Algorithm Design-II (CSE 4131)

TERM PROJECT REPORT

(March'2023-July'2023)

On

Efficient Resource Allocation in Trap-Neuter-Release Campaigns through 0/1 Knapsack Optimization

Submitted By

Sudiksha Sharma

Registration No.: 2141019487

B.Tech. 4th Semester CSE (C)

Department of Computer Science and Engineering

Institute of Techinical Education and Research

Siksha 'O' Anusandhan Deemed To Be University

Bhubaneswar, Odisha-751030

DECLARATION

I, Sudiksha Sharma, bearing registration number 2141019487 do hereby declare that this term project entitled "**Efficient Resource Allocation in Trap-Neuter-Release Campaigns through 0/1 Knapsack Optimization**" is an original project work done by me and has not been previously submitted to any university or research institution or department for the award of any degree or diploma or any other assessment to the best of my knowledge.

Sudiksha Sharma

Regd. No.: 2 1 4 1 0 1 9 4 8 7

Date: 13/06/2023

CERTIFICATE

This is to certify that the thesis entitled ”**Efficient Resource Allocation in TNR through 0/1 Knapsack Optimization**” submitted by Sudiksha Sharma, bearing registration number 2141019487 of B.Tech. 4th Semester Comp. Sc. and Engg.,ITER, SOADU is absolutely based upon his/her own work under my guidance and supervision.

The term project has reached the standard fulfilling the requirement of the course Algorithm Design 2 (CSE4131). Any help or source of information which has been available in this connection is duly acknowledged.

Satya Ranjan Das

Assistant Professor,

Department of Comp. Sc. and Engg.

ITER, Bhubaneswar 751030,

Odisha, India

Prof. (Dr.) Debahuti Mishra

Professor and Head,

Department of Comp. Sc. and Engg.

ITER, Bhubaneswar 751030,

Odisha, India

ABSTRACT

Trap-Neuter-Return (TNR) campaigns play a crucial role in managing stray dog populations and promoting community welfare. Effective resource allocation is paramount to maximize the impact of TNR initiatives. This research focuses on the integration of colony prioritization and 0/1 knapsack optimization techniques to optimize resource allocation in TNR campaigns.

The proposed approach involves prioritizing colonies based on factors such as population density, public interaction, and community input. These factors are quantified and normalized to ensure fair comparison and equitable decision-making. Leveraging the 0/1 knapsack optimization algorithm, the limited available resources are allocated efficiently by selecting the most impactful colonies for intervention.

By utilizing the 0/1 knapsack problem, the research aims to address the challenges of resource scarcity and varying colony characteristics. The objective is to maximize the overall impact of TNR campaigns by strategically selecting colonies and allocating resources for trap-neuter-return activities.

This study highlights the potential of combining colony prioritization and 0/1 knapsack optimization as a promising approach for resource allocation in TNR campaigns. The proposed model provides a framework for decision-makers to make informed choices, enhance efficiency, and allocate resources effectively to achieve the desired impact in managing stray dog populations and improving community well-being.

Keywords: Trap-Neuter-Return campaigns, resource allocation, impact maximization, colony prioritization, 0/1 knapsack optimization, efficient allocation, stray dog populations, community welfare.

CONTENTS

Serial No.	Title	
1.	INTRODUCTION	
2.	DESIGNING ALGORITHM	
3.	IMPLEMENTATION DETAILS	
4.	RESULTS AND DISCUSSIONS	
5.	LIMITATIONS	
6.	FUTURE ENHANCEMENTS	
7.	REFERENCES	

INTRODUCTION

In the realm of animal welfare, the issue of stray dogs demands our attention due to its far-reaching consequences. It is staggering to realize that a single unsterilized stray dog, over the course of its lifetime, can give rise to an astonishing 64,000 more stray dogs. This exponential growth exacerbates the challenges faced by communities worldwide.

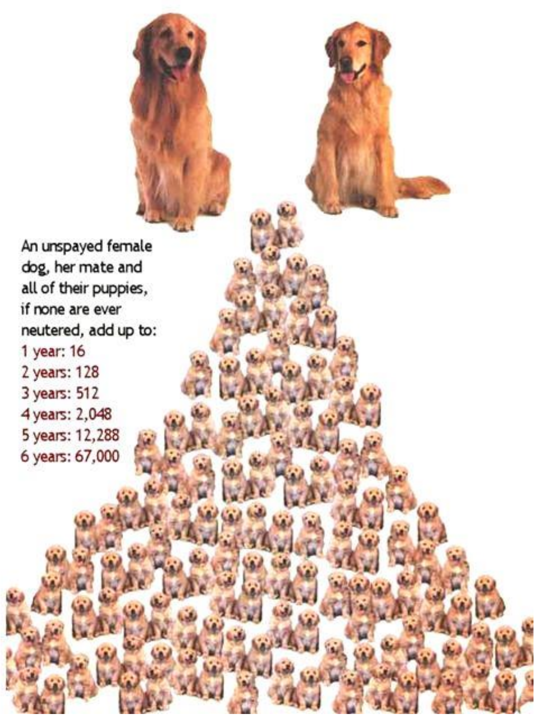
In India alone, the population of stray dogs stands at a staggering 35 million, reflecting the magnitude of the issue. Disturbingly, approximately 70% of all reported rabies cases in the world occur in India, highlighting the critical need for effective interventions.

Amidst these challenges, Trap-Neuter-Return (TNR) campaigns have emerged as the most lawful, humane, and dignified approach to addressing the stray dog crisis. Recognized by animal welfare laws and regulations, TNR campaigns provide a comprehensive solution that tackles the issue at its core. By implementing a systematic approach of trapping, neutering, and returning stray dogs to their habitats, TNR campaigns not only control population growth but also alleviate public health risks associated with rabies and other diseases.

