Algorithms and Complexity theory

Thibaut Barthelemy
Some slides kindly provided by Fabien Tricoire

University of Vienna

WS 2014
Outline

1 Algorithms
   - Overview
   - How to write an algorithm

2 Complexity theory

3 3 NP-hard problems - Heuristics needed
   - The Traveling Salesman Problem
   - The Parallel Machines Scheduling Problem
   - The Warehouse Location Problem

4 Introducing Heuristics
   - Introduction
   - Construction and Improvement
   - Limitations of “simple” heuristics

5 Introducing Metaheuristics
   - Etymology and Historical Aspects
   - Evaluating Metaheuristics
Outline

1 Algorithms
   • Overview
   • How to write an algorithm

2 Complexity theory

3 NP-hard problems - Heuristics needed
   • The Traveling Salesman Problem
   • The Parallel Machines Scheduling Problem
   • The Warehouse Location Problem

4 Introducing Heuristics
   • Introduction
   • Construction and Improvement
   • Limitations of “simple” heuristics

5 Introducing Metaheuristics
   • Etymology and Historical Aspects
   • Evaluating Metaheuristics
Computers are not clever

Finding the minimum of function $f(x)$

What can you do?

✓ can see the whole curve
✓ can identify immediately the minimum

What can a computer do?

× cannot see the whole curve
× can only see pairs of points: $f(x_1)$ vs. $f(x_2)$
Computers are fast

Finding the minimum of function $f(x)$

What can a computer do?

- × can only see pairs of points: $f(x_1)$