

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)

ІІ-14 Бабіч Денис

(шифр, прізвище, ім'я, по батькові)

Перевірив

Головченко М.Н.

(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1	МЕТА ЛАБОРАТОРНОЇ РОБОТИ	3
2	ЗАВДАННЯ	4
3	ВИКОНАННЯ	6
3.1	ПОКРОКОВИЙ АЛГОРИТМ	6
3.2	ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ	9
3.2.1	<i>Вихідний код.....</i>	9
3.2.2	<i>Приклади роботи.....</i>	12
3.3	ТЕСТУВАННЯ АЛГОРИТМУ	14
	ВИСНОВОК	16
	КРИТЕРІЇ ОЦІНЮВАННЯ	17

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

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

2 ЗАВДАННЯ

Згідно варіанту, формалізувати алгоритм вирішення задачі відповідно загальної методології.

Записати розроблений алгоритм у покроковому вигляді. З достатнім ступенем 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.

1	<p>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.</p> <p>Задача часто виникає при розподілі ресурсів, коли наявні фінансові обмеження, і вивчається в таких областях, як комбінаторика, інформатика, теорія складності, криптографія, прикладна математика.</p>
3	<p>Бджолиний алгоритм:</p> <ul style="list-style-type: none"> – кількість ділянок; – кількість бджіл (фуражирів і розвідників).
2	Задача про рюкзак + Бджолиний алгоритм

3.1 Покроковий алгоритм

```

function Fill(items) returns int
    weight = 0
    selectedItems = List<Item>()
    items = items.OrderByDescending(item => item.Nectar).ToList()

    foreach (item in items) do
        if (weight + item.Weight <= this.Capacity) then
            weight += item.Weight
            selectedItems.Add(item)
            Console.WriteLine($"Був доданий новий предмет.
                               Вага: {item.Weight}. Ціна: {item.Price}")
        else
            break
        end if
    end loop

    return selectedItems.Sum(item => item.Price)

function Fill(items, numberOfScouts, numberOfForagers, numberOfFields) returns it
    appropriateItems = GetAppropriateItems(items)
    while (true) do
        if (items.Count < 1 OR Weight > Capacity
            OR appropriateItems.Count == items.Count)
            break
        end if

        SpawnScouts(numberOfFields, numberOfScouts, items, out scoutedItems)
        SpawnForagers(numberOfForagers, scoutedItems, items, out scoutedItems)
        scoutedItems = GetItems(scoutedItems, appropriateItems)

        if (scoutedItems.Count != 0) then
            this.Weight += CalculateWeight(scoutedItems)
            this.items.AddRange(scoutedItems)
            foreach (Item item in scoutedItems) do
                Console.WriteLine($"Був доданий новий предмет.
                                   Вага: {item.Weight}. Ціна: {item.Price}")
            end loop
            items.RemoveAll(item => scoutedItems.Contains(item))
        end if
    end while
    return CalculatePrice(this.items)

```

```
function SpawnScouts(numberOfFields, numberOfScouts, items, out scoutedItems)
    scoutedItems = new List<Item>()
    scoutedItems.AddRange(GetRandomItems(0, items.Count, numberOfScouts, items))
```

```
function SpawnForagers(count, itemsToVisit, items, out visitedItems) returns void
    visitedItems = new List<Item>()
    visitedItems.AddRange(itemsToVisit)

    if (count > visitedItems.Count) then
        otherItems = items.Except(visitedItems).ToList()
        visitedItems.AddRange(GetRandomItems(0, otherItems.Count,
                                              count - visitedItems.Count, items))
    end if
```

```
function GetRandomItems(min, max, numberOfElements, items) returns List<Item>
    Random random = Random()
    selectedItems = List<Item>()

    positions[] = Enumerable.Range(min, max)
                          .OrderBy(i => random.Next(min, max))
                          .Take(numberOfElements).ToArray()

    for (i = 0; i < positions.Length; ++i) do
        selectedItems.Add(items[positions[i]])
    end loop

    return selectedItems
```

```
function GetAppropriateItems(items) return List<Item>
    weight = 0
    appropriateItems = List<Item>()
    items = items.OrderByDescending(item => item.Nectar).ToList()

    GetList(items, ref weight)

    newItemList = items.Except(appropriateItems)
                  .OrderBy(item => item.Weight)
                  .OrderBy(item => item.Price)
                  .ToList()

    GetList(newItemList, ref weight)
    return appropriateItems;
```

```
function GetList(items, ref weight) => returns void
    foreach (item in newItemList) do
        if (weight + item.Weight < this.Capacity) then
            weight += item.Weight
            appropriateItems.Add(item)
        else
            break
        end if
    end loop
```

```
function GetItems(scoutedItems, appropriateItems) returns List<Item>
    scoutedItems.Intersect(appropriateItems).ToList()
```

```
function CalculateWeight(selectedItems) returns int
    selectedItems.Sum(item => item.Weight)
```

```
function CalculatePrice(selectedItems) returns int
    selectedItems.Sum(item => item.Price)
```