These campaigns have achieved remarkable success stories across India. In Malappuram district of Kerala and Jaipur, for example, TNR efforts have led to a substantial reduction of almost 70% in the stray dog population. These success stories serve as testaments to the efficacy of TNR campaigns in bringing about lasting change and creating harmonious coexistence between humans and animals.

Against this backdrop, my research project delves into the intricacies of TNR campaigns, focusing on the critical aspect of resource allocation. By examining the connection between TNR campaigns and the renowned 0/1 knapsack problem, I aim to contribute to the field of animal welfare by offering insights into efficient resource distribution and enhancing the impact of TNR campaigns.

By combining empirical analysis, mathematical modeling, and practical case studies, this study seeks to shed light on the potential of TNR campaigns as a humane, sustainable, and effective solution to the challenges posed by stray dog populations. Through this work, I aspire to inspire innovative approaches, foster informed decision-making, and pave the way for a brighter future for both stray dogs and the communities they coexist with.



DESIGNING ALGORITHMS

As available resources are often limited, making informed decisions about where to allocate them becomes essential. So, in the pursuit of optimizing TNR Campaigns, Resource Allocation plays a pivotal role.

It is within this context that the connection between TNR campaigns and the renowned 0/1 knapsack problem, a classic optimization problem, emerges.

In this problem, we are given a set of items, each with a weight and a value. We are also given a capacity, which is the maximum weight that we can carry. The goal is to find a subset of items that maximizes the total value, while satisfying the capacity constraint.

By leveraging the principles of this optimization challenge, we can devise strategies for allocating resources efficiently, maximizing the impact of TNR campaigns and ultimately improving the lives of countless stray dogs.

We can map TNR campaigns to the 0/1 knapsack problem as follows:

- Items in knapsack- different colonies where TNR campaigns can be conducted
- Weight of each item- Overall cost of TNR campaign in that colony, ie. the stray dog population in the colony
- Value of each item- Social Impact value of conducting TNR campaign in the colony
- Capacity of knapsack- Total budget of the TNR Campaign, ie. the total dog population of the city

Impact Value Calculation Model For Colony Prioritisation-

I have designed a simplified yet comprehensive model that calculates the overall impact value of a colony based on various factors typically considered in TNR (Trap-Neuter-Return) campaigns:

- Population Density: Assigned a score to each colony based on its population density. A higher population density indicates a greater potential impact on population growth and community interactions. Assigned a scale from 0 to 55, with 55 indicating the highest population density.
- Public Interactions :Prioritized areas with higher public interactions, ie. resident population as higher public interactions signals more public well-being. Assigned a scale from 0 to 3, with 35 indicating the highest level of public interaction.
- Community Input: Incorporate community input by assigning a score based on the severity and frequency of reported issues, such as nuisance behavior, public health concerns, or risks to vulnerable populations. Assigned a scale from 0 to 10, with 10 indicating the highest level of concern.
-

Pseudocode-

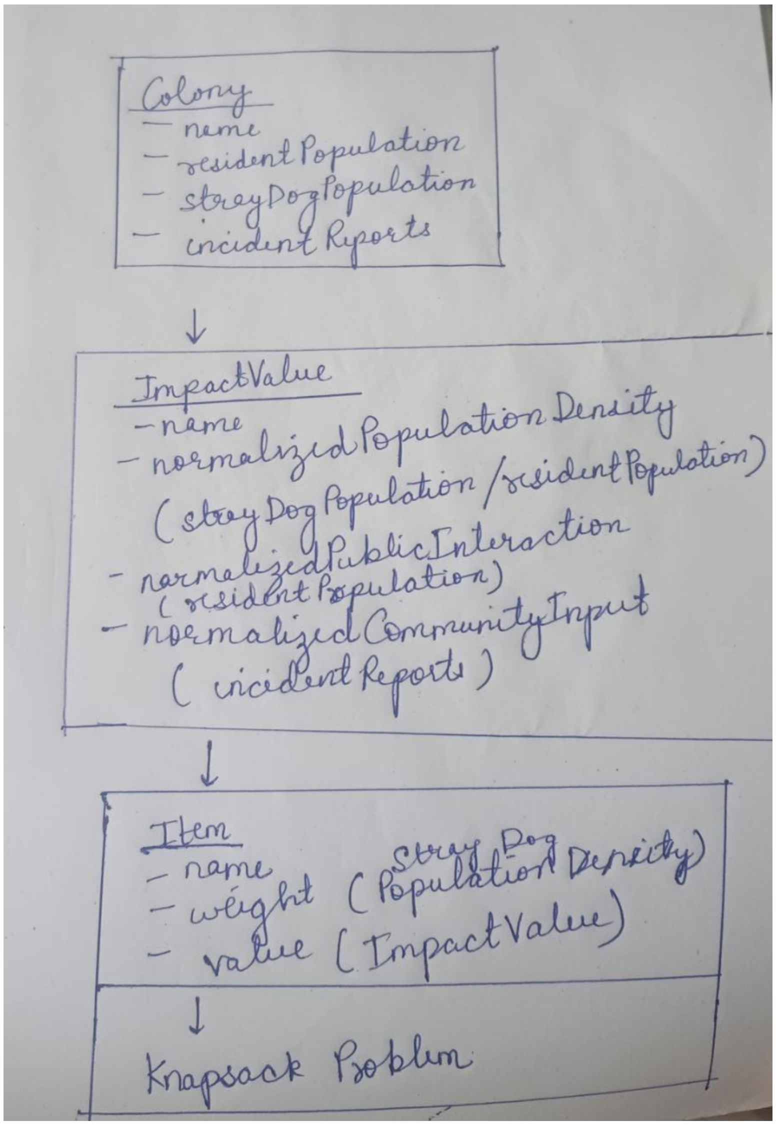
```
function TNRResourceAllocation(colonies, totalResources):

{

  // Create arrays to store the characteristics and impact values of colonies

  populationDensity = []

  publicInteraction = []

  communityInput = []

  impactValue = []


  // Calculate and normalize the characteristics of each colony

  for each colony in colonies:
```

```
      populationDensity.add(Normalize(colony.populationDensity))
      publicInteraction.add(Normalize(colony.publicInteraction))
      communityInput.add(Normalize(colony.communityInput))

   // Calculate the impact value for each colony
   for i from 0 to length(colonies) - 1:
      impactValue.add(CalculateImpactValue(populationDensity[i],
publicInteraction[i], communityInput[i]))

   // Create a 2D array to store the dynamic programming table
   dpTable = new 2D array[length(colonies) + 1][totalResources + 1]

   // Initialize the base cases
   for i from 0 to length(colonies):
      dpTable[i][0] = 0
   for j from 0 to totalResources:
      dpTable[0][j] = 0

   // Perform the knapsack dynamic programming algorithm
   for i from 1 to length(colonies):
      for j from 1 to totalResources:
         if populationDensity[i-1] <= j:
            dpTable[i][j] = max(impactValue[i-1] + dpTable[i-1][j-populationDensity[i-
1]], dpTable[i-1][j])
         else:
```

