AI Laboratory

Bicing

Project Members:

Anshuman Mehta

Andrea Gritti

Matteo de Martino

1 Contents

2 The Bicing Problem

2.1 Identification And Description of The Problem

Bicing is a service that lends bicycles to use for small displacements inside Barcelona area for an annual fee (www.bicing.com).

Bicing wishes to optimize the distribution of bicycles among its stations so their users can find a bicycle when they need one and has collected certain statistics related to the expected demand and the movement of the bicycles for each station and each hour of the day.

More specifically, the information that Bicing has collected is

-The predicted number of bicycles that are not going to be used in a station in the current hour and that are not needed, that can be moved to another station.

-The predicted number of bicycles that would have a station the next hour. This information only takes in account the movements due to the users of the service.

-The predicted number of bicycles that should have a station the next hour to meet the expected demand.

Also, the capacity to move bicycles from a station to another is limited, given

-F vans that are able to move 30 bicycles each.

-Each van in an hour can only perform one trip that moves bicycles from an origin station.

-There can be only one van for each origin station.

In addition there are two scenarios of demand namely

-Balanced : All the stations have a similar demand.

-Rush Hour : Some stations have more demand than others.

In the generation of the above scenarios it is necessary that the ratio between the stations and bicycles has to be at least 1 to 50.

The co-ordinates of each of the stations are given and it can be assumed that the length of the path of movement of a van from one station to another is the sum of the euclidean distances among the origin station and the destination station.

Given the above variables and constraints, the problem is to find a solution that gives all the movements of bicycles using vans for a given hour that optimize the following possible goals

-To meet the expected demand of bicycles as much as possible.

-To minimize the paths of the vans among stations while meeting as much of the expected demand as possible.

If we assume 'N' stations that Bicing has and 'F' Vans that are available to Bicing at an hour, then

-the best case scenario would be that there exists no such station which requires a bicycle

-the worst case scenario would be that when every station requires a bicycle.

However both these cases are highly improbable and hence we look at an average scenario where half the station can supply the excess of bicycles to meet the expected demand of the other half.
In such a scenario it is probable that the demand would be able to be satisfied.

For F>N, there will be excess Vans left over. This will be useful in the scenario if we were to use an operator where in a receiving station on receiving bicycles more than its requirement could turn into a supply station. In fact, as there is no limit on the number of bicycles that a station can have we will consider this possibility and enhance it to satisfy even more demand.

2.2 Problem State and its Representation

The above mentioned Bicing problem has to be solved using Local Search, which operates on a current state and moves to only its neighbors.

From the information that Bicing provides to us, we can get the information of the various elements of Bicing. We provide a list of definitions below for future reference

BicingStation : Its a station from where users lend or move bicycles to. Also, Bicing also moves bicycles from a station to other stations.
Vans : Bicing moves bicycles from a station to another station using Vans. A van's capacity is of 30 bicycles maximum and can only move to two stations from one station.
Move : A Van "Move's" from one station to another station, or from one station to two other stations.
BicingState : A state of 'N' BicingStation's and at maximum 'F' Move's of Vans.

BicingState is our choice of representing the problem's state. It is also interesting to note that from the information that Bicing provides us it is possible to classify BicingStations as
a) DemandStation : BicingStation's that require bicycles the next hour
b) SupplyStation : BicingStation's that have an excess of bicycles the next hour.
Given F Vans our problem now reduces to moving these F Vans from SupplyStations to the DemandStations to maximize the fulfillment of the expected demand. Hence the definition as provided above.

Additionally, from the information Bicing provides to us, the following information can be derived and we will store in a BicingState
1. GlobalDemand : The initial demand for bicycles across all stations that has to be satisified.
2. GlobalSupply : The initial excess of bicycles available across all stations that can be supplied.
3. CurrentDemandSatisfied : The demand for bicycles that has been satisfied till reaching this BicingState.
4. CurrentSupplyGiven : The number of bicycles that have been supplied till reaching this BicingState.
5. AvailableVans : The total number of free Vans that are available to do a Move from one station to another.
6. AssignedVans : The total number of Vans that are not free but are assigned to do a Move or have already moved bicycles to one station out of two possible stations.
7. FinishedAssignmentVans : The total number of Vans that have completed moving bicycles from their origin station to all possible stations.
8. MovesForThisState : The number of Move's that have been performed to reach this BicingState
9. DistanceTravelled : The total number of distance travelled to reach this BicingState.

An example of a state representation :

```
StateNo[=2039]                        //  The nodeID for this BicingState
GlobalDemand=63                        //  The initial demand that has to be satisfied
```

```
GlobalSupply=25                                  //  The initial supply that is available
CurrentDemandSatisfied=18                        //  The demand that has been satisfied till this
BicingState
CurrentSupplyGiven=18                            //  The supply that has been given till this BicingState
AvailableVans=freeVans4                          //  The free vans available for assignment
FinishedVans=doneVans4                           //  The vans that have completed their assignment
AssignedVans=3                                   //  The vans that are currently doing their assignent
Moves=[Move[Van=1]from[Sup=3]to[Dem=0],      //  The moves taken to reach this BicingState.
        Move[Van=5]from[Sup=9]to[Dem=0],
        Move[Van=0]from[Sup=2]to[Dem=0]and[Dem2=5]]
DistanceTravelled=288.43                         //   The distance travelled to reach this BicingState
```

3 Local Search

3.1 Initial State

3.1.1 Representation and Analysis of the Initial State

LocalSearch begins to search from a state which is a solution, hence our initial state has to be a solution. We will try to model our BicingState into an initial solution.

The following choices that we have to consider an initial state :

1.<u>An Empty Set of Vans</u>

This is a BicingState in which no Van is assigned to any SupplyStation.This BicingState only consists of 'N' BicingStations as is minus the Move's of F Vans

2.<u>A Set of Vans Assigned to SupplyStations</u>

As there is no cost associated with the assignment of Vans to a BicingStation which is a SupplyStation we have taken the choice of not considering the case of an empty set of Vans as an initial solution. But rather the assignment of Vans randomly to SupplyStations.

3.<u>A Set of Vans Assigned to SupplyStations with Maximum Supply</u>

Another choice for an initial solution is the option of assigning Vans to the SupplyStations with the maximum excess bicycles (i.e. supply). This would help us realize our first goal , however might not aid in the second goal. This has an associated cost of ordering the SupplyStations.

4.<u>A Set of Vans Assigned to SupplyStations with Maximum Supply to deliver to DemandStations with Maximum Requirement.</u>

An even better choice over the previous choice for an initial solution is the option of assigning Vans to the SupplyStations with the maximum excess bicycles (i.e. supply) wherein the Vans are assigned to deliver to the DemandStations which have the maximum requirements. This would help us realize our first goal of maximizing the fulfillment of expected demand as much as possible, however might not aid in the second goal. This has an associated cost of ordering both the SupplyStations and DemandStations.

