IJMRRS

International Journal for Multidisciplinary
Research, Review and Studies

ISSN: 3049-124X

Volume 1 - Issue 2

2024

Derivation and Applications of a Formula for Balancing Numbers Using Range Endpoints

By

Sakibur Rahman Utshow

Abstract

This paper presents a mathematical derivation of a formula to determine the balancing number x within a given range [A, L]. A balancing number is a unique integer in the range that equally partitions the sum of integers on both sides.

The method relies solely on the starting A and ending L numbers, eliminating the need for a balancer. This efficient approach is validated with examples and visualizations, showcasing its accuracy and potential applications in resource allocation and optimization.

Keywords: Balancing Numbers, Number Theory, Optimization, Math, Data Visualization, Diophantine Equations

Introduction

Balancing numbers is a fascinating concept in number theory. These numbers satisfy the condition where the sum of integers to their left equals the sum of integers to their right within a specific range. Traditional methods to identify balancing numbers often involve a balancer.

This paper derives a formula to compute balancing numbers directly using only the starting number A and ending number L of the range. This approach not only simplifies the computation

but also is more productive, enabling its application to a wide variety of mathematical and practical contexts.

Problem Statement

Given a range of integers from A to L, find the balancing number x such that:

The sum of integers from A to (x-1) equals the sum of integers from (x+1) to L.

Derivation

1. **Formula for the Sum of Integers in a Range**

   The sum of integers from p to q is given by:

$$Sum = \frac{(q - p + 1)(p + q)}{2}$$

2. **Left-Hand Side (LHS)**

   The sum of integers from A to x-1 is:

$$LHS = \frac{(x - 1 - a + 1)(a + (x - 1))}{2}$$

Simplify:

$$LHS = \frac{(x - a)(a + x - 1)}{2}$$

3. **Right-Hand Side (RHS)**

   The sum of integers from x+1 to L is:

$$RHS = \frac{(l - (x + 1) + 1)((x + 1) + l)}{2}$$

Simplify:

$$RHS = \frac{(l-x)(x+1+l)}{2}$$

4. **Equating LHS and RHS**

$$\frac{(x-a)(a+x-1)}{2} = \frac{(l-x)(x+1+l)}{2}$$

Multiply both sides by 2:

$$(x-a)(a+x-1) = (l-x)(x+1+l)$$

5. **Expanding Both Sides**

Expand the left-hand side:

$$(x-a)(a+x-1) = xa + x^2 - x - a^2 - ax + a$$

Expand the right-hand side:

$$(l-x)(x+1+l) = lx + l + l^2 - x^2 - x - xl$$

6. **Combine Like Terms**

Rearranging the terms, we get:

$$2x^2 = a^2 - a + l + l^2$$

7. **Solve for x**

Divide by 2 and take the square root:

$$x = \sqrt{\frac{a(a-1) + l(l+1)}{2}}$$

Example

Find the balancing number for the range [2,15]:

Using the formula:

$$x = \sqrt{\frac{2(2-1)+15(15+1)}{2}} = \sqrt{\frac{2+15\cdot 16}{2}} = \sqrt{121} = 11$$

The balancing number is 11.

Find the balancing number for the range [1,8]:

Using the formula:

$$x = \sqrt{\frac{1\cdot 0+8\cdot(8+1)}{2}} = 6$$

The balancing number is 6.

When dealing with non-integer balancing numbers, the idea remains the same as with integer cases. For example, consider the interval from 5 to 15. Here, the balancing number is approximately 11.40175425.

$$\sqrt{\frac{5(5-1)+15(15+1)}{2}} = \sqrt{\frac{20+240}{2}} = \sqrt{130} = 11.40175425$$

This means that if you sum all the numbers from 5 up to (but not including) 11.40175425,

$$\sum_{i=5}^{11.40175424} i \approx 121.401754$$

and then sum all the numbers from just above 11.40175425 up to 15, both sums turn out to be equal—approximately 121.401754

$$\sum_{i=11.40175426}^{15} i \approx 121.401754$$

The calculation is precise up to five decimal places.Same goes for 10.40175425 and 12.40175425 If (n - 1) is considered.

Methodology

A Python program Is created where It finds A and L numbers which satisfy the above rules for finding out the range for A and L.

```
import math

def is_square(n):
    """
    Checks if a number is a perfect square.

    Args:
        n: The number to check.

    Returns:
        True if n is a perfect square, False otherwise.
```

```
    """
    root = math.sqrt(n)
    return root == int(root)

def find_a_and_l():
    """
    Finds values for a and l that satisfy the given conditions.

    Returns:
        A list of tuples containing the values of a and l.
    """
    results = []
    for a in range(1, 2):  # change range
        for l in range(a + 3, 10000000000):  # Ensure l > a and l - a >= 3
            expression_value = (a * (a - 1) + l * (l + 1)) // 2
            if is_square(expression_value):
                results.append((a, l))
    return results

if __name__ == "__main__":
    results = find_a_and_l()
    if results:
        print(f"Found {len(results)} pairs of a and l:")
        for a, l in results:
            print(f"a = {a}, l = {l}")
    else:
        print("No values of a and l found that satisfy the conditions.")
```

The above code gives values of L up to 10,000,000,000 which fulfills all the conditions for Balancing Numbers from which the values can be plugged [those values of L] into the equation for balancing numbers and It will get the balancing numbers for that specific L range. From which a simple python code can be created where It will just plug in those L numbers and get me the solution as A values dont matter for ranges starting from 1.

