Artificial Intelligence
Course Project Report

Application of Genetic Algorithm to Solve Graph Colouring

Akshat Sharda
2014009

Motivation

Graph coloring is an NP-complete problem of vital importance. It can easily be modelled as a scheduling problem. An instance of the same a given set of jobs need to be assigned to time slots, each job requires one such slot. Jobs can be scheduled in any order, but pairs of jobs may be in conflict in the sense that they may not be assigned to the same time slot, for example because they both rely on a shared resource.[1]
Evolutionary algorithms are a class of algorithms that are very powerful at solving a variety of problems. However, they do not get this level of attention from the research community. I chose this project to show that solving meaningful problems like Graph coloring is possible, and effective when using Genetic Algorithms.

Objective

In this project, I aimed to formulate the graph coloring problem as a genetic algorithm. This genetic algorithm was to be implemented in Python, keeping in mind that it should be easy to modify the parameters for the same. Once implemented for a given mutation and crossover function, it was expected that multiple mutation and crossover functions would be implemented so that these can be compared in the analysis. Once this would be done, the performance of these functions would be compared based on a few parameters like :

- Final score of the best chromosome
- Time taken to reach maxima
- Iterations required to reach the maxima
- Number of colors that were used to finally color the graph

Milestones

First evaluation

- Decide on mutation function, survivor selection and fitness criteria. (DONE)
- Create genetic learning algorithm for a specific graph in form of pseudocode. (DONE)

Second evaluation

- Implement complete genetic learning algorithm in Python for a specific mutation function. (DONE)

Tasks

1. Decide on the following for a genetic algorithm to solve graph colouring :
    a. Mutation functions to be compared
    b. Survivor selection
    c. Fitness function for the algorithm
    d. End condition for the genetic algorithm

    (DONE)
2. Create pseudocode for the algorithm proposed. (DONE)
3. Implement algorithm in python. (DONE)
4. Create data set comprising of graphs of multiple sizes, as input to this code.(DONE)
5. Compare mutation functions for genetic learning algorithm. (DONE)
6. Visualize data to see variation of performance with n, the number of nodes in the graph. (DONE)
7. Draw inferences from this visualization.(DONE)
8. Created GUI for better interpretation of code run, better control over parameters that can be tweaked. (EXTENDED)

Implementation

MUTATION FUNCTIONS :

Following mutation (and crossover) functions will be considered for comparison:

- **C** Selecting color of first *n* points from first parent, and remaining points from second
- **C** Selecting colour of alternate point from each parent, i.e, node *i* will be coloured as it is in parent *i*%2
- **M** Swapping colors of n random points
- **M** Modifying color of worst placed point to a valid colour, based on adjacent nodes
- **M** Modifying color of worst placed point to a random colour
- **M** Modifying color of ANY wrongly coloured point to a valid colour
- **M** Modifying color of ANY wrongly coloured point to a random

* **M** and **C** stand for mutation and crossover respectively

SURVIVOR SELECTION :

The fittest *n* chromosomes will be selected as parents for next generation.

FITNESS CRITERIA:

The fitness of any chromosome will be defined by the following pseudocode :

```
def fitness(chromosome, graph) :
    score = 0
    for edge in graph.edges :
       if
chromosome.get_colour(edge.start) ==
chromosome.get_colour(edge.end):
             score += 1
    return score
```

PSEUDOCODE FOR GA :

```
function GA(mutation,
regularization) {
    Initialize population with
random chromosomes
    Evaluate population
    while(termination criteria not
satisfied OR repetition detected for
more than r times){
       Select top n parents for
reproduction
       Perform mutation using
argument function
       Introduce regularization if
input function is not None.
       Evaluate population
    }
}
```

TERMINATION CRITERIA :

The GA will be terminated successfully if :

- The fittest parent in population will have 0 score
- The fittest parent has been the same for last *r* iterations.

