нулі.

$$C^{IV}=\begin{bmatrix}0&2&2&4&0\\1&0&1&0&3\\0&1&2&2&0\\0&1&0&1&3\\1&0&1&3&3\end{bmatrix}\rightarrow\begin{bmatrix}0&2&-&4&0\\1&0&-&0&3\\0&1&-&2&0\\-&-&-^{*}&-&-\\1&0&-&3&3\end{bmatrix}\rightarrow\begin{bmatrix}0&2&-&-&0\\-&-&-&-^{*}&-\\0&1&-&-&0\\-&-&-^{*}&-&-\\1&0&-&-&3\end{bmatrix}\rightarrow\begin{bmatrix}0&-&-&-&0^{*}\\-&-&-&-^{*}&-\\0^{*}&-&-&-&0\\-&-&-^{*}&-&-\\-&-^{*}&-&-&-\end{bmatrix}$$

У результаті отримали матрицю 2х2. Обираємо будь-яку її діагональ. Кількість виділених нулів дорівнює розмірності матриці. Задача вирішена. Переносимо виділені нулі на вихідну матрицю.

$$C=\begin{bmatrix}4&2&3&1&5^{*}\\2&3&3&4^{*}&1\\3^{*}&2&2&2&4\\5&4&6^{*}&5&3\\2&4^{*}&4&2&2\end{bmatrix}$$

Переносимо позначенні позиції у вихідну матрицю і шукаємо сумарну продуктивність

Відповідно до одержаного рішення можемо зробити висновок, що для одержання максимальної ефективності (найбільшої продуктивності), оптимальним варіантом призначення працівників на роботу буде наступний: $x_{15} = x_{24} = x_{31} = x_{43} = x_{52} = 1$, всі інші $x_{ij} = 0$, тобто перший виконавець назначається на п’яту роботу, другий - на четверту, третій на першу, четвертий – на третю, п'ятий – на другу. Сумарна продуктивність складає 3+4+6+4+5=22.

Висновки. Як бачимо із прикладів, модернізований угорський метод менш заплутаний і більш раціональний. Тому рекомендується до використання в учбовому процесі як альтернатива класичному.

Література

1. Зайченко Ю.П. Исследование операций. Учеб. пособие для студентов вузов. – К.: ВШ, 1075.-320с.

2. Вентцель Е.С. Исследование опреций. – М.:Советское радио, 1972.-552с.

3. *Гужевська Л.А., Ануфрієва Т.Г.* Методичні вказівки до виконання практичних робіт з дисципліни "Логістика" для студентів напряму "Транспортні технології".– К.: НТУ.

УДК 519.876.3

ВИКОРИСТАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ ДЛЯ ВИРІШЕННЯ Т-ЗАДАЧ

Гужевська Л.А., кандидат технічних наук
Кара О.О.
Литвин О.В.

Постановка проблеми. Задача комівояжера (комівояжер — бродячий торговець; англ. Travelling Salesman Problem, TSP; нім. Problem des Handlungsreisenden) полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай вказується, що маршрут повинен проходити через кожне місто тільки один раз, в такому разі вибір здійснюється серед гамільтонових циклів.

Існує маса різновидів узагальненої постановки задачі, зокрема геометрична задача комівояжера (коли матриця відстаней відображає відстані між точками на площині), трикутна задача комівояжера (коли на матриці вартостей виконується нерівність трикутника), симетрична та

асиметрична задачі комівояжера.

Постановка завдання. Задачі оптимізації - найбільш розповсюджений і важливий для практики клас задач. Їх приходиться вирішувати кожному з нас або в побуті, розподіляючи свій час між різними справами, або на роботі, домагаючись максимальної швидкості роботи програми чи максимальної прибутковості компанії - у залежності від посади. Серед цих задач є розв'язувані простим шляхом, але є і такі, точне рішення яких знайти практично неможливо. Відомо два основні шляхи рішення таких задач - переборний та градієнтний. Переборний метод є найпростішим. Для пошуку оптимального рішення (максимум цільової функції) потрібно послідовно обчислити значення функції у всіх точках. Недоліком є велика кількість обчислень.

