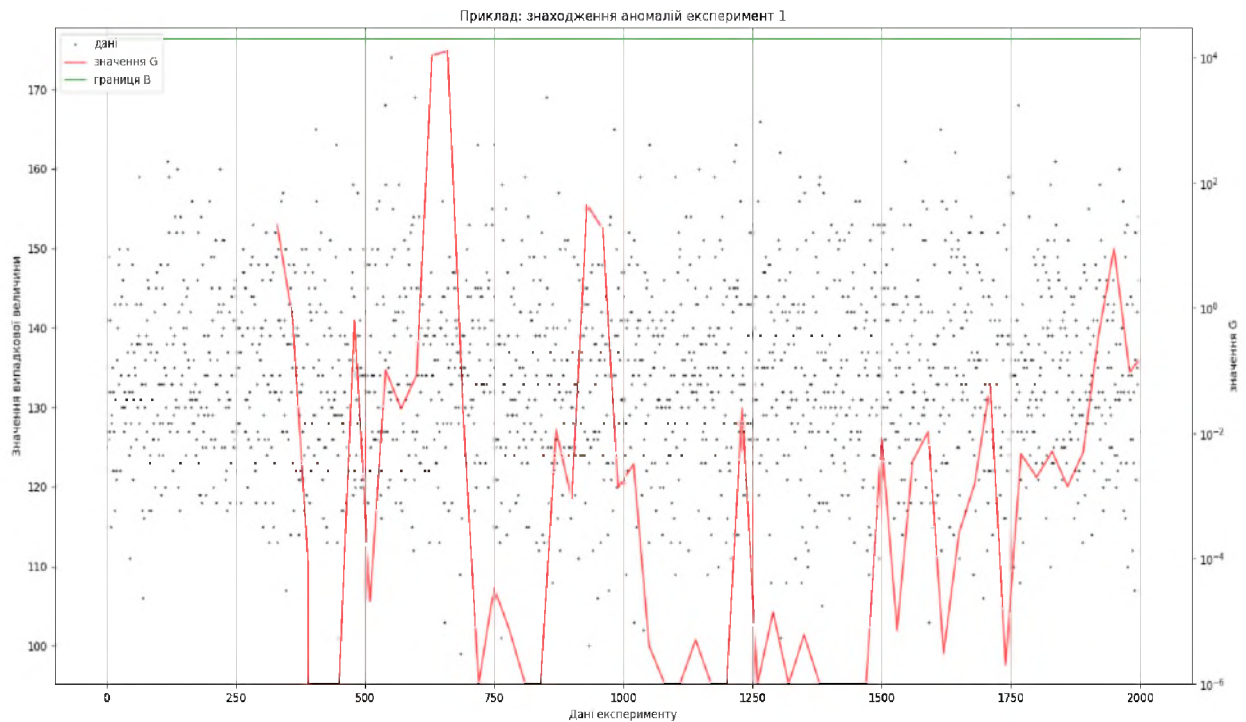
Мінімальні розміри вікон: $M_{min} = 40$, $N_{min} = 30$.

Очікуваним результатом експерименту буде якнайбільша кількість випадків, коли алгоритм не визначив розладку.

Із загальної кількості симуляцій 939 симуляції не визначили розладку і 61 – визначили розладку. Зауважимо, що експерименти, для яких алгоритм

зупинився з результатом «помилкова тривога», не залежать від вибору λ , рівномірно розподілені по вибірці.

Приведемо приклад роботи алгоритму алгоритму на одному з наборів даних (див. малюнок 3.1).



Малюнок 3.1. Візуалізація роботи запропонованого алгоритму з параметрами $\lambda_0 = 134, \text{step} = 30$

3.4.2 Випадкова послідовність без розладки з шумом

Метою даного експерименту є перевірка стійкості алгоритму до наявності шумів у знегерованих даних. Тому кожна випадкова величина обирається з математичним очікуванням з певного інтервалу $\Delta \in \mathbb{N}$. Очікуваним результатом експерименту буде якнайбільша кількість випадків коли алгоритм не визначив розладку спостережуваної залежності від величини інтервалу Δ .

Проведемо 1000 симуляцій для кожної Δ , де $\Delta = [1,5]$, в кожній з симуляцій випадково обремо $const_0 \in [10,250]$, де $const_1 = const_0 + \Delta$, а потім для реалізації кожної випадкової величини оберемо параметр $\lambda \in (const_0, const_1)$. Для кожного експерименту випадково згенеруємо 2000 величин, з пуассонівським розподілом і математичним очікуванням λ .

Визначимо початкові параметри:

Параметри зупинки алгоритму: $\beta = 5 * 10^{-5}$ $\alpha = 5 * 10^{-5}$

Початкові розміри вікон: $M_{init} = N_{init} = 150$.

Мінімальні розміри вікон: $M_{min} = 40$, $N_{min} = 30$.

Приведемо таблицю 3.1 результатів для реалізації експерименту 2.

Таблиця 3.1. – Залежність кількості вірних зупинок алгоритму від величини шуму Δ

Δ	Помилкова тривога	Розладку не знайдено
1	69	931
2	66	934
3	54	943
4	77	923
5	65	935

Середнє значення помилкової тривоги 66.5, що майже не відрізняється від результатів отриманих у першому експерименті. Тобто можна зробити висновок, що алгоритм є стійким до незначних шумів.

3.4.3 Випадкова послідовність з розладкою

Дослідження полягає в обробці потоку даних зі сталими лямбдами, які змінюють значення в деякий момент часу.