By doing all this It found 5681 L numbers. Plugging in the L numbers In the equation will give 5681 balancing numbers(Except 1). From those 5681 It will make a line chart based on those balancing numbers the code for this is given below:

```
import math
import matplotlib.pyplot as plt

def calculate_x():
  """
  Calculates the value of 'x' for given 'l' values and plots the results.

  Args:
    None (data is hardcoded within the function)

  Returns:
    None (the function creates and displays the plot)
  """

  # Input data: copy-paste the values here
  input_data = """
a = 1, l = 8
a = 1, l = 49
a = 1, l = 288
a = 1, l = 1681
a = 1, l = 9800
a = 1, l = 57121
a = 1, l = 332928
Did Not paste all the data as there are more than 5680 data in this input data
  """

  # Split the pasted data into lines
  lines = input_data.strip().split("\n")

  # Extract 'l' values and calculate 'x'
```

```
  l_values = []
  x_values = []
  for line in lines:
    l = int(line.split(', l =')[1].strip())
    x = math.sqrt((l * (l + 1)) / 2)
    l_values.append(l)
    x_values.append(x)

  # Create the plot
  plt.figure(figsize=(10, 6))
  plt.plot(range(1, len(l_values) + 1), l_values, label='l', color='blue')
  plt.plot(range(1, len(l_values) + 1), x_values, label='x', color='red')
  plt.xlabel('Set # (nth Balancing Number)')
  plt.ylabel('Values')
  plt.title('l Value and Balancing Number')
  plt.legend()
  plt.grid(True)
  plt.show()

# Run the function
calculate_x()
```

From these two python programs the Balancing Number Graphs and Scatter Plots can be created.

Charts & Scatter Plots

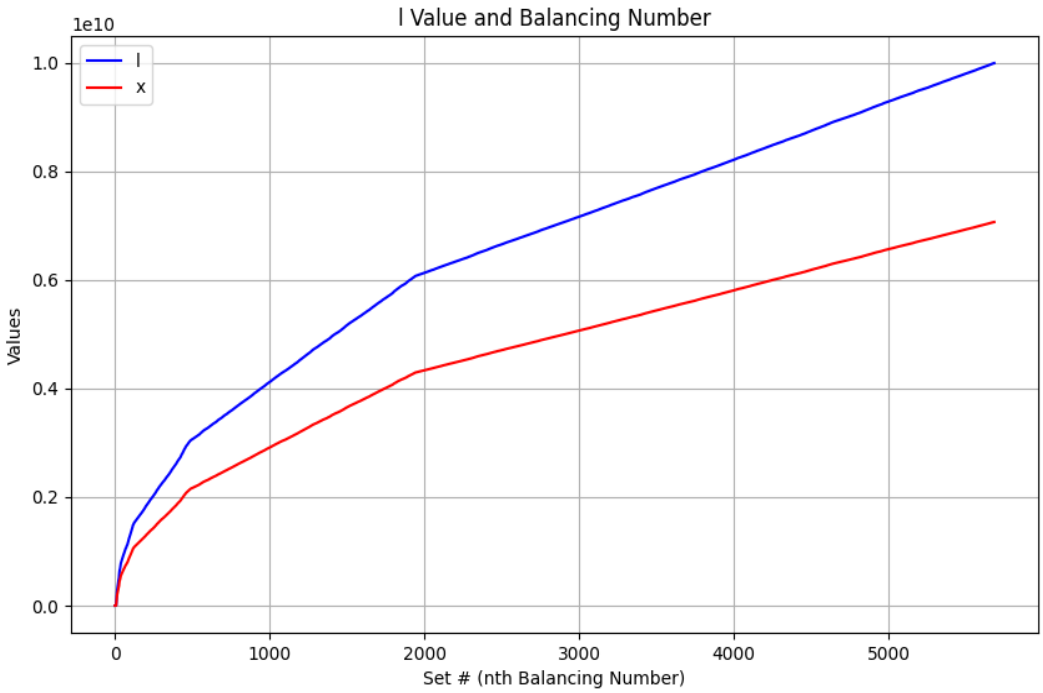


Fig [1]

Fig [1] shows the line chart for the 5681th balancing number in the x axis and shows that the L value slope is steeper than that of balancing numbers value slopes.