Іншим способом є градієнтний спуск. Обираємо випадкові значення параметрів, а потім значення поступово змінюють, досягаючи найбільшої швидкості зросту цільової функції. Алгоритм може зупинитись, досягнувши локального максимуму. Градієнтні методи швидкі, але не гарантують оптимального рішення (оскільки цільова функція має декілька максимумів).

Генетичний алгоритм уявляє собою комбінацію переборного та градієнтного методів. Механізми кросоверу (схрещування) та мутації реалізують переборну частину, а відбір кращих рішень - градієнтний спуск. Тобто, якщо на деякій множині задана складна функція від декількох змінних, тоді генетичний алгоритм є програмою, яка за зрозумілий час знаходить точку, де значення функції знаходиться достатньо близько до максимально можливого значення. Обираючи прийнятний час розрахунку, отримуємо одне з кращих рішень, які можна отримати за цей час. Генетичні алгоритми найкраще використовувати для вирішення задачі на програмному рівні.

Робота генетичного алгоритму. Уявимо собі штучний світ, населений множиною істот, причому, кожна особина - це деяке рішення нашої задачі. Будемо вважати істоту більш пристосованою до середовища, чим краще відповідне їй рішення (чим більше значення цільової функції воно дає). Тоді задача максимізації цільової функції зводиться до пошуку найбільш пристосованої істоти. Звичайно, ми не можемо поселити у віртуальний світ усі істоти відразу, тому що їх дуже багато. Замість цього ми будемо розглядати багато поколінь, що змінюють одине одного. Тепер, якщо ми зуміємо ввести в дію природний відбір і генетичне спадкування, тоді отримане середовище буде підкорятися законам еволюції. Помітимо, що, відповідно до нашого визначення пристосованості, метою цієї штучної еволюції буде саме створення найкращих рішень. Очевидно, еволюція - нескінченний процес, у ході якого пристосованість особин поступово підвищується. Примусово зупинивши цей процес через досить довгий час після його початку і вибравши найбільш пристосовану особину у поточному поколінні, ми одержимо не абсолютно точну, але близьку до оптимальної відповідь.

Для того щоб говорити про генетичне спадкування, потрібно наділити наші особини хромосомами. У генетичному алгоритмі хромосома - це деякий числовий вектор, що відповідає параметру, який підбирається, а набір хромосом даної особини визначає рішення задачі. Які саме вектори варто розглядати в конкретній задачі, вирішує сам користувач. Кожна з позицій вектора хромосоми називається ген.

Простий генетичний алгоритм випадковим образом генерує початкову популяцію структур. Робота генетичного алгоритму уявляє собою ітераційний процес, що продовжується доти, поки не виконаються задане число поколінь або будь-який інший критерій зупинки. В кожному поколінні генетичного алгоритму реалізується відбір пропорційно пристосованості, одноточковий кросинговер і мутація. Спочатку, пропорційний відбір призначає кожній структурі імовірність Ps(i) рівну відношенню її пристосованості до сумарної пристосованості популяції:

$$P_S(i) = \frac{f(i)}{\sum_{i=1}^{n} f(i)} \qquad (1)$$

Потім відбувається відбір (із заміщенням) усіх n особин для подальшої генетичної обробки, відповідно до величини Ps(i).

При такому відборі члени популяції з більш високою пристосованістю з більшою імовірністю будуть частіше вибиратися, ніж особини з низькою пристосованістю. Після відбору, n обраних особин випадковим чином розбиваються на n/2 пари. Для кожної пари з імовірністю Pc може застосовуватися кросинговер. Відповідно з імовірністю 1-Pc кросинговер не відбувається і незмінені

особини переходять на стадію мутації. Якщо кросинговер відбувається, отримані нащадки заміняють собою батьків і переходять до мутації.

Визначимо тепер поняття, що відповідають мутації і кросинговеру в генетичному алгоритмі.

Мутація - це перетворення хромосоми, що випадково змінює одну чи декілька її позицій (генів). Найбільш розповсюджений вид мутацій - випадкова зміна тільки одного з генів хромосоми.

