**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 № 5 з дисципліни

«Проектування алгоритмів»

„**Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.2**”

**Виконав(ла)**

*ІП-21 Симоненко Михайло Олександрович*

(шифр, прізвище, ім'я, по батькові)

**Перевірив**

*Головченко М. Н.*

(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1. [МЕТА ЛАБОРАТОРНОЇ РОБОТИ 3](#_bookmark0)
2. [ЗАВДАННЯ 4](#_bookmark1)
3. [ВИКОНАННЯ 11](#_bookmark2)
   1. [ПОКРОКОВИЙ АЛГОРИТМ 11](#_bookmark3)
   2. [ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ 11](#_bookmark4)
      1. [Вихідний код 11](#_bookmark5)
      2. [Приклади роботи 14](#_bookmark6)
   3. [ТЕСТУВАННЯ АЛГОРИТМУ 16](#_bookmark7)

[ВИСНОВОК 21](#_bookmark8)

[КРИТЕРІЇ ОЦІНЮВАННЯ 22](#_bookmark9)

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи розробки метаеврестичних алгоритмів для типових прикладних задач. Опрацювати методологію підбору прийнятних параметрів алгоритму.

# ЗАВДАННЯ

Згідно варіанту, формалізувати алгоритм вирішення задачі відповідно загальної методології.

Записати розроблений алгоритм у покроковому вигляді. З достатнім степенем деталізації.

Виконати його програмну реалізацію на будь-якій мові програмування. Перелік задач наведено у таблиці 2.1.

Перелік алгоритмів і досліджуваних параметрів у таблиці 2.2. Задача і алгоритм наведені в таблиці 2.3.

Змінюючи параметри алгоритму, визначити кращі вхідні параметри алгоритму. Для цього необхідно:

* обрати критерій зупинки алгоритму (кількість ітерацій або значення

ЦФ);

* зафіксувати усі параметри крім одного і змінювати цей параметр,

поки не буде досягнуто пікової ефективності;

* + після цього параметр фіксується і змінюються інші параметри;
  + далі повторюємо процедуру спочатку, з першого зафіксованого параметру;
  + зупиняємось коли будуть знайдені оптимальні параметри для даної задачі або встановлена залежність одних параметрів від інших.

Зробити узагальнений висновок в якому обов’язково описати залежність якості розв’язку від вхідних параметрів.

Таблиця 2.1 – Прикладні задачі

|  |  |
| --- | --- |
| **№** | **Задача** |
| 1 | **Задача про рюкзак** (місткість P=500, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 20 (випадкова)). Для заданої множини предметів, кожен з яких має вагу і цінність, визначити яку  кількість кожного з предметів слід взяти, так, щоб сумарна вага не |

|  |  |
| --- | --- |
|  | перевищувала задану, а сумарна цінність була максимальною. Задача часто виникає при розподілі ресурсів, коли наявні фінансові обмеження, і вивчається в таких областях, як комбінаторика,  інформатика, теорія складності, криптографія, прикладна математика. |
| 2 | **Задача комівояжера** (300 вершин, відстань між вершинами випадкова від 5 до 150) полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай задано, що маршрут повинен проходити через кожне місто тільки один раз, в такому випадку розв'язок знаходиться серед гамільтонових циклів.  **Розглядається симетричний, асиметричний та змішаний варіанти.** В загальному випадку, асиметрична задача комівояжера відрізняється тим, що ребра між вершинами можуть мати різну вагу в залежності від напряму, тобто, задача моделюється орієнтованим графом. Таким чином, окрім ваги ребер графа, слід також зважати і на те, в якому напрямку знаходяться ребра.  У випадку симетричної задачі всі пари ребер між одними й тими самими вершинами мають однакову вагу.  У випадку реальних міст може бути як симетричною, так і асиметричною в залежності від тривалості або довжини маршрутів і напряму руху.  Застосування:   * доставка товарів (в цьому випадку може бути більш доречна постановка транспортної задачі - доставка в кілька магазинів з декількох складів); * доставка води; * моніторинг об'єктів; |

