A comparison between Genetic Algorithm and Simulated Annealing using Travelling Salesman Problem Instances

Alexandru Ioan Cuza University - Department of Computer Science

Luca Năstasă Baraș,Dragoș Damian

January 12, 2023

Abstract

Travelling salesman problem(TSP) is a NP-hard problem in combinatorial optimisation and in this report we want to do a comparison between Simulating Annealing and a Genetic Algorithm, both metaheuristic methods, on 10 instances of our optimisation problem.

Keywords: Genetic Algorithm, Simulated Annealing

1 Introduction

The travelling salesman problem (TSP) is a combinatorial optimisation problem that is derived from many production and scheduling problems and is presented as a salesman being required to visit n cities but given the traveling costs between each city, the salesman has to find the shortest distance required to visit all required cities.

In this paper we will try to approach the problem optima with a genetic algorithm(GA) and with a trajectory method Simulating Annealing(SA). When we will do a comparison between both we will see that GA has better results as the number of the cities is increasing.

The test problems I used consists in 10 distinct instances and the difficulty of those is increasing from the first one to the last(the number of the cities of every instance represents the difficulty). Our chosen instances are : bayg29, berlin52, bier127, d198, gil262, pcb442, pa561, rl1889, rl5915, d15112

The implemented SA algorithm starts with a temperature of 100 and it is cooling to 10^{-3}. We chose a medium cooling function because there are many instances with big dimension differences and we chose a result in a reasonable time even if with a slower cooling function we could achieve better results.

The GA is projected as follows: the genotype represents the order in which a salesman visits the cities and the mutation and crossover must ensure that no city is visited more than once (We placed the restrictions on our used techniques as selection, mutation and crossover). So our GA follows the classic schema with some optimisations added for better results.The dimension of population is 200 individuals and the number of generations is 2000. We tried both Tournament and Wheel Of Fortune selections and we chose the Tournament one .

And as mutations we tried for :SBM, Slide Mutation, IRGIBNM, RGIBNM and in the end we chose the (Selection Best Mutation)SBM one which is a greedy method of choosing the best mutation and it is shown to be more stable , it is made from IRGIBNM mutation ,RGIBNM mutation Slide mutation and Inversion mutation. The chosen cross-over between (Order Cross Over) OX, (Cycle Cross Over)CX and GX is GX one, that has a greedy strategy too, it is proposed by Grefenstette in 1984.

2 General Setting

2.1 Genetic Algorithm

In this following pseudo code is the general schema of our genetic algorithm :

```
CreatePopulation($population$) ;          ▷ We generate a random population
evaluate($population$)
$i \leftarrow 1$;
while i $\leq$ Number of Generations do
    select($population$)
    mutate($population$)
    crossover($population$)
    evaluate($population$)
    $i \leftarrow i + 1$
end while
return best($pop$)
```

The chromosome is represented by a route of n cities (all cities are included and exclusive) so it can be consider a permutation(values are $1->n$). Our population of candidate solutions are generated by shuffling the permutation and then for finding the optimum we apply the genetic operators : selection mutation and cross over.

Our Fitness formula is :

$$fit = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

As selection method we chose the Tournament one because we observed in our process that if we use the Wheel of Fortune, we have to deal with premature convergence and we can control the selection pressure better. The Tournament selection randomly chooses k elements($1 < k < n$) (from a population) that represents our pool and then from those k elements we choose the best element and we put it in our new population.

The cross over operation is done with one cutting point where each child replaces its parents. The generation is sorted ascending by its own crossover probability (that is generated randomly at the initialization step) and the cutting point is decided when first individual has a better crossover probability than the global crossover probability. In our algorithm we used GX Cross Over that uses an heuristic approach invented by Grefenstette :

1. randomly select the first city from our parent
2. find the city that follows right after selected city in both parents
3. select the city which yields the lowest cost
4. append the city to the new route list

5. select the appended city from the new route list as template
6. repeat the process

We used elitism in proportion of 12%(replacing the best individuals from population) and the cross-over probability is $4 \cdot 10^{-1}$.

For better precision we chose to first use a small space search for every individual of our population and start the GA from the better point. The search is done with a selection pressure and the neighbours are chosen after the reverse mutation.

The mutation operator is made after BMS(Best mutation Selection) strategy and it has to choose the best mutation between : IRGIBNM, RGIBNM, Slide Mutation and Inversion Mutation(randomly choose 2 position in our route and then reverse the city order between those 2 position).

