РОЗРОБКА МОДЕЛІ ОЦІНКИ ЧАСОВИХ ВИТРАТ ІТ ПРОЕКТУ НА ОСНОВІ ТИМЧАСОВИХ ГРАФОВИХ МЕРЕЖ

Толстолузький Є.Д., Вишняк М.Ю.

Кафедра системотехніки.

Харківський національний університет радіоелектроніки,

Україна.

E-mail:

yevhen.tolstoluzkyi@nure.ua

mykhailo.vyshniak@nure.ua

Abstract

Accurate estimation of task duration, lead time, and cycle time is critical for planning and risk control in software projects. Traditional methods and tabular ML overlook dynamic interactions among tasks, people, and code, while static GNNs lose temporal evolution. We model projects as dynamic graphs and apply Temporal Graph Networks (TGN), which process event streams and update node states in real time via messages, node memory, and temporal encoding. Using a minimal event schema (src, dst, timestamp, type, payload) and basic node attributes, TGN delivers early, continuously refreshed predictions and calibrated uncertainty (P50/P90). Evaluation with strictly time-ordered splits and rolling backtests shows improved accuracy and planning utility. We outline deployment as a streaming service and discuss common pitfalls – temporal leakage, cold-start, and process drift – along with practical safeguards and extensions for explainability and hierarchical planning.

Ефективна оцінка часових витрат (тривалість задач, найкращий час, циклічний час) є критичною для планування роботи, управління ризиками та виконання цілей в IT-проєктах. Традиційні методи та табличні моделі машинного навчання часто ігнорують динамічні взаємодії між задачами, виконавцями та ресурсами. Сучасні підходи на основі графових нейромереж враховують структуру залежностей, але у статичній постановці втрачають часову еволюцію процесів. Тимчасові графові мережі (Temporal Graph Networks, TGN) забезпечують подієве оновлення станів вузлів і дозволяють формувати ранні та оновлювані прогнози під впливом поточних подій.

Temporal Graph Networks – це сімейство моделей для динамічних графів, де зв'язки й атрибути змінюються у часі. На відміну від статичних графових нейронних мереж (GNN), TGN обробляє потік подій (interaction stream) і оновлює подання вузлів у реальному часі. TGN визначається трьома ключовими компонентами:

- Повідомлення. Для кожної події ми готуємо компактний вектор, який враховує поточний стан джерела/цілі, сам тип події й «свіжість» (інтервал із попередньою взаємодією). Це дозволяє моделі відчути різницю між давньою й щойно сталою подією.
- Пам'ять вузлів. Вузол зберігає стан, який оновлюється, коли на нього проходять повідомлення. Починати краще з оновлення і простої агрегації (середнє), це дає стабільність і менше гіперпараметрів.
- Прогноз. Нескладний шар, що перетворює актуальний стан задачі на оцінку тривалості. За потреби додаємо короткий контекст – узагальнення з найближчих пов'язаних вузлів (виконавець, «гарячий» модуль, сусідні задачі). На практиці цього достатньо, щоб уже обійти табличні моделі.

Завдяки цьому модель взаємодіє з графом не у статиці, а відстежує його еволюцію [1].

TGN доречний там, де важлива послідовність взаємодій: рекомендаційні системи, соціальні мережі, виявлення шахрайства, графи знань, інженерні процеси на кшталт Jira/Git/CI. Його сила – у ранніх і постійно оновлюваних прогнозах: модель уточнює оцінку щоразу, коли змінюється стан – з'явився додатковий розробник, зросло навантаження на ключового розробника, тощо. У підсумку зазвичай отримуємо кращу точність, ніж у табличних підходів машинного навчання (ML) чи статичних GNN, адже враховано не лише структуру, а й ритм подій. Додатково нескладно отримати інтервали невизначеності (P50/P90) і контролювати їхнє покриття, що покращує планування.