Розглянемо випадок, коли λ змінюється у деякий момент часу, для даного експерименту час розладки 2000, тобто до розладки математичне очікування $\lambda_0 = const_0$, а після розладки $\lambda_1 = const_1$. Вибірка після розладки налічує 700 елементів. Проведемо експерименти з різними $\Delta = |const_0 - const_1|$. Зауважимо, що обирати $\Delta = const$ для різних λ_0 є недоцільно, тому визначимо Δ як відсоток від λ_0 , а саме $\Delta = \lambda_0 * k, k \in \{0.05, 0.1, 0.15, \dots, 0.5\}$. Параметр λ_0 оберемо випадково у межах від 10 до 250. Розглянемо таблицю залежності кількості правильно визначених моментів розладки в залежності від розміру k . Для кожного k проведемо 1000 експериментів із випадкового згенерованими величинами. Визначимо початкові параметри:

Параметри зупинки алгоритму: $\beta = 5 * 10^{-5}$ $\alpha = 5 * 10^{-5}$

Початкові розміри вікон: $M_{init} = N_{init} = 150$.

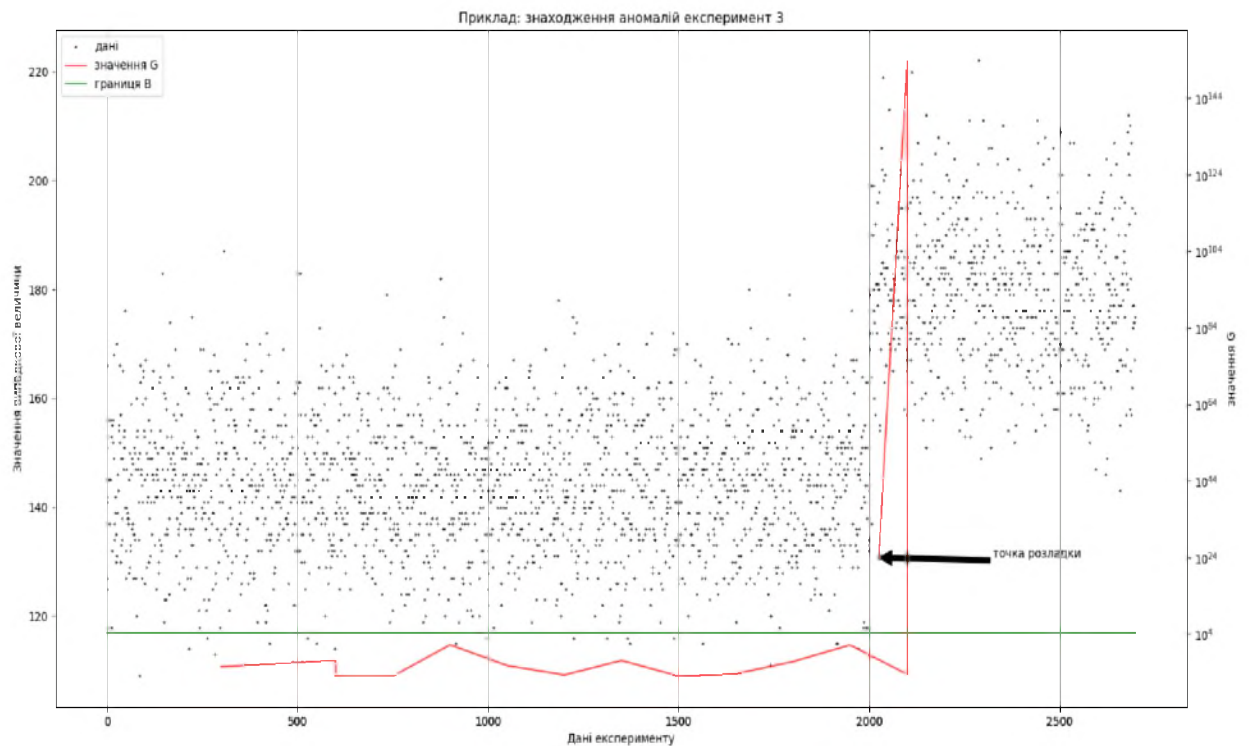
Мінімальні розміри вікон: $M_{min} = 40, N_{min} = 30$.

Представимо залежність правильно розпізнаних моментів розладки від величини k .

Як видно з результатів експерименту (таблиця 3.2), алгоритм дає досить високу точність при зміні Δ більше, ніж на 20 – 25%. Також відмітимо, що показник фальшивої тривоги значно не змінюється в залежності від Δ . Має середнє значення 66.5 випадків на 1000 експериментів, тобто приблизно у 6.6% випадків. Приклад роботи на одному з наборів даних при $k = 0.25$ (малюнок 3.2).

Таблиця 3.2. – Залежність кількості різних зупинок алгоритму від величини шуму Δ

k	Розладку знайдено вірно	Помилкова тривога	Розладку не знайдено
0.05	60	61	879
0.1	253	55	692
0.15	640	58	302
0.2	817	71	112
0.25	902	70	28
0.3	916	75	9
0.35	933	64	3
0.4	930	69	1
0.45	931	69	0
0.5	937	63	0



Малюнок 3.2. – Візуалізація роботи запропонованого алгоритму з параметрами $\lambda_0 = 144, \text{step} = N$

Проаналізуємо як впливає величина математичного λ_0 на правильність відшукування розладки. До уваги візьмемо результати для $k = 0.2$ та $k = 0.25$. Також розіб'ємо інтервал $[10,250]$ на 6 частин. Продемонструємо залежність відсотку вірно знайдених розладок від величини λ_0 .

