**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**”

Варіант 1

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

ІП-з21 Стихун М.В.

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2022

Зміст

1. Мета лабораторної роботи 3
2. Завдання 4
3. Виконання 11
   1. Програмна реалізація алгоритму 11
      1. Вихідний код 11
      2. Приклади роботи 17
   2. Тестування алгоритму 18
      1. Значення цільової функції зі збільшенням кількості ітерацій 18
      2. Графіки залежності розв'язку від числа ітерацій 21

Висновок 22

Критерії оцінювання 23

# 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

package main

import (

"github.com/Nav1Cr0ss/algorithms-lab4/internal/service"

)

func main() {

populationSize := 100

generations := 100

capacity := 250

numItems := 100

service.Run(populationSize, generations, capacity, numItems)

}

package app

import (

"fmt"

"github.com/Nav1Cr0ss/algorithms-lab4/internal/domain"

"math/rand"

"sort"

"time"

)

var (

randSource = rand.NewSource(time.Now().UnixNano())

randGenerator = rand.New(randSource)

)

type Genetic struct {

title string

generations int

}

func NewGenetic() \*Genetic {

return &Genetic{title: "Genetic Algorithm"}

}

func (g \*Genetic) CreateItems(itemCount int) []domain.Item {

items := make([]domain.Item, itemCount)

for i := 0; i < itemCount; i++ {

items[i] = domain.NewRandItem()

}

return items

}

func (g \*Genetic) localImprovement(bestIndividual domain.Individual, bp domain.Backpack) domain.Individual {

for i := range bestIndividual.Genes {

if bestIndividual.Genes[i] {

newGenes := make([]bool, len(bestIndividual.Genes))

copy(newGenes, bestIndividual.Genes)

newGenes[i] = false

newFitness := bp.EvaluateFitness(domain.Individual{Genes: newGenes})

if newFitness > bestIndividual.Fitness {

bestIndividual.Genes = newGenes

bestIndividual.Fitness = newFitness

}

}

}

return bestIndividual

}

func (g \*Genetic) CalculateGenerations(population []domain.Individual, bp domain.Backpack, generations int) (int, []bool) {

populationSize := len(population)

bestOverallFitness := 0

var bestOverallGenes []bool

for generation := 0; generation < generations; generation++ {

for i := range population {

population[i].Fitness = bp.EvaluateFitness(population[i])

}

sort.SliceStable(population, func(i, j int) bool {

return population[i].Fitness > population[j].Fitness

})

bestIndividual := g.localImprovement(population[0], bp)

population[len(population)-1] = bestIndividual

fmt.Printf("Generation %d: Best Result = %d\n", generation, bestIndividual.Fitness)

if bestIndividual.Fitness > bestOverallFitness {

bestOverallFitness = bestIndividual.Fitness

bestOverallGenes = bestIndividual.Genes

}

newPopulation := make([]domain.Individual, populationSize)

for i := 0; i < populationSize; i++ {

parent1 := population[randGenerator.Intn(populationSize)]

parent2 := population[randGenerator.Intn(populationSize)]

childGenes := make([]bool, len(parent1.Genes))

crossoverPoint := randGenerator.Intn(len(parent1.Genes))

for j := 0; j < len(parent1.Genes); j++ {

if j < crossoverPoint {

childGenes[j] = parent1.Genes[j]

} else {

childGenes[j] = parent2.Genes[j]

}

}

newPopulation[i] = bp.Mutate(domain.Individual{Genes: childGenes})

}

population = newPopulation

}

return bestOverallFitness, bestOverallGenes

}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 – Вхідні данні

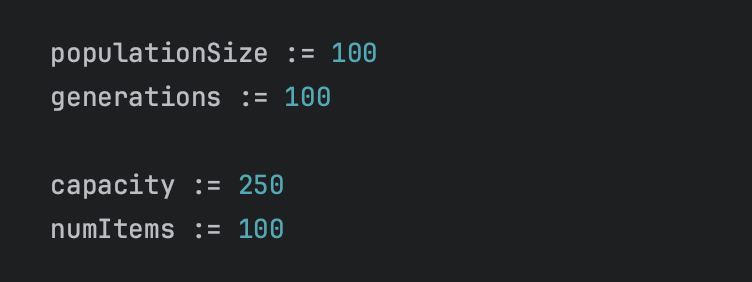
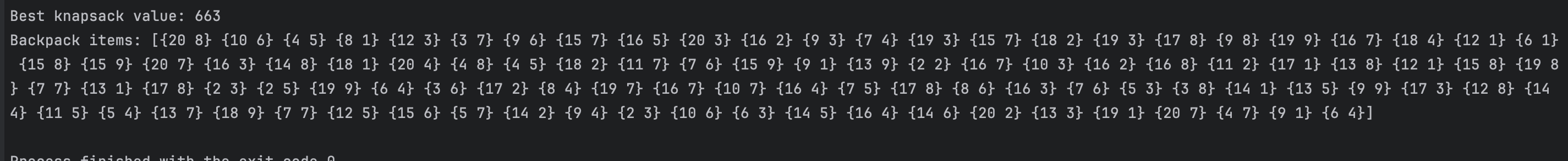