IRGIBNM is a new hybrid mutation that :

1. randomly chooses 2 position of our route list and reverse the cities order from that sequence.
2. uses RGIBNM mutation on the new genotype, which says that after we chose a random city on position k from our route list,we look for the closest city of our random chosen city on position p,after that we chose another city near the city on the p position which is on position q and then swapping the city on k+1 position(neighbour of our random chosen city) with the city on the position q.

The mutation happens when the mutation probability, which is set to 0.02, is higher than a randomly generated number between 0 and 1.

2.2 Simulating Annealing

In this following pseudo code is the general schema of our SA :

```
$iter \leftarrow 100$;
while $iterations \geq 1$ do
    $route \leftarrow rangomPerm(\{1, 2, \ldots, k\})$
    $t \leftarrow 100$;
    $CostOfRoute \leftarrow 0$; $neighbour \leftarrow 0$
    $CostOfRoute \leftarrow Cost(route)$
    $best \leftarrow 1000000$
    while t $\geq$ 0.001 do
        $neighbour \leftarrow route$
        $RandomSwap(neighbour)$
        $NeighbourCost \leftarrow Cost(neighbour)$
        if $Neighbourcost < CostOfRoute$ then
            $route \leftarrow neighbour$
            $CostOfRoute \leftarrow Neighbuorcost$
        else
            if $random(0, 0.999999) < exp(-abs(CostOfRoute - Neighbourcost)/t)$
then
                $route \leftarrow neighbour$
                $CostOfRoute \leftarrow NeighbourCost$
                $t \leftarrow t \cdot 0.995$
            end if
        end if
    end while
    $best \leftarrow min\,(bestCost, CostOfRoute)$
    $i \leftarrow i - 1$
end while
return best
```

3 Results

This are or results for both algorithms :

Simulated Annealing					
Instance	Optimal cost	Best Cost	Mean	StDev	Duration
bayg29	1610	8443	8872	224	7min 43s
berlin52	7542	8250	8719.3	254.1198	3min 15s
bier127	118282	154329	163216.6	3029.44	4min 25s
d198	15780	41762	52251	9609	2min 07s
gil262	2378	7632	9243	444.0558	57s
pcb442	50778	201420	212654	6874	5min 37s
pa561	2763	72031	73041.7	2623	2min 54s
rl1889	316536	$4.171830e+06$	$4.532114e+06$	$1.343237e+05$	23m 04s
rl5915	565530	$1.382040e+07$	$1.576210e+07$	$1.420373e+06$	1h36m51s
d15112	$1.573084e+06$	$7.471423e+07$	$8.002034e+07$	$1.871274e+06$	1h 27m

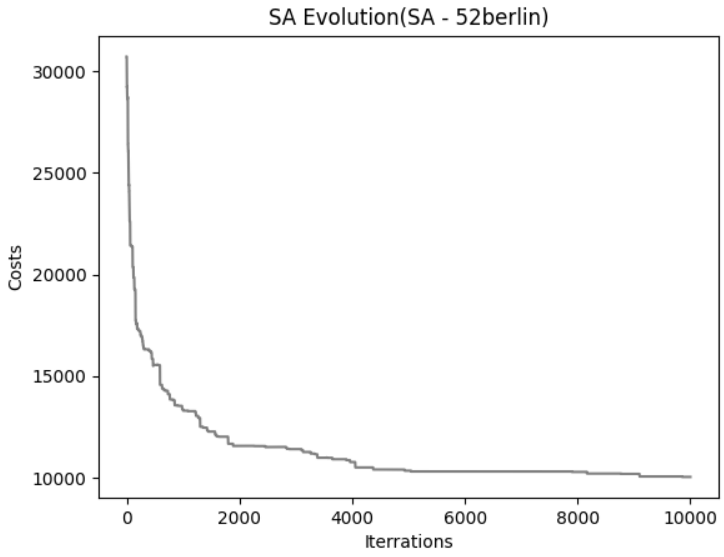
Genetic Algorithm					
Instance	Optimal cost	Best Cost	Mean	StDev	Duration
bayg29	1610	1610	1749	24	1m24s
berlin52	7542	7542	7640	234	4min 35s
bier127	118282	120526	130973.1	3192	4min 24s
d198	15780	15783	16896	263	5min 16s
gil262	2378	2378	2547	87	6min 47s
pcb442	50778	56432	60213	4520	26min 06s
pa561	2763	28436	3276.7	4163.0741	17min 57s
rl1889	316536	665423	865423	342567	1h 24m
rl5915	565530	764522	864520	123421	2h 17m
d15112	$1.573084e+06$	$2.272324e+07$	$2.872321e+07$	689231	7h42m

3.1 Interpretation

For the genetic algorithm we see that it has better results on 10 instances. As we can see, as the instance gets harder, it has more cities and the Genetic algorithm is farther from the optimum.

