Implementation of the Ant Colony and Firefly Optimization Algorithm on NP-Hard Bin Packing Problem (1D packing)

Abhirupa Mitra 17BCE0437, Nikhil Anand 17BCB0053

Abstract — **Bin Packing Problem is categorized as a NP - Hard Optimization Problem. This classic problem aims at reducing the no. of bins used for storing a set of items, exactly once with different weights. The algorithm of bin packing can be utilized in many real world applications such as transportation planning, container loading, resource allocation, cargo planes and ships. Traditional methods such as First-fit and Best-Fit algorithms are usually implemented to solve this problem. In this project we are going to use a bio-inspired algorithms like firefly and ant colony optimization techniques. Firefly algorithm is inspired by the flashing behavior of fireflies. The ant colony optimization is based on the behavior of ants seeking a path between their colony and source food. These algorithms are metaheuristic algorithms which aim at further reducing bin space wastage and execution time when compared with the traditional algorithms. Bio-inspired algorithmic usage has made a significant impact in many fields such as Computer Science, Electronics, Mechanical Engineering and Artificial Intelligence. The field of nature-inspired computing and optimization techniques have helped solve a diverse range of optimization problems in the field of science and technology. This project will implement the Firefly and the Ant Colony Optimization Algorithms for 1D Bin Packing.**

Index Terms— **Bin packing, ant colony optimization, firefly, meta heuristic, firefly algorithm,**

I. Introduction

Bin Packing

Bin packing have been categorized as a NP- hard combinatorial optimization problem. NP-hardness(non-deterministic polynomial-time hardness), in computational theory of an algorithm for solving it can be translated into an algorithm fit for solving any NP problem.

In classical bin packing problem, a set of items (Say, n) of different sizes or weights has to be packed into bins of a limited capacity. Optimization will require us to find out the minimum no. of bins that can be used for the packing. Given a set of bins with the same size and a list of items with sizes to pack, find an integer number of bins and a -partition of the set such that for all A solution is *optimal* if it has minimal . The B -value for an optimal solution is denoted **OPT** below. A possible Integer Linear Programming formulation of the problem is:

Bin packing algorithm is efficiently used in solving a variety of manufacturing problems such as cutting stock problems and waste minimization resulting in cost minimization.

$$\text{minimize } B = \sum_{i=1}^{n} y_i$$

$$\text{subject to } B \geq 1,$$

$$\sum_{j=1}^{n} a_j x_{ij} \leq V y_i, \forall i \in \{1,\dots,n\}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \{1,\dots,n\}$$

$$y_i \in \{0,1\}, \qquad \forall i \in \{1,\dots,n\}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \{1,\dots,n\} \, \forall j \in \{1,\dots,n\}$$

where $y_i = 1$ if bin i is used and $x_{ij} = 1$ if item j is put into bin i.

Nature is viewed as a great and immense source of inspiration for solving hard and complex problems. It reflects the tendency of biological creatures to adapt to natural conditions. It always attempts to figure out an optimal solution to solve its problem, thus maintaining a perfect balance among its components. Nature inspired algorithms are meta-heuristics that mimic nature to solve optimization problems opening a new era in computation. Optimization means finding the best possible or the most optimum solution. Nature inspired algorithm have the ability to describe and solve complex relationship based problems. Each phenomenon occurring in nature and the simultaneous response of a living creature to it acts as a subtle example of optimal strategy and complex interaction, adaption to constraining conditions and thus balancing the ecosystem.

Classical Bin Packing Problem Statement:

Given n items with weights w_i and bin capacity c, assign the number of bins so that total weight of items in each bin does not exceed c.

Models and Bounds:

The generalized bin packing problem can be formulated as upper bound GBPP and lower bound GBPP.

Let *I* denote the set of items that has to be place in the bins and w_i and p_i be the weight and the profit of item I ⍰ *I*. Let $I_C \subseteq I$. the subset of compulsory items and $I_{NC} = I \setminus I_C$ the subset of non-compulsory items. Let J denote the set of available bins and T be the set of bin types. For any bin j ∈ J, let σ(j) = t ∈ T be the type t of bin j. Define, for each bin type t ∈T, the minimum L_t and the maximum U_t number of bins of that type that may be selected, as well as the cost C_t and the volume W_t of the bin. Finally, denote U ≤P ∈T U_t the total number of available bins of any type. The item-to-bin accommodation rules of the GBPP are stated as follows:

• All items in I^C must be loaded
• For all used bins, the sum of the volumes of the items loaded into a bin must be less than or equal to the bin volume
• The number of bins used for each type $t \in T$ must be within the lower and upper availability limits L_t and U_t
• The total number of used bins cannot exceed the total number of available bins.

Traditional Algorithms Used to solve the Bin Packing Problem:

1. First-Fit Algorithm: This algorithm assumes that the items are arbitrarily arranged and starts putting in items into bins. Each item is assigned the the lowest indexed bin. If an item does not fit into a particular bin, then a new bin is introduced. The time complexity of this algorithm is O(n).
2. Best-Fit Algorithm: Best fit algorithm improves upon the first fit algorithm, by assigning the current item into a bin which has the minimum residual capacity. Contrary to that the worst fit selects a partially filled bin having the largest residual capacity.

II. LITERATURE REVIEW

The GBPP is a novel packing problem recently introduced by Baldietal. [2012a]. In their paper, the authors propose two models and preliminary bounds. A branch and-price method and beam search heuristics have been proposed in [Baldi et al., 2012b]. The stochastic variant of the problem has been studied by Perboli et al. [2012].. The BPP is the simplest mono-dimensional bin packing problem, introduced by Ullman [1971]. Johnson proposed some preliminary algorithms. In particular, in [Johnson, 1973a], he proposed the Next Fit (NF) algorithm, he proposed the First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD), and Best Fit Decreasing (BFD). Basing their studies on the work of Johnson, algorithms in the years. Yao [1980] presented the Refined First Fit algorithm, with performance ratio 5/3. Afterward, van Vliet [1992] increased the lower bound to 1.54014. Lee and Lee [1985] presented a family of bounded space algorithms, named Harmonic M. To the best of our knowledge the best result to date is due to Seiden [2002] who proposed the Harmonic++ algorithm with performance ratio at most 1.58889. Preliminary bounds to the BPP were proposed by Martello and Toth [1990]. New lower bounds were developed by Fekete and Schepers [2001] by means of dual feasible functions. Epsteinand van Stee [2005] studied the problem by means of resource augmentation. Kouakou et al. [2005] studied the problem with respect to the differential competitivity ratio. Finally, György et al. [2010] studied on-line Sequential Bin Packing Problem. Epstein and Levin [2012] have recently designed an asymptotic fully polynomial time approximation scheme for this problem. Seiden[2000] proposed an optimal on-line algorithm for the bounded space(i.e., the number of open bins is constant) problem. Seiden et al. [2003] proposed improved bounds but with two bin sizes only. Alves and Valério de Carvalho [2007] first proposed an improved column generation technique trying to solve the VSBPP to optimality. An on-line variant of the VSBPP was introduced by Zhang [1997] Kang and Park [2003] studied the problem assuming that the cost of the unit size of each bin does not increase as the bin size increases. The authors proposed two greedy algorithms and computed their asymptotic worst case ratio under three assumptions that the sizes of items and bins are divisible (i.e., the succeeding item (bin) exactly divides the previous item (bin)), the sizes of bins are divisible, and the sizes of bins are not divisible. The authors proved that both algorithms yield an asymptotic worst case ratio equal to 1 (i.e., the two algorithms are optimal) under assumption 1, equal to 11/9 under assumption 2, and equal to 3/2 under assumption 3. For Epstein and Levin [2008] designed an asymptotic polynomial time approximation scheme. Correia et al. [2008] proposed a formulation that explicitly includes the bin volumes occupied by the corresponding packings, together with a series of valid inequalities improving the quality of the lower bounds obtained from the linear relaxation of the proposed model. Approximation algorithms have been proposed by Haouari and Serairi [2009] and Hemmelmayr et al. [2012]. Recently, Bettinelli et al. [2010] introduced a branch-and-price algorithm for the resolution o.f a variant of the VCSBPP with the addition of filling constraints. The latest work dealing with exact methods to solve VCSBPP is by Haouari and Serairi [2011].The OKP has been studied by Iwama and Taketomi [2002], Iwama and Zhang [2007,2010].