Таблиця 3.3. – Відсоток правильно розпізнаних розладок в залежності від λ_0

k	Проміжки інтервалу $[10,250]$					
	$[10,50]$	$[51,90]$	$[91,130]$	$[131,170]$	$[171,210]$	$[211,250]$
0.2	56.4	70.1	88.5	92.3	92.5	92.9
0.25	85.5	89.1	89.6	92.3	91.8	93.6

Можна побачити, що алгоритм краще розпізнає розладку для великої різниці Δ .

Проведемо дослідження залежності задання початкових параметрів на правильність знаходження розладки. Поставимо експеримент, визначений в пункті 3.4.3, але замість досліджуваного параметра раниці між середніми значеннями до і після розладки $k = \frac{\Delta}{\lambda_0}$, будемо маніпулювати початковими і мінімальними розміра вікон $M_{init}, N_{init}, M_{min}, N_{min}$. Зафіксуємо параметри $\beta = 5 * 10^{-5}$ $\alpha = 5 * 10^{-5}$. Зафіксуємо $k = 0.2$ та проведемо 300 експериментів для кожної комбінації розмірів вікон. Послідовність до розладки налічує 3000 випадкових величин, після розладки 1000.

З таблиці 3.4. бачимо, що відсоток помилкових тривог дуже зростає при вузьких вікнах через чутливість до шумів. Виходячи з даних оптимальним є набір параметрів (300, 300, 50, 50). Його і оберемо в якості основного для проведення подальших експериментів.

Таблиця 3.4. – Відсоток правильно розпізнаних розладок в залежності від початкових параметрів.

Розміри вікон ($N_{init}, M_{init}, N_{min}, M_{min}$)	Розладку знайдено вірно	Помилкова тривога	Розладку не знайдено
(100, 100, 10, 10)	75	218	7
(150, 150, 20, 20)	157	136	7
(150, 150, 30, 30)	242	52	6
(200, 200, 40, 40)	262	38	0
(200, 250, 40, 50)	266	17	17
(300, 300, 50, 50)	282	18	0
(300, 400, 50, 60)	277	18	5
(300, 500, 50, 60)	269	19	12

3.4.4 Послідовність осмисленого тексту та випадкова послідовність

Експеримент полягає в аналізі послідовності, що змінюється в деякий момент часу з осмисленого тексту на випадкову послідовність. Символи, що генеруються обома послідовностями обираються з одного алфавіту.

Визначимо алфавіт Z . Для простоти даного експерименту оберемо алфавіт $Z = \{a, b, c, \dots, z\}$, тобто латинські літери нижнього регістру. Оберемо тестовий осмислений текст з алфавіту Z для підрахунку частот символів, що зустрічаються у тексті. Приведемо текст до алфавіту Z видаляючи всі символи не з Z . Обчислимо частоти, з якими кожен з елементів алфавіту зустрічається у тестовому тексті та відсортуємо алфавіт за частотами в прямому порядку. Елементи, які зустрічаються часто в кінці, рідко вживані на початку.

Далі використаємо перетворення літер з алфавіту у числа з \mathbb{N} за правилом: найменш вживаному символу алфавіту поставимо у відповідність одиницю, наступному за вживаністю двійку і так далі. Оберемо відмінний від попереднього тестового досить великий текст для подальших експериментів. Будемо обирати з нього уривки фіксованої довжини, що можуть перетинатися між експериментами, але не в одному експерименті. Зафіксуємо довжину уривку у 3000 символів. За перетворенням описаним вище будемо обробляти текст зводячи його до послідовності натуральних чисел починаючи з 1. В подальшому будемо додавати до потоку натуральних чисел, що аналізуються згенеровану випадкову послідовність з того самого алфавіту Z . Для даного експерименту довжина випадково згенерованої послідовності буде дорівнювати 1000 символів. Проведемо 1000 симуляцій даного експерименту.

Розладку знайдено вірно	Помилкова тривога	Розладку не знайдено
998	2	0

Визначимо початкові параметри:

параметри зупинки алгоритму: $\beta = 5 * 10^{-7}$ $\alpha = 5 * 10^{-7}$;

початкові розміри вікон: $M_{init} = N_{init} = 150$;

мінімальні розміри вікон: $M_{min} = 40$, $N_{min} = 30$.

Результати наведені нижче в таблиці 3.4.

Таблиця 3.5. – Сумарні показники зупинки алгоритму

Змінимо початкові параметри на

початкові розміри вікон: $M_{init} = 500$, $N_{init} = 300$;

мінімальні розміри вікон: $M_{min} = 200$, $N_{min} = 150$.

