BTP REPORT

CROWD ANALYTICS FOR LARGE EVENTS

November 29, 2017

Chakka Gowri Manasa
Moturi Praneetha
Shruti Gupta

November 29, 2017

INTRODUCTION

Managing large crowds can be a challenging task. Our project aims to exploit technology to its full potential, for monitoring crowd movements in big events to help with smart event planning and avoiding accidents. This calls for an automated crowd monitoring system. This project aims to automatically analyze the crowd mobility patterns and generate timely alerts to avert any unwanted stampede situation.

MOTIVATION

India is a country with a large population which actively involves itself in events composed of large crowds. However as evidenced by countless reports of accidents, stampedes, deaths in these events caused of lack of awareness, lack of sensitivity towards the safety measures to be taken to prevent unforeseen accidents. Large crowds always invite accidents if preventive measures are not taken with proper planning. Excessive crowding and poor crowd management can cost loss of precious life very easily. Hence the need for developing an automated crowd management system will be highly appreciated by the society.

PROBLEM STATEMENT

To develop an automated system that analyses crowd mobility patterns in large events. The project aims to facilitate smart event planning and an optimised resource allocation.
We are considering an airport scenario, specifically Rajiv Gandhi International airport.

EVENTS

- Open Area:
  It is susceptible to external conditions such as weather.
  e.g: Kumbh Mela, Rallies
- Closed Area:
  It is independent of external conditions. It is confined to a fixed region.

e.g: Airport

- Disciplined:
  These events are ordered. They follow certain well-defined paths
  e.g: Airport, Conferences

- Undisciplined:
  These events are less ordered. They proceed in a more chaotic fashion.
  e.g: Concerts, Kumbh Mela

CROWD ANALYSIS

- Density:
  Number of people attending a particular event, spread out over time intervals.

- Crowd division:
  The possible paths people can take (at an event) , as well as crowd distribution.

- Resource analysis:
  Identify what resources may be required by people and analyze their availability.

USE CASES

- Find best path
- Generate alerts to Stakeholders
- Suggest alternate paths
- Resource Allocation

DATA COLLECTION

For data collection i.e history of past flights over an year, we requested quotations from multiple websites such as FlightStats, Flightradar24, OAG, FlightAware, skyscanner and many more.Based on the pricing, size and structure of data available, we shortlisted three of them. The cross-analysis of these websites involved evaluation of the following parameters...

	FlightAware	FlightStats	Flightradar24
Departure Airport	✓	✓	✓
Arrival Airport	✓	✓	✓
Departure Time	✓	✓	✓
Arrival Time	✓	✓	✓
Airline	✓	✓	✓
Aircraft	✓	✓	✗
Gate Number	✗	✓	✗

DATA ANALYSIS

The parameters collected from Flightaware were evaluated for synthesizing crowd count/density:

- Day/Time : Days such as Friday and Sunday were given more weightage, similarly evening rush hours (6pm, 7pm..) etc.

- Airline : Popular airline carriers such as Indigo and Spicejet were given higher weights.
- Destination airport : Frequented destinations such as Delhi, Mumbai were given higher weights.

Based on the above parameters, we estimated the mean number of people at the airport at a given time, i.e we have the occupancy for each flight.
Furthermore, we have assigned various values for the parameters and generated three different sets of data.

CURRENT SCENARIO : AIRPORT

It can be classified as a closed and disciplined event. In our scenario :-

Notations:

F : set of Flights
G : set of Gates
R : set of Resources
n_i^o : number of origin passengers of flight i
n_i^d : number of departure passengers of flight i
d_j^s : distance from a security checkpoint to gate j
d_j^b : distance from gate j to baggage claim
d_j^r : distance from a resource to gate j
w_r : weight of a resource r
x_{ij} : decision variable that indicates whether flight i is assigned to gate j
x_{kl} : decision variable that indicates whether flight k is assigned to gate l
n_{ik} : number of transfer passengers between flights i and k
d_{jl} : distance between gates j and l

Metric I: Walking Distance To Gate

The passenger movement is observed from the security checkpoint to the departure gates and the arrival gates to baggage claim, also in the case of connecting flights, passengers moving from one to another is taken into consideration. Our motive is to minimize the walking distance of the passengers. So we formulate an objective function which is to be minimized :

$$\sum_{i\in F}\sum_{j\in G}(n_i^o d_j^s + n_i^d d_j^b)x_{ij} + \sum_{i\in F}\sum_{j\in G}\sum_{k\in F,k>i}\sum_{l\in G} n_{ik}d_{jl}x_{ij}x_{kl} \tag{1}$$

Metric II: Walking Distance To Resources

The passenger movement is observed to and fro the resources and the departure gates. There are weights assigned to each resource according to the probability of them being visited.

$$\sum_{i\in F}\sum_{j\in G}\sum_{r\in R}(n_i^o d_j^r)\,w_r x_{ij} + \sum_{i\in F}\sum_{j\in G}\sum_{k\in F,k>i}\sum_{l\in G}\sum_{r\in R}(n_{ik} d_j^r)\,w_r x_{ij} x_{kl} \tag{2}$$

So the combined objective function is:

Minimize:

$$w_1 * 1 + w_2 * 2$$

Conditions:

$$w_1 + w_2 = 1 \tag{3}$$

$$w_1, w_2 \geq 0 \tag{4}$$

$$\sum_{j\in G} x_{ij} = 1, \forall i \in F \tag{5}$$

$$x_{ij} \in \{0,1\}, \forall i \in F, \forall j \in G \tag{6}$$

Constrained Objective Function

However due to data constraints, i.e lack of information on the connecting passengers, the objective function is modified:

Notations:

n_i^o : number of origin passengers of flight i F = Flights

d_j^s : distance from a security checkpoint to gate j G = Gates

x_{ij} : decision variable that indicates whether flight i is assigned to gate j

Objective Function:

Minimize:

$$\sum_{i \in F} \sum_{j \in G} (n_i^o d_j^s) x_{ij} \tag{7}$$

Conditions:

$$\sum_{j \in G} x_{ij} = 1, \forall i \in F \qquad x_{ij} \in \{0,1\}, \forall i \in F, \forall j \in G \tag{8}$$

ALGORITHMS

The following algorithms were explored:

1. Constrained Non Linear:
    - Barrier
    - Penalty
2. Convex Optimization:
    - Convex Minimization
    - Cutting Plane
    - Sub Gradient
3. Combinatorial:
    - Greedy
    - Branch and Bound
    - Genetic Algorithm
4. Metaheuristic
    - Hill Climbing
    - Local search
    - Simulated Annealing
    - Tabu Search

TABU SEARCH

It is a local search method which takes a potential solution to a problem and checks its immediate neighbours in the hope of finding an improved solution.
However, local searches have a tendency to get stuck in suboptimal regions.
Tabu search enhances the performance by:
a) Accepting worsening moves at each step, if no improving moves available.
b) Discourages going back to previously-visited solutions(using Tabu memory).

Definitions

- **Move:** A modification made to a solution. If the number of tabu moves is small, previously visited solutions are more frequent whereas if they are large, then the search is less likely to find good local optima due to lack of available moves.
- **Tabu list:** In order to prevent visiting a previously visited solution, TS uses a tabu list in which tabu moves or attributes of moves are listed.
- **Short-term memory:** A type of memory that is limited in terms of time and storage capacity. With a short-term memory, a previsited solution may be revisited with a different neighborhood. Short tabu lists may not prevent cycling resulting in information loss.
- **Long-term memory:** A type of memory that is larger in terms of time and storage capacity. The likelihood of visiting a previously visited solution using long-term memory is very small.

TABU SEARCH IMPLEMENTATION

We are using Tabu search with long-term memory (limit=200)
We have experimented with three types of moves :

- Insert Move
- Swap Intervals Move
- Single Swap Move

Along with two types of inputs:

- Greedy output
- Round robin assignment with randomized starting points

We chose a combination of Insert and Swap Intervals Move.
Also the initialization was done with greedy.
The complexity is given as $O(g * m^3)$ where the number of flights is specific for a given day.
g : no. of iterations
m : no. of flights

GREEDY APPROACH

A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.

GENETIC ALGORITHM

A genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection.

INITIAL POPULATION

- We are employing steady state population model, i.e, we generate one or two off-springs in each iteration and they replace one or two individuals from the population.
- Two types of initializations were used:
  - *Random Initialization*: N (population size) random parents initially using Round robin.
  - *Heuristic Initialization*: Starting with 2 parents (GreedyâĂŹs output) and evolving over iterations using Crossover and Mutation until we reach N.

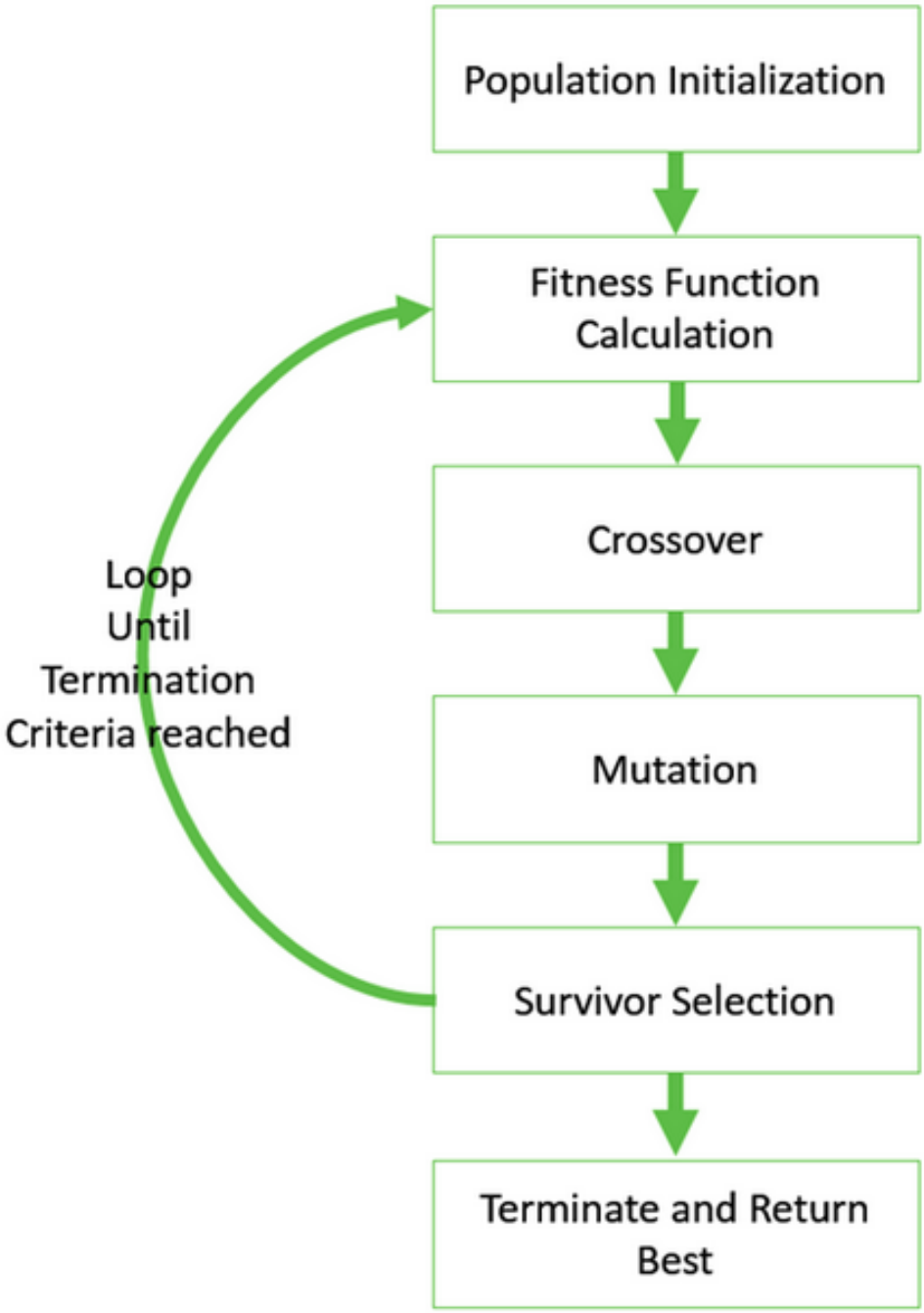
CROSSOVER

- One Point Crossover
  A random crossover point is selected and the heads of the two parentsâĂŹ gate assignment of flights are swapped to get new off-springs.
- Multi Point Crossover
  Alternating segments are swapped to get new off-springs.
- Uniform Crossover
  For each gene in a chromosome, a coin is flipped and accordingly it is swapped.