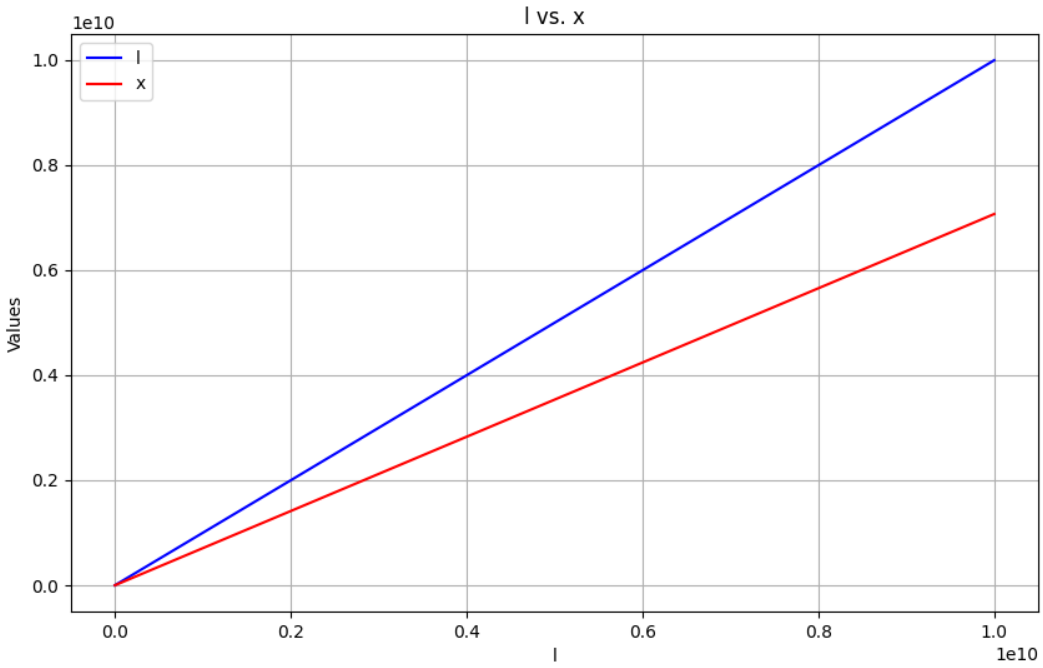


Fig [2]

Fig [2] shows that as the value of L increases, both the balancing number and the Ls value increase; however, the Ls value increases at a steeper rate compared to the balancing number.

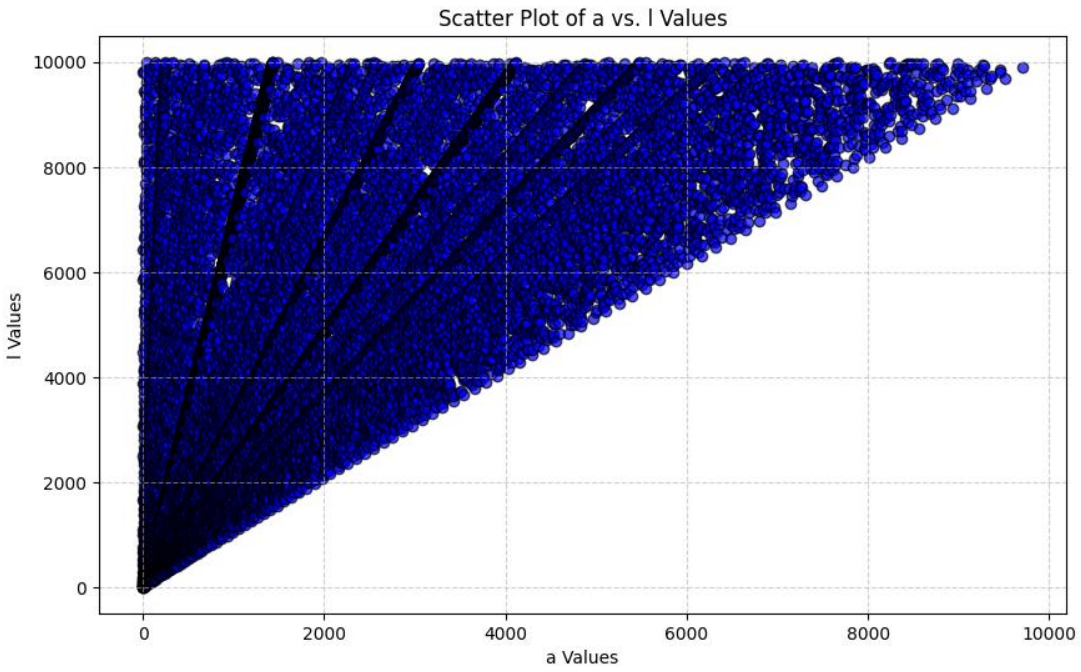


Fig [3]

Fig [3] is a scatter plot illustrating the relationship between A and L. For each value of A, there are multiple corresponding L values; therefore, a scatter plot is used to visualize data for a range where both A and L extend up to 10,000.

1. Triangular Distribution:

The overall shape of the scatter plot resembles a filled triangle. The density of points is higher near the origin and decreases as you move towards the upper-right corner.

This pattern suggests a relationship where the increase in "A Values" correlates with a more dispersed set of "L Values".

2. Distinct Linear Bands:

Similar to the previous plot, several diagonal bands are visible, each representing distinct linear relationships. The bands become denser towards the left (lower "A Values") and sparser as "A Values" increase.

This could indicate the presence of different linear trends or formulaic relationships within subsets of the data.

3. Dense Lower Bound:

The bottom edge of the triangle, close to the x-axis, is very dense, suggesting that for many values of "A," there are corresponding "L Values" close to zero or increasing gradually.

This could indicate that lower values of "A" often correspond to lower "L Values".