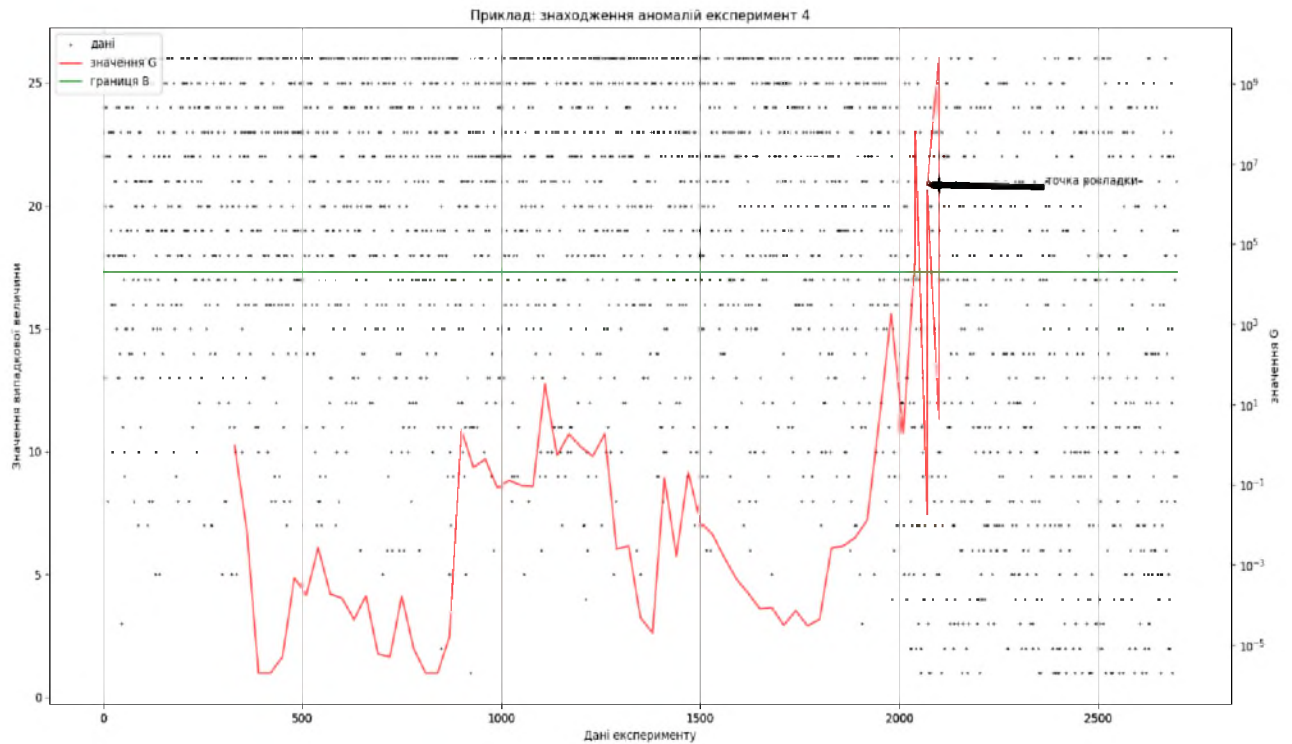
Результати наведені нижче в таблиці 3.5.

Таблиця 3.6. – Сумарні показники зупинки алгоритму

Розладку знайдено вірно	Помилкова тривога	Розладку не знайдено
972	25	3

За результатами отриманих даних бачимо, що алгоритм з досить високою вірогідністю розпізнає осмислений текст від випадкової послідовності символів.

Продемонструємо реалізацію одного з експериментів (див. малюнок 3.3).



Малюнок 3.3. Візуалізація роботи запропонованого алгоритму з параметром $step = 30$

Подальшим кроком експерименту буде визначення відхилення експериментально визначеного алгоритмом моменту розладки від реального значення в залежності від розміру початково обраних вікон. Використаємо дані отримані на попередньому етапі. З них оберемо лише ті експерименти, в яких розладку було знайдено вірно.

Таблиця 3.7. – Відхилення експериментально визначеного алгоритмом моменту розладки від реального значення в залежності від розміру початково обраних вікон; параметри: $M_{init} = 500, N_{init} = 300, M_{min} = 200, N_{min} = 150$

Реальний момент розладки	Експериментальний момент розладки	Кількість
3000	3050	845
3000	3200	153

Таблиця 3.8. – Відхилення експериментально визначеного алгоритмом моменту розладки від реального значення в залежності від розміру початково обраних вікон; параметри: $M_{init} = N_{init} = 150, M_{min} = 40, N_{min} = 30$

Реальний момент розладки	Експериментальний момент розладки	Кількість
3000	3008	2
3000	3017	1
3000	3037	49
3000	3075	920

Для експериментальних даних в таблиці 3.6. $ARL1$ дорівнює 72.9, а в таблиці 3.7 73. Як бачимо, вибір початкової та мінімальної ширини опорного та експериментального вікна спостережень впливає на розкид визначення розладки та швидкість реагування алгоритму, але середній час затримки залишається майже рівним. При більших вікнах можливі менш точні оцінки кроку зупинки алгоритму.

3.4.5 Послідовність шифртексту та випадкова послідовність

Експеримент полягає в аналізі послідовності, що змінюється в деякий момент часу з шифрованого тексту на випадкову послідовність. Символи, що генеруються обома послідовностями обираються з одного алфавіту.

В подальшому експерименті, на відміну від попереднього, пропонується додати шум до тексту. Наприклад, зашифрувати текст деяким шифром задля зменшення властивостей мови, а саме нерівномірності зустрічання символів з алфавіту мови. Шифротекст отриманий на виході сучасних стійких шифрів має статистично не відрізнятися від випадкової послідовності. Отриманий в результаті накладання шуму текст будемо порівнювати із випадковою послідовністю чисел з алфавіту цілих чисел що лежать на інтервалі $[1,256]$. Для даного експерименту використаємо реалізацію блочного шифру Camellia в режимі CBC (Cipher Block Chaining). Реалізація даного і подальших шифрів та криптопримітивів взята з пакету з відкритим кодом Cryptography на мові програмування Python.

Зашифруємо тестовий текст на довільному випадковому ключі шифром Camellia в режимі CBC. Ключ візьмемо довжиною 128 біт. Обчислимо частоти, з якими кожен з елементів зустрічається у зашифрованому тестовому тексті та відсортуємо алфавіт за частотами в прямому порядку. Елементи, які зустрічаються часто в кінці, рідко вживані на початку.

