2017

Final Dossier

FINAL DOSSIER EXPLAINING THE WORK DONE FOR THE SUBJECT

SERGIO MARTÍN SANTANA

SISTEMAS INTELIGENTES E INTERACCION PERSONA COMPUTADOR | Master de Ingeniería Informática

Topic Index

Introduction

In this Project we can show 4 ways to solve the travelling salesman problem or TSP. In the following chapters we'll explain each of the solutions as well as some comparisons between them.

Examples

With the source code we attached two datasets to test the solutions, these datasets are:

- Burma14, that contains 14 cities in Burma with their geographical coordinates.
- FT70, which contains 70 nodes and it describes an asymmetric TSP.

What is the TSP?

The travelling salesman problem (TSP), or, in recent years, the travelling salesperson problem, asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

Described as a graph problem, TSP can be modelled as an undirected weighted graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's weight. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once. Often, the model is a complete graph (i.e. each pair of vertices is connected by an edge). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour.

Metaheuristics (Development)

Hill Climbing.

In numerical analysis, hill climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

Hill climbing achieves optimal solutions in convex problems – otherwise it will find only local optima (solutions that cannot be improved by considering a neighbouring configuration), which are not necessarily the best possible solution (the global optimum) out of all possible solutions (the search space). Examples of algorithms that solve convex problems by hill-climbing include the simplex algorithm for linear programming and binary search. To attempt overcoming being stuck in local optima, one could use restarts (i.e. repeated local search), or more complex schemes based on iterations (like iterated local search), or on memory (like reactive search optimization and tabu search), or on memory-less stochastic modifications (like simulated annealing).

The relative simplicity of the algorithm makes it a popular first choice amongst optimizing algorithms. It is used widely in artificial intelligence, for reaching a goal state from a starting node. Choice of next node and starting node can be varied to give a list of related algorithms. Although more advanced algorithms such as simulated annealing or tabu search may give better results, in some situations hill climbing works just as well. Hill climbing can often produce a better result than other algorithms when the amount of time available to perform a search is limited, such as with real-time systems, so long as a small number of increments typically converges on a good solution (the optimal solution or a close approximation). At the other extreme, bubble sort can be viewed as a hill climbing algorithm (every adjacent element exchange decreases the number of disordered element pairs), yet this approach is far from efficient for even modest N, as the number of exchanges required grows quadratically.

Hill climbing is an anytime algorithm: it can return a valid solution even if it's interrupted at any time before it ends.

Pseudocode

```
currentNode = startNode;
   loop do
      L = NEIGHBORS(currentNode);
      nextEval = -INF;
      nextNode = NULL;
      for all x in L
         if (EVAL(x) > nextEval)
              nextNode = x;
              nextEval = EVAL(x);
      if nextEval <= EVAL(currentNode)
         return currentNode;
      currentNode = nextNode;
```