|  |  |
| --- | --- |
|  | * поповнення банкоматів готівкою; * збір співробітників для доставки вахтовим методом. |
| 3 | **Розфарбовування графа** (300 вершин, степінь вершини не більше 30, але не менше 2) – називають таке приписування кольорів (або натуральних чисел) його вершинам, що ніякі дві суміжні вершини не набувають однакового кольору. Найменшу можливу кількість кольорів у розфарбуванні називають хроматичне число.  Застосування:   * розкладу для освітніх установ; * розкладу в спорті; * планування зустрічей, зборів, інтерв'ю; * розклади транспорту, в тому числі - авіатранспорту; * розкладу для комунальних служб; |
| 4 | **Задача вершинного покриття** (300 вершин, степінь вершини не більше 30, але не менше 2)**.** Вершинне покриття для неорієнтованого графа G = (V, E) - це множина його вершин S, така, що, у кожного ребра графа хоча б один з кінців входить в вершину з S.  Задача вершинного покриття полягає в пошуку вершинного покриття найменшого розміру для заданого графа (цей розмір називається числом вершинного покриття графа).  На вході: Граф G = (V, E).  Результат: множина C ⊆ V - найменше вершинне покриття графа G. |

|  |  |
| --- | --- |
|  | Застосування:   * розміщення пунктів обслуговування; * призначення екіпажів на транспорт; * проектування інтегральних схем і конвеєрних ліній. |
| 5 | **Задача про кліку** (300 вершин, степінь вершини не більше 30, але не менше 2)**.** Клікою в неорієнтованому графі називається підмножина вершин, кожні дві з яких з'єднані ребром графа. Іншими словами, це повний підграф первісного графа. Розмір кліки визначається як число вершин в ній.  Задача про кліку існує у двох варіантах: у **задачі розпізнавання** потрібно визначити, чи існує в заданому графі G кліка розміру k, тоді як в **обчислювальному варіанті** потрібно знайти в заданому графі G кліку максимального розміру або всі максимальні кліки (такі, що не можна збільшити).  Застосування:   * біоінформатика; * електротехніка; |
| 6 | **Задача про найкоротший шлях** (300 вершин, відстань між вершинами  випадкова від 5 до 150, степінь вершини не більше 10, але не менше 1) - |

|  |  |
| --- | --- |
|  | задача пошуку найкоротшого шляху (ланцюга) між двома точками (вершинами) на графі, в якій мінімізується сума ваг ребер, що складають шлях.  Важливість задачі визначається її різними практичними застосуваннями. Наприклад, в GPS-навігаторах здійснюється пошук найкоротшого шляху між точкою відправлення і точкою призначення. Як вершин виступають перехрестя, а дороги є ребрами, які лежать між ними. Якщо сума довжин доріг між перехрестями мінімальна, тоді  знайдений шлях найкоротший. |

Таблиця 2.2 – Варіанти алгоритмів і досліджувані параметри

|  |  |
| --- | --- |
| **№** | **Алгоритми і досліджувані параметри** |
| 1 | **Генетичний алгоритм:**   * оператор схрещування (мінімум 3); * мутація (мінімум 2); * оператор локального покращення (мінімум 2). |
| 2 | **Мурашиний алгоритм**:   * α; * β; * ρ; * Lmin; * кількість мурах М і їх типи (елітні, тощо…); * маршрути з однієї чи різних вершин. |
| 3 | **Бджолиний алгоритм:**   * кількість ділянок; * кількість бджіл (фуражирів і розвідників). |

Таблиця 2.3 – Варіанти задач і алгоритмів

|  |  |
| --- | --- |
| **№** | **Задачі і алгоритми** |
| 24 | Задача про рюкзак + Бджолиний алгоритм |

1. ВИКОНАННЯ
   1. Покроковий алгоритм
2. Перехід на наступну ділянку
3. Відправка бджіл розвідників на випадкові предмети
4. Передача інформації про нектар на предметах
5. Відправка бджіл-робітників на обрані предмети в залежності від нектару на кожному предметі
6. Вибір предмета де прибуло найбільше робітників
7. Перевірка портфеля на наявність вільного предмету для даного предмету: якщо місце є, кладемо предмет
8. Перевірка на вільне місце у портфелі: якщо місце є – перехід до п. 2 інакше - вибір найкращої ділянки, перехід до п.1
   1. Програмна реалізація алгоритму
      1. Вихідний код

public class ABCManager

{

public List<int> Manage(Hive hive, List<Item> items, int backpackWeight)