Візьмемо інший текст та зашифруємо його на іншому довільному випадковому ключі шифром Camellia в режимі CBC. Довжина ключа 128 біт. Експеримент полягає у виборі із зашифрованого тексту уривку фіксованої довжини, та доповнення потоку згенеровану випадковою послідовністю фіксованої довжини. Уривки шифрованого тексту можуть перетинатися між експериментами, але не в одному експерименті. Зафіксуємо довжину уривку у 3000 символів. Довжину випадкової послідовності 2000 символів. Визначимо початкові параметри:

параметри зупинки алгоритму: $\beta = 5 * 10^{-7}$ $\alpha = 5 * 10^{-7}$;

початкові розміри вікон: $M_{init} = 300, N_{init} = 300$;

мінімальні розміри вікон: $M_{min} = 50, N_{min} = 50$.

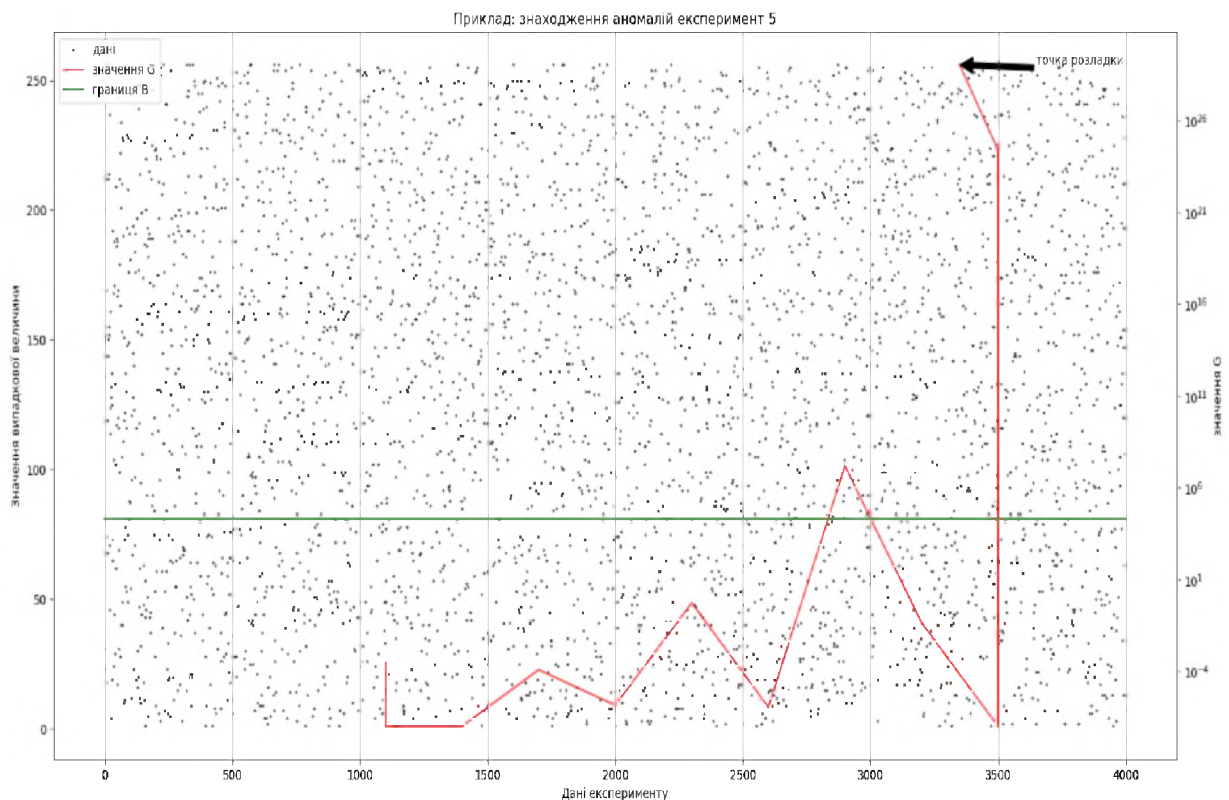
Проведемо 1000 симуляцій та подамо результати дослідження в таблиці 3.7.

Таблиця 3.9. – Сумарні показники зупинки алгоритму

Розладку знайдено вірно	Помилкова тривога	Розладку не знайдено
17	819	164

Як показують дані, лише у 17 випадках із 1000 алгоритм знаходить розладку вірно. Це свідчить, що шифр Camellia є стійким до частотного аналізу.

Розглянемо результат успішного експерименту, в якому розладку знайдено (малюнок 3.4).



Малюнок 3.4. Візуалізація роботи запропонованого алгоритму, стрілкою зображена точка розладки в успішному експерименті

Видозмінимо алгоритм та критерій «Розладку знайдено вірно» таким чином:

1. При знаходженні розладки процедура буде запам'ятовувати момент розладки і не буде зупиняти алгоритм допоки в потік даних є не пустою множиною.
2. Після знаходження розладки алгоритм переходить до набору початкової вибірки.
3. Розладку знайдено вірно – хоча б один з кроків зупинки алгоритму більший або дорівнює теоретичному моменту розладки та менший ніж сума теоретичного моменту розладки та початкових розмірів опорного та експериментального вікон, тобто задовольняє (3.1).

Подамо результати для експерименту 5 з аналогічними початковими умовами для видозміненого алгоритму.

