ДКР на тему

«Генетичний алгоритм розв’язання задачі прорюкзак»

Виконав студент групи ІС-72

Гороховський Іван

ЭТАП I (теоретичний)

Для вирішення задачі комбінаторної оптимізації про рюкзак будемо використовувати генетичний алгоритм. Відповідь будемо представляти у вигляді бінарного вектора X, в якому на i-му місті стоїть 1 якщо ми беремо i-ту реч і 0 якщо не беремо.

Обмеженням данній задачі є максимальна вага яку ми можемо покласти у рюкзак, тобто якщо W - вектор ваги предметів, а w - максимальна допустима вага, то:

$$X * W^T \leqslant w$$

Цільовою функцією є максимізація вартості предметів рюкзака, тобто якщо P - вектор вартостей предметів, то

$$Z = X * P^T \rightarrow \max$$

1) Сформувати популяцію без повторів

Для формування початкової популяції (матриці G) будемо використовувати наступний алгоритм:
Випадковим чином заповнемо вектор X, далі перевіримо якщо вектор з такими самими значеннями вже присутній, то будемо генерувати новий.

2) “Тіло” алгоритму

2.1 Выбір батьків

Для кожного наступного нащадка будемо використовувати аутбридінг.

Аутбридинг — первый родитель выбирается случайно, а вторым выбирается такой, который наименее похож на первого родителя.

Також зауважемо, що для генерації нащадків ми не будемо повторно використовувати одних і тих самих батьків.

2.2 Створення нащадків

Для створення нащадків будемо використовувати наступний алгоритм:
Позначимо X_1 - першого батька, X_2 - другога батька, X - вектор

нащадка, m_1 - значення метрики для першого батька, m_2 - значення метрики для другого батька:

Якщо $X_{1i} = X_{2i}$, тоді $X_i := X_{1i}$, інакше будемо використовувати випадковий підхід для того щоб не “застряти” в локальному екстремумі.

$$P(X_i := X_{1i}) = \frac{m1}{m1 + m2}$$
$$P(X_i := X_{2i}) = \frac{m2}{m1 + m2}$$

Таким чином ми будемо більш схилятися до значення кращого з батьків, але залишимо елемент стохастики.

2.3 Перевірка допустимості нащадка і його реанімація

2.3.1 Перевірка допустимості нащадка

В нашій задачі ми маємо лише одне обмеження:

$X * W^T \leqslant w$

Якщо воно виконується - нащадок є допустимим. Інакше переходимо до пунтку 2.3.2.

2.3.2 Реанімація нащадка

Для недопутимого нащадка X, візьмемо всі об’єкти де $x_i = 1$ і відсортуємо їх за вартістю. Поки не буде виконуватись умова допустимості нащадка X, будемо викидати найменш цінну реч.

2.4 Мутація нащадка

Якщо L - довжина вектора X, то з ймовірністю в $\frac{1}{4L}$ і-ій клітинці може замінитись значення на протилежне.

2.5 Локальне покращення

Застосуємо локальне покращення - будемо додавати до рюкзака предмети, що залишилися, починаючи з найменшої маси, до тих пір, поки не буде порушена умова допустимості.

2.6 Оновлення популяції

З поточної популяції залишаємо N найкращих за значенням ЦФ нащадків.

ЭТАП II (практичний)

3.1 Задати умову персональної задачі

- Кількість предметів (L) = 12

- Вектор ваг W:
[8, 5, 4, 4, 15, 18, 6, 10, 17, 12, 9, 16]

- Вектор вартості P:
[10, 7, 23, 8, 14, 28, 11, 15, 14, 15, 20, 23]

- Максимальна вага рюкзака (w) = 50

- Кількість осіб популяції (N) = 15

- Ймовірність мутації, згідно формули вище (p) = 2.1%

3.2 Рішення персональної задачі

Для розв’язку персональної задачі, мною була написана програма на мові Python.

Ссилка на код:

https://github.com/doctorblinch/Backpack-generative-algorithm

Приклад розв’зку:

У початковому стані, ми маємо деякі значення які є більшими ніж ті, що ми бачимо далі, але часто такі розв’язки не є допустимими (бо початковий стан ми генеруємо випадково) тому вони відкидуються.

