**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 4 з дисципліни

«Проектування алгоритмів»

„**Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.1**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Трикош Іван*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](#_Toc51260917)

[2 Завдання 4](#_Toc51260918)

[3 Виконання 10](#_Toc51260919)

[3.1 Програмна реалізація алгоритму 10](#_Toc51260920)

[3.1.1 Вихідний код 10](#_Toc51260921)

[3.1.2 Приклади роботи 10](#_Toc51260922)

[3.2 Тестування алгоритму 11](#_Toc51260923)

[3.2.1 Значення цільової функції зі збільшенням кількості ітерацій 11](#_Toc51260924)

[3.2.2 Графіки залежності розв'язку від числа ітерацій 11](#_Toc51260925)

[Висновок 12](#_Toc51260926)

[Критерії оцінювання 13](#_Toc51260927)

# Мета лабораторної роботи

Мета роботи – вивчити основні підходи формалізації метаевристичних алгоритмів і вирішення типових задач з їхньою допомогою.

# Завдання

Згідно варіанту, розробити алгоритм вирішення задачі і виконати його програмну реалізацію на будь-якій мові програмування.

Задача, алгоритм і його параметри наведені в таблиці 2.1.

Зафіксувати якість отриманого розв'язку (значення цільової функції) після кожних 20 ітерацій до 1000 і побудувати графік залежності якості розв'язку від числа ітерацій.

Зробити узагальнений висновок.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |
| --- | --- |
| **№** | **Задача і алгоритм** |
| 1 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 2 | Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 30, починають маршрут в різних випадкових вершинах). |
| 3 | Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники). |
| 4 | Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 5 | Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 35, починають маршрут в різних випадкових вершинах). |
| 6 | Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники). |
| 7 | Задача про рюкзак (місткість P=150, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 8 | Задача комівояжера (200 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм (α = 3, β = 2, ρ = 0,3, Lmin знайти жадібним алгоритмом, кількість мурах М = 45, починають маршрут в різних випадкових вершинах). |
| 9 | Задача розфарбовування графу (150 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 25 із них 3 розвідники). |
| 10 | Задача про рюкзак (місткість P=150, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 11 | Задача комівояжера (250 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм (α = 2, β = 4, ρ = 0,6, Lmin знайти жадібним алгоритмом, кількість мурах М = 45, починають маршрут в різних випадкових вершинах). |
| 12 | Задача розфарбовування графу (300 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 60 із них 5 розвідники). |
| 13 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий 30% і 70%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 14 | Задача комівояжера (250 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 4, β = 2, ρ = 0,3, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (10 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових вершинах). |
| 15 | Задача розфарбовування графу (100 вершин, степінь вершини не більше 20, але не менше 1), класичний бджолиний алгоритм (число бджіл 30 із них 3 розвідники). |
| 16 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий 30%, 40% і 30%, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 17 | Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 2, β = 4, ρ = 0,7, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (15 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових вершинах). |
| 18 | Задача розфарбовування графу (300 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 60 із них 5 розвідники). |
| 19 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 20 | Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 3, β = 2, ρ = 0,7, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (10 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах). |
| 21 | Задача розфарбовування графу (200 вершин, степінь вершини не більше 30, але не менше 1), класичний бджолиний алгоритм (число бджіл 40 із них 2 розвідники). |
| 22 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 23 | Задача комівояжера (300 вершин, відстань між вершинами випадкова від 1 до 60), мурашиний алгоритм (α = 3, β = 2, ρ = 0,6, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (15 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах). |
| 24 | Задача розфарбовування графу (400 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 70 із них 10 розвідники). |
| 25 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 26 | Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 30, починають маршрут в різних випадкових вершинах). |
| 27 | Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники). |
| 28 | Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 29 | Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 35, починають маршрут в різних випадкових вершинах). |
| 30 | Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники). |
| 31 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 32 | Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 30, починають маршрут в різних випадкових вершинах). |
| 33 | Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники). |
| 34 | Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 35 | Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 35, починають маршрут в різних випадкових вершинах). |

# Виконання

## Програмна реалізація алгоритму

### Вихідний код

import random

class Stuff:

"""Предмет"""

def \_\_init\_\_(self):