Таблиця 3.10. – Сумарні показники зупинки алгоритму

Розладку знайдено вірно	Розладку не знайдено
257	743

Звернемо увагу на те, що алгоритм знаходить розладку з більшою вірогідністю. А кількість точок зупинки алгоритму не перевищує 5. Можна зробити висновок, що алгоритм, який не зупиняється на першій розладці набагато точніше відрізняє зашифровані дані від випадкової послідовності, але тим самим він зупиняється доволі часто з результатом розладка.

Проведемо ще один експеримент, на меті якого буде ціль відрізнити шифрований текст, зашифрований двома різними алгоритмами на різних ключах. Для дослідження візьмемо алгоритми Triple DES і AES. Для обох будемо використовувати довжину ключа 128 біт і режим CBC. Зафіксуємо довжину першого уривку зашифрованого Triple DES у 3000 символів, а уривку зашифрованого AES у 2000 символів. В даному експерименті будемо

використовувати алгоритм, що при знаходженні розладки зупиняє свою роботу. Проведемо 1000 експериментів. Задамо початкові параметри:

параметри зупинки алгоритму: $\beta = 5 * 10^{-7}$ $\alpha = 5 * 10^{-7}$;

початкові розміри вікон: $M_{init} = 300, N_{init} = 300$;

мінімальні розміри вікон: $M_{min} = 50, N_{min} = 50$.

Таблиця 3.11. – Сумарні показники зупинки алгоритму

Розладку знайдено вірно	Помилкова тривога	Розладку не знайдено
18	848	134

Як бачимо, у порівнянні з результатами поданими в таблиці 3.8. маємо майже однаковий відсоток правильно розпізнаних розладок.

Висновки до розділу 3

У даному розділі докладно описаний алгоритм пошуку розладки. Побудовано параметричний алгоритм пошуку розладки із використанням узагальненого розподілу Пуасона. Описані експерименти для оцінки якості побудованого алгоритму, як на штучно згенерованих даних, так і на реальних даних. Визначено характеристики та рівні якості роботи алгоритму такі як точність визначення моменту розладки алгоритмом, кількість правильно знайдених точок розладки, за допомогою яких оцінені результати експериментів. Проведено експерименти з різними початковими і керуючими параметрами і дано рекомендації щодо їх вибору.

ВИСНОВКИ

У ході даної роботи було оглянуто алгоритми для знаходження розладки в дискретних процесах. Побудовано параметричний алгоритм знаходження розладки, за основу якого взято алгоритм із [4]. Модернізовано початковий алгоритм, введенням параметру кількості послідовних підтверджень наявності розладки для зменшення спрацювань алгоритму з результатом «помилкова тривога» на даних з ймовірними викидами. Також змінено функції зміни розміру вікон вибірок і введено обмеження знизу на розмір вікна. Додано також можливість зміни параметру кроку алгоритму.

Проведено серію експериментів на синтетично згенерованих даних для оцінки якості роботи алгоритму, а саме: дані без розладки, дані без розладки з наявним шумом, потік даних з розладкою. Досліджено здатність алгоритму правильно знайти розладку, в залежності від величини різниці в математичному сподіванні до розладки та після. Дано оцінку точності визначення точки розладки.

Також в рамках оцінки алгоритму проведено серію експериментів на реальних даних: здатність відрізнити осмислений текст від випадково згенерованої рівномірно розподіленої послідовності. Алгоритм відрізняв дані у 99.8% відсотках випадків. Також було перевірено здатність алгоритму відрізнити шифрований текст від випадкової послідовності. Для шифру Camellia було визначено відсоток правильно знайдених моментів розладки на рівні 1.8%. Після даного експерименту було поставлено експеримент для перевірки спроможності відрізнити текст зашифрований за допомогою алгоритму AES від тексту зашифрованого за допомогою алгоритму 3DES і отримано 1.7% правильно знайдених моментів розладки.

Також було досліджено поведінку алгоритму в залежності від задання початкових параметрів і дано рекомендації для їх задання. А саме оптимальними початковими параметрами розміру вікон спостережень обрано $N_{init} = 300, M_{init} = 300, N_{min} = 50, M_{min} = 50$.

ПЕРЕЛІК ПОСИЛАНЬ

1. Basseville M. Detection of Abrupt Changes: Theory and Application / M. Basseville, Igor V. Nikiforov., 1993. – 528 с. – (Prentice Hall). – (Prentice Hall information and system sciences series).
2. Tartakovsky A. Sequential Analysis Hypothesis Testing and Change-point Detection / A. Tartakovsky, M. Basseville, I. Nikiforov., 2014. – 603 с. – (Chapman and Hall/CRC). – (Chapman & Hall/CRC Monographs on Statistics & Applied Probability (Book 136)).
3. Aminikhanghahi S. A Survey of Methods for Time Series Change Point Detection [Електронний ресурс] / S. Aminikhanghahi, D. J. Cook // Knowl Inf Syst.. – 2016. – Режим доступу до ресурсу: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5464762/>.
4. Thatte G. Parametric Methods for Anomaly Detection in Aggregate Traffic (Extended Version) / G. Thatte, U. Mitra, J. S. Heidemann., 2010. – 525 с. – (IEEE/ACM).
5. Ширяев А. Н. Стохастические задачи о разладке / А. Н. Ширяев., 2016. – 392 с. – (МЦНМО).
6. Марчик И. И. Пуассоновская модель: вероятностные и статистические вопросы ее анализа / И. И. Марчик, Г. И. Ивченко., 2015. – 256 с.
7. Bodenham D. A. Adaptive estimation with change detection for streaming data [Електронний ресурс] / Bodenham // Imperial College London. – 2014. – Режим доступу до ресурсу: <http://hdl.handle.net/10044/1/24484>.
8. Bhuiyan Z. A. Collusion Attack Detection in Networked Systems / Z. A. Bhuiyan, J. Wu. – Auckland, New Zealand: IEEE, 2016.
9. Pricing Cryptocurrency options: the case of CRIX and Bitcoin Cathy [Електронний ресурс] / C. Y.Chen, W. K. Härdle, A. J. Hou, W. Wang. – 2018. – Режим доступу до ресурсу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3159130.