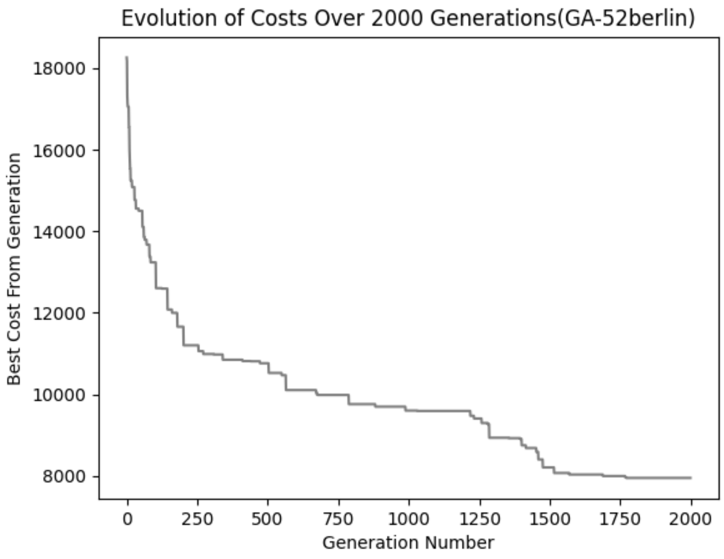
The time complexity for GAs is increasing as the instances are harder but for simulating annealing this isn't happening because of the cooling function.For instance the exponential expression is very small so the probability to generate a smaller number than that expression is very low and our SA starts to have a greedy strategy(choosing the best next solution).

Here we have a graph of evolution for SA :



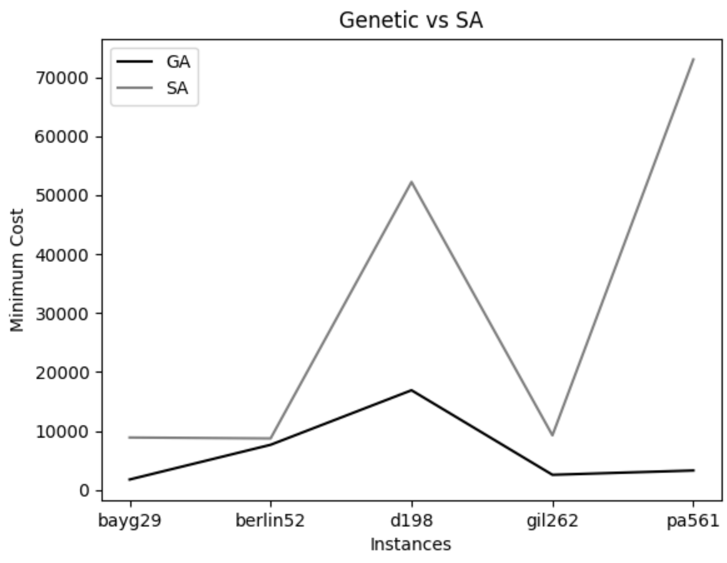
We can see in the figure that our SA starts to converge much faster to optimum but at iteration 5000 it starts to be locked on the plateau with the approximate value 10000.

Here we have a graph of evolution for out Genetic Algorithm :