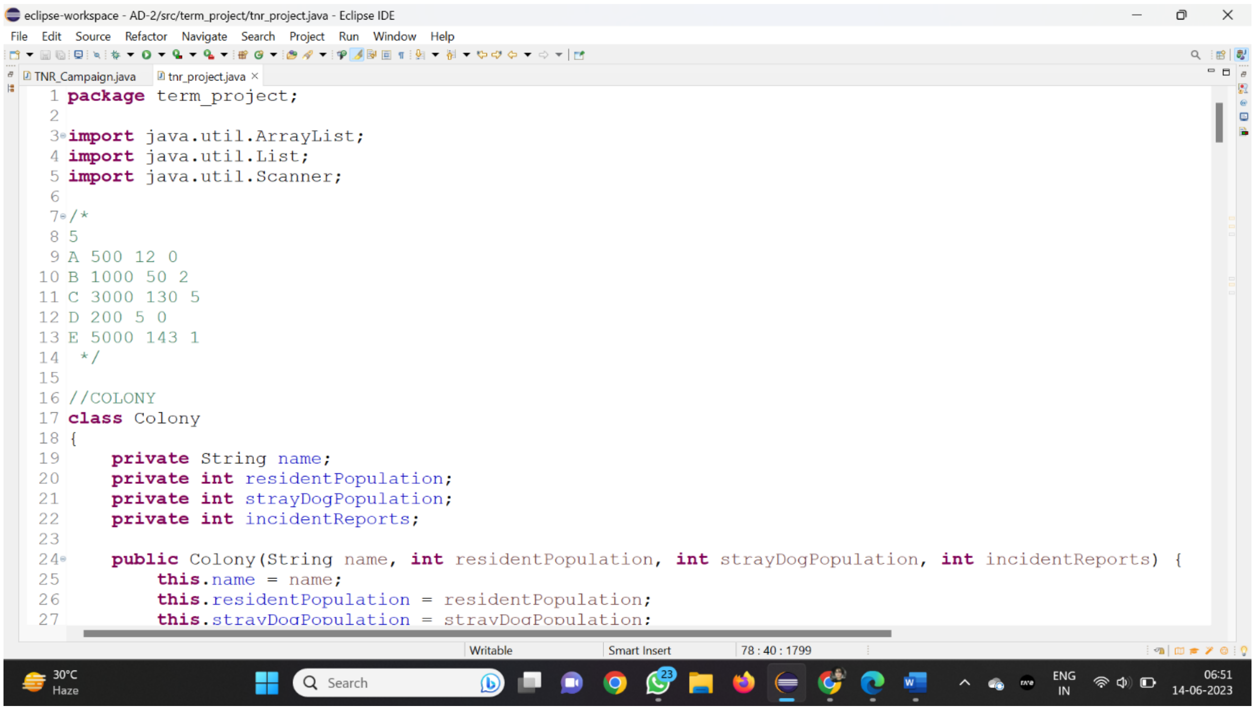
```
            dpTable[i][j] = dpTable[i-1][j]

    // Trace back the selected colonies
    selectedColonies = []
    i = length(colonies)
    j = totalResources
    while i > 0 and j > 0:
        if dpTable[i][j] != dpTable[i-1][j]:
            selectedColonies.add(colonies[i-1])
            j = j - populationDensity[i-1]
        i = i - 1

    return selectedColonies
}
```

