УДК 621.98

А.В. Гагалюк, канд. техн. наук, Р.А. Скляров, канд. техн. наук, доц.
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ГЕНЕТИЧНІ АЛГОРИТМИ В ГЕНЕРАТИВНОМУ ДИЗАЙНІ

A. Gagalyuk, Ph.D., R. Sklyarov, Assoc. Prof., Ph.D.

GENETIC ALGORITHMS IN GENERATIVE DESIGHN.

Технологія генеративного проектування була винайдена зовсім недавно, а саме у період 2014 – 2017 років. Найвідомішим дослідженням є спільна робота Autodesk та Airbus з реалізації проекту щодо зниження маси перегородки між пасажирським салоном і відсіком бортпровідників авіалайнера Airbus A320. Внаслідок аналізу 10 тис. варіантів конструкцій, отримано деталь масу якої вдалося зменшити на 55% у порівнянні з аналогом [1]. В основу алгоритму розрахунку було закладено поведінку одноклітинного організму – слизової цвілі, ймовірно (лат. Mycetozoa).

Проте залишається незрозумілим, яким чином написаний комп'ютерний код, котрий здатний з певною ймовірністю імітувати поведінку організму і на її основі створити абсолютно нову конструкцію.

Можна припустити, що у основу алгоритмів генеративного дизайну закладено генетичний алгоритму (GA), який є одним із найдавніших та найпопулярніших метаевристичних алгоритмів оптимізації дизайну [2]. Перший опис алгоритму еволюційних обчислень був запропонований Дж. Голландом у книзі «Адаптація в природних та штучних системах», у якій описано перший в історії генетичний алгоритм (GA). У праці [2] наводиться вирішення проблеми, описаної у книзі Д. Шиффмана, у якій описано генетичні алгоритми. Для демонстрації масштабів проблеми описано відтворення фрази В.Шекспіра to_be_or_not_to_be (рис.1). У описаній фразі міститься 18 символів, а у англ. мові включно з абеткою (26 літер), цифрами та із знаками пунктуації автор наводить 96 символів. Тобто, на місці однієї клітинки може бути 1 із 96 символів. Якщо ми підрахуємо загальну кількість можливих рішень, то отримаємо:

96^{18} = 479,603,335,372,621,236,652,373,132,533,825,536

Якщо перефразувати на українську мову «бути_чи_не_бути» (15 символів) з врахуванням тих же знаків пунктуації, але нашою абеткою (33 літери), то ми отримаємо:

110^{15} = 4,177,248,169,415,651,000,000,000,000,000

Англійський варіант містить на $4{,}796\times10^{35}$ комбінацій більше, або на 50%, а це 479,6 декліонів варіантів. Якщо припустити, що для створення та оцінки кожної опції потрібен лише 1 комп'ютерний цикл, опрацювання всіх варіантів процесора 2,6 ГГц зайняло б приблизно $58{,}5\times10^{17}$ років. Використовуючи дуже простий GA, описаний Д. Шиффманом, і записаний у36 рядках комп'ютерного коду вирішує цю проблему за 32 секунди, переглянувши лише 38000 можливих рішень. Насправді існують й інші алгоритми та більш складні варіанти GA, які дозволяють вирішити цю проблему ще швидше, проте просте порівняння доводить потужність та універсальність, яка стоїть за основними концепціями GA.

Хоча генетичний алгоритм добре підходить у вирішенні деяких дуже складних задач, сам алгоритм керується лише чотирма основними операторами.

Генерація. GA генерує набір конструкцій, які утворюють початкове «покоління». Найпоширенішим методом є випадкове вибірка конструкцій з проектного простору. У цьому випадку ми використовуємо кількість вибірки в 1000 конструкцій.

Виділення (ранжування). Далі GA вибирає, які з початкових проектів будуть використовуватися для генерації наступного покоління. Хороші конструкції, знайдені в 1-му поколінні, будуть перенесені в наступне. У такому випадку ми створюємо «спарювальний басейн», що містить найкращі конструкції відповідно до кількості літер, які вони мають спільні з цільовою фразою (рис.2).

