Intelligent Systems

UCA Universidad de Cádiz

Unit 2
Trajectory-based Metaheuristics

Dra. Elisa Guerrero Vázquez
elisa.guerrero@uca.es
University of Cadiz - Spain

Contents

1. Introduction
    1. Classical Methods for Combinatorial optimization problems
    2. Metaheuristics
2. Annealing Search
3. Tabu Search

1.1 Local Search

For combinatorial optimization problems

1. Start with initial configuration
2. Repeatedly search neighborhood (Successors) and select the best neighbor as candidate
3. Apply a cost function (or fitness function) and accept candidate if it is better than current
4. Stop if quality is sufficiently high, if no improvement can be found or after some fixed time

- Candidate is always and only accepted if cost is lower than current configuration
- Stop when no neighbor with lower cost (higher fitness) can be found

1.1 Implementation

1. A method to generate initial configuration
2. A Successor function to generate new states
3. A Cost function
4. A Decision Criterion to select next candidates from the list of successors
5. A Stop Criterion

1.1 Classical search algorithms

- The cost of finding the optimal solution can be very high (Backtracking, AC3, etc.)

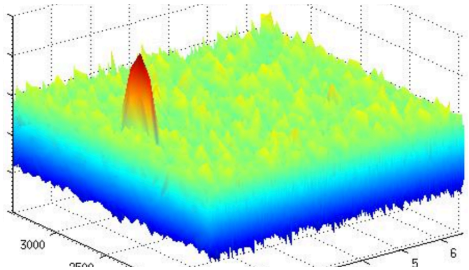
- They do not guarantee finding the optimal solution: high probability of falling in local max/minimal (Local Search)

⇨ Alternative: **Metaheuristics** that allow escaping from local optima

1.2 Improving local search

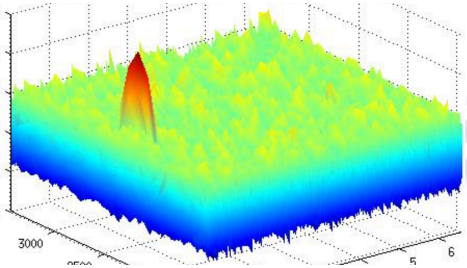
- Repeat algorithm many times with different initial configurations
- Use information gathered in previous runs
- Use a more complex Successor Function to jump out of local optimum
- Use a more complex Decision Criterion that sometimes could accept other alternative solutions (even worse than current)

1.2 Metaheuristics



Metaheuristics are high-level search strategies that may provide a sufficiently good solution to an optimization problem