Рисунок 3.2 – Вихідні данні

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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

Таблиця 3.1 - Ітерації

| **Generation:** | **Result** |
| --- | --- |
| 1: | 511 |
| 2: | 490 |
| 3: | 523 |
| 4: | 528 |
| 5: | 493 |
| 6: | 507 |
| 7: | 514 |
| 8: | 527 |
| 9: | 552 |
| 10: | 529 |
| 11: | 511 |
| 12: | 497 |
| 13: | 509 |
| 14: | 514 |
| 15: | 542 |
| 16: | 527 |
| 17: | 522 |
| 18: | 508 |
| 19: | 529 |
| 20: | 535 |
| 21: | 560 |
| 22: | 579 |
| 23: | 526 |
| 24: | 557 |
| 25: | 521 |
| 26: | 513 |
| 27: | 506 |
| 28: | 534 |
| 29: | 602 |
| 30: | 536 |
| 31: | 573 |
| 32: | 545 |
| 33: | 592 |
| 34: | 520 |
| 35: | 564 |
| 36: | 504 |
| 37: | 551 |
| 38: | 573 |
| 39: | 526 |
| 40: | 562 |
| 41: | 535 |
| 42: | 546 |
| 43: | 521 |
| 44: | 545 |
| 45: | 544 |
| 46: | 541 |
| 47: | 507 |
| 48: | 521 |
| 49: | 539 |
| 50: | 548 |
| 51: | 567 |
| 52: | 531 |
| 53: | 560 |
| 54: | 555 |
| 55: | 558 |
| 56: | 549 |
| 57: | 517 |
| 58: | 589 |
| 59: | 587 |
| 60: | 559 |
| 61: | 571 |
| 62: | 560 |
| 63: | 540 |
| 64: | 576 |
| 65: | 563 |
| 66: | 542 |
| 67: | 603 |
| 68: | 611 |
| 69: | 629 |
| 70: | 588 |
| 71: | 585 |
| 72: | 560 |
| 73: | 553 |
| 74: | 612 |
| 75: | 562 |
| 76: | 573 |
| 77: | 556 |
| 78: | 568 |
| 79: | 578 |
| 80: | 550 |
| 81: | 550 |
| 82: | 514 |
| 83: | 573 |
| 84: | 541 |
| 85: | 536 |
| 86: | 540 |
| 87: | 536 |
| 88: | 559 |
| 89: | 540 |
| 90: | 540 |
| 91: | 568 |
| 92: | 530 |
| 93: | 525 |
| 94: | 541 |
| 95: | 544 |
| 96: | 543 |
| 97: | 567 |
| 98: | 538 |
| 99: | 554 |
| ck | tions: |
| : 6 |  |
| t: |  |

### 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 ході виконання лабораторної роботи був розглянутий алгоритм розв'язання задачі про рюкзак за допомогою генетичного алгоритму в мові програмування Go. Задача полягає в тому, щоб вибрати певну кількість предметів з обмеженою місткістю рюкзака так, щоб максимізувати загальну цінність цих предметів, дотримуючись обмежень на місткість рюкзака та вагу кожного предмета.

Після завершення всіх поколінь був знайдений найкращий результат серед усіх поколінь, і виведено вміст рюкзака (які предмети були включені) та загальну вагу рюкзака.

Лабораторна робота показала, що генетичний алгоритм є потужним інструментом для розв'язання складних комбінаторних задач, таких як задача про рюкзак.

Критерії оцінювання

При здачі лабораторної роботи до 10.12.2023 включно максимальний бал дорівнює – 5. Після 10.12.2023 максимальний бал дорівнює – 4,5.

Критерії оцінювання у відсотках від максимального балу:

* програмна реалізація алгоритму – 55%;
* робота з гіт – 20%;
* тестування алгоритму– 20%;
* висновок – 5%.

+1 додатковий бал можна отримати за виконання роботи до 3.12.2023