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) *ІП-15 Мочалов Дмитро Юрійович* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Головченко М.Н.* _____
(прізвище, ім'я, по батькові)

Київ 2022

Зміст

1 Мета лабораторної роботи

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

2 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 ($\alpha = 2$, $\beta = 4$, $\rho = 0{,}4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 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 ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 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 ($\alpha = 3$, $\beta = 2$, $\rho = 0,3$, Lmin знайти жадібним алгоритмом, кількість мурах M = 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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,6$, Lmin знайти жадібним алгоритмом, кількість мурах M = 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 ($\alpha = 4$, $\beta = 2$, $\rho = 0,3$, Lmin знайти жадібним алгоритмом, кількість мурах M = 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 M = 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 M = 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 M = 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 M = 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 M = 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 M = 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 M = 35, починають маршрут в різних випадкових вершинах).

3 Виконання

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
4   internal static class Solver
5       {
6           public static BackPack backPack = new BackPack();
7           static Population population = new Population(backPack);
8           static Chromosome Record;
9
10          public static void Solve()
11          {
12              population.GeneratePopulation();
13              Record = population.GetTheWorstChromosome;
14              int iterations = 0;
15              while (iterations <= 1000)
16              {
17                  Chromosome FirstParent = population.GetBestChromosome;
18                  Chromosome SecondParent;
19                  while (true)
20                  {
21                      Random random  = new Random();
22                      int index = random.Next(0, 100);
23                      if (population.Chromosomes[index] != FirstParent)
24                      {
25                          SecondParent = population.Chromosomes[index];
26                          break;
27                      }
28                  }
29                  Chromosome Offspring = CrossOver(FirstParent, SecondParent);
30                  Chromosome mutation = Mutation(Offspring);
31                  Chromosome improved = LocalImprovement(mutation);
32                  if(improved.GetValue > Record.GetValue)
33                  {
34                      Record= improved;
35                  }
36                  Chromosome worstChromo = population.GetTheWorstChromosome;
37                  if (improved.GetWeight <= backPack.Capacity)
38                  {
39
40                      population.Chromosomes.Remove(worstChromo);
41                      population.Chromosomes.Add(improved);
42                  }
43                  else
44                  {
45                      if(Offspring.GetWeight <= backPack.Capacity)
46                      {
47                          population.Chromosomes.Remove(worstChromo);
48                          population.Chromosomes.Add(Offspring);
49                      }
50                      else
51                      {
52                          continue;
53                      }
54                  }
55                  if(iterations%20==0)
56                  {
57                      WriteRecordToFile(iterations, Record.GetValue);
```

```
58                  }
59                  iterations++;
60              }
61              for (int i = 0; i < population.Chromosomes.Count; i++)
62              {
63                  Console.WriteLine("Chomosome: " + i + "  Weight: " +
    population.Chromosomes[i].GetWeight + "  Value: " +
    population.Chromosomes[i].GetValue);
64              }
65          }
66
67          private static Chromosome CrossOver(Chromosome FirstParent,Chromosome
    SecondParent)
68          {
69              List<int> mergedList = new List<int>(100);
70              int flag = 1;
71              for (int i = 0; i < 100; i++)
72              {
73                  if (i % 25 == 0 && i!= 0) flag *= -1;
74                  if(flag > 0)
75                  {
76                      mergedList.Add(FirstParent.Gene[i]);
77                  }
78                  else
79                  {
80                      mergedList.Add(SecondParent.Gene[i]);
81                  }
82
83              }
84              Chromosome Offspring = new Chromosome(mergedList);
85              return Offspring;
86          }
87
88          private static Chromosome Mutation(Chromosome chromosome)
89          {
90              Random random = new Random();
91              int chance = random.Next(1, 101);
92              if(chance <= 5)
93              {
94                  int index_1 = random.Next(0, chromosome.Gene.Count);
95                  int index_2;
96                  while (true)
97                  {
98                      index_2 = random.Next(0, chromosome.Gene.Count);
99                      if (index_2 != index_1) break;
100                 }
101                 (chromosome.Gene[index_1], chromosome.Gene[index_2]) =
    (chromosome.Gene[index_2], chromosome.Gene[index_1]);
102             }
103             return chromosome;
104         }
105
106         private static Chromosome LocalImprovement(Chromosome chromosome)
107         {
108             Random random = new Random();
109             while (true)
110             {
111                 int index = random.Next(0,chromosome.Gene.Count);
112                 if (chromosome.Gene[index] == 0)
113                 {
114                     chromosome.Gene[index] = 1;
115                     break;
116                 }
```