Results

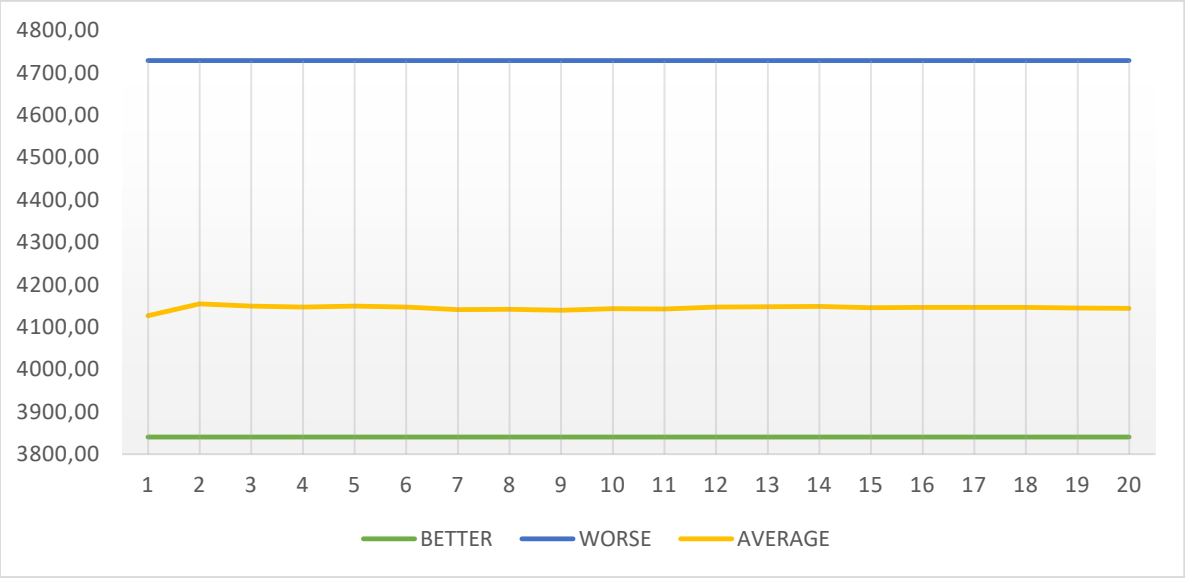
In that case the example is quite small, only 14 cities, and if iterate 100 times on the same algorithm and problem the probability to start in the same cities is very high. That is why the values better and worse, are the same for all executions.

BETTER	3841,00	3841,00	3841,00	3841,00	3841,00
WORSE	4729,00	4729,00	4729,00	4729,00	4729,00
AVERAGE	4126,69	4154,64	4149,37	4146,99	4149,29

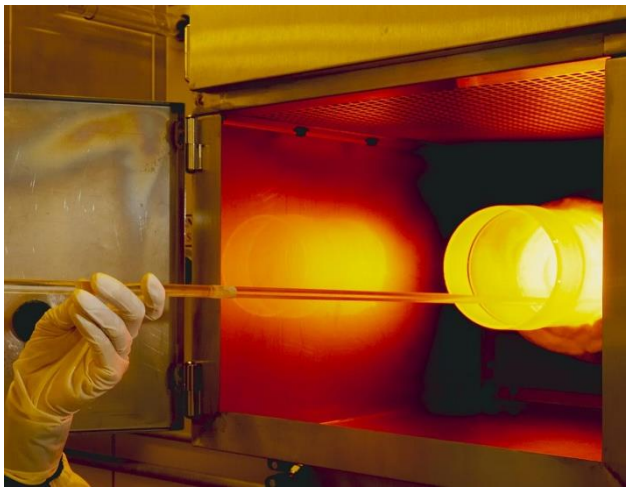
BETTER	3841,00	3841,00	3841,00	3841,00	3841,00
WORSE	4729,00	4729,00	4729,00	4729,00	4729,00
AVERAGE	4147,38	4141,58	4141,72	4139,93	4143,65

BETTER	3841,00	3841,00	3841,00	3841,00	3841,00
WORSE	4729,00	4729,00	4729,00	4729,00	4729,00
AVERAGE	4146,28	4146,66	4146,70	4145,03	4144,45

BETTER	3841,00	3841,00	3841,00	3841,00	3841,00
WORSE	4729,00	4729,00	4729,00	4729,00	4729,00
AVERAGE	4142,98	4147,14	4148,17	4148,50	4145,92



Simulated annealing.



Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to alternatives such as gradient descent.

The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Both are attributes of the material that depend on its thermodynamic free energy. Heating and cooling the material affects both the temperature and the thermodynamic free energy. The simulation of annealing as an approach that reduces a minimization of a function of large number of variables to the statistical mechanics of equilibration (annealing) of the mathematically equivalent artificial multiatomic system[jargon] was first formulated by Armen G. Khachaturyan, Svetlana V. Semenovskaya, Boris K. Vainshtein in 1979 and by Armen G. Khachaturyan, Svetlana V. Semenovskaya, Boris K. Vainshtein in 1981. These authors used computer simulation mimicking annealing and cooling of such a system to find its global minimum.

