**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-12 Дулов Денис Валерійович*

**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 метаеврестичних 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

using System;

namespace lab4

{

class Program

{

public const int CAPACITY = 150;

public const int AMOUNT = 100;

public const int NUMBER\_OF\_ITERATIONS = 1000;

public static Item[] itemList = createItems(AMOUNT);

static void Main(string[] args)

{

var backpacks = initBackpacks(AMOUNT);

printItems(itemList);

backpacks = geneticAlgorithm(backpacks);

Console.WriteLine("\n\n\nBEST ITEM:");

backpacks[getBestIndex(backpacks)].print();

}

static Backpack[] geneticAlgorithm(Backpack[] backpacks)

{

for(int i = 0;i < NUMBER\_OF\_ITERATIONS; ++i)

{

backpacks = geneticIteration(backpacks);

}

return backpacks;

}

static Backpack[] geneticIteration(Backpack[] backpacks)

{

(var bestBP, var randomBP) = chooseAncestors(backpacks);

var child = evenSex(bestBP, randomBP);

var mutated = mutate(child);

if (getTotalWeight(child) > CAPACITY && getTotalWeight(mutated) > CAPACITY) return backpacks;

var improved = localImprovement(child, mutated);

var best = chooseBest(child, mutated, improved);

int worstIndex = getWorstIndex(backpacks);

backpacks[worstIndex] = best;

return backpacks;

}

static int getWorstIndex(Backpack[] backpacks)

{

int worstCost = int.MaxValue;

int worstIndex = -1;

for(int i = 0;i < AMOUNT; ++i)

{

if(getTotalCost(backpacks[i]) < worstCost)

{

worstCost = getTotalCost(backpacks[i]);

worstIndex = i;

}

}

return worstIndex;

}

static Backpack chooseBest(Backpack child, Backpack mutated, Backpack improved)

{

if(getTotalWeight(child) > CAPACITY)

{

if(getTotalWeight(improved) > CAPACITY)

{

return mutated;

}

if (getTotalCost(mutated) > getTotalCost(improved)) return mutated;

else return improved;

}else if(getTotalWeight(mutated) > CAPACITY)

{

if(getTotalWeight(improved) > CAPACITY)

{

return child;

}

if (getTotalCost(child) > getTotalCost(improved)) return child;

else return improved;

}else if (getTotalWeight(improved) > CAPACITY)

{

if (getTotalCost(mutated) > getTotalCost(child)) return mutated;

else return child;

}

else

{

int maxCost = Math.Max(getTotalCost(child), Math.Max(getTotalCost(mutated), getTotalCost(improved)));

if (maxCost == getTotalCost(improved)) return improved;

else if (maxCost == getTotalCost(mutated)) return mutated;

else return child;

}

}

static Backpack localImprovement(Backpack child, Backpack mutated)

{

if (getTotalWeight(mutated) > CAPACITY)

{

if (getTotalWeight(child) <= CAPACITY)

{

return improve(child);

}

}

else

{

if (getTotalWeight(child) <= CAPACITY)

{

if (getTotalCost(child) >= getTotalCost(mutated))

{

return improve(child);

}

else

{

return improve(mutated);

}

}

else

{

return improve(mutated);

}

}

return null;

}

static Backpack improve(Backpack bp)

{

var result = new Backpack(bp);

if (result.items[getBestItemIndex()] == false)

{

result.items[getBestItemIndex()] = true;

}

return result;

}

static int getBestItemIndex()

{

double bestBenefit = 0;

int bestIndex = -1;

for(int i = 0;i < AMOUNT; ++i)

{

if(itemList[i].Cost / itemList[i].Weight > bestBenefit)

{

bestBenefit = itemList[i].Cost / itemList[i].Weight;

bestIndex = i;

}

}

return bestIndex;

}

static Backpack mutate(Backpack bp)

{

var rand = new Random();

if(rand.NextDouble() > 0.1)

{

int randIndex1 = rand.Next(0, bp.items.Length);

int randIndex2 = rand.Next(0, bp.items.Length);

while(randIndex1 == randIndex2)

{

randIndex2 = rand.Next(0, bp.items.Length);

}

bool tmp = bp.items[randIndex1];

bp.items[randIndex1] = bp.items[randIndex2];

bp.items[randIndex2] = tmp;

}

return bp;