```
117             }
118             return chromosome;
119         }
120         private static void WriteRecordToFile(int iteration,int value)
121         {
122             using(var stream = new StreamWriter("records.txt", true))
123             {
124                 stream.WriteLine(value);
125             }
126         }
```

126.1.1 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 –

```
Chomosome: 0  Weight: 245  Value: 602
Chomosome: 1  Weight: 248  Value: 603
Chomosome: 2  Weight: 246  Value: 603
Chomosome: 3  Weight: 246  Value: 604
Chomosome: 4  Weight: 246  Value: 601
Chomosome: 5  Weight: 246  Value: 620
Chomosome: 6  Weight: 244  Value: 605
Chomosome: 7  Weight: 246  Value: 620
Chomosome: 8  Weight: 247  Value: 605
Chomosome: 9  Weight: 246  Value: 601
Chomosome: 10  Weight: 248  Value: 601
Chomosome: 11  Weight: 250  Value: 604
Chomosome: 12  Weight: 250  Value: 606
Chomosome: 13  Weight: 250  Value: 603
Chomosome: 14  Weight: 250  Value: 604
Chomosome: 15  Weight: 246  Value: 601
Chomosome: 16  Weight: 248  Value: 623
Chomosome: 17  Weight: 247  Value: 608
Chomosome: 18  Weight: 249  Value: 602
Chomosome: 19  Weight: 250  Value: 604
Chomosome: 20  Weight: 250  Value: 614
Chomosome: 21  Weight: 249  Value: 623
Chomosome: 22  Weight: 249  Value: 604
Chomosome: 23  Weight: 250  Value: 610
Chomosome: 24  Weight: 250  Value: 610
Chomosome: 25  Weight: 249  Value: 604
Chomosome: 26  Weight: 249  Value: 605
Chomosome: 27  Weight: 246  Value: 613
Chomosome: 28  Weight: 246  Value: 613
Chomosome: 29  Weight: 249  Value: 623
Chomosome: 30  Weight: 246  Value: 613
Chomosome: 31  Weight: 250  Value: 611
Chomosome: 32  Weight: 247  Value: 608
Chomosome: 33  Weight: 250  Value: 611
Chomosome: 34  Weight: 249  Value: 623
Chomosome: 35  Weight: 249  Value: 605
Chomosome: 36  Weight: 249  Value: 623
Chomosome: 37  Weight: 250  Value: 611
Chomosome: 38  Weight: 250  Value: 608
Chomosome: 39  Weight: 250  Value: 611
Chomosome: 40  Weight: 250  Value: 611
Chomosome: 41  Weight: 249  Value: 605
Chomosome: 42  Weight: 247  Value: 608
Chomosome: 43  Weight: 250  Value: 608
Chomosome: 44  Weight: 250  Value: 611
Chomosome: 45  Weight: 249  Value: 604
Chomosome: 46  Weight: 250  Value: 611
Chomosome: 47  Weight: 250  Value: 615
Chomosome: 48  Weight: 250  Value: 611
Chomosome: 49  Weight: 249  Value: 604
Chomosome: 50  Weight: 249  Value: 607
```