The GA will be terminated unsuccessfully if either of the following are true :

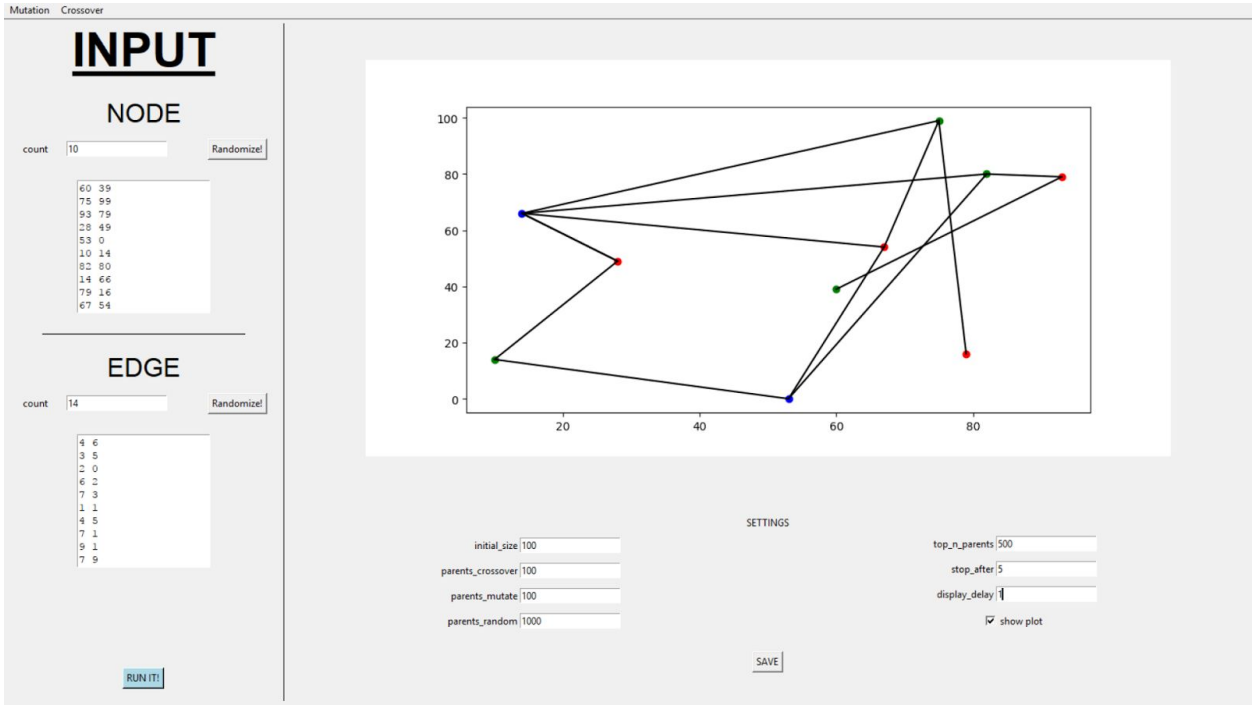
- Limit of *t* iterations is exceeded.

Link to github repository:
https://bitbucket.org/acethepace/ai-project

Results

A genetic algorithm for solving Graph Coloring was proposed and implemented. This GA was exposed through a Graphical User Interface for better control over parameters.

Attached below is a screenshot for the same :

