Інформатика та методика її викладання

Данилюк К.В., Величко В.Є.

[1]Студентка 5 курсу магістратури фізико-математичного факультету СДПУ,
[2]Завідувач кафедри алгебри СДПУ

Наближені методи розв'язку задачі про рюкзак

В статті розглянута NP-повна задача про рюкзак. Наведені різноманітні методи її розв'язання. Наведені основні принципи перспективних методів – генетичний алгоритм та алгоритм мурашиної колонії.

Ключові слова: *задача про рюкзак, генетичний алгоритм, алгоритм мурашиної колонії.*

Математична постановка задачі. Нехай задана скінчена множина $Q=\{q_1,q_2,\ldots,q_n\}$, для кожного $q_i \in Q$ відома вартість c_i і визначений об'єм a_i. Також є рюкзак об'ємом B. Необхідно запакувати рюкзак так, щоб загальна вартість запакованих предметів була якнайбільшою, а їх загальний об'єм не перевищував B. Традиційно вважають, що c_i, a_i, B – цілі невід'ємні числа.

Введемо двійкові змінні $x_1, x_2, \ldots, x_n$:

- $x_i = 1$, якщо предмет $q_i \in Q$ обраний для упаковки;
- $x_i = 0$ в протилежному випадку.

Тоді задача про рюкзак зводиться до наступної задачі лінійного цілочисельного програмування з булевими змінними: знайти такі значення змінних $x_1, x_2, \ldots, x_n$, при яких досягається максимум суми

$$W = \sum_{i=1}^{n} c_i x_i, \tag{1}$$

і виконується обмеження

$$\sum_{i=1}^{n} a_i x_i \le B. \tag{2}$$

Якщо є тільки одне обмеження виду (2), то задачу про рюкзак називають одновимірною, в протилежному випадку – багатовимірною.

Ця задача має складність ***NP-повної*** задачі, тобто її можна розв'язати за поліноміальний час на недетермінованій машині Тюрінга.

Огляд відомих методів розв'язання задачі. ***NP-повні*** задачі, які часто виникають на практиці, настільки важливі, що відмовитися від їхніх розв'язків неможливо. Звичайно, немає надій побудувати для них поліноміальний алгоритм. Однак це ще не означає, що з даною задачею взагалі нічого не можна зробити. По-перше, може виявитися, що деякий експонентний алгоритм, наприклад, переборний, працює прийнятний час на реальних даних. По-друге, можна спробувати знайти поліноміальний алгоритм, що дає не оптимальний розв'язок задачі, а близький до нього, тобто наближений розв'язок.

Знайти наближений розв’язок може бути цілком достатньо для практичного застосування.

До наближених методів для задачі про рюкзак відносять:

- жадібні алгоритми;
- алгоритми мурашиної колонії;
- генетичні алгоритми та ін.

Жадібний алгоритм для задачі про рюкзак полягає в наступному:

- спочатку множина предметів Q упорядковується за зменшенням «питомої цінності» (або ціни одиниці ваги) предметів, тобто так, щоб

$$\frac{c_1}{a_2} \geq \frac{c_2}{a_3} \geq \ldots \geq \frac{c_n}{a_n};$$

- потім, починаючи з порожньої множини, до наближеного розв’язку Q' (спочатку воно порожнє) послідовно додаються предмети з упорядкованої множини Q;
- при кожному черговому додаванні предмета в рюкзак виконується перевірка, чи не перевершує/перевищує? його вага величини того об'єму рюкзака, що залишився;
- закінчення перегляду всіх предметів завершує процес побудови наближеного розв’язку задачі про рюкзак.

Алгоритм мурашиної колонії базується на аналізі поведінки мурах. Тобто алгоритм виконує ті самі дії, які можуть виконувати мурахи під час пошуку шляхів до предмету. Для кожного мурахи дія взяти предмет залежить від трьох складових: пам’яті мурахи, важливості та віртуального сліду феромону.

Пам'ять мурахи – це список взятих мурахою предметів, які взяти повторно неможливо. Також в список необхідно включити ті предмети, взявши які ми порушуємо обмеження за об’ємом рюкзака. Позначимо через j_k список предметів, які k-та мураха може взяти на ітерації t.

Важливість – величина, обернена об’єму предмета: $\eta = 1/a_j$, a_j – об’єм i-го предмету. Важливість – це локальна інформація, яка виражає евристичне бажання взяти предмет, чим менше предмет, тим більше бажання покласти його в рюкзак.

Віртуальний слід феромону τ_j на предметі j є підтверджене досвідом мурахи бажання взяти предмет j.