This notion of slow cooling implemented in the Simulated Annealing algorithm is interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored (accepting worse solutions is a fundamental property of metaheuristics because it allows for a more extensive search for the optimal solution). The simulation can be performed either by a solution of kinetic equations for density functions (A. G. Khachaturyan, S.V. Semenovskaya, B.K.Vainstein in

1979 and A. Khachaturyan, S. Semenovsakaya, B. Vainshtein in 1981) or by using the stochastic sampling method which was independently described by Scott Kirkpatrick, C. Daniel Gelatt and Mario P. Vecchi in 1983, by Vlado Černý in 1985 and by Svetlana V. Semenovskaya, Karen A. Khachaturyan and Armen G. Khachaturyan in 1985. The method is an adaptation of the Metropolis–Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, invented by M.N. Rosenbluth and published by N. Metropolis et al. in 1953.

Pseudocode

```
Let s = s0
For k = 0 through kmax (exclusive):
    T ← temperature(k / kmax)
    Pick a random neighbour, snew ← neighbour(s)
    If P(E(s), E(snew), T) ≥ random(0, 1):
        s ← snew
return s
```

Results

BETTER	3841	3841	3841	3841	3841
WORSE	4729	4729	4729	4729	4729
AVERAGE	4126,69	4154,64	4149,37	4146,99	4149,29

BETTER	3841	3841	3841	3841	3841
WORSE	4729	4729	4729	4729	4729
AVERAGE	4147,38	4141,58	4141,72	4139,93	4143,65

BETTER	3841	3841	3841	3841	3841
WORSE	4729	4729	4729	4729	4729
AVERAGE	4142,98	4147,14	4148,17	4148,50	4145,92

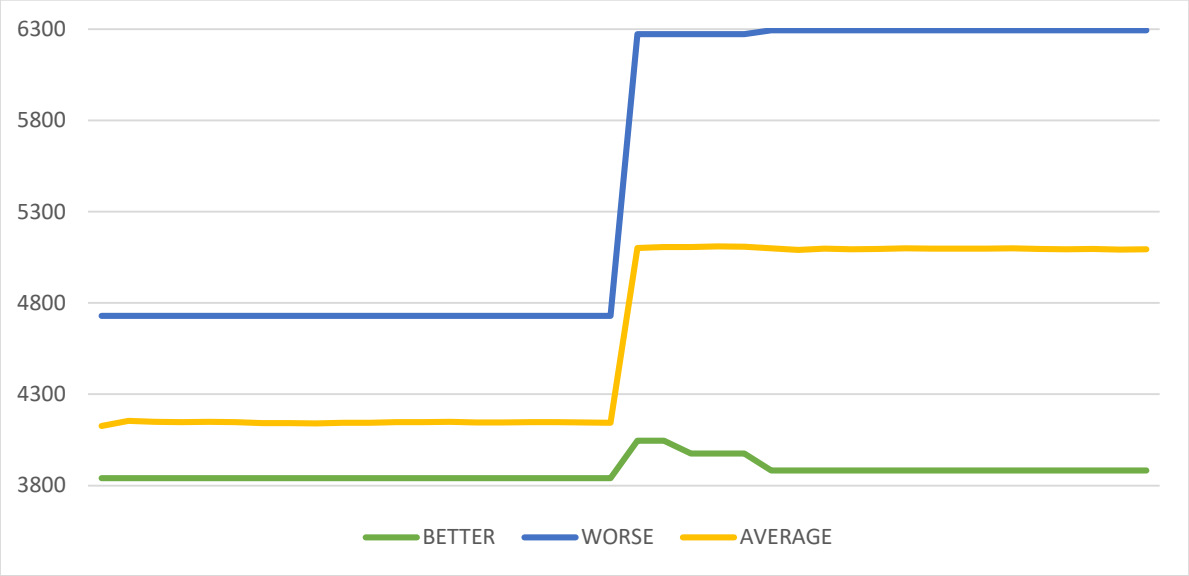
BETTER	3841	3841	3841	3841	3841
WORSE	4729	4729	4729	4729	4729
AVERAGE	4146,28	4146,66	4146,70	4145,03	4144,45