10. Salem O. An Efficient Online Anomalies Detection Mechanism for High-speed Networks / O. Salem, S. Vaton, A. Gravey. – Toulouse, France: Dépt. Informatique (Institut Mines-Télécom-Télécom Bretagne-UEB), 2007. – 35 с.
11. Changepoint-based Anomaly Detection in a Core Router System / S.Jin, Z. Zhang, K. Chakrabarty, X. Gu., 2017.
12. Grant J. Bayesian Changepoint Detection in Solar Activity Data [Электронный ресурс] / Grant. – 2014. – Режим доступа до ресурсу: <http://www.lancaster.ac.uk/pg/grantj/mastersdiss.pdf>.

ДОДАТОК А

Лістинг програмної реалізації допоміжних функцій

```

#sample_measures.py
import numpy as np
from math import factorial
import math
import decimal as dc
dc.getcontext().prec = 500
DEFAULT_RATE = 0

def calculateSampleTeta(sMean, sVar, rate=DEFAULT_RATE):
    # for Null hypothesis r = 0
    sampleTeta = (((sMean - rate) ** 3) / max(1, sVar)) ** 0.5
    return sampleTeta

def calculateSampleLambda(sMean, sVar, rate=DEFAULT_RATE):
    # for Null hypothesis r = 0
    sampleLambda = 1 - ((sMean - rate) / max(1, sVar)) ** 0.5
    return sampleLambda

def pdf(sample, sTeta, sLambda, rate=DEFAULT_RATE):
    # function to apply
    pdf = dc.Decimal(1)
    sample_size = sample.size
    result = [dc.Decimal(0) for i in range(sample_size)]
    sample_list = sample.tolist()
    for i in range(sample_size):
        x = float(sample_list[i])
        precalculate_base = sTeta + sLambda * (x - rate)
        precalculate_base2 = x - rate - 1
        result[i] = dc.Decimal(dc.Decimal(sTeta) *
(dc.Decimal(precalculate_base) ** dc.Decimal(precalculate_base2)) * \
                                dc.Decimal(np.exp(-sTeta - sLambda * (x
- rate)))) / dc.Decimal(factorial(x - rate))

```

```

for i in range(sample_size):
    pdf *= result[i]

return pdf

def calculate_statistics(n_sample, m_sample):
    m_var = np.var(m_sample, ddof=1)
    m_mean = np.mean(m_sample)
    teta_zero = calculateSampleTeta(m_mean, m_var)
    lambda_zero = calculateSampleLambda(m_mean, m_var)

    # evaluation for n sample
    n_var = np.var(n_sample, ddof=1)
    n_mean = np.mean(n_sample)
    # rate for evaluating difference
    rate = max(m_sample.append(n_sample).min(), np.math.floor(((1 -
teta_zero) / (1 - lambda_zero)) + m_mean))
    teta_one = calculateSampleTeta(n_mean, n_var, rate)
    lambda_one = calculateSampleLambda(n_mean, n_var, rate)

    fZero = pdf(n_sample, teta_zero, lambda_zero)
    fOne = pdf(n_sample, teta_one, lambda_one, rate)

    G = dc.Decimal(fOne) / dc.Decimal(fZero)
    return G

def char_to_num(character, dictionary):
    if character in dictionary:
        char_index = dictionary.index(character) + 1
        return char_index
    else:
        return -1

def occur_list(text, dictionary):
    occur_dict = dict(zip(dictionary, [0] * len(dictionary)))
    for letter in dictionary:
        number_occur = text.count(letter)
        occur_dict[letter] = number_occur
    return sorted(occur_dict, key=occur_dict.get)

def get_dictionary(text):
    dictionary = set(text)

```

```
        return dictionary
import os

from cryptography.hazmat.primitives import padding
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.ciphers.modes import CBC

def encrypt_camellia(key, text):
    pad = padding.PKCS7(128).padder()
    text = pad.update(text.encode('utf-8')) + pad.finalize()
    algorithm = algorithms.Camellia(key)
    iv = os.urandom(16)
    mode = CBC(iv)
    cipher = Cipher(algorithm, mode=mode, backend=default_backend())
    encryptor = cipher.encryptor()
    ct = encryptor.update(text) + encryptor.finalize()
    return ct

def encrypt_3DES(key, text):
    pad = padding.PKCS7(128).padder()
    text = pad.update(text.encode('utf-8')) + pad.finalize()
    algorithm = algorithms.TripleDES(key)
    iv = os.urandom(8)
    mode = CBC(iv)
    cipher = Cipher(algorithm, mode=mode, backend=default_backend())
    encryptor = cipher.encryptor()
    ct = encryptor.update(text) + encryptor.finalize()
    return ct

def encrypt_AES(key, text):
    pad = padding.PKCS7(128).padder()
    text = pad.update(text.encode('utf-8')) + pad.finalize()
    algorithm = algorithms.AES(key)
    iv = os.urandom(16)
    mode = CBC(iv)
    cipher = Cipher(algorithm, mode=mode, backend=default_backend())
    encryptor = cipher.encryptor()
    ct = encryptor.update(text) + encryptor.finalize()
    return ct

def encrypt_IDEA(key, text):
```

```
pad = padding.PKCS7(128).padder()
text = pad.update(text.encode('utf-8')) + pad.finalize()
algorithm = algorithms.IDEA(key)
iv = os.urandom(8)
mode = CBC(iv)
cipher = Cipher(algorithm, mode=mode, backend=default_backend())
encryptor = cipher.encryptor()
ct = encryptor.update(text) + encryptor.finalize()
return ct

def parse_ciphertext(byte_text):
    str_int_list = [str(i) for i in byte_text]
    return str_int_list

def get_init_sample(src_queue, init_m, init_n, step):
    mSample = pd.Series()
    nSample = pd.Series()
    while len(mSample) < init_m and not src_queue.empty():
        mSample = mSample.append(pd.Series(src_queue.get()))
    while len(nSample) < init_n and not src_queue.empty():
        nSample = nSample.append(pd.Series(src_queue.get()))
    return mSample, nSample, init_m, init_n, init_m + init_n + step
```