```
Chomosome: 51  Weight: 250  Value: 608
Chomosome: 52  Weight: 249  Value: 605
Chomosome: 53  Weight: 249  Value: 607
Chomosome: 54  Weight: 250  Value: 608
Chomosome: 55  Weight: 246  Value: 613
Chomosome: 56  Weight: 249  Value: 604
Chomosome: 57  Weight: 246  Value: 613
Chomosome: 58  Weight: 250  Value: 611
Chomosome: 59  Weight: 247  Value: 624
Chomosome: 60  Weight: 242  Value: 622
Chomosome: 61  Weight: 245  Value: 604
Chomosome: 62  Weight: 249  Value: 623
Chomosome: 63  Weight: 239  Value: 612
Chomosome: 64  Weight: 244  Value: 602
Chomosome: 65  Weight: 250  Value: 606
Chomosome: 66  Weight: 250  Value: 606
Chomosome: 67  Weight: 248  Value: 634
Chomosome: 68  Weight: 244  Value: 625
Chomosome: 69  Weight: 247  Value: 627
Chomosome: 70  Weight: 243  Value: 607
Chomosome: 71  Weight: 250  Value: 609
Chomosome: 72  Weight: 246  Value: 616
Chomosome: 73  Weight: 245  Value: 620
Chomosome: 74  Weight: 247  Value: 624
Chomosome: 75  Weight: 250  Value: 637
Chomosome: 76  Weight: 242  Value: 622
Chomosome: 77  Weight: 244  Value: 602
Chomosome: 78  Weight: 243  Value: 614
Chomosome: 79  Weight: 243  Value: 610
Chomosome: 80  Weight: 244  Value: 602
Chomosome: 81  Weight: 250  Value: 611
Chomosome: 82  Weight: 248  Value: 612
Chomosome: 83  Weight: 249  Value: 631
Chomosome: 84  Weight: 246  Value: 602
Chomosome: 85  Weight: 249  Value: 610
Chomosome: 86  Weight: 247  Value: 611
Chomosome: 87  Weight: 246  Value: 627
Chomosome: 88  Weight: 239  Value: 612
Chomosome: 89  Weight: 247  Value: 608
Chomosome: 90  Weight: 247  Value: 612
Chomosome: 91  Weight: 247  Value: 620
Chomosome: 92  Weight: 246  Value: 602
Chomosome: 93  Weight: 247  Value: 611
Chomosome: 94  Weight: 247  Value: 606
Chomosome: 95  Weight: 243  Value: 610
Chomosome: 96  Weight: 243  Value: 607
Chomosome: 97  Weight: 239  Value: 612
Chomosome: 98  Weight: 247  Value: 609
Chomosome: 99  Weight: 246  Value: 627
```

Рисунок 3.2 –

```
Chomosome: 0  Weight: 249  Value: 559
Chomosome: 1  Weight: 250  Value: 561
Chomosome: 2  Weight: 249  Value: 559
Chomosome: 3  Weight: 250  Value: 561
Chomosome: 4  Weight: 249  Value: 559
Chomosome: 5  Weight: 250  Value: 561
Chomosome: 6  Weight: 249  Value: 590
Chomosome: 7  Weight: 245  Value: 561
Chomosome: 8  Weight: 241  Value: 566
Chomosome: 9  Weight: 245  Value: 587
Chomosome: 10  Weight: 247  Value: 569
Chomosome: 11  Weight: 247  Value: 586
Chomosome: 12  Weight: 250  Value: 568
Chomosome: 13  Weight: 242  Value: 568
Chomosome: 14  Weight: 246  Value: 560
Chomosome: 15  Weight: 246  Value: 579
Chomosome: 16  Weight: 247  Value: 566
Chomosome: 17  Weight: 250  Value: 578
Chomosome: 18  Weight: 245  Value: 572
Chomosome: 19  Weight: 247  Value: 582
Chomosome: 20  Weight: 243  Value: 576
Chomosome: 21  Weight: 248  Value: 573
Chomosome: 22  Weight: 250  Value: 582
Chomosome: 23  Weight: 241  Value: 566
Chomosome: 24  Weight: 244  Value: 573
Chomosome: 25  Weight: 246  Value: 574
Chomosome: 26  Weight: 250  Value: 590
Chomosome: 27  Weight: 247  Value: 577
Chomosome: 28  Weight: 247  Value: 571
Chomosome: 29  Weight: 246  Value: 574
Chomosome: 30  Weight: 250  Value: 560
Chomosome: 31  Weight: 244  Value: 564
Chomosome: 32  Weight: 249  Value: 582
Chomosome: 33  Weight: 249  Value: 572
Chomosome: 34  Weight: 250  Value: 585
Chomosome: 35  Weight: 250  Value: 569
Chomosome: 36  Weight: 248  Value: 569
Chomosome: 37  Weight: 248  Value: 584
Chomosome: 38  Weight: 247  Value: 576
Chomosome: 39  Weight: 244  Value: 563
Chomosome: 40  Weight: 250  Value: 566
Chomosome: 41  Weight: 247  Value: 565
Chomosome: 42  Weight: 248  Value: 569
Chomosome: 43  Weight: 248  Value: 563
Chomosome: 44  Weight: 249  Value: 582
Chomosome: 45  Weight: 246  Value: 564
Chomosome: 46  Weight: 247  Value: 571
Chomosome: 47  Weight: 246  Value: 563
Chomosome: 48  Weight: 248  Value: 570
Chomosome: 49  Weight: 249  Value: 569
Chomosome: 50  Weight: 247  Value: 577
```