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

Program.cs

```
Console.InputEncoding = System.Text.Encoding.Unicode;
Console.OutputEncoding = System.Text.Encoding.Unicode;

int capacity = InputInt("Введіть значення місткості рюкзака. Значення згідно до варіанту - 500. Введіть значення: ", 0, Int32.MaxValue);
int numberOfItems = InputInt("Введіть значення кількості предметів. Значення згідно до варіанту - 100. Введіть значення: ", 0, Int32.MaxValue);

int minPrice = InputInt("Введіть мінімальну цінність предметів у рюкзаку. Значення згідно до варіанту - 2. Введіть значення: ", 0, Int32.MaxValue);
int maxPrice = InputInt("Введіть максимальну цінність предметів у рюкзаку. Значення згідно до варіанту - 30. Введіть значення: ", 0, Int32.MaxValue);

while (minPrice > maxPrice)
{
    Console.WriteLine("Мінімальна ціна має бути менше максимальної!");
    minPrice = InputInt("Введіть мінімальну цінність предметів у рюкзаку. Значення згідно до варіанту - 2. Введіть значення: ", 0, Int32.MaxValue);
    maxPrice = InputInt("Введіть максимальну цінність предметів у рюкзаку. Значення згідно до варіанту - 30. Введіть значення: ", 0, Int32.MaxValue);
}

int minWeight = InputInt("Введіть мінімальну вагу предметів у рюкзаку. Значення згідно до варіанту - 1. Введіть значення: ", 0, Int32.MaxValue);
int maxWeight = InputInt("Введіть максимальну вагу предметів у рюкзаку. Значення згідно до варіанту - 20. Введіть значення: ", 0, Int32.MaxValue);

while (minWeight > maxWeight)
{
    Console.WriteLine("Мінімальна вага має бути менше максимальної!");
    minWeight = InputInt("Введіть мінімальну вагу предметів у рюкзаку. Значення згідно до варіанту - 1. Введіть значення: ", 0, Int32.MaxValue);
    maxWeight = InputInt("Введіть максимальну вагу предметів у рюкзаку. Значення згідно до варіанту - 20. Введіть значення: ", 0, Int32.MaxValue);
}

int numberOfScouts = InputInt("Введіть кількість розвідників: ", 0, Int32.MaxValue);
int numberOfForagers = InputInt("Введіть кількість фуражирів: ", 0, Int32.MaxValue);
int numberOfFields = InputInt("Введіть кількість ділянок для дослідження: ", 0, Int32.MaxValue);

Backpack backpack = new Backpack(capacity);
List<Item> items = new List<Item>(numberOfItems);

for (int i = 0; i < numberOfItems; ++i)
    items.Add(Item.GenerateItemsParameters(minWeight, maxWeight, minPrice, maxPrice));

Console.WriteLine();
Console.WriteLine($"Максимальна ціна стандартним методом: {backpack.Fill(items)}");
Console.WriteLine();
Console.WriteLine($"Максимальна ціна за допомогою бджолиного алгоритму: {backpack.Fill(items, numberOfScouts, numberOfForagers, numberOfFields)}");

private static int InputInt(string text, int min, int max)
{
    int value;

    do
    {
        Console.Write(text);

        if (Int32.TryParse(Console.ReadLine(), out value))
        {
            if (value <= min)
            {
                Console.WriteLine($"Введене значення має бути більше {min}!");
                continue;
            }

            if (value >= max)
            {
                Console.WriteLine($"Введене значення має бути менше {max}!");
                continue;
            }

            return value;
        }

        Console.WriteLine("Введена величина має бути числом!");
    } while (true);
}
```