III. ALGORITHMS

Ant Colony Optimization:

In computer science and in optimization problems, the ant colony optimization algorithm is a technique which can be used to finding good paths through graphs.

Some species of ants travel randomly searching for food, and on finding it, return to their colony leaving a pheromone trail on their way back. If other ants locate this trail, they stop wandering randomly but follow the trail retuning and reinforcing if they eventually find food. With time the pheromone starts evaporating, thus reducing its attractive strength. The greater is the time taken by an ant to travel the path and come back again, the greater amount of pheromone is evaporated. A shorter path gets marched over more frequently, hence the pheromone density is high. The phenomenon of pheromone evaporation prevent ants to converge to a locally optimal solution.

The overall result is that when an ant finds/locates a good/shorter path, more ants are likely to follow that path, thus increasing pheromone content, thus resulting in all ants to follow this path.

Framework for Ant Colony Optimization

```
WHILE termination conditions not met DO
  ScheduleActivities
    AntBasedSolutionConstruction()
    PheromoneUpdate()
    DaemonActions() {optional}
  END ScheduleActivities
ENDWHILE
```

Firefly Algorithm

Firefly algorithm, developed by Xin-She Yang in 2008, is inspired by the light attenuation over the distance and fireflies' mutual attraction, rather than by the phenomenon of the fireflies' light flashing. Algorithm considers what each firefly observes at the point of its position, when trying to move to a greater light-source, than is his own. Some of the assumptions taken in the algorithm are:

1. **All fireflies are unisexual**, so that any individual firefly will be attracted to all other fireflies;
2. **Attractiveness is proportional to their brightness**, and for any two fireflies, the less bright one will be attracted by (and thus move towards) the brighter one; however, the intensity (apparent brightness) decrease as their mutual distance increases;
3. **If there are no fireflies brighter than a given firefly, it will move randomly.**

Framework for Firefly Algorithm

```
GENERATE AN INITIAL POPULATION OR FIREFLIES
AND FORMULATE LIGHT INTENSITY
DEFINE ABSORPTION COEFFICIENT
TRAVERSE FROM 1 TO NTH FIREFLY IN A NESTED
LOOP FOR 2 LOOP VARIABLES i AND j
      CHECK FOR ATTRACTION AND MOVE FIREFLY
      ACCORDINGLY
POST PROCESS THE RESULTS AND VISUALIZATION
END
```

IV. Traditional Methods

First Fit: Arrange the items in an order and insert it. Take a new bin if the new item cannot be accommodated in the first one.

```
assignBins()
bins ← 1
i ← 0
tw ← 0;
while i < num do
      tw ← tw + items[i]
      if tw > cap do
         bins ← bins + 1
         tw ← 0
      else
         i ← i + 1
return bins
```

Best Fit: Insert the item in the bin with the minimum residual capacity

```
assignBins()
i ← 0
bins ← 0
for k ← 0 upto num do
      binlist[k] ← cap;
      while i < num do
            sortbins()
            for j ← 0 upto num do
            if items[i] <= binlist[j] do
                  binlist[j] ← items[i]
                  break
            i ← i + 1
            for i←0 upto num do
                  if binlist[i] != cap do
                        bins ← bins + 1
return bins
```

V. Ant Colony

Each ant is a simple agent with the following characteristics:

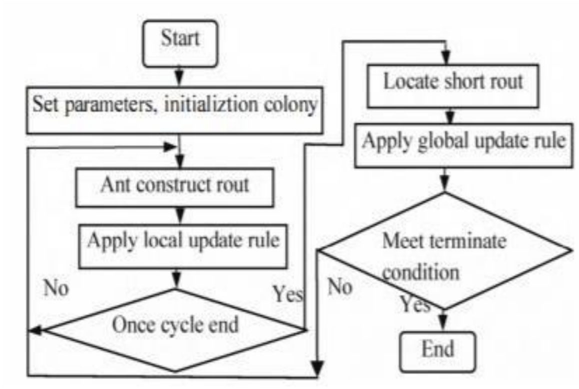
- it chooses the town to go to with a probability that is a function of the town distance and of the amount of trail present on the connecting edge;
- to force the ant to make legal tours, transitions to already visited towns are disallowed until a tour is completed (this is controlled by a tabu list);
- when it completes a tour, it lays a substance called trail on each edge (i,j) visited.

Artificial ants have several characteristics similar to real ants, namely:
1) artificial ants have a probabilistic preference for paths with a larger amount of pheromone;
2) shorter paths tend to have larger rates of growth in their amount of pheromone;
3) the ants use an indirect communication system based on the amount of pheromone deposited on each path.