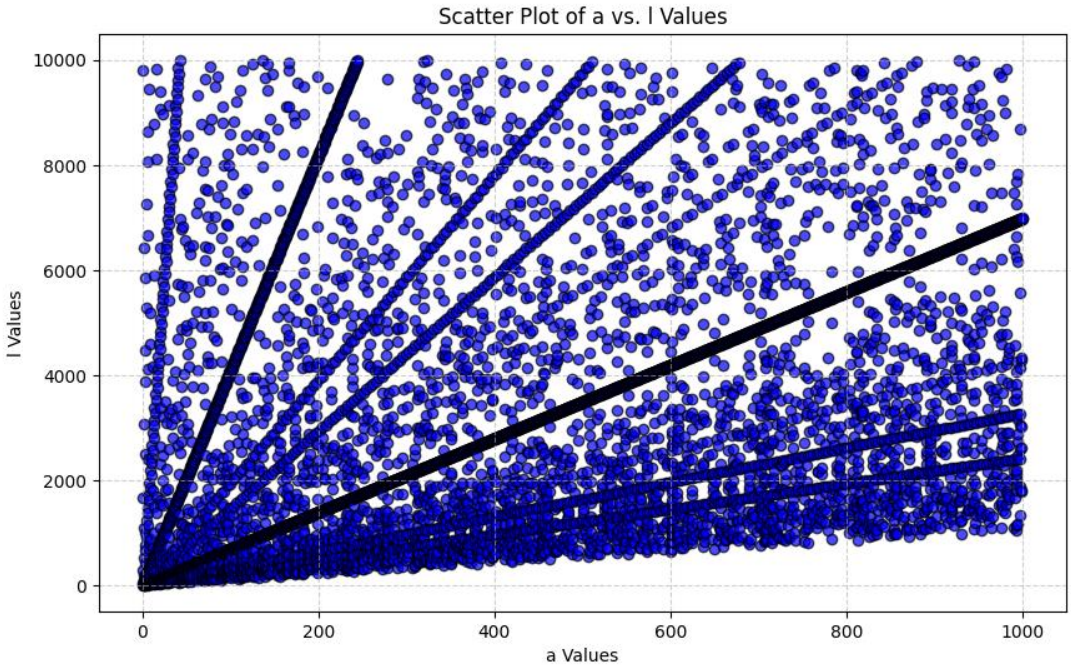


Fig [4]

Fig [4] scatter plot doesn't show a right triangle shape as the L range is greater than a range.

The x-axis ("A Values") ranges from 0 to 1000, and the y-axis ("L Values") ranges from 0 to 10000. This shows a broad range of data, suggesting varied values for both variables

The scatter plot shows the relationship between two variables, labeled as "A Values" on the x-axis and "L Values" on the y-axis. Here are some key observations:

Clustered Points with Diagonal Patterns:

The plot exhibits several distinct diagonal patterns, each representing a linear relationship between the two variables. These lines suggest that there might be multiple linear relationships or groups within the data.

Increasing Trend:

In general, as "A Values" increase, "L Values" also tend to increase. This indicates a positive correlation between the two variables.

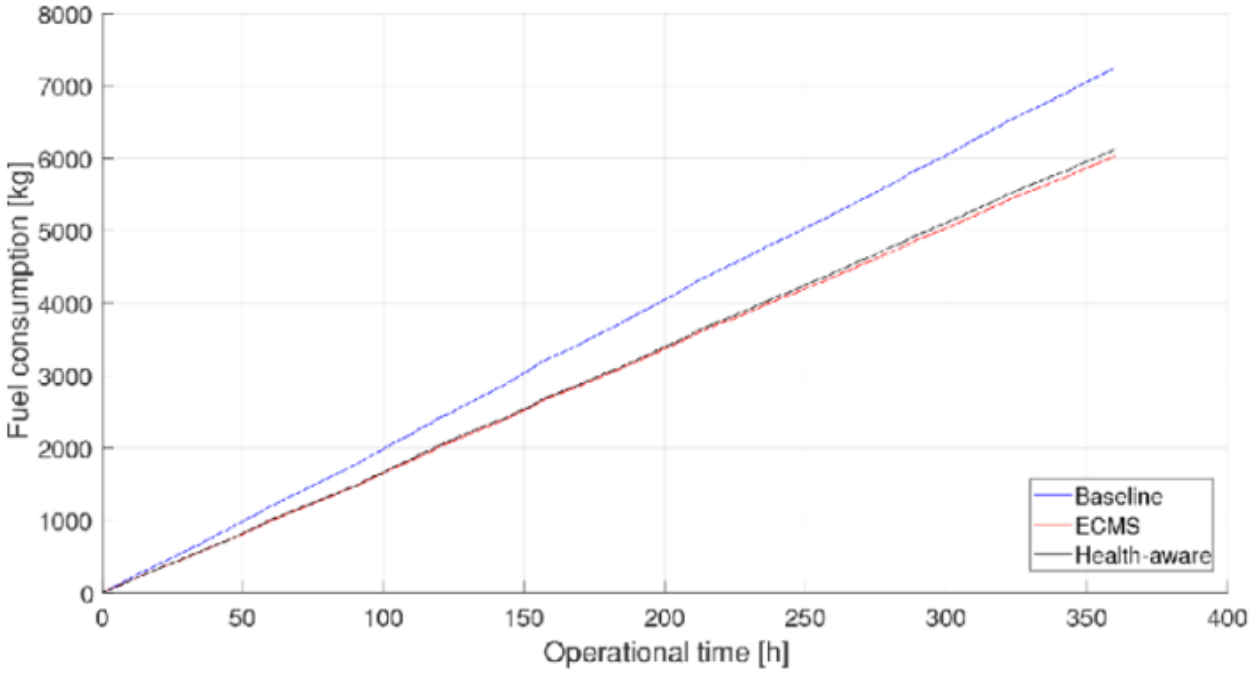
Spread of Data:

The data points are scattered but follow clear lines, which may imply discrete sets of relationships or categorizations within the dataset.