З локальним покращувачем

Starting state
Metrics = 112 State([1 0 1 0 1 1 0 0 1 0 0 1])
Metrics = 63 State([0 0 1 0 1 0 1 1 0 0 0 0])
Metrics = 117 State([1 1 1 0 0 1 0 1 1 0 1 0])
Metrics = 105 State([0 1 1 1 1 0 0 1 0 1 0 1])
Metrics = 153 State([0 1 1 1 1 1 0 1 0 1 1 1])
Metrics = 106 State([0 0 1 0 0 0 1 1 1 0 1 1])
Metrics = 73 State([0 0 0 0 1 1 1 0 0 0 1 0])
Metrics = 141 State([1 1 1 0 0 1 0 1 0 1 1 1])
Metrics = 77 State([0 0 0 0 0 1 0 0 1 1 1 0])
Metrics = 85 State([0 0 0 0 0 1 0 0 1 0 1 1])
Metrics = 119 State([1 0 1 0 0 1 0 0 0 1 1 1])
Metrics = 132 State([1 1 1 1 0 0 1 1 0 1 1 1])
Metrics = 115 State([0 1 1 1 1 1 0 0 0 1 1 0])
Metrics = 110 State([0 1 0 1 1 1 0 1 0 1 0 1])
Metrics = 145 State([0 0 1 1 1 1 0 0 1 1 1 1])

Iteration #1

Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 1 0 1 0 0 0 1 1 0])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 1 0 1 0 0 0 1 1 0])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 1 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 99 State([0 1 1 1 0 1 0 1 0 0 1 0])

Iteration #2

Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([1 1 1 1 0 0 1 1 0 0 1 0])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])

Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 1 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 0 1 0 0 1 1 0 0 1 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 101 State([0 1 1 1 0 1 0 1 0 0 1 0])

Iteration #3

Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([1 1 1 1 0 0 1 1 0 0 1 0])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 1 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 0 1 0 0 1 1 0 0 1 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 101 State([0 1 1 1 0 1 0 1 0 0 1 0])

Iteration #4

Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 1 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 97 State([0 0 1 0 0 1 1 0 0 1 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])

Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 101 State([0 1 1 1 0 1 0 1 0 0 1 0])

Iteration #5

Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 94 State([0 0 1 0 0 1 0 0 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 1 0 0 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 0 0 1 1 1])
Metrics = 96 State([0 1 1 1 0 0 0 1 0 0 1 1])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 97 State([0 1 1 1 0 1 1 0 0 0 1 0])
Metrics = 101 State([0 1 1 1 0 1 0 1 0 0 1 0])

Без локального покращувача

Starting state
Metrics = 100 State([1 1 1 1 1 1 0 0 1 0 0 0])
Metrics = 77 State([1 1 0 0 1 1 0 0 0 0 1 0])
Metrics = 83 State([0 0 1 0 1 0 1 0 0 1 1 0])
Metrics = 104 State([1 1 1 0 1 0 0 1 0 1 1 0])
Metrics = 60 State([0 1 0 0 0 0 0 1 0 1 0 1])
Metrics = 124 State([1 0 1 1 1 0 1 1 0 0 1 1])
Metrics = 139 State([0 1 0 1 1 1 1 1 0 1 1 1])
Metrics = 63 State([0 1 1 1 1 0 0 1 0 0 0 0])
Metrics = 102 State([0 0 0 0 0 0 1 1 1 1 1 1])
Metrics = 133 State([1 0 0 1 0 1 1 1 1 0 1 1])
Metrics = 85 State([0 1 1 0 0 0 0 1 1 0 1 0])
Metrics = 116 State([1 0 1 0 0 1 0 1 1 0 1 0])
Metrics = 48 State([0 0 1 1 0 0 0 1 0 0 0 0])
Metrics = 79 State([1 0 1 1 1 1 0 0 0 0 0 0])
Metrics = 41 State([0 1 0 0 1 0 1 1 0 0 0 0])

Iteration #1

Metrics = 79 State([1 0 1 1 1 1 0 0 0 0 0 0])
Metrics = 85 State([0 1 1 0 0 0 0 1 1 0 1 0])
Metrics = 89 State([1 1 1 0 1 0 0 0 0 1 1 0])
Metrics = 91 State([0 0 1 1 1 0 1 0 0 1 1 0])
Metrics = 91 State([0 1 1 1 1 1 0 1 0 0 0 0])
Metrics = 100 State([1 1 1 1 1 1 0 0 1 0 0 0])
Metrics = 105 State([0 0 0 0 0 1 0 1 0 1 1 1])
Metrics = 108 State([0 0 0 1 1 1 1 1 1 0 1 0])
Metrics = 110 State([1 0 0 0 0 1 1 0 1 0 1 1])
Metrics = 112 State([1 0 0 0 0 0 1 1 1 1 1 1])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 116 State([1 0 1 0 0 1 0 1 1 0 1 0])
Metrics = 121 State([0 1 1 1 1 0 1 1 0 0 1 1])
Metrics = 126 State([0 0 1 0 0 1 1 1 0 0 1 1])
Metrics = 127 State([1 0 1 0 0 1 1 0 1 1 1 0])