Algorithm:

```
class ACO main{

public static Ant ACO()
     Matrix phmGraph
         Initialize the arraylist
         population to hold Ant objects.
         Count the number of fitness
         evaluations
         double best=Integer.MAX_VALUE;
         while(true) {
              For every ant
              Build the solution
               For every item, choose the
            bin it will go into
              temp.updateFitness(b, BPP1);
                   count++;
                   population.add(temp);
              }
          Update the pheromone on the graph
              while(pop_it.hasNext()) {

                   Retreive the ant
                   Ant temp=pop_it.next();
                   Calculate the pheromone
                   update; the amount of
                   pheromone it will deposit
                   on each edge it traversed
                   Add phmUpdate to every
                   edge traversed.
                   for(int c=0; c<items; c++)
{
  phmGraph.set(temp.getBin(c), c,
  phmGraph.get(temp.getBin(c), c) +
  phmUpdate);

                  }//for each node of ant's
  path
              }
          Evaporate pheromone on the
  construction graph
              phmGraph=phmGraph.multiply(e);
          population.sort((Ant a1, Ant a2) -
  > a1.compareTo(a2));

          if(population.get(0).getFitness()
  < best) {

  best=population.get(0).getFitness();
          }
          Clear population; ants will
  generate all new solutions next iteration
          population.clear();
          }
  }
  public static int selectNext() {
    int bins=m.getM();
    fitnessArray[0]=m.get(0, curItem);
    for (int i  = 1; i < bins; i++)
    {
        fitnessArray[i] = fitnessArray[i - 1]
           + m.get(i, curItem);
    }
        double random = Math.random() *
        fitnessArray[bins - 1];
        int binNum =
  Arrays.binarySearch(fitnessArray, random);
  if (binNum < 0)
  {
      binNum = Math.abs(binNum + 1);
  }
     Return the bin number (row index)
}
public static void main(String[] args)throws
  IOException

          Read data
          Initialize construction graph with
    a random amount of          pheromone
    (between 0 and 1)

  conGraph.randomPheromoneInitialization();
      Call ACO
      Ant result=ACO();
      Print the output ant
      result.print();
```

FLOW DIAGRAM



VI. Firefly Algorithm

Conditions:
The dimensions of the bins is such that it can handle any number of boxes until the total weight of the boxes in it does not exceed the carrying capacity.
The boxes are kept linearly from the bins.

Factors:
Attractiveness: The attractiveness here is the residual space that a box will leave in the bin once it has been kept in it. In other words, the less the weight of the box, the more the attractiveness
Distance: The distance is referred as the difference in position with the box that is attracting with the current box taken into account
Favorability: The net result obtained by subtracting the attractiveness over the distance. As the distance from the current attracting body and the body that is taken into account increases, the favorability decreases.

Algorithm:

```
assignBins(arr[])
tw ← 0
bins ← 1
max_favor ← 0
for I ← 0 upto num do
      max_favor ← 0
      tw ← tw + sortedlist[i]
      if tw > cap do
            bins ← bins + 1
            tw ← sortedlist[i]
      max_favor_pos←i
      for j ← 0 upto num do
            if arr[j]=0 or
            sortedlist[i]=arr[j] do
                  continue
            favor ←
            attractiveness(sortedlist[i],ar
            r[j]) -
            distance(sortedlist[i],j)
            if favor > max_favor do
                  max_favor ← favor
```

```
                max_favor_pos ← j
        if max_favor_pos != i do
            tw ← tw + arr[max_favor_pos]
        if tw > cap do
            bins ← bins+1
            tw ← arr[max_favor_pos]
        items[max_favor_pos]=0
return bins
```

attractiveness(x,y)

```
        return cap - x + y
```

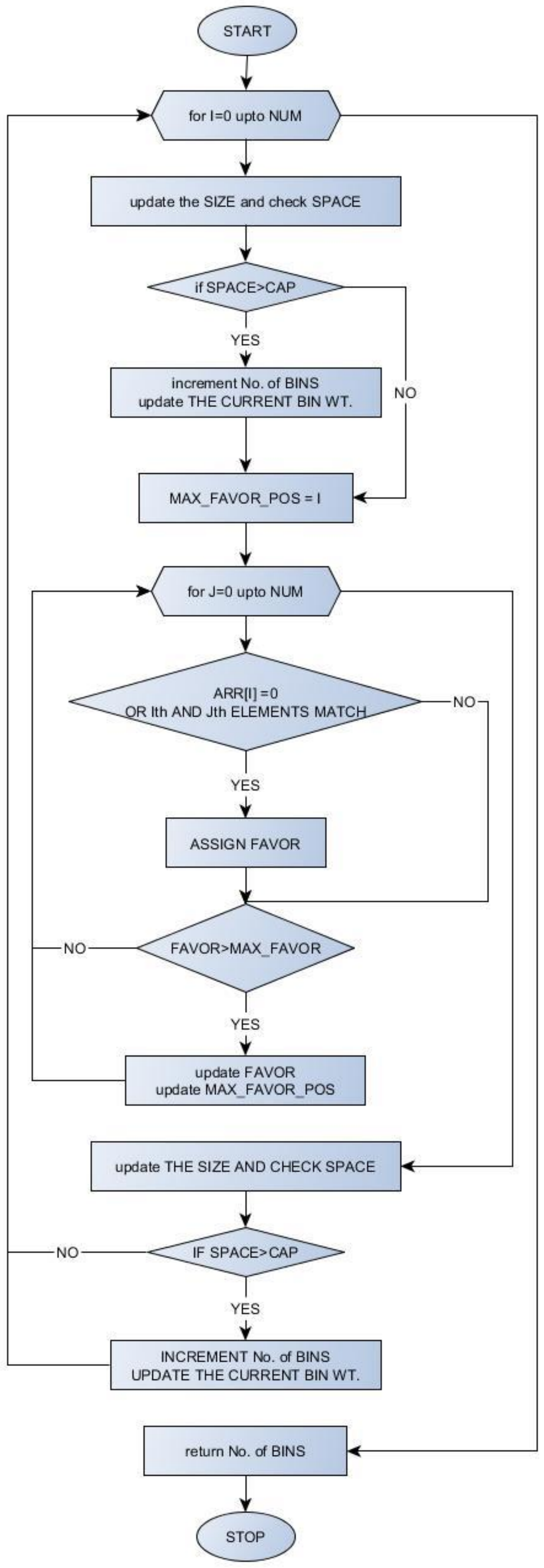
distance(a,b)

```
for i ← 0 upto num do
        if items[i] = a do
            break
return |i-b|
```

implementFirefly()

```
sortList()
max ← 100 //arbitrary number
currBin ← min ← assignBins(items)
shuffle[num]
for i ← 0 upto max-1 do
        while numbers.size() < num do
            ran ← rand.nextint(num)
            if numbers does not contain ran
        do
                add ran to numbers
        for j ← 0 upto num do
            shuffle[j] ←
items[numbers.get(i)]
        currBin ← assignBins(shuffle)
        if currBin < min do
            min ← currBin
        numbers.clear()
return min
```

Flow Diagram:



VII. OUTPUT & CONCLUSIONS

Both the code for Firefly Algorithm and Ant Colony Optimization was run for 720 different test cases and the following data was obtained.

The name of the file follow the given instruction about the data set

Parameter	**Meaning**
n	Number of items [1..4]
w_j	Weight of each item. [1..4]
C	Bin capacity [1..4]

Parameter	**Values**
N	50, 100, 150, 200
w_j	[1..100], [20..100], [40..100]
C	100,120,150

Each set is named as **N<>C<>W<>_<A…T>**

Time calculated is in milliseconds.

