The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; in the decision to publish the results.
УДК 004.023
DOI: https://doi.org/10.17721/1812-5409.2025/1.22

Олександр ФРАНКІВ, асп.
ORCID ID: 0009-0005-5264-358X
e-mail: o.frankiv@ukma.edu.ua
Національний університет "Києво-Могилянська академія", Київ, Україна

МАШИННЕ НАВЧАННЯ В ПОКРАЩЕННІ ВІЗУАЛІЗАЦІЇ ПРОСТОРОВОЇ МОДЕЛІ АРХІТЕКТУРИ ПРОГРАМИ

Візуальна естетичність та обчислювальна ефективність є однаково важливими аспектами у контексті створення якісного представлення даних для подальшого аналізу. Гармонійне поєднання цих характеристик забезпечує не лише зручність сприйняття, але й оптимізує процеси оброблення даних, що є критично важливим у сучасному програмному забезпеченні.

У пропонованій статті розкрито новий комбінований підхід до розміщення графів у просторі. Особливу увагу приділено просторовим моделям архітектури програмного забезпечення, що є ключовим інструментом для візуалізації складних взаємозв'язків між компонентами. Застосування графових нейронних мереж як спеціалізованої евристики – центральний елемент зазначеного підходу. Завдяки використанню методів машинного навчання, запропонований підхід дає змогу поліпшити результати візуалізації та підвищити обчислювальну ефективність.

Використання графової нейронної мережі забезпечує адаптивність і здатність моделі враховувати специфічні особливості графу. Це, у комбінації із силовим алгоритмом, сприяє збереженню високого рівня естетичності візуального представлення без значного збільшення ресурсних витрат. Отже, новий метод пропонує практичне рішення для ефективного поєднання візуальної естетики й обчислювальної ефективності, що є важливим кроком у вдосконаленні аналізу просторових моделей архітектури програмного забезпечення.

Ключові слова: ***нейронна мережа, машинне навчання, граф, графова нейронна мережа, графовий згортковий оператор (GCNConv), архітектура програмного забезпечення, автоматична візуалізація архітектури програмного забезпечення, часова складність алгоритму.***

Класифікація відповідно до AMS 2020: QA76.9.A43.

Вступ

Забезпечення якості продукту є важливим аспектом виробництва програмного забезпечення. Нині існує велика кількість інструментів, що дають змогу оптимізувати певні частини цього процесу. Зважаючи на стрімкий розвиток галузі, створення нових і розвиток наявних інструментів залишається актуальним.

Серед великої кількості рішень, що дають змогу автоматизувати аналіз якості програмного коду, важливим залишається напрям аналізу саме архітектурних рішень. Це, зокрема, передбачає створення інструментів, що не вимагають значних додаткових затрат для створення спеціальної документації. Раніше, у роботах (Франків, 2023; Франків, & Глибовець, 2024) описано розроблений нами програмний комплекс для автоматичної візуалізації архітектури програмного модуля, спираючись лише на вихідний код. Аналізатор створює візуальну модель архітектури програми у вигляді графа, вузлами якого позначено структурні елементи програми, а ребрами – зв'язки між ними. За допомогою візуалізатора комплекс створює візуально естетичне розміщення цього графа у тривимірному просторі. За допомогою спеціального iOS-застосунку цей граф можна відобразити в доповненій реальності, що забезпечує природне сприйняття структурованої інформації.

Проблема автоматичного створення прийнятного для сприйняття розміщення графа в просторі є складною з огляду на те, що запропонований метод має працювати для графів різних форм і розмірів, забезпечувати відповідність загальноприйнятим критеріям естетичності й обраховуватись із прийнятною швидкістю. Підвищення швидкодії візуалізації є особливо затребуваним у контексті створення розміщень на пристроях з обмеженими ресурсами, зокрема й мобільних.

У пропонованій статті розглянуто спосіб застосування алгоритмів машинного навчання для створення спеціалізованої евристики, що дає змогу поліпшити швидкодію наявного рішення без негативного впливу на рівень естетичності розміщення графа в просторі. Описано підхід до створення цієї евристики в загальному випадку, що відкриває можливість створення аналогів для різних технологій аналізованих проєктів. Досліджено доцільність використання спеціалізованих евристик для графів різних розмірів.

1. Розміщення графів у просторі

Натепер задача відображення графів у просторі досліджена добре. Існує велика кількість способів представлення, що відповідають різним запитам користувача. Варто зауважити, що залежно від кількості, а головне – природи даних, поданих у формі графа, різні види візуалізації мають свої переваги й недоліки.

Оскільки ми розглядаємо задачу в контексті створення подання для аналізу архітектури програми, важливо визначити основні вимоги до застосованого підходу. По-перше, ваги ребер відіграють ключову роль у відображенні міри взаємної залежності компонентів архітектури, а отже, мають обов'язково враховуватись. По-друге, зважаючи на розмір програм і, як наслідок, потенційно великий розмір графів, бажано, щоб обраний підхід мав невелику часову складність. По-третє, критичною є висока естетичність подання, що допоможе не лише відобразити кластери вузлів, але й зберегти відносні відстані між ними

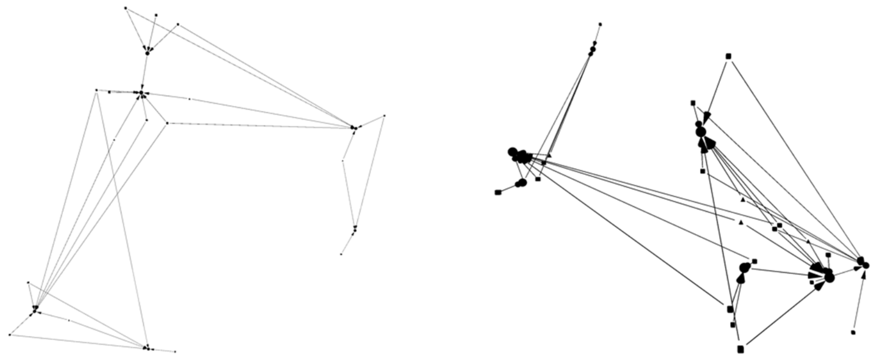
З огляду на конкретні потреби, серед класичних методів візуалізації очевидним є вибір на користь силового алгоритму, що дає змогу досягнути порівняно кращого сприйняття інформації користувачем (Pohl, Schmitt, & Diehl, 2009). Важливою його перевагою є можливість працювати зі зваженими графами.

За силовим алгоритмом (Fruchterman, & Reingold, 1991) граф варто розглядати як фізичну модель, у якій вузлам графа відповідають позитивні точкові заряди, що відштовхуються один від одного за законом Кулона, а ребрам –

пружини, що їх пов'язують і створюють силу протидії за законом Гука. У класичному варіанті на початковому етапі заряди розміщуються в одній точці, або ж частіше випадковим чином, а далі в циклі відбувається постійний обрахунок нового положення системи відповідно до застосування всіх наявних у системі сил.