Iteration #2

Metrics = 94 State([0 0 1 1 1 0 0 1 1 0 1 0])
Metrics = 97 State([0 0 1 1 0 0 1 0 1 1 1 0])
Metrics = 100 State([1 1 1 1 1 1 0 0 1 0 0 0])
Metrics = 100 State([1 0 0 1 0 1 0 1 0 1 1 0])
Metrics = 105 State([0 0 0 0 0 1 0 1 0 1 1 1])
Metrics = 106 State([0 1 1 1 1 0 1 0 0 0 1 1])
Metrics = 108 State([0 0 0 1 1 1 1 1 1 0 1 0])
Metrics = 110 State([1 0 0 0 0 1 1 0 1 0 1 1])
Metrics = 112 State([1 0 0 0 0 0 1 1 1 1 1 1])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 116 State([1 0 1 0 0 1 0 1 1 0 1 0])
Metrics = 126 State([0 0 1 0 0 1 1 1 0 0 1 1])
Metrics = 127 State([1 0 1 0 0 1 1 0 1 1 1 0])
Metrics = 128 State([1 0 1 0 0 1 1 1 0 1 1 0])

Iteration #3

Metrics = 105 State([0 0 0 0 0 1 0 1 0 1 1 1])
Metrics = 105 State([0 0 0 0 0 1 0 1 0 1 1 1])
Metrics = 106 State([0 1 1 1 1 0 1 0 0 0 1 1])
Metrics = 106 State([0 0 1 0 0 1 0 1 1 0 1 0])
Metrics = 108 State([0 0 0 1 1 1 1 1 1 0 1 0])
Metrics = 110 State([1 0 0 0 0 1 1 0 1 0 1 1])

Metrics = 112 State([1 0 0 0 0 0 1 1 1 1 1 1])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 115 State([0 0 0 1 0 1 1 1 1 1 1 0])
Metrics = 117 State([1 0 0 0 0 1 1 1 1 1 1 0])
Metrics = 123 State([0 0 0 1 0 1 1 1 1 0 1 1])
Metrics = 126 State([0 0 1 0 0 1 1 1 0 0 1 1])
Metrics = 128 State([1 0 1 0 0 1 1 1 0 1 1 0])
Metrics = 132 State([0 0 1 0 0 1 1 1 1 1 1 0])

Iteration #4

Metrics = 108 State([0 0 0 1 1 1 1 1 1 0 1 0])
Metrics = 110 State([1 0 0 0 0 1 1 0 1 0 1 1])
Metrics = 112 State([1 0 0 0 0 0 1 1 1 1 1 1])
Metrics = 113 State([0 1 1 0 0 1 0 1 1 0 1 0])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 115 State([0 0 0 1 0 1 1 1 1 1 1 0])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 117 State([1 0 0 0 0 1 1 1 1 1 1 0])
Metrics = 117 State([1 0 0 0 0 1 1 1 1 1 1 0])
Metrics = 123 State([0 0 0 1 0 1 1 1 1 0 1 1])
Metrics = 123 State([0 0 0 1 0 1 1 1 1 0 1 1])
Metrics = 124 State([1 0 1 0 0 1 0 0 1 0 1 1])
Metrics = 126 State([0 0 1 0 0 1 1 1 0 0 1 1])
Metrics = 132 State([0 0 1 0 0 1 1 1 1 1 1 0])