"""Конструктор"""

self.price = random.randint(2, 20)

self.weight = random.randint(1, 10)

return

def get\_stuff(self):

"""Отримання предмету як рядка"""

return "price = " + str(self.price) + ", weight = " + str(self.weight) + "; "

class Backpack:

"""Рюкзак"""

def \_\_init\_\_(self):

"""Конструктор"""

self.capacity = 250

self.number = 100

self.result = []

return

class Algorithm:

"""Генетичний алгоритм"""

def \_\_init\_\_(self):

"""Конструктор"""

self.backpack = Backpack()

self.stuffs = [Stuff() for i in range(self.backpack.number)]

self.population = []

self.set\_start\_population()

self.backpack.result = self.best\_chromosome()

self.numb = 0

return

def set\_start\_population(self):

"""Встановлення початкової популяції"""

for i in range(self.backpack.number):

self.population += [[0 for j in range(self.backpack.number)]]

self.population[i][i] = 1

return

def chromosome\_price(self, chromosome):

"""Ціна хромосоми (особи)"""

price = 0

for i in range(len(chromosome)):

price += chromosome[i] \* self.stuffs[i].price

return price

def chromosome\_weight(self, chromosome):

"""Вага хромосоми (особи)"""

weight = 0

for i in range(len(chromosome)):

weight += chromosome[i] \* self.stuffs[i].weight

return weight

def best\_chromosome(self):

"""Найкраща хромосома (особа) в популяції"""

best = self.population[0]

index = 0

for i in range(1, len(self.population)):

if self.chromosome\_price(best) < self.chromosome\_price(self.population[i]):

best = self.population[i]

index = i

return index

def worst\_chromosome(self):

"""Найгірша хромосома (особа) в популяції"""

worst = self.population[0]

index = 0

for i in range(1, len(self.population)):

if self.chromosome\_price(worst) > self.chromosome\_price(self.population[i]):

worst = self.population[i]

index = i

return index

def choose\_parents(self):

"""Вибір батьків"""

index = self.best\_chromosome()

temp = self.population.copy()

parent1 = temp.pop(index)

parent2 = random.choice(temp)

return parent1, parent2

def crossing(self, parent1, parent2):

"""Схрещування хромосом"""

result = parent1[:len(parent1) // 2] + parent2[len(parent2) // 2:]

if self.chromosome\_weight(result) > self.backpack.capacity:

result = None

return result

def mutation(self, chromosome):

"""Мутація хромосоми"""

result = chromosome.copy()

numb = random.randint(1, 20)

if numb == 1:

index = random.choice(range(len(result)))

result[index] = 0 if result[index] else 1

self.numb += 1

if self.chromosome\_weight(result) > self.backpack.capacity:

result = chromosome.copy()

return result

def get\_min\_stuff(self, chromosome):

"""Отримання предмету з мінімальною вагою"""

index = 0

while chromosome[index]:

index += 1

min\_stuff = self.stuffs[index]

for i in range(index + 1, len(self.stuffs)):

if not chromosome[i]:

if min\_stuff.weight > self.stuffs[i].weight:

min\_stuff = self.stuffs[i]

index = i

return index

def improvement(self, chromosome):

"""Локальне покращення"""

index = self.get\_min\_stuff(chromosome)

result = chromosome.copy()

result[index] = 1

if self.chromosome\_weight(result) > self.backpack.capacity:

result = chromosome.copy()

return result

def start(self):

"""Запуск алгоритму"""

for i in range(1000):

parent1, parent2 = self.choose\_parents()

temp = self.crossing(parent1, parent2)

if temp:

temp = self.mutation(temp)

temp = self.improvement(temp)

self.population += [temp]

self.population.pop(self.worst\_chromosome())

self.backpack.result = self.population[self.best\_chromosome()]

if i % 20 == 19:

print(f"Iteration:{i + 1}, value:{self.chromosome\_price(self.backpack.result)}")

# print("\n\nBest", i)

# print("The weight of stuffs = ", self.chromosome\_weight(self.backpack.result), "\nThe price of stuffs = ", self.chromosome\_price(self.backpack.result))