Backpack.cs

```
internal sealed class Backpack
{
    public readonly int Capacity;

    4 references
    public int Weight { get; private set; }

    1 reference
    public int TotalPrice { get; private set; }

    private List<Item> items;

    1 reference
    public Backpack(int capacity)
    {
        this.Capacity = capacity;
        this.items = new List<Item>();
    }

    0 references
    public void Add(Item item)
    {
        if (item.Weight + this.Weight <= this.Capacity)
        {
            items.Add(item);
            this.Weight += item.Weight;
            this.TotalPrice += item.Price;
        }
    }
}
```

```
public int Fill(List<Item> items)
{
    int weight = 0;
    List<Item> selectedItems = new List<Item>();
    items = items.OrderByDescending(item => item.Nectar).ToList();

    foreach (Item item in items)
    {
        if (weight + item.Weight <= this.Capacity)
        {
            weight += item.Weight;
            selectedItems.Add(item);
            Console.WriteLine($"Був доданий новий предмет. Вага: {item.Weight}. Ціна: {item.Price}");
        }
        else
            break;
    }

    return selectedItems.Sum(item => item.Price);
}
```

```
List<Item> scoutedItems;
List<Item> appropriateItems = GetAppropriateItems(items);

while (true)
{
    if (items.Count < 1 || this.Weight > this.Capacity || appropriateItems.Count == this.items.Count)
        break;

    SpawnScouts(numberOfFields, numberOfScouts, items, out scoutedItems);
    SpawnForagers(numberOfForagers, scoutedItems, items, out scoutedItems);
    scoutedItems = GetItems(scoutedItems, appropriateItems);

    if (scoutedItems.Count != 0)
    {
        this.Weight += CalculateWeight(scoutedItems);
        this.items.AddRange(scoutedItems);

        foreach (Item item in scoutedItems)
            Console.WriteLine($"Був доданий новий предмет. Вага: {item.Weight}. Ціна: {item.Price}");

        items.RemoveAll(item => scoutedItems.Contains(item));
    }
}

return CalculatePrice(this.items);
```

```

void SpawnScouts(int numberOfFields, int numberOfScouts, List<Item> items, out List<Item> scoutedItems)
{
    scoutedItems = new List<Item>();
    scoutedItems.AddRange(GetRandomItems(0, items.Count, numberOfScouts, items));
}

void SpawnForagers(int count, List<Item> itemsToVisit, List<Item> items, out List<Item> visitedItems)
{
    visitedItems = new List<Item>();
    visitedItems.AddRange(itemsToVisit);

    if (count > visitedItems.Count)
    {
        List<Item> otherItems = items.Except(visitedItems).ToList();
        visitedItems.AddRange(GetRandomItems(0, otherItems.Count, count - visitedItems.Count, items));
    }
}

```

```

List<Item> GetRandomItems(int min, int max, int numberOfElements, List<Item> items)
{
    Random random = new Random();
    List<Item> selectedItems = new List<Item>();

    int[] positions = Enumerable.Range(min, max).OrderBy(i => random.Next(min, max)).Take(numberOfElements).ToArray();

    for (int i = 0; i < positions.Length; ++i)
        selectedItems.Add(items[positions[i]]);

    return selectedItems;
}

List<Item> GetAppropriateItems(List<Item> items)
{
    int weight = 0;
    List<Item> appropriateItems = new List<Item>();
    items = items.OrderByDescending(item => item.Nectar).ToList();

    GetList(items, ref weight);

    List<Item> newItemList = items.Except(appropriateItems).OrderBy(item => item.Weight).OrderBy(item => item.Price).ToList();

    GetList(newItemList, ref weight);

    return appropriateItems;

    void GetList(List<Item> items, ref int weight)
    {

```

```

List<Item> GetItems(List<Item> scoutedItems, List<Item> appropriateItems) => scoutedItems.Intersect(appropriateItems).ToList();

int CalculateWeight(List<Item> selectedItems) => selectedItems.Sum(item => item.Weight);

int CalculatePrice(List<Item> selectedItems) => selectedItems.Sum(item => item.Price);

```

```

internal sealed class Item
{
    12 references
    public int Weight { get; }

    8 references
    public int Price { get; }

    2 references
    public float Nectar { get => (float)this.Price / this.Weight; }

    1 reference
    public Item(int weight, int price)
    {
        this.Weight = weight;
        this.Price = price;
    }

    1 reference
    public static Item GenerateItemsParameters(int minWeight, int maxWeight, int minPrice, int maxPrice)
    {
        Random random = new Random();
        return new Item(random.Next(minWeight, maxWeight), random.Next(minPrice, maxPrice));
    }
}