Перехрещування. Після того, як алгоритм вибирає перспективні конструкції, він рекомбінує їх для створення нової сукупності конструкцій. Деталі процесу описано у [2].

Мутація. Вибір та **Перехрещування** гарантують, що більшість підходящих рішень кожного покоління пробиваються у наступні покоління конструкцій. Однак, лише за допомогою цих методів ми можемо легко зупинитися на найкращому рішенні. Наприклад, якби в якісь конструкції першого покоління не було б літери «е» на останньому місці, то жоден «нащадок» ніколи б не отримав цієї інформації, і правильне рішення ніколи не знайшлося б. Як і в природі, нам потрібен механізм, який може випадковим чином вводити нову інформацію у генофонд. Це робиться через оператор «мутації», який випадковим чином змінює входи випадкової кількості «нащадків» (як правило до1%) до того, як вони перейдуть до наступного покоління. У цьому випадку ми випадковим чином присвоюємо 1% вхідних даних у кожне покоління.

Застосовуючи ці оператори до послідовності поколінь, алгоритм зрештою може прийти до правильного рішення. На зображенні показана частина конструкцій 1-го та останнього поколінь. Видно, що в першому поколінні конструкції абсолютно випадкові. Однак через 38 поколінь усі конструкції мають багато правильних компонентів цілі, включаючи одну конструкцію, яка отримала її цілком правильно. У вікні праворуч показано найкращу конструкцію у кожному поколінні та її оцінку (як співвідношення правильних символів). Це показує, як алгоритм здатний поступово вдосконалювати конструкції з кожним поколінням.

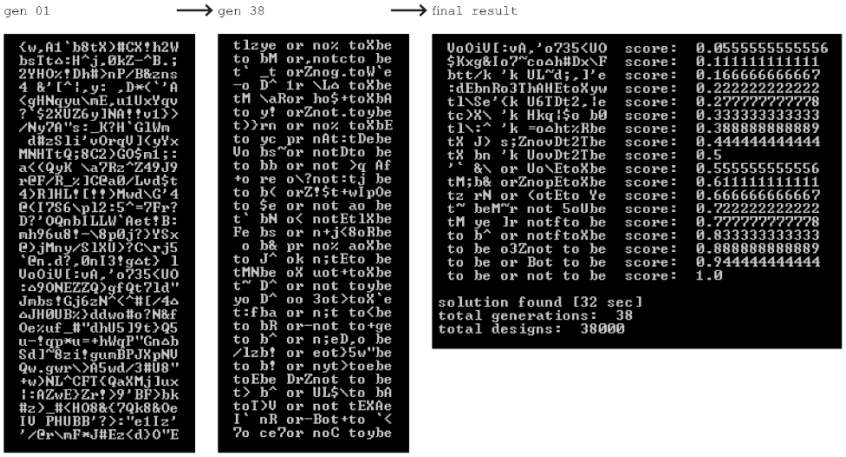

Рис. 1. Процес генетичного алгоритму (GA) для пошуку проблеми Шекспіра

Цей приклад описує дуже базовий генетичний алгоритм, який використовується для вирішення проблеми лише з однією складовою – наблизити рядок якомога ближче до цілі. Однак найцікавіші дизайнерські проблеми визначаються багатьма різними цілями, які можуть бути пов'язані між собою складними, не інтуїтивними способами. У такому випадку вирішити, яка конструкція краща не так вже й просто.

Цей приклад не відображає, зокрема найпоширенішої задачі сучасного генеративного дизайну – зменшення маси при збереженні міцнісних характеристик. Проте створює уявлення про алгоритми, які використовуються у сучасних обчислювальних системах.

На практиці найбільш типовим підходом є налаштування параметрів за допомогою експерименту, шляхом проведення декількох оптимізацій та визначення того, які параметри найкраще працюють для даної проблеми.

Література

1. From grand vision to practical product – Режим доступу до ресурсу: https://www.autodesk.com/customer-stories/airbus

2. Danil Nagy. Evolving design. – Режим доступу до ресурсу: https://medium.com/generative-design/evolving-design-b0941a17b759.