Серед недоліків силового алгоритму варто згадати високу часову складність обчислень $O(N^2)$ для однієї ітерації, що зумовлена високою часовою складністю обходу графа. З іншого боку, через використання випадкових евристик кількість ітерацій зовнішнього циклу не є фіксованою, що, зі свого боку, призводить до нестабільності та проблеми локальних мінімумів.

Існують оптимальніші аналоги класичного алгоритму. Для обрахування сил взаємодії для великої кількості тіл, наприклад, пропонують розглядати простір як дерево (Barnes, & Hut, 1986). Двовимірний простір ділять на чотири частини (тривимірний на вісім) рекурсивно до моменту, коли в кожному підпросторі не залишиться лише один вузол. Тоді обхід графа зводиться до обходу дерева і часова складність однієї ітерації становить $O(N * logN)$. Однак, оскільки зміщення кожного вузла в алгоритмі можливе лише в межах його власного підпростору, то з'являється висока залежність від першопочаткового розміщення вузлів. Це може негативно впливати на естетичність подання (див. рис. 1) і робить цей метод малопридатним для застосування в контексті нашої задачі.



а б

Рис. 1. Результат розміщення за допомогою класичного силового алгоритму (а) та алгоритму оптимізованого обрахування взаємодій (б)

У дослідженнях (Walshaw, 2001; Didimo, Liotta, & Romeo, 2011) автори пропонують зваженіші підходи до початкового розміщення графа в просторі, однак ці методи мають обмеження і не є універсальними, оскільки складно передбачити властивості графів, побудованих з реальних даних.

Абсолютно іншим підходом до задачі візуалізації графів є використання алгоритмів машинного навчання. Головні виклики в цій сфері – забезпечення високої естетичності розміщення, ефективність й універсальність (Wang et al., 2024).

Серед більш універсальних методів можна вирізнити node2vec (Grover, & Leskovec, 2016), force2vec (Rahman, Sujon, & Azad, 2020) та DeepDrawing (Wang et al., 2020). Кожен із наведених методів дає змогу отримати естетичні представлення графів, що все ж не будуть абсолютно відповідати результату роботи силового алгоритму і, зокрема, вимогам нашої задачі.

Складність застосування node2vec полягає в тому, що в його основі лежить генерація випадкових проходів по графу. Для отримання високої якості візуалізації необхідне тонке налаштування параметрів ймовірності повернення та ймовірності розширення, що складно зробити для графів, які потенційно можуть бути абсолютно різними за структурою.

Застосування force2vec до нашої задачі не вирішує проблеми залежності від початкового розміщення графа. Окрім того, класичний метод не працює з вагами ребер.

DeepDrawing передбачає роботу із графами лише фіксованого розміру, що робить цей метод не достатньо універсальним для наших потреб.

З огляду на описані вище вимоги до візуалізації й аналіз наявних, зокрема комбінованих (Huang et al., 2024), підходів було ухвалено рішення запропонувати універсальний метод, що передбачає неявне задання спеціалізованої евристики на основі аналізу реальних даних та її застосування для початкового розміщення в контексті класичного силового алгоритму. Це дасть змогу забезпечити високу якість візуалізації, а також підвищити стабільність алгоритму, на відміну від використання випадкових евристик, та потенційно зменшити кількість "дорогих" ітерацій зовнішнього циклу.

2. Використання машинного навчання для створення спеціалізованої евристики

Створення спеціалізованої евристики для розміщення графа у просторі в явному вигляді є складною задачею. З одного боку, детальний аналіз графа з метою визначення особливостей його топології передбачає обхід графа, що, власне, і є слабким місцем класичного силового алгоритму. З іншого боку, з огляду на те, що реальна робота візуалізатора передбачає розміщення графів довільної конфігурації (зумовлено довільною природою архітектури аналізованих програм), зробити явно задану евристику універсальною досить складно.

Натепер використання машинного навчання й нейронних мереж є перспективним способом неявного програмування знань про домен, отриманих унаслідок аналізу реальних даних. Ми висунули припущення про те, що за умови використання спеціалізованого апаратного забезпечення завдяки використанню нейронної мережі, побудованої за допомогою аналізу природних даних, можна значно зменшити час виконання силового алгоритму. Це можливо за одночасного виконання двох умов:

- Використання отриманої евристики зменшить кількість необхідних ітерацій обходу графа на деяку вагому кількість ΔI.

• Час, необхідний на обчислення початкового розміщення за допомогою нейронної мережі, буде меншим, ніж потрібний час на виконання ΔI ітерацій обходу графа.

Для перевірки цього припущення було підготовано колекцію даних та запропоновано архітектуру простої нейронної мережі. Ми провели тренування для отримання моделей і порівняння продуктивності виконання силового алгоритму з випадковою евристикою та новою, спеціалізованою.

3. Збір та підготовка даних

Евристика, що відображає певний досвід у розміщенні графа архітектури програми, є чутливою до конкретних технологій, що використовуються під час її побудови. Саме тому автоматизація збору даних є важливою. Вона дає змогу легко отримати нові дані для аналізу в майбутньому, якщо спосіб побудови програм конкретного призначення зміниться з появою нових технологій та підходів.

Окрім того, ми автоматизували збір проєктів зі сховища програмного забезпечення з відкритим вихідним кодом GitHub. Оскільки оригінальна імплементація нашого комплексу дає змогу аналізувати вихідний код програм, написаних мовою Swift, то і в пропонованому дослідженні ми перевіряли наше припущення на них.

Процес побудови графів архітектури ми також автоматизували шляхом створення спеціальної утиліти аналізу файлів проєкту, що автоматично готує необхідні параметри та формує команду аналізатору. Також було вдосконалено наявні інтерфейси аналізатора та візуалізатора з наголосом на тому, щоб внаслідок його роботи можна було в явному вигляді отримати ваги ребер, адже вони безпосередньо впливають на результат розміщення у просторі.

У контексті підготовки візуальних представлень для навчання важливою проблемою є вибір умови зупинки силового алгоритму. До найчастіше використовуваних належить зупинка за умови досягнення максимальної кількості ітерацій, порогу максимального зміщення всіх вузлів, або порогу потенціальної енергії системи (Fruchterman, & Reingold, 1991).

Очевидною перевагою зупинки за досягненням максимальної кількості ітерацій є визначеність часу роботи алгоритму. З іншого боку, такий підхід має два серйозні недоліки. По-перше, стала кількість ітерацій абсолютно не гарантує досягнення якомога більш естетичного положення розміщення, що відповідає стану рівноваги фізичної моделі. По-друге, такий підхід зовсім не є універсальним, адже не враховує особливості розмірів і топології аналізованих графів. До того ж, розроблення функцій отримання очікуваного числа ітерацій для конкретного графа передбачатиме детальний і складний аналіз, або ж вагомі спрощення.