We will only test the second and fourth choice of initial solutions to see if there is any significant improvement in search results in using these as initial solutions.

3.2 Actions / Search Operators

3.2.1 Representation and Analysis of the Actions / Search Operators

From the description of the problem the following are the constraints on the operators for the local search

- There a total of F Vans. This number is totally independent of the number of SupplyStations and hence demand may or may not be completely satisfied.
- The Van has a maximum capacity of 30 bicycles.

- The Van can move from one station to another station or it can move from one station to another two stations.

A Van can only move excess bicycles from a SupplyStation it is assigned to, and thus it does not make sense to move a Van from a SupplyStation to another SupplyStation.

Hence our search operators would be
- To move a Van from its assigned SupplyStation to one of all other possible DemandStations.
- To move a Van from its assigned SupplyStation to two of all other possible DemandStaions.

Also a sample action or Move would include the following operations
- Load the Van with only the excess bicycles from the SupplyStation upto a maximum of 30 bicycles.
- Moving the Van from one station to another, udpating its distance travelled.
- Updating the DemandStation's requirement with the number of bicycles it has received from the Van.
- If the Van has finished its second trip, then it dumps all its bicycles on its current DemandStation.

As there is no limit on the number of bicycles that a station can have we propose to handle the peculiar case when there are more Vans available than there are SupplyStations and a DemandStation was dumped with excess bicycles from a Van on its second trip. We then assign the next available Van to this DemandStation and treat it as a SupplyStation satisfying even more demand.

3.3 Heuristic Functions

3.3.1 Analysis of the Heuristic Functions

There are two separate goals that we have to achieve and hence there are two different heuristic functions that we have created for both of these goals.

3.3.1.1 Satisfying maximum global demand

In order to reach this goal the CurrentDemand must tend toward GlobalDemand , i.e that for each progressive state the demand that has been satisfied in order to reach that BicingState must be closer to the GlobalDemand or in other words the difference between the GlobalDemand and CurrentDemand should tend towards zero. Hence in order to satisfy the maximum global demad we have defined our heuristic as

$$\textit{minimizing} \quad \textit{GlobalDemand} - \textit{CurrentDemand}$$

3.3.1.2 Minimizing the distance travelled while statisfying the global demand

In order to reach this goal we will need to track the distances travelled by the Vans in order to reach this BicingState. The DistanceTravelled stored in the BicingState represents exactly this function. Also it is important to note that we have to take into account the satisfaction of Demand, therefore we introduced the factor of CurrentDemandSatisfied into the heuristic.
We then need to try to minimize the DistanceTravelled and maximize the CurrentDemandSatisified. In other words we want to maximize the CurrentDemandSatisfied per DistanceTravelled.
However during our tests we found that the heuristic will amost always choose the BicingState with DistanceTravelled equal to zero.

This is not an acceptable solution for us as we need to satisfy some demand atleast. Therefore we introduce the factor of the Number Of Vans Available – No of Vans that have Finished their tasks to the heuristic. In this way we know as Vans move and finish their tasks we are satisfying Demand and we try to minimize the distance travelled by these Vans, which is indeed our second goal.
Also just to make
Hence our second heuristic aims to

minimize (No Of Vans Available – No of Vans that Have Finished their task) + (CurrentDemandSatisfied)^2/CurrentDistanceTravelled

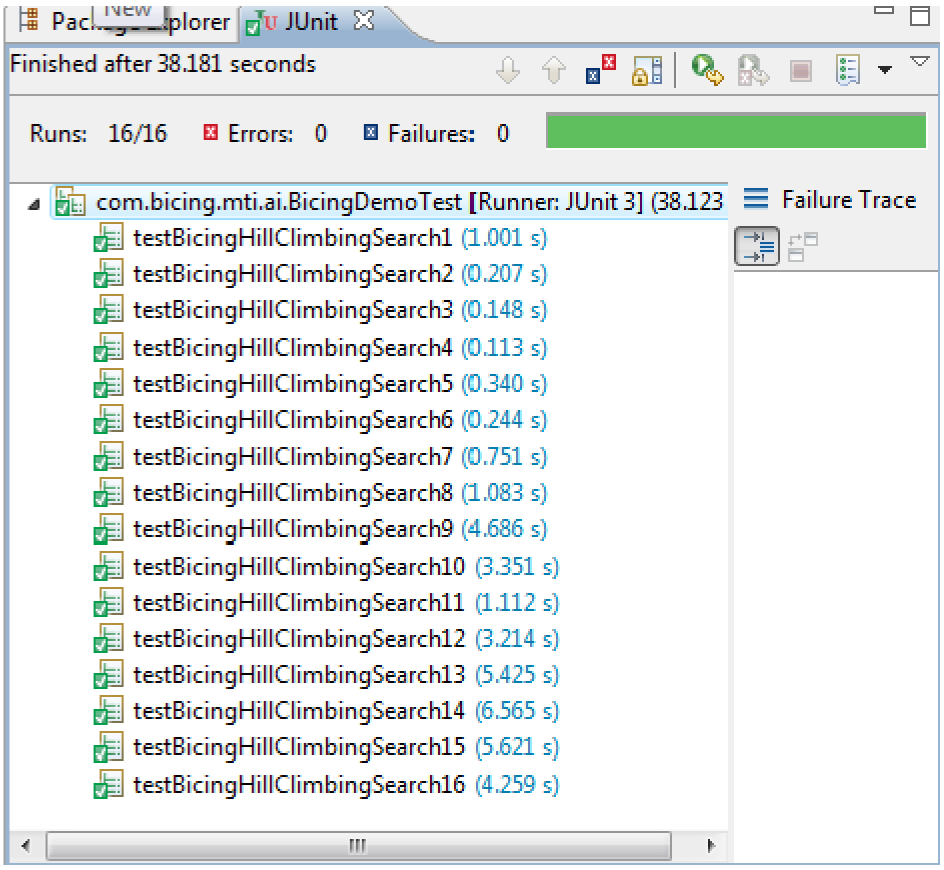
3.4 Experiments

We have defined a total of 25 tests using different scenarios which can be found in the attached source code for both HillClimbing and SimulatedAnnealing.
The comments and the necessary inputs to the test cases are mentioned in the code.
Please refer to BicingDemoTest.java for HillClimbing test cases.
Also refer to BicingDemoSimAnnTest.java for SimulatedAnnealing test cases.