{

ABC abc = new ABC();

FinalCalculate finalCalculate = new FinalCalculate(); NumAreaManager numAreaManager = new NumAreaManager(); List<int> usedItems = abc.Start(hive, items, backpackWeight); int maxCost = finalCalculate.Cost(items, usedItems);

int numArea = numAreaManager.Manage(); MassageWriter.Write(Massages.BeeSend); for (int i = 0; i < numArea - 1; i++)

{

List<int> newUsedItems = abc.Start(hive, items, backpackWeight); int cost = finalCalculate.Cost(items, newUsedItems);

if (cost > maxCost)

{

maxCost = cost;

usedItems = newUsedItems;

}

}

return usedItems;

}

}

public class ABC

{

public List<int> Start(Hive hive, List<Item> items, int backpackWeight)

{

ScoutsSendler scoutsSendler = new ScoutsSendler(); WorkersSendler workersSendler = new WorkersSendler(); ItemsAnalisis itemsAnalisis = new ItemsAnalisis(); MaxItemFinder maxItemFinder = new MaxItemFinder(); MinWeightFinder minWeightFinder = new MinWeightFinder(); int minWeight = minWeightFinder.Find(items);

List<int> usedItems = new List<int>(Enumerable.Repeat(0, items.Count)); int totalWeight = 0;

while (backpackWeight - totalWeight > minWeight)

{

List<int> randomItems = scoutsSendler.Send(hive.scouts, items); List<int> probabilities = itemsAnalisis.CalcProbability(items,

randomItems);

List<int> distribution = workersSendler.Send(probabilities, hive.workers);

int chooseItemIndex = randomItems[maxItemFinder.Find(distribution)];

if (totalWeight + items[chooseItemIndex].Weight <= backpackWeight)

{

totalWeight += items[chooseItemIndex].Weight; usedItems[chooseItemIndex]++;

}

}

return usedItems;

}

}

public class ScoutsSendler

{

public List<int> Send(int scouts, List<Item> items)

{

Random random = new Random();

List<int> randomItems = new List<int>();

List<int> itemsIndexes = new List<int>(Enumerable.Range(0, items.Count));

while (scouts > 0 && itemsIndexes.Count != 0)

{

int randomIndex = random.Next(0, itemsIndexes.Count); randomItems.Add(itemsIndexes[randomIndex]); itemsIndexes.RemoveAt(randomIndex);

scouts--;

}

return randomItems;

}

}

public class ItemsAnalisis

{

public List<int> CalcProbability(List<Item> items, List<int> randomItems)

{

List<int> probabilities = new List<int>();

SumNectarCalculator sumNectarCalculator = new SumNectarCalculator(); double sumNectar = sumNectarCalculator.Culc(items, randomItems);

for (int i = 0; i < randomItems.Count; i++)

{

probabilities.Add((int)Math.Round(items[randomItems[i]].Nectar() / sumNectar \* 1000));

}

return probabilities;

}

}

public class WorkersSendler

{

public List<int> Send(List<int> probabilities, int workers)

{

ItemSelecter itemSelecter = new ItemSelecter(); Random random = new Random();

List<int> distribution = new List<int>(Enumerable.Repeat(0, probabilities.Count));

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

{

int rand = random.Next(0, 1000);

int indexItem = itemSelecter.Select(probabilities, rand); distribution[indexItem]++;

}

return distribution;

}

}

* + 1. Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 – Приклад для 10 предметів, 50 фуражирів та розвідників