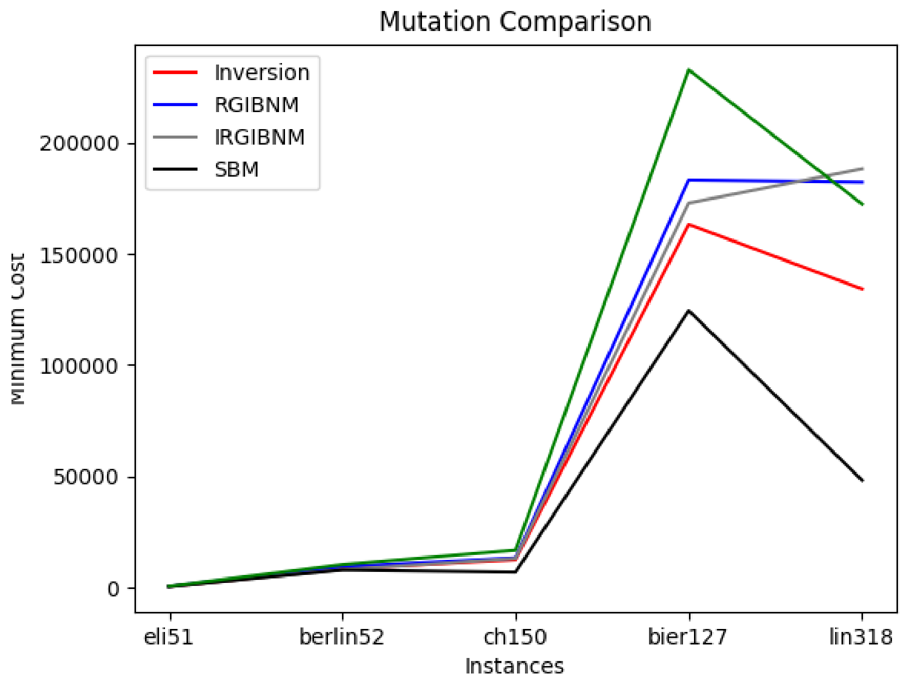
We can say about the upper figure that the GA converges slower than SA to the optimum but it start to reach some plateaus and pass them until a close solutions to the optimum.

Here we can see the values on some instances for both algorithms :



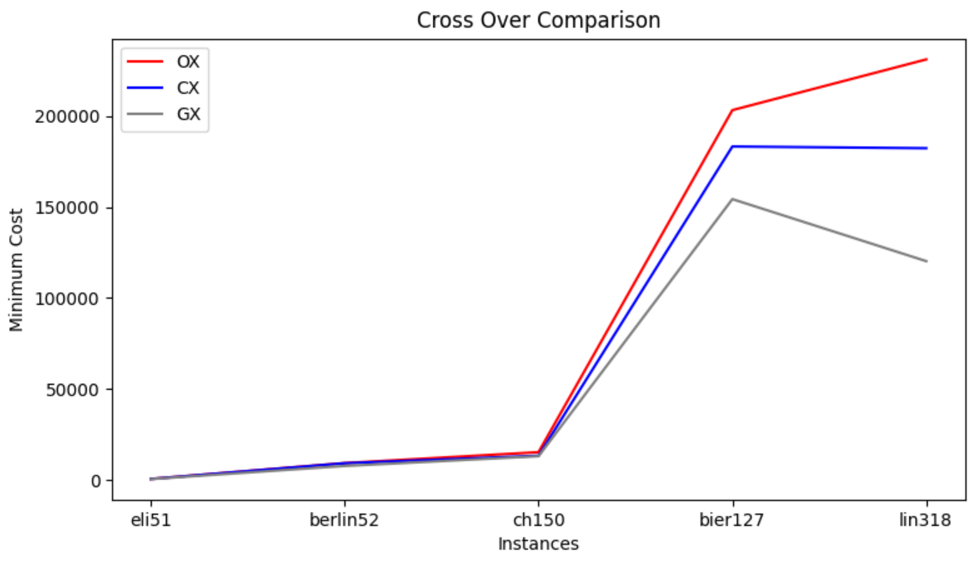
3.2 Comparison Between Mutation Strategies

Our mutation strategies are : Slide Inversion RGIBNM IRGIBNM SBM



As we see in the upper graphs, the best convergence for some instances is for SBM and some mutation are better than others in some cases but as a media SBM looks to be more stable (its errors are not so big).

3.3 Comparison Between Cross Over Strategies



In this picture we can see that for the majority of cases, in our problem GX Cross Over converges better than others to optimum cost.

4 Conclusion

The Genetic Algorithms are very powerful in optimisation problems like Travelling Salesman and for some instances they can find even the optimum in a polynomial time. Our GAs have different strategies, so we can say that they are very diverse and they can be modeled for every instance having a better precision.

In this report we have done a comparison between SA and GA and we can say that SA is powerful especially on small instances but in the evolutive process it has a premature convergence and it is difficult to escape the plateaus. But the GA with some different optimisations can converge far difficult than SA and it can escape the plateaus, improving better over time.

As a conclusion, we must choose every algorithm after the nature of every instance because SA can work better on some instances than GA but overall GA looks to have better results on more instances than SA.

References

[1] Grefenstette J.J. (Ed.) - *Proc. of the First Int. Conf. on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1982

[2] Seminar Genetic Algorithms, author Eugen Croitoru magenta`https://profs.info.uaic.ro/~eugennc/teaching/ga/`.