# numb = 0

# for j in temp:

# if j:

# numb += 1

# print("The number of stuffs:", numb)

# print("\n\nResult:", i)

# for j in range(len(temp)):

# if temp[j]:

# print(self.stuffs[j].get\_stuff(), end="")

# print("\nThe weight of stuffs = ", self.chromosome\_weight(temp), "\nThe price of stuffs = ", self.chromosome\_price(temp))

# numb = 0

# for j in temp:

# if j:

# numb += 1

# print("The number of stuffs:", numb)

return self.backpack.result

def main():

algorithm = Algorithm()

result = algorithm.start()

print("\n\nStuffs:")

for i in range(len(algorithm.stuffs)):

print(algorithm.stuffs[i].get\_stuff(), end="")

if i % 5 == 4:

print()

print("\nResult:")

numb = 0

for i in range(len(result)):

if result[i]:

print(algorithm.stuffs[i].get\_stuff(), end="")

if numb % 5 == 4:

print()

numb += 1

print("\nThe weight of stuffs = ", algorithm.chromosome\_weight(result), "\nThe price of stuffs = ", algorithm.chromosome\_price(result))

numb = 0

for j in result:

if j:

numb += 1

print("The number of stuffs:", numb)

return

if \_\_name\_\_ == "\_\_main\_\_":

main()

### Приклади роботи

На рисунках 3.1-3.4 показані приклади роботи програми.