	FIREFLY		**ANT COLONY**	
NAME	**BINS**	**TIME**	**BINS**	**TIME**
N1C1W1_A	25	33	25	143
N1C1W1_B	31	38	31	158
N1C1W1_C	21	10	21	140
N1C1W1_D	28	6	28	116
N1C1W1_E	26	7	26	117
N1C1W1_F	27	6	27	126
N1C1W1_G	25	6	25	126
N1C1W1_H	31	6	31	136
N1C1W1_I	25	5	25	115
N1C1W1_J	26	4	26	104
N1C1W1_K	26	4	26	134
N1C1W1_L	33	4	33	114
N1C1W1_M	30	4	30	124
N1C1W1_N	26	5	26	125
N1C1W1_O	32	5	32	115
N1C1W1_P	26	5	26	135
N1C1W1_Q	28	5	28	115
N1C1W1_R	25	4	25	124
N1C1W1_S	28	5	28	125
N1C1W1_T	28	3	28	133
N1C1W2_A	29	3	29	113
N1C1W2_B	30	2	30	112
N1C1W2_C	33	3	33	103
N1C1W2_D	31	2	31	132
N1C1W2_E	36	2	36	132
N1C1W2_F	30	3	30	133
N1C1W2_G	30	3	30	113
N1C1W2_H	33	2	33	132
N1C1W2_I	35	3	35	113
N1C1W2_J	34	4	34	124
N1C1W2_K	35	4	35	124
N1C1W2_L	31	2	31	132
N1C1W2_M	30	2	30	122
N1C1W2_N	33	2	33	102
N1C1W2_O	29	2	29	132
N1C1W2_P	33	5	33	125
N1C1W2_Q	36	2	36	112
N1C1W2_R	34	2	34	122
N1C1W2_S	37	2	37	132
N1C1W2_T	38	2	38	122
N1C1W4_A	35	2	35	102
N1C1W4_B	40	2	40	132
N1C1W4_C	36	2	36	122
N1C1W4_D	38	2	38	132
N1C1W4_E	38	2	38	112
N1C1W4_F	32	2	32	122
N1C1W4_G	38	2	38	112
N1C1W4_H	40	2	40	112
N1C1W4_I	35	2	35	112
N1C1W4_J	37	2	37	112
N1C1W4_K	41	2	41	112
N1C1W4_L	35	3	35	113
N1C1W4_M	41	3	41	103
N1C1W4_N	39	3	39	133
N1C1W4_O	34	2	34	132
N1C1W4_P	38	2	38	102
N1C1W4_Q	34	2	34	122
N1C1W4_R	38	2	38	132
N1C1W4_S	36	3	36	123
N1C1W4_T	42	3	42	113
N1C2W1_A	21	3	21	113
N1C2W1_B	26	2	26	112
N1C2W1_C	23	2	23	112
N1C2W1_D	21	2	21	122
N1C2W1_E	17	3	17	103
N1C2W1_F	22	3	22	103
N1C2W1_G	21	3	21	103
N1C2W1_H	23	3	23	123
N1C2W1_I	27	3	27	133

N1C2W1_J	27	8	27	108
N1C2W1_K	24	3	24	113
N1C2W1_L	25	3	25	123
N1C2W1_M	26	3	26	113
N1C2W1_N	21	2	21	132
N1C2W1_O	15	3	15	113
N1C2W1_P	21	5	21	115
N1C2W1_Q	24	3	24	133
N1C2W1_R	23	4	23	134
N1C2W1_S	22	2	22	112
N1C2W1_T	22	2	22	112
N1C2W2_A	24	2	24	112
N1C2W2_B	27	4	27	124
N1C2W2_C	29	2	29	112
N1C2W2_D	24	4	24	114
N1C2W2_E	33	3	33	113
N1C2W2_F	26	2	26	122
N1C2W2_G	29	2	29	112
N1C2W2_H	24	2	24	112
N1C2W2_I	25	2	25	112
N1C2W2_J	25	2	25	122
N1C2W2_K	29	2	29	112
N1C2W2_L	30	2	30	122
N1C2W2_M	30	2	30	112
N1C2W2_N	26	2	26	112
N1C2W2_O	29	2	29	122
N1C2W2_P	23	3	23	123
N1C2W2_Q	30	2	30	102
N1C2W2_R	25	2	25	132
N1C2W2_S	24	2	24	102
N1C2W2_T	26	2	26	122
N1C2W4_A	30	2	30	112
N1C2W4_B	32	3	32	133
N1C2W4_C	30	3	30	123
N1C2W4_D	28	3	28	113
N1C2W4_E	30	3	30	133
N1C2W4_F	32	3	32	133
N1C2W4_G	30	3	30	123
N1C2W4_H	31	3	31	113
N1C2W4_I	35	3	35	123
N1C2W4_J	30	3	30	123
N1C2W4_K	32	2	32	122
N1C2W4_L	31	2	31	122
N1C2W4_M	31	2	31	112
N1C2W4_N	32	2	32	112
N1C2W4_O	30	2	30	122
N1C2W4_P	28	3	28	103
N1C2W4_Q	33	2	33	122
N1C2W4_R	35	2	35	132
N1C2W4_S	38	2	38	132
N1C2W4_T	29	2	29	122
N1C3W1_A	17	2	17	112
N1C3W1_B	16	2	16	102
N1C3W1_C	18	2	18	102
N1C3W1_D	19	2	19	112
N1C3W1_E	16	2	16	122
N1C3W1_F	20	2	20	112
N1C3W1_G	15	2	15	132
N1C3W1_H	19	2	19	122
N1C3W1_I	17	2	17	112
N1C3W1_J	16	2	16	132
N1C3W1_K	17	2	17	112
N1C3W1_L	17	2	17	112
N1C3W1_M	17	2	17	132
N1C3W1_N	21	2	21	132
N1C3W1_O	16	2	16	112
N1C3W1_P	19	2	19	112
N1C3W1_Q	20	2	20	132
N1C3W1_R	21	2	21	132
N1C3W1_S	16	2	16	132
N1C3W1_T	18	2	18	112
N1C3W2_A	19	2	19	132
N1C3W2_B	21	2	21	122
N1C3W2_C	22	2	22	102
N1C3W2_D	20	2	20	112
N1C3W2_E	21	2	21	112
N1C3W2_F	23	2	23	122
N1C3W2_G	23	2	23	122
N1C3W2_H	23	2	23	132
N1C3W2_I	20	2	20	132
N1C3W2_J	22	2	22	122
N1C3W2_K	22	2	22	112
N1C3W2_L	21	2	21	102
N1C3W2_M	22	2	22	122
N1C3W2_N	22	2	22	112
N1C3W2_O	21	2	21	112
N1C3W2_P	19	2	19	132
N1C3W2_Q	19	4	19	124
N1C3W2_R	20	3	20	113
N1C3W2_S	21	3	21	133