IMPLEMENTATION DETAILS

Flowchart-



Code-



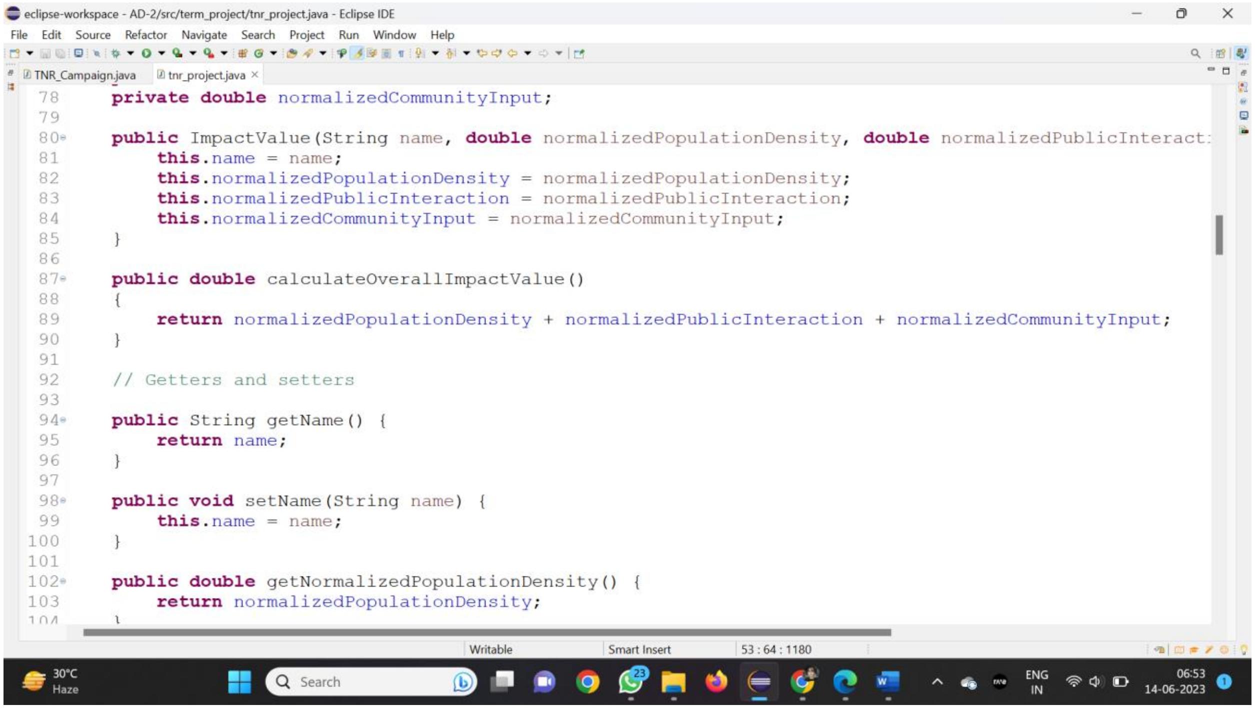

```java
package term_project;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/*
5
A 500 12 0
B 1000 50 2
C 3000 130 5
D 200 5 0
E 5000 143 1
 */

//COLONY
class Colony
{
    private String name;
    private int residentPopulation;
    private int strayDogPopulation;
    private int incidentReports;

    public Colony(String name, int residentPopulation, int strayDogPopulation, int incidentReports) {
        this.name = name;
        this.residentPopulation = residentPopulation;
        this.strayDogPopulation = strayDogPopulation;
```

eclipse-workspace - AD-2/src/term_project/tnr_project.java - Eclipse IDE

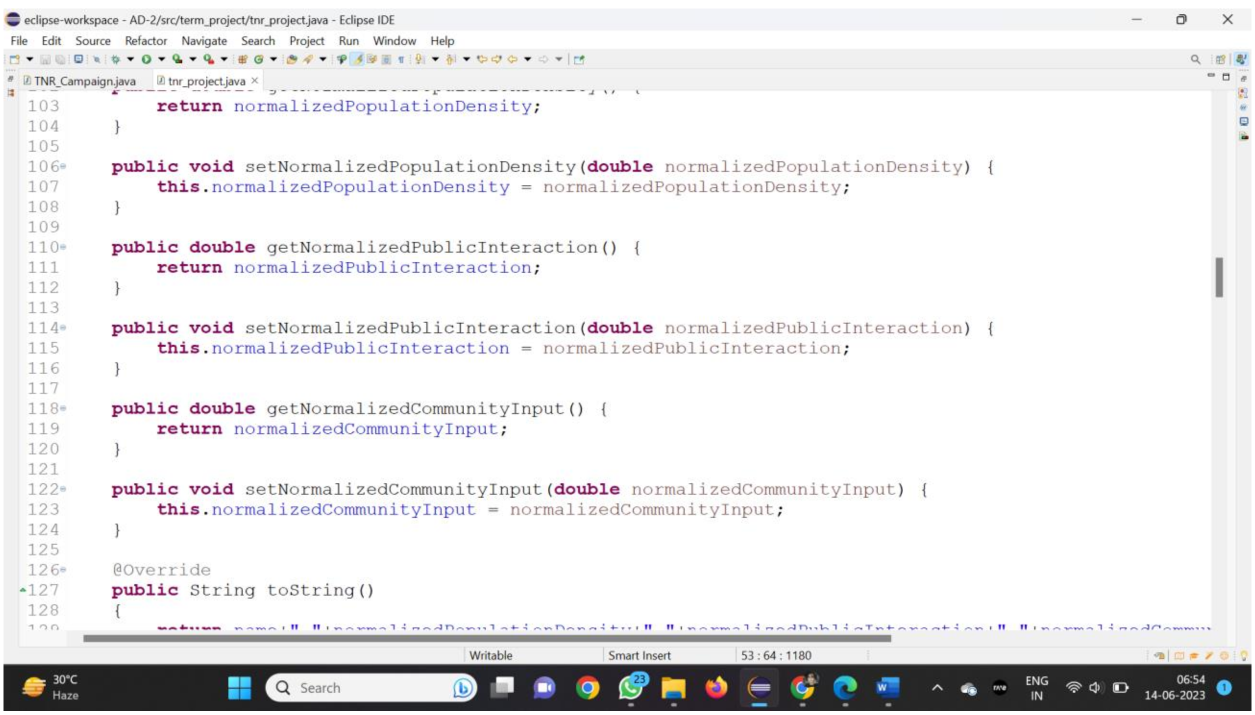
```java
        this.strayDogPopulation = strayDogPopulation;
        this.incidentReports = incidentReports;
    }

    // Getters and setters

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getResidentPopulation() {
        return residentPopulation;
    }

    public void setResidentPopulation(int residentPopulation) {
        this.residentPopulation = residentPopulation;
    }

    public int getStrayDogPopulation() {
        return strayDogPopulation;
    }

