



Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря

Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

## Комп’ютерний практикум №4

**Комп’ютерне моделювання**

**дискретно-подійних систем**

Виконав студент групи ІІІ-21: Плугатирьов Д.В.		Перевірів:
		Стеценко І.В.
		Дата:
		Оцінка:

## Завдання

1. Розробити модель масового обслуговування, яка складається з  $N$  систем масового обслуговування. Число  $N$  є параметром моделі. Кількість подій в моделі оцінюється числом  $N+1$ . **20 балів.**
2. Виконати експериментальну оцінку складності алгоритму імітації мережі масового обслуговування. Для цього виконайте серію експериментів, в якій спостерігається збільшення часу обчислення алгоритму імітації при збільшенні кількості подій в моделі. **40 балів.**
3. Виконати теоретичну оцінку складності побудованого алгоритму імітації. **30 балів.**
4. Повторіть експеримент при зміні структури мережі масового обслуговування. **10 балів.**

### Завдання 1. Розробка моделі масового обслуговування

Було розроблено модель, яка складається з ланцюжка з  $N$  систем масового обслуговування (СМО). Кількість вузлів  $N$  є вхідним параметром функції `create_model`.

У функції `create_model` використовується цикл для динамічного створення вузлів:

```
# N є параметром num_nodes
def create_model(num_nodes: int, ...):
    # ...
    # Цикл створює N вузлів типу SystemQueueingNode
    for idx in range(num_nodes):
        next_node = SystemQueueingNode(...)
        # З'єднання попереднього вузла з наступним
        prev_node.set_next_node(node_connection_logic)
```

Рисунок 1 - Цикл для динамічного створення вузлів

Це дозволяє будувати мережі довільного розміру (у даній роботі використано  $N=20$ ).

## Завдання 2. Експериментальна оцінка складності

Для оцінки складності було проведено серію з 10 експериментів зі збільшенням модельного часу симуляції (*simulation\_time*) від 1000 до 10000 умовних одиниць. Це призвело до лінійного зростання кількості подій у системі.

Час виконання вимірювався за допомогою системного таймера *time.perf\_counter()*:

```
start_time = time.perf_counter()
model.simulate(end_time=simulation_time, ...)
end_time = time.perf_counter()
measured_time = end_time - start_time # Реальний час виконання (Синя лінія)
```

Рисунок 2 – Вимірювання часу за допомогою системного таймера

На отриманих графіках (синя лінія *measured*) спостерігається чітка лінійна залежність часу виконання від кількості подій, що відповідає очікуванням для алгоритмів дискретно-подієвого моделювання.

## Завдання 3. Теоретична оцінка складності

В основі алгоритму імітації лежить структура даних "Календар подій" (Priority Queue), складність вставки в яку становить  $O(\log Q)$ , де  $Q$  — поточна довжина черги подій.

Для кожного завдання в історії симуляції розраховувалась теоретична складність:

$$T_{pred} = N_{events} \times 2(1 + mean(\log_2(Q_{size}))) \times C,$$

де  $C(\text{ELEMENTARY\_OPERATION\_TIME})$  — емпірична константа часу однієї операції (підібрана як  $7e-5$  сек).

Для збору статистики було створено спеціальний клас `SystemQueueingMetrics`, який записує розмір черги при кожному додаванні завдання.

```
def num_elementary_operations(model):  
    # ... збір історії ...  
    # Розрахунок логарифмічної складності для кожної події  
    num_insert_operations = [math.log2(num_tasks) for num_tasks in num_tasks_history]  
    return 2 * (1 + mean(num_insert_operations))
```

Рисунок 3 - Розрахунок теоретичної оцінки складності алгоритму

На графіках червона лінія (predicted) майже ідеально збігається з синьою (measured) за кутом нахилу. Це підтверджує коректність теоретичної моделі складності  $O(N \log N)$ .

#### Завдання 4. Дослідження зміни структури мережі

Експеримент було повторено для трьох різних топологій мережі, що регулювалися ймовірністю зворотного зв'язку (`prev_proba`). Це додає цикли в мережу, збільшуючи навантаження та кількість подій.