N1C3W2_T	22	4	22	124
N1C3W4_A	22	3	22	133
N1C3W4_B	23	2	23	102
N1C3W4_C	24	2	24	122
N1C3W4_D	22	2	22	122
N1C3W4_E	23	3	23	133
N1C3W4_F	22	3	22	103
N1C3W4_G	24	2	24	132
N1C3W4_H	23	2	23	112
N1C3W4_I	23	2	23	132
N1C3W4_J	23	2	23	112
N1C3W4_K	24	2	24	132
N1C3W4_L	21	2	21	102
N1C3W4_M	21	2	21	102
N1C3W4_N	21	3	21	123
N1C3W4_O	22	3	22	123
N1C3W4_P	25	2	25	132
N1C3W4_Q	25	2	25	132
N1C3W4_R	23	2	23	122
N1C3W4_S	22	2	22	112
N1C3W4_T	25	2	25	112
N2C1W1_A	48	12	48	142
N2C1W1_B	49	10	49	130
N2C1W1_C	46	10	46	130
N2C1W1_D	50	10	50	120
N2C1W1_E	58	8	58	128
N2C1W1_F	50	10	50	120
N2C1W1_G	60	12	60	122
N2C1W1_H	52	13	52	143
N2C1W1_I	62	9	62	139
N2C1W1_J	59	8	59	118
N2C1W1_K	55	8	55	118
N2C1W1_L	55	9	55	139
N2C1W1_M	46	9	46	119
N2C1W1_N	48	8	48	128
N2C1W1_O	48	9	48	119
N2C1W1_P	54	8	54	118
N2C1W1_Q	46	10	46	130
N2C1W1_R	56	9	56	129
N2C1W1_S	45	9	45	129
N2C1W1_T	52	8	52	138
N2C1W2_A	64	8	64	138
N2C1W2_B	61	8	61	128
N2C1W2_C	68	11	68	121
N2C1W2_D	74	8	74	118
N2C1W2_E	65	9	65	139
N2C1W2_F	65	8	65	128
N2C1W2_G	73	9	73	129
N2C1W2_H	70	8	70	118
N2C1W2_I	67	9	67	129
N2C1W2_J	67	8	67	128
N2C1W2_K	72	8	72	128
N2C1W2_L	62	8	62	138
N2C1W2_M	65	8	65	118
N2C1W2_N	64	8	64	138
N2C1W2_O	64	8	64	138
N2C1W2_P	68	9	68	129
N2C1W2_Q	65	8	65	138
N2C1W2_R	67	8	67	118
N2C1W2_S	66	9	66	129
N2C1W2_T	66	9	66	119
N2C1W4_A	73	12	73	142
N2C1W4_B	71	13	71	123
N2C1W4_C	77	15	77	115
N2C1W4_D	82	9	82	109
N2C1W4_E	73	11	73	131
N2C1W4_F	77	11	77	121
N2C1W4_G	71	9	71	119
N2C1W4_H	75	9	75	119
N2C1W4_I	73	8	73	108
N2C1W4_J	74	9	74	139
N2C1W4_K	70	9	70	109
N2C1W4_L	75	8	75	138
N2C1W4_M	72	9	72	119
N2C1W4_N	71	10	71	140
N2C1W4_O	80	9	80	139
N2C1W4_P	67	8	67	118
N2C1W4_Q	75	8	75	128
N2C1W4_R	70	10	70	130
N2C1W4_S	80	11	80	111
N2C1W4_T	70	8	70	128
N2C2W1_A	42	8	42	138
N2C2W1_B	50	9	50	109
N2C2W1_C	40	11	40	131
N2C2W1_D	42	8	42	138
N2C2W1_E	40	9	40	119
N2C2W1_F	49	11	49	141
N2C2W1_G	45	8	45	138
N2C2W1_H	46	8	46	118
N2C2W1_I	45	9	45	119

N2C2W1_J	42	8	42	128
N2C2W1_K	41	8	41	118
N2C2W1_L	49	8	49	138
N2C2W1_M	44	8	44	128
N2C2W1_N	43	8	43	128
N2C2W1_O	50	9	50	109
N2C2W1_P	46	8	46	118
N2C2W1_Q	49	8	49	118
N2C2W1_R	41	8	41	138
N2C2W1_S	44	8	44	138
N2C2W1_T	39	8	39	128
N2C2W2_A	52	9	52	119
N2C2W2_B	56	8	56	118
N2C2W2_C	53	8	53	128
N2C2W2_D	51	11	51	131
N2C2W2_E	54	12	54	122
N2C2W2_F	48	11	48	121
N2C2W2_G	53	8	53	128
N2C2W2_H	53	8	53	138
N2C2W2_I	49	9	49	119
N2C2W2_J	56	8	56	118
N2C2W2_K	50	8	50	138
N2C2W2_L	52	8	52	118
N2C2W2_M	54	8	54	138
N2C2W2_N	51	9	51	129
N2C2W2_O	51	8	51	138
N2C2W2_P	50	12	50	132
N2C2W2_Q	54	8	54	138
N2C2W2_R	51	10	51	140
N2C2W2_S	58	12	58	122
N2C2W2_T	56	8	56	128
N2C2W4_A	57	9	57	119
N2C2W4_B	60	12	60	122
N2C2W4_C	65	11	65	131
N2C2W4_D	61	10	61	130
N2C2W4_E	60	10	60	110
N2C2W4_F	57	8	57	108
N2C2W4_G	61	8	61	118
N2C2W4_H	61	8	61	108
N2C2W4_I	58	9	58	129
N2C2W4_J	60	8	60	118
N2C2W4_K	59	8	59	118
N2C2W4_L	57	10	57	120
N2C2W4_M	60	9	60	129
N2C2W4_N	63	9	63	119
N2C2W4_O	62	11	62	111
N2C2W4_P	60	10	60	140
N2C2W4_Q	62	13	62	143
N2C2W4_R	57	14	57	134
N2C2W4_S	55	14	55	134
N2C2W4_T	57	14	57	114
N2C3W1_A	35	14	35	144
N2C3W1_B	35	12	35	122
N2C3W1_C	35	12	35	142
N2C3W1_D	37	12	37	132
N2C3W1_E	34	12	34	122
N2C3W1_F	35	13	35	143
N2C3W1_G	33	10	33	140
N2C3W1_H	35	12	35	142
N2C3W1_I	34	8	34	138
N2C3W1_J	33	8	33	108
N2C3W1_K	36	11	36	131
N2C3W1_L	35	8	35	138
N2C3W1_M	31	8	31	128
N2C3W1_N	32	9	32	139
N2C3W1_O	35	8	35	128
N2C3W1_P	35	8	35	128
N2C3W1_Q	34	10	34	120
N2C3W1_R	33	13	33	133
N2C3W1_S	36	12	36	132
N2C3W1_T	35	12	35	132
N2C3W2_A	42	11	42	131
N2C3W2_B	43	9	43	129
N2C3W2_C	42	8	42	128
N2C3W2_D	42	9	42	129
N2C3W2_E	40	14	40	124
N2C3W2_F	40	13	40	123
N2C3W2_G	41	9	41	139
N2C3W2_H	38	8	38	118
N2C3W2_I	45	8	45	108
N2C3W2_J	44	8	44	108
N2C3W2_K	42	10	42	140
N2C3W2_L	42	10	42	140
N2C3W2_M	44	10	44	110
N2C3W2_N	42	10	42	110
N2C3W2_O	45	8	45	138
N2C3W2_P	41	8	41	128
N2C3W2_Q	42	8	42	128
N2C3W2_R	41	8	41	108
N2C3W2_S	44	8	44	108