```

eclipse-workspace - AD-2/src/term_project/tnr_project.java - Eclipse IDE

```java
    public void setStrayDogPopulation(int strayDogPopulation) {
        this.strayDogPopulation = strayDogPopulation;
    }

    public int getIncidentReports() {
        return incidentReports;
    }

    public void setIncidentReports(int incidentReports) {
        this.incidentReports = incidentReports;
    }

    @Override
    public String toString()
    {
        return name+" "+residentPopulation+" "+strayDogPopulation+" "+incidentReports;
    }
}

//IMPACT VALUE
class ImpactValue
{
    private String name;
    private double normalizedPopulationDensity;
    private double normalizedPublicInteraction;
    private double normalizedCommunityInput;
```




```java
    private double normalizedCommunityInput;

    public ImpactValue(String name, double normalizedPopulationDensity, double normalizedPublicInteract
        this.name = name;
        this.normalizedPopulationDensity = normalizedPopulationDensity;
        this.normalizedPublicInteraction = normalizedPublicInteraction;
        this.normalizedCommunityInput = normalizedCommunityInput;
    }

    public double calculateOverallImpactValue()
    {
        return normalizedPopulationDensity + normalizedPublicInteraction + normalizedCommunityInput;
    }

    // Getters and setters

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getNormalizedPopulationDensity() {
        return normalizedPopulationDensity;
```




```java
        return normalizedPopulationDensity;
    }

    public void setNormalizedPopulationDensity(double normalizedPopulationDensity) {
        this.normalizedPopulationDensity = normalizedPopulationDensity;
    }

    public double getNormalizedPublicInteraction() {
        return normalizedPublicInteraction;
    }

    public void setNormalizedPublicInteraction(double normalizedPublicInteraction) {
        this.normalizedPublicInteraction = normalizedPublicInteraction;
    }

    public double getNormalizedCommunityInput() {
        return normalizedCommunityInput;
    }

    public void setNormalizedCommunityInput(double normalizedCommunityInput) {
        this.normalizedCommunityInput = normalizedCommunityInput;
    }

    @Override
    public String toString()
    {
```

eclipse-workspace - AD-2/src/term_project/tnr_project.java - Eclipse IDE

```java
        return name+" "+normalizedPopulationDensity+" "+normalizedPublicInteraction+" "+normalizedCommu
    }
}

//ITEMS
class Item
{
    private String name;
    private int weight; // Stray dog population
    private double value; // Impact value calculated from the ImpactValue class

    public Item(String name, int weight, double value) {
        this.name = name;
        this.weight = weight;
        this.value = value;
    }

    // Getters and setters

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

Writable | Smart Insert | 53 : 64 : 1180

eclipse-workspace - AD-2/src/term_project/tnr_project.java - Eclipse IDE

```java
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public double getValue() {
        return value;
    }

    public void setValue(double value) {
        this.value = value;
    }

    @Override
    public String toString()
    {
        return name+" "+weight+" "+value;
    }
}

//TESTER CLASS
```

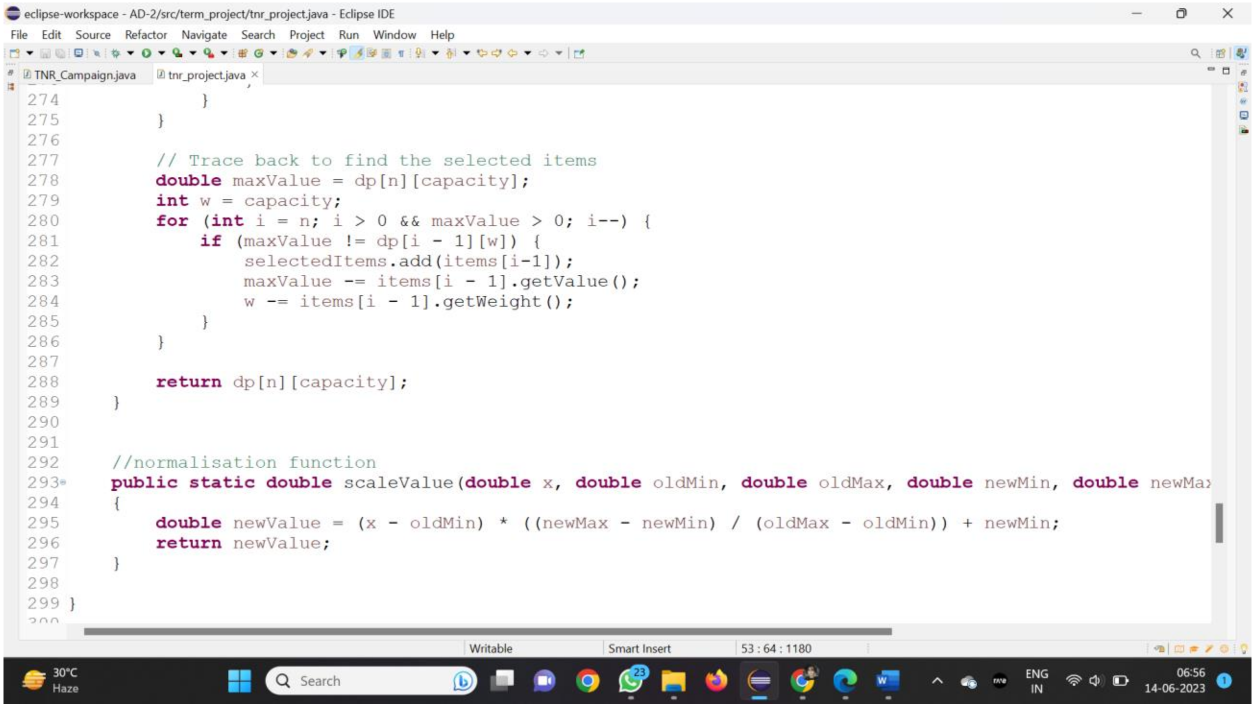
Writable | Smart Insert | 53 : 64 : 1180