BETTER	4046	4046	3975	3975	3975
WORSE	6272	6272	6272	6272	6272
AVERAGE	5101,54	5107,06	5105,64	5109,14	5107,75

BETTER	3882	3882	3882	3882	3882
WORSE	6294	6294	6294	6294	6294
AVERAGE	5099,78	5090,20	5098,30	5093,86	5095,31

BETTER	3882	3882	3882	3882	3882
WORSE	6294	6294	6294	6294	6294
AVERAGE	5099,30	5098,33	5097,34	5097,11	5098,99

BETTER	3882	3882	3882	3882	3882
WORSE	6294	6294	6294	6294	6294
AVERAGE	5096,41	5094,69	5094,98	5091,65	5093,84



Variable neighborhood search.

Variable neighborhood search (VNS), proposed by Mladenović, Hansen, 1997,[2] is a metaheuristic method for solving a set of combinatorial optimization and global optimization problems. It explores distant neighborhoods of the current incumbent solution, and moves from there to a new one if and only if an improvement was made. The local search method is applied repeatedly to get from solutions in the neighborhood to local optima. VNS was designed for approximating solutions of discrete and continuous optimization problems and according to these, it is aimed for solving linear program problems, integer program problems, mixed integer program problems, nonlinear program problems, etc.

Pseudocode

```
neighborhood = 0
loop (neighborhood < maxNeighborhood)
    shakedSolution = randomSolution()
    simulatedAnnealingSolution = solveSimulatedAnnealing(shakedSolution)
    if(cost(simulatedAnnealingSolution) < costTour(finalSolution))
        finalSolution = simulatedAnnealingSolution
        neighborhood = 0
    else
        neighborhood++
```

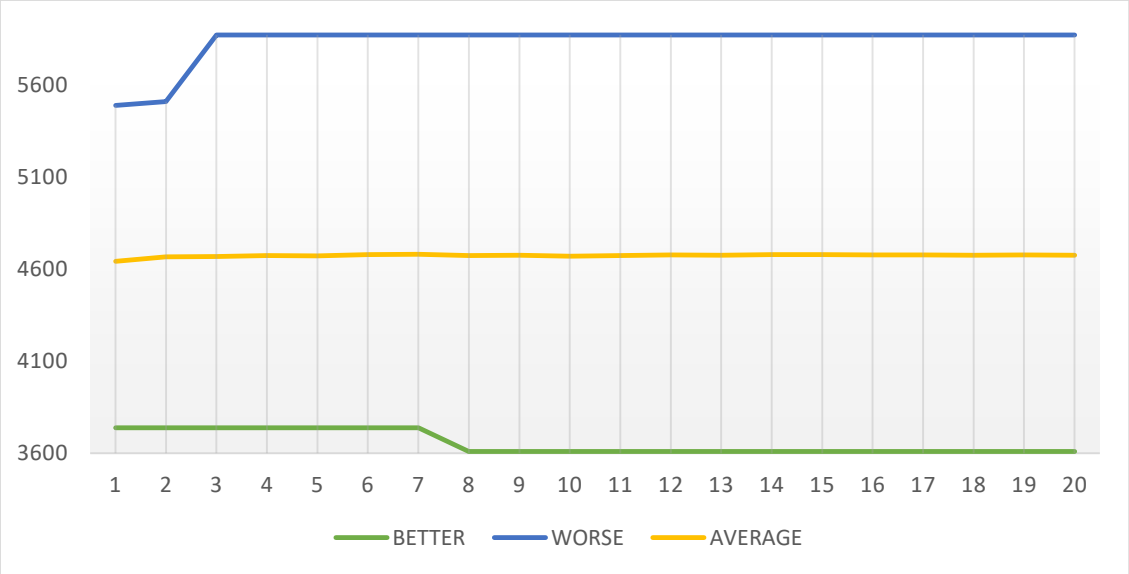
Results (Max. Neighborhood = 2)

