

УДК 621.396.6 : 004.8

Малий О.Ю.¹, Поспеева І.Є.², Калашник О.А.³

¹ канд. тех. наук, доц. зав. каф. ІТЕЗ НУ «Запорізька політехніка»

² ст. викл. каф. ІТЕЗ НУ «Запорізька політехніка»

³ студ. гр. БК-613м НУ «Запорізька політехніка»

РОЗМІЩЕННЯ КОМПОНЕНТІВ НА ДРУКОВАНІЙ ПЛАТІ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Сьогодні більшу частину навантаження на проектування друкованої плати бере на себе комп'ютер, де великі програмні пакети автоматизації електронного проектування (EDA) автоматизують більшу частину процесу. Програмне забезпечення EDA економить дизайнеру незліченну кількість годин, покращуючи швидкість і послідовність, з якою вони можуть будувати схеми, імітувати функціональність і виготовляти нову друковану плату. Однак, коли справа доходить до фактичного розміщення компонентів і маршрутизації, більша частина навантаження все ще лежить на розробниках і їх ручній роботі. Багаторічний досвід розміщення та маршрутизації друкованих плат перевершує більшість передових алгоритмів.

Оптимальна маршрутизація друкованої плати є прикладом давньої проблеми комівояжера NP-Complete. Оскільки це складно вирішити і функціонально неможливо для великих плат, більшість сучасних автоматичних маршрутизаторів значною мірою покладаються на евристичні методи та перевірку правил проектування для з'єднання компонентів друкованої плати. Хоча ці автоматичні маршрутизатори можуть працювати добре в деяких випадках, результати часто розчаровують і потребують налаштування людиною перед завершенням розробки конструкції друкованої плати. В більшості випадків, розробникам топології рекомендується використовувати автомаршрутизатори як спосіб спрощення початкових етапів, а не як повна заміна ручного трасування, натякаючи на їхні слабкі сторони. Останніми роками маршрутизація викликала великий інтерес

завдяки таким компаніям, як InstaDeep, які використовують штучний інтелект для керування алгоритмами маршрутизації. Незважаючи на це, керовані штучним інтелектом алгоритми маршрутизації все ще страждають від складності, добре працюють лише на простих двошарових платах із менш ніж 150 парами з'єднань.

Проблема розміщення друкованої плати полягає в тому, що за наявності списку з'єднань (графіка, що описує необхідні компоненти та з'єднання друкованої плати), необхідно розмістити усі компоненти на двовимірному полотні (платі) таким чином, щоб не було перекриття компонентів і можна було виконати всі з'єднання без перекриття друкованих доріжок.

В роботі, процес розміщення компонентів на друкованій платі був змодельований як проблема підкріплення навчання (RL), де алгоритмічний актор, який називається агентом, навчається виконувати завдання, взаємодіючи з середовищем. У конкретному випадку розміщення друкованої плати агент спостерігає за станом частково розміщеної плати (середовищем), а також за метаданими про наступний компонент для розміщення. Потім агент приймає рішення про розміщення, яке оновлює стан друкованої плати. Якщо розміщення не порушує жодних правил розміщення, агент отримує позитивну винагороду, і процес продовжується з оновленим станом друкованої плати та іншим компонентом, який агент має розмістити. Якщо розміщення було недійсним і розміщений компонент порушує правило дизайну, агент отримує негативну винагороду, і процес перезапущається з порожньою друкованою платою. Оновлюючи свою політику, нейронну мережу, яку агент використовує для відображення станів друкованої плати для можливих наступних розміщень, агент може з часом навчитися максимізувати свою винагороду та, отже, покращити розміщення, які він робить.

Мережа кодувальника, яка використовувалася в цьому дослідженні, була графовою нейронною мережею з двома шарами згортки графіка та останнім максимальним шаром об'єднання для перетворених функцій вузла. Починаючи з необроблених метаданих як характеристик вузла (тобто ширини компонента, висоти, x , y та обертання), мережа кодувальника оновила характеристики вузла на кожному шарі згортки графіка на основі інформації, що міститься в сусідніх компонентах вузла, таким чином вивчаючи невеликі локальні взаємодії. Через кілька рівнів цього процесу було вивчено глобальні моделі. Останній максимальний рівень об'єднання зменшив усі вузли до остаточного кодування графа: функція фіксованого розміру, що представляє всі вузли в графі. Це кодування графа мало стандартну форму і могло використовуватися для представлення широкого діапазону розмірів списку з'єднань і форм зв'язності.