##### Результат 1: Лінійна мережа (Probability = 0.0)

**Опис:** Завдання проходять послідовно від вузла до вузла без повернень.

**Аналіз:** Графік показує стабільне лінійне зростання. Максимальний час виконання ~65-75 секунд.

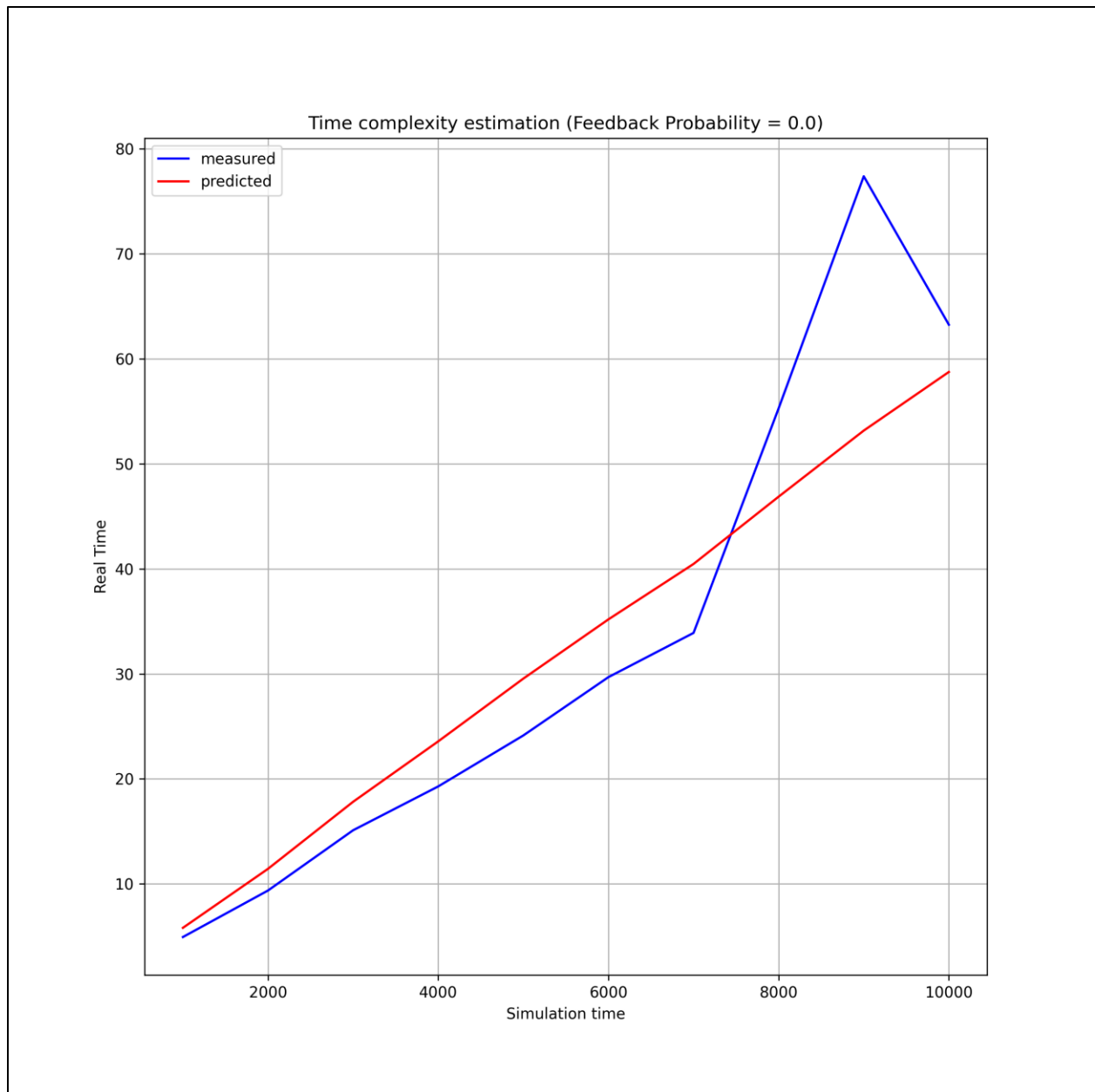


Рисунок 4 – Графік оцінки часової складності із ймовірністю повернення 0.0

## Результат 2: Мережа з малим зворотним зв'язком (Probability = 0.1)

**Опис:** 10% завдань повертаються на попередній етап обробки.

**Аналіз:** Час виконання дещо зростає, лінії стають більш гладкими. Теоретична оцінка продовжує точно описувати реальний час.