BETTER	3739	3739	3739	3739	3739
WORSE	5488	5508	5870	5870	5870
AVERAGE	4641,10	4665,25	4668,24	4672,21	4671,36

BETTER	3739	3739	3610	3610	3610
WORSE	5870	5870	5870	5870	5870
AVERAGE	4678,66	4680,55	4672,08	4674,74	4669,40

BETTER	3610	3610	3610	3610	3610
WORSE	5870	5870	5870	5870	5870
AVERAGE	4672,54	4676,61	4674,35	4677,51	4678,19

BETTER	3610	3610	3610	3610	3610
WORSE	5870	5870	5870	5870	5870
AVERAGE	4676,58	4676,29	4674,80	4675,68	4675,19



Taboo search.

Taboo or Tabu search, created by Fred W. Glover in 1986 and formalized in 1989, is a metaheuristic search method employing local search methods used for mathematical optimization.

Local (neighborhood) searches take a potential solution to a problem and check its immediate neighbors (that is, solutions that are similar except for very few minor details) in the hope of finding an improved solution. Local search methods have a tendency to become stuck in suboptimal regions or on plateaus where many solutions are equally fit.

Tabu search enhances the performance of local search by relaxing its basic rule. First, at each step worsening moves can be accepted if no improving move is available (like when the search is stuck at a strict local minimum). In addition, prohibitions (henceforth the term tabu) are introduced to discourage the search from coming back to previously-visited solutions.

The implementation of tabu search uses memory structures that describe the visited solutions or user-provided sets of rules. If a potential solution has been previously visited within a certain short-term period or if it has violated a rule, it is marked as "tabu" (forbidden) so that the algorithm does not consider that possibility repeatedly.

Pseudocode

```
currentSolution ← HillClimbing(U);
finalSolution ← currentSolution;
do
    currentSolution ← newSolution(currentSolution,Tabu);
    if(cost(currentSolution) < cost(finalSolution))
        finalSolution ← currentSolution;
    update(Tabu)
while(noCriPar)
```