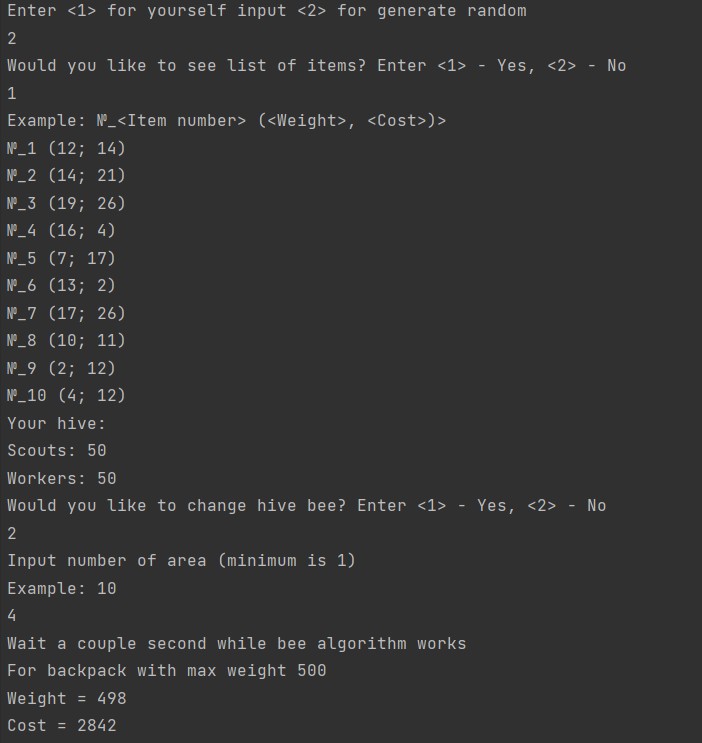
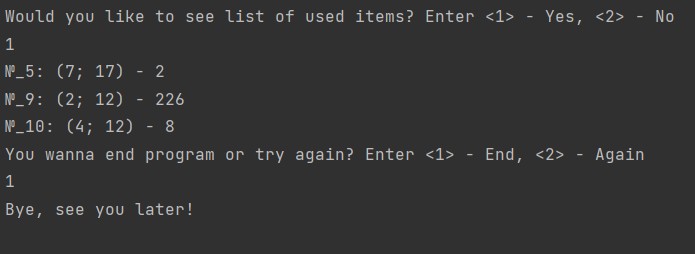


Рисунок 3.2 – Приклад розподілу предметів які були використані



* 1. Тестування алгоритму

ЦФ

Тестування залежності ЦФ від кількості фуражирів Таблиця 3.1 Стала кількість розвідників = 5

|  |  |  |
| --- | --- | --- |
| К-ість фуражирів | Загальна вага | Загальна цінність |
| 1 | 499 | 1574 |
| 2 | 500 | 1877 |
| 3 | 499 | 1837 |
| 4 | 499 | 2250 |
| 5 | 500 | 2846 |
| 10 | 500 | 2510 |
| 20 | 499 | 3230 |
| 30 | 500 | 2819 |
| 40 | 499 | 2880 |
| 50 | 499 | 2806 |
| 60 | 499 | 2928 |
| 70 | 499 | 3195 |
| 80 | 499 | 3270 |
| 90 | 499 | 3665 |
| 100 | 500 | 3447 |

Графік 3.1



Залежність ЦФ від к-ті фуражирів

4000

3500

3000

2500

2000

1500

1000

500

0

0

20

40

60

Фуражири

80

100

120

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Таблиця 3.2 Стала кількість розвідників = 50

ЦФ

|  |  |  |
| --- | --- | --- |
| К-ість фуражирів | Загальна вага | Загальна цінність |
| 1 | 500 | 2261 |
| 2 | 500 | 2420 |
| 3 | 499 | 2333 |
| 4 | 499 | 2730 |
| 5 | 499 | 3180 |
| 10 | 499 | 5085 |
| 20 | 499 | 7121 |
| 30 | 499 | 8014 |
| 40 | 499 | 8483 |
| 50 | 499 | 9102 |
| 60 | 499 | 8998 |
| 70 | 500 | 9626 |
| 80 | 499 | 9609 |
| 90 | 499 | 9899 |
| 100 | 499 | 10037 |

Графік 3.2



Залежність ЦФ від к-ті фуражирів

12000

10000

8000

6000

4000

2000

0

0

20

40

60

Фуражири

80

100

120

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Тестування залежності ЦФ від кількості Розвідників Таблиця 3.3 Стала кількість фуражирів = 20