Приклад 1

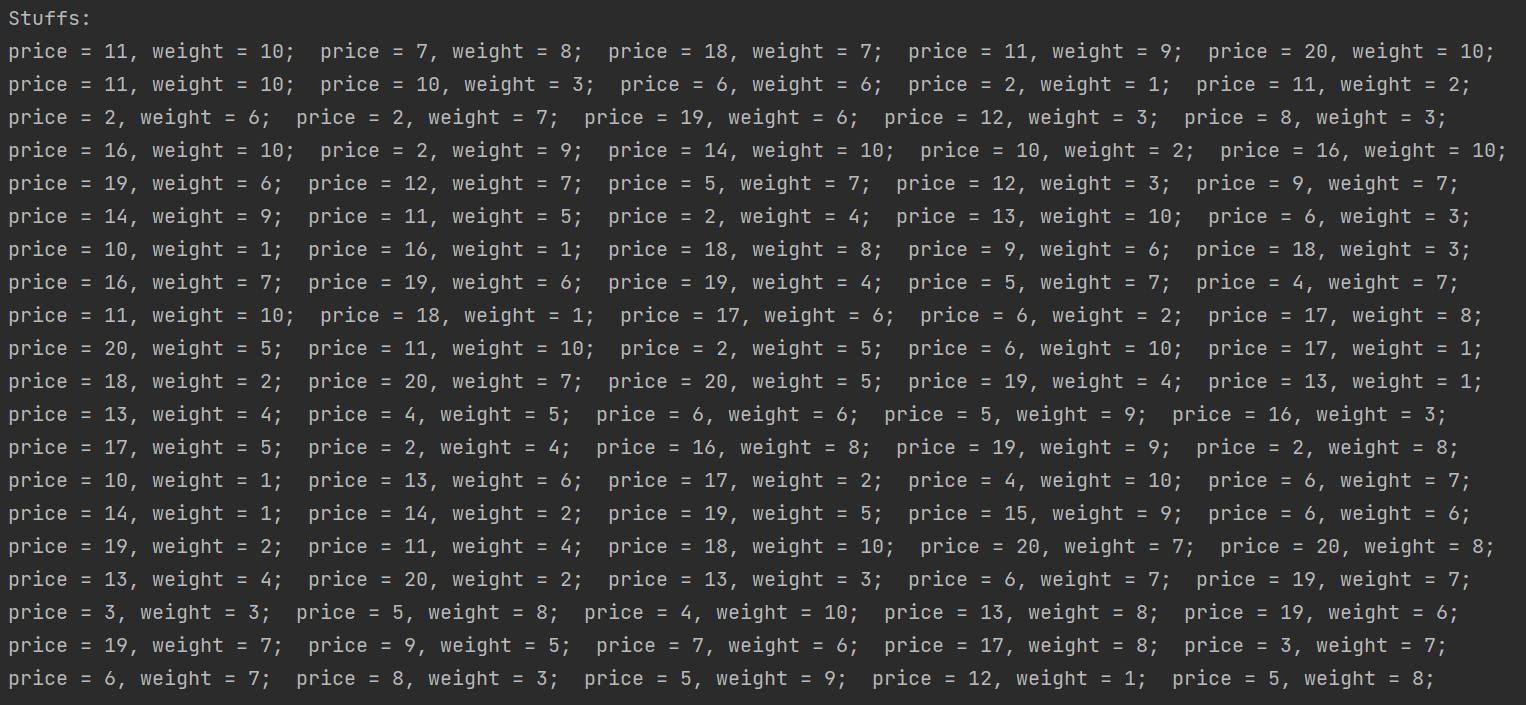


Рисунок 3.1 – Згенеровані предмети

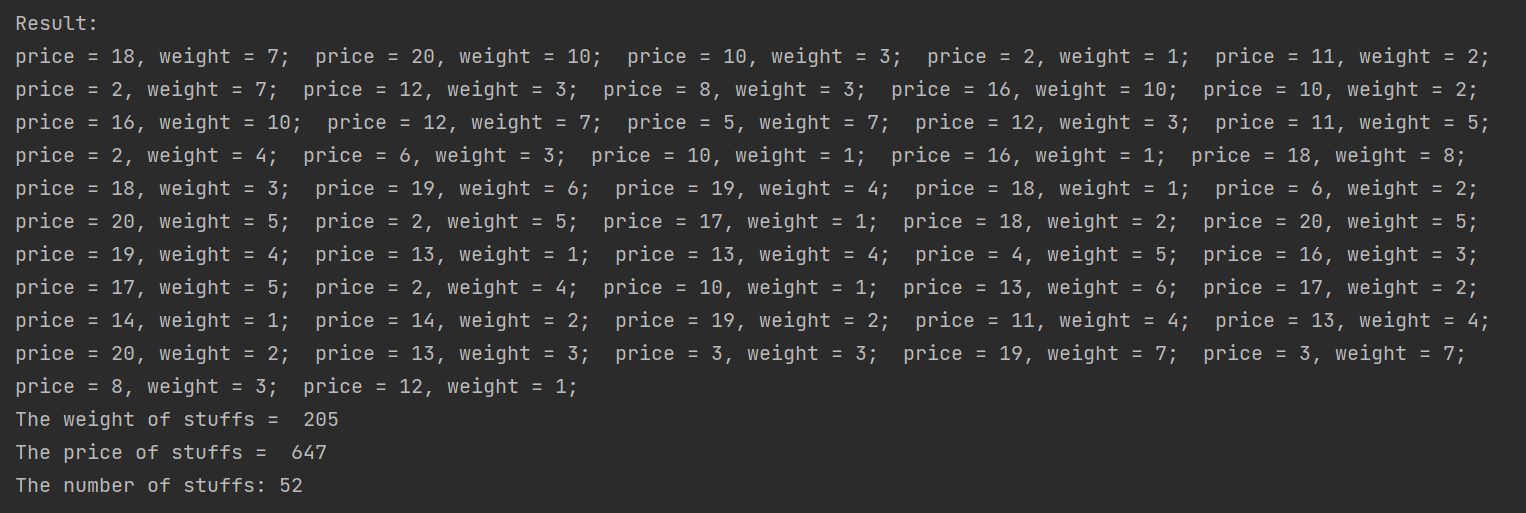


Рисунок 3.2 – Вміст рюкзака

Приклад 2

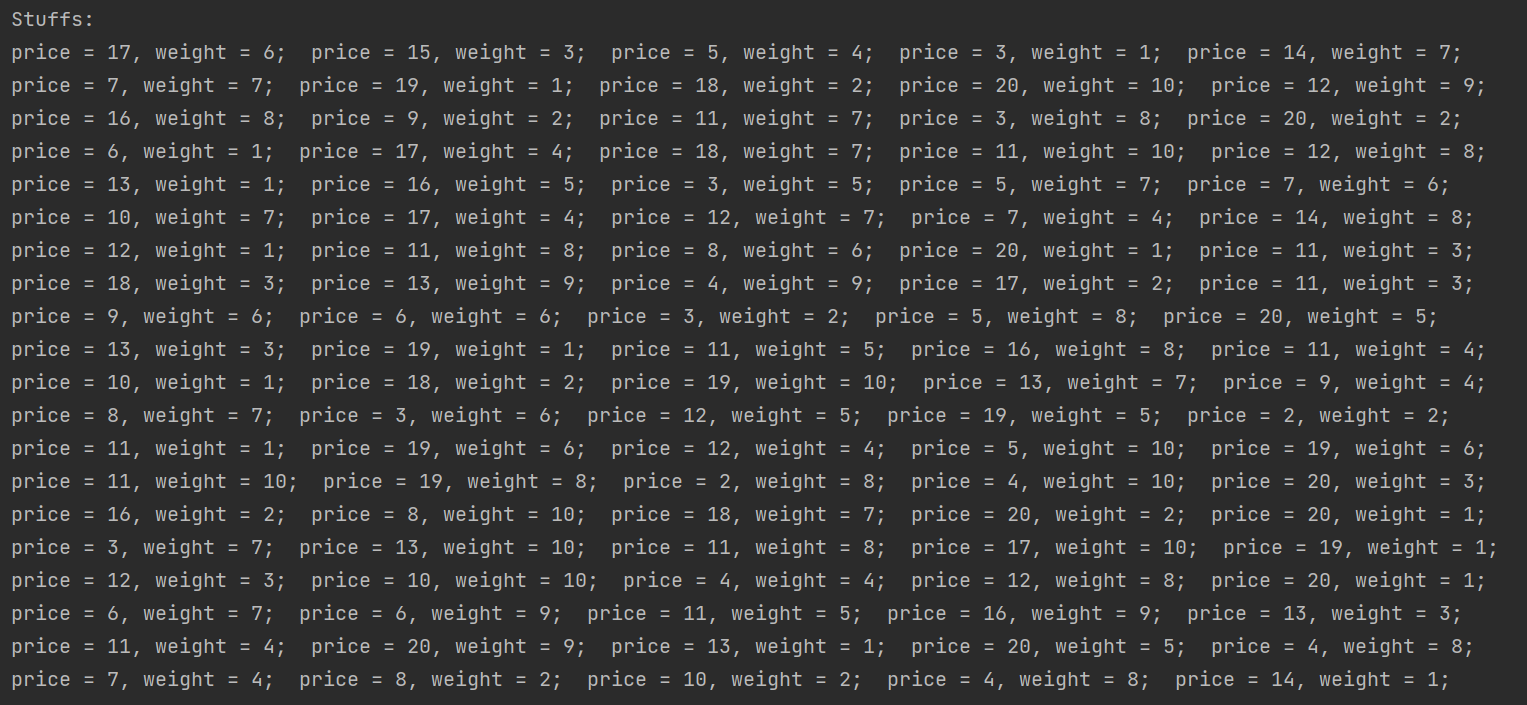


Рисунок 3.3 – Згенеровані предмети

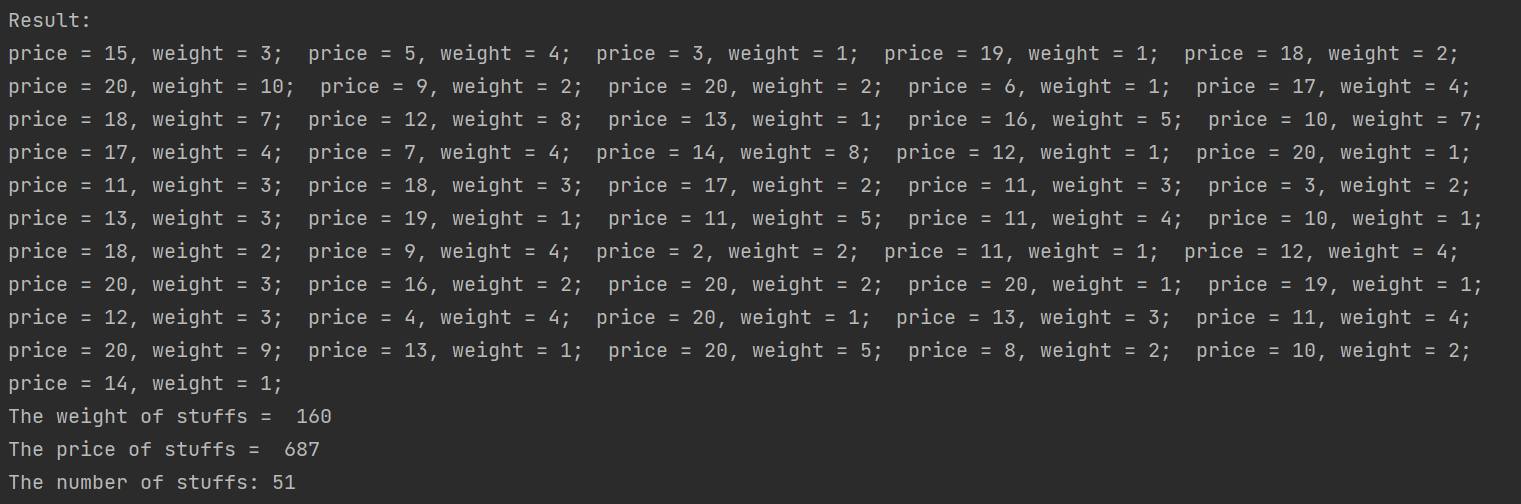


Рисунок 3.4 – Вміст рюкзака

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

У таблиці 3.1 наведено значення цільової функції зі збільшенням кількості ітерацій.

Таблиця 3.1 – Залежність значення цільової функції від кількості ітерацій

|  |  |
| --- | --- |
| Кількість ітерацій | Значення функції |
| 20 | 141 |
| 40 | 174 |
| 60 | 186 |
| 80 | 205 |
| 100 | 206 |
| 120 | 224 |
| 140 | 224 |
| 160 | 225 |
| 180 | 225 |
| 200 | 243 |
| 220 | 275 |
| 240 | 275 |
| 260 | 275 |
| 280 | 289 |
| 300 | 289 |
| 320 | 289 |
| 340 | 315 |
| 360 | 336 |
| 380 | 360 |
| 400 | 374 |