Події мають містити такі поля: джерело (відправник), одержувач, мітка часу, тип, вміст події, а також потрібен довідник вузлів із базовими атрибутами. Джерело й одержувач указують на вузли (розробник → задача, задача → репозиторій, сервіс → сервіс); час потрібен для порядку та інтервалів; тип події – для змісту (створили запис про проблему, змінили статус, відкрили запит на злиття (PR), злили PR, почали перевірку коду, запустили фінальну версію, тощо). У вмісті подій зберігаються кілька компактних числових/категоріальних деталей (наприклад, скільки рядків змінено або з якого статусу в який перейшли). Атрибути вузлів – окремі дані: для задач – тип/пріоритет/компонент; для людей – роль/належність до команди; для модулів – підсистема. Усе – в форматі всесвітнього координованого часу (UTC); ідентифікатори мають бути стабільні та уніфіковані [2].

На рис. 1 зображено модель оцінки часових витрат IT-проєкту на основі тимчасових графових мереж у вигляді схеми.

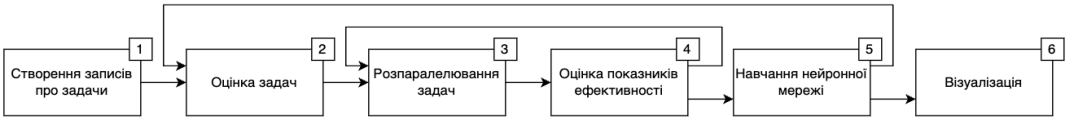

Рис. 1. Модель оцінки загального часу

1. Створення та нормалізація записів про задачі: створюються довідник вузлів, подій та залежностей. На виході: уніфіковані ідентифікатори й події, готові до тренування.
2. Оцінка задач: отримуємо поточні прогнози тривалості/залишкового часу для кожної задачі. TGN тримає пам'ять вузлів і оновлює її з кожною подією. Прогноз дає центральну оцінку (P50) і «песимістичну» межу (P90) з урахуванням близького контексту (виконавець, модуль, сусідні задачі). На виході: таблиця прогнозів на загальну дошку завдань.
3. Розпаралелювання задач: плануємо паралельність/черги, щоб зменшити вузькі місця. Використовуємо P50/P90 і «фактори впливу» як сигнали при розкладці: не ставимо в один слот задачі, які не взаємопов'язанні, та мають великий вплив або ризики. На виході: рекомендований план паралелізації з позначками ризику.
4. Оцінка показників ефективності: вимірюємо якість прогнозів і процесу. Використовуємо модельні та бізнес метрики: MAE, RMSE, R^2, P50/P90 [3], ширина інтервалів, стабільність у часі (rolling-backtest). На виході: отримуємо звіт за період, базуючись на якому можна змінити розпаралелювання задач задля кращої оцінки.
5. Навчання нейронної мережі: тренуємо та перенавчаємо модель тимчасових графових мереж із часовим поділом даних (навчання → перевірка даних → тестування), калібруємо інтервали. Послідовність обробки: повідомлення з урахуванням проміжку часу (Δt) → узагальнення → оновлення пам'яті вузла → прогноз. На виході: перевірена версія моделі та калібровані інтервали, готові до роботи. Якщо потрібне донавчання, перекалібрування або перегляд ознак/подій, повертаємося до кроку 2.
6. Візуалізація: показуємо живі оцінки і тренди так, щоб цим реально користувались. На виході: прозора картина для менеджера проєкту та технічного керівника – де ми зараз, де є проблеми і що слід змінити.

Дуже важливо розділяти дані за часом: тренування на історії до контрольної дати, перевірка – на новішому інтервалі, тест – на ще більш новому. Так ми моделюємо реальний сценарій «навчилися вчора – прогнозуємо завтра». Будь-які агрегати (середні, частоти, «характеристики модуля») також мають обмежуватися знанням до моменту поділу за часом; інакше виникає прихований витік

інформації. Добре працює перевірка з ковзним початком: кілька послідовних вікон оцінювання, щоб побачити, як тримається якість у різні періоди (випуски, свята, реорганізації).. [4]

На рис. 2 зображено як зменшується похибка при оцінці часу під час навчання.