ЦФ

|  |  |  |
| --- | --- | --- |
| К-ість розвідників | Загальна вага | Загальна цінність |
| 1 | 499 | 1048 |
| 2 | 500 | 1447 |
| 3 | 500 | 1924 |
| 4 | 500 | 2590 |
| 5 | 500 | 2420 |
| 10 | 500 | 4672 |
| 20 | 500 | 6796 |
| 30 | 499 | 6866 |
| 40 | 499 | 7116 |
| 50 | 499 | 6893 |
| 60 | 499 | 6941 |
| 70 | 500 | 6182 |
| 80 | 499 | 6922 |
| 90 | 499 | 6550 |
| 100 | 499 | 6206 |

Графік 3.3



Залежність ЦФ від к-ті Розвідників

8000

7000

6000

5000

4000

3000

2000

1000

0

0

20

40

60

Розвідники

80

100

120

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Таблиця 3.4 Стала кількість фуражирів = 40

ЦФ

|  |  |  |
| --- | --- | --- |
| К-ість  розвідників | Загальна  вага | Загальна цінність |
| 1 | 499 | 1080 |
| 2 | 499 | 1469 |
| 3 | 500 | 2296 |
| 4 | 499 | 2725 |
| 5 | 500 | 3101 |
| 10 | 500 | 5342 |
| 20 | 499 | 7166 |
| 30 | 499 | 8094 |
| 40 | 499 | 8755 |
| 50 | 500 | 8307 |
| 60 | 499 | 8784 |
| 70 | 499 | 8439 |
| 80 | 499 | 8859 |
| 90 | 499 | 8619 |
| 100 | 499 | 8794 |

Графік 3.4



Залежність ЦФ від к-ті Розвідників

10000

9000

8000

7000

6000

5000

4000

3000

2000

1000

0

0

20

40

60

Розвідники

80

100

120

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Тестування залежності ЦФ від кількості ділянок

ЦФ

Таблиця 3.5 Стала кількість розвідників та фуражирів = 50

|  |  |  |
| --- | --- | --- |
| К-ість ділянок | Загальна вага | Загальна  цінність |
| 1 | 500 | 8764 |
| 2 | 499 | 8466 |
| 3 | 500 | 9272 |
| 4 | 500 | 9257 |
| 5 | 499 | 9254 |
| 10 | 500 | 9041 |
| 20 | 499 | 9333 |
| 30 | 499 | 9507 |
| 40 | 499 | 9391 |
| 50 | 499 | 9512 |
| 60 | 499 | 9468 |
| 70 | 499 | 9482 |
| 80 | 499 | 9587 |
| 90 | 499 | 9392 |
| 100 | 500 | 9938 |

Графік 3.5



Залежність ЦФ від к-ті фуражирів

10200

10000

9800

9600

9400

9200

9000

8800

8600

8400

8200

0

20

40

60

Кількість делянок

80

100

120

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

ВИСНОВОК

В рамках даної лабораторної роботи було досліджено алгоритм бджолиної колонії(варіант ABC відповідно до лекції) для задачі про рюкзак. У ході чого було розроблено програмне забезпечення для дослідження даного алгоритму у залежності від кількості фуражирів, розвідників та кількості ділянок, і даній задачі кількість ділянок фактична рівна кількості викликам основного алгоритму у ході чого обирається найуспішніша ділянка.

Провівши тестування та аналіз алгоритму можна зробити наступні висновки щодо початкових даних для алгоритму:

1. ЦФ покращується при збільшені кількості фуражирів оскільки тоді буде більша вірогідність вибору предмету, у якого більше нектару
2. ЦФ покращується при кількості розвідників рівній 20% від кількості предметів та зберігає потенціал із незначним приростом після збільшення кількості розвідників.
3. ЦФ покращується від кількості ділянок.

КРИТЕРІЇ ОЦІНЮВАННЯ

При здачі лабораторної роботи до 24.12.2023 включно максимальний бал дорівнює – 5. Після 24.12.2023 максимальний бал дорівнює – 4,5.

Критерії оцінювання у відсотках від максимального балу:

* покроковий алгоритм – 10%;
* програмна реалізація алгоритму – 45%;
* робота з гіт – 20%;
* тестування алгоритму– 20%;
* висновок – 5%.

+1 додатковий бал можна отримати за виконання та захист роботи до 17.12.2023