Als please refer to the attached excel file

3.4.1 Influence of the Initial Solution

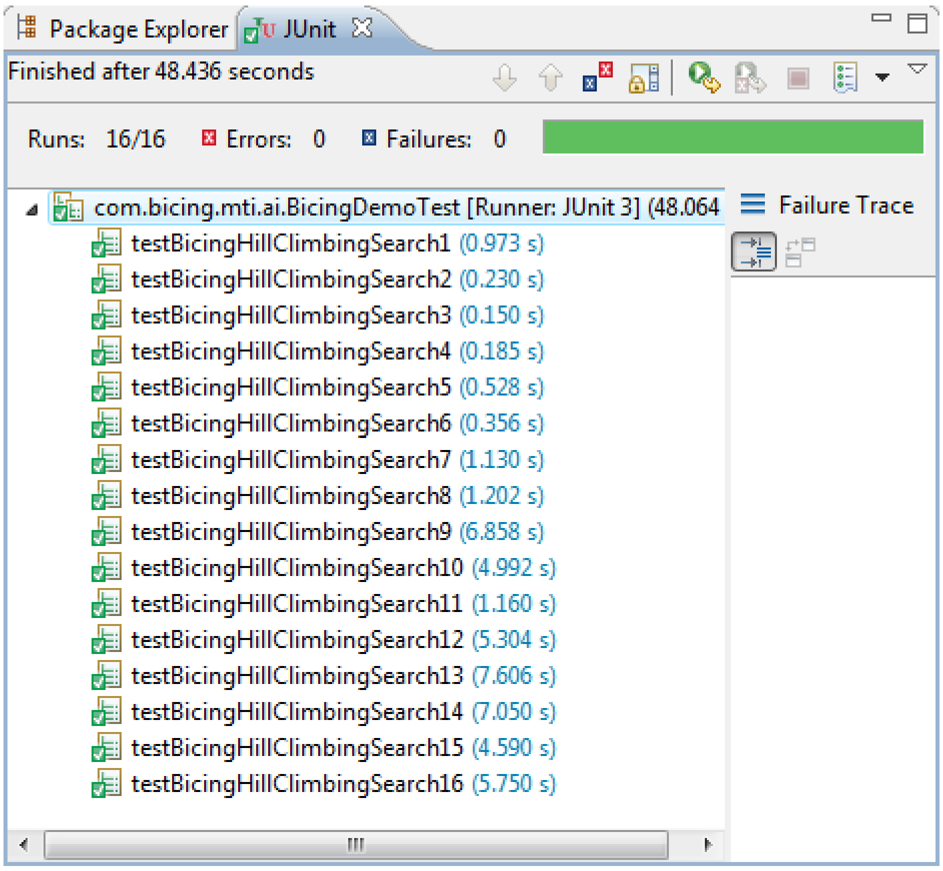
Taking into account the running of the 2nd and 4th choices as mentioned in section 3 the results of the snapshot test run were as follows.



```
Finished after 38.181 seconds
Runs: 16/16   Errors: 0   Failures: 0
com.bicing.mti.ai.BicingDemoTest [Runner: JUnit 3] (38.123
  testBicingHillClimbingSearch1 (1.001 s)
  testBicingHillClimbingSearch2 (0.207 s)
  testBicingHillClimbingSearch3 (0.148 s)
  testBicingHillClimbingSearch4 (0.113 s)
  testBicingHillClimbingSearch5 (0.340 s)
  testBicingHillClimbingSearch6 (0.244 s)
  testBicingHillClimbingSearch7 (0.751 s)
  testBicingHillClimbingSearch8 (1.083 s)
  testBicingHillClimbingSearch9 (4.686 s)
  testBicingHillClimbingSearch10 (3.351 s)
  testBicingHillClimbingSearch11 (1.112 s)
  testBicingHillClimbingSearch12 (3.214 s)
  testBicingHillClimbingSearch13 (5.425 s)
  testBicingHillClimbingSearch14 (6.565 s)
  testBicingHillClimbingSearch15 (5.621 s)
  testBicingHillClimbingSearch16 (4.259 s)
```

Figure 6.1 : 2nd choice of initial solution, A set of Vans supplied to a Van Station

This finishes in 38.181 seconds and the individual time taken for each of the tests is shown.



Package Explorer | JUnit

Finished after 48.436 seconds

Runs: 16/16 Errors: 0 Failures: 0

```
com.bicing.mti.ai.BicingDemoTest [Runner: JUnit 3] (48.064
    testBicingHillClimbingSearch1 (0.973 s)
    testBicingHillClimbingSearch2 (0.230 s)
    testBicingHillClimbingSearch3 (0.150 s)
    testBicingHillClimbingSearch4 (0.185 s)
    testBicingHillClimbingSearch5 (0.528 s)
    testBicingHillClimbingSearch6 (0.356 s)
    testBicingHillClimbingSearch7 (1.130 s)
    testBicingHillClimbingSearch8 (1.202 s)
    testBicingHillClimbingSearch9 (6.858 s)
    testBicingHillClimbingSearch10 (4.992 s)
    testBicingHillClimbingSearch11 (1.160 s)
    testBicingHillClimbingSearch12 (5.304 s)
    testBicingHillClimbingSearch13 (7.606 s)
    testBicingHillClimbingSearch14 (7.050 s)
    testBicingHillClimbingSearch15 (4.590 s)
    testBicingHillClimbingSearch16 (5.750 s)
```

Failure Trace

Figure 6.2 : 4th choice of initial solution, A Set of Vans Assigned to SupplyStations with Maximum Supply to deliver to DemandStations with Maximum Requirement.

This finished in 48.436 seconds and the individual time taken for each of the tests is shown.

As can be seen the 2nd choice generates solutions much faster . However lets delve a little deeper under the scenes and se what is happening. We will like to consider the number of nodes that are generated for each of the above initial solutions and also which yields a better solution.

States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
2nd Initial Solution : Scenario Equilibrium									
118	55	35	0	0	0	4	3	0	0
122	55	35	18	18	64	4	2	1	32.7
177	55	35	35	35	225	4	2	2	63.6
2nd Initial Solution : Scenario RushHour									
210	88	64	0	0	0	3	4	0	0
220	88	64	29	29	45.5	3	3	1	33
250	88	64	47	47	109	3	2	2	53.4
279	88	64	64	64	137	3	1	3	72.7
4th Initial Solution : Scenario Equilibrium									
118	55	35	0	0	0	4	3	0	0
120	55	35	18	18	64	4	2	1	32.7
177	55	35	35	35	170	4	2	2	63.6
4th Initial Solution : Scenario RushHour									
219	88	64	0	0	0	3	4	0	0
221	88	64	29	29	45.5	3	3	1	33
258	88	64	47	47	107	3	2	2	53.4
286	88	64	64	64	134	3	1	3	72.7

Key	
GS	GlobalSupply
GD	GlobalDemand
CDS	CurrentDemandSatisfied
CSG	CurrentSupplyGiven
Dist	DistanceTravelled
A	AvailableVans
As	AssignedVans
F	FinishedVans
sRate	CDS*100/GD

Figure 6.3 : Output, Number Of Vans = 7, Number Of Stations = 15, Ratio of Stations to Bicycles = 50

where sRate is defined as Satisfaction Rate or the percentage of CurrentDemandSatisfied over GlobalDemand per BicingState.