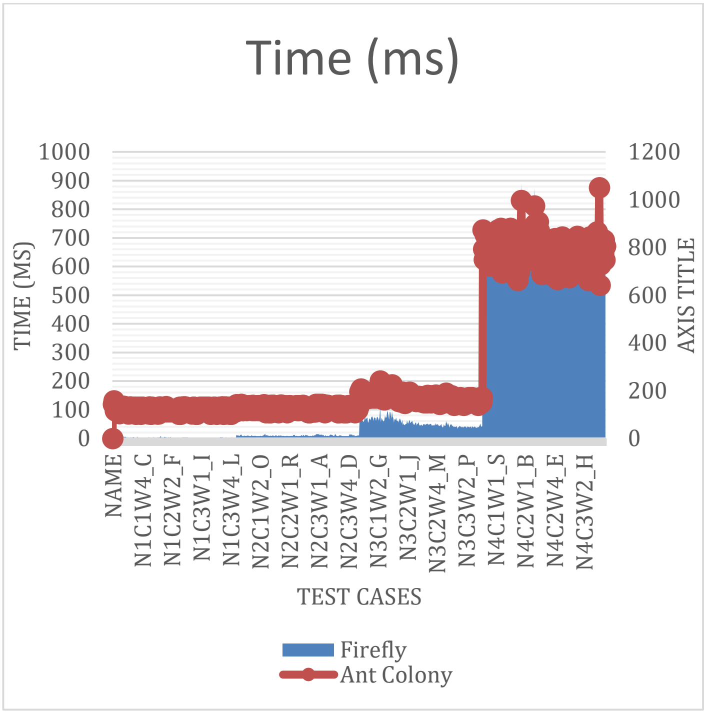
N2C3W2_T	43	8	43	128
N2C3W4_A	44	8	44	118
N2C3W4_B	46	8	46	108
N2C3W4_C	43	8	43	118
N2C3W4_D	45	8	45	138
N2C3W4_E	47	8	47	138
N2C3W4_F	46	13	46	133
N2C3W4_G	45	15	45	125
N2C3W4_H	45	12	45	112
N2C3W4_I	46	13	46	133
N2C3W4_J	44	12	44	132
N2C3W4_K	47	9	47	119
N2C3W4_L	46	11	46	141
N2C3W4_M	45	11	45	111
N2C3W4_N	46	8	46	108
N2C3W4_O	46	8	46	138
N2C3W4_P	46	9	46	119
N2C3W4_Q	47	8	47	118
N2C3W4_R	43	11	43	121
N2C3W4_S	43	11	43	131
N2C3W4_T	47	8	47	138
N3C1W1_A	106	76	106	196
N3C1W1_B	114	78	114	178
N3C1W1_C	99	88	99	208
N3C1W1_D	108	74	108	204
N3C1W1_E	98	61	98	161
N3C1W1_F	113	59	113	169
N3C1W1_G	111	57	111	187
N3C1W1_H	104	67	104	197
N3C1W1_I	100	74	100	184
N3C1W1_J	108	65	108	185
N3C1W1_K	102	72	102	192
N3C1W1_L	98	73	98	183
N3C1W1_M	106	55	106	185
N3C1W1_N	93	57	93	187
N3C1W1_O	99	69	99	169
N3C1W1_P	108	68	108	178
N3C1W1_Q	98	63	98	193
N3C1W1_R	99	73	99	173
N3C1W1_S	101	77	101	187
N3C1W1_T	102	71	102	191
N3C1W2_A	125	60	125	170
N3C1W2_B	127	75	127	195
N3C1W2_C	127	70	127	190
N3C1W2_D	140	71	140	181
N3C1W2_E	133	60	133	170
N3C1W2_F	125	75	125	205
N3C1W2_G	132	74	132	194
N3C1W2_H	132	69	132	179
N3C1W2_I	127	61	127	191
N3C1W2_J	126	74	126	194
N3C1W2_K	123	111	123	241
N3C1W2_L	137	57	137	187
N3C1W2_M	137	71	137	181
N3C1W2_N	137	65	137	165
N3C1W2_O	128	60	128	180
N3C1W2_P	126	59	126	189
N3C1W2_Q	135	62	135	162
N3C1W2_R	125	79	125	179
N3C1W2_S	131	71	131	201
N3C1W2_T	137	74	137	184
N3C1W4_A	151	96	151	196
N3C1W4_B	150	91	150	211
N3C1W4_C	147	78	147	208
N3C1W4_D	150	65	150	175
N3C1W4_E	143	93	143	223
N3C1W4_F	143	91	143	211
N3C1W4_G	148	68	148	188
N3C1W4_H	143	95	143	225
N3C1W4_I	145	76	145	176
N3C1W4_J	145	72	145	182
N3C1W4_K	148	55	148	185
N3C1W4_L	150	68	150	178
N3C1W4_M	152	64	152	194
N3C1W4_N	150	71	150	191
N3C1W4_O	144	61	144	171
N3C1W4_P	145	60	145	190
N3C1W4_Q	147	61	147	181
N3C1W4_R	146	55	146	155
N3C1W4_S	146	66	146	196
N3C1W4_T	147	69	147	169
N3C2W1_A	91	62	91	172
N3C2W1_B	83	53	83	153
N3C2W1_C	84	65	84	195
N3C2W1_D	85	46	85	176
N3C2W1_E	87	53	87	173
N3C2W1_F	88	46	88	156
N3C2W1_G	88	47	88	147
N3C2W1_H	87	49	87	169
N3C2W1_I	87	50	87	180

