

# Solving TSP: Tabu Search and code examples

Shan He

School for Computational Science  
University of Birmingham

Module 06-27818 and 27819: Advanced Aspects of  
Nature-Inspired Search and Optimisation (Ext)

# Outline of Topics

1 Code example

2 Tabu search

## Code example: Generating solutions for TSP

- Download the source code from Canvas
- `load('cities.mat')`
- Open the matrix 'cities', which is a  $2 \times 48$  cities TSP problem
  - 48 USA state capital cities
  - The minimal tour has length 10628.
  - Check this [page](#)
- Open the optimal solution 'att48.s.txt' file and copy
- `optimalsolution = [ paste the solution ]`
- `inputcities = cities(:,optimalsolution)`
- `distance(inputcities)`
- Read help file about how to generate random permutation:  
`help randperm`

## Tip: How to implement Simulated Annealing

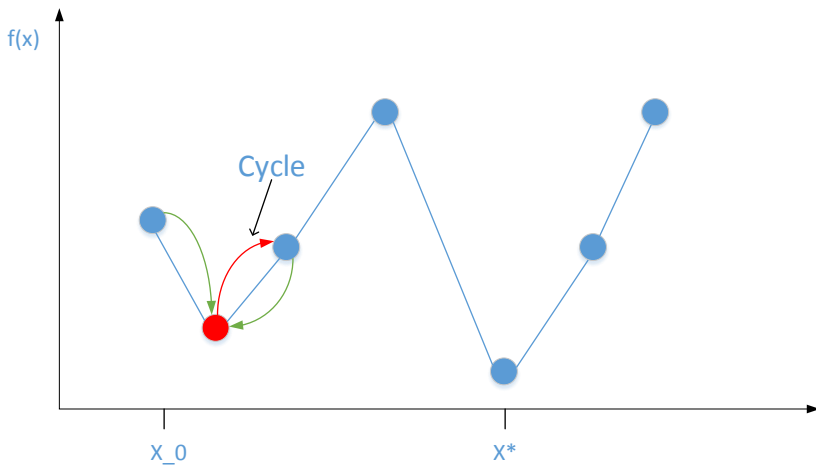
- The best starting point is a local search algorithm, e.g., hill climbing
- Open `simple_hill_climbing_two_opt.m'` file
- Complete the code to get a hill climbing algorithm for TSP
- Implement the random **non-improving** step (Lecture 4, pages 17-18)

## Question

- We have learned Simulated Annealing which can escape from local optima by accepting worse solutions with some probability.
- **Question:** is there any other strategies to find better local optima?

*Avoid cycle*

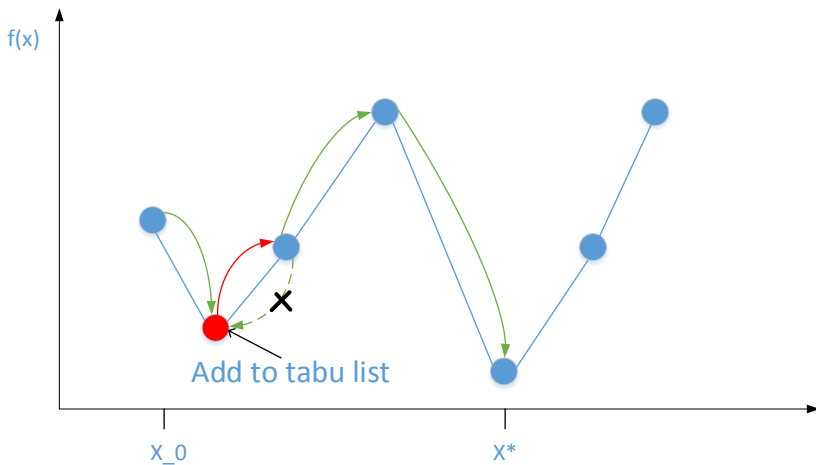
## Search trajectory: cycles



# Tabu search

- Invented by Professor Fred W. Glover in 1986 and formalized in 1989
- Many applications since publications
- ‘Tabu’ means “things that cannot be touched because they are sacred”
- Main idea: use **memory** to guide local search process away from local optima
  - Maintains a memory structure called **tabu list** that memorises previously visited solutions
  - **Forbids** the local search algorithm immediately return to **previously visited solutions**

## Search trajectory: Tabu search





## More about Tabu list

- Tabu list consists of: *metaheuristic*
  - banned solutions; or
  - a set of rules to ban solutions → *ML*
- Use tabu list to exclude some neighborhood solutions for local search  
*only remember recently visited solutions*
- Essentially construct a neighborhood  $N^*(x)$  solutions to be explored
- The simplest Tabu list:
  - Recently visited solutions
  - The duration of memory (in search steps) called Tabu tenure
  - Rule out any search attempts that would lead back to those previously visited solutions
- Extension: maintain a tabu list to avoid unfavourable neighbourhood solutions

# Problem: How to construct a Tabu list for TSP

- We aim to solve TSP using tabu search
- The key component: Tabu list
- Assuming the we use the 2-OPT algorithm for local search
- Question: How to design a tabu list?

# Tabu search algorithm pseudocode

## Tabu search algorithm

```
while (terminationflag != true)
    Determine set  $N^*(x)$  of non-tabu neighbours of  $x$ 
     $x_{new} = LocalSearch(N^*(x))$ 
    Update tabu list based on  $x_{new}$ 
     $x = x_{new}$ 
Output  $x^i$ 
```

## Take home message

- Main idea: escape or **avoid** local optima
- Tabu search tutorial: [here](#)