}

static Backpack evenSex(Backpack bp1, Backpack bp2)

{

var rand = new Random();

var bpRes = new Backpack(AMOUNT);

for(int i = 0;i < bp1.items.Length; ++i)

{

if(rand.NextDouble() < 0.5)

{

bpRes.items[i] = bp1.items[i];

}

else

{

bpRes.items[i] = bp2.items[i];

}

}

return bpRes;

}

static (Backpack, Backpack) chooseAncestors(Backpack[] backpacks)

{

int bestIndex = getBestIndex(backpacks);

var bestBP = backpacks[bestIndex]; // getting best backpack

var rand = new Random();

int randomIndex = rand.Next(0, backpacks.Length);

while (randomIndex == bestIndex)

{

randomIndex = rand.Next(0, backpacks.Length); //getting random backpack except best one

}

var randomBP = backpacks[randomIndex];

return (bestBP, randomBP);

}

static int getBestIndex(Backpack[] bps)

{

int bestIndex = -1;

int bestCost = 0;

for(int i = 0;i < bps.Length; ++i)

{

if(getTotalCost(bps[i]) > bestCost)

{

bestCost = getTotalCost(bps[i]);

bestIndex = i;

}

}

return bestIndex;

}

static Item[] createItems(int amount)

{

Item[] items = new Item[amount];

for(int i = 0;i < items.Length; ++i)

{

items[i] = new Item();

}

return items;

}

static Backpack[] initBackpacks(int amount)

{

Backpack[] backpacks = new Backpack[amount];

for(int i = 0;i < backpacks.Length; ++i)

{

backpacks[i] = new Backpack(AMOUNT);

backpacks[i].items[i] = true;

}

return backpacks;

}

static void printItems(Item[] items)

{

Console.Write("Weights: ");

foreach(var item in items)

{

Console.Write(item.Weight + " ");

}

Console.Write("\nCosts: ");

foreach (var item in items)

{

Console.Write(item.Cost + " ");

}

Console.WriteLine();

}

static void printBackpacks(Backpack[] backpacks)

{

foreach(var item in backpacks)

{

item.print();

}

Console.WriteLine();

}

public static int getTotalCost(Backpack bp)

{

int totalCost = 0;

for(int i = 0;i < bp.items.Length; ++i)

{

if (bp.items[i]) totalCost += itemList[i].Cost;

}

return totalCost;

}

public static int getTotalWeight(Backpack bp)

{

int totalWeight = 0;

for (int i = 0; i < bp.items.Length; ++i)

{

if (bp.items[i]) totalWeight += itemList[i].Weight;

}

return totalWeight;

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace lab4

{

class Backpack

{

public bool[] items;

public Backpack(int amount)

{

items = new bool[amount];

}

public Backpack(Backpack backpack)

{

this.items = backpack.items;

}

public void print()

{

Console.Write("[");

for(int i = 0;i < this.items.Length;++i)

{

if(this.items[i]) Console.Write("1");

else Console.Write("0");

if (i != this.items.Length - 1) Console.Write(",");

}

Console.WriteLine("] Cost: " + Program.getTotalCost(this) + " Weight: " + Program.getTotalWeight(this));

}

public static bool operator ==(Backpack bp1, Backpack bp2)

{

for(int i = 0;i < bp1.items.Length; ++i)

{

if (bp1.items[i] != bp2.items[i]) return false;

}

return true;

}

public static bool operator !=(Backpack bp1, Backpack bp2)

{

return !(bp1 == bp2);

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace lab4

{

class Item

{

public int Weight { get; set; }

public int Cost { get; set; }

public Item()

{

var rand = new Random();

Weight = rand.Next(2, 11);

Cost = rand.Next(1, 6);