Досягнення порогу зміщення вузлів або енергії системи є універсальнішими підходами, оскільки спираються саме на фізичну модель. Очевидним недоліком обох є невизначеність часу виконання алгоритму. На противагу ці підходи дають змогу отримати природні з позиції фізичної моделі і, як наслідок, естетичніші розміщення. Зупинка за умови досягнення порогу енергії системи дає змогу в загальному випадку досягнути глобального мінімуму, а отже, найбільш якісного розміщення. Однак обрахунок енергії системи є витратнішим, а сам критерій чутливий до визначеного порогу й потребує тонкого налаштування. Зупинка за умови досягнення порогу зміщення вузлів може відбутися у точці локального мінімуму, однак сама умова обраховується швидше. У цьому підході визначити поріг значно простіше, бо, по суті, потрібно просто визначити мінімальну відстань зміщення вузлів, що сприймається незначною і такою, що нею можна знехтувати.

Ми обрали комбіновану умову зупинки з використанням порогу зміщення вузлів, а також максимальною кількістю ітерацій. Використання додаткового обмеження за кількістю ітерацій обумовлене тим, що аналізовані графи можуть бути зовсім різними за розмірами, а обрахунок часу їхнього розміщення може значно різнитись. До того ж, оскільки метою є створення евристики для початкового розміщення, тенденції поведінки фізичних моделей саме на певній кількості перших ітерацій алгоритму є найбільш універсальними.

Також у загальному випадку граф архітектури програми може бути не суцільним, тому досягнення повного спокою системи може бути неможливим. Щоб вирішити цю проблему, на розміщення графу накладено додаткову умову – воно має вміщуватись у підпросторі фіксованого розміру.

У підсумку процес підготовки даних є модульним й охоплює три етапи: збір проєктів і початковий аналіз структури проєкту; побудову зваженого графа з розміщенням в просторі; генерацію представлення для навчання. Процес повністю автоматизований. Перевикористання розробленого інструменту для проєктів, що застосовують інші технології, не викликає труднощів і, по суті, зводиться до автоматизації початкового аналізу структури проєкту й імплементації аналізатора для конкретної мови й технології.

4. Використання графових нейронних мереж

Складність застосування нейронних мереж для задачі візуалізації довільного зваженого графа визначається тим, що у нашому випадку вузли графа не містять жодних метаданих, корисних для побудови евристики. По суті, єдиною інформацією, що впливає на розміщення в просторі, є наявність ребер між певними вузлами та їхні ваги. Саме тому підходи, що спираються на попередню кластеризацію вузлів за певними ознаками, у цьому випадку не можна застосувати.

Важливим аспектом під час підбору технологій і розроблення архітектури запропонованого рішення був намір зробити його крос-платформним, а саме – забезпечити можливість обчислення на мобільних пристроях. Тоді за умови досягнення достатньої продуктивності новоствореної евристики, візуалізатор можна було б перемістити з комп'ютерної частини програмного комплексу до мобільного додатка-переглядача (див. рис. 2). Це забезпечить гнучкість і можливість створення якісніших взаємодій користувача з візуальним представленням програми, наприклад, побудову представлення певної частини загального графа відповідно до конкретних потреб.

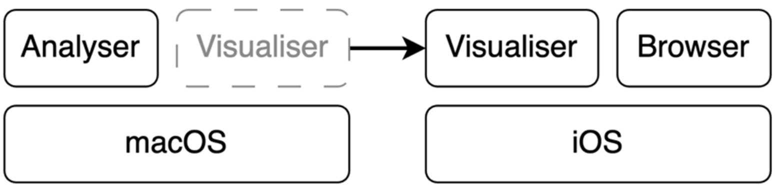


Рис. 2. Схема переміщення візуалізатора в межах програмного комплексу

З огляду на це в основу моделі нейронної мережі було покладено графовий згортковий оператор GCNConv, запропонований у (Kipf, & Welling, 2016). Серед переваг цього оператора варто звернути увагу на можливість роботи з вагами ребер, а також повну сумісність внутрішньої реалізації з CoreML, що, зі свого боку, дає змогу конвертувати модель для роботи в застосунках для macOS й iOS із забезпеченням сумісності зі спеціальним апаратним забезпеченням.

Архітектура моделі є послідовністю кількох шарів GCNConv (див. рис. 3). Перший шар приймає на вхід набір вузлів з їхніми властивостями (V) та набір ребер (E) з їхніми вагами (W). Варто зауважити, що, оскільки інформативних метаданих вузли графів у нашій задачі не мають, то для забезпечення сумісності як властивість вузла було вирішено використати степінь вузла графу. Розмір даних на виході першого шару позначаємо hidden_dim (hidden layer dimension). Ця розмірність загалом може бути довільною, однак достатньою для оброблення властивостей всіх вузлів графа, а отже, не меншою за їхню кількість. У ході аналізу згенерованих даних було встановлено, що кількість вузлів для кожного з графів не перевищувала 2000, тому значення hidden_dim було встановлено відповідно.

Останній шар приймає на вхід дані розмірності hidden_dim, а повертає набір тривимірних векторів (X), що позначає набір точок у тривимірному просторі – координати кожного вузла у відповідному розміщенні графа.

Між першим й останнім шаром може бути довільна кількість допоміжних шарів, які можуть додати гнучкості та точності моделі. Розмірність вхідних і вихідних даних цих шарів однакова – hidden_dim. В оригінальному дослідженні (Kipf, & Welling, 2016) автори зазначають, що для їхньої задачі позитивні результати вдалось отримати з використанням дво- та трирівневої моделі. Очевидним є те, що збільшення кількості шарів моделі збільшує її розмір і час обчислення, а отже, несприятливо впливає на можливість портування. З іншого боку, передбачити оптимальну кількість допоміжних шарів з урахуванням різноманітності вхідних даних досить складно. Саме тому в контексті експерименту було підготовано три споріднені моделі: дворівневу (L2) – без додаткових шарів, трирівневу (L3) – з одним прихованим шаром, а також чотирирівневу (L4) – із двома прихованими шарами.