We choose this testCase as the number of Vans are nearly half the number of stations and taking into account the average scenario of half demand and half supply stations per number of 'N' BicingStations this would be a good average case.

From the output above it is clear that both the solution yield exactly the same result exhausting all the supplies to achieve maximum demand satisfaction. Note the difference in the distances travelled.
In the Equilibrium scenario the state search space is identical and both initial solutions take the same number of BicingStates to yield a solution .
In the RushHour scenario the state search space are not identical and infact the the 4th intial solution choice results in even more nodes than the 2nd inital solution. But both in the end yiel d a similar result by trying to maximize the demand satisfaction rate.

Thus it can be seen that the outcome of the result of the search does not change but the 4th initial solution uses more states to reach the solution. Lets try this on a more exhaustive test case as below:

Figure 6.4 : Output, Number Of Vans= 15, Number Of Stations= 15, Ratio of Stations to Bicycles= 50

States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
2nd Initial Solution : Scenario Equilibrium									
916	82	82	0	0	0	7	8	0	0
929	82	82	18	18	19.1	7	7	1	22
1332	82	82	33	33	138.03	7	6	2	40.2
1660	82	82	46	46	183.48	7	5	3	56.1
1983	82	82	56	56	189.8	7	4	4	68.3
2371	82	82	65	65	259.98	7	4	5	79.3
2455	82	82	73	73	360.65	7	4	6	89
2499	82	82	80	80	404.93	7	3	7	97.6
2537	82	82	82	82	567.17	7	3	8	100
2nd Initial Solution : Scenario RushHour									
2544	114	112	0	0	0	5	10	0	0
2553	114	112	26	26	19.21	5	9	1	22.8
2816	114	112	47	47	94.38	5	8	2	41.2
3049	114	112	63	63	146.24	5	7	3	55.3
3234	114	112	76	76	162.4	5	6	4	66.7
3345	114	112	89	89	166.87	5	5	5	78.1
3403	114	112	99	99	173.19	5	4	6	86.8
3448	114	112	107	107	205.64	5	3	7	93.9
3476	114	112	110	110	251.09	5	2	8	96.5
3489	114	112	112	112	304.69	5	1	9	98.2
4th Initial Solution : Scenario Equilibrium									
948	82	82	0	0	0	7	8	0	0
950	82	82	18	18	30.15	7	7	1	22
1344	82	82	33	33	82.78	7	6	2	40.2
1689	82	82	46	46	148.81	7	5	3	56.1
1982	82	82	56	56	155.13	7	4	4	68.3
2256	82	82	65	65	275.63	7	4	5	79.3
2479	82	82	73	73	329.54	7	4	6	89
2609	82	82	80	80	493.46	7	4	7	97.6
2648	82	82	82	82	646.1	7	4	8	100
4th Initial Solution : Scenario RushHour									
2651	114	112	0	0	0	5	10	0	0
2653	114	112	26	26	30.15	5	9	1	22.8
2904	114	112	47	47	105.32	5	8	2	41.2
3131	114	112	63	63	127.79	5	7	3	55.3
3332	114	112	76	76	149.05	5	6	4	66.7
3509	114	112	89	89	185.45	5	5	5	78.1
3661	114	112	99	99	211.53	5	4	6	86.8
3804	114	112	107	107	261.98	5	4	7	93.9
3888	114	112	110	110	308.02	5	3	8	96.5
3924	114	112	112	112	344.14	5	2	9	98.2

Key	
GS	GlobalSupply
GD	GlobalDemand
CDS	CurrentDemandSatisfied
CSG	CurrentSupplyGiven
Dist	DistanceTravelled
A	AvailableVans
As	AssignedVans
F	FinishedVans
sRate	CDS*100/GD

where sRate is defined as Satisfaction Rate or the percentage of CurrentDemandSatisfied over GlobalDemand per BicingState.

We choose this testCase as this is an extreme test case where the number of Vans are equivalent to the numbe of BicingStations which implies that there are possibly more Vans than the number of SupplyStations.

From the output above again, it can be seen that both the solutions yield exactly the same result exhausting all the supplies to achieve maximum demand satisfaction. Note the difference in the distances travelled.
In both the Equilibrium scenario and RushHour scenario the state search space are not identical and the 4th initial solution results in the creation more states than the 1st initial solution.

We expected the SatisfactionRate to be more in the case of the 4th initial solution, however it seems that is not the case. The 2nd initial solution delivers exactly the same result and results in even lesser generation of nodes by the algorithm.

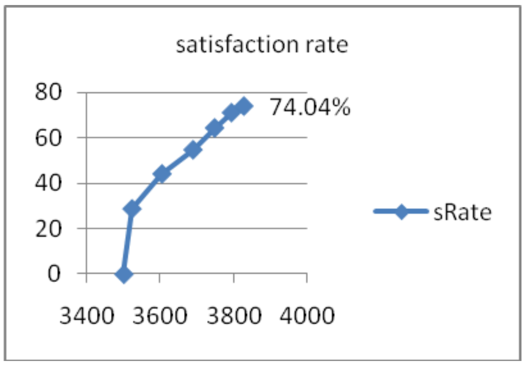
Thus the following two conclusions can be comprehensively made
It is worth the computational cost of obtaining an initial solution but given the quality of final solutions obtained it is not worth the computational cost of obtaining a 'good' initial solution.

3.4.2 Comparison of the Scenarios

Although we have the data for all our defined tests we choose 3 tests to generalize the test cases.