```
Chomosome: 51  Weight: 245  Value: 571
Chomosome: 52  Weight: 250  Value: 571
Chomosome: 53  Weight: 246  Value: 575
Chomosome: 54  Weight: 245  Value: 571
Chomosome: 55  Weight: 249  Value: 576
Chomosome: 56  Weight: 248  Value: 573
Chomosome: 57  Weight: 249  Value: 569
Chomosome: 58  Weight: 250  Value: 581
Chomosome: 59  Weight: 249  Value: 576
Chomosome: 60  Weight: 250  Value: 579
Chomosome: 61  Weight: 247  Value: 565
Chomosome: 62  Weight: 250  Value: 569
Chomosome: 63  Weight: 248  Value: 569
Chomosome: 64  Weight: 245  Value: 572
Chomosome: 65  Weight: 248  Value: 569
Chomosome: 66  Weight: 250  Value: 571
Chomosome: 67  Weight: 250  Value: 560
Chomosome: 68  Weight: 248  Value: 584
Chomosome: 69  Weight: 250  Value: 571
Chomosome: 70  Weight: 250  Value: 580
Chomosome: 71  Weight: 247  Value: 565
Chomosome: 72  Weight: 250  Value: 577
Chomosome: 73  Weight: 247  Value: 571
Chomosome: 74  Weight: 249  Value: 576
Chomosome: 75  Weight: 248  Value: 585
Chomosome: 76  Weight: 250  Value: 571
Chomosome: 77  Weight: 249  Value: 571
Chomosome: 78  Weight: 250  Value: 577
Chomosome: 79  Weight: 250  Value: 579
Chomosome: 80  Weight: 247  Value: 583
Chomosome: 81  Weight: 250  Value: 577
Chomosome: 82  Weight: 245  Value: 571
Chomosome: 83  Weight: 250  Value: 573
Chomosome: 84  Weight: 247  Value: 577
Chomosome: 85  Weight: 250  Value: 580
Chomosome: 86  Weight: 250  Value: 586
Chomosome: 87  Weight: 248  Value: 585
Chomosome: 88  Weight: 250  Value: 573
Chomosome: 89  Weight: 248  Value: 575
Chomosome: 90  Weight: 250  Value: 596
Chomosome: 91  Weight: 247  Value: 577
Chomosome: 92  Weight: 246  Value: 575
Chomosome: 93  Weight: 245  Value: 601