Важливу роль в мурашиних алгоритмах відіграє ймовірносно-пропорційне правило, яке визначає ймовірність k-го мурахи взяти предмет j на ітерації t:

$$P_{j,k}(t) = \begin{cases} \dfrac{\left[\tau_j(t)\right]^{\alpha} \cdot \left[\eta_j\right]^{\beta}}{\sum\limits_{j \in J_k} \left[\tau_j(t)\right]^{\alpha} \cdot \left[\eta_j\right]^{\beta}}, & j \in J_k, \\ 0, & j \notin J_k \end{cases}$$

де α, β – два параметра, що задають вагу сліду феромону і важливість при виборі маршруту. Після того, як всі мурахи наповнили свої рюкзаки, відбувається зміна сліду феромону на предметах. Мураха k відкладає на предметі j наступну кількість феромону:

$$\Delta\tau_{j,k}(t) = \begin{cases} q \cdot f(T_k(t)), \text{ якщо } j \in T_k(t), \\ 0, \text{ якщо } j \notin T_k(t), \end{cases}$$

де $T_k(t)$ – набір предметів у k-го мурахи на ітерації t; $f(T_k(t))$ – важливість цього набору предметів; q – параметр.

Для дослідження всього простору предметів необхідно забезпечити випаровування феромону за правилом: $\tau_j(t+1) = (1-p)\cdot\tau_j(t) + \Delta\tau_j(t)$, де $\Delta\tau_j = \sum_{k=1}^{m} \Delta\tau_j(t)$, m – кількість мурах в колонії, $p \in [0,1]$ – коефіцієнт випаровування феромону.

На початку оптимізації кількість феромону приймають рівною невеликому додатньому числу. Кількість мурах можна назначити рівну кількості предметів.

Генетичний алгоритм базується на еволюційних принципах спадковості, мінливості і природного відбору. Загальну схему генетичного алгоритму можна представити наступним чином:

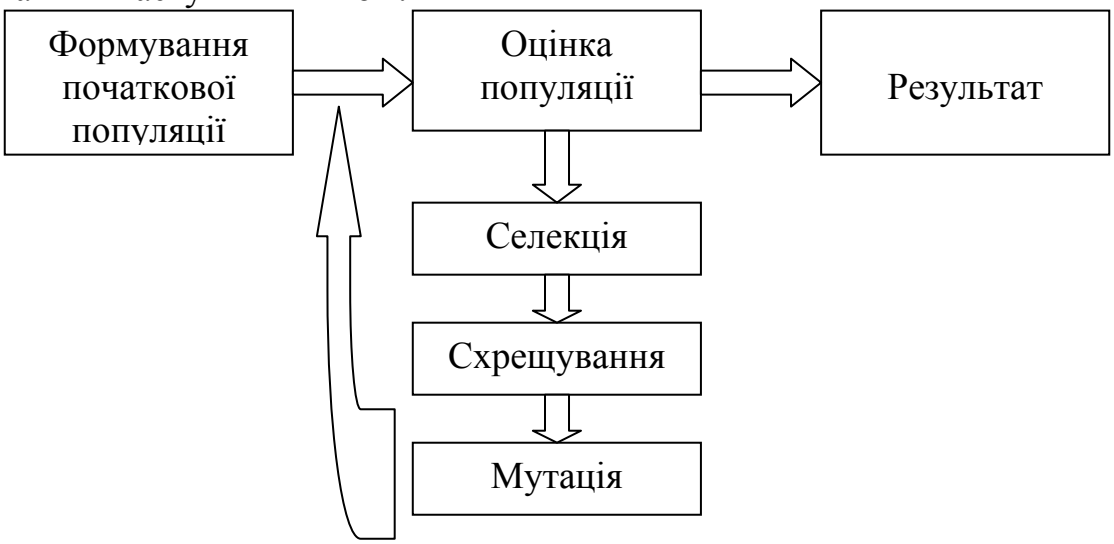

Рис. 1. Загальна схема генетичного алгоритму.

Генетичний алгоритм працює з популяцією особин, в хромосомі (генотип) кожної з яких закодовано можливий розв'язок задачі (фенотип). На початку роботи алгоритму популяція формується випадковим чином (блок «Формування початкової популяції»). Для того, щоб оцінити якість закодованих розв'язків, використовують функцію пристосованості, яка необхідна для обчислення пристосованості кожної особини (блок «Оцінювання популяції»). За результатами оцінювання особин найбільш пристосовані з них вибираються (блок «Селекція») для схрещування. В результаті схрещування вибраних особин за допомогою вживання генетичного оператора кросовера

створюється потомство, генетична інформація якого формується в результаті обміну хромосомною інформацією між батьківськими особинами (блок «Схрещування»). Створені нащадки формують нову популяцію, причому частина нащадків мутує (використовується генетичний оператор мутації), що виражається у випадковій зміні їх генотипів (блок «Мутація»). Етап, що включає послідовність «Оцінювання популяції» – «Селекція» – «Схрещування» – «Мутація», називається поколінням. Еволюція популяції складається з послідовності таких поколінь.

Для кодування інформації скористаємося наступним. Кожна хромосома представляє собою бітовий рядок, в якому закодовано розв'язок задачі. Тобто, якщо певний предмет ми будемо брати до рюкзака, то для цього предмету поставимо у відповідність 1, якщо предмет не беремо до рюкзака, то такому предмету поставимо у відповідність число 0.

Формування початкової популяції, як правило, формується випадковим чином. При цьому гени ініціалізувалися випадковими значеннями.

