УДК 004

ДОСЛІДЖЕННЯ ТА РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ ОБРАННЯ ЗАМОВЛЕНЬ НА ВИКОНАННЯ ПОСЛУГ СИСТЕМНИМ АДМІНІСТРАТОРОМ КОМП'ЮТЕРНИХ МЕРЕЖ

Збаращук П. В., Грибков С. В., Сєдих О. Л., Доля С. О.
Національний університет харчових технологій, Київ, Україна
E-mail: p.v.zbarashchuk@gmail.com

Research and Development of Information Support System for Selecting Service Orders by a Computer Network System Administrator

The paper proposes the creation of a decision support system for the system administrator, which is based on the backpack method and modified genetic algorithms. Using the created system will allow the system administrator to make quick decisions when choosing and planning their work for a given period of time.

Будь-яка сучасна фірма не обходиться без використання інформаційних технологій. Варто зазначити, що чим більшою є компанія, тим більшою є ймовірність існування власної комп'ютерної мережі, яку необхідно постійно обслуговувати. Залежно від розмірів компанії в її структурі може існувати окремий відділ системних адміністраторів, які обслуговують комп'ютерну мережу та інформаційні системи. У випадках, коли одночасно виникають задачі, пов'язані з усуненням позаштатних ситуацій, слід визначати, яку з проблем необхідно усунути першочергово. Найгірший варіант, коли необхідно усунути декілька проблем одночасно та є обмеження щодо виконавців. В цьому випадку виникає проблема прийняття рішення щодо послідовності виконання таких задач або відмови від їхнього розв'язання в призначений час. У роботі пропонується створення системи підтримки прийняття рішень для системного адміністратора, в основу якої покладено метод рюкзака та модифіковані генетичні алгоритми.

Використання задачі про пакування рюкзака обрано тому, що адміністратори та обслуговуючий персонал обирають собі задачі в залежності від своєї кваліфікації. У випадку, коли замовлення є терміновим, менеджер це замовлення відмічає як пріоритетне для певного виконавця.

Необхідно зазначити, що така задача належить до числа широко відомих задач комбінаторної оптимізації [1]. Найчастіше ця задача розв'язується за необхідності розподілення обмеженої кількості ресурсів для отримання найбільшої сумарної користі, як при завантаженні човнів або літаків, вибору багажів для оптимального завантаження транспортного засобу тощо [2]. При застосуванні до нашої задачі умовою пошуку є знаходження такого розподілу між усіма виконавцями робіт, при якому вони отримають максимальну користь від виконання замовлень, що в кінцевому результаті буде стимулювати їх як виконавців.

Для випадків, коли вхідної інформації для розв'язання задачі достатньо, а розмірність не дуже велика, доцільно застосовувати алгоритм розв'язання, наведений у роботі [3]. Вхідні дані (ВД) подані як масив, що містить цілу вагу W та матеріальну цінність P предметів $W(1...N) > 0$ *і* $P(1...N) > 0$, де N — число предметів і $C > 0$ — місткість рюкзака. Вихідні дані, які являють собою масив логічних даних $X(1...N)$, де $X(i) = 1$, якщо предмет з номером i входить у розв'язок, і $X(i) = 0$, якщо предмет із номером i не входить у розв'язок.

START:

Етап 1. Сортування вхідних даних у порядку спадання окремої вартості предметів: $P(1)/W(1) \geq P(2)/W(2) \geq ... \geq P(i)/W(i) \geq ... \geq P(N) / W(N)$, де $P(i) > 0$ — вартість предмету i, $W(i) > 0$ — вага предмета i. У масиві $X(1...N)$ всі елементи початково дорівнюють 0. Для зменшення потреб у пам'яті для алгоритму визначають мінімальну вагу в наборі вхідних даних $W\_min = min(W)$.

Етап 2. Ініціалізація робочих масивів — створюємо масив дійсних чисел LP розмірністю $(W\_min... C)$ і масив цілих чисел LCr розмірністю $(W\_min... C)$. Заносимо в масив LP і LCr дані першого елемента з відсортованого списку вхідних даних, де $P(1)$ — вартість і $W(1)$ — вага першого предмета у відсортованому списку вхідних даних.