```java
public class tnr_project
{
    static Scanner sc=new Scanner(System.in);

    //driver method
    public static void main(String[] args)
    {

        System.out.println("Enter no. of records: ");
        int n=sc.nextInt();

        System.out.println("Enter weightage of populationDensity, publicInteraction and communityInput
        double pdWt=sc.nextDouble(), piWt=sc.nextDouble(), ciWt=sc.nextDouble();
        double minPd=Integer.MAX_VALUE, maxPd=Integer.MIN_VALUE;
        int minPi=Integer.MAX_VALUE, maxPi=Integer.MIN_VALUE, minCi=Integer.MAX_VALUE, maxCi=Integer.MI

        Colony[] colonies=new Colony[n];

        for(int i=0;i<n;i+=1)
        {
            System.out.println("Enter Colony name, residentPopulation, strayDogPopulation, no. of incide
            String name=sc.next();
            int residentPopulation=sc.nextInt();
            int strayDogPopulation=sc.nextInt();
            int incidentReports=sc.nextInt();

```

```java
            minPd=Math.min(minPd, 1.0*strayDogPopulation/residentPopulation);
            maxPd=Math.max(maxPd, 1.0*strayDogPopulation/residentPopulation);
            minPi=Math.min(minPi, residentPopulation);
            maxPi=Math.max(maxPi, residentPopulation);
            minCi=Math.min(minCi, incidentReports);
            maxCi=Math.max(maxCi, incidentReports);

            colonies[i]=new Colony(name,residentPopulation,strayDogPopulation,incidentReports);
        }

        System.out.println("Colonies: ");
        for(Colony c: colonies)System.out.println(c);

        //create impact value array and pass normalized data into it
        ImpactValue[] impactValues=new ImpactValue[n];

        for(int i=0;i<n;i+=1)
        {
            //normalised values
            double pd=scaleValue(1.0*colonies[i].getStrayDogPopulation()/colonies[i].getResidentPopulat
            double pi=scaleValue(colonies[i].getResidentPopulation(), minPi, maxPi, 0, piWt);
            double ci=scaleValue(colonies[i].getIncidentReports(), minCi, maxCi, 0, ciWt);

            impactValues[i]=new ImpactValue(colonies[i].getName(), pd, pi, ci);

        }
```

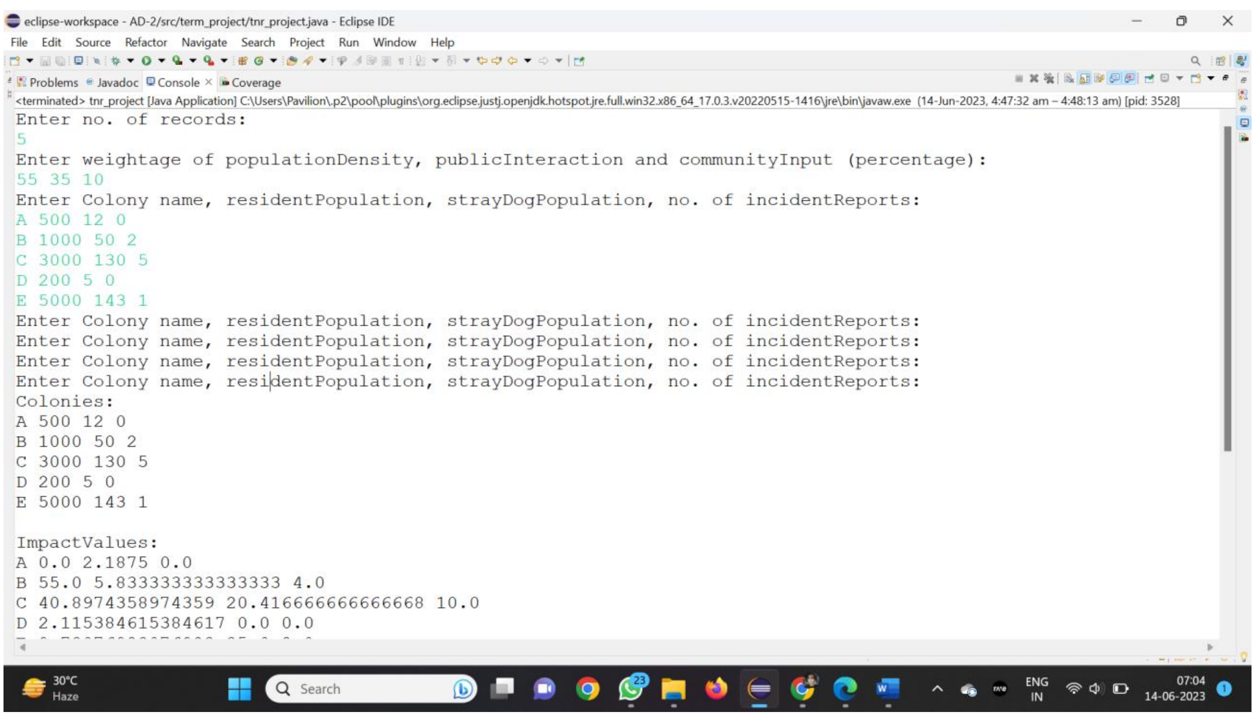
```java
        System.out.println("\nImpactValues: ");
        for(ImpactValue i: impactValues)System.out.println(i);