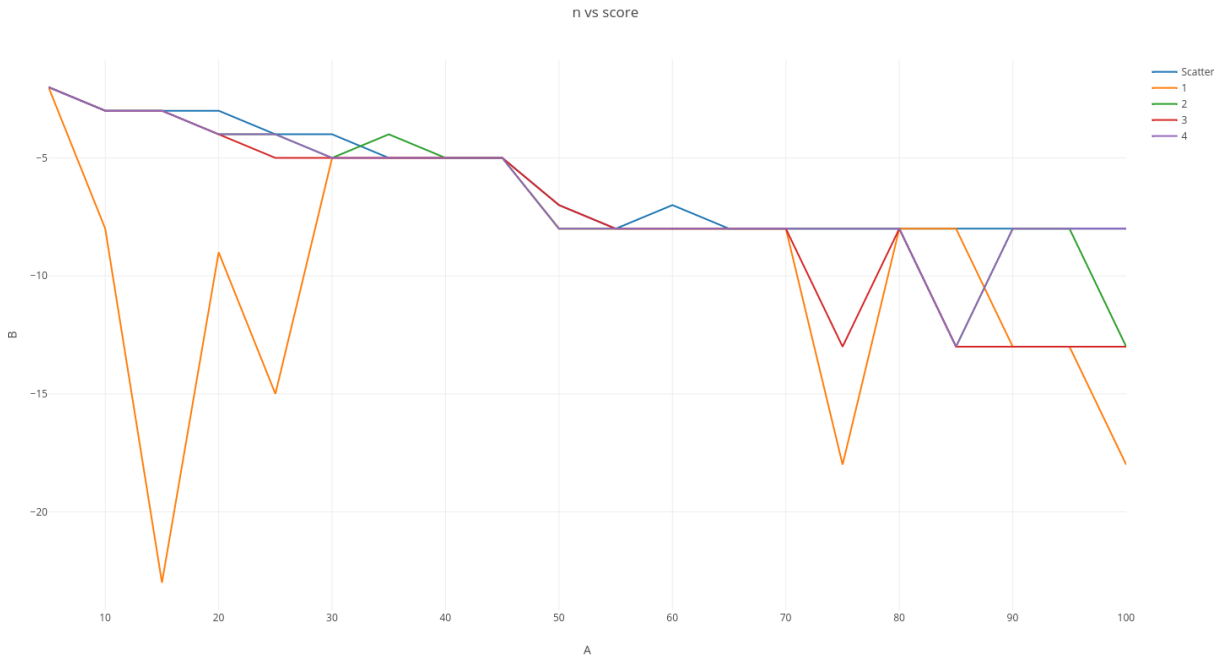


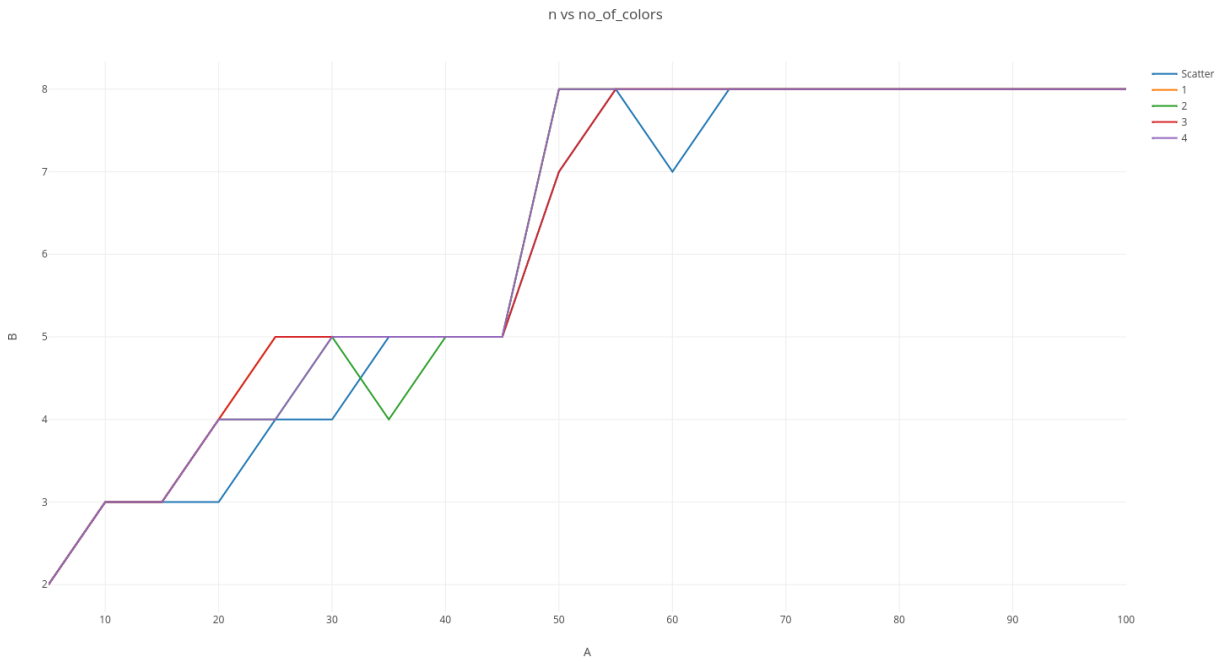
The graph is a real time depiction of the algorithm. Each black line represents an edge, and each vertex represents the nodes. The color of each node is the color of that node as assigned by the best performing chromosome in the Genetic Algorithm after each iteration. The settings can be modified by altering them from the settings panel in the GUI. Mutation and Crossover functions can be selected from the drop down menu.