На кожному рівні в мережі графової згортки (GCN) властивість вузла оновлюється, щоб відобразити інформацію, що міститься в його однопрохідних сусідах. На рис.1 простого графа «хаб-і-спиця» інформація передається від спільних вузлів до вузла-концентратора на першому рівні згортки графіка. Об'єднавши всі функції вузла, представлення графа можна звести до одного тензора фіксованого розміру (у цьому випадку розмір 3).

Ключовим атрибутом мережі кодувальника було не лише кодувати список з'єднань у вектор ознак фіксованого розміру, але й робити це таким чином, щоб надавати повну інформацію агенту політика щодо стану розміщення.

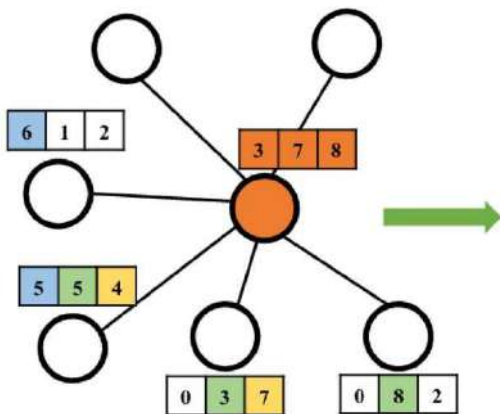


Рисунок 1 - Стандартні компоненти

Мережа політики була 5-рівневою TCNN, яка розширювала (18×1) тензор до $(10 \times 10 \times 2)$ розподілу логітів. Між кожним шаром застосовувалася нелінійна активація ReLU(). Перед входом у Transpose Convolutional Layers тензор стану було збільшено до (32×1) прихованого розміру, що дозволяє плавно змінювати форму між шарами.

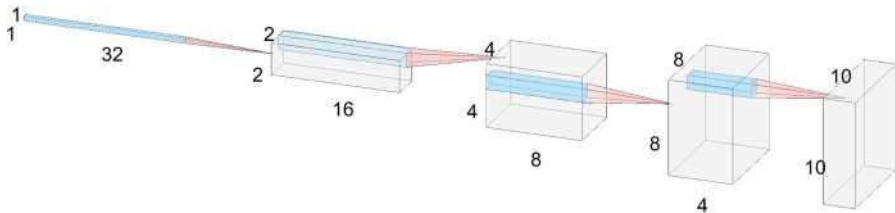


Рисунок 2 - Мережа політики

Політика агента – це мережа глибокої згортки з 5 транспонованих шарів згортки, розділених функціями активації ReLU(). Мережа політики посилює кодування стану (32×1) у тензор логітів розподілу дій ($10 \times 10 \times 2$).

У розташуванні, що наведено на рис.3, згенерованому штучним інтелектом, агент успішно розмістив 8 компонентів, у результаті чого на дошці загалом було 12 компонентів. Тут видно, що агент розміщений з меншою щільністю в нижній лівій частині дошки. Перші кілька розміщених компонентів зазвичай більше, ніж решта, і агент залишає простір для цього.

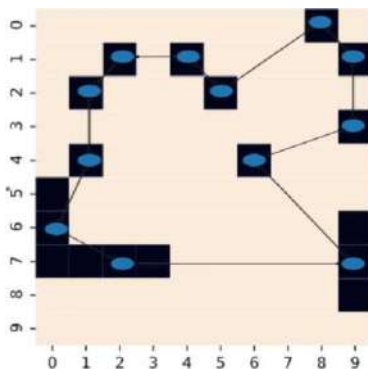


Рисунок 3 - Розташування, створене штучним інтелектом, із 12 компонентами

Хоча модифіковані методи, запропоновані в цьому дослідженні, демонструють успіх у розміщенні невеликих кілець, вони розкривають природні обмеження системи.

По-перше, попереднє навчання мережі кодувальника на наборі даних згенерованих кільцевих розміщень обмежує розміри плат, які ця система може розмістити. У цій роботі кодер був навчений на 10 компонентах, що призвело до втрати продуктивності понад 12 розміщень компонентів. Це пов'язано з тим, що кодер не в змозі витягнути важливу інформацію з місць розташування з більшою кількістю компонентів, ніж було навчено.

По-друге, фізично перевірені розміщення створюють рівень навантаження на процес RL, який можна краще вирішити за допомогою налаштування та сукупності алгоритмів розміщення. Ця напруга спричинена величезною кількістю варіантів розміщення, які може мати кожен компонент. Наприклад, кожен розміщений компонент на реальній друкованій платі визначається його розташуванням, обертанням, дзеркалом, масштабом, шаром, формою та іншими атрибутами, що залежать від варіанту використання. Кожен атрибут розміщення додає величезної складності до

простору дії, і всі вони повинні враховуватися, якщо розміщення залишатимуться фізично дійсними на всіх етапах процесу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. CoRR, abs/1707.06347, 2017