```

3.2.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

```
Був доданий новий предмет. Вага: 2. Ціна: 25
Був доданий новий предмет. Вага: 1. Ціна: 12
Був доданий новий предмет. Вага: 2. Ціна: 22
Був доданий новий предмет. Вага: 1. Ціна: 10
Був доданий новий предмет. Вага: 3. Ціна: 25
Був доданий новий предмет. Вага: 3. Ціна: 23
Був доданий новий предмет. Вага: 3. Ціна: 23
Був доданий новий предмет. Вага: 2. Ціна: 15
Був доданий новий предмет. Вага: 3. Ціна: 22
Був доданий новий предмет. Вага: 4. Ціна: 25
Був доданий новий предмет. Вага: 4. Ціна: 24
Був доданий новий предмет. Вага: 5. Ціна: 29
Був доданий новий предмет. Вага: 5. Ціна: 28
Був доданий новий предмет. Вага: 4. Ціна: 21
Був доданий новий предмет. Вага: 6. Ціна: 28
Був доданий новий предмет. Вага: 2. Ціна: 9
Був доданий новий предмет. Вага: 5. Ціна: 21
Був доданий новий предмет. Вага: 3. Ціна: 11
Був доданий новий предмет. Вага: 6. Ціна: 21
Був доданий новий предмет. Вага: 6. Ціна: 20
Був доданий новий предмет. Вага: 9. Ціна: 29
Був доданий новий предмет. Вага: 5. Ціна: 16
Був доданий новий предмет. Вага: 8. Ціна: 25
Був доданий новий предмет. Вага: 6. Ціна: 18
Був доданий новий предмет. Вага: 1. Ціна: 3
Був доданий новий предмет. Вага: 3. Ціна: 9
Був доданий новий предмет. Вага: 4. Ціна: 12
Був доданий новий предмет. Вага: 10. Ціна: 29
Був доданий новий предмет. Вага: 9. Ціна: 25
Був доданий новий предмет. Вага: 4. Ціна: 11
Був доданий новий предмет. Вага: 2. Ціна: 5
Був доданий новий предмет. Вага: 11. Ціна: 27
Був доданий новий предмет. Вага: 5. Ціна: 12
Був доданий новий предмет. Вага: 5. Ціна: 12
Був доданий новий предмет. Вага: 5. Ціна: 12
Був доданий новий предмет. Вага: 11. Ціна: 25
Був доданий новий предмет. Вага: 12. Ціна: 27
Був доданий новий предмет. Вага: 11. Ціна: 24
Був доданий новий предмет. Вага: 11. Ціна: 23
Був доданий новий предмет. Вага: 14. Ціна: 29
Був доданий новий предмет. Вага: 13. Ціна: 26
Був доданий новий предмет. Вага: 14. Ціна: 27
Був доданий новий предмет. Вага: 6. Ціна: 11
Був доданий новий предмет. Вага: 11. Ціна: 19
Був доданий новий предмет. Вага: 18. Ціна: 29
Був доданий новий предмет. Вага: 10. Ціна: 16
Був доданий новий предмет. Вага: 7. Ціна: 11
Був доданий новий предмет. Вага: 14. Ціна: 21
Був доданий новий предмет. Вага: 18. Ціна: 27
Був доданий новий предмет. Вага: 14. Ціна: 21
Був доданий новий предмет. Вага: 8. Ціна: 12
Був доданий новий предмет. Вага: 11. Ціна: 16
Був доданий новий предмет. Вага: 19. Ціна: 27
Був доданий новий предмет. Вага: 13. Ціна: 18
Був доданий новий предмет. Вага: 9. Ціна: 12
Був доданий новий предмет. Вага: 7. Ціна: 9
Був доданий новий предмет. Вага: 19. Ціна: 24
Був доданий новий предмет. Вага: 4. Ціна: 5
Був доданий новий предмет. Вага: 19. Ціна: 22
Був доданий новий предмет. Вага: 7. Ціна: 8
Був доданий новий предмет. Вага: 17. Ціна: 18
Був доданий новий предмет. Вага: 17. Ціна: 18
```