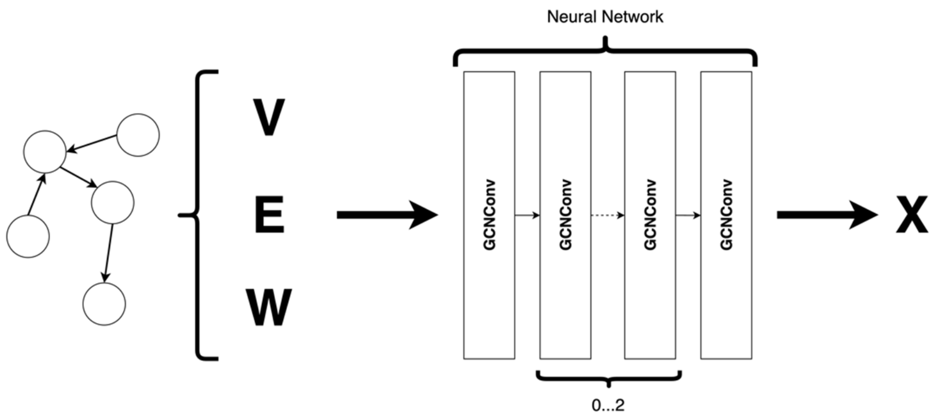


Рис. 3. Принцип роботи спеціалізованої евристики

Як функцію втрат під час навчання було використано середньоквадратичну похибку попарних відстаней між передбаченими й еталонними координатами. Бажане розміщення, тобто набір еталонних координат, обчислювали за допомогою силового алгоритму, описаного вище. З огляду на нестабільність алгоритму, очевидним є те, що такі координати можуть представляти не ідеальне розміщення, однак з точністю до створення евристики для цього ж алгоритму такий підхід є виправданим.

Як функцію активації було використано ReLU, що дало змогу враховувати нелінійні залежності у даних, зберігаючи ефективність обчислень.

Під час навчання мережі значення помилки спадало, однак залишалось високим. Ймовірно, це зумовлено великою різницею між розмірами графів. Через це складно вважати запропонований підхід ефективним для самостійного використання з метою передбачення координат усіх вузлів одразу.

Отримані моделі мереж було конвертовано в CoreML за допомогою спеціального набору утиліт coremltools. Унаслідок конвертування розміри готових до використання файлів *.mlpackage становили 50 кБ для L2, 8,1 МБ для L3 та 16,1 МБ для L4.

Отже, ми створили новий візуалізатор, який приймає на вхід модель мережі та зважений граф, застосовує її для обчислення початкового розміщення вузлів, а потім використовує силовий алгоритм для генерації розміщення в просторі.

5. Перевірка евристики й інтерпретація результатів

Для перевірки продуктивності роботи візуалізатора зі спеціалізованою евристикою створено спеціальну програму, що дає змогу фіксувати кількість ітерацій силового алгоритму під час побудови розміщення, а також час, витрачений на його обчислення.

Експеримент із заміру продуктивності було проведено на комп'ютері Apple MacBook M1 Pro з 10 ядрами CPU, 14 ядрами GPU та 16 Neural Engine. Графи було отримано автоматично, що допомогло в дослідженні особливостей роботи запропонованого рішення для задач різних розмірів і конфігурацій.

Для кожного з візуалізаторів було проведено по п'ять побудов кожного зразка. Візуалізатори сформовано на основі класичного силового алгоритму, однак кожен використовує власну евристику. Default застосовує випадкову евристику, L2, L3 та L4 використовують нашу спеціалізовану евристику, імплементовану у формі нейронної мережі з відповідною кількістю рівнів.

У загальній таблиці (дод. 1) наведено частину результатів проведення побудов розміщень графів архітектури для різних програм. Для кожного експерименту надано інформацію про кількість вузлів N(V) і ребер N€ у графі, а також мінімальне, максимальне та середнє значення для кількості ітерацій *I* та часу *t* в секундах до зупинки алгоритму.

Важливою перевагою використання запропонованої евристики стала стабілізація кількості ітерацій, причому незалежно від глибини конкретно взятої нейронної мережі. Це, відповідно, робить виконання алгоритму передбачуваним і дає змогу краще оцінити очікуваний час завершення.

У контексті підвищення ефективності алгоритму за допомогою евристики вдалось дослідити певні особливості залежності *I* та *t* від комбінації розмірів графа N(V), N€ та конкретної моделі нейронної мережі. Зауважимо, що тут і далі розподіл графів на малі, середні та великі є умовним і стосується, радше, зручності ілюстрації конкретних результатів (табл. 1).

Таблиця 1

Результати застосування евристики для малих графів
з $\mathrm{N(V)} < 50, \mathrm{N(E)} < 50$

	N(V)	N(E)	min(I)	max(I)	avg(I)	min(t)	max(t)	avg(t)
Sample 6								
Default	24	21	8550	10000	9516,67	0,20	0,23	0,22
L2	24	21	5860	5860	5860,00	0,38	0,40	0,39
L3	24	21	7675	7675	7675,00	0,50	0,59	0,53
L4	24	21	10000	10000	10000,00	0,63	0,64	0,64
Sample 13								
Default	15	19	950	1373	1141,67	0,01	0,06	0,03
L2	15	19	875	875	875,00	0,25	0,25	0,25
L3	15	19	2282	2282	2282,00	0,34	0,38	0,36
L4	15	19	1117	1117	1117,00	0,36	0,38	0,37

У табл. 1 добре видно, що для малих графів з N(V) < 50, N(E) < 50 добре працює евристика на базі L2. Однак попри те, що значення *I* значно менше і в середньому, і навіть для мінімального випадку, значення *t* виходить значно більшим. Такий результат пояснюємо тим, що час на виконання ΔI ітерацій для невеликого розміру вхідних даних незрівнянно менший, ніж час, необхідний для обчислення евристики. Отже, у такому випадку її використання не є ефективним.

Із табл. 2 випливає, що для графів більшого розміру в контексті значення *I* ефективні показники мають евристики L2 та L3. Що стосується мінімізації часу виконання, то зі збільшенням розмірів графа час, необхідний для виконання ΔI ітерацій, стає більшим і, як наслідок, застосування евристики демонструє певну ефективність.

Таблиця 2

Результати застосування евристики для середніх графів
з $N(V) \in (50, 100), N(E) \in (50, 100)$

	N(V)	N(E)	min(I)	max(I)	avg(I)	min(t)	max(t)	avg(t)
Sample 1								
Default	91	97	3833	10000	6092,33	0,57	1,49	0,91
L2	91	97	3537	3537	3537,00	0,74	0,75	0,74
L3	91	97	5416	5416	5416,00	1,08	1,10	1,09
L4	91	97	5324	5324	5324,00	1,16	1,18	1,17
Sample 12								
Default	68	78	8834	10000	9611,33	0,85	0,95	0,92
L2	68	78	10000	10000	10000,00	1,16	1,29	1,21
L3	68	78	5735	5735	5735,00	0,81	0,89	0,84
L4	68	78	10000	10000	10000,00	1,28	1,33	1,31

Як видно з табл. 3 для графів більшого розміру використання еристики L3 забезпечує значне спадання значень *I* та *t*. Зважаючи на ці дані, можна стверджувати, що застосування спеціалізованої евристики для графів, починаючи з такого розміру, є виправданим і скорочує середній час, необхідний для виконання алгоритму, приблизно до 50–60 % у середньому і приблизно до 40 %, якщо порівнювати максимальні значення.