3.4.2.1 *Test Case in which F (No of Vans) < = N (number of BicingStations) / 2*

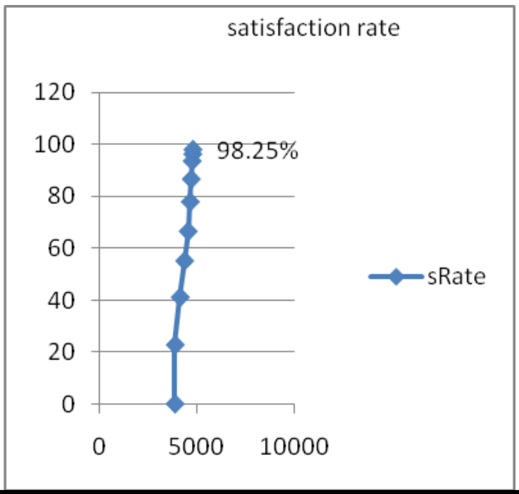
Scenario : Equilibrium



States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
3501	104	87	0	0	0	4	6	0	0
3523	104	87	30	30	1.41	4	5	1	28.85
3605	104	87	46	46	29	4	4	2	44.23
3690	104	87	57	57	51.02	4	3	3	54.81
3749	104	87	67	67	101	4	2	4	64.42
3795	104	87	74	74	140.96	4	1	5	71.15

3829	104	87	77	77	172.58	4	0	6	74.04
TEST:Vans=10,Stations=20,**Equilibrium**									

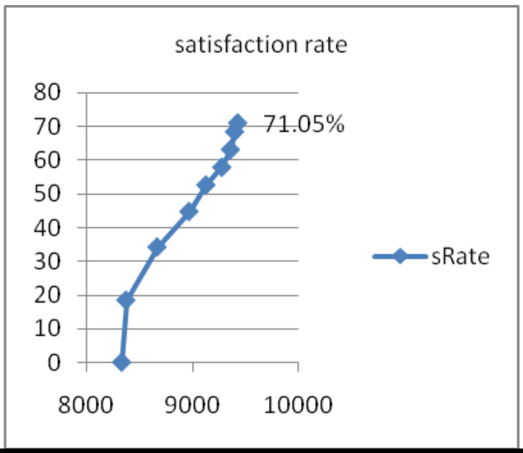
Scenario : RushHour



States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
3845	114	112	0	0	0	5	10	0	0
3854	114	112	26	26	19.21	5	9	1	22.81
4117	114	112	47	47	94.38	5	8	2	41.23
4350	114	112	63	63	146.24	5	7	3	55.26
4535	114	112	76	76	162.4	5	6	4	66.67
4646	114	112	89	89	166.87	5	5	5	78.07
4704	114	112	99	99	173.19	5	4	6	86.84
4749	114	112	107	107	205.64	5	3	7	93.86
4777	114	112	110	110	251.09	5	2	8	96.49
4790	114	112	112	112	304.69	5	1	9	98.25
TEST:Vans=10,Stations=20,**RushHour**									

3.4.2.2 *TestCases in which F (No of Vans) >= N (Number of BicingStations) / 2*

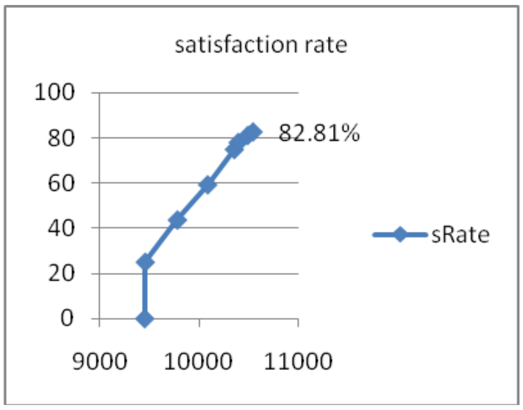
Scenario : Equilibrium



States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
8337	38	27	0	0	0	6	9	0	0
8377	38	27	7	7	23.02	6	8	1	18.42
8670	38	27	13	13	56.56	6	7	2	34.21
8970	38	27	17	17	84.22	6	6	3	44.74
9130	38	27	20	20	134.2	6	5	4	52.63
9280	38	27	22	22	185.4	6	4	5	57.89
9361	38	27	24	24	272.86	6	3	6	63.16
9400	38	27	26	26	360.72	6	2	7	68.42
9429	38	27	27	27	400.28	6	1	8	71.05

TEST:Vans=15,Stations=10,**Equilibrium**

Scenario : RushHour

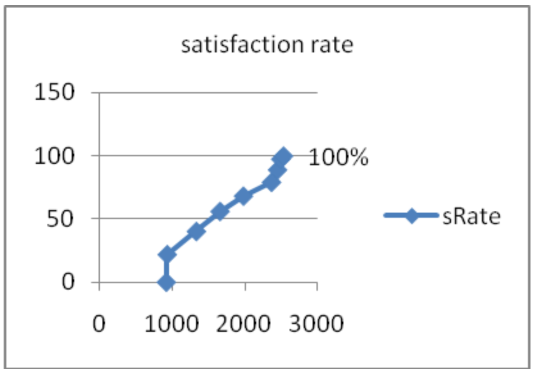


States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
9449	64	53	0	0	0	6	9	0	0
9455	64	53	16	16	66.03	6	8	1	25
9779	64	53	28	28	85.24	6	7	2	43.75
10084	64	53	38	38	133.32	6	6	3	59.38
10353	64	53	48	48	271.56	6	6	4	75
10392	64	53	50	50	315.84	6	5	5	78.13
10488	64	53	52	52	345.92	6	4	6	81.25
10541	64	53	53	53	367.02	6	3	7	82.81
TEST:Vans=10,Stations=20,RushHour									

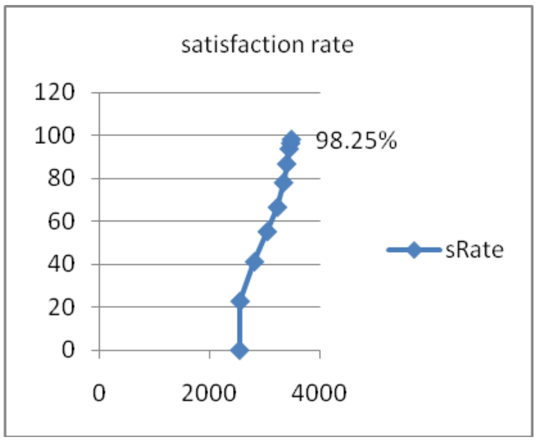
3.4.2.3 TestCases in which F (No of Vans) = N (Numbe of BicingStations)

Refer to data set in Figure 6.4

Scenario : Equilibrium



Scenario : RushHour



As the number of nodes expanded are lesser during the equilibrium scenario it can be realized that the problem is easier to solve in the Equilibrium scenario rather than the RushHour scenario.