MUTATION

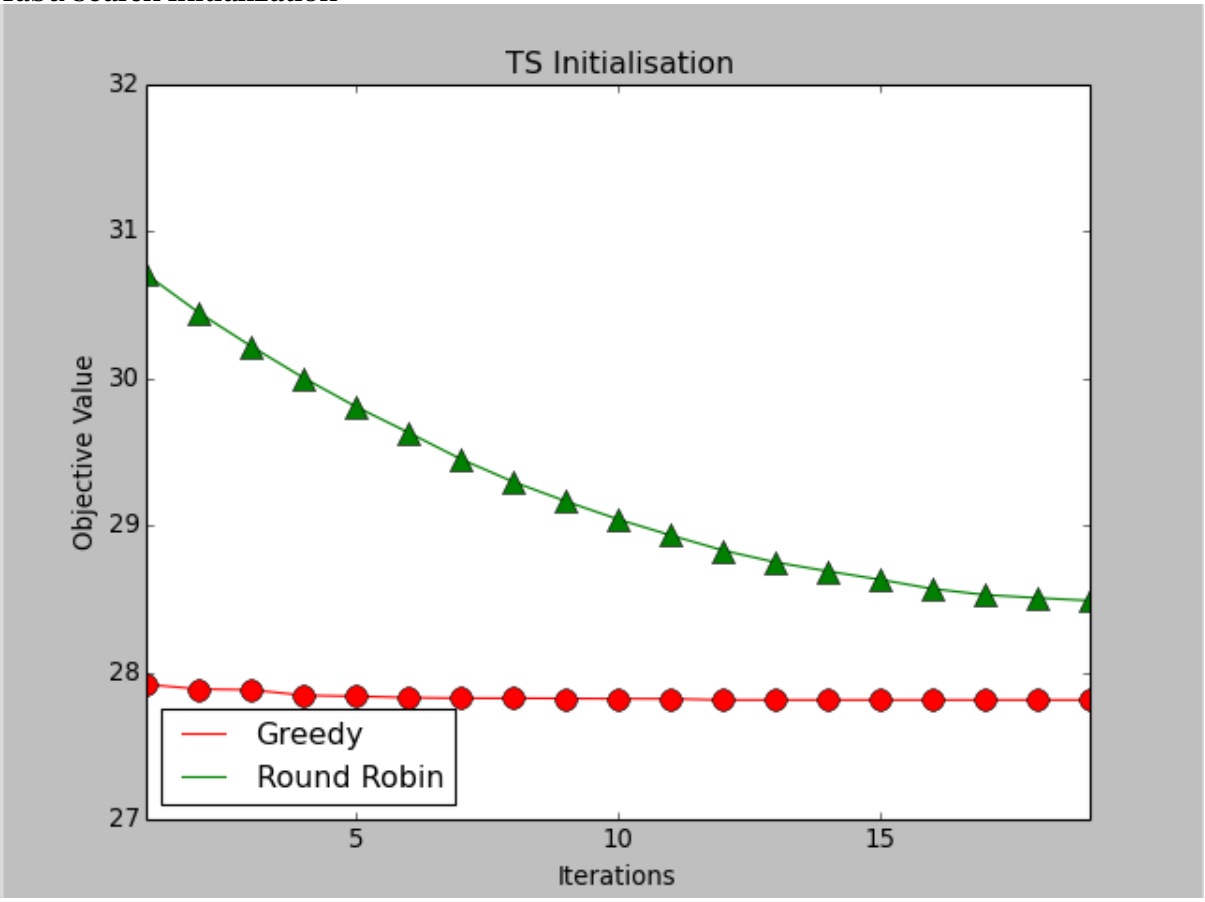
- Random Resetting Mutation
  We randomly choose one gene and assign a randomly chosen value from the set of permissible values.
- Swap Mutation
  We randomly choose two genes and interchange the gate assignment.

The combination of uniform crossover and random resetting mutation was chosen.
Complexity is given by $O(gn^2m)$ where :
g : no. of generations/iterations
n : population size
m : size of individuals

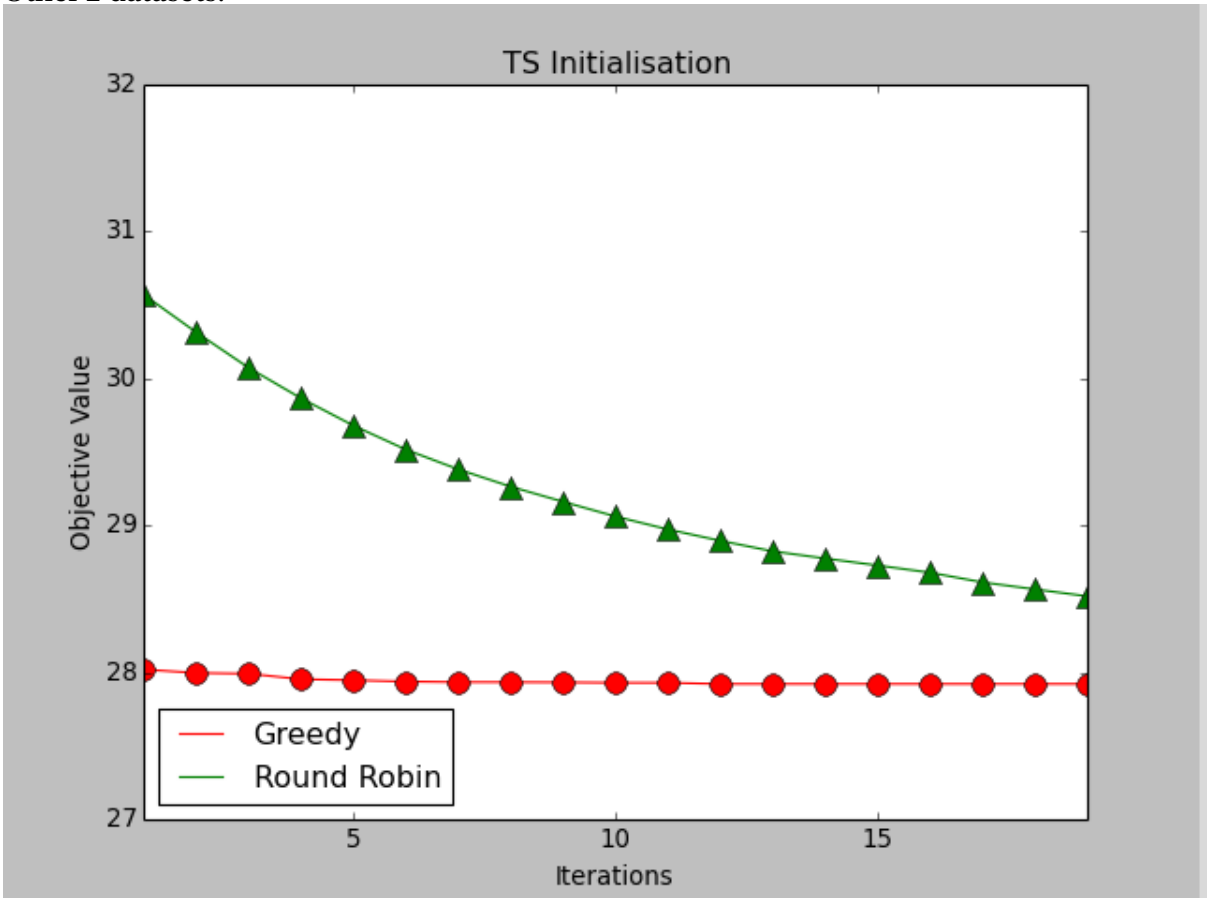

Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached

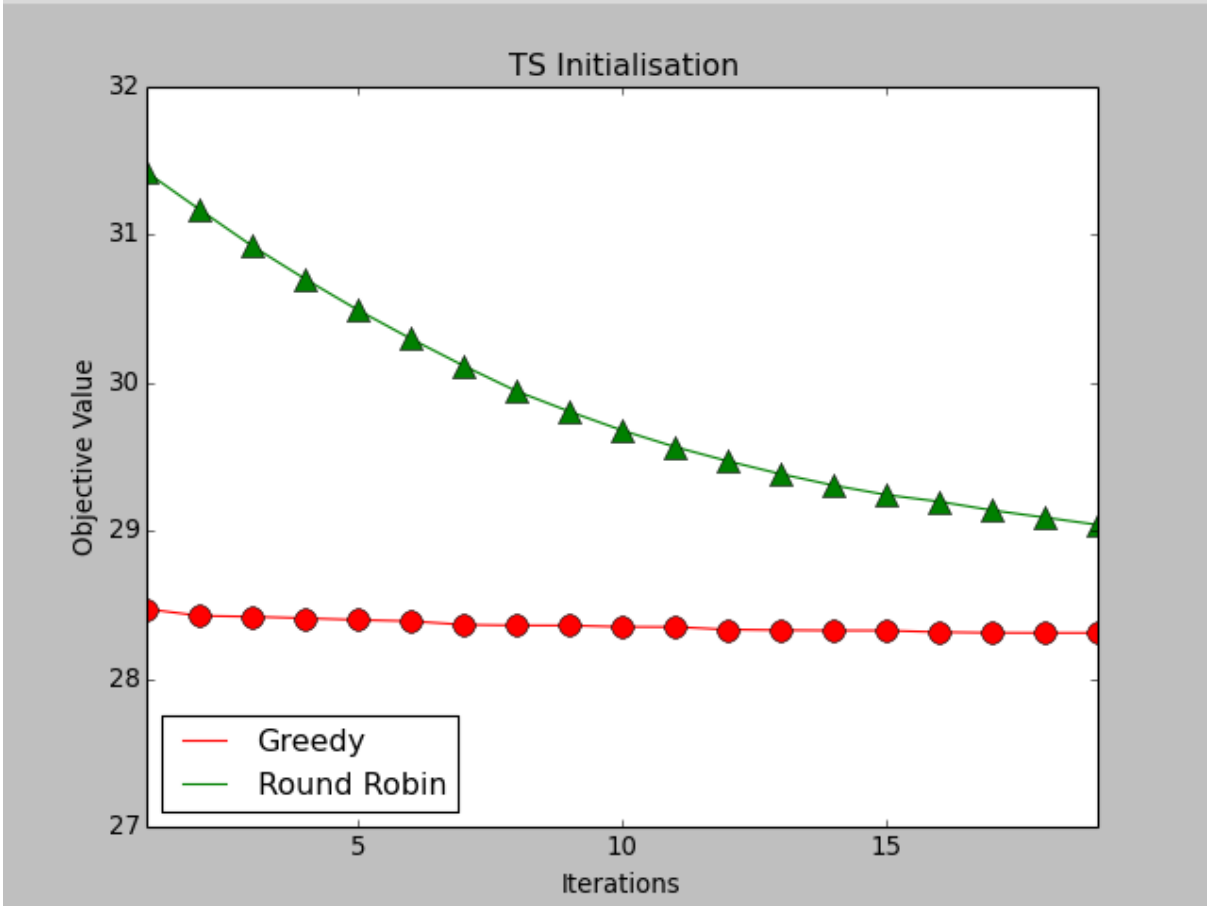
RESULTS

Tabu Search Initialization

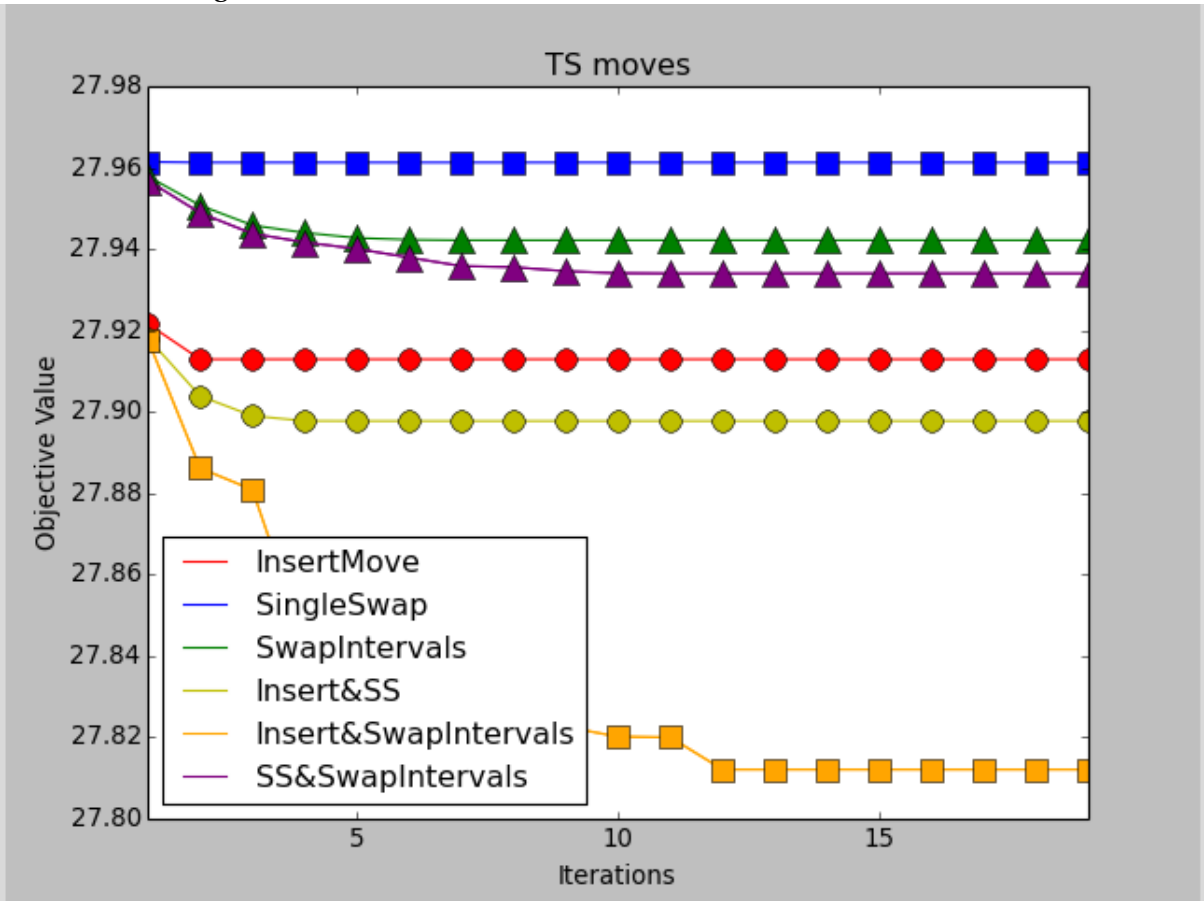


Other 2 datasets:

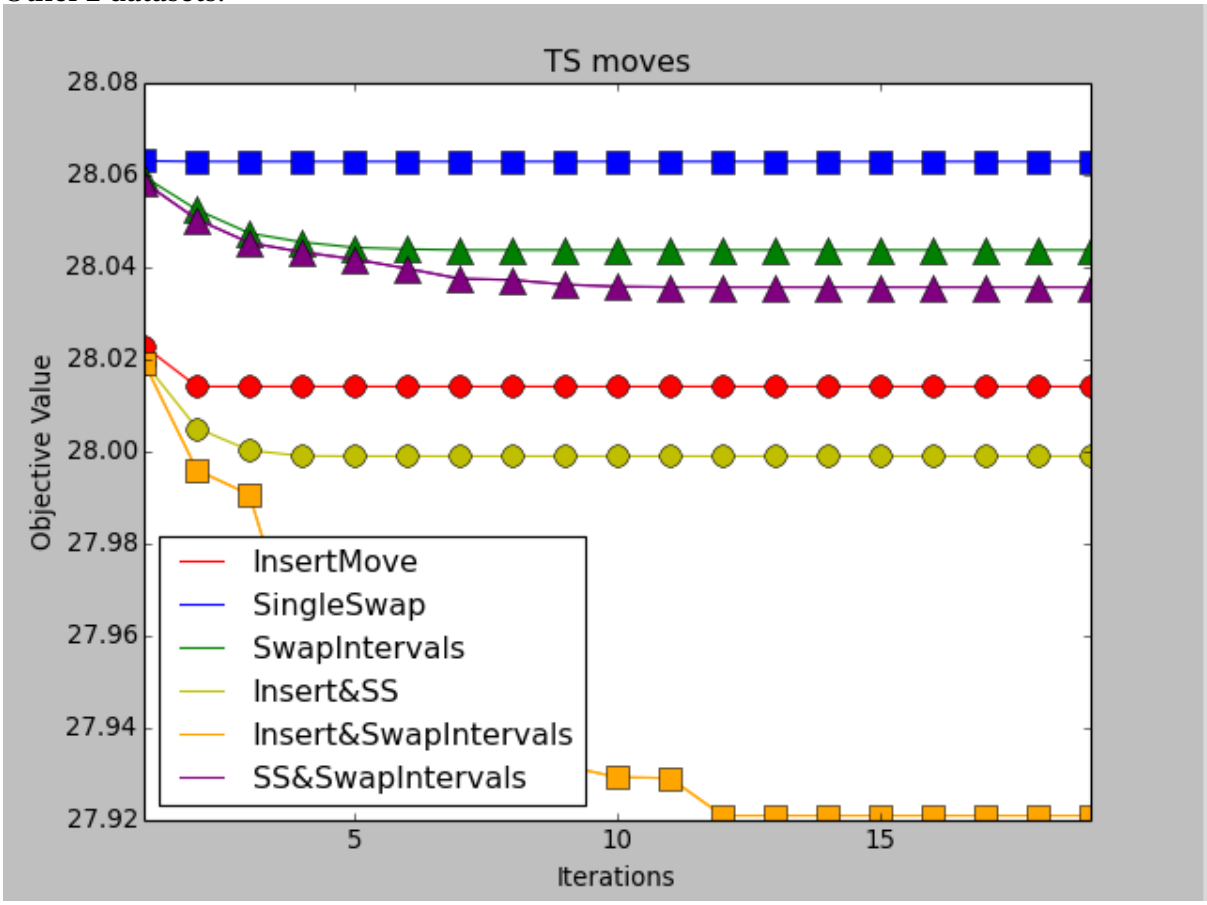


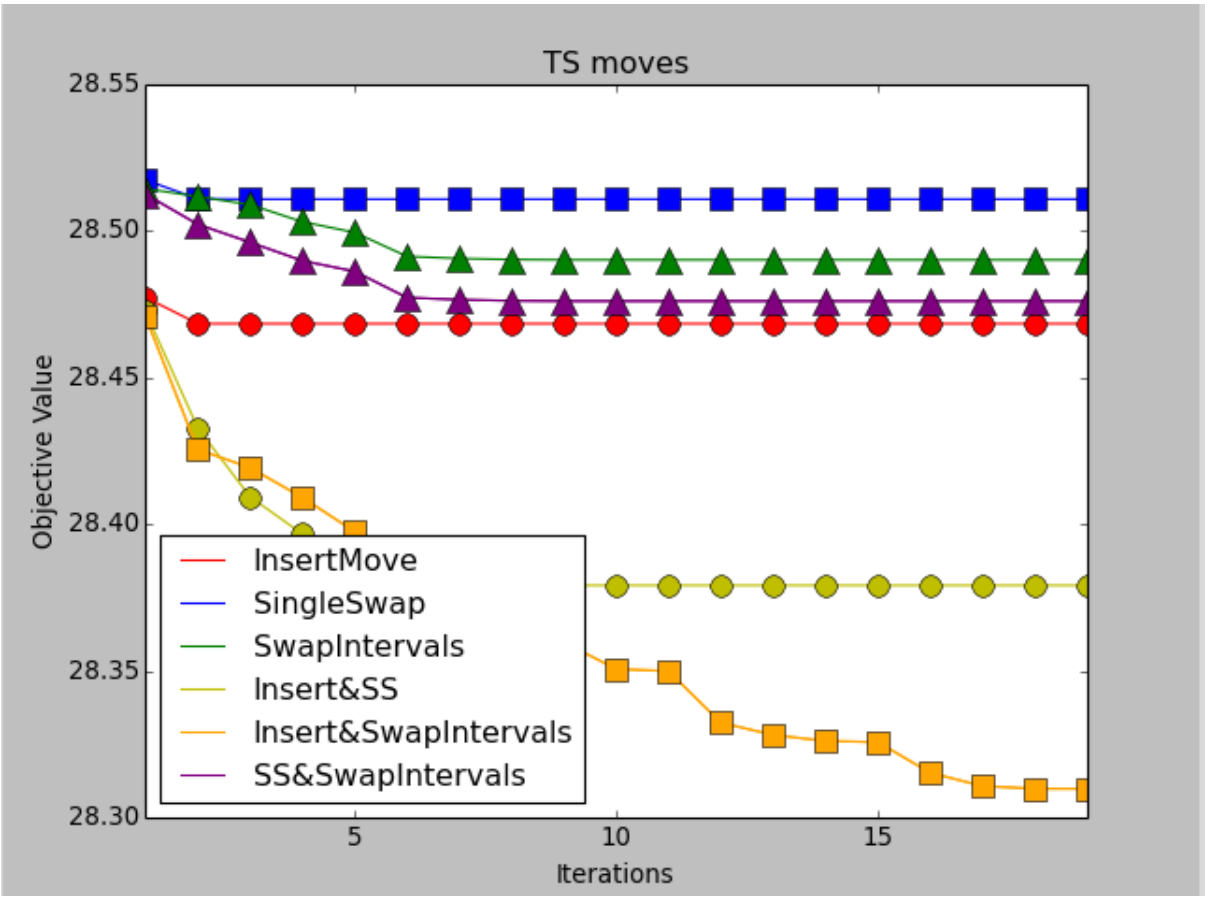

TS Initialisation
Objective Value
Iterations
32
31
30
29
28
27
5
10
15
Greedy
Round Robin

Tabu Search Neighborhood Moves:

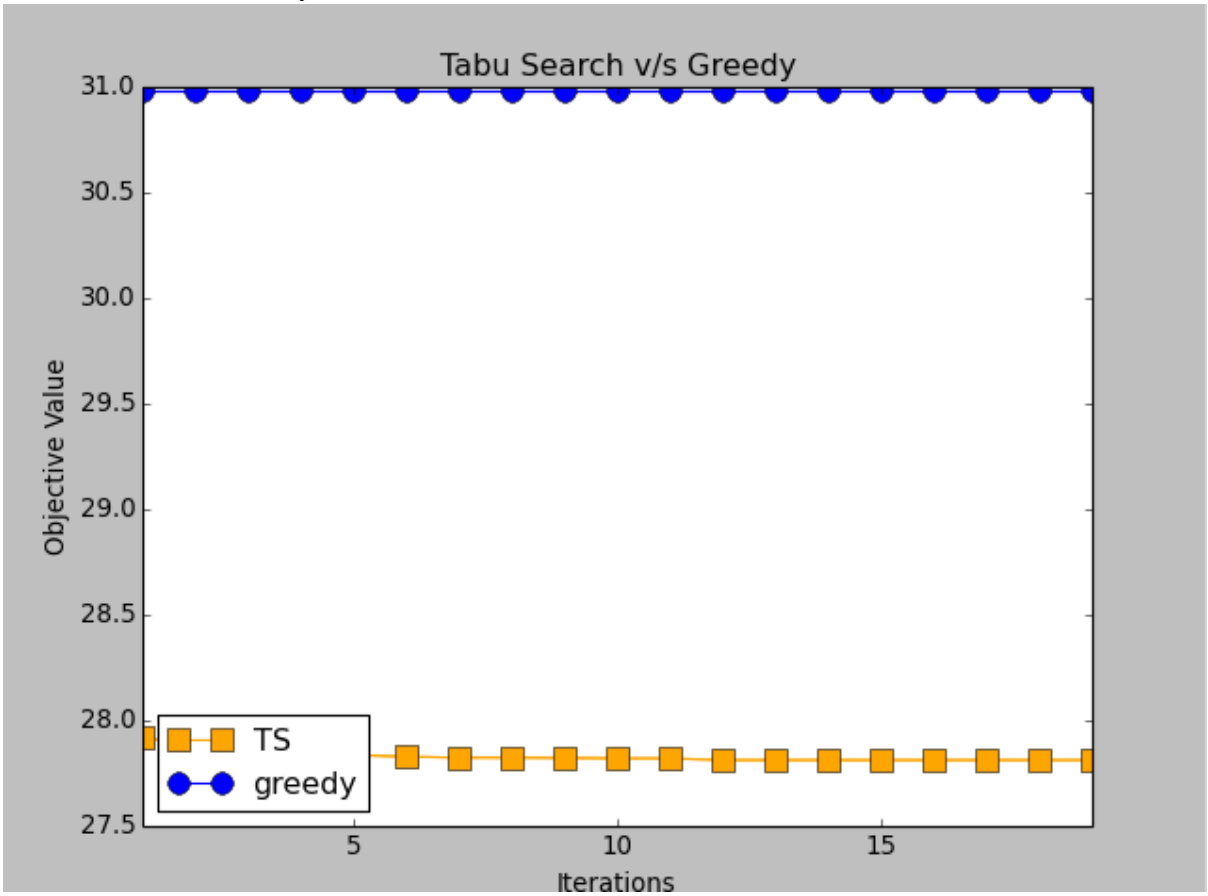


Other 2 datasets:

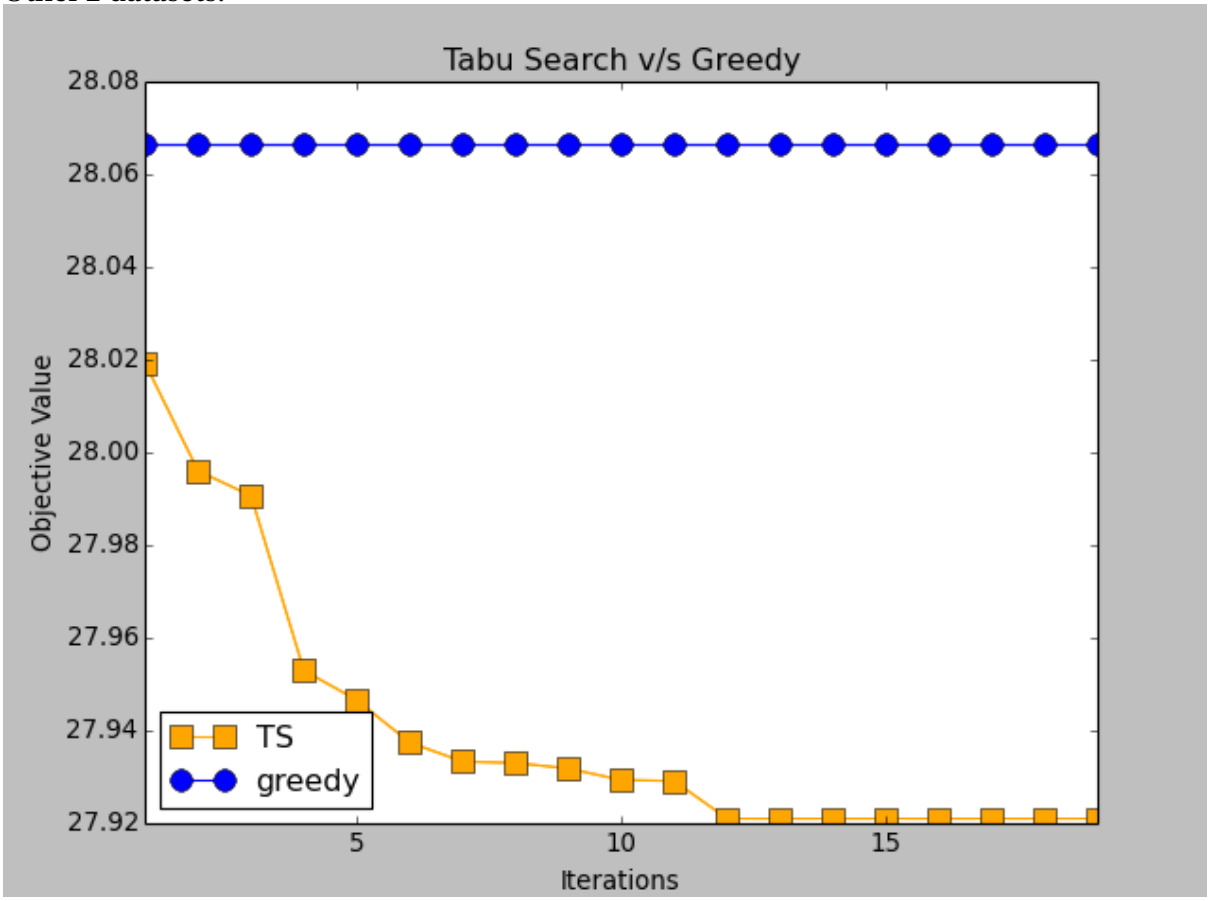


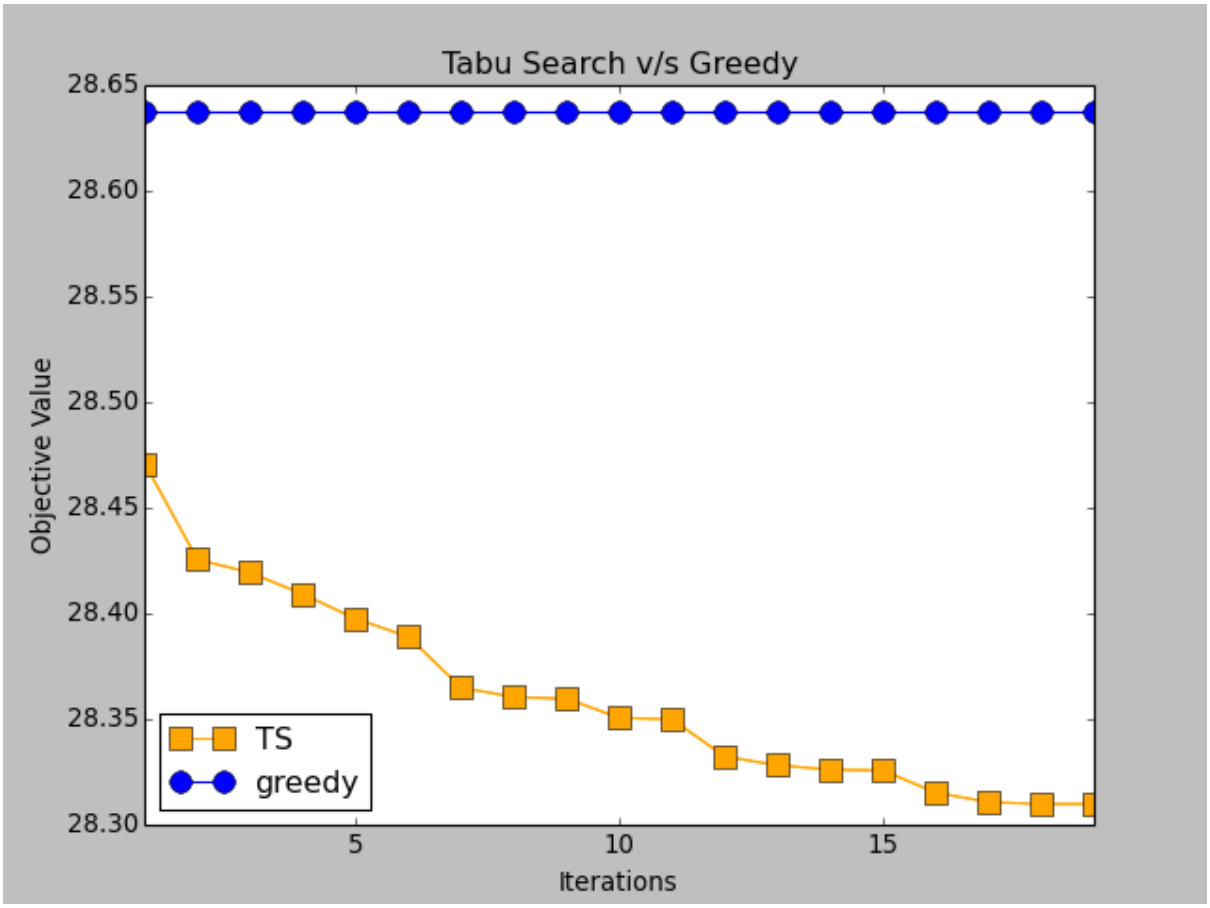

TS moves
Objective Value
Iterations
28.55
28.50
28.45
28.40
28.35
28.30
5
10
15
InsertMove
SingleSwap
SwapIntervals
Insert&SS
Insert&SwapIntervals
SS&SwapIntervals

Tabu Search v/s Greedy

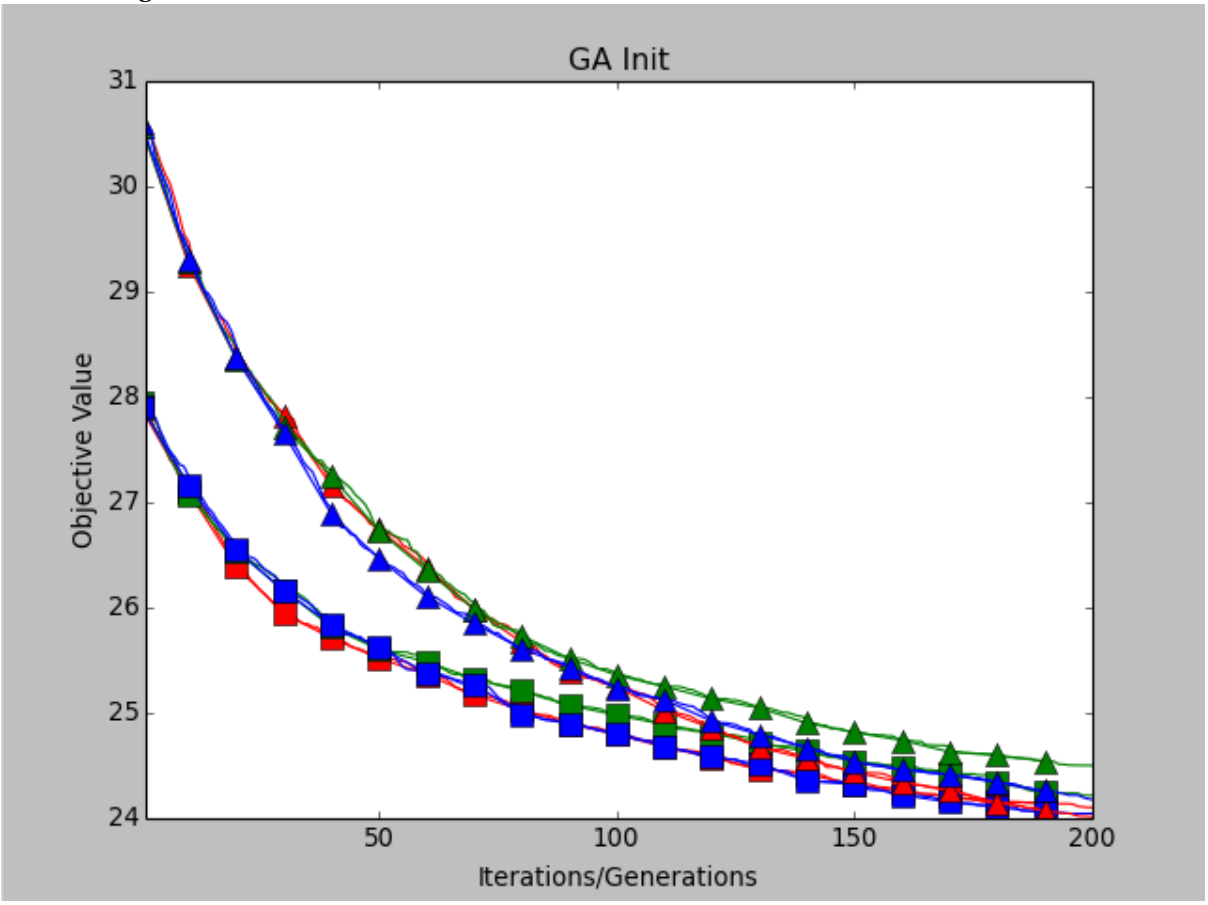


Other 2 datasets:

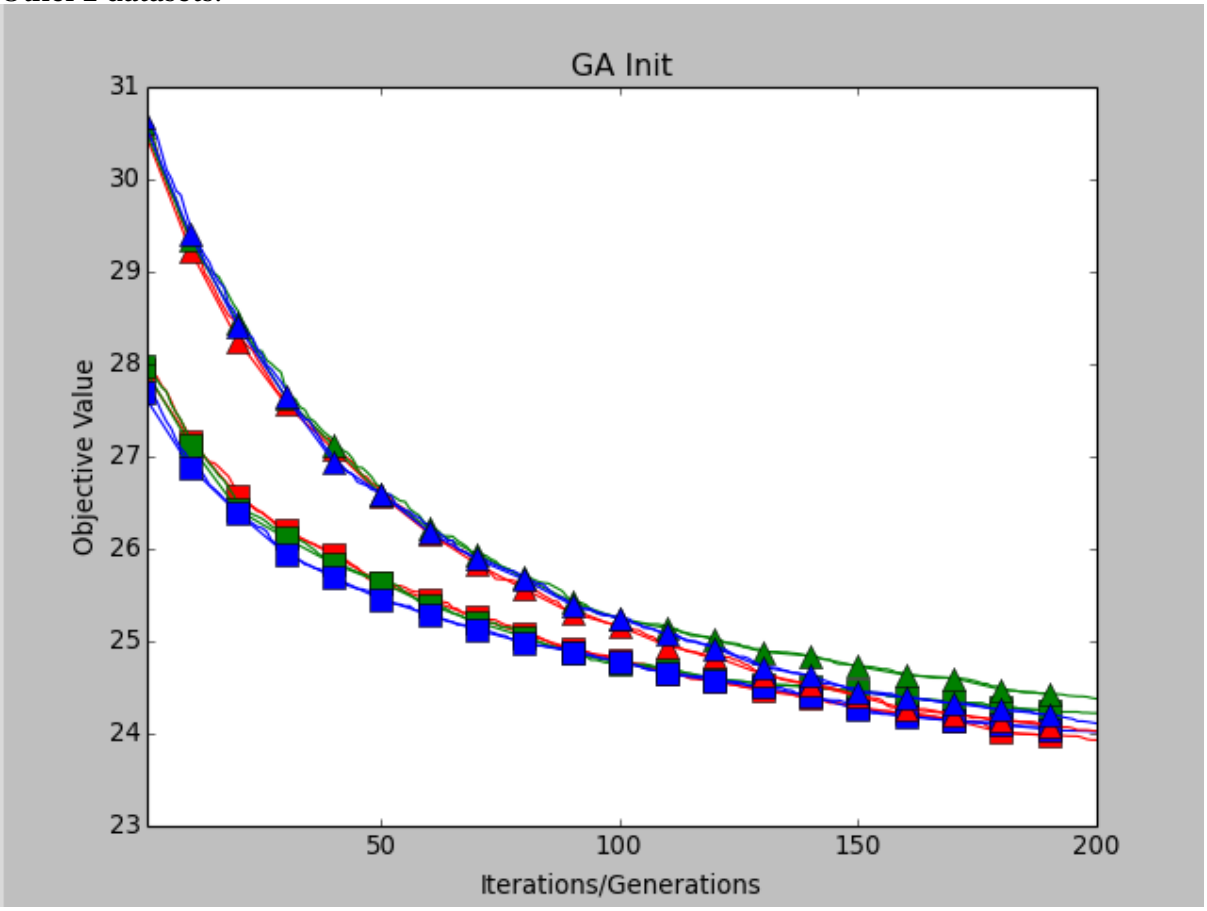


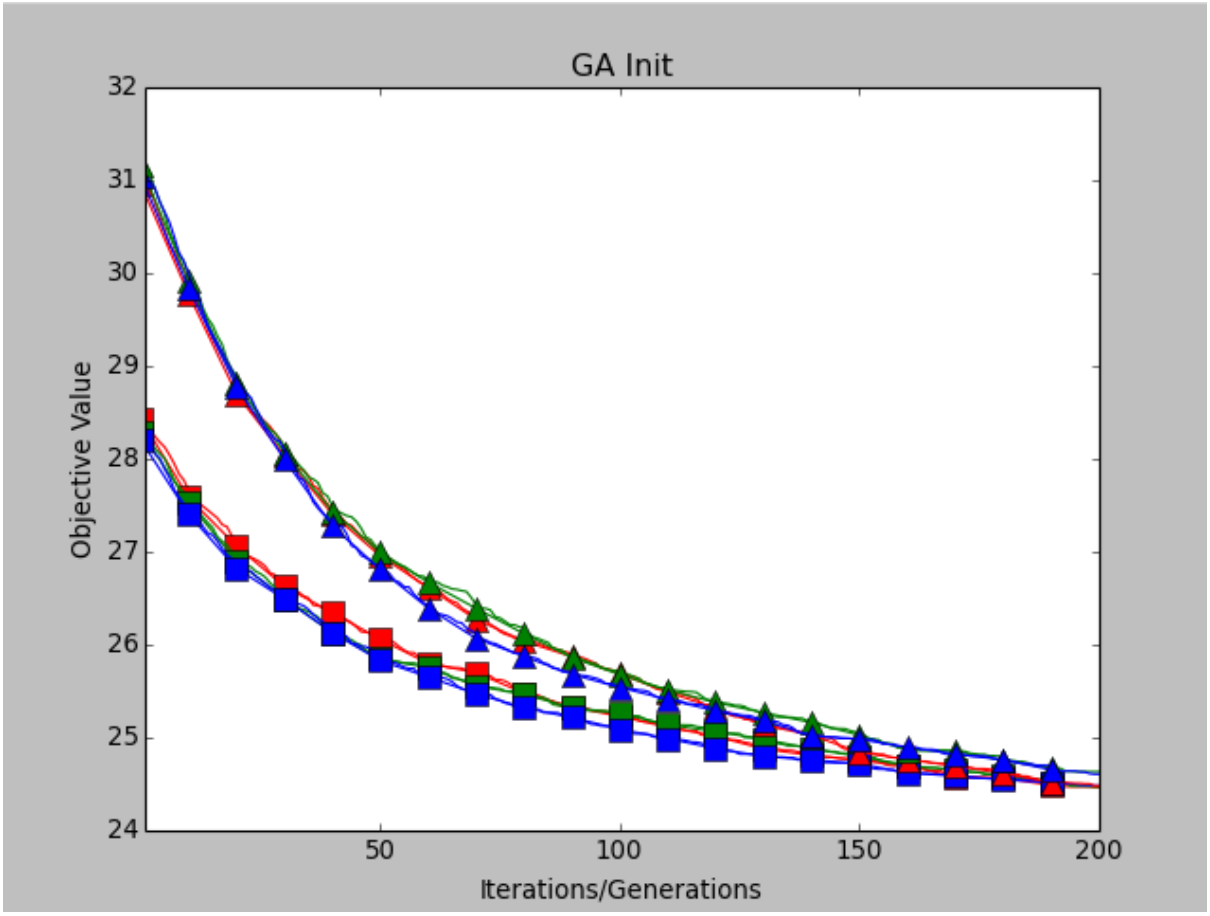

Tabu Search v/s Greedy
28.65
28.60
28.55
28.50
28.45
28.40
28.35
28.30
Objective Value
5
10
15
Iterations
TS
greedy

Genetic Algorithm Initialization:

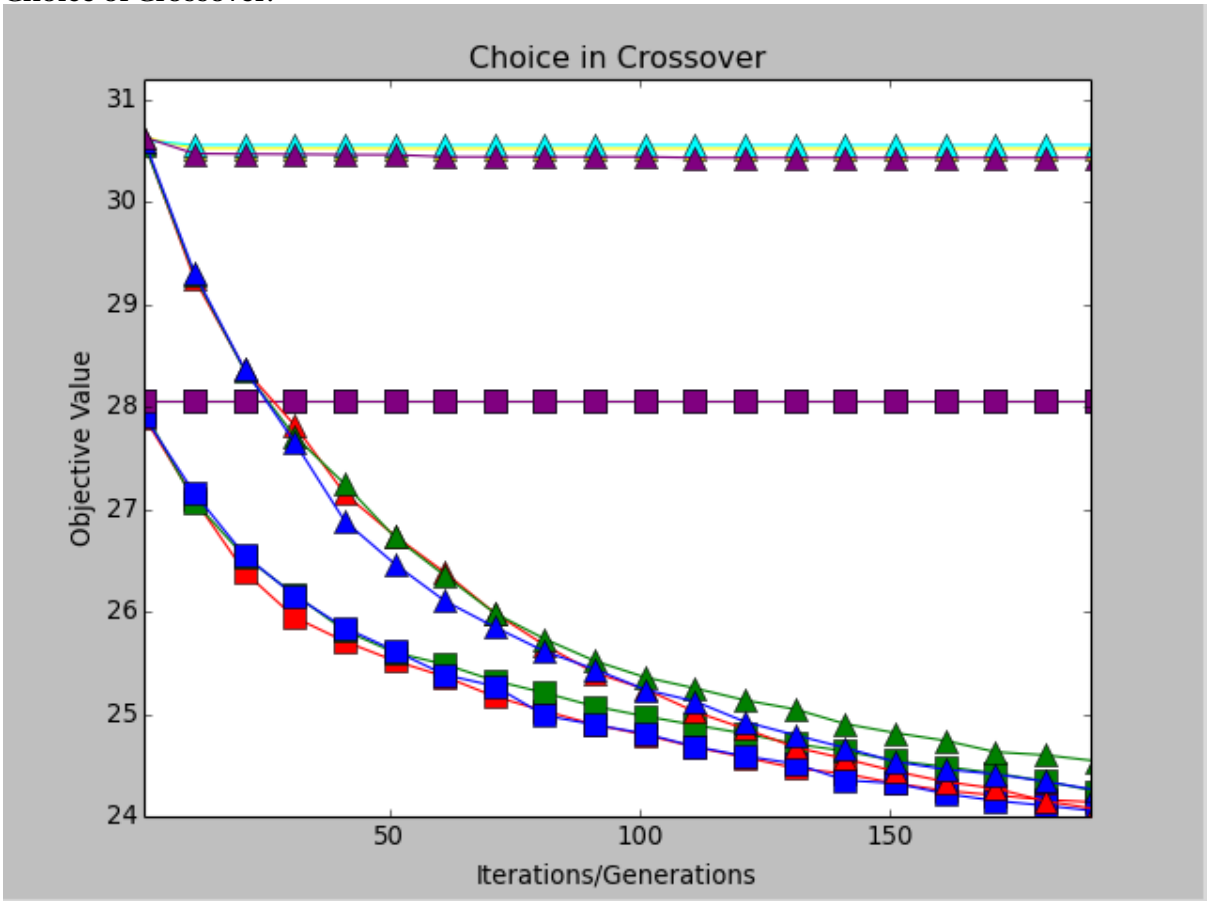


Other 2 datasets:

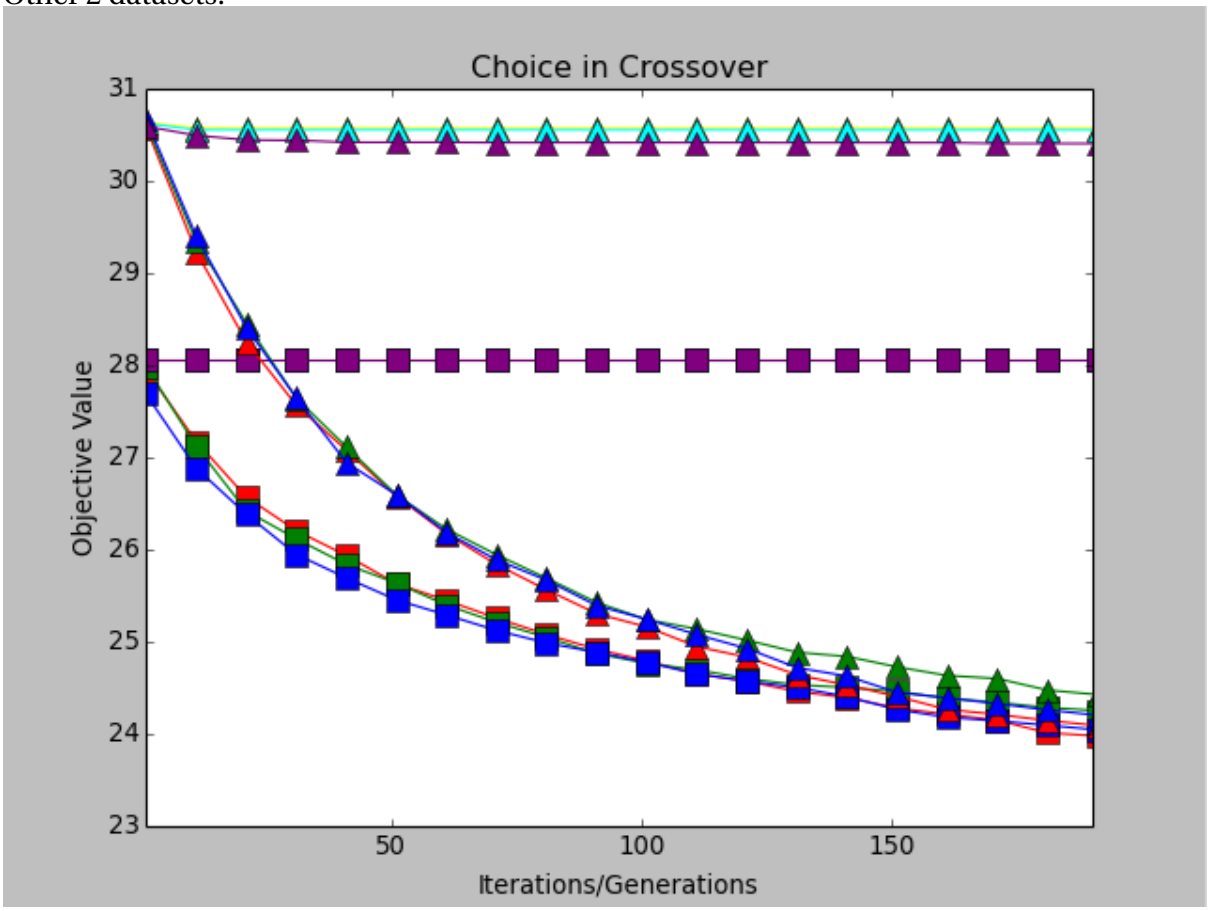


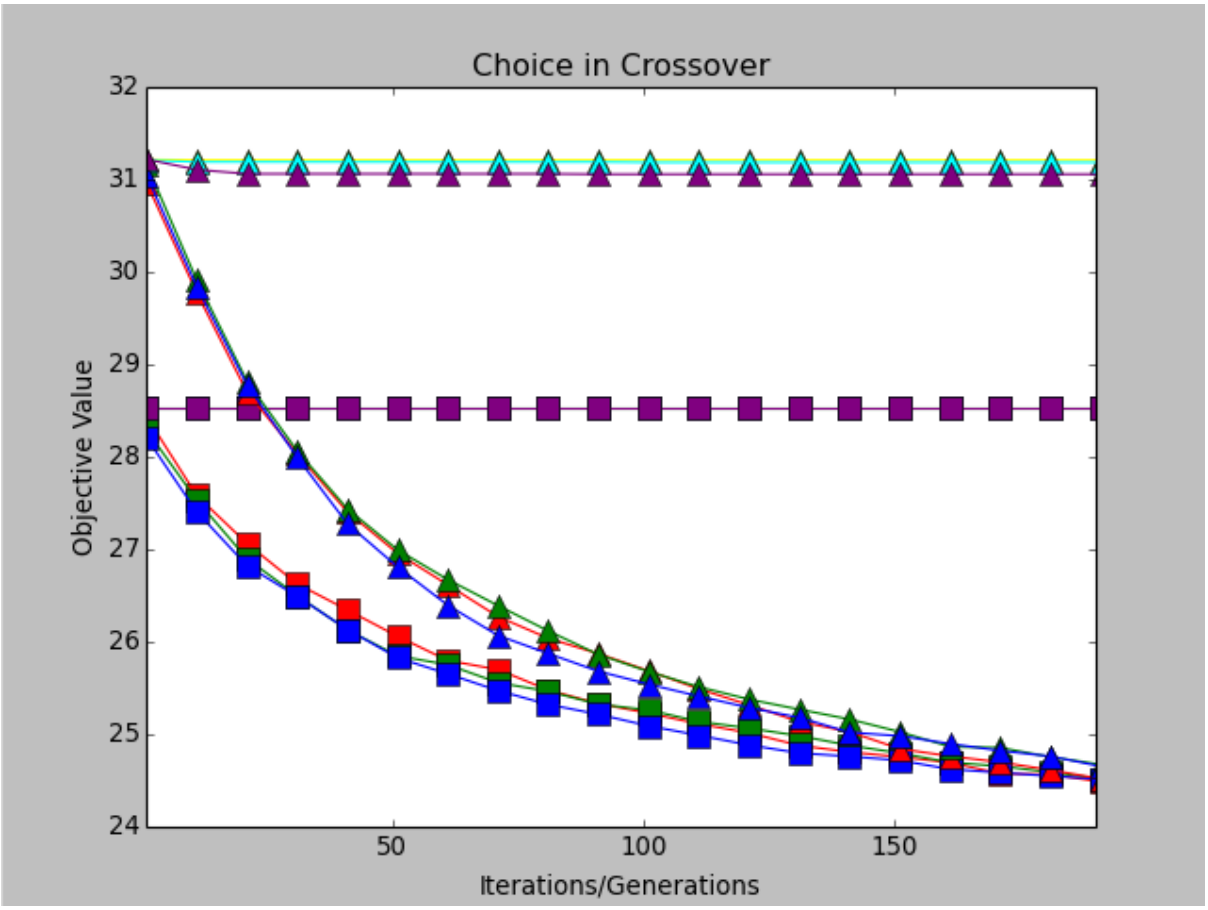

GA Init
Objective Value
Iterations/Generations
32
31
30
29
28
27
26
25
24
50
100
150
200