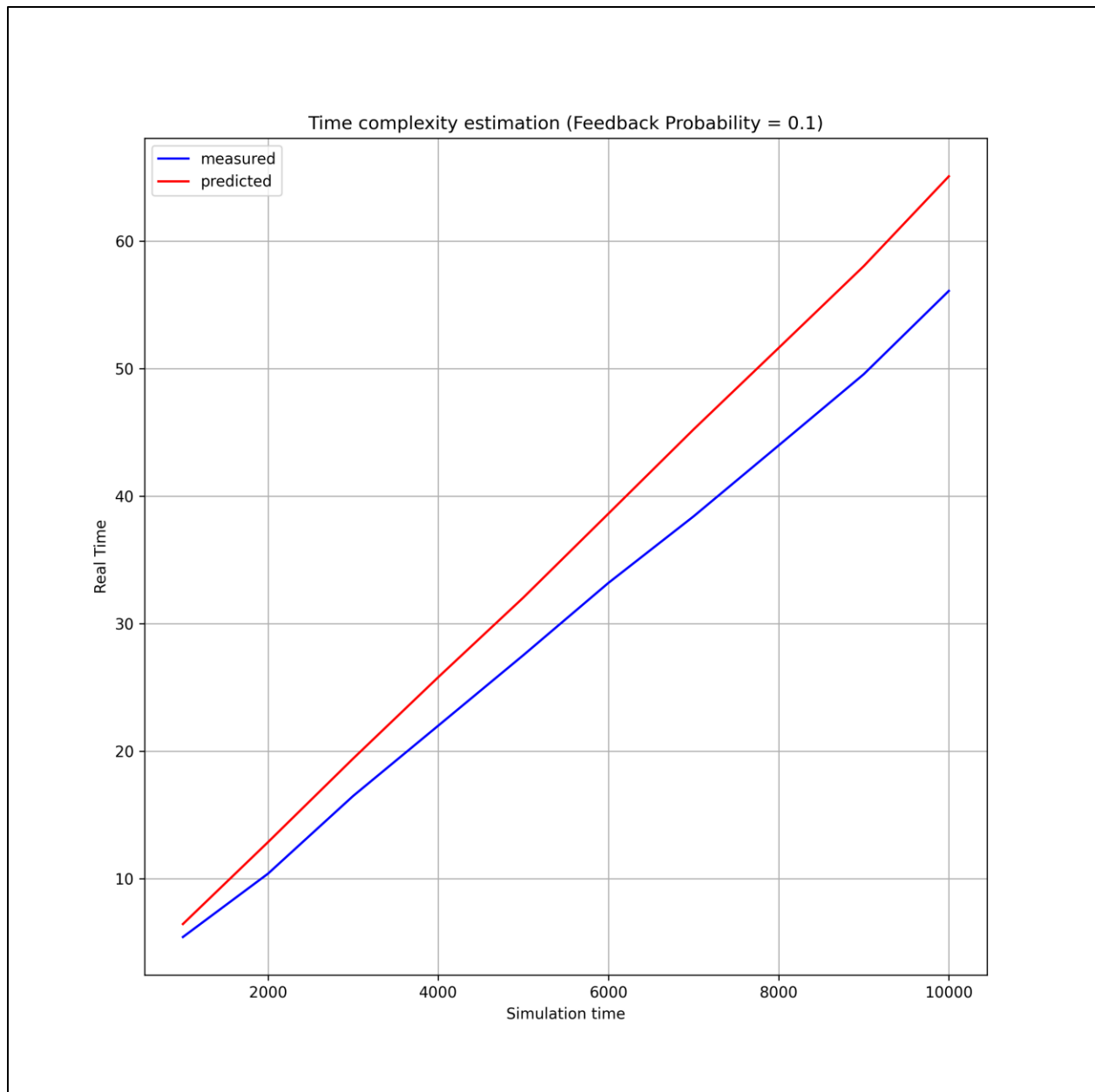


Рисунок 5 – Графік оцінки часової складності із ймовірністю повернення 0.1

### Результат 3: Високонавантажена циклічна мережа (Probability = 0.5)

**Опис:** 50% завдань повертаються назад. Це створює "ефект снігової кулі" (кількість подій різко зростає).

**Аналіз:** Максимальний час виконання стрибає до **>100 секунд**.

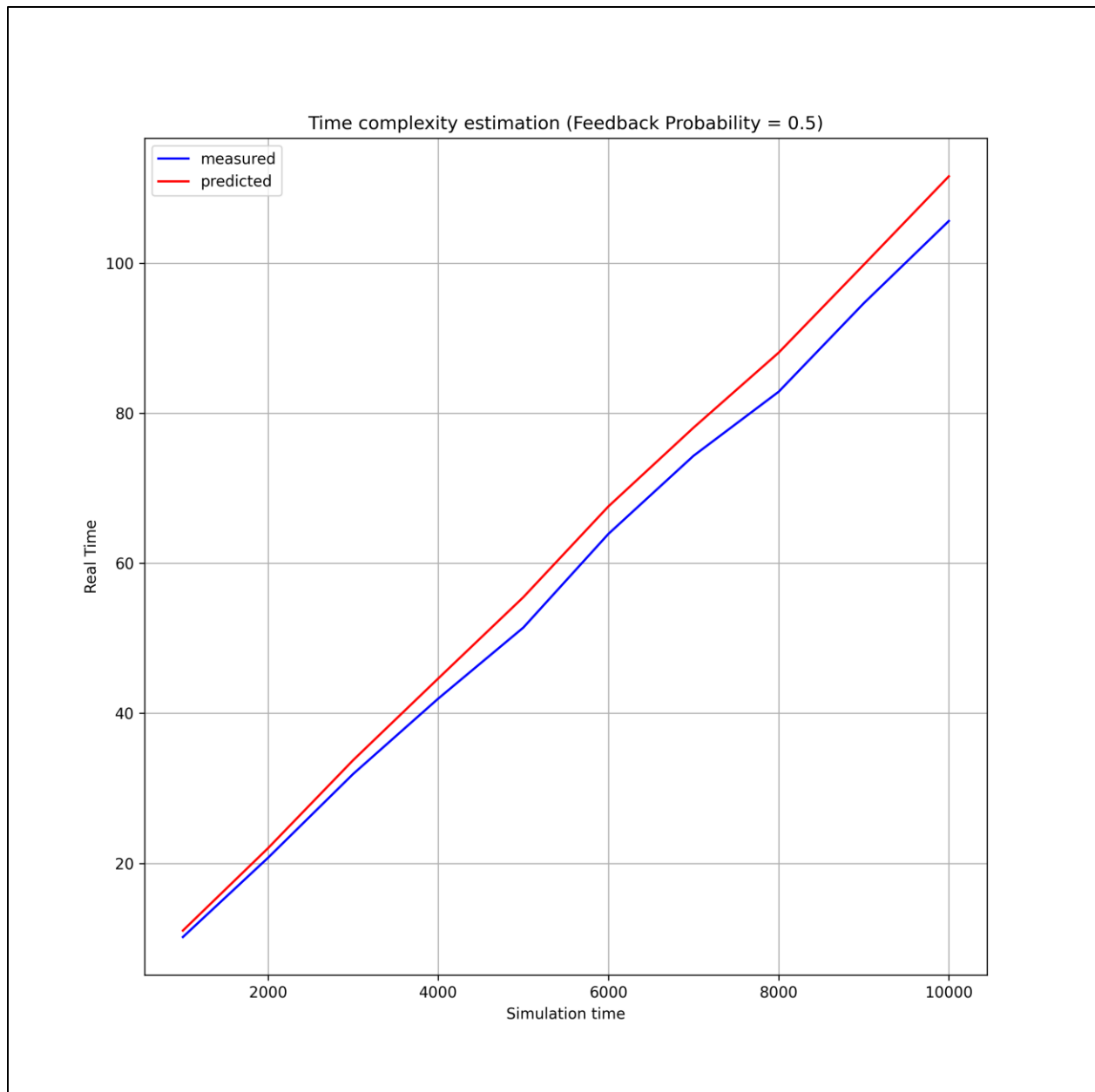


Рисунок 6 – Графік оцінки часової складності із ймовірністю повернення 0.5

Зміна структури мережі (додавання циклів) суттєво впливає на час симуляції через збільшення кількості подій, проте алгоритмічна складність залишається передбачуваною (червона і синя лінії паралельні).

## Висновок

У ході роботи було розроблено параметричну модель СМО та проведено комплексний аналіз її продуктивності. Експериментально підтверджено, що час виконання симуляції лінійно залежить від кількості модельних подій. Теоретична модель, що базується на логарифмічній складності операцій з календарем подій, продемонструвала високу точність прогнозування реального часу виконання для різних топологій мережі.