Dense Concentration Along Lines:

There are several thick, bold lines where data points are densely packed, indicating that certain specific relationships or values of "A" and "L" are more common or occur frequently.

Applications of Relation Between L And Balancing Number:

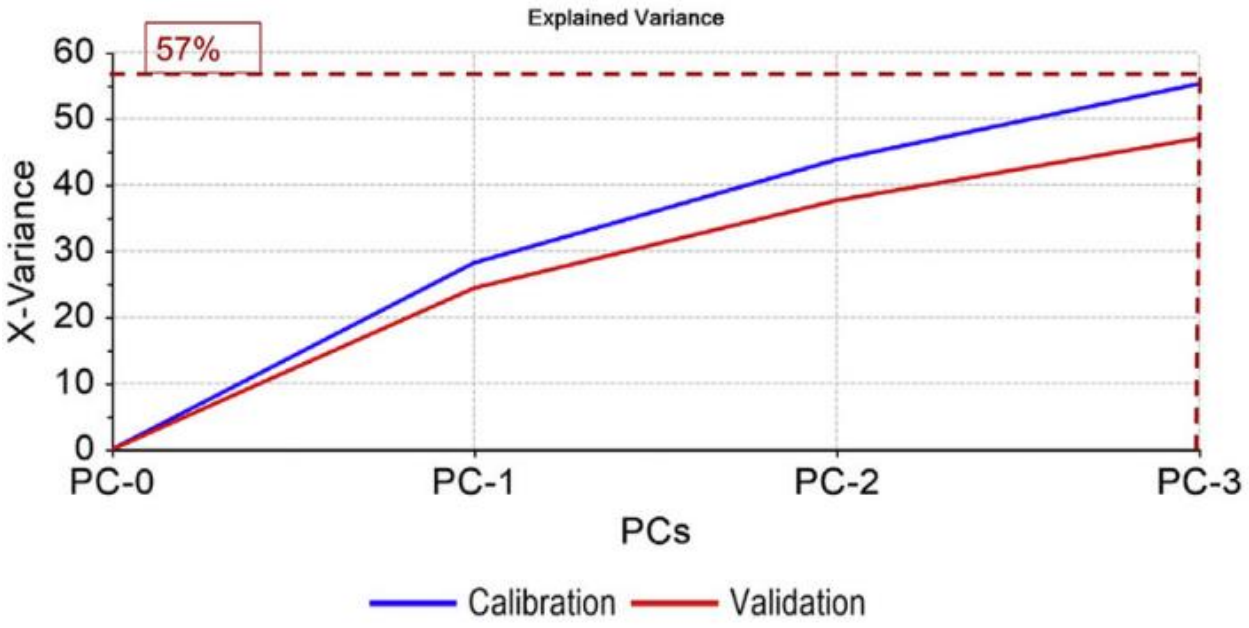


Total (cumulative) fuel consumption comparison results.

Fig [5]

- **Predictive Maintenance:**
    - Just as the fuel consumption graph illustrates the accumulation of fuel usage over time, our graph can be used to model the accumulation of "wear and tear" in a machine.
    - By representing the number of operating hours as 'L' and the balancing number as the accumulated wear and tear, we can predict when the machine might require maintenance or replacement. This proactive approach can help prevent costly breakdowns and ensure optimal equipment performance.

- **Resource Management:**
    - Similar to how the fuel consumption graph tracks resource usage (fuel), our graph can model the consumption of other resources such as water or energy.
    - By representing time as 'L' and the balancing number as the resource consumption, we can predict future usage patterns.
    - This information is valuable for resource planning, conservation efforts, and optimizing resource allocation.



Cumulative explained variance versus the number of principle components.

Fig [6]

Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset while preserving the most important information. It achieves this by identifying linear combinations of the original variables, known as principal components, that capture the maximum variance in the data.

While this Papers graph directly depicts the relationship between 'L' values and balancing numbers, its overall shape and the increasing trend of both variables bear some resemblance to

the outcomes of a PCA analysis. In PCA, we observe how much of the total variance in the data can be explained by each successive principal component. Similarly, our graph shows how both 'L' values and balancing numbers increase, suggesting a growing cumulative effect or variance.