1.2 Metaheuristics

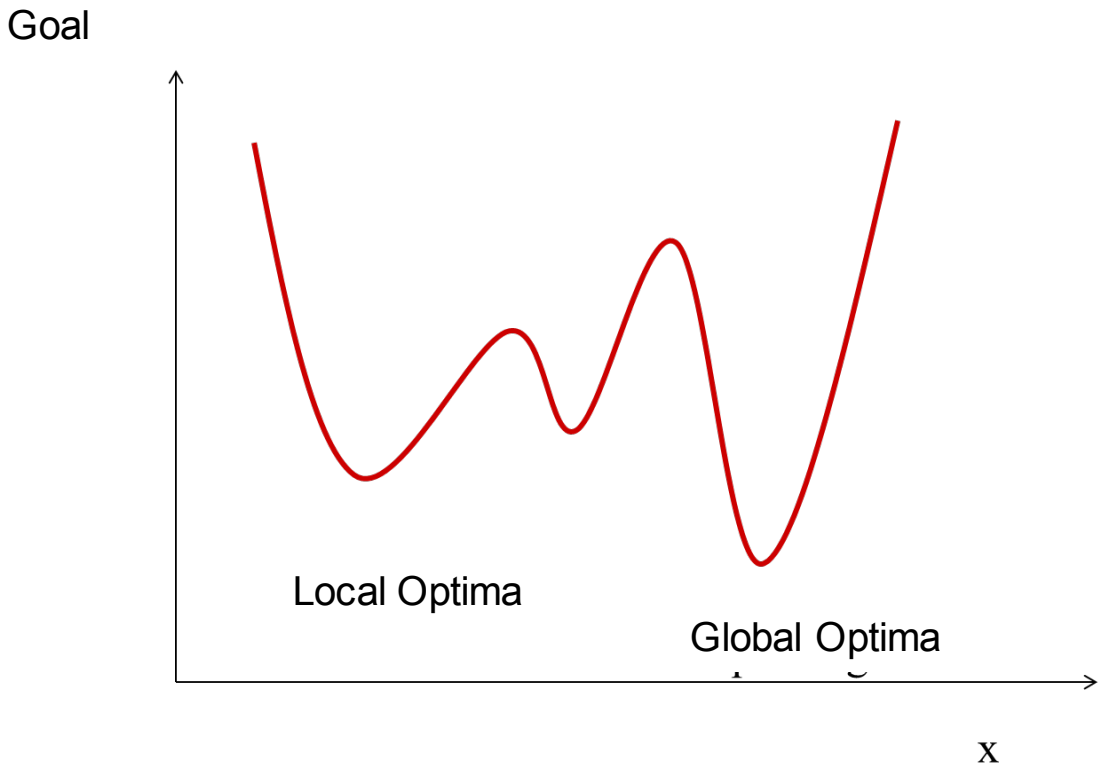


- The goal is to efficiently explore the search space in order to find (near)optimal solutions.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- Metaheuristics are not problem-specific.
- Metaheuristics may make use of domain-specific knowledge

1.2 Trajectory search Methods

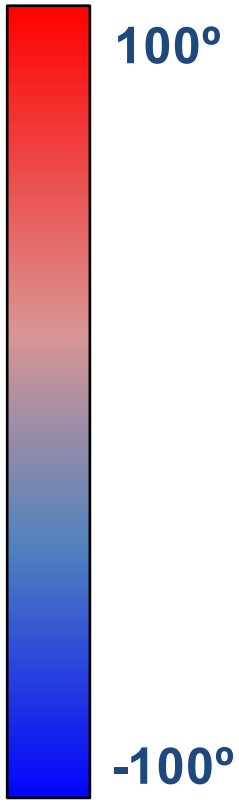
Describe the trajectory in the search space through the execution

- Simulated Annealing
- Tabu Search

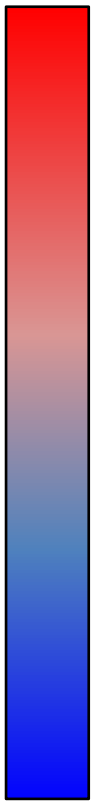


2. Simulated Annealing

- Metaheuristic search method based on local search methods
- Based upon the **analogy** with the simulation of the annealing of solids
- Do sometimes ***accept candidates with higher cost*** to escape from local optimum



2.1 Annealing Process



- **Raising the temperature up** to a very high level (melting temperature, for example), the atoms have a higher energy state and ***a high possibility to re-arrange the crystalline*** structure.

- **Cooling down slowly**, the atoms have a *lower and lower energy state* and a smaller and smaller possibility to re-arrange the crystalline structure.

2.2 SA vs. Local Search

In comparison with Local Search

- SA only evaluates some successors
  - If the New state is better then is taken
  - If the New state is worse a likelihood function will decide if this state is accepted

- At the beginning this likelihood is very high but decreases as the search progresses:
  - Temperature is related to this likelihood.
  - At the beginning T is very high, and the likelihood of considering different options in the state space is very high, but as the search progresses T is going down, meaning that the process is converging to the optimal solution, then bad states are not very likely to be considered.

2.3 SA Algorithm v1

```
Algorithm SA
    T=upper limit
    Current= random initial state
    while (T does not reach lower limit)
        New = random Successor from Current
        if New improves Current
                Current=New
        else
                Accept New with probability P
        end_if
        cool(T)
    end_while

    Return Current
End SA
```

2.4 SA: Probability Function

- The probability of updating the state depends on the extent to which one solution is worse than the other:
  - ΔE is the difference between cost values of the current and new states
  - *T:* temperature of the system that is progressively decreasing

$$p = \exp\left(\frac{-\Delta E}{T}\right)$$

$$\Delta E = fcost(New) - fcost(Current)$$

** *when we are trying minimizing the cost function*

2.5 Influence of T

- Suppose a generic problem in which the values returned by the cost functions of the current state and the new successor are as follows:

  fcost(Current)=15

  fcost(New)=19

$$\Delta E = fcost(New) - fcost(Current)$$

$$p = \exp\left(\frac{-\Delta E}{T}\right)$$

T	P
1000	
100	
50	
30	
10	
1	

2.5 Influence of T

- Suppose a generic problem in which the values returned by the cost functions of the current state and the new successor are as follows:

  fcost(Current)=15

  fcost(New)=19

$$\Delta E = fcost(New) - fcost(Current)$$

$$p = \exp\left(\frac{-\Delta E}{T}\right)$$

When T approaches zero, the algorithm will only accept states with a better evaluation function than the current state

T	P
1000	0.9960
100	0.9608
50	0.9231
30	0.8752
10	0.6703
1	0.0183

2.6 SA Algorithm v2

```
Algorithm SA
    Initialize T, T_min
    Current= random state
    while (T>T_min) & (No Solution)
        New = Random Successor from Current
        deltaE = fcost(New) - fcost(Current)
        if (deltaE<0)   %% if candidate is better is accepted directly
                Current=New
        else
                p=exp(-deltaE/T)
                if p>rand(0, 1)  %% accept solution randomly
                        Current=New
                end_if
        end_if
        cool(T)
    end_while
    Return Current
```