Максимальна ціна стандартним методом: 1224

Рисунок 3.1 та 3.2 – Приклад роботи алгоритму при стандартних початкових значеннях згідно до варіанту та кількості розвідників – 5, кількість фуражирів – 100, кількість ділянок – 50

```

Був доданий новий предмет. Вага: 17. Ціна: 18
Був доданий новий предмет. Вага: 1. Ціна: 3
Був доданий новий предмет. Вага: 4. Ціна: 11
Був доданий новий предмет. Вага: 14. Ціна: 29
Був доданий новий предмет. Вага: 4. Ціна: 25
Був доданий новий предмет. Вага: 11. Ціна: 19
Був доданий новий предмет. Вага: 2. Ціна: 2
Був доданий новий предмет. Вага: 3. Ціна: 22
Був доданий новий предмет. Вага: 3. Ціна: 23
Був доданий новий предмет. Вага: 11. Ціна: 27
Був доданий новий предмет. Вага: 6. Ціна: 21
Був доданий новий предмет. Вага: 11. Ціна: 23
Був доданий новий предмет. Вага: 5. Ціна: 16
Був доданий новий предмет. Вага: 11. Ціна: 24
Був доданий новий предмет. Вага: 8. Ціна: 12
Був доданий новий предмет. Вага: 9. Ціна: 12
Був доданий новий предмет. Вага: 18. Ціна: 27
Був доданий новий предмет. Вага: 14. Ціна: 21
Був доданий новий предмет. Вага: 6. Ціна: 18
Був доданий новий предмет. Вага: 5. Ціна: 12
Був доданий новий предмет. Вага: 4. Ціна: 12
Був доданий новий предмет. Вага: 2. Ціна: 27
Був доданий новий предмет. Вага: 13. Ціна: 18
Був доданий новий предмет. Вага: 5. Ціна: 12
Був доданий новий предмет. Вага: 14. Ціна: 27
Був доданий новий предмет. Вага: 19. Ціна: 24
Був доданий новий предмет. Вага: 3. Ціна: 11
Був доданий новий предмет. Вага: 19. Ціна: 22
Був доданий новий предмет. Вага: 19. Ціна: 27
Був доданий новий предмет. Вага: 4. Ціна: 24
Був доданий новий предмет. Вага: 10. Ціна: 29
Був доданий новий предмет. Вага: 2. Ціна: 5
Був доданий новий предмет. Вага: 13. Ціна: 26
Був доданий новий предмет. Вага: 11. Ціна: 25
Був доданий новий предмет. Вага: 11. Ціна: 16
Був доданий новий предмет. Вага: 3. Ціна: 25
Був доданий новий предмет. Вага: 9. Ціна: 25
Був доданий новий предмет. Вага: 2. Ціна: 15
Був доданий новий предмет. Вага: 6. Ціна: 11
Був доданий новий предмет. Вага: 1. Ціна: 12
Був доданий новий предмет. Вага: 5. Ціна: 28
Був доданий новий предмет. Вага: 7. Ціна: 11
Був доданий новий предмет. Вага: 3. Ціна: 9
Був доданий новий предмет. Вага: 18. Ціна: 29
Був доданий новий предмет. Вага: 2. Ціна: 9
Був доданий новий предмет. Вага: 6. Ціна: 28
Був доданий новий предмет. Вага: 4. Ціна: 5
Був доданий новий предмет. Вага: 8. Ціна: 25
Був доданий новий предмет. Вага: 7. Ціна: 9
Був доданий новий предмет. Вага: 3. Ціна: 23
Був доданий новий предмет. Вага: 5. Ціна: 12
Був доданий новий предмет. Вага: 5. Ціна: 29
Був доданий новий предмет. Вага: 7. Ціна: 8
Був доданий новий предмет. Вага: 5. Ціна: 21
Був доданий новий предмет. Вага: 2. Ціна: 25
Був доданий новий предмет. Вага: 4. Ціна: 21
Був доданий новий предмет. Вага: 9. Ціна: 29
Був доданий новий предмет. Вага: 12. Ціна: 27
Був доданий новий предмет. Вага: 10. Ціна: 16
Був доданий новий предмет. Вага: 17. Ціна: 18
Був доданий новий предмет. Вага: 6. Ціна: 20
Був доданий новий предмет. Вага: 1. Ціна: 13
Був доданий новий предмет. Вага: 14. Ціна: 21

```

Максимальна ціна за допомогою бджолиного алгоритму: 1226

Рисунок 3.3 та 3.4 – Приклад роботи алгоритму при стандартних початкових значеннях згідно до варіанту та кількості розвідників – 10, кількість фуражирів – 50, кількість ділянок – 20

3.3 Тестування алгоритму

За умови, що кількість фуражирів дорівнює 50.