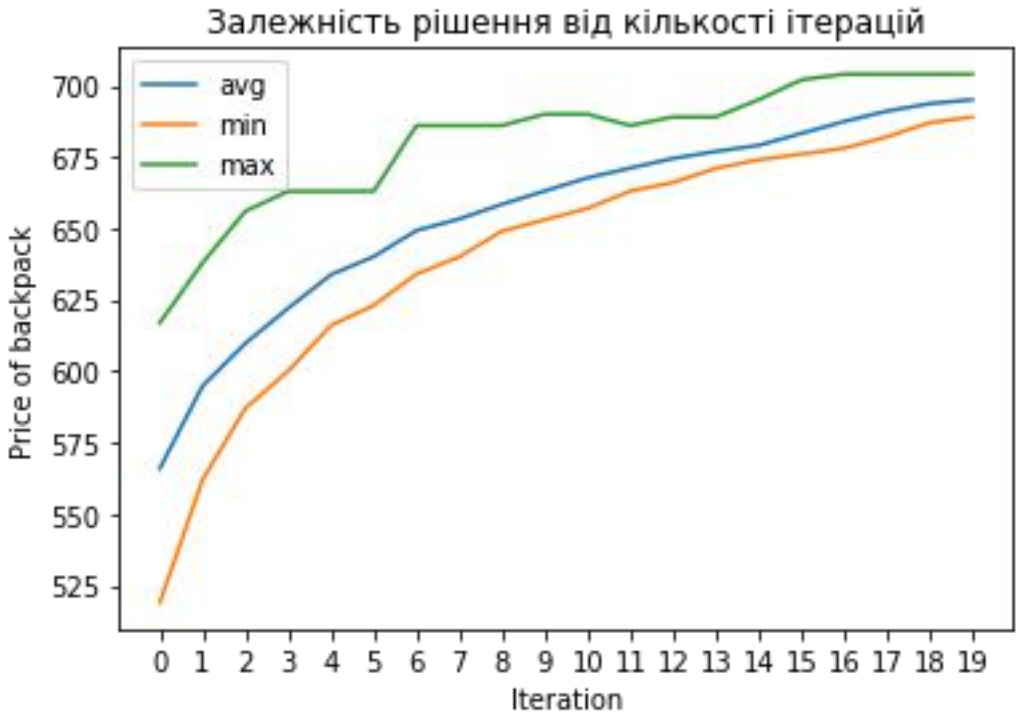
Iteration #5

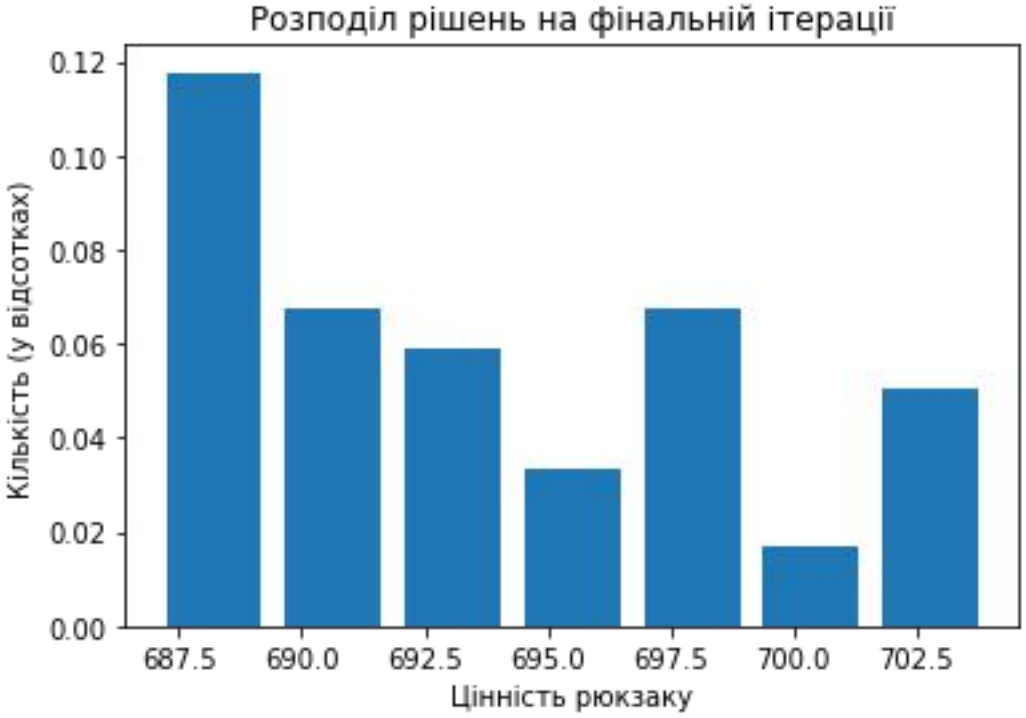
Metrics = 112 State([0 0 0 1 0 1 0 1 1 0 1 1])
Metrics = 113 State([0 1 1 0 0 1 0 1 1 0 1 0])
Metrics = 113 State([1 0 1 0 0 1 1 1 0 0 1 0])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 115 State([0 0 0 1 0 1 1 1 1 1 1 0])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 115 State([0 0 0 0 0 1 1 1 1 0 1 1])
Metrics = 117 State([1 0 0 0 0 1 1 1 1 1 1 0])
Metrics = 117 State([1 0 0 0 0 1 1 1 1 1 1 0])
Metrics = 123 State([0 0 0 1 0 1 1 1 1 0 1 1])
Metrics = 124 State([1 0 1 0 0 1 0 0 1 0 1 1])
Metrics = 126 State([0 0 1 0 0 1 1 1 0 0 1 1])
Metrics = 129 State([0 0 1 0 0 1 0 1 1 0 1 1])
Metrics = 129 State([0 0 1 0 0 1 0 1 1 0 1 1])

Metrics = 132 State([0 0 1 0 0 1 1 1 1 1 1 0])

Як бачимо якщо присутній локальний покращувач, то всі наші розв’зки стають більш-менш схожими. Вони сходяться до локального екстремума швидше, але без покращувача є більша ймовірність попасти в кращий екстремум.

Для кількості предметів = 75
Чисельність популяції = 49
Максимальна вага = 297





Штучний маленький приклад для демонстрації як сходиться популяція з 1 нащадка.