| 420 | 407 |
| 440 | 407 |
| 460 | 415 |
| 480 | 427 |
| 500 | 427 |
| 520 | 436 |
| 540 | 438 |
| 560 | 446 |
| 580 | 465 |
| 600 | 477 |
| 620 | 477 |
| 640 | 508 |
| 660 | 520 |
| 680 | 521 |
| 700 | 521 |
| 720 | 521 |
| 740 | 533 |
| 760 | 539 |
| 780 | 539 |
| 800 | 579 |
| 820 | 611 |
| 840 | 615 |
| 860 | 623 |
| 880 | 644 |
| 900 | 644 |
| 920 | 644 |
| 940 | 644 |
| 960 | 644 |
| 980 | 672 |
| 1000 | 687 |

### Графіки залежності розв'язку від числа ітерацій

На рисунку 3.5 наведений графік, який показує якість отриманого розв'язку.

Рисунок 3.5 – Графіки залежності розв'язку від числа ітерацій

Висновок

В рамках даної лабораторної роботи я вивчив основні підходи формалізації метаевристичних алгоритмів і вирішення типових задач з їхньою допомогою. Я вирішував задачу про рюкзак (місткість – 250, є 100 предметів, цінність предметів від 2 до 20 (випадкова), вага предметів від 1 до 10 (випадкова)) за допомогою генетичного алгоритму, що має такі особливості: початкова популяція складається зі 100 осіб по 1 різному предмету, вибір батьків: найкраща особа і випадкова особа, схрещування одноточкове по 50 генів, ймовірність мутації – 5%, при мутації змінюється 1 випадковий ген. Я розробив такий оператор покращення: після мутації він знаходить предмет з найменшою вагою, якого нема в рюкзаку, і якщо його можна додати в рюкзак, то цей предмет додається.

Я протестував алгоритм і побудував графік залежності якості розв’язку від кількості ітерацій (до 1000 ітерацій).

Критерії оцінювання

При здачі лабораторної роботи до 27.11.2021 включно максимальний бал дорівнює – 5. Після 27.11.2021 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* програмна реалізація алгоритму – 75%;
* тестування алгоритму– 20%;
* висновок – 5%.