Таблиця 3

Результати застосування евристики для середніх графів
з $N(V) \in (100, 300), N(E) \in (100, 300)$

	N(V)	N(E)	min(I)	max(I)	avg(I)	min(t)	max(t)	avg(t)
Sample 11								
Default	152	131	4802	10000	7683,33	1,92	3,98	3,07
L2	152	131	10000	10000	10000,00	4,19	4,21	4,20
L3	152	131	3728	3728	3728,00	1,77	1,78	1,78
L4	152	131	9311	9311	9311,00	4,10	4,11	4,11
Sample 42								
Default	129	138	4668	10000	7008,33	1,34	2,87	2,01
L2	129	138	4691	4691	4691,00	1,56	1,58	1,57
L3	129	138	1901	1901	1901,00	0,82	0,84	0,83
L4	**129**	**138**	**3389**	**3389**	**3389,00**	**1,29**	**1,35**	**1,32**

Варто зауважити, що для деяких графів середніх розмірів, як видно з табл. 4, евристика на основі L4 працює краще в контексті мінімізації *I*, однак через розмір моделі використання цієї мережі вимагає більше часу, а тому використання L4 не зменшує *t* порівняно з L3.

Таблиця 4

Результати застосування евристики для середніх графів з $N(V) \in (50, 100), N(E) \in (100, 300)$

	N(V)	N(E)	min(I)	max(I)	avg(I)	min(t)	max(t)	avg(t)
Sample 31								
Default	89	101	7069	10000	9023,00	1,14	1,49	1,37
L2	89	101	7870	7870	7870,00	1,35	1,39	1,37
L3	89	101	6406	6406	6406,00	1,20	1,22	1,21
L4	89	101	5972	5972	5972,00	1,22	1,24	1,23
Sample 5								
Default	86	103	3754	9341	6103,67	0,51	1,28	0,83
L2	86	103	8285	8285	8285,00	1,33	1,35	1,34
L3	86	103	5500	5500	5500,00	1,02	1,04	1,03
L4	86	103	5278	5278	5278,00	1,07	1,10	1,09

Цікавим є також те, що хоча використання евристики на основі L3 не завжди гарантує прискорення виконання, її все ж варто розглядати як інструмент стабілізації, адже різниці Δt та ΔI для максимальних значень можуть бути відчутними.

З отриманих результатів також можна зробити припущення, що для графів більшого розміру ефективнішими можуть бути евристики, побудовані на основі мереж з більшою кількістю рівнів. Приклад 35 (див. табл. 5) ілюструє ефективність застосування L4 для графів з кількістю вузлів та ребер понад 300. Варто зауважити, що для більших за розміром графів обмеження кількості ітерацій може виявитись занадто сильним і негативно впливати на результати замірів, адже впорядкування більшої за розміром системи природно вимагає більшої кількості ітерацій. З огляду на це детальніше дослідження поведінки евристики для великих графів варто проводити у більшому часовому вікні.

Таблиця 5

Результати застосування евристики для великих графів з $N(V) > 300, N(E) > 300$

	N(V)	N(E)	min(I)	max(I)	avg(I)	min(t)	max(t)	avg(t)
Sample 35								
Default	313	370	5082	10000	6855,67	8,33	16,38	11,24
L2	313	370	10000	10000	10000,00	16,57	16,83	16,68
L3	313	370	10000	10000	10000,00	16,62	16,69	16,66
L4	313	370	6759	6759	6759,00	11,51	11,68	11,60

Загалом маємо зазначити, що приблизно для третини тестових зразків досягти порогу зміщення вузлів у межах 10 000 ітерацій не вдалось. Однією з причин є, власне, великий розмір вхідних даних. Іншою причиною можуть бути особливості будови графа, як-от розрідженість чи наявність ізольованих частин.

Спираючись на отримані результати, можна зробити висновок, що для достатньо великих графів є сенс застосовувати запропонований підхід.

Дискусія і висновки

У цій статті запропоновано спосіб використання машинного навчання, а саме нейронних мереж, для створення спеціалізованих евристик для силового алгоритму, що застосовують для розміщення графових моделей архітектури програми в просторі.

Збирання реальних даних і процес створення евристик повністю автоматизовано так, що з невеликими адаптаціями його можна з легкістю використати для довільних технологій та мов програмування.

Для мови програмування Swift для iOS/macOS застосунків було створено спеціальний візуалізатор, що використовує запропоновані евристики. На основі проведених досліджень наочно продемонстровано ефективність застосування цього підходу для достатньо великих графів, що стабілізує процес роботи алгоритму, а також може значно зменшувати кількість часу обчислень.

Описано доцільність використання нейронних мереж з різною кількістю рівнів залежно від розмірів графу. Детальніше дослідження цієї проблеми залишається предметом майбутньої роботи.

Список використаних джерел

Франків, О. О. (2023). Використання доповненої реальності для візуалізації архітектур програмних модулів. *Наукові записки НаУКМА,* 5, 26–30. https://doi.org/10.18523/2617-3808.2022.5.26-30

Франків, О. О., & Глибовець, М. М. (2024). Автоматизована візуалізація компонентів архітектури програми для мови Swift. *Кібернетика та системний аналіз,* 60(6), 190–198. https://doi.org/10.1007/s10559-024-00737-9

Barnes, J., & Hut, P. (1986). A hierarchical O (N log N) force-calculation algorithm. *Nature,* 324(6096), 446–449. https://doi.org/10.1038/324446a0

Didimo, W., Liotta, G., & Romeo, S. A. (2011). Topology-driven force-directed algorithms. In U. Brandes, & S. Cornelsen (Eds.), *Graph Drawing. GD 2010. Lecture Notes in Computer Science, Vol. 6502* (pp. 165–176). Springer. https://doi.org/10.1007/978-3-642-18469-7_15

Fruchterman, T. J., & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software – Practice & Experience,* 21, 1129–1164. https://doi.org/10.1002/spe.4380211102

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 855–864). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939754

Huang, R., Gai, H., Jing, R., & Wan, F. (2024). Word spectral visualization based on Fruchterman-Reingold optimized ForceAtlas2. *Journal of Electrical Systems,* 20(2), 7–15.