Кросинговер (у літературі по генетичних алгоритмах також вживається назва кросовер або схрещування) - це операція, при якій із двох хромосом породжується одна чи декілька нових хромосом.

Після того як закінчується стадія кросинговеру, виконуються оператори мутації. У кожному рядку, що піддається мутації, кожен біт з імовірністю Pm змінюється на протилежний.

Популяція, отримана після мутації записує поверх старої і цим цикл одного покоління завершується. Наступні покоління обробляються подібним чином: відбір, кросинговер і мутація.

Блок-схема генетичного алгоритму зображена на рис. 1. Спочатку генерується початкова популяція особин (індивідуумів), тобто деякий набір рішень задачі. Як правило, це робиться випадковим образом. Потім ми повинні змоделювати розмноження всередині цієї популяції. Для цього випадково відбирається декілька пар індивідуумів, відбувається схрещування між хромосомами в кожній парі, а отримані нові хромосоми втілюються в популяцію нового покоління. У генетичному алгоритмі зберігається основний принцип природного відбору - чим пристосованіше індивідуум (чим більше відповідне йому значення цільової функції), тим з більшою імовірністю він буде брати участь у схрещуванні. Тепер моделюються мутації - у декількох випадково обраних особинах нового покоління змінюються деякі гени. Потім стара популяція частково або цілком знищується і ми переходимо до розгляду наступного покоління. Популяція наступного покоління в більшості реалізацій генетичних алгоритмів містить стільки ж особин, скільки початкова, але в силу відбору пристосованість у ній у середньому вище. Тепер описані процеси відбору, схрещування й мутації повторюються вже для цієї популяції і т.д.

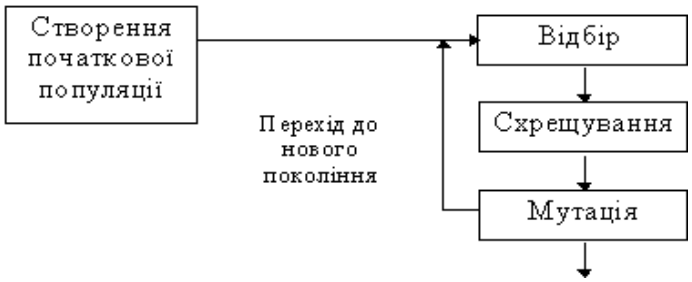

Рис. 1. Блок-схема генетичного алгоритму

В кожному наступному поколінні ми будемо спостерігати виникнення зовсім нових рішень нашої задачі. Серед них будуть як погані, так і хороші, але завдяки відбору число прийнятних рішень буде зростати. Помітимо, що в природі не буває абсолютних гарантій, і найпристосованіший тигр може загинути від рушничного пострілу, не залишивши нащадків. Імітуючи еволюцію на комп'ютері, ми можемо уникати подібних небажаних подій і завжди зберігати життя кращому з індивідуумів поточного покоління - така методика називається "стратегією елітизму".

Застосовування генетичних алгоритмів

Генетичні алгоритми в різних формах застосовуються до вирішення багатьох наукових і технічних проблем. Генетичні алгоритми використовуються при створенні інших обчислювальних структур, наприклад, автоматів або мереж сортування. У машинному навчанні вони використовуються при проектуванні нейронних мереж або керуванні роботами. Вони також застосовуються при моделюванні розвитку в різних предметних областях, включаючи біологічні (екологія, імунологія і популяційна генетика) та соціальні (такі як економіка і політичні системи) системи.

Проте, можливо найбільш популярне застосування генетичних алгоритмів - оптимізація багатопараметричних функцій. Багато реальних задач можуть бути сформульовані як пошук оптимального значення, де значення - складна функція, що залежить від певних вхідних параметрів. У деяких випадках, потрібно знайти ті значення параметрів, при яких досягається найкраще точне значення функції. В інших випадках, точний оптимум не потрібний - рішенням може вважатися будь-яке значення, краще за певну задану величину. У цьому випадку, генетичні алгоритми - часто найбільш прийнятний метод для пошуку "прийнятних" значень. Сила генетичного алгоритму полягає

в його здатності маніпулювати одночасно багатьма параметрами, що використовується в сотнях прикладних програм, включаючи проектування літаків, налаштування параметрів алгоритмів і пошуку стійких станів систем нелінійних диференціальних рівнянь.

Наприкінці можна підсумувати, що генетичні алгоритми є ефективною процедурою пошуку, що конкурує з іншими процедурами. Ефективність генетичних алгоритмів сильно залежить від таких деталей, як метод кодування рішень, операторів, налаштування параметрів, окремих критеріїв успіху. Теоретична робота, відбита в літературі, присвяченої генетичним алгоритмам, не дає підстав говорити про вироблення певних строгих механізмів для чітких передбачень.

Метод мурашиних колоній.

Мурашині алгоритми серйозно досліджуються європейськими вченими із середини 1990-х років. На сьогоднішній день вже отримано гарні результати для оптимізації таких складних комбінаторних задач, як задача комівояжера, задача оптимізації маршрутів вантажівок, задача розфарбування графа, квадратична задача про призначення, задача оптимізації сіткових графіків, задача календарного планування і т.ін. Особливо ефективні мурашині алгоритми при динамічній оптимізації процесів у розподілених нестаціонарних системах, наприклад, трафіків у телекомунікаційних мережах.

Метод мурашиних колоній заснований на взаємодії декількох *мурах* (програмних агентів, що є членами великої колонії) і використовується для вирішення різних оптимізаційних задач. Агенти спільно вирішують проблему й допомагають іншим агентам у подальшій оптимізації розв'язку.

Методи мурахи (Ant algorithms), або оптимізація за принципом мурашиної колонії, мають специфічні особливості, властиві мурахам, і використовують їх для орієнтації у фізичному просторі. Природа пропонує різні методики для оптимізації деяких процесів. Методи мурашиних колоній особливо цікаві тому, що їх можна використовувати для вирішення не тільки статичних, але й динамічних задач, наприклад, задач маршрутизації в мінливих мережах. Базова ідея методу мурашиних колоній полягає у вирішенні оптимізаційної задачі шляхом застосування непрямого зв'язку між автономними агентами. У методі мурашиних колоній передбачається, що навколишнє середовище являє собою закриту двовимірну мережу – групу вузлів, з'єднаних за допомогою граней. Кожна грань має вагу, що позначається як відстань між двома вузлами, з'єднаними нею. Граф – двонаправлений, тому агент може подорожувати по грані в будь-якому напрямку.

Агент забезпечується набором простих правил, які дозволяють йому вибирати шлях у графі. Він підтримує список табу *tList* – список вузлів, які він вже відвідав. Таким чином, агент повинен проходити через кожний вузол тільки один раз. Вузли в списку "поточної подорожі" *tList* розташовуються в тому порядку, у якому агент відвідував їх. Пізніше список використовується для визначення довжини шляху між вузлами. Справжня мураха під час переміщення по шляху залишає за собою деяку кількість *феромону*. У методі мурашиних колоній агент залишає феромон на гранях мережі після завершення подорожі.

Послідовність виконання методу. Метод мурашиних колоній виконується в наступній послідовності кроків.

Крок 1. Задати параметри методу: α – коефіцієнт, що визначає відносну значимість шляху (кількість феромона на шляху); β – параметр, що означає пріоритет відстані над кількістю феромона; ρ – коефіцієнт кількості феромона, що агент залишає на шляху, де $(1-\rho)$ показує коефіцієнт випару феромона на шляху після його завершення; Q – константу, що відноситься до кількості феромона, яку було залишено на шляху.

Крок 2. Ініціалізація методу. Створення популяції агентів. Після створення популяція агентів нарівно розподіляється по вузлах мережі. Необхідно рівномірний розподіл агентів між вузлами, щоб всі вузли мали однакові шанси стати відправною точкою. Якщо всі агенти почнуть рух з однієї точки, це буде означати, що дана точка вважається оптимальною для старту, а насправді вона такою може і не бути.

Крок 3. Рух агентів. Якщо агент ще не закінчив шлях, тобто не відвідав всі вузли мережі, для визначення наступної грані шляху використовується формула:

$$P = \frac{\tau_{ru}(t)^{\alpha} \cdot \eta_{ru}(t)^{\beta}}{\sum_{k \in J} \tau_{ru}(t)^{\alpha} \cdot \eta_{ru}(t)^{\beta}} \qquad (2)$$

де J – множина граней, ще не відвіданих агентом; $\tau_{ru}(t)$ – інтенсивність феромона на грані між вузлами r і u, які утворять k-y грань, у момент часу t; $\eta_{ru}(t)$ – функція, що представляє собою зворотну величину відстані грані.

Агент подорожує тільки по вузлах, які ще не були відвідані ним, тобто по вузлах, яких не має у списку табу *tList*. Тому ймовірність розраховується тільки для граней, які ведуть до ще не відвіданих вузлів.

Крок 4. Пройдений агентом шлях відображається, коли агент відвідає всі вузли мережі. Цикли заборонені, оскільки в метод включений список табу *tList*. Після завершення може бути розрахована *довжина шляху*. Вона дорівнює сумі всіх граней, по яких подорожував агент. *Кількість феромону*, що було залишено на кожній грані шляху i-го агенту, визначається за формулою:

$$\Delta\tau_{ru}^{i}(t)=\frac{Q}{L^{i}(t)} \tag{3}$$

де $L^{i}(t)$ – довжина шляху i-го агенту.

Результат є засобом вимірювання шляху: короткий шлях характеризується високою концентрацією феромону, а більш довгий шлях – більш низкою. Потім отриманий результат використовується для збільшення кількості феромону уздовж кожної грані пройденого i-им агентом шляху:

$$\tau_{ru}(t+1)=\tau_{ru}(t)+(\Delta\tau_{ru}^{i}(t)\cdot\rho) \tag{4}$$

де r, u – вузли, що утворюють грані, які відвідав i-ий агент.

Дана формула застосовується до всього шляху, при цьому кожна грань позначається феромоном пропорційно довжині шляху. Тому варто дочекатися, поки агент закінчить подорож і тільки потім оновити рівні феромону, у противному випадку дійсна довжина шляху залишиться невідомою. Константа ρ приймає значення між 0 і 1.

$$\tau_{ru}(t)=\tau_{ru}(t)\cdot(1-\rho) \tag{5}$$

Тому для випаровування феромону використовується зворотний коефіцієнт відновлення шляху $(1-\rho)$.

Крок 5. Перевірка на досягнення оптимального результату. Перевірка може виконуватися для постійної кількості шляхів або до моменту, коли протягом декількох запусків не було отримано повторних змін у виборі найкращого шляху. Якщо перевірка дала позитивний результат, то відбувається закінчення роботи методу (перехід до кроку 7), у противному випадку – перехід до кроку 6.

Крок 6. Повторний запуск. Після того як шлях агента завершений, грані оновлені відповідно до довжини шляху, й відбулося випаровування феромону на всіх гранях, метод виконується повторно. Список табу очищується, і довжина шляху онулюється. Після цього виконується до кроку 3.

Крок 7. Кінець. Визначається кращий шлях, що і є розв’язком.

Висновки. Прості методи розв'язання задачі комівояжера: повний лексичний перебір, жадібні алгоритми (метод найближчого сусіда, метод включення найближчого міста, метод найдешевшого включення), метод мінімального остовного дерева. На практиці застосовуються різні модифікації ефективніших методів: метод гілок і меж та генетичних алгоритмів.

Усі ефективні (такі, що скорочують повний перебір) методи розв'язання задачі комівояжера — евристичні. У більшості евристичних методів знаходиться не найефективніший маршрут, а наближений розв'язок. Користуються популярністю так звані any-time алгоритми, тобто алгоритми, що поступово покращують деякий поточний наближений розв'язок. Евристичні алгоритми найбільш ефективні для NP-повних задач.

Література

1. *Пожуева И.С., Субботин С.А., Олейник А.А.* Эволюционная оптимизация многомерных функций // Нові матеріали і технології в металургії та машинобудуванні. – 2006. – № 1. –

С. 70 - 72.
2. *Colorni A., Dorigo M., Maniezzo V., Trubian M.* Ant system for job-shop scheduling // Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL). – 1994. – №34. – P. 39–53.
3. *Costa D., Hertz A.* Ants can colour graphs // Journal of the Operational Research Society. – 1997. – №48. – P. 295–305.

УДК 656.07:658.51

УПРАВЛІННЯ ПРОЕКТАМИ МІСЬКОГО ТРАНСПОРТУ В УМОВАХ СТАЛОГО РОЗВИТКУ (НА ПРИКЛАДІ РЕСПУБЛІКИ ФРАНЦІЯ)

Далека В.Х., доктор технічних наук
Штомпель Г.Е.
Далека М.В.

Постановка проблеми. Забезпечення якості пасажирських перевезень у містах нашої країни потребує вирішення комплексної проблеми, що включає організаційні, технічні, економічні та соціальні складові. Організаційно-правову основу системи управління транспортом визначено в Законі України «Про транспорт», який визначає правові, організаційні та соціально-економічні засади функціонування транспорту на ринку транспортних послуг і спрямований на створення сприятливих умов для його розвитку, задоволення потреб у доступних, якісних і безпечних перевезеннях [1]. Але суспільна діяльність в міських середовищах ускладнюється в контексті політики, що намагається відповідати поставленим інтересам, а саме таким як сталий розвиток. Загальноприйняте визначення якої таке: *„Розвиток означає задоволення потреб людства, передбачає щоб бути сталим не чинити собі власних перешкод. Наслідки та обрані орієнтири не повинні призводити до глухих кутів соціальних, економічних, біологічних та екологічних”* [2].

Аналіз нормативних та законодавчих документів дозволяє визначити яким чином ураховані ці інтереси в прийнятті рішень, а саме в розвитку транспортних проектів.

Робота над удосконаленням нормативно-правової бази на Україні у сфері транспортних послуг триває. Вона спрямована на створення ефективної економічної моделі функціонування галузі, забезпечення раціонального використання транспортних ресурсів, але не вирішує багатьох питань, оскільки не є достатньо комплексною і не передбачає основних принципів управління проектами [1]. Корисним в цьому є досвід республіки Франція [2 - 17].

Метою даної статті є аналіз практики оцінки проектів міського транспорту на прикладі Франції, результати якого доцільно використовувати для підвищення якості транспортного обслуговування населення України.

Основний зміст. Міністерство Транспорту, інфраструктури, туризму і морських справ в спільній роботі з Національної радою з питань оцінки (НРО) і Генеральної ради мостів і мостових, відіграли певну роль у практиці і методологічному розвитку оцінки в галузі транспорту. НРО була затверджена декретом в 1998 році, замінив Наукову раду з питань оцінки і мала за мету щорічне впровадження програми оцінки підпорядкованої постановам прем’єр-міністра. Після реорганізації в червні 2009 року міністерство отримало нову назву, а саме Міністерство екології, енергетики, сталого розвитку і морських справ [3].

Протягом більше двох десятиліть процес оцінки базувався на наступних принципах:

- інституціоналізації;
- залучення громадськості до прийняття рішень, у тому числі через публічні дискусії;
- розширення та напрацювання метрологічної бази в галузі транспорту у зв’язку із зростаючими проблемами для навколишнього середовища.

У порівнянні із ситуацією на початку 80-х років концепція оцінки проектів міського транспорту на практиці набула суттєвих змін.

Скласти найбільш об’єктивну оцінку було предметом багатьох спроб, дві з яких набули перевагу починаючи з 1970 р.

Першою великою спробою була Раціоналізація вибору бюджетних рішень (РВБР). РВБР - один з різновидів фінансового програмування, головними цілями якого є поліпшення управління державними коштами, підвищення ефективності витрат державних коштів, припинення на ранній стадії підготовки малоефективних проектів і програм шляхом використання системи варіантів,