}

}

}

### Приклади роботи

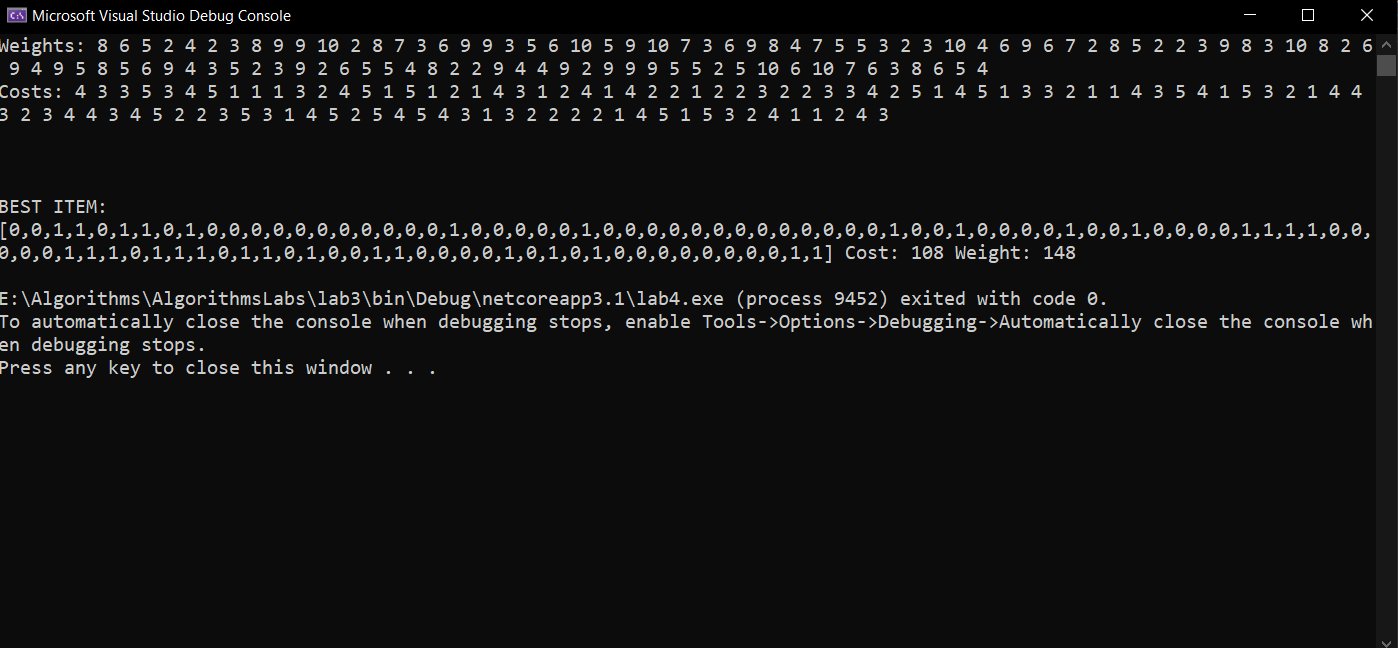
На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1