When ready, press blue button, “RUN IT!” for starting the GA.

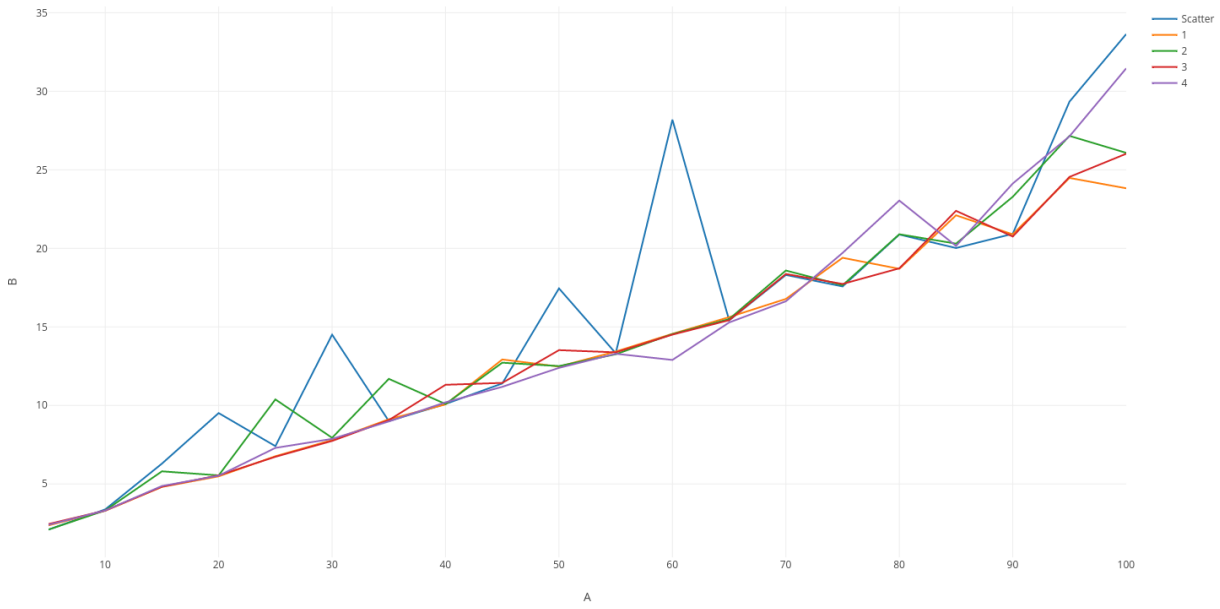
Analysis

Mutation Functions

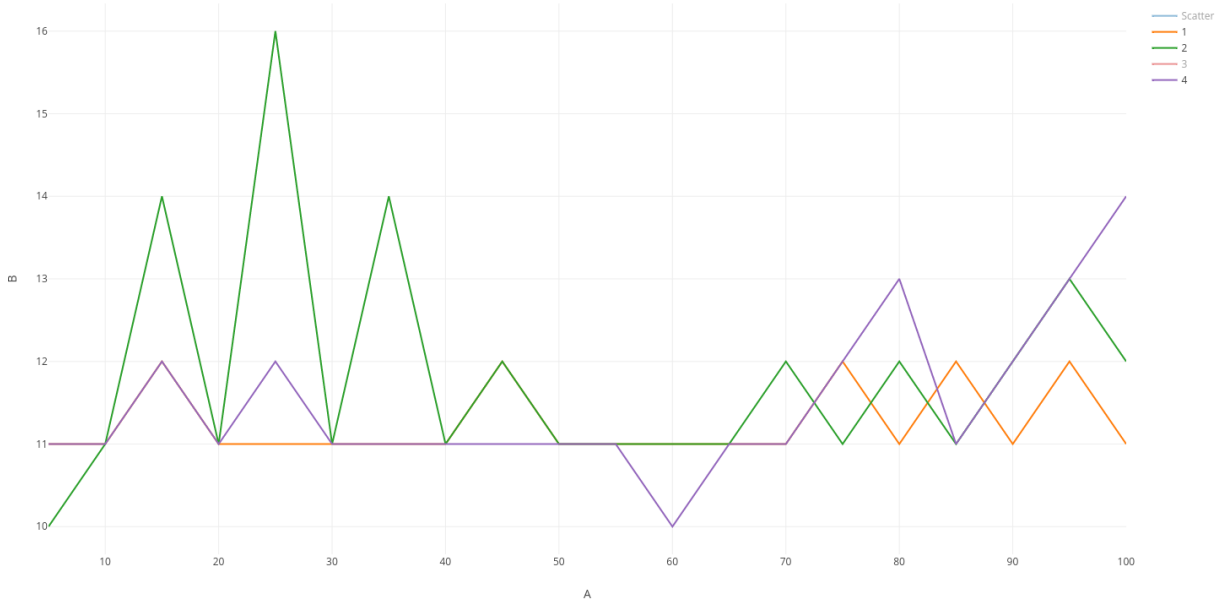




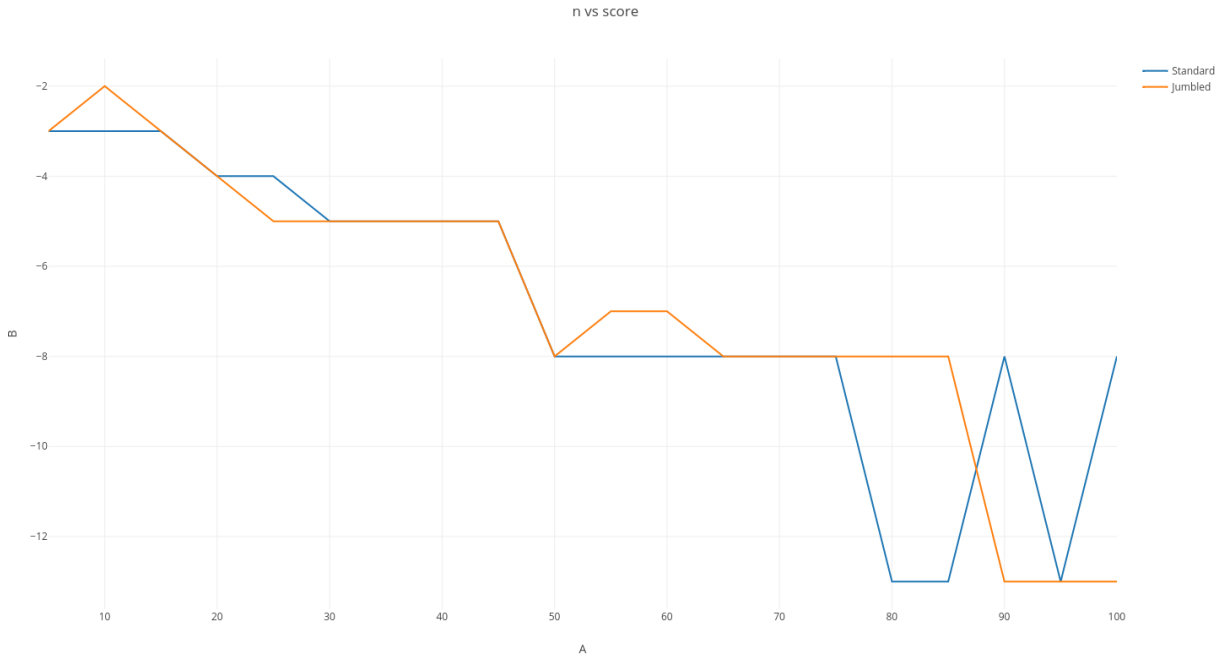
n vs time taken