Kipf, T. N., & Welling, M. (2016). *Semi-supervised classification with graph convolutional networks.* arXiv. https://doi.org/10.48550/arXiv.1609.02907

Pohl, M., Schmitt, M., & Diehl, S. (2009). Comparing the readability of graph layouts using eye-tracking and task-oriented analysis. In *CompAesth 09: Workshop on Computational Aesthetics* (pp. 49–56). https://doi.org/10.2312/COMPAESTH/COMPAESTH09/049-056
Rahman, M. K., Sujon, M. H., & Azad, A. (2020, November). Force2Vec: Parallel force-directed graph embedding. In *2020 IEEE International Conference on Data Mining* (pp. 442–451). IEEE. https://doi.org/10.1109/ICDM50108.2020.00056
Walshaw, C. (2001). A multilevel algorithm for force-directed graph drawing. In J. Marks (Ed.), *Graph Drawing. GD 2000. Lecture Notes in Computer Science, Vol. 1984* (pp. 171–182). Springer. https://doi.org/10.1007/3-540-44541-2_17
Wang, X., Yen, K., Hu, Y., & Shen, H. W. (2024). SmartGD: A GAN-based graph drawing framework for diverse aesthetic goals. *IEEE Transactions on Visualization and Computer Graphics*, 30, 5666–5678 https://doi.ieeecomputersociety.org/10.1109/TVCG.2023.3306356
Wang, Y., Jin, Z., Wang, Q., Cui, W., Ma, T., & Qu, H. (2020). DeepDrawing: A deep learning approach to graph drawing. *IEEE Transactions on Visualization and Computer Graphics,* 26(1), 676–686. https://doi.ieeecomputersociety.org/10.1109/TVCG.2019.2934798

References

Barnes, J., & Hut, P. (1986). A hierarchical O (N log N) force-calculation algorithm. *Nature,* 324(6096), 446–449. https://doi.org/10.1038/324446a0
Didimo, W., Liotta, G., & Romeo, S. A. (2011). Topology-driven force-directed algorithms. In U. Brandes, & S. Cornelsen (Eds.), *Graph Drawing. GD 2010. Lecture Notes in Computer Science, Vol. 6502.* (pp. 165–176). Springer. https://doi.org/10.1007/978-3-642-18469-7_15
Frankiv, O. O. (2023). Using augmented reality for visualizing architectures of software modules. *NaUKMA Research Papers. Computer Science*, 5, 26–30 [in Ukrainian]. https://doi.org/10.18523/2617-3808.2022.5.26-30
Frankiv, O. O., & Glybovets, M. M. (2024). Automated visualization of program architecture components for the Swift language. *Cybernetics and Systems Analysis,* 60(6), 190–198 [in Ukrainian]. https://doi.org/10.1007/s10559-024-00737-9
Fruchterman, T. J., & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software – Practice & Experience,* 21, 1129–1164. https://doi.org/10.1002/spe.4380211102
Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 855–864). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939754
Huang, R., Gai, H., Jing, R., & Wan, F. (2024). Word spectral visualization based on Fruchterman-Reingold optimized ForceAtlas2. *Journal of Electrical Systems,* 20(2), 7–15.
Kipf, T. N., & Welling, M. (2016). *Semi-supervised classification with graph convolutional networks.* arXiv. https://doi.org/10.48550/arXiv.1609.02907
Pohl, M., Schmitt, M., & Diehl, S. (2009). Comparing the readability of graph layouts using eye-tracking and task-oriented analysis. In *CompAesth 09: Workshop on Computational Aesthetics* (pp. 49–56). https://doi.org/10.2312/COMPAESTH/COMPAESTH09/049-056
Rahman, M. K., Sujon, M. H., & Azad, A. (2020, November). Force2Vec: Parallel force-directed graph embedding. In *2020 IEEE International Conference on Data Mining* (pp. 442–451). IEEE. https://doi.org/10.1109/ICDM50108.2020.00056
Walshaw, C. (2001). A multilevel algorithm for force-directed graph drawing. In J. Marks (Ed.), *Graph Drawing. GD 2000. Lecture Notes in Computer Science, Vol. 1984* (pp. 171–182). Springer. https://doi.org/10.1007/3-540-44541-2_17
Wang, X., Yen, K., Hu, Y., & Shen, H. W. (2024). SmartGD: A GAN-based graph drawing framework for diverse aesthetic goals. *IEEE Transactions on Visualization and Computer Graphics*, 30, 5666–5678 https://doi.ieeecomputersociety.org/10.1109/TVCG.2023.3306356
Wang, Y., Jin, Z., Wang, Q., Cui, W., Ma, T., & Qu, H. (2020). DeepDrawing: A deep learning approach to graph drawing. *IEEE Transactions on Visualization and Computer Graphics,* 26(1), 676–686. https://doi.ieeecomputersociety.org/10.1109/TVCG.2019.2934798

Отримано редакцією журналу / Received: 01.02.25
Прорецензовано / Revised: 10.03.25
Схвалено до друку / Acepted: 12.03.25

Oleksandr FRANKIV, PhD Student
ORCID ID: 0009-0005-5264-358X
e-mail: o.frankiv@ukma.edu.ua
National University of Kyiv-Mohyla Academy, Kyiv, Ukraine

MACHINE LEARNING IN ENHANCING VISUALIZATION OF THE SPATIAL SOFTWARE ARCHITECTURE MODEL

Visual aesthetics and computational efficiency are equally important aspects in the context of creating high-quality data representations for further analysis. A harmonious combination of these characteristics not only enhances the ease of perception but also optimizes data processing workflows, which is critically important in modern software systems.

This paper proposes a novel combined approach for spatial graph placement. Special attention is given to spatial models of software architecture, which serve as key tools for visualizing complex relationships between components. The use of graph neural networks as a specialized heuristic forms the central element of this approach. Leveraging machine learning methods, the proposed solution improves visualization outcomes while enhancing computational efficiency.

The application of a graph neural network ensures adaptability and enables the model to account for the specific features of the graph. In combination with a force-directed algorithm, this allows for maintaining a high level of visual aesthetics without significant increases in resource consumption. Thus, the new method offers a practical solution for effectively combining visual aesthetics with computational efficiency, representing an important step forward in enhancing the analysis of spatial models in software architecture.

Keywords: neural network, machine learning, graph, graph neural network, graph convolutional operator (GCNConv), software architecture, automatic visualization of software architecture, time complexity of an algorithm.

Автор заявляє про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The author declares no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; in the decision to publish the results.