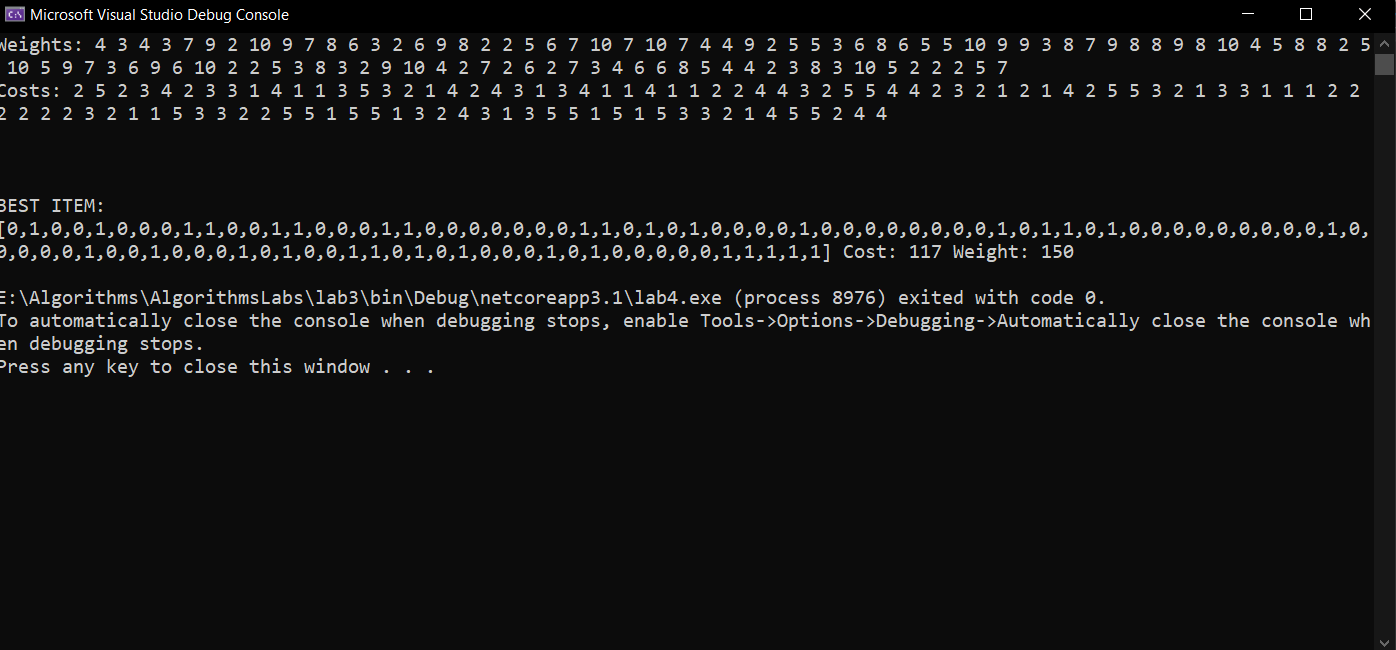


Рисунок 3.2

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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

Таблиця 3.1

|  |  |
| --- | --- |
| Кількість ітерацій | Значення цільової функції |
| 0 | 5 |
| 20 | 21 |
| 40 | 21 |
| 60 | 23 |
| 80 | 28 |
| 100 | 35 |
| 120 | 35 |
| 140 | 35 |
| 160 | 35 |
| 180 | 35 |
| 200 | 39 |
| 220 | 41 |
| 240 | 42 |
| 260 | 48 |
| 280 | 49 |
| 300 | 52 |
| 320 | 59 |
| 340 | 59 |
| 360 | 59 |
| 380 | 60 |
| 400 | 65 |
| 420 | 67 |
| 440 | 70 |
| 460 | 71 |
| 480 | 76 |
| 500 | 81 |
| 520 | 83 |
| 540 | 83 |
| 560 | 85 |
| 580 | 86 |
| 600 | 86 |
| 620 | 97 |
| 640 | 99 |
| 660 | 103 |
| 680 | 108 |
| 700 | 108 |
| 720 | 108 |
| 740 | 108 |
| 760 | 108 |
| 780 | 112 |
| 800 | 113 |
| 820 | 115 |
| 840 | 115 |
| 860 | 115 |
| 880 | 116 |
| 900 | 119 |
| 920 | 119 |
| 940 | 122 |
| 960 | 122 |
| 980 | 122 |
| 1000 | 122 |

### Графіки залежності розв'язку від числа ітерацій

На рисунку 3.3 наведений графік, який показує якість отриманого розв'язку.

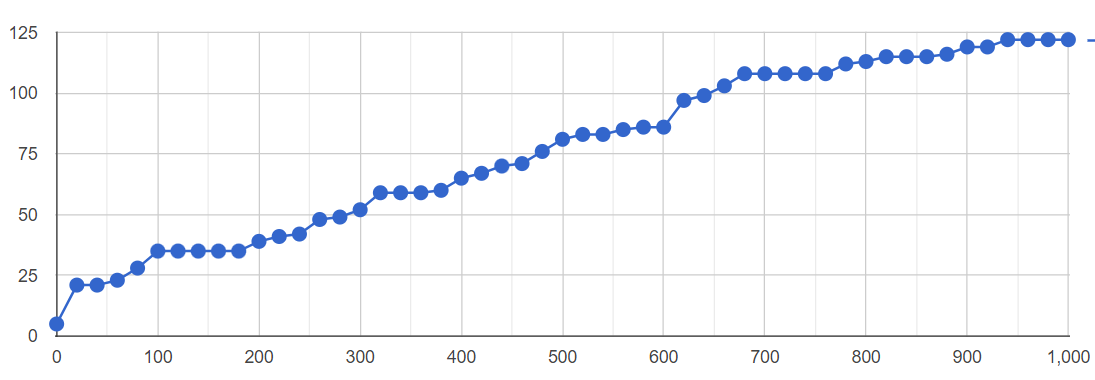


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

Висновок

В рамках даної лабораторної роботи було досліджено метаевристичні алгоритми, а також набуто навичок їх практичної реалізації на прикладі задачі про рюкзак та генетичного алгоритму.

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%.