N3C2W1_J	87	61	87	171
N3C2W1_K	78	54	78	184
N3C2W1_L	91	51	91	161
N3C2W1_M	85	59	85	179
N3C2W1_N	91	65	91	195
N3C2W1_O	82	52	82	162
N3C2W1_P	89	58	89	168
N3C2W1_Q	83	58	83	168
N3C2W1_R	83	60	83	170
N3C2W1_S	89	47	89	177
N3C2W1_T	83	53	83	173
N3C2W2_A	107	46	107	166
N3C2W2_B	105	52	105	172
N3C2W2_C	105	56	105	156
N3C2W2_D	108	45	108	165
N3C2W2_E	116	50	116	160
N3C2W2_F	107	51	107	161
N3C2W2_G	103	57	103	177
N3C2W2_H	117	47	117	157
N3C2W2_I	102	45	102	155
N3C2W2_J	107	46	107	166
N3C2W2_K	110	48	110	158
N3C2W2_L	105	43	105	153
N3C2W2_M	108	45	108	175
N3C2W2_N	107	49	107	169
N3C2W2_O	108	50	108	150
N3C2W2_P	107	48	107	168
N3C2W2_Q	105	47	105	167
N3C2W2_R	110	49	110	169
N3C2W2_S	107	50	107	150
N3C2W2_T	107	51	107	181
N3C2W4_A	114	47	114	157
N3C2W4_B	114	49	114	179
N3C2W4_C	134	48	134	178
N3C2W4_D	115	50	115	180
N3C2W4_E	113	50	113	150
N3C2W4_F	115	45	115	165
N3C2W4_G	123	45	123	155
N3C2W4_H	114	46	114	166
N3C2W4_I	116	45	116	165
N3C2W4_J	121	48	121	168
N3C2W4_K	117	43	117	153
N3C2W4_L	118	52	118	182
N3C2W4_M	121	47	121	177
N3C2W4_N	119	47	119	177
N3C2W4_O	115	50	115	160
N3C2W4_P	123	46	123	166
N3C2W4_Q	118	42	118	152
N3C2W4_R	124	44	124	144
N3C2W4_S	119	46	119	156
N3C2W4_T	119	50	119	170
N3C3W1_A	66	41	66	161
N3C3W1_B	71	38	71	158
N3C3W1_C	69	39	69	149
N3C3W1_D	63	49	63	179
N3C3W1_E	69	44	69	154
N3C3W1_F	69	44	69	174
N3C3W1_G	65	60	65	190
N3C3W1_H	69	53	69	173
N3C3W1_I	69	51	69	161
N3C3W1_J	65	45	65	155
N3C3W1_K	63	54	63	154
N3C3W1_L	68	37	68	157
N3C3W1_M	72	48	72	158
N3C3W1_N	69	48	69	158
N3C3W1_O	66	49	66	179
N3C3W1_P	73	42	73	162
N3C3W1_Q	73	45	73	145
N3C3W1_R	66	40	66	170
N3C3W1_S	68	39	68	139
N3C3W1_T	70	41	70	161
N3C3W2_A	85	39	85	159
N3C3W2_B	83	42	83	162
N3C3W2_C	83	40	83	150
N3C3W2_D	80	34	80	164
N3C3W2_E	80	40	80	170
N3C3W2_F	83	38	83	158
N3C3W2_G	82	43	82	143
N3C3W2_H	83	41	83	171
N3C3W2_I	81	38	81	168
N3C3W2_J	85	40	85	160
N3C3W2_K	84	40	84	150
N3C3W2_L	83	37	83	167
N3C3W2_M	85	39	85	139
N3C3W2_N	79	41	79	161
N3C3W2_O	84	39	84	159
N3C3W2_P	83	42	83	152
N3C3W2_Q	77	35	77	145
N3C3W2_R	81	42	81	162
N3C3W2_S	81	38	81	168

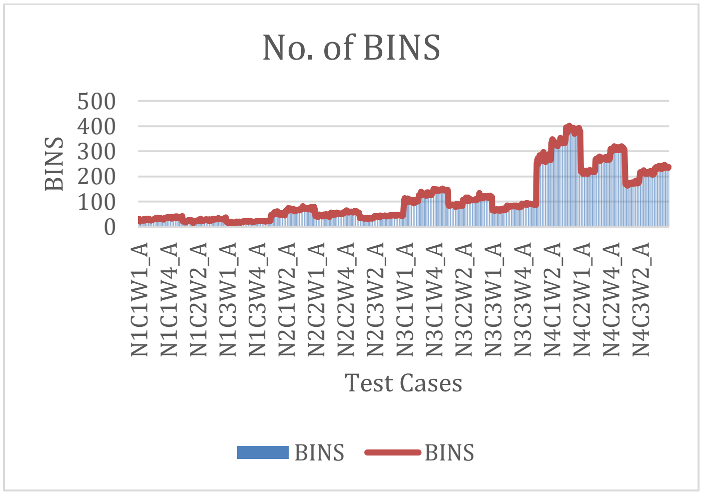
N3C3W2_T	80	41	80	171
N3C3W4_A	92	38	92	168
N3C3W4_B	90	40	90	150
N3C3W4_C	91	41	91	171
N3C3W4_D	89	35	89	155
N3C3W4_E	88	42	88	172
N3C3W4_F	88	40	88	170
N3C3W4_G	95	38	95	168
N3C3W4_H	87	39	87	159
N3C3W4_I	94	41	94	161
N3C3W4_J	91	38	91	158
N3C3W4_K	90	37	90	157
N3C3W4_L	92	37	92	157
N3C3W4_M	91	38	91	168
N3C3W4_N	90	39	90	139
N3C3W4_O	90	39	90	149
N3C3W4_P	89	45	89	155
N3C3W4_Q	91	51	91	171
N3C3W4_R	91	42	91	172
N3C3W4_S	86	40	86	150
N3C3W4_T	87	42	87	172
N4C1W1_A	253	755	253	875
N4C1W1_B	272	694	272	794
N4C1W1_C	250	619	250	749
N4C1W1_D	256	734	256	854
N4C1W1_E	285	688	285	798
N4C1W1_F	280	714	280	844
N4C1W1_G	267	664	267	774
N4C1W1_H	263	732	263	862
N4C1W1_I	271	709	271	829
N4C1W1_J	297	604	297	724
N4C1W1_K	260	631	260	751
N4C1W1_L	268	606	268	726
N4C1W1_M	258	726	258	856
N4C1W1_N	267	698	267	808
N4C1W1_O	267	652	267	752
N4C1W1_P	278	641	278	761
N4C1W1_Q	288	649	288	749
N4C1W1_R	266	721	266	851
N4C1W1_S	271	689	271	799
N4C1W1_T	265	745	265	845
N4C1W2_A	333	648	333	768
N4C1W2_B	348	647	348	747
N4C1W2_C	344	744	344	874
N4C1W2_D	342	649	342	769
N4C1W2_E	327	676	327	806
N4C1W2_F	335	652	335	752
N4C1W2_G	329	761	329	881
N4C1W2_H	331	690	331	820
N4C1W2_I	320	585	320	695
N4C1W2_J	330	650	330	770
N4C1W2_K	330	690	330	800
N4C1W2_L	332	651	332	771
N4C1W2_M	353	680	353	800
N4C1W2_N	335	712	335	832
N4C1W2_O	333	670	333	800
N4C1W2_P	332	588	332	698
N4C1W2_Q	339	690	339	800
N4C1W2_R	341	659	341	779
N4C1W2_S	333	714	333	844
N4C1W2_T	343	645	343	775
N4C1W4_A	395	771	395	881
N4C1W4_B	373	677	373	797
N4C1W4_C	383	638	383	748
N4C1W4_D	389	715	389	825
N4C1W4_E	402	686	402	786
N4C1W4_F	390	654	390	764
N4C1W4_G	384	571	384	691
N4C1W4_H	387	698	387	798
N4C1W4_I	384	672	384	772
N4C1W4_J	394	730	394	830
N4C1W4_K	393	731	393	851
N4C1W4_L	370	553	370	663
N4C1W4_M	385	667	385	767
N4C1W4_N	384	579	384	699
N4C1W4_O	378	716	378	826
N4C1W4_P	391	767	391	867
N4C1W4_Q	391	888	391	998
N4C1W4_R	393	691	393	801
N4C1W4_S	380	703	380	803
N4C1W4_T	380	662	380	772
N4C2W1_A	220	654	220	784
N4C2W1_B	219	643	219	773
N4C2W1_C	221	726	221	856
N4C2W1_D	211	725	211	855
N4C2W1_E	224	675	224	805
N4C2W1_F	212	707	212	817
N4C2W1_G	221	734	221	834
N4C2W1_H	224	677	224	787
N4C2W1_I	218	714	218	844