Кількість ділянок	Кількість розвідників	Цінність предметів
15	10	1147
20	20	1165
25	30	1424
30	40	1398
35	50	1467

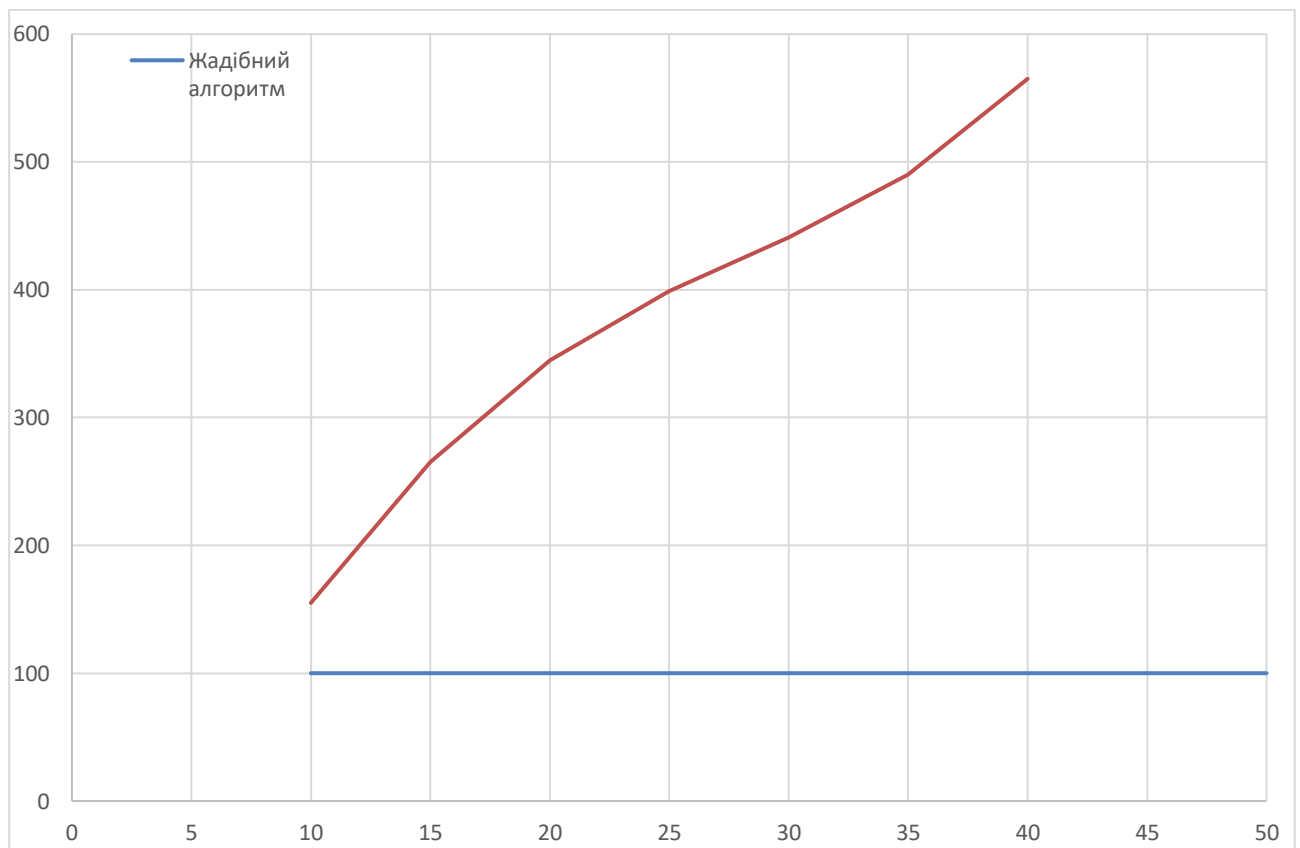


Рисунок 3.5 – Залежність цінності сформованого рюкзака від кількості розвідників (алгоритм бджолої колонії)

За умови, що кількість розвідників дорівнює 10.