Choice of Crossover:

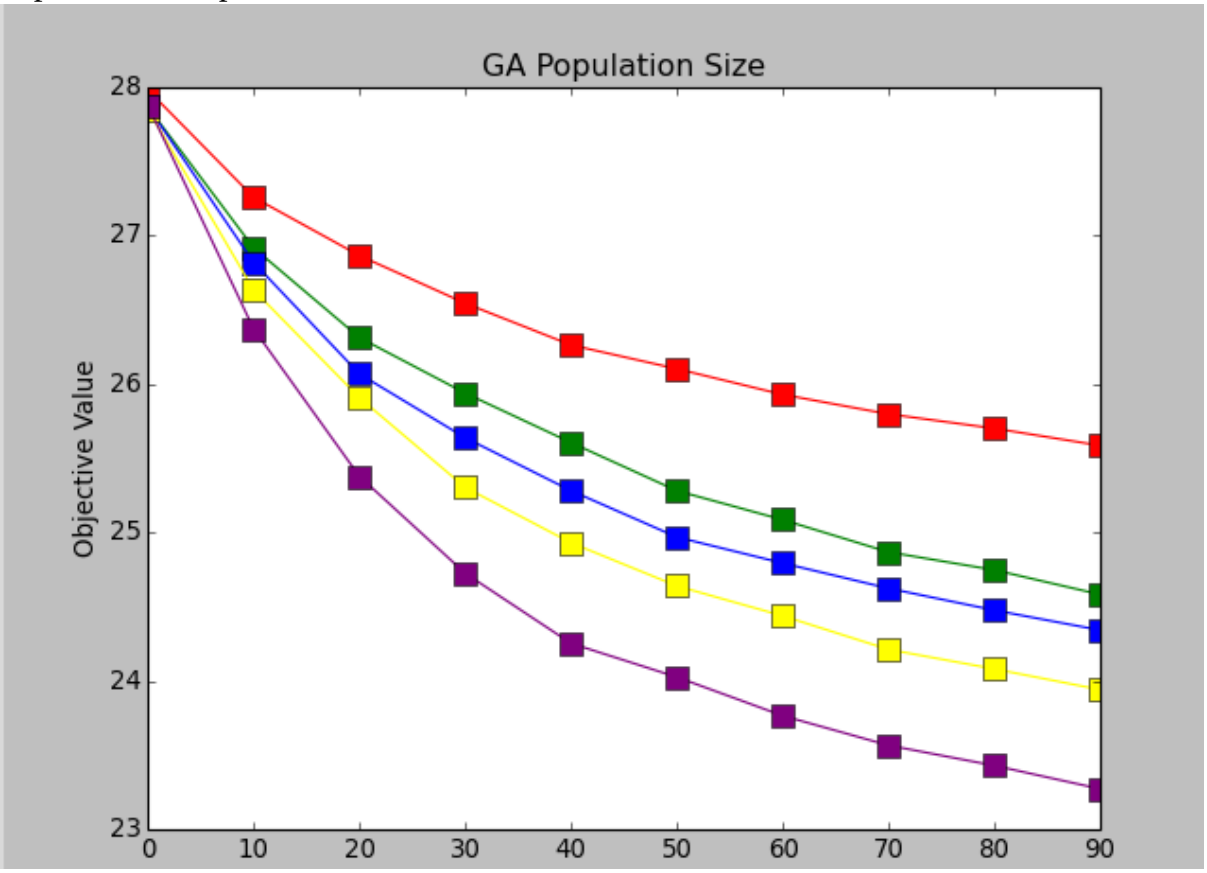


Other 2 datasets:

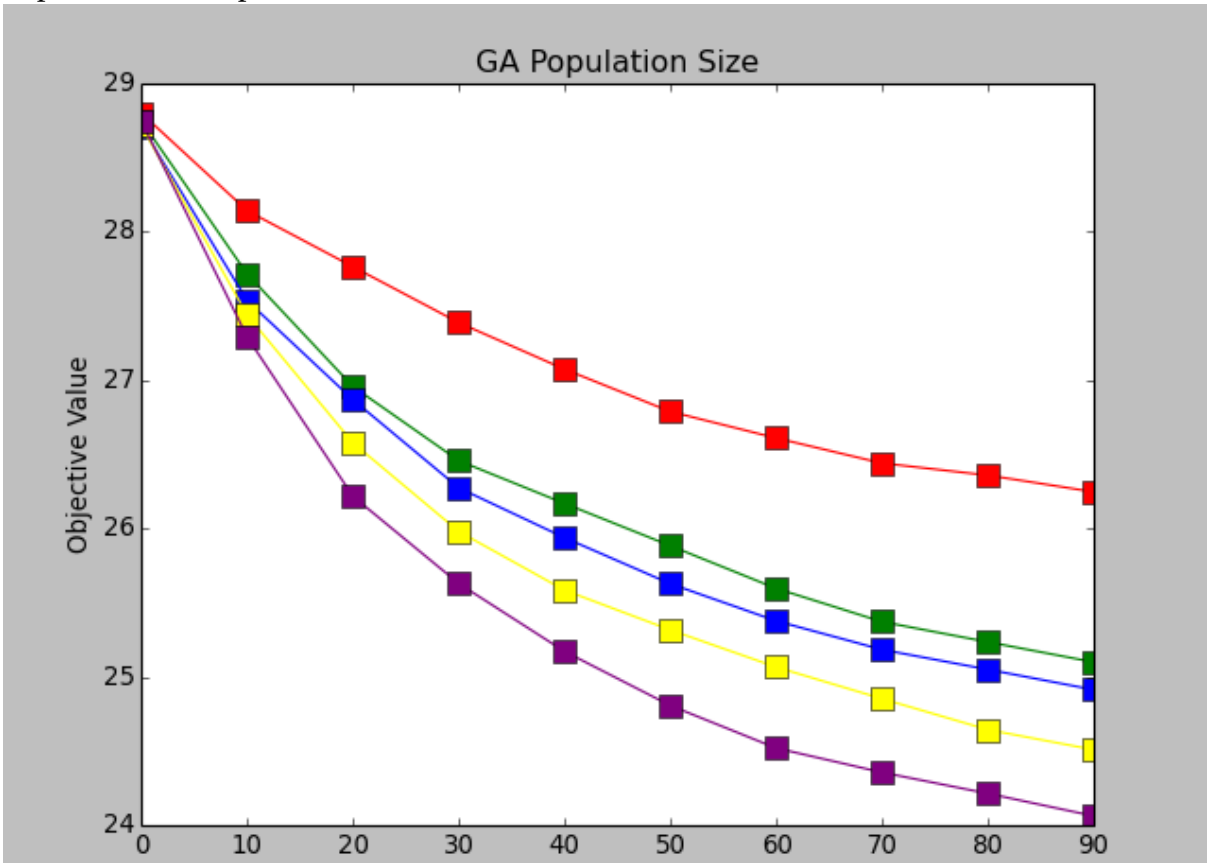



Choice in Crossover
Objective Value
Iterations/Generations
32
31
30
29
28
27
26
25
24
50
100
150

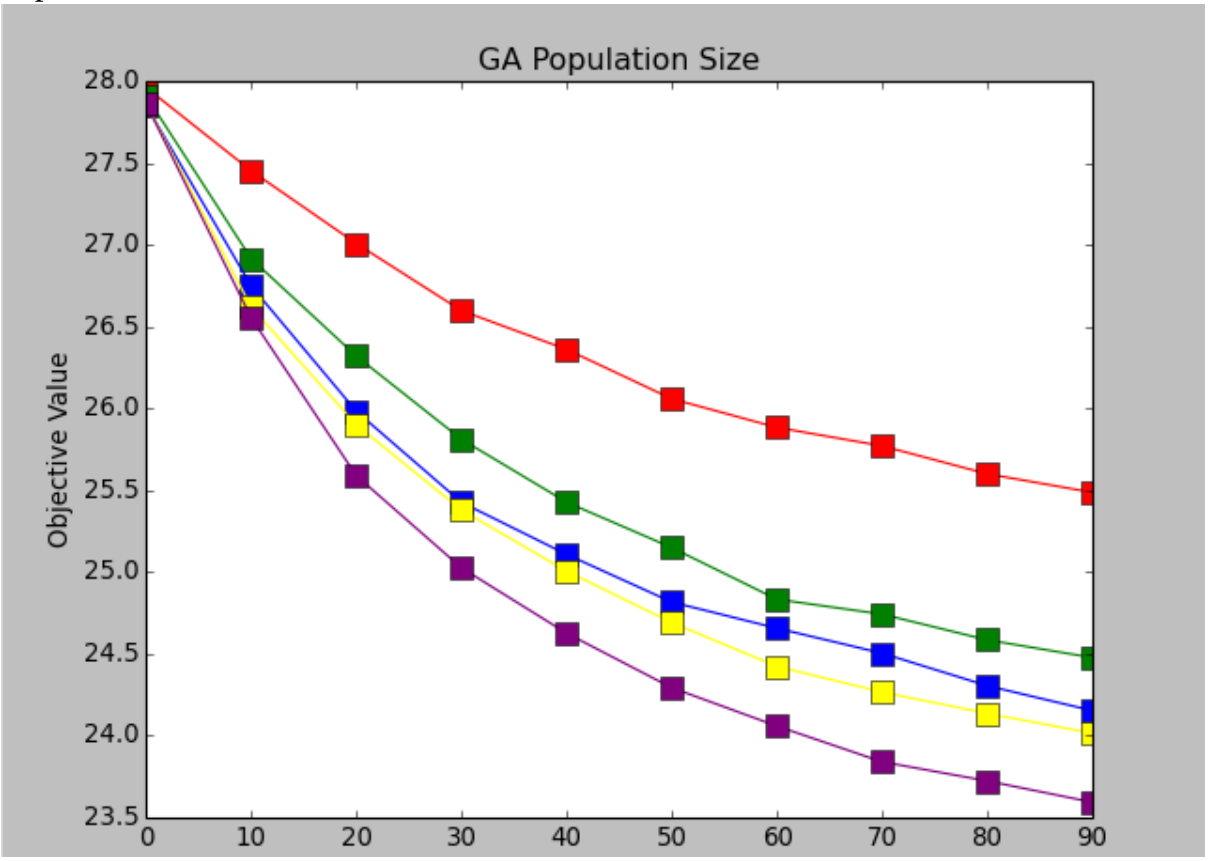
Population (Onepoint):



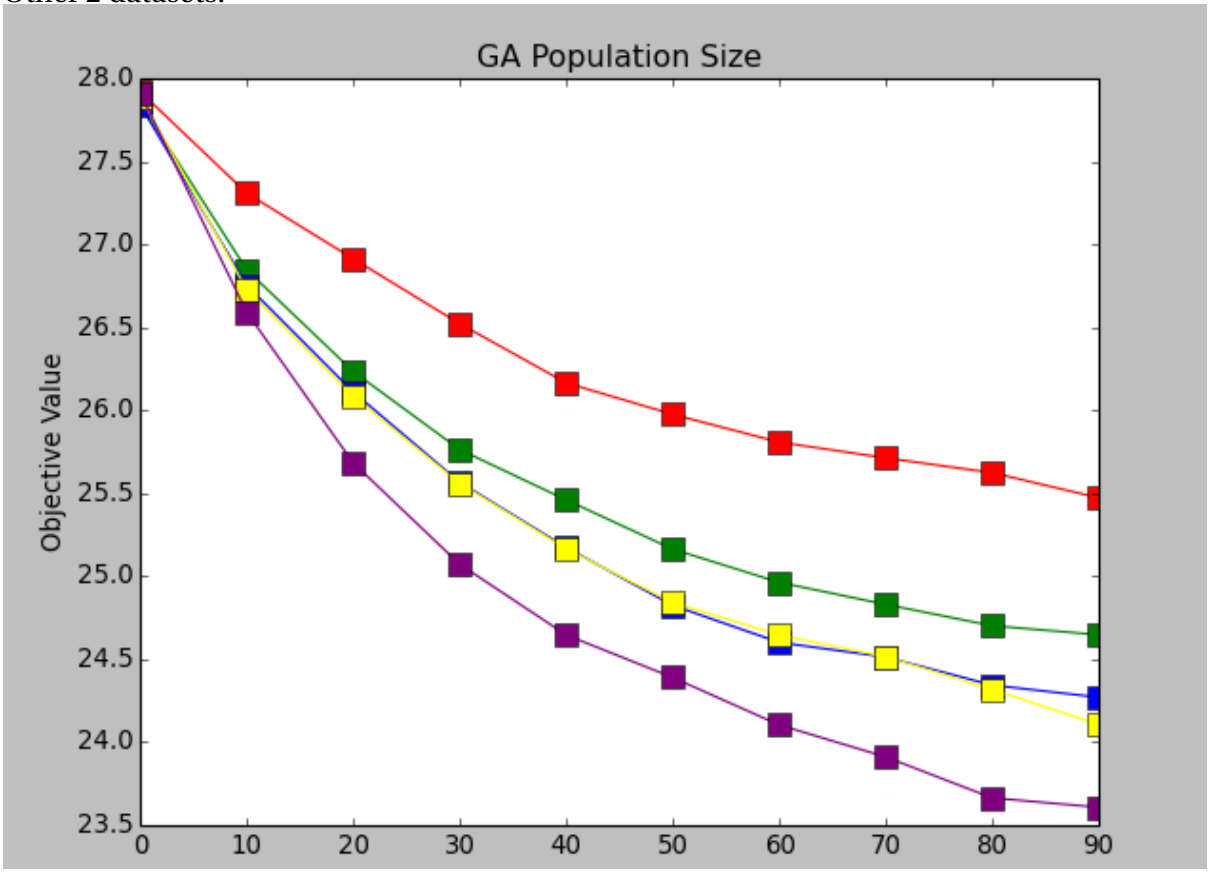
Population (Multipoint):