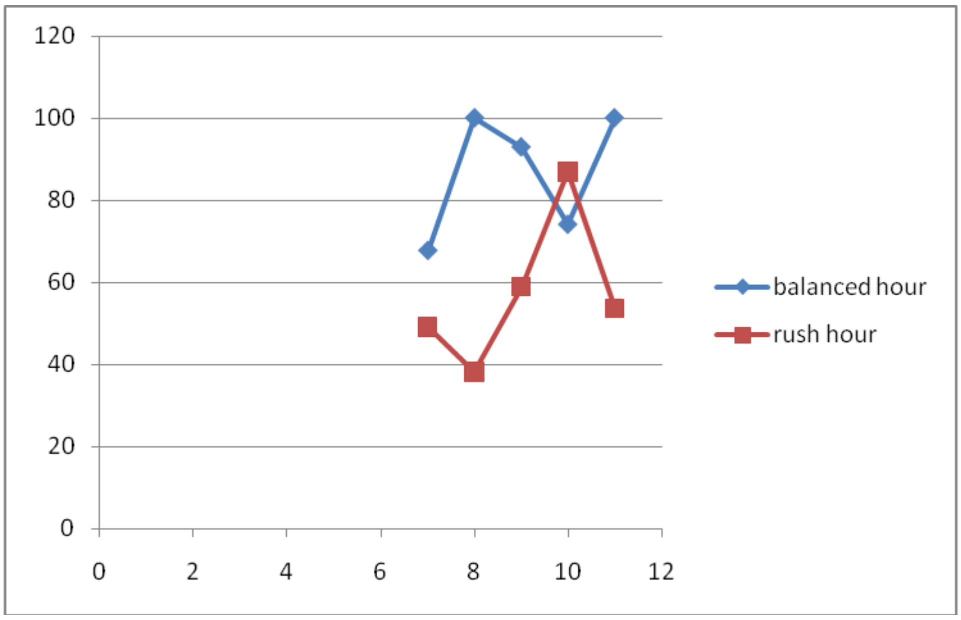
3.4.3 Threshold of Vans

We decided to conduct a series of experiments on a set number of stations and gradually increase the number of Vans to determine the outcome of this threshold.
The final results are attached here for reference, the attached excel sheet contains the complete description.

We plotted these values against the satisfaction rate, the same as defined above. The number of staions was taken constant as 20.

The results were

Vans	balanced hr	rush hr
7	67.6829268	49.115
8	100	38.125
9	92.920354	58.929
10	74.0384615	86.842
11	100	53.612



We expected the threshold of Vans to be at half the number of Stations. But due to the randomness of the data we experienced a variance. However, all subsequent tests satisfied the demand and hence it can concluded that the threshold of vans after which it does not affect the solution will be greater than the number of stations divided by 2.

3.4.4 Influence of the Actions

There are three actions that we have considered

- To move a Van from its assigned SupplyStations to one of all other possible DemandStations.
  This is the scenario where in we only consider the movement of a Van from one possible Supply Station to any of the other Demand Stations.

- To move a Van from its assigned SupplyStations to two of all other possible DemandStations.
  This is the scenario where in we only consider the movement of a Van from one possible Supply Station to any of the other two Demand Stations.

- To move a Van from its assigned SupplyStations to either one of the other possible Demand Stations or to any pair of two other possible Demand Stations.

3.4.5 Influence of the Heuristic Functions

During our tests we have played and tested with various heuristic functions.
In order to maximize the first goal , the maximum fulfilment of demand, the Heuristic function we have chosen tries to satisfy the goal completely and searches a lot of state spaces as different movements of a Van in a state will result in even greater states to evaluate. And this evaluation heuristic actually takes into account the changing state of the influential paramenters and has a major impact on the solution.

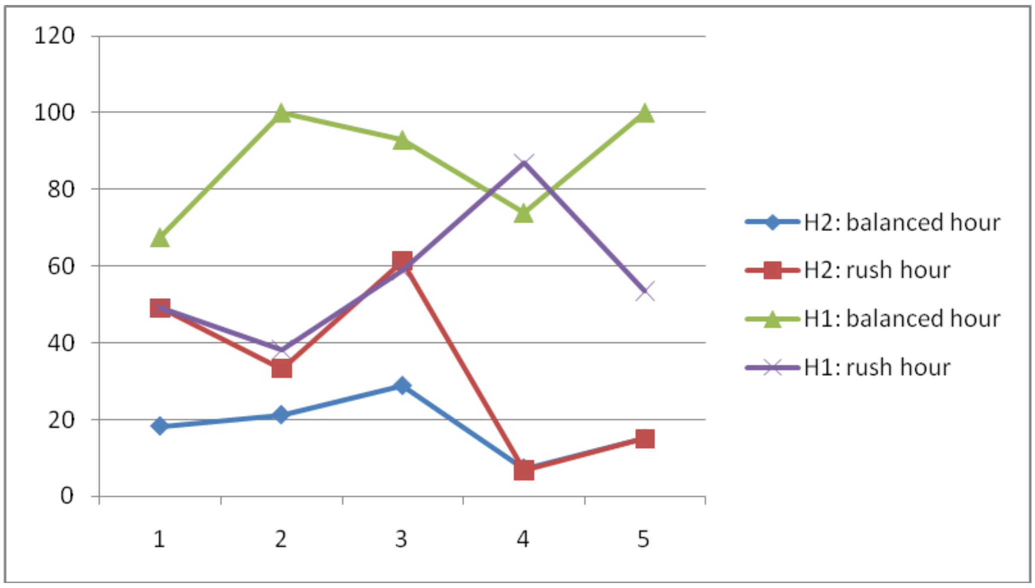
We tried other heuristic functions like , the total number of Vans – the total number of Vans that Finished their tasks , taking the assumption that if F Vans are moved then if all possible F Van movements are done then a solution can be found. However the parameters used in this heuristic do not evaluate the influential parameters , that is, the supply and demand but is based on the movement of Vans and does not therefore help us in maximizing our first goal.

For the second goal the objective is to minimize the path while trying to satisfy demand. We have to choose our heuristic function carefully in this case because if we try to just minimize the distance travelled then the Vans might just not move.

We first tried the heuristic function to minimize the distance travelled, using the variable CurrentDistanceTravelled. This as expected gave us a solution of no vans to travel.
But this as explained before was not an acceptable solution and hence we tried to maximize the CurrentDemandSatisfied per CurrentDistanceTravelled.
But during our test runs we found that this heuristic yet again chose the BicingState with DistanceTravelled equal to zero.
Hence in our third attempt at the Heuristic function we introduce the factor for Van movements, which assumes that as a Van moves it satisfied demand. Hence the more the number of Vans move the more the demand is satisfied. This gave us an acceptable solution of minimum distance travelled and demand satisfied.

3.4.6 Comparison of Heuristics

Also we tried to compare the performance of one heuristic with respect to the other . The simple test of one Heuristic against another is shown below for reference. The details are provided in the attached excel sheet.



where H1 stands for heuristic function 1 and H2 stands for heuristic function 2.

It can be seen that for both balanced and rush hour scenarios the Heuristic function H1 performs better than heuristic function H2.