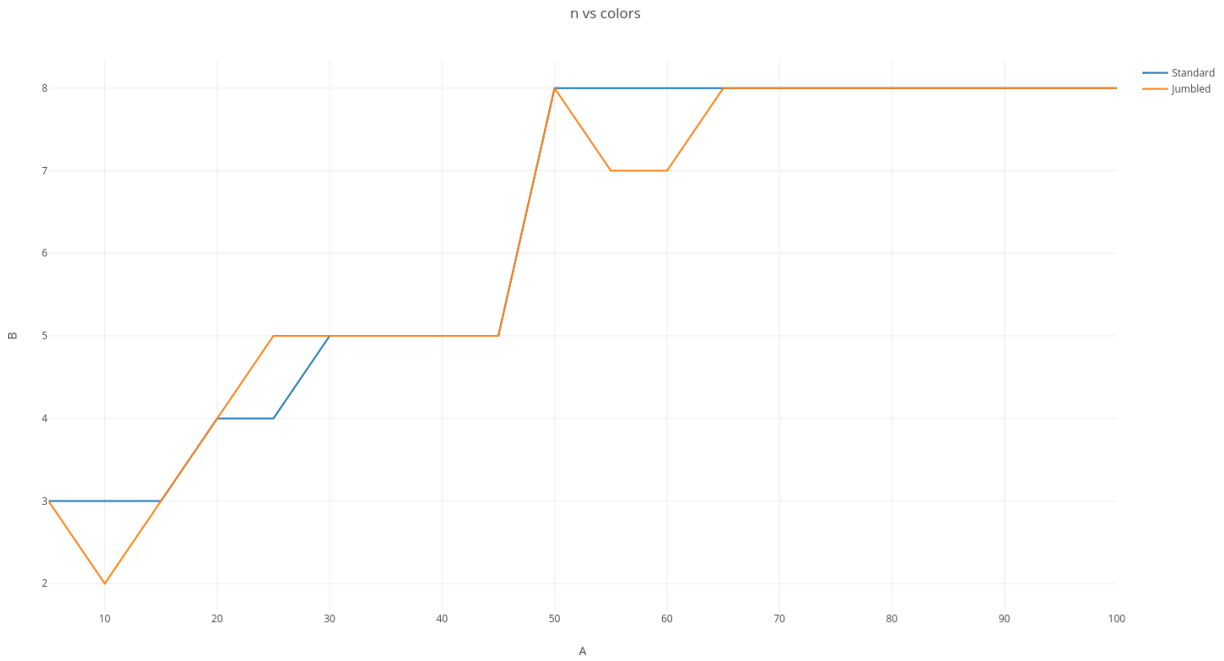


n vs no_of_iterations

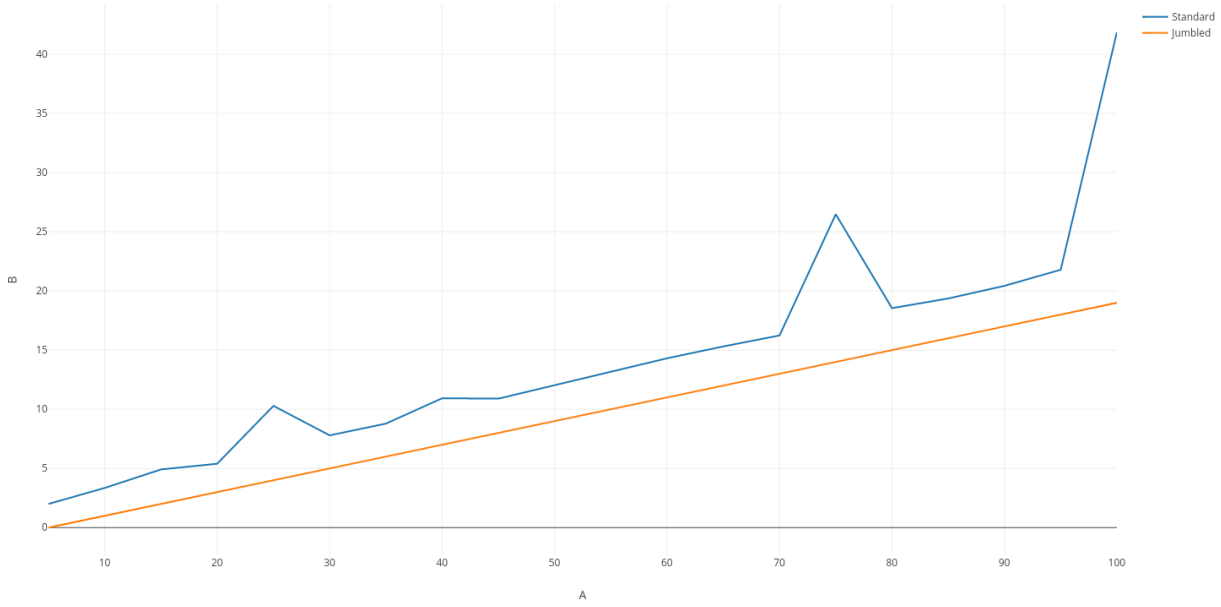


Crossover Functions

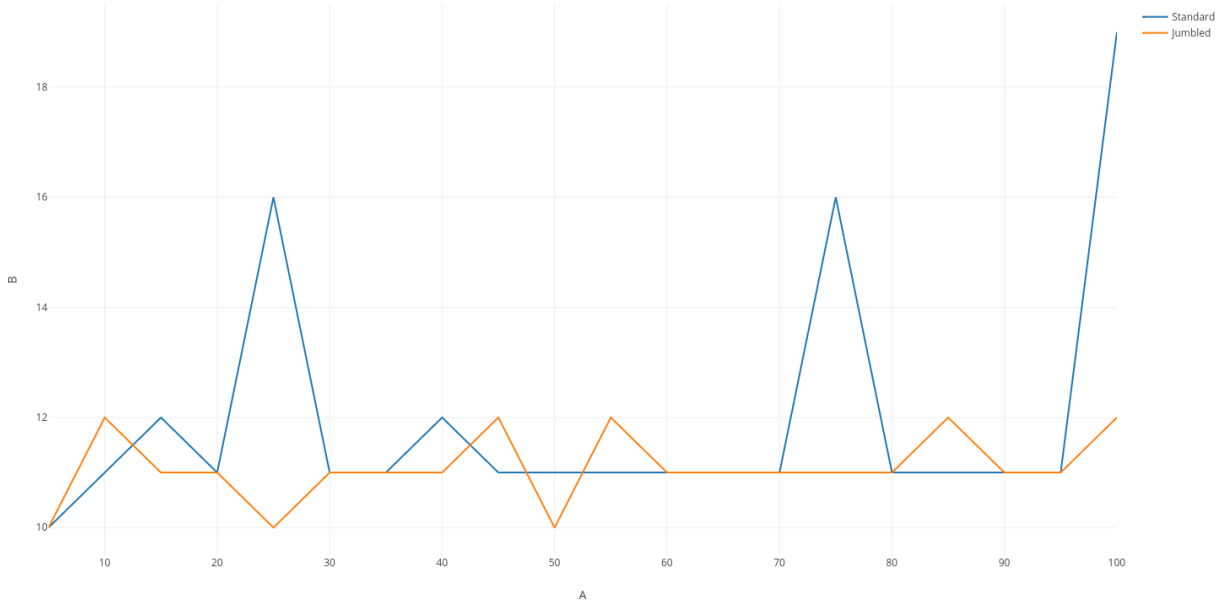




n vs time_taken



n vs number_of_iterations



Links to datasets and more detailed graphs:

- https://plot.ly/create/?fid=Mallock:52
- https://plot.ly/create/?fid=Mallock:57
- https://plot.ly/create/?fid=Mallock:58
- https://plot.ly/create/?fid=Mallock:59
- https://plot.ly/create/?fid=Mallock:64
- https://plot.ly/create/?fid=Mallock:65
- https://plot.ly/create/?fid=Mallock:67
- https://plot.ly/create/?fid=Mallock:68

References :

[1] : https://en.wikipedia.org/wiki/Graph_coloring#Applications
[2] : http://ceur-ws.org/Vol-841/submission_10.pdf