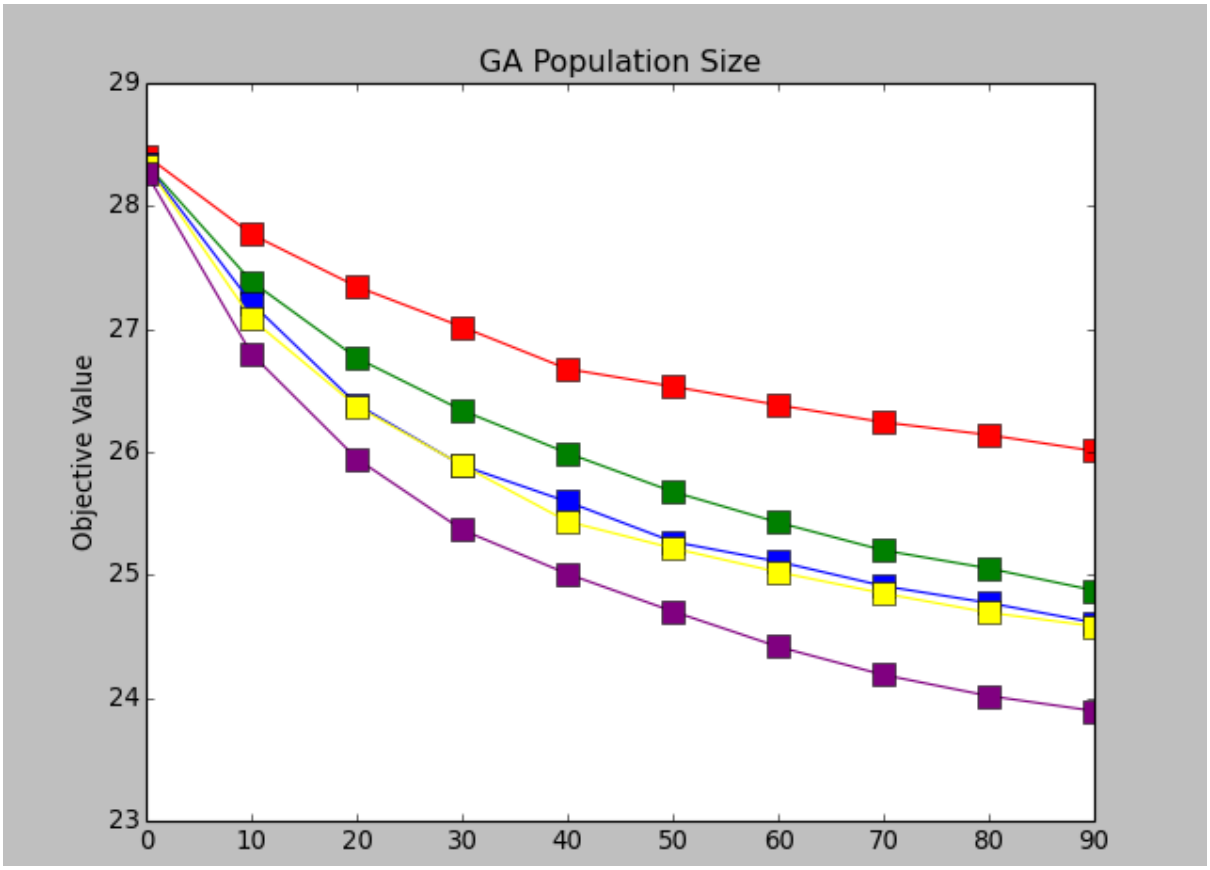


Population (Uniform):

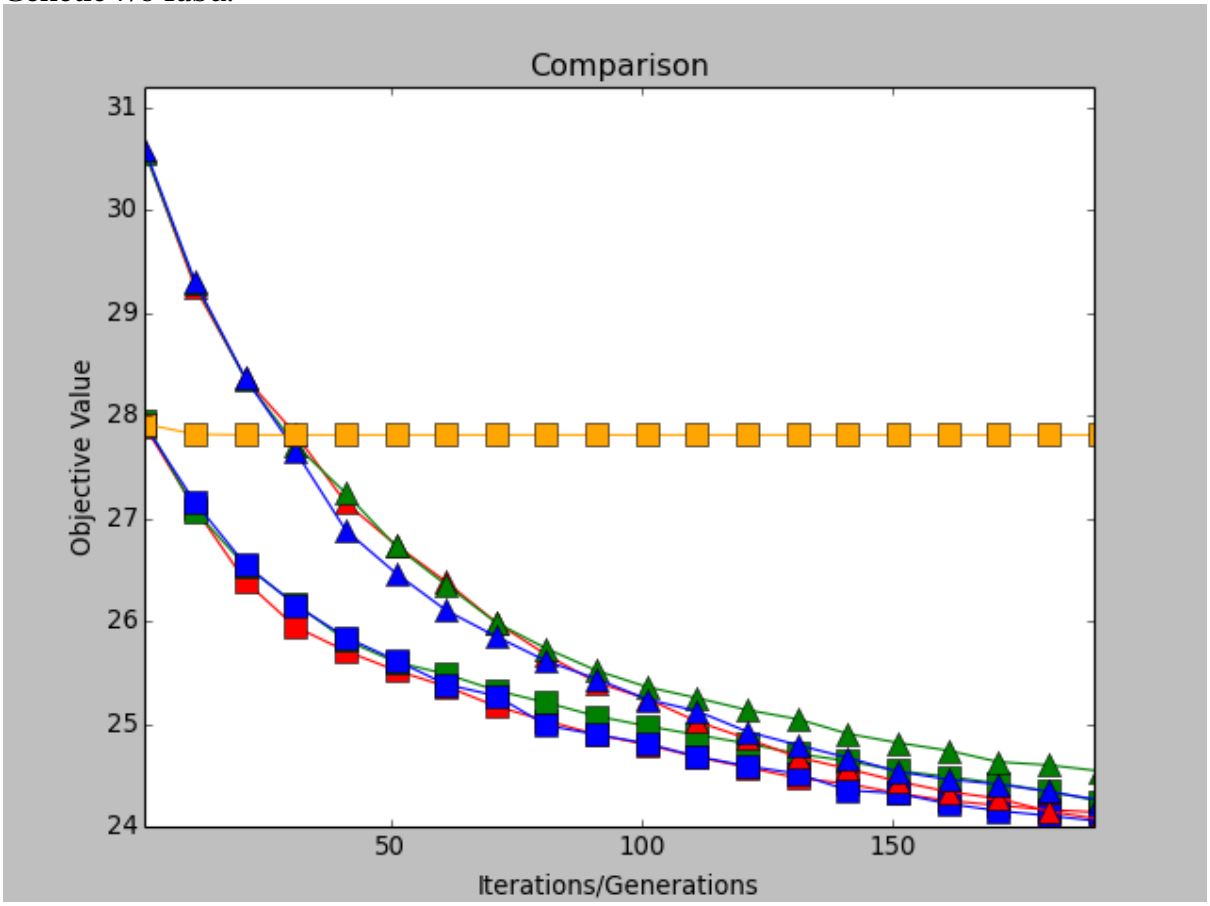


Other 2 datasets:

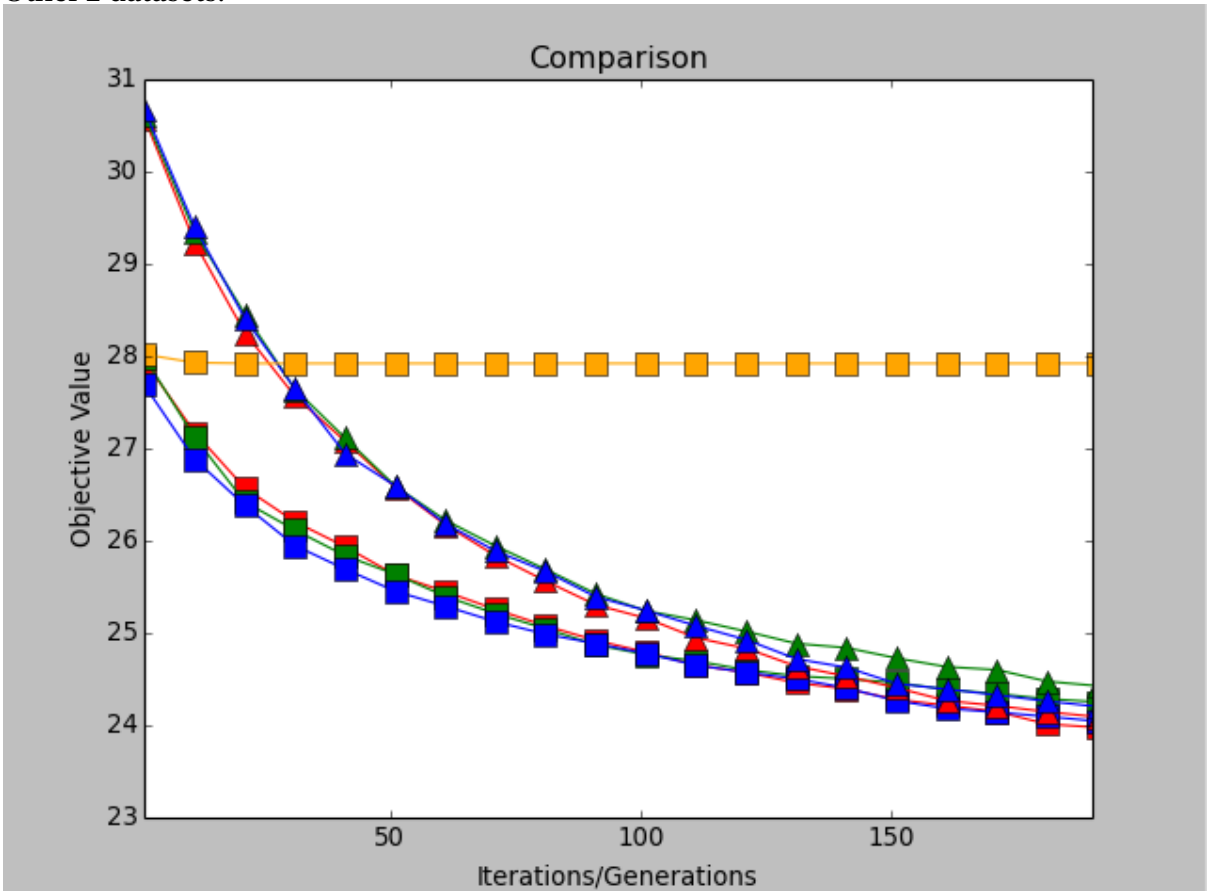


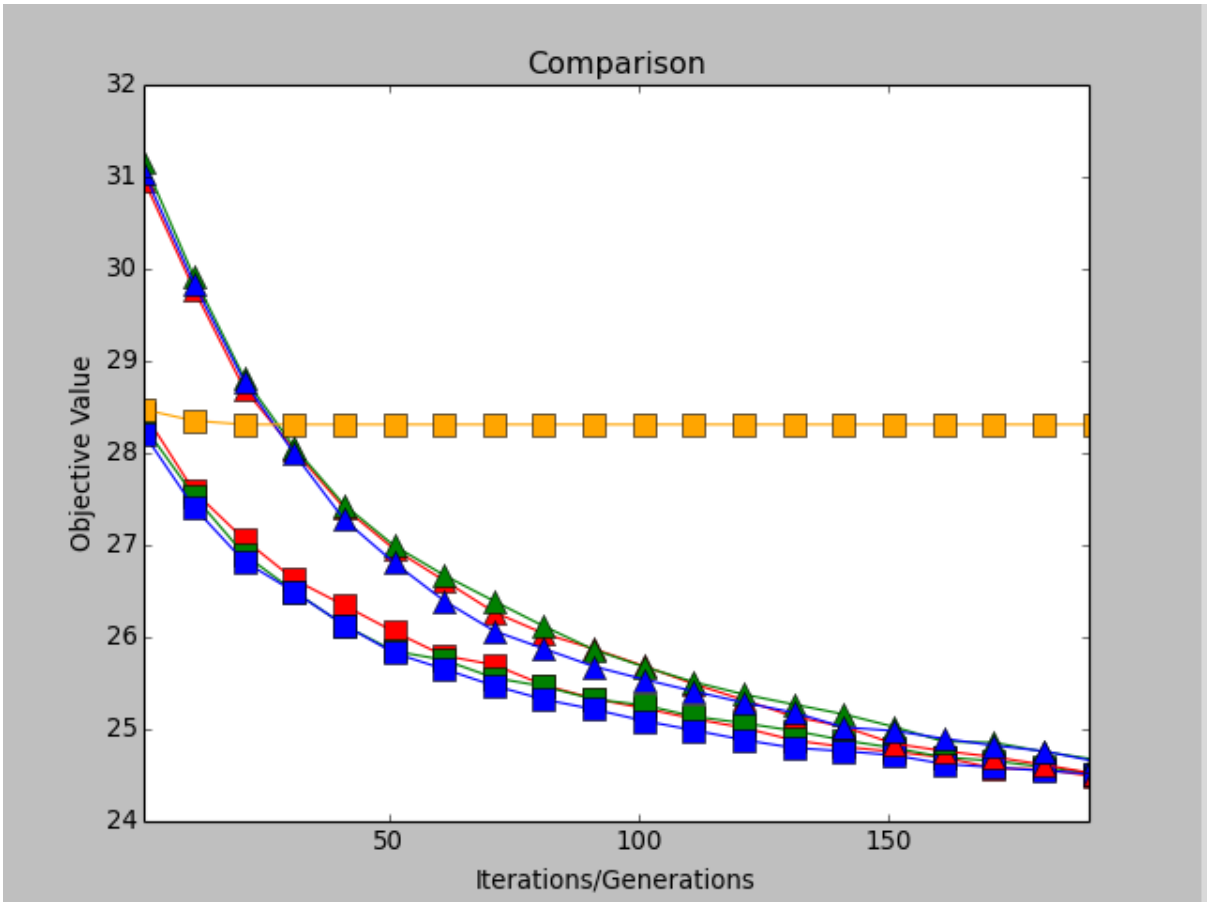

GA Population Size
Objective Value
29
28
27
26
25
24
23
0
10
20
30
40
50
60
70
80
90

Genetic v/s Tabu:

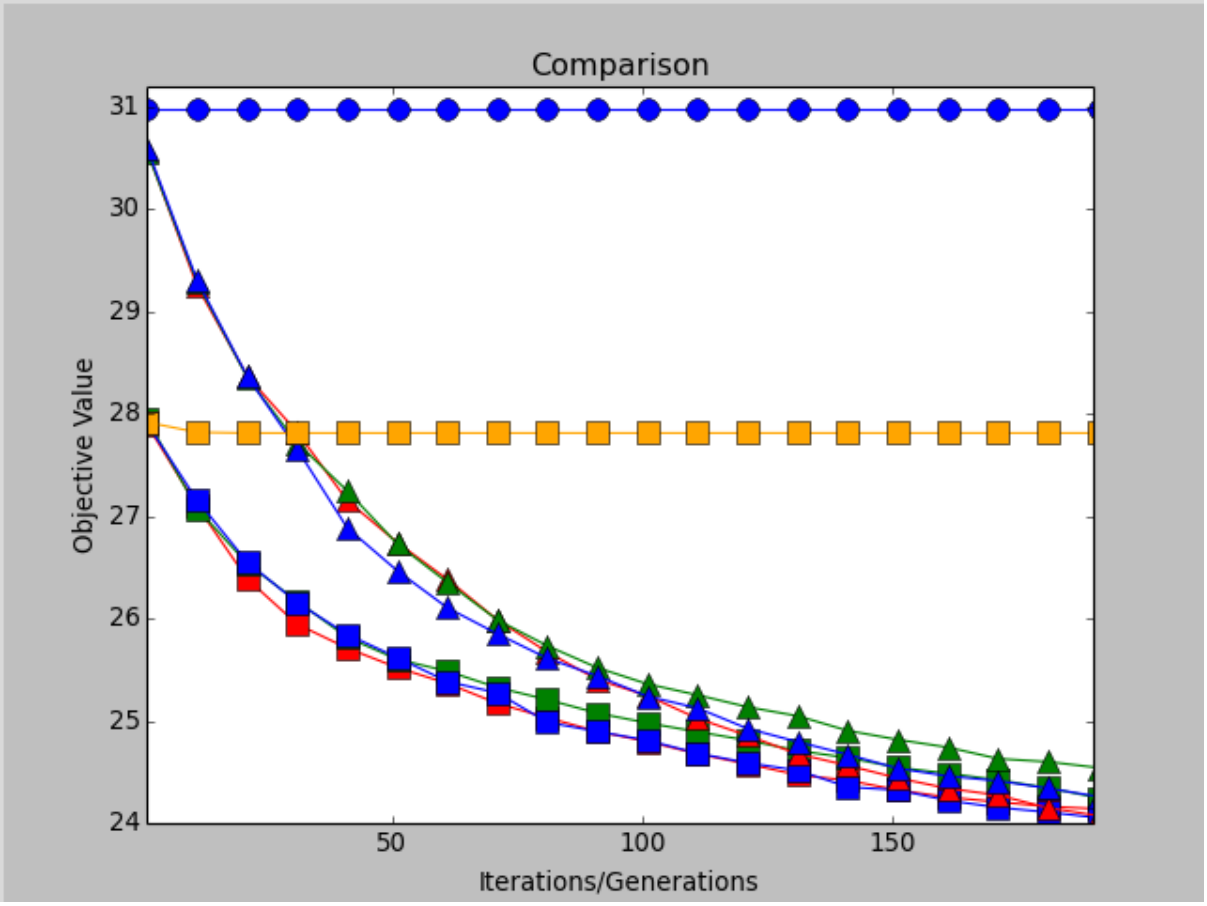


Other 2 datasets:

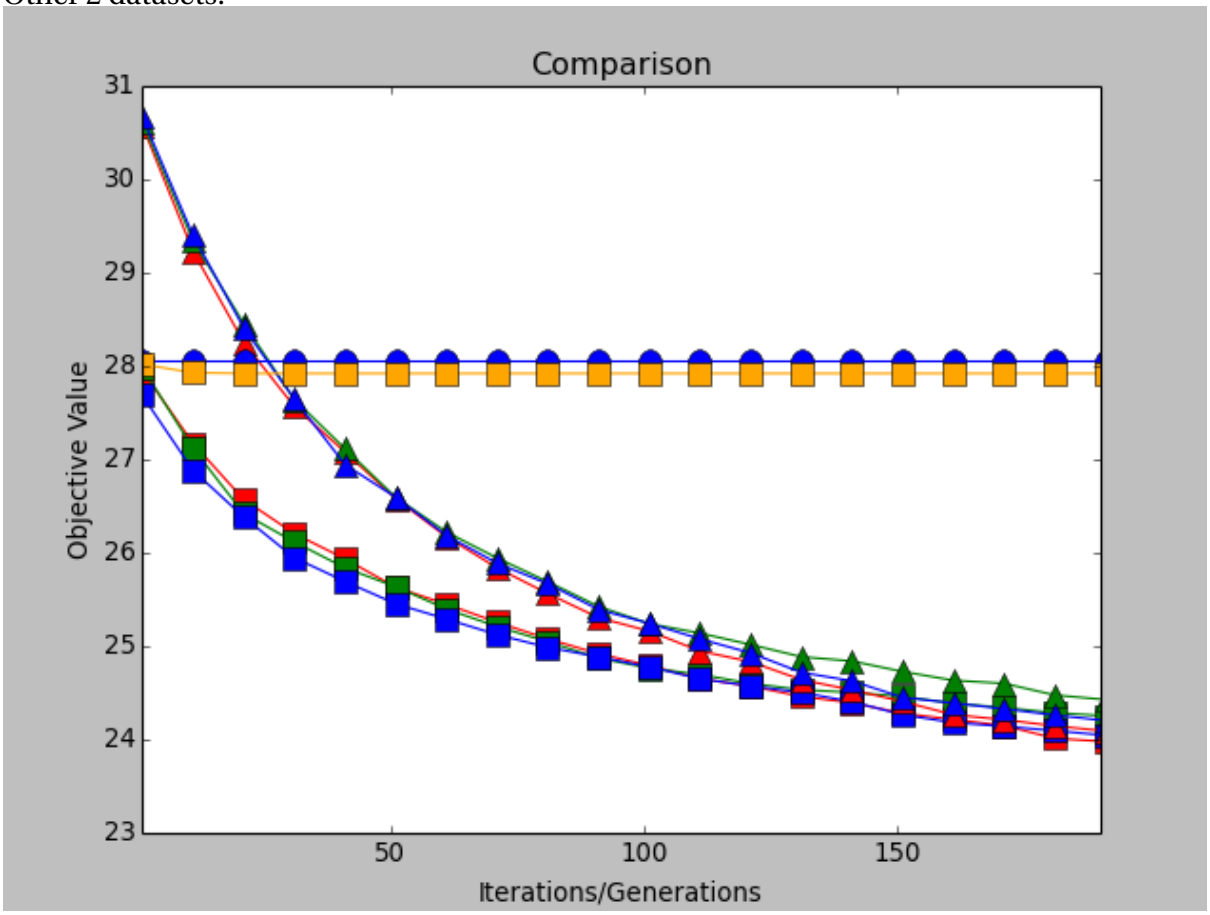


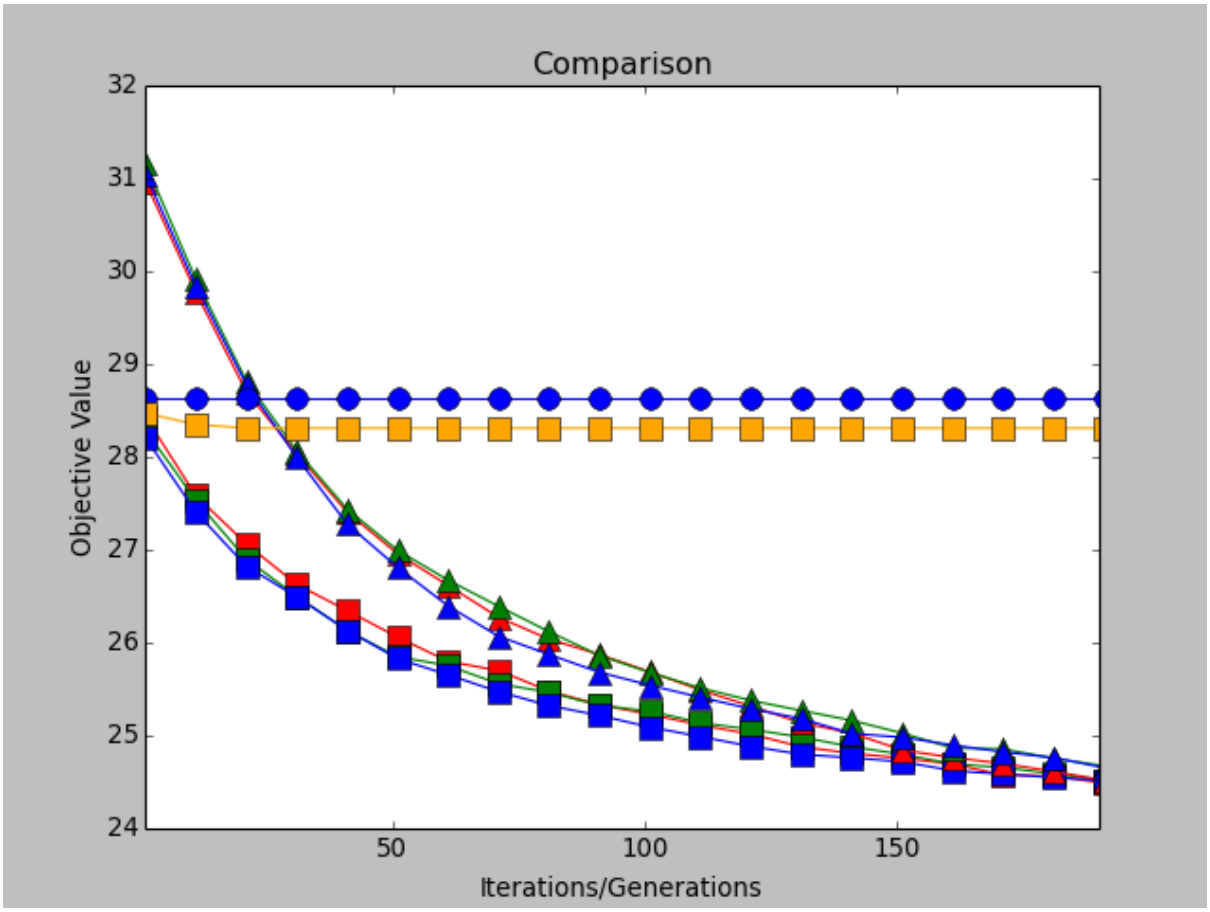

Comparison
Objective Value
Iterations/Generations
32
31
30
29
28
27
26
25
24
50
100
150

Genetic v/s Tabu v/s Greedy:



Other 2 datasets:




Comparison
Objective Value
Iterations/Generations
32
31
30
29
28
27
26
25
24
50
100
150