        //create items array
        Item[] items=new Item[n];

        for(int i=0;i<n;i+=1)
        {
            double iv=impactValues[i].calculateOverallImpactValue();
            items[i]=new Item(colonies[i].getName(), colonies[i].getStrayDogPopulation(), iv);
        }

        System.out.println("\nItems: ");
        for(Item i: items)System.out.println(i);

        //call 0/1 knapsack function
        System.out.println("Enter total budget, cost per dog: ");
        int budget=sc.nextInt(), cpd=sc.nextInt();

        List<Item> selectedItems=new ArrayList<>();
        int capacity=budget/cpd;

        System.out.println("\nMaximum impact value: "+knapsack(items,capacity,selectedItems));
        System.out.println("Selected Colonies: "+selectedItems);

    }
```

```java
    //knapsack function
    public static double knapsack(Item[] items, int capacity, List<Item> selectedItems)
    {
        int n = items.length;
        double[][] dp = new double[n + 1][capacity + 1];

        // Build the dynamic programming table
        for (int i = 1; i <= n; i++) {
            for (int w = 1; w <= capacity; w++) {
                if (items[i - 1].getWeight() <= w) {
                    dp[i][w] = Math.max(items[i - 1].getValue() + dp[i - 1][w - items[i - 1].getWeight(
                } else {
                    dp[i][w] = dp[i - 1][w];
                }
            }
        }

        // Trace back to find the selected items
        double maxValue = dp[n][capacity];
        int w = capacity;
        for (int i = n; i > 0 && maxValue > 0; i--) {
            if (maxValue != dp[i - 1][w]) {
                selectedItems.add(items[i-1]);
                maxValue -= items[i - 1].getValue();
                w -= items[i - 1].getWeight();
            }
```

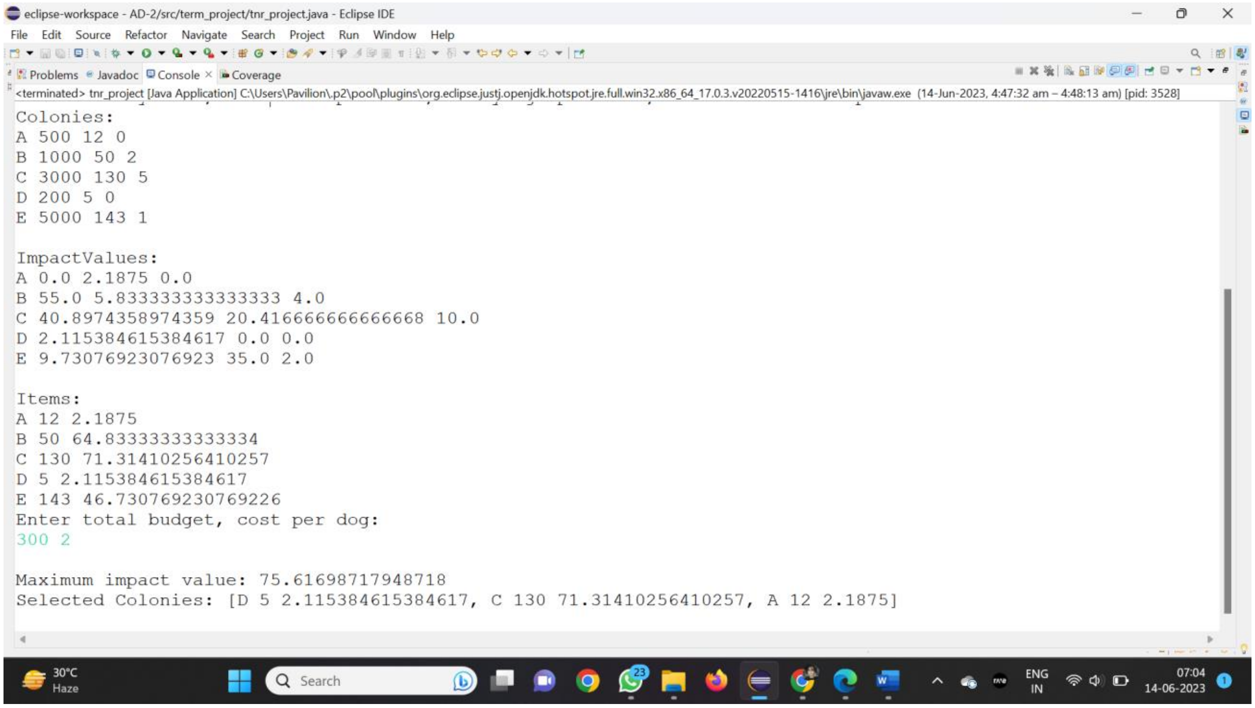


```
            }
        }

        // Trace back to find the selected items
        double maxValue = dp[n][capacity];
        int w = capacity;
        for (int i = n; i > 0 && maxValue > 0; i--) {
            if (maxValue != dp[i - 1][w]) {
                selectedItems.add(items[i-1]);
                maxValue -= items[i - 1].getValue();
                w -= items[i - 1].getWeight();
            }
        }

        return dp[n][capacity];
    }


    //normalisation function
    public static double scaleValue(double x, double oldMin, double oldMax, double newMin, double newMax
    {
        double newValue = (x - oldMin) * ((newMax - newMin) / (oldMax - oldMin)) + newMin;
        return newValue;
    }