N4C2W1_J	210	658	210	778
N4C2W1_K	219	673	219	793
N4C2W1_L	218	670	218	790
N4C2W1_M	226	723	226	853
N4C2W1_N	219	772	219	872
N4C2W1_O	222	701	222	821
N4C2W1_P	223	875	223	975
N4C2W1_Q	219	789	219	899
N4C2W1_R	222	754	222	864
N4C2W1_S	217	712	217	832
N4C2W1_T	221	639	221	739
N4C2W2_A	268	666	268	796
N4C2W2_B	272	798	272	908
N4C2W2_C	265	761	265	871
N4C2W2_D	271	722	271	822
N4C2W2_E	268	645	268	775
N4C2W2_F	280	681	280	801
N4C2W2_G	262	558	262	688
N4C2W2_H	266	653	266	763
N4C2W2_I	276	680	276	800
N4C2W2_J	269	652	269	782
N4C2W2_K	272	706	272	806
N4C2W2_L	272	668	272	778
N4C2W2_M	269	600	269	700
N4C2W2_N	277	613	277	743
N4C2W2_O	265	677	265	807
N4C2W2_P	277	590	277	700
N4C2W2_Q	269	644	269	764
N4C2W2_R	265	611	265	741
N4C2W2_S	284	593	284	713
N4C2W2_T	268	655	268	765
N4C2W4_A	310	691	310	801
N4C2W4_B	298	629	298	739
N4C2W4_C	307	573	307	683
N4C2W4_D	311	633	311	733
N4C2W4_E	304	673	304	803
N4C2W4_F	320	718	320	838
N4C2W4_G	310	667	310	767
N4C2W4_H	318	698	318	818
N4C2W4_I	305	652	305	782
N4C2W4_J	313	556	313	666
N4C2W4_K	305	628	305	748
N4C2W4_L	316	630	316	750
N4C2W4_M	305	616	305	716
N4C2W4_N	314	656	314	766
N4C2W4_O	309	664	309	774
N4C2W4_P	320	687	320	787
N4C2W4_Q	306	735	306	845
N4C2W4_R	315	629	315	729
N4C2W4_S	309	629	309	749
N4C2W4_T	307	703	307	833
N4C3W1_A	170	576	170	706
N4C3W1_B	170	659	170	769
N4C3W1_C	172	670	172	800
N4C3W1_D	164	644	164	754
N4C3W1_E	171	647	171	777
N4C3W1_F	168	604	168	724
N4C3W1_G	174	555	174	675
N4C3W1_H	175	570	175	680
N4C3W1_I	175	703	175	823
N4C3W1_J	177	611	177	741
N4C3W1_K	170	659	170	789
N4C3W1_L	171	660	171	760
N4C3W1_M	173	679	173	799
N4C3W1_N	182	624	182	744
N4C3W1_O	175	684	175	814
N4C3W1_P	173	600	173	700
N4C3W1_Q	184	707	184	817
N4C3W1_R	172	707	172	837
N4C3W1_S	175	717	175	847
N4C3W1_T	179	636	179	756
N4C3W2_A	217	589	217	719
N4C3W2_B	216	604	216	704
N4C3W2_C	213	587	213	717
N4C3W2_D	217	657	217	767
N4C3W2_E	216	653	216	763
N4C3W2_F	225	696	225	816
N4C3W2_G	217	694	217	794
N4C3W2_H	216	676	216	776
N4C3W2_I	210	608	210	738
N4C3W2_J	217	639	217	749
N4C3W2_K	218	595	218	705
N4C3W2_L	215	622	215	752
N4C3W2_M	212	660	212	780
N4C3W2_N	212	620	212	740
N4C3W2_O	222	532	222	662
N4C3W2_P	215	630	215	740
N4C3W2_Q	216	725	216	845
N4C3W2_R	207	590	207	720
N4C3W2_S	210	611	210	711

N4C3W2_T	209	622	209	742
N4C3W4_A	233	597	233	717
N4C3W4_B	235	719	235	829
N4C3W4_C	238	638	238	748
N4C3W4_D	230	663	230	783
N4C3W4_E	239	719	239	839
N4C3W4_F	238	729	238	859
N4C3W4_G	242	717	242	827
N4C3W4_H	238	737	238	867
N4C3W4_I	241	699	241	829
N4C3W4_J	230	711	230	841
N4C3W4_K	237	922	237	1052
N4C3W4_L	235	542	235	642
N4C3W4_M	233	598	233	728
N4C3W4_N	246	709	246	839
N4C3W4_O	242	656	242	776
N4C3W4_P	237	663	237	793
N4C3W4_Q	240	646	240	776
N4C3W4_R	235	701	235	831
N4C3W4_S	234	618	234	748
N4C3W4_T	237	686	237	806



All these data's were tabulated graphically.



VIII. References

NATURE INSPIRED ALGORITHMS: SUCCESS AND CHALLENGES – XIN-SHE YANG

BACHELOR THESIS IMPLEMENTATION OF A DISCRETE FIREFLY ALGORITHM FOR THE QAP PROBLEM WITHIN THE SEAGE FRAMEWORK – KAREL DURKOTA – MAY 2011

IX. Authors

Abhirupa Mitra Currently pursuing her Bachelor of Technology in Computer Science and Engineering and takes special interest in Web Development and Blockchain technology.

Nikhil Anand: Currently pursuing his Bachelor of Technology in Computer Science and Engineering with specialization in Bioinformatics.