2.7 Parameter Analysis

- If T is very high, convergence is very slow
- If T is very low, procedure will not converge
- T_min can be replaced by the maximum number of iterations
- The function *cool(T)* decreases in each iteration
  - Depends upon the problem
  - Lineal decreasing is the most common
  - Other options: exponential decreasing, Cauchy, etc.

$$Cauchy \quad T_i = \frac{T_o}{1+i}$$

2.8 Performance of Simulated Annealing

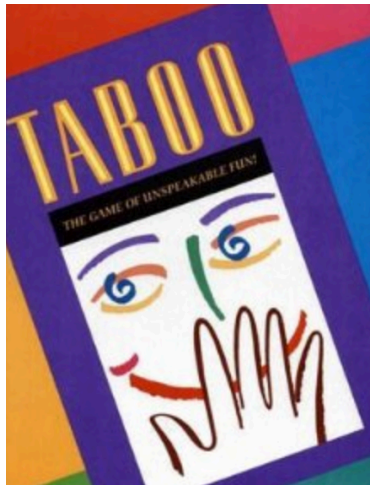
- SA is a general solution method that is easily applicable to a large number of problems
- "Tuning" of the parameters (initial T, decrement of T, stop criterion) is relatively easy
- Generally the quality of the results of SA is good, although it can take a lot of time

2.8 Performance of Simulated Annealing

- Results are generally not reproducible: another run can give a different result
- SA can leave an optimal solution and not find it again (so try to remember the best solution found so far)
- Proven to find the optimum under certain conditions; one of these conditions is that you must run forever

3. Tabu Search

Metaheuristic search method based on local search methods



Tabu states are states already visited or actions performed recently.

Tabu states are introduced to discourage the search from coming back to previously-visited solutions.

3.1 Basis

- In each iteration
  - the best Successor (no Tabu) is selected
    - Current is included into the Tabu list for a certain number of iterations
  - If a Successor is Tabu, then ”avoid” it and take the next one

3.2 Main loop

- Runs until an external completion condition is satisfied
- The generated Successor could be worse than Current state
- Adaptive memory: a list of Tabu states is updated, the undesirable states
- Tenure: number of iterations that will keep a state on the Tabu list

3.3 Aspiration Criterion

- Some Successor, even being Tabu, might offer an exceptionally good solution.
- The aspiration criterion allows you to check whether a Tabu state is better than the best state, and if so, to select it, by updating the corresponding Tabu list.

3.4 Algorithm v1

1. Generate the ordered list of Succesors from Current
2. Select the first (best) one as candidate
   a. Remove Current from Successors list

Aspiration criterion

2.1 If the selected candidate is best than the Better one

   a. Select this state as Current
   b. Include Current in the Tabu list
   c. and go to 1

Tabu List

2.2 If the selected candidate is not best than the Better one

- AND it is NO Tabu:
  a. select this state as Current,
  b. Include Current in the Tabu list
  c. and go to step 1
- AND it is TABU:
  a. go to step 2

3.5 Stopping Criteria

- Number of iterations
- Reach an acceptable solution
- Be stuck in a local optimum
- There are no alternatives for a Current state (Successors are worse and all of them are in the Tabu list)

3.6 Algorithm v2

```
Algorithm Tabu_Search
    Current= random or initial state
    Best=Current
    Initial values for tabu list, tenure, iteration counter and max number of iterations
    While <stop conditions>
        Obtain the list of succesor states ordered by cost function
        While <successor list not empty and not a new current state is generated >
            New=Next(Sucessor list)
            if Cost(New) < Cost(Best) %% Tabu or not tabu but improves the best
                Current=Sucessor
                Best=Current
            elsif New is Not Tabu
                Current=Sucessor
            else
                Take next sucessor from the ordered list of sucessors
            endif
            Update tabu list (decreasing tenure and adding new tabu state)
        endWhile
        Update rest of variables
    endWhile
end
```

4. Conclusions

Both strategies:

- Use Metaheuristics to guide the search and avoid local optima
- Are based on one single solution (vs. poblaciones)
- Are trajectory-based strategies:
  - Iterative procedure
  - Successor function must be implemented

References

- Russell, S. y Norvig, P. *Inteligencia Artificial (un enfoque moderno) (Pearson Educación, 2004). Segunda edición.* Cap. 5: "Problemas de Satisfacción de Restricciones"
- García Serrano, A. *Inteligencia artificial. Fundamentos, práctica y aplicaciones* (RC Libros, 2012)
- Michalewicz, Z. y Fogel D. *How to solve it. Modern Heuristics* (Springer, 2004)
- Tim Jones, J. *Artificial Intelligence. A systems approach.* Computer Sciences Series. (Jones & Bartlett publishers, 2008)