We also ran a test for the Heuristc functions against different weights to each of the Heuristic in Heuristic 2. The table lists the selection of the nodes on the conditions applied.

States	GD	GS	CDS	CSG	Dist	A	As	F
More Weightage to Heuristic1								
1	263	146	0	0	0	5	6	0
18	263	146	18	18	1.41	5	5	1
Equal Weightage to both heuristics								
1	263	146	0	0	0	5	6	0
18	263	146	18	18	1.41	5	5	1
More Weightage to Heuristic2								
1	263	146	0	0	0	5	6	0

3.4.7 Comparison of the Different Algorithms

Please refer to the attached excel sheet for even more details.

3.4.7.1 Test of Lamda for Simulated annealing.

Please refer to the attached excel sheet in Sheet lamda test. We ran a few tests to determine the value of lamda, the number of steps that the Simulated annealing algorithm should take.

We finally decided to settle with the values of

```
int k = 20;

double lam = 0.045;

int limit = 20;

int steps = 100;
```

3.4.7.2 Test of HillClimbing and Simulated annealing.

We ran a couple of tests against Hill Climbing and Simulated Annealing. For specific details refer to Sheet : Hillc-Heuristic1. This gives the scenarios of the various tests run, specifically for 20 stations the number of vans were increases from 7 to 11. Again the same approach was applied for Simulated Annealing. Please refer to the Sheet : Simulated-Annealing in the attached excel sheet.

It can be noticed that the state space generated by the Simulated Annealing is much larger. Lets look at two equal tests of both the scenarios, Equilibrium and RushHour.

HillClimbing – Heuristic 1

States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
3501	104	87	0	0	0	4	6	0	0
3523	104	87	30	30	1.41	4	5	1	28.84615
3605	104	87	46	46	29	4	4	2	44.23077
3690	104	87	57	57	51.02	4	3	3	54.80769
3749	104	87	67	67	101	4	2	4	64.42308
3795	104	87	74	74	140.96	4	1	5	71.15385
3829	104	87	77	77	172.58	4	0	6	74.03846
TEST:Vans=10,Stations=10,**Equilibrium**									

States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
3845	114	112	0	0	0	5	#	0	0
3854	114	112	26	26	19.21	5	9	1	22.80702
4117	114	112	47	47	94.38	5	8	2	41.22807
4350	114	112	63	63	146.24	5	7	3	55.26316
4535	114	112	76	76	162.4	5	6	4	66.66667
4646	114	112	89	89	166.87	5	5	5	78.07018
4704	114	112	99	99	173.19	5	4	6	86.84211
4749	114	112	107	107	205.64	5	3	7	93.85965
4777	114	112	110	110	251.09	5	2	8	96.49123
4790	114	112	112	112	304.69	5	1	9	98.24561
TEST:Vans=10,Stations=20,**RushHour**									

It can be noticed that HillClimbing takes 3829 nodes to reach a solution in Equilibrium scenario and 4790 nodes to reach a solution in RushHour scenario.

In comparison lets have a look at Simulated Annealing

SimulatedAnnealing Heuristic1

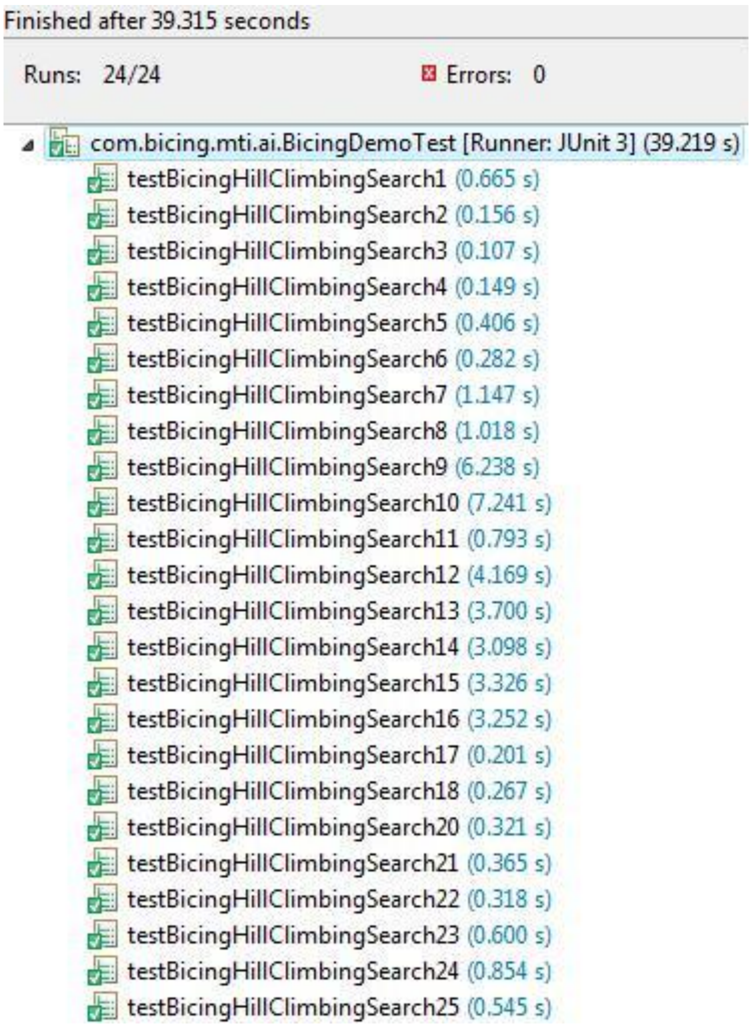
States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
4163	104	87	0	0	0	4	6	0	0
4210	104	87	16	16	118.75	4	5	2	15.38462
4306	104	87	27	27	198.49	4	4	4	25.96154
4380	104	87	34	34	360.05	4	3	6	32.69231
4450	104	87	64	64	367.78	4	3	7	61.53846
4462	104	87	67	67	384.27	4	2	8	64.42308
4487	104	87	77	77	535.84	4	2	9	74.03846
TEST:Vans=10,Stations=20,**Equilibrium**									

States	GD	GS	CDS	CSG	Dist	A	As	F	sRate
4587	114	112	0	0	0	5	10	0	0
4703	114	112	13	13	147.13	5	9	2	11.40350877
4855	114	112	14	14	266.06	5	9	2	12.28070175
5089	114	112	14	14	303.4	5	8	3	12.28070175
5229	114	112	27	27	321.84	5	7	4	23.68421053
5347	114	112	29	29	356.21	5	6	5	25.43859649
5406	114	112	32	32	401.66	5	5	6	28.07017544
TEST:Vans=10,Stations=20,**RushHour**									

Simulated Annealing takes 4487 nodes to reach a solution in the case of an Equilibrium scenario and reached upto the same satisfaction rate as HillClimbing.