ДОДАТОК Б

Лістинг програмної реалізації алгоритму

```

#poisson.py
import pandas as pd
import numpy as np
import decimal as dc

from utilities.sample_measures import calculate_statistics

# constants
DEFAULT_RATE = 0

def execute_big_window_poisson(srcQueue, A, B, init_n, init_m, min_n,
min_m, max_warn):
    stop = False
    warn = 0
    step = 0
    G_step = []
    max_g = 0.0
    mSample, nSample, M, N, step = get_init_sample(srcQueue, init_m,
init_n, step)
    while (not stop) and (not srcQueue.empty()):
        G = calculate_statistics(nSample, mSample)
        if G > max_g:
            max_g = G
        if G >= B:
            warn += 1
            if warn == max_warn:
                stop = True
                step -= N
            N = max(int(np.ceil(N / 2)), min_n)
            nSample = nSample[:N]

        elif G <= A and M > min_m:

            warn = max(warn - 1, 0)
            M = max(int(np.ceil(M / 2)), min_m)
            mSample = mSample[-M:]
    else:

```

```
warn = max(warn - 1, 0)
mSample = mSample.append(nSample)
mSample = mSample[-M:]
nSample = pd.Series()

while len(nSample) < N and (not srcQueue.empty()):
    nSample =
nSample.append(pd.Series(srcQueue.get(timeout=0.001)))
    step += 1
    G_step.append((step, G.quantize(dc.Decimal('.000000001'),
rounding=dc.ROUND_UP)))

return step, (A, max_g, B), (M, N), G_step
```

ДОДАТОК В

Лістинг програмної реалізації експерименту

```

def stand_approbation_3():
    # constants for launch
    NUMBER_OF_EXPERIMENTS = 1000
    DIFFERENCE_LAMBDA = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4,
0.45, 0.5]
    LAMBDA_SAMPLE = [random.randint(10, 250) for i in
range(NUMBER_OF_EXPERIMENTS)]
    print('started with params')
    print(LAMBDA_SAMPLE)
    results = []
    table_sum = []
    for i in range(len(DIFFERENCE_LAMBDA)):
        stopped_falsep = 0
        stopped_falsen = 0
        stopped_true = 0
        for j in range(NUMBER_OF_EXPERIMENTS):
            print('experiment:' + str(j))
            testQ = queue.Queue()
            number_before_anomaly = 2000
            number_after_anomaly = 700
            for k in range(number_before_anomaly):
                testQ.put(np.random.poisson(LAMBDA_SAMPLE[j]))
            for k in range(number_after_anomaly):
                testQ.put(np.random.poisson(LAMBDA_SAMPLE[j] +
(LAMBDA_SAMPLE[j] * DIFFERENCE_LAMBDA[i])))
            stop_step, borders, sample_sizes, G_step =
execute_big_window_poisson(testQ, A, B, INIT_N, INIT_M, MIN_N, MIN_M, 2)
            if SAMPLE_TEXT_SIZE - sample_sizes[1] > stop_step:
                stopped_right = 'False P'
                stopped_falsep += 1
            elif stop_step > (SAMPLE_TEXT_SIZE + sample_sizes[1] +
sample_sizes[0]):
                stopped_right = 'False N'
                stopped_falsen += 1
            else:
                stopped_right = 'True'
                stopped_true += 1

```



```

        results.append([LAMBDA_SAMPLE[j], LAMBDA_SAMPLE[j] *
DIFFERENCE_LAMBDA[i], DIFFERENCE_LAMBDA[i],
                        number_before_anomaly,
number_after_anomaly, stop_step, stopped_right, sample_sizes])
        table_sum.append([DIFFERENCE_LAMBDA[i], stopped_true,
stopped_falsep, stopped_falsen])
        with open("./output/output3.csv", "w") as f:
            writer = csv.writer(f)
            writer.writerow(['LAMBDA_SAMPLE[j]', 'LAMBDA_SAMPLE[j] *
DIFFERENCE_LAMBDA[i]', 'DIFFERENCE_LAMBDA[i]',
                            'number_before_anomaly',
'number_after_anomaly', 'stop_step', 'stopped_right', 'sample_sizes'])
            writer.writerows(results)
        with open("./output/table_sum3.csv", "w") as f:
            writer = csv.writer(f)
            writer.writerow(['DIFFERENCE_LAMBDA[i] %', 'stopped_true',
'stopped_falsep', 'stopped_falsen'])
            writer.writerows(table_sum)
        print('finished')

```