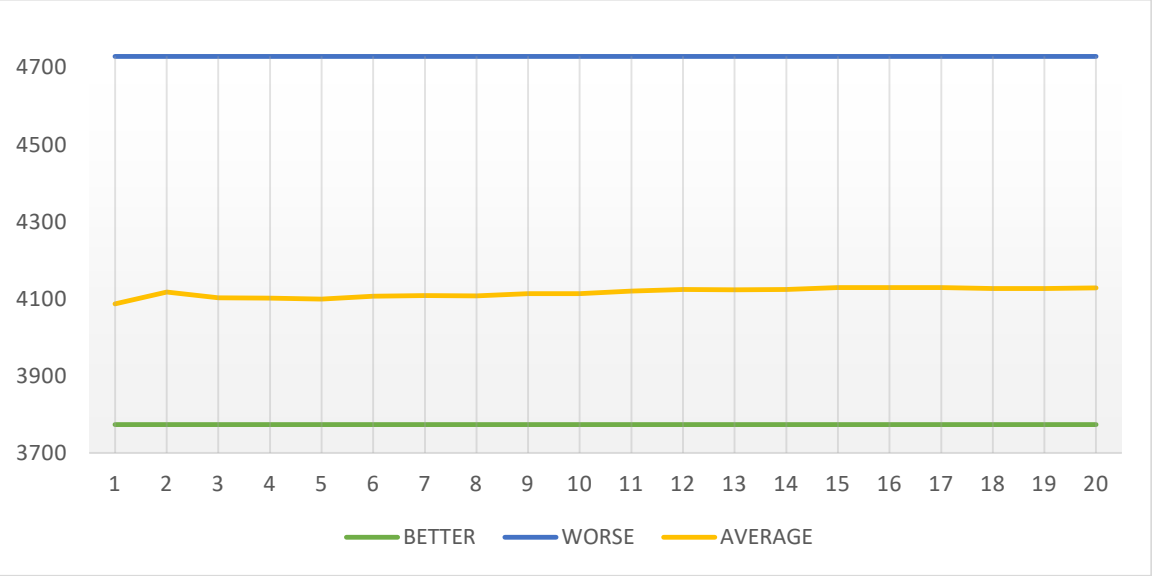
Results

BETTER	3774	3774	3774	3774	3774
WORSE	4729	4729	4729	4729	4729
AVERAGE	4086,52	4117,65	4102,96	4101,40	4099,55

BETTER	3774	3774	3774	3774	3774
WORSE	4729	4729	4729	4729	4729
AVERAGE	4106,47	4108,06	4107,48	4113,43	4113,35

BETTER	3774	3774	3774	3774	3774
WORSE	4729	4729	4729	4729	4729
AVERAGE	4120,35	4123,94	4123,28	4124,42	4128,88

BETTER	3774	3774	3774	3774	3774
WORSE	4729	4729	4729	4729	4729
AVERAGE	4128,89	4129,52	4126,48	4126,56	4128,36



Local search modification.

In our source code it's also call as 2-opt, and it consists in a simple local search algorithm for solving the traveling salesman problem, this algorithm take a route that crosses over itself and reorder it so that it does not.

Pseudocode

```
finalSolution = generateSolution()
loop (!stopCondition){
    newSolution = generateNewSolution()
    if(newSolution > finalSolution){
        finalSolution = newSolution
    else
        if(probability)
            finalSolution = newSolution
```

Comparatives

Between solutions

Now we get an averaged between the values obtained in each specific study, to know which values are the best in general. And that values are:

Solution	Better	Worse	Average
Hill Climbing	3841,00	4729,00	4145,19
Simulated Annealing	3876,67	5508,75	4621,86
Variable Neighborhood Search	3655,15	5832,80	4672,56
Taboo Search	3774,00	4729,00	4115,88

After this research we see that the best solution to solve this problem of the Tabu search, since it assures us that in the worst case, it will be the least cost, and that the average solution is the least cost. As an alternative would be VNS, but of course, we would only be a winner if we happen to get the minimum value of Better.

Modifying Neighborhood variable.

Now we can test if the value of the maximum neighborhood affects the best, worst, mean values, and to what extent.

Max Neighborhood	Average	Cost Reduction
2	4672,57	---
4	4543,03	-2.77%
8	4420,82	-5.39%

As we see, by increasing the neighborhood value, the average cost of our solution is reduced by a slight percentage. So we can determine that if the neighborhood is increased, the cost is reduced so we can determine that Neighborhood and Cost are inversely proportional

References

- Metaheurísticas.GitHub.2016. Available in: github.com/oxcar103/Metaheuristicas
- MP-TESTDATA - The TSPLIB Symmetric Traveling Salesman Problem Instances. Available in: elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
- TSP. Wikipedia Available in: en.wikipedia.org/wiki/Travelling_salesman_problem
- Hill Climbing. Wikipedia. Available in: en.wikipedia.org/wiki/Hill_climbing
- Simulated Annealing. Wikipedia. Available in: en.wikipedia.org/wiki/Simulated_annealing
- Variable Neighborhood Search. Wikipedia. Available in: en.wikipedia.org/wiki/Variable_neighborhood_search
- Tabu search. Wikipedia. Avalaible in: en.wikipedia.org/wiki/Tabu_search