Додаток 1

Загальні результати ефективності використання спеціалізованої евристики

	N(V)	N(E)	min(I)	max(I)	avg(I)	min(t)	max(t)	avg(t)
Sample 1								
Default	91	97	3833	10000	6092,33	0,57	1,49	0,91
L2	91	97	3537	3537	3537,00	0,74	0,75	0,74
L3	91	97	5416	5416	5416,00	1,08	1,10	1,09
L4	91	97	5324	5324	5324,00	1,16	1,18	1,17
Sample 2								
Default	62	91	8492	10000	9497,33	0,67	0,80	0,75
L2	62	91	10000	10000	10000,00	1,00	1,03	1,02
L3	62	91	10000	10000	10000,00	1,06	1,07	1,07
L4	62	91	10000	10000	10000,00	1,14	1,16	1,15

Sample 3								
Default	56	56	2919	7569	5219,00	0,23	0,61	0,42
L2	56	56	8457	8457	8457,00	0,93	0,95	0,94
L3	56	56	10000	10000	10000,00	1,11	1,17	1,14
L4	56	56	4458	4458	4458,00	0,75	0,76	0,76
Sample 4								
Default	50	57	8445	10000	9481,67	0,49	0,58	0,54
L2	50	57	8409	8409	8409,00	0,68	0,71	0,70
L3	50	57	8402	8402	8402,00	0,74	0,77	0,76
L4	50	57	9058	9058	9058,00	0,85	0,89	0,87
Sample 5								
Default	86	103	3754	9341	6103,67	0,51	1,28	0,83
L2	86	103	8285	8285	8285,00	1,33	1,35	1,34
L3	86	103	5500	5500	5500,00	1,02	1,04	1,03
L4	86	103	5278	5278	5278,00	1,07	1,10	1,09
Sample 6								
Default	24	21	8550	10000	9516,67	0,20	0,23	0,22
L2	24	21	5860	5860	5860,00	0,38	0,40	0,39
L3	24	21	7675	7675	7675,00	0,50	0,59	0,53
L4	24	21	10000	10000	10000,00	0,63	0,64	0,64
Sample 7								
Default	1807	2260	10000	10000	10000,00	556,34	557,41	557,04
L2	1807	2260	10000	10000	10000,00	556,57	557,05	556,81
L3	1807	2260	10000	10000	10000,00	570,83	573,00	571,83
L4	1807	2260	10000	10000	10000,00	577,32	579,72	578,92
Sample 8								
Default	41	39	6000	10000	8666,67	0,23	0,37	0,32
L2	41	39	10000	10000	10000,00	0,57	0,59	0,58
L3	41	39	10000	10000	10000,00	0,64	0,65	0,64
L4	41	39	10000	10000	10000,00	0,71	0,71	0,71
Sample 9								
Default	343	462	1584	10000	7194,67	3,12	19,67	14,15
L2	343	462	10000	10000	10000,00	19,87	19,89	19,88
L3	343	462	10000	10000	10000,00	19,98	20,01	20,00
L4	343	462	10000	10000	10000,00	20,08	20,12	20,10
Sample 10								
Default	91	111	6709	10000	8540,67	1,32	1,95	1,67
L2	91	111	10000	10000	10000,00	2,26	2,82	2,53
L3	91	111	7451	7451	7451,00	1,79	1,81	1,80
L4	91	111	5917	5917	5917,00	1,59	1,59	1,59
Sample 11								
Default	152	131	4802	10000	7683,33	1,92	3,98	3,07
L2	152	131	10000	10000	10000,00	4,19	4,21	4,20
L3	152	131	3728	3728	3728,00	1,77	1,78	1,78
L4	152	131	9311	9311	9311,00	4,10	4,11	4,11
Sample 12								
Default	68	78	8834	10000	9611,33	0,85	0,95	0,92
L2	68	78	10000	10000	10000,00	1,16	1,29	1,21
L3	68	78	5735	5735	5735,00	0,81	0,89	0,84
L4	68	78	10000	10000	10000,00	1,28	1,33	1,31
Sample 13								
Default	15	19	950	1373	1141,67	0,01	0,06	0,03
L2	15	19	875	875	875,00	0,25	0,25	0,25
L3	15	19	2282	2282	2282,00	0,34	0,38	0,36
L4	15	19	1117	1117	1117,00	0,36	0,38	0,37
Sample 14								
Default	91	121	3792	7282	5583,67	0,67	1,28	0,98
L2	91	121	1434	1434	1434,00	0,48	0,49	0,48
L3	91	121	4061	4061	4061,00	1,01	1,03	1,02
L4	91	121	3305	3305	3305,00	0,94	0,97	0,95
Sample 15								
Default	68	73	5537	10000	7539,33	0,48	0,87	0,66
L2	68	73	5534	5534	5534,00	0,71	0,77	0,73
L3	68	73	7768	7768	7768,00	0,97	1,00	0,98
L4	68	73	7162	7162	7162,00	0,97	0,98	0,97
Sample 16								
Default	105	113	4636	8346	5977,33	0,89	1,61	1,15
L2	105	113	4046	4046	4046,00	0,99	0,99	0,99
L3	105	113	1589	1589	1589,00	0,59	0,62	0,60
L4	105	113	3070	3070	3070,00	0,97	0,99	0,98