}
```

RESULTS AND DISCUSSIONS



```
Enter no. of records:
5
Enter weightage of populationDensity, publicInteraction and communityInput (percentage):
55 35 10
Enter Colony name, residentPopulation, strayDogPopulation, no. of incidentReports:
A 500 12 0
B 1000 50 2
C 3000 130 5
D 200 5 0
E 5000 143 1
Enter Colony name, residentPopulation, strayDogPopulation, no. of incidentReports:
Enter Colony name, residentPopulation, strayDogPopulation, no. of incidentReports:
Enter Colony name, residentPopulation, strayDogPopulation, no. of incidentReports:
Enter Colony name, residentPopulation, strayDogPopulation, no. of incidentReports:
Colonies:
A 500 12 0
B 1000 50 2
C 3000 130 5
D 200 5 0
E 5000 143 1

ImpactValues:
A 0.0 2.1875 0.0
B 55.0 5.833333333333333 4.0
C 40.8974358974359 20.416666666666668 10.0
D 2.115384615384617 0.0 0.0
```



From the results, it is clear that colonies D, C and A must be prioritized first as they yield the maximum possible Social Impact Value within the limited budget.

So, by solving the 0/1 knapsack problem, we can find the optimal set of colonies where TNR campaigns should be conducted using the limited amount of budget. This will help us to allocate resources in a way that is most effective by utilizing the budget as much as possible while the Social Impact Value is maximized.

LIMITATIONS

While resource allocation for TNR campaigns using the 0/1 knapsack problem can be a useful approach, it also has some limitations. Here are a few limitations to consider:

- Limited consideration of interdependencies: The 0/1 knapsack problem treats each colony as an independent item without considering potential interdependencies among colonies. However, in TNR campaigns, the presence of neighboring colonies and their dynamics can affect the effectiveness of resource allocation.
- Lack of real-time updates: The 0/1 knapsack problem assumes that all information about colonies and factors is known in advance. In practice, data about colonies and their characteristics may change over time, requiring a mechanism for real-time updates and dynamic adjustment of resource allocation.
- Complexity of factors: The 0/1 knapsack problem simplifies factors such as population density, public interaction, and community input into single values for each colony. However, these factors can be complex and multidimensional in reality, requiring a more nuanced approach for accurate resource allocation.

FUTURE ENHANCEMENTS

There are several potential future enhancements that can be considered to improve resource allocation for TNR campaigns using the 0/1 knapsack problem. Here are a few ideas:

1. Dynamic resource allocation: Develop an algorithm that dynamically adjusts resource allocation based on changing factors such as available budget, seasonality, colony dynamics, and other relevant variables. This would allow for more adaptive and responsive resource allocation.

2. Incorporate spatial considerations: Integrate geographic information systems (GIS) data and spatial analysis techniques to incorporate spatial factors such as proximity, connectivity, and geographic barriers into the resource allocation model. This would enable more effective allocation based on the spatial distribution of colonies.

3. Consider population dynamics: Explore ways to incorporate population growth and dynamics of stray dogs into the resource allocation model. This could involve estimating future population changes based on historical data and implementing strategies to address colonies with high reproductive rates.

4. Multi-objective optimization: Extend the resource allocation model to consider multiple objectives, such as maximizing impact value while minimizing costs or considering other factors like healthcare accessibility, vaccination coverage, or local regulations. This would allow for a more comprehensive and balanced approach to resource allocation.

5. Machine learning and data analytics: Utilize machine learning and data analytics techniques to analyze large-scale data sets related to TNR campaigns, colony

characteristics, and community feedback. This could help identify patterns, correlations, and predictive models to enhance resource allocation decisions.

6. Real-time data integration: Implement mechanisms to collect and integrate real-time data from various sources, such as reported incidents, citizen feedback, or demographic changes. This would enable dynamic updates and adjustments to the resource allocation strategy based on the most up-to-date information.

These future enhancements aim to make resource allocation for TNR campaigns more dynamic, efficient, and adaptive to changing circumstances. By incorporating advanced technologies and considering additional factors, the effectiveness and impact of TNR campaigns can be further improved.

REFERENCES

1. Kleinberg, J., & Tardos, E. (2006). *Algorithm design*. Pearson Education India.

2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

3.PashudhanPraharee, *STREET DOG ISSUE : SOLUTIONS & STRAY DOG'S LEGAL RIGHTS IN INDIA*

4. Integrating Trap-Neuter-Return Campaigns Into a Social Framework: Developing Long-Term Positive Behavior Change Toward Unowned Cats in Urban Areas, Jennifer L. McDonald,1,2,* Mark J. Farnworth,3 and Jane Clements1

5. A Pilot Study to Develop an Assessment Tool for Dogs Undergoing Trap-Neuter-Release (TNR) in Italy. An Overview on the National Implementation of TNR Programmes

Greta Veronica Berteselli,1,2,* Cristina Rapagnà,1,3 Romolo Salini,1 Pietro Badagliacca,1 Fabio Bellucci,1,4 Filomena Iannino,1 and Paolo Dalla Villa1

6. Stray Dogs and Public Health: Population Estimation in Punjab, India

Gurlal S. Gill,1,2 Balbir B. Singh,1,3,* Navneet K. Dhand,3 Rabinder S. Aulakh,1 Michael P. Ward,3 and Victoria J. Brookes3