Оцінювання популяції необхідне для того, щоб виявити більш пристосованих і менш пристосованих особин. Для підрахунку пристосованості кожної особини використовується функція пристосованості (цільова функція)

$$f_i = f(G_i),$$

де $G_i = \{g_{ik} \mid k = 1,2,\ldots,N\}$ – хромосома i-ої особини, g_{ik} – значення k-го гена i-ої особини, N – кількість генів в хромосомі.

Селекція (відбір) необхідна, щоб вибрати більш пристосовані особини для схрещування. Існує безліч варіантів селекції, опишемо найбільш відомі з них.

Селекція рулетки. У даному варіанті селекції вірогідність, що i-та особина візьме участь у схрещуванні p_i, пропорційна значенню її пристосованості f_i і дорівнює

$$p_i = \frac{f_i}{\sum_j f_j}.$$

Процес відбору особин для схрещування нагадує гру в «рулетку». Круг рулетки ділиться на сектори, причому площа i-го сектора пропорційна значенню p_i. Після цього n разів «обертається» рулетка, де n – розмір популяції, і по сектору, на якому зупиняється рулетка, визначається особина, вибрана для схрещування.

Селекція усіканням. При відборі усіканням після обчислення значень пристосованості для схрещування вибираються $l \cdot n$ кращих особин, де l – «поріг відсікання», $0 < l < 1$, n – розмір популяції. Чим менше значення l, тим сильніше тиск селекції, тобто менше шансів на виживання у погано пристосованих особин. Як правило, вибирають l в інтервалі від 0,3 до 0,7.

Турнірний відбір. У разі використання турнірного відбору для схрещування, як і при селекції рулетки, відбираються n особин. Для цього з популяції випадково вибираються t особин, і найбільш пристосована з них допускається до схрещування. Говорять, що формується турнір з t особин, t – розмір турніру. Ця операція повторюється n разів. Чим більше значення t, тим більше тиск селекції. Варіант турнірного відбору, коли $t = 2$, називають бінарним турніром. Типові значення розміру турніру $t = 2,3,4,5$.

Відібрані в результаті селекції особини (батьківські) схрещуються і дають потомство. Хромосоми нащадків формуються в процесі обміну генетичною інформацією (із застосуванням оператора кросовера) між батьківськими особинами. Створені таким чином нащадки складають популяцію наступного покоління. Розглядатимемо випадок, коли з безлічі батьківських особин випадковим чином вибираються 2 особини і схрещуються з вірогідністю P_C, внаслідок чого створюються 2 нащадки. Цей процес повторюється до тих пір, поки не буде створено n нащадків. Вірогідність схрещування P_C є одним з ключових параметрів генетичного алгоритму і в більшості випадків її значення знаходиться в діапазоні від 0,6 до 1.

Кросовер працює наступним чином. Для одноточкового кросовера вибирають точку розриву і хромосоми батьків обмінюються значеннями починаючи з цієї точки і до кінця. Для двоточкового кросовера в хромосомах батьків випадково вибирають дві точки. Потім хромосоми батьків обмінюються між собою генами, що знаходяться між вибраними точками.

Оператор мутації використовується для внесення випадкових змін до хромосом особин. Це дозволяє «вибиратися» з локальних екстремумів і, тим самим, ефективніше досліджувати простір пошуку. Так само, як і для оператора кросовера, існує вірогідність вживання мутації P_M.

Таблиця 1

№ п/п	Оцінка	Метод гілок і кордонів	Метод динамічного програмування	Генетичний алгоритм
1	**середня точність**	0.9986	N/A	0.8258
	кількість обчислень	1253	1126877	1000
2	**середня точність**	0.9995	N/A	0.9026
	кількість обчислень	3103	3235522	1000
3	**середня точність**	0.9998	N/A	0.9661
	кількість обчислень	121963	6074444	1000
4	**середня точність**	0.9997	N/A	0.9942
	кількість обчислень	83676	8132412	1000

У серії експериментів розглядались окремі завдання, які вирішувались методом гілок і кордонів, методом динамічного програмування і генетичним алгоритмом. Експеримент показав, що для знаходження оптимального розв’язку класичні алгоритми використовують більшу кількість обчислень, ніж генетичний для популяції. Проте, середня точність класичного алгоритму вища.

Отримані результати дозволяють говорити про необхідність пошуку більш точних наближених методів розв’язання задачі про рюкзак. Одним з таких напрямків є використання так званого «методу досвіду» як для генетичного алгоритму, так і для алгоритму мурашиної колонії.

Література

1. *Кормен Т.* Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М.: МЦНМО: БИНОМ. Лаборатория знаний, 2004.
2. *Сигал И.Х., Иванова А.П.* Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. Учебное пособие. – М.: Физматлит, 2002.
3. *Вороновский Г. К.* Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. – Х.: ОСНОВА, 1992.
4. *Макконнелл Дж.* Основы современных алгоритмов. – М.: Техносфера, 2004.
5. *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского. – М.: Горячая линия – Телеком, 2004.
6. *Неймарк Е.А.* Решение нестационарной задачи о ранце при помощи генетического алгоритма. / Е.А. Неймарк // Вестник ННГУ. – 2006. – Вып.3(32). – С.133-137.
7. *Штовба С.Д.* Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. – 2004. – 4(4).