Етап 3. Заповнення робочих масивів. Нехай $W(i)$ та $P(i)$ — вага та вартість поточного елемента вхідного набору. Створюємо порожній масив дійсних чисел *Clone* розмірністю $(W\_min...C)$. Вносимо до масиву *Clone* вартість поточного елемента вхідного масиву. Копіюємо в масив *Clone* ненульові дані з масиву LP, додаючи вартість $P(i)$ поточного елемента та збільшуючи його індекс на його вагу $W(i)$ за умови, що індекс *Clone* не перевищить місткості рюкзака C. Проводимо модифікацію масивів LP, LCr на основі даних масиву *Clone*. Оновлюємо в масивах LP та LCr тільки ті елементи, вартість яких у *Clone* більша, ніж у LP.

Етап 4. Формування результату, зворотний спуск. У масиві LP знаходимо максимальне значення вартості $Pmax = MAX(LP)$, що і буде оптимальним розв'язком. Індекс знайденого у масиві елемента дорівнює вазі розв'язку, позначимо його Wr, тобто $LP(Wr) = Pmax$. Записуємо перший знайдений елемент у X.

Повторюємо етапи 2, 3, 4 (тільки на етапі 2 масиви LP та LCr не створюємо, а заповнюємо нулями). Повторювати етап 1 не треба. Це по суті рекурсія, але через попереднє сортування вхідних даних вона буде не глибокою. На деяких наборах вхідних даних рекурсії взагалі може не бути. При повторенні розрахунків розглядаємо ті вхідні дані, індекс яких менше від $LCr(Wr)$ і знижуємо необхідний розмір рюкзака до досягнутої ваги Wr.

Кінець. Вартість знайденого розв'язку буде розраховано як $\Sigma\, P(i)\, X(i)$, а також вага $\Sigma\, W(i)\, X(i)$.

Правильність розрахунку підсумкової вартості рюкзака легко доводиться за індукцією. Відновлення оптимального набору предметів теж не викликає труднощів.

Варто відзначити наступні зауваження, що наводились у роботах [3–4]. Загальна складність наведеного алгоритму складається зі складності сортування

вхідних даних і складності виконання етапу 3 алгоритму (з урахуванням числа ітерацій). Час роботи 3-го етапу пропорційний до кількості предметів на місткість рюкзака ($N * C$). Наперед визначити кількість ітерацій досить складно. Число ітерацій може варіюватися від 0 до числа предметів у розв'язку $X(i)$. На кожній ітерації, що виникає на етапі 4, обсяг обчислень на етапах 2, 3 зменшується. Верхня оцінка тимчасової складності всього алгоритму визначається $N * C *$ (*число ітерацій* + 1).

Потреба алгоритму в пам'яті пропорційна до місткості рюкзака C і залежить від кількості предметів у вхідному наборі даних N, що вигідно відрізняє його від методу динамічного програмування. Внутрішні цикли етапу 3 легко виконуються паралельно. При великому розкиді питомої вартості предметів, якщо на етапі 3 алгоритму у верхній частині масиву LP перестають відбуватися зміни, можна переривати етап 3 і не розглядати предмети, що залишилися, з низькою питомою вартістю.

Якщо місткість рюкзака досить велика, так що масиви розмірності не можуть бути створені з технічних причин або ваги предметів є дійсними числами, то запропонований алгоритм може бути легко модифіковано шляхом заміни масивів на зв'язані списки [4].

Якщо задача має великий обсяг, доцільно використати генетичний алгоритм і його модифікації, адже вони не зводяться до безладного блукання в пошуковому просторі допустимих розв'язків завдяки можливості ефективного використання досвіду, набутого кожною популяцією у визначенні нової області пошуку розв'язків, у якій передбачається поліпшення значення цільової функції [5–8]. Механізм кожного генетичного алгоритму завжди складається з трьох основних операторів [5–7]: репродукція — процес, у якому хромосоми обираються з кращим значенням цільової функції; кроссовер — схрещування батьківських пар, генерація нащадків; мутація — дія випадкових чинників.