Sample 17								
Default	28	27	4083	8518	6224,67	0,13	0,30	0,22
L2	28	27	1739	1739	1739,00	0,27	0,30	0,28
L3	28	27	770	770	770,00	0,30	0,33	0,31
L4	28	27	5011	5011	5011,00	0,57	0,66	0,62
Sample 18								
Default	824	955	10000	10000	10000,00	111,19	111,28	111,23
L2	824	955	10000	10000	10000,00	111,47	111,53	111,50
L3	824	955	10000	10000	10000,00	111,51	111,60	111,56
L4	824	955	10000	10000	10000,00	111,68	111,76	111,72
Sample 19								
Default	26	20	3862	9699	6257,67	0,13	0,35	0,23
L2	26	20	10000	10000	10000,00	0,56	0,60	0,58
L3	26	20	4898	4898	4898,00	0,38	0,44	0,41
L4	26	20	3153	3153	3153,00	0,43	0,48	0,45
Sample 20								
Default	45	41	6383	10000	7808,67	0,33	2,05	0,98
L2	45	41	4007	4007	4007,00	0,40	0,54	0,45
L3	45	41	10000	10000	10000,00	0,73	0,77	0,75
L4	45	41	10000	10000	10000,00	0,82	0,83	0,82
Sample 21								
Default	10	9	1539	7589	5516,33	0,08	0,26	0,20
L2	10	9	2799	2799	2799,00	0,28	0,33	0,31
L3	10	9	3971	3971	3971,00	0,35	0,43	0,39
L4	10	9	2085	2085	2085,00	0,40	0,48	0,45
Sample 22								
Default	4	3	2729	5501	4366,67	0,09	0,20	0,14
L2	4	3	5509	5509	5509,00	0,37	0,43	0,39
L3	4	3	2728	2728	2728,00	0,30	0,33	0,32
L4	4	3	8953	8953	8953,00	0,64	0,68	0,66
Sample 23								
Default	18	11	4969	6712	5984,33	0,06	0,09	0,08
L2	18	11	2144	2144	2144,00	0,23	0,23	0,23
L3	18	11	2518	2518	2518,00	0,29	0,31	0,30
L4	18	11	7114	7114	7114,00	0,37	0,43	0,41
Sample 24								
Default	20	14	7616	10000	8835,00	0,15	0,32	0,22
L2	20	14	7195	7195	7195,00	0,43	0,46	0,44
L3	20	14	2931	2931	2931,00	0,38	0,46	0,41
L4	20	14	4842	4842	4842,00	0,49	0,49	0,49
Sample 25								
Default	512	524	10000	10000	10000,00	41,88	42,30	42,10
L2	512	524	10000	10000	10000,00	41,53	42,48	41,96
L3	512	524	10000	10000	10000,00	41,64	42,73	42,29
L4	512	524	10000	10000	10000,00	41,63	41,70	41,66
Sample 26								
Default	493	635	10000	10000	10000,00	56,47	57,13	56,76
L2	493	635	10000	10000	10000,00	57,29	57,47	57,35
L3	493	635	10000	10000	10000,00	57,07	58,24	57,55
L4	493	635	10000	10000	10000,00	46,91	56,73	53,28
Sample 27								
Default	16	14	7586	10000	9111,67	0,20	0,32	0,25
L2	16	14	8852	8852	8852,00	0,42	0,43	0,43
L3	16	14	10000	10000	10000,00	0,55	0,58	0,56
L4	16	14	10000	10000	10000,00	0,57	0,68	0,62
Sample 28								
Default	87	114	3432	10000	6456,00	0,67	1,84	1,20
L2	87	114	8418	8418	8418,00	1,82	2,03	1,94
L3	87	114	7220	7220	7220,00	1,67	1,68	1,67
L4	87	114	10000	10000	10000,00	2,29	2,79	2,53
Sample 29								
Default	49	56	10000	10000	10000,00	0,52	0,57	0,55
L2	49	56	10000	10000	10000,00	0,75	0,77	0,76
L3	49	56	10000	10000	10000,00	0,81	0,83	0,82
L4	49	56	10000	10000	10000,00	0,89	0,91	0,90
Sample 30								
Default	35	46	3650	10000	6308,00	0,13	0,39	0,24
L2	35	46	3376	3376	3376,00	0,38	0,39	0,39
L3	35	46	10000	10000	10000,00	0,69	0,72	0,70
L4	35	46	7253	7253	7253,00	0,67	0,68	0,68

Sample 31								
Default	89	101	7069	10000	9023,00	1,14	1,49	1,37
L2	89	101	7870	7870	7870,00	1,35	1,39	1,37
L3	89	101	6406	6406	6406,00	1,20	1,22	1,21
L4	89	101	5972	5972	5972,00	1,22	1,24	1,23
Sample 32								
Default	46	46	9983	10000	9994,33	0,55	0,56	0,55
L2	46	46	10000	10000	10000,00	0,80	0,87	0,83
L3	46	46	10000	10000	10000,00	0,86	0,88	0,87
L4	46	46	10000	10000	10000,00	0,95	0,96	0,96
Sample 33								
Default	106	92	2986	5714	4514,00	0,56	1,08	0,85
L2	106	92	3449	3449	3449,00	0,86	0,87	0,87
L3	106	92	1396	1396	1396,00	0,54	0,54	0,54
L4	106	92	4878	4878	4878,00	1,26	1,27	1,27
Sample 34								
Default	30	27	10000	10000	10000,00	0,30	0,30	0,30
L2	30	27	10000	10000	10000,00	0,55	0,58	0,56
L3	30	27	10000	10000	10000,00	0,61	0,64	0,62
L4	30	27	10000	10000	10000,00	0,70	0,71	0,70
Sample 35								
Default	313	370	5082	10000	6855,67	8,33	16,38	11,24
L2	313	370	10000	10000	10000,00	16,57	16,83	16,68
L3	313	370	10000	10000	10000,00	16,62	16,69	16,66
L4	313	370	6759	6759	6759,00	11,51	11,68	11,60
Sample 36								
Default	56	65	2645	10000	5680,67	0,19	0,66	0,39
L2	56	65	4869	4869	4869,00	0,53	0,56	0,54
L3	56	65	6693	6693	6693,00	0,72	0,73	0,72
L4	56	65	5185	5185	5185,00	0,65	0,73	0,69
Sample 37								
Default	623	1148	10000	10000	10000,00	69,74	69,84	69,78
L2	623	1148	10000	10000	10000,00	69,72	69,98	69,85
L3	623	1148	10000	10000	10000,00	69,67	70,36	69,92
L4	623	1148	10000	10000	10000,00	70,38	70,49	70,43
Sample 38								
Default	64	49	2378	4474	3282,33	0,20	0,36	0,27
L2	64	49	4336	4336	4336,00	0,55	0,58	0,57
L3	64	49	2832	2832	2832,00	0,50	0,51	0,51
L4	64	49	2228	2228	2228,00	0,51	0,54	0,53
Sample 39								
Default	244	245	2304	4471	3543,00	2,27	4,40	3,49
L2	244	245	3628	3628	3628,00	3,79	3,84	3,82
L3	244	245	2303	2303	2303,00	2,59	2,75	2,65
L4	244	245	10000	10000	10000,00	10,27	10,52	10,37
Sample 40								
Default	63	67	3378	10000	6604,67	0,27	0,78	0,52
L2	63	67	3430	3430	3430,00	0,48	0,49	0,48
L3	63	67	10000	10000	10000,00	1,07	1,08	1,07
L4	63	67	10000	10000	10000,00	1,14	1,15	1,14
Sample 41								
Default	179	231	4681	6025	5266,67	2,62	3,37	2,95
L2	179	231	10000	10000	10000,00	5,75	5,83	5,79
L3	179	231	10000	10000	10000,00	5,83	5,84	5,84
L4	179	231	10000	10000	10000,00	5,89	5,91	5,90
Sample 42								
Default	129	138	4668	10000	7008,33	1,34	2,87	2,01
L2	129	138	4691	4691	4691,00	1,56	1,58	1,57
L3	129	138	1901	1901	1901,00	0,82	0,84	0,83
L4	129	138	3389	3389	3389,00	1,29	1,35	1,32
Sample 43								
Default	14	11	10000	10000	10000,00	0,29	0,33	0,31
L2	14	11	10000	10000	10000,00	0,49	0,55	0,51
L3	14	11	10000	10000	10000,00	0,52	0,62	0,57
L4	14	11	10000	10000	10000,00	0,66	0,71	0,68
Sample 44								
Default	361	531	10000	10000	10000,00	23,11	23,34	23,26
L2	361	531	10000	10000	10000,00	22,53	22,87	22,70
L3	361	531	10000	10000	10000,00	23,38	23,55	23,45
L4	361	531	10000	10000	10000,00	23,35	23,65	23,53