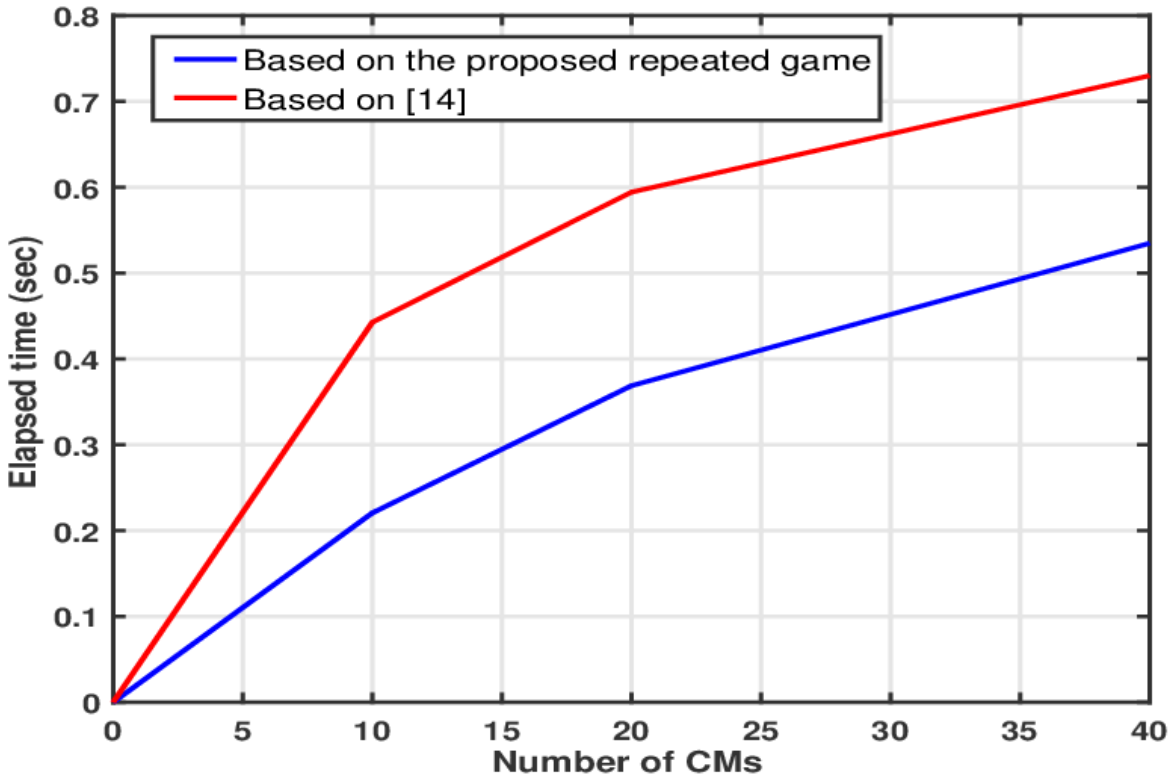


Fig [7]

"A Game-Theoretic Approach for Enhancing Security and Data Trustworthiness in IoT Applications". The paper investigates security and data trustworthiness challenges in Internet of Things (IoT) networks. It employs game theory to model interactions among devices and identify strategies for mitigating threats, such as malicious attacks and data manipulation. By analyzing

these interactions, the paper aims to enhance the security and reliability of data exchange within IoT systems.

Applications of the Graph:

- **Resource Allocation in IoT Networks:** The graph demonstrates a pattern of exponential growth, reminiscent of resource consumption in IoT networks. As the number of devices ('L' values) in a network increases, the demand for resources such as bandwidth and energy (represented by the balancing number) also rises significantly. This visual representation highlights the critical need for efficient resource allocation strategies in IoT systems to ensure optimal performance and avoid bottlenecks.

- **Security and Trust in IoT Systems:** The graph can be interpreted as a model for the increasing security risks associated with growing IoT networks. As the number of devices ('L' values) in the network expands, the complexity of managing security and ensuring data trustworthiness (represented by the balancing number) increases exponentially. This visualization underscores the importance of robust security measures and proactive threat mitigation strategies in large-scale IoT deployments.

- **Data Trustworthiness and Management:** In the context of IoT, the graph can represent the relationship between the number of data sources ('L' values) and the overall level of data trustworthiness (represented by the balancing number). As the number of data sources increases, ensuring data accuracy, reliability, and security becomes more challenging. This visualization emphasizes the need for robust data validation,

verification, and trust management mechanisms in data-driven IoT applications.

These applications demonstrate how the insights gained from your graph can be valuable for understanding and addressing critical challenges in the development and deployment of IoT systems, including resource management, security, and data trustworthiness.

Applications for Higher Productive Starting Points:

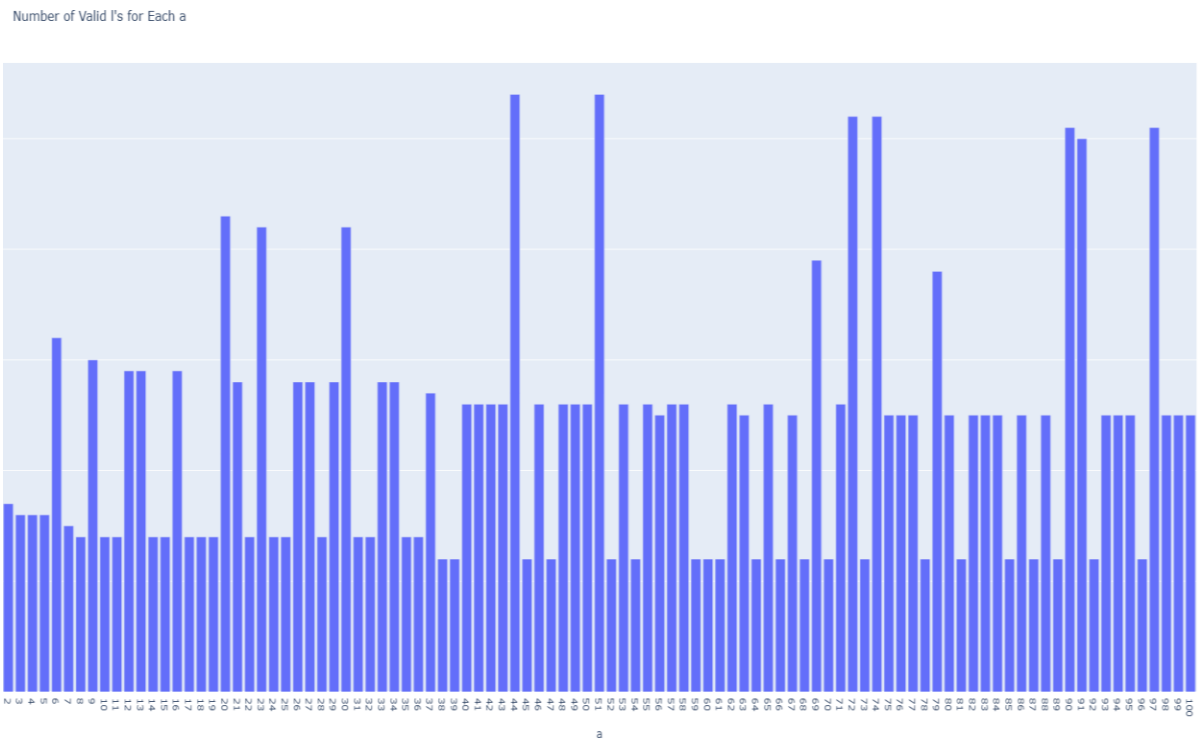


Fig [8]

Fig [8] illustrates how different starting points can have more balancing numbers than the starting point of L within a specific range of [A + 3, 1000000]. Fig [8] also demonstrates that

starting points such as 44 and 51 yield 54 balancing numbers, compared to only 8 balancing numbers when starting from 1—resulting in a 675% (54/8) improvement in efficiency.

If the range for the starting number is increased from 1 to 1000 and the ending number has a range of [A + 3, 100,000], the program can more efficiently search for productive starting points for balancing numbers.
for example, below is a bar chart for the same range given, as mentioned above:

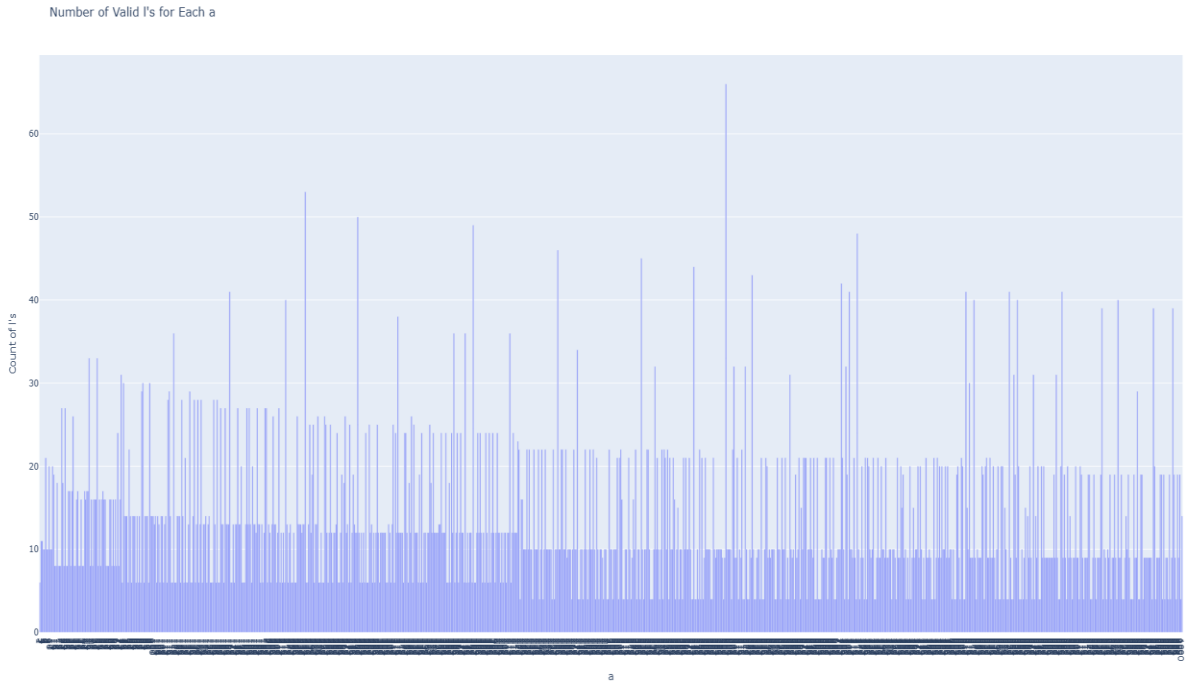


Fig [9]

Here 601 has 66 balancing numbers within a shorter range and 1 has 6 balancing numbers.this improves the efficiency to 1100% (66/6). This way the equation can find more productive starting points for finding balancing numbers within a short range. This information can be used in various fields in real life. Given below are some applications:

Resource Allocation and Distribution

In systems where resources need to be distributed evenly, identifying productive starting points can maximize efficiency.