Chomosome: 94  Weight: 237  Value: 575
Chomosome: 95  Weight: 243  Value: 564
Chomosome: 96  Weight: 238  Value: 569
Chomosome: 97  Weight: 228  Value: 590
Chomosome: 98  Weight: 235  Value: 593
Chomosome: 99  Weight: 229  Value: 571
```

126.1.2 Значення цільової функції зі збільшенням кількості ітерацій

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

Iterations	Record
0	37
20	121
40	121
60	168
80	168
100	168
120	183
140	205
160	205
180	205
200	205
220	222
240	337
260	391
280	391
300	392
320	415
340	415
360	415
380	415
400	504
420	510
440	510
460	510
480	525
500	525
520	528
540	528
560	528
580	528
600	544
620	544
640	544
660	544
680	546
700	553
720	561
740	561
760	568
780	568
800	575

820	575
840	577
860	577
880	578
900	581
920	581
940	595
960	598
980	598
1000	608

Таблиця 3.1

126.1.3 Графіки залежності розв'язку від числа ітерацій

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

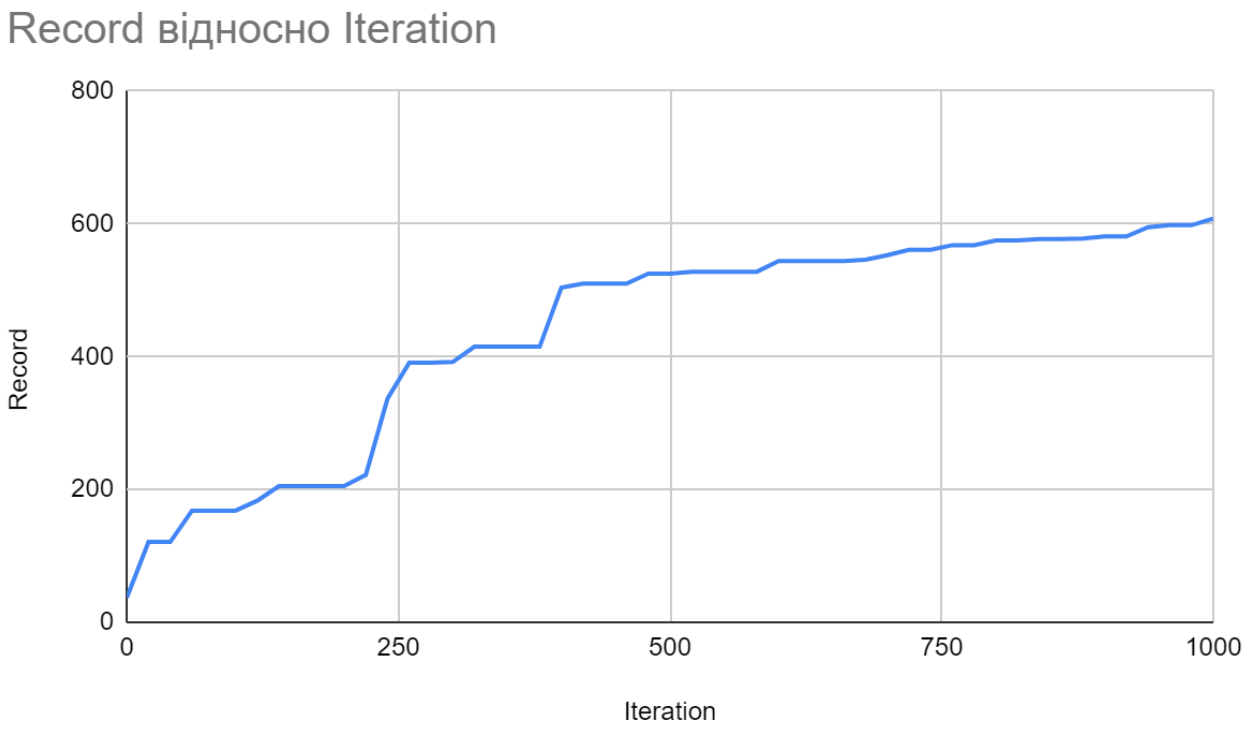

Рисунок 3.3 – Графік залежності розв'язку від числа ітерацій

Висновок

В рамках даної лабораторної роботи був використаний генетичний алгоритм, бля вирішення NP-складної задачі про рюкзак. В якості локального покращення, я змінював в новоотриманій хромосомі рандомний нульовий ген на 1, що збільшувало його значення та вагу. За допомогою такого покращення можемо швидше знайти рішення, оскільки при звичайному схрещуванні і мутації може бути таке, що найгірший ген просто не буде рости в значенні цільової функції. З проведеного дослідження отримали, що чим більше ітерації відбулось, тим повільніше зростає значення цільової функції, а наприкінці майже не змінюється.

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%.