Кількість ділянок	Кількість фуражирів	Цінність предметів
15	10	1481
20	20	1401
25	30	1383
30	40	1331
35	50	1291

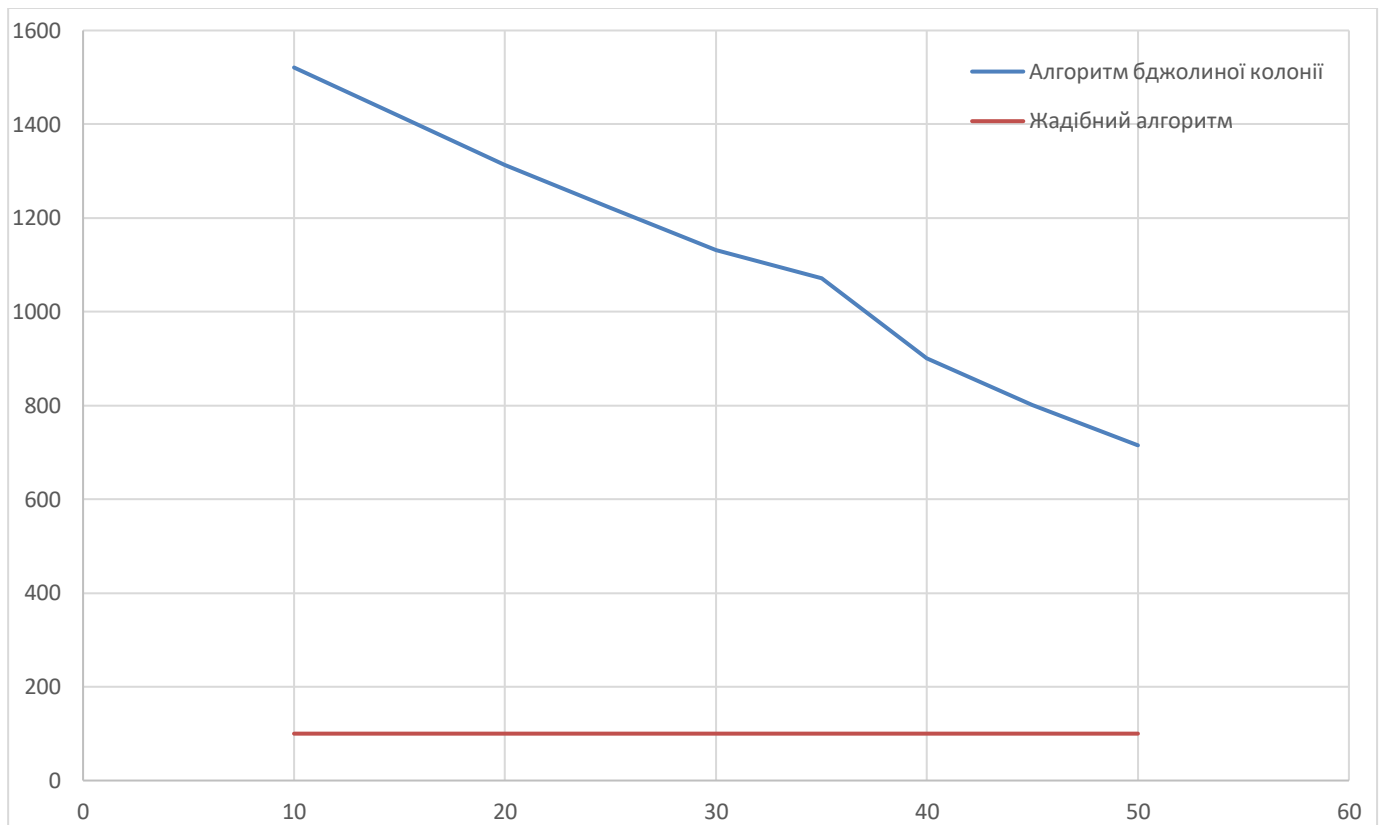


Рисунок 3.6 – Залежність цінності сформованого рюкзака від кількості розвідників (алгоритм бджолиної колонії)

ВИСНОВОК

В рамках даної лабораторної роботи був розроблений алгоритм бджолоїної колонії для задачі пошуку найбільшої цінності рюкзака. Також були проведені порівняння роботи алгоритму бджолоїної колонії та жадібного алгоритму для вирішення цієї проблеми, були наведені відповідні графіки.

КРИТЕРІЇ ОЦІНЮВАННЯ

При здачі лабораторної роботи до 11.12.2022 включно максимальний бал дорівнює – 5. Після 11.12.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- покроковий алгоритм – 15%;
- програмна реалізація алгоритму – 50%;
- тестування алгоритму – 30%;
- висновок – 5%.