Example: Distributing supplies from a warehouse to multiple locations. Starting distribution from a productive point (one that balances supply and demand efficiently) minimizes travel time and reduces resource wastage. For instance, if starting from warehouse #44 results in 13 balanced routes while others yield fewer, it's optimal to begin there.

Load Balancing in Networks

Productive starting points can improve load balancing in computing or communication networks by distributing tasks more evenly among nodes.

Example: In a data center with interconnected servers, starting load distribution at a productive node reduces the risk of overloading some servers while under-utilizing others. This leads to faster processing and prevents system bottlenecks.

Cryptography

Application: Balancing numbers can be utilized in cryptographic algorithms to enhance security.

Example: In the paper "Balancing and Lucas-Balancing Numbers and their Application to Cryptography," researchers explore how these numbers can be applied to develop cryptographic schemes.

How It Helps: Cryptographic schemes rely on unique, hard-to-predict keys. Having more balancing numbers from different starting points increases the available pool of secure keys or sequences. This diversity makes cryptographic systems more robust against attacks.

Coding Theory

Application: Balancing numbers are used in coding theory to construct error-detecting and error-correcting codes.

Example: The study "Coding Theory Based on Balancing Polynomials" discusses how balancing numbers contribute to the development of codes that can detect and correct errors in data transmission.

How It Helps: Error-detecting and error-correcting codes often depend on unique sequences with specific properties. More balancing numbers across various ranges can allow for more flexible and efficient code constructions.

Image Scrambling

Application: Balancing numbers are applied in digital image processing for scrambling methods to secure image data.

Example: The article "A Digital Scrambling Method Based on Balancing Numbers" presents a new approach to image scrambling using balancing transformations, enhancing the security of image data.

How It Helps: Image scrambling techniques benefit from unique and varied transformations. Using balancing numbers from different starting points introduces more ways to scramble data, making it harder for attackers to reverse-engineer the scrambling process.

Solving Diophantine Equations

Application: Balancing numbers assist in finding integer solutions to certain types of Diophantine equations.

Example: The paper "Balancing Numbers and Application" elaborates on the role of balancing numbers in solving these equations, contributing to number theory research.

How It Helps: Finding integer solutions to Diophantine equations often involves searching through specific number sets. Having balancing numbers from various starting points expands the solution space, increasing the chances of finding valid solutions.

Discussion

The research demonstrates how the derived formula can identify productive starting points to maximize the number of balancing numbers within a range, leading to improved efficiency in various fields like logistics, cryptography, and data distribution.

Key contributions:

1. **Mathematical Derivation**:
    - The paper provides a step-by-step derivation of the balancing number formula.
2. **Visualization**:
    - Graphs, including scatter plots, line charts, and bar charts, illustrate the behavior of x for different ranges of A and L.
3. **Applications**:
    - Real-world use cases include resource allocation, network optimization, and predictive maintenance, among others.

Conclusion

This paper presents an efficient formula for computing balancing numbers using only the starting and ending numbers of a range. The derived formula,

$$x = \sqrt{\frac{a(a-1) + l(l+1)}{2}}$$

eliminates balancer computations, making it a valuable tool for finding large nth balancing numbers.Showing how this equation has a way to find more productive starting numbers which

has more balancing numbers and the graph for L and balancing numbers shows some real world application for predicting future results.

References

- Tsoumpris, Charalampos, and Gerasimos Theotokatos. " A Health-Aware Energy Management Strategy for Autonomous Ships Power Plants Operation." Transportation Research Procedia (2022): https://doi.org/10.1016/j.trpro.2023.11.716.

- Parhizkar, Tarannom, Elham Rafi eipour, and Aram Parhizkar. " Principal Component Analysis (PCA): Prognostics Health Monitoring of Complex Systems using Principal Component Analysis based Feature Reduction." Journal of Cleaner Production, March 2021. https://doi.org/10.1016/j.jclepro.2020.123866.

- Abdalzaher, Mohamed S., and Osamu Muta. "A Game-Theoretic Approach for Enhancing Security and Data Trustworthiness in IoT Applications."IEEE Internet of Things Journal November 2020. https://doi.org/10.1109/jiot.2020.2996671

- Swain, Sujata, Chidananda Pratihary, and Prasanta Kumar Ray. "Balancing and Lucas-Balancing Numbers and Their Application to Cryptography." ResearchGate, 2018. https://doi.org/10.18495/comengapp.v5i1.46.

- Prasad, Bandhu. "Coding Theory Based on Balancing Polynomials." Communications and Computing, 2021. https://doi.org/10.2478/candc-2021-0017.

- Swain, S., Pratihary, C., & Ray, P. K. (2016). Balancing and Lucas-balancing numbers and their application to cryptography. *Computer Engineering and Applications*. https://doi.org/10.18495/comengapp.v5i1.46