Задача кодується так, щоби її розв'язок міг бути поданий у вигляді масиву, подібного до інформації щодо складу хромосоми. Випадковим чином у масиві створюється деяка кількість початкових елементів-«осіб», або початкова популяція. Особи оцінюються з використанням функції застосовності, в результаті якої кожній особі присвоюється певне значення застосовності, яке визначає можливість виживання особи. Після цього з урахуванням отриманих значень застосовності обираються особи, допущені до схрещення (селекція). До осіб застосовуються генетичні оператори. Особи наступного покоління також оцінюються застосуванням генетичних операторів, і виконується селекція та мутація. Так моделюється еволюційний процес, що продовжується декілька життєвих циклів (поколінь), доки не буде досягнуто критерію зупинки алгоритму [5–7]. Таким критерієм може бути: знаходження глобального або наближеного розв'язку; закінчення заданої кількості поколінь на еволюцію; закінчення часу, заданого на еволюцію.

На основі розглянутих методів створена інформаційна система підтримки обрання замовлень на виконання послуг системного адміністратора комп'ютерних мереж. Систему створено з використанням Python, Django та CSS. Такий вибір має як переваги, так і недоліки. До переваг належать:

мінімальні витрати для отримання готового продукту; повний контроль над створеним додатком; можливість повного його адміністрування. Додаток має надзвичайно малий термін окупності, оскільки враховуються лише трудові години співробітника, який уже працює на підприємстві, та вартість оренди хост-сервера. Створена система дозволяє обирати замовлення для їх виконання адміністратору комп'ютерної мережі чи цілій команді. На етапі обрання набору замовлень на виконання відбувається обрання тільки тих замовлень, які дозволять отримати найкращий ефект для виконавця у вигляді кінцевої оплати, але це не означає, що інші замовлення не буде виконано. Спочатку обираються критичні та більш важливі замовлення, а потім — менш критичні та вартісні. За рахунок людино-машинного інтерфейсу є змога скоригувати будь-який набір замовлень із власних міркувань адміністратору чи менеджеру, що розподіляє роботи між виконавцями. Використання створеної системи забезпечить системному адміністратору можливість швидко приймати рішення при обранні та плануванні своїх робіт на заданий проміжок часу.

Практичне значення роботи полягає в тому, що створену система та її модулі можна використати для створення web-орієнтованих систем підтримки прийняття ефективних рішень при управлінні підприємствами з надання різних послуг, що дасть змогу вивести весь процес управління на якісно новий рівень.

Література

1. Левитин А. В. (2006) 'Метод грубой силы: Задача о рюкзаке', *Алгоритмы: введение в разработку и анализ = Introduction to The Design and Analysis of Aigorithms*, М.: Вильямс, с. 160–163.
2. Романовская А. М., Мендзив М. В. (2010) *Динамическое программирование: Учебное пособие.* Омск, 58 с.
3. Пападимитриу Х., Стайглиц К. (1985) Комбинаторная оптимизация: Алгоритмы и сложность, М.: Мир, 510 с.
4. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. (2014) Алгоритмы: построение и анализ, К.: Вильямс, 1328 с.
5. Poli R., Langdon W. B., McPhee N. F. (2008). *A Field Guide to Genetic Programming* [online], Lulu.com, URL: http://www.gp-field-guide.org.uk.
6. Gendereau M., Potvin J.-I. (2019) *Handbook of Metaheuristic,* 3rd ed, Springer International Publishing AG, 604 pp.
7. Hrybkov S., Kharkianen O., Ovcharuk V., Ovcharuk I. (2020) 'Development of Information Technology for Planning Order Fulfillment at a Food Enterprise', *Eastern-European Journal of Enterprise Technologies*, Vol. 1(103), pp. 62–73.
8. Garg R., Mittal S. (2014) 'Optimization by Genetic Algorithm', *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 4(4), pp. 587–589.
9. Hrybkov S., Oliinyk H., Litvinov V. (2018) 'Web-oriented decision support system for planning agreements execution', *Eastern-European Journal of Enterprise Technologies*, vol. 3(2), pp. 13–24.