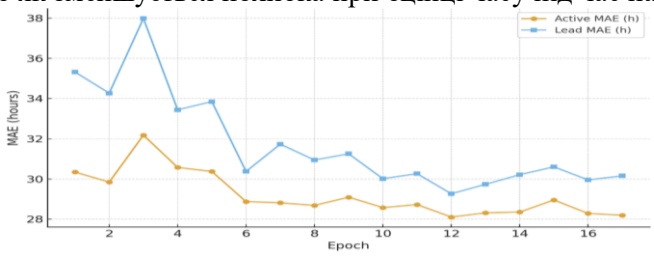

Рис. 2. Зміна кількості помилок під час циклів навчання

Типові ризики, які можуть бути при роботі з TGN:

- Часовий витік даних. Найпоширеніша пастка – непомітно використовувати майбутні дані. Як запобігти: усі розрахункові показники обрізати на потрібну дату; перевіряти процеси обробки даних спеціальними контрольними перевірками.
  - Початковий стан (без історії) та розріджені події. Для нових задач або модулів невизначеність вища. Можливе вирішення: використовувати прості початкові припущення (історичні медіани за типом/компонентом) і задавати ширші інтервали оцінки.
  - Дрейф процесів. Нові політики рев'ю, реорганізації, зміни ритму релізів змінюють розподіли. Можливе вирішення: періодичне донавчання або хоча б перекалібрування, плюс загальна таблиця із метриками в часі.
  - Інфраструктурна складність. Потокова обробка подій і підтримання стану пам'яті потребують трохи більшої дисципліни, ніж раз на місяць запускати звичайну табличну модель градієнтного бустингуПочинати варто з простого офлайн-режиму (регулярні перерахунки), а вже потім переходити на майже реальний час.[5]

  Для подальшого покращення:
  - Пояснюваність. Показувати, які події або які «гілки» взаємодій вплинули на оцінку — це зменшує суперечки й пришвидшує реакцію команди.
  - Ієрархія задач. Увімкнути рівні «заплановані завдання→ завдання→ задача», щоб прогнозувати не тільки окремі завдання, а й великі блоки.
  - Аналіз сценаріїв («що, якщо»). Дає змогу відповідати на практичні запитання: що буде, якщо перерозподілити перевірку коду, скоротити чергу запитів на злиття, перенести частину робіт між модулями.
  - Активне навчання. Запитувати увагу експерта рівно там, де невизначеність найбільша, щоб не перевантажувати людей.

Представлення IT-проєкту як динамічного графа та застосування TGN дає досить точні оцінки тривалості задач. Модель природно оновлює прогнози під впливом нових подій і надає інтервали невизначеності (P50/P90), що робить планування більш керованим і прозорим. Критично дотримуватися часових розрізів і стежити за дрейфом процесів; для початкового стану без історії слід використовувати прості початкові припущення та проводити періодичне перекалібрування. Практичний шлях упровадження – мінімальна подієва схема, базова архітектура тимчасових графових мереж, панель показників і поступовий перехід до потокової обробки подій.

Література

1. Rossi E., Chamberlain B., Frasca F., Eynard D., Monti F., Bronstein M. Temporal Graph Networks for Deep Learning on Dynamic Graphs // arXiv preprint arXiv:2006.10637. 2020.
2. Xu D., Ruan C., Korpeoglu E., Kumar S., Achan K. Inductive Representation Learning on Temporal Graphs (TGAT) // Proc. of ICLR. 2020.
3. Hyndman R.J., Athanasopoulos G. Forecasting: Principles and Practice (3rd ed.) // OTexts. 2021. URL: otexts.com/fpp3.
4. Kazemi S.M., Goel R., Jain K., Kobyzev I., Sethi A., Forsyth P., Poupart P. Representation Learning for Dynamic Graphs: A Survey // Journal of Machine Learning Research. 2020.
5. Skarding J., Gabrys B., Musial K. Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey // IEEE Access. 2021.