But in the case of RushHour, Simulated Annealing takes 5406 and reaches even a worse state than HillClimbing.

Also by referring to the time taken for the execution of all the test please refer below :

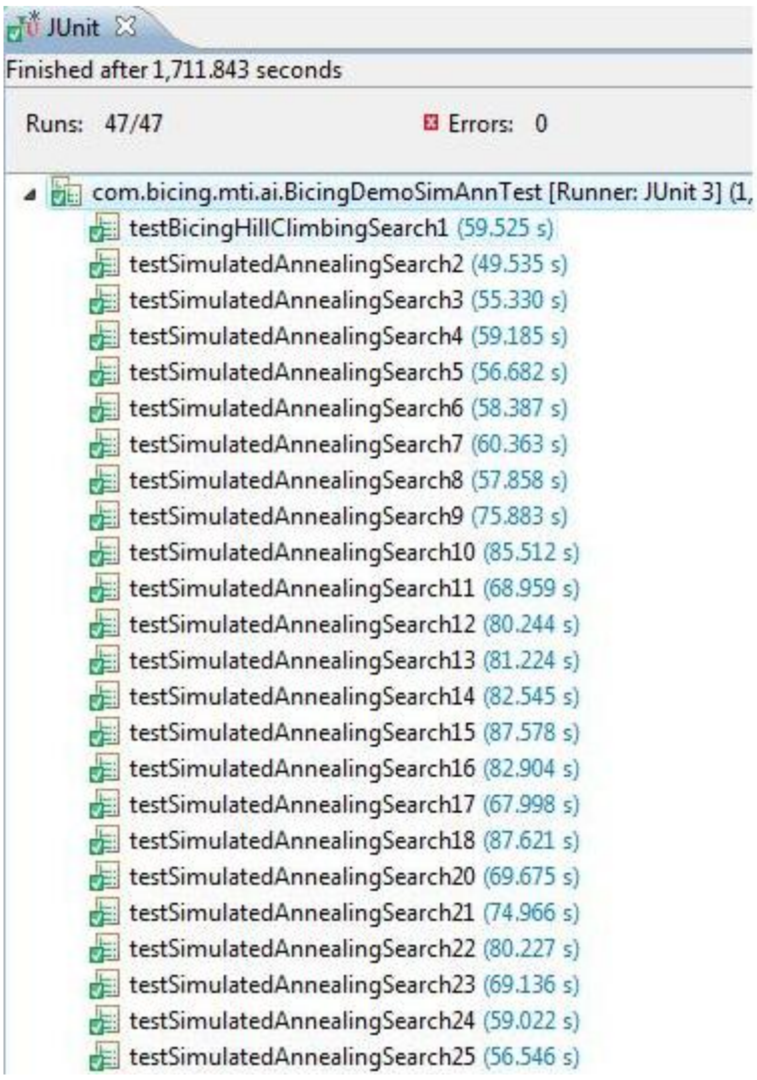


```
Finished after 39.315 seconds

Runs: 24/24                    Errors: 0

com.bicing.mti.ai.BicingDemoTest [Runner: JUnit 3] (39.219 s)
    testBicingHillClimbingSearch1 (0.665 s)
    testBicingHillClimbingSearch2 (0.156 s)
    testBicingHillClimbingSearch3 (0.107 s)
    testBicingHillClimbingSearch4 (0.149 s)
    testBicingHillClimbingSearch5 (0.406 s)
    testBicingHillClimbingSearch6 (0.282 s)
    testBicingHillClimbingSearch7 (1.147 s)
    testBicingHillClimbingSearch8 (1.018 s)
    testBicingHillClimbingSearch9 (6.238 s)
    testBicingHillClimbingSearch10 (7.241 s)
    testBicingHillClimbingSearch11 (0.793 s)
    testBicingHillClimbingSearch12 (4.169 s)
    testBicingHillClimbingSearch13 (3.700 s)
    testBicingHillClimbingSearch14 (3.098 s)
    testBicingHillClimbingSearch15 (3.326 s)
    testBicingHillClimbingSearch16 (3.252 s)
    testBicingHillClimbingSearch17 (0.201 s)
    testBicingHillClimbingSearch18 (0.267 s)
    testBicingHillClimbingSearch20 (0.321 s)
    testBicingHillClimbingSearch21 (0.365 s)
    testBicingHillClimbingSearch22 (0.318 s)
    testBicingHillClimbingSearch23 (0.600 s)
    testBicingHillClimbingSearch24 (0.854 s)
    testBicingHillClimbingSearch25 (0.545 s)
```

And for simulated annealing



```
JUnit
Finished after 1,711.843 seconds
Runs: 47/47        Errors: 0
com.bicing.mti.ai.BicingDemoSimAnnTest [Runner: JUnit 3] (1,
    testBicingHillClimbingSearch1 (59.525 s)
    testSimulatedAnnealingSearch2 (49.535 s)
    testSimulatedAnnealingSearch3 (55.330 s)
    testSimulatedAnnealingSearch4 (59.185 s)
    testSimulatedAnnealingSearch5 (56.682 s)
    testSimulatedAnnealingSearch6 (58.387 s)
    testSimulatedAnnealingSearch7 (60.363 s)
    testSimulatedAnnealingSearch8 (57.858 s)
    testSimulatedAnnealingSearch9 (75.883 s)
    testSimulatedAnnealingSearch10 (85.512 s)
    testSimulatedAnnealingSearch11 (68.959 s)
    testSimulatedAnnealingSearch12 (80.244 s)
    testSimulatedAnnealingSearch13 (81.224 s)
    testSimulatedAnnealingSearch14 (82.545 s)
    testSimulatedAnnealingSearch15 (87.578 s)
    testSimulatedAnnealingSearch16 (82.904 s)
    testSimulatedAnnealingSearch17 (67.998 s)
    testSimulatedAnnealingSearch18 (87.621 s)
    testSimulatedAnnealingSearch20 (69.675 s)
    testSimulatedAnnealingSearch21 (74.966 s)
    testSimulatedAnnealingSearch22 (80.227 s)
    testSimulatedAnnealingSearch23 (69.136 s)
    testSimulatedAnnealingSearch24 (59.022 s)
    testSimulatedAnnealingSearch25 (56.546 s)
```

By taking as reference the tests that we have run and the data formulate in the Sheets Hillc-Heuristic1 and Simulated Annealing it can be concluded that

1) HillClimbing has better computational cost both in terms of state space generation and time execution.
2) The quality of the solutions obtained in our case were better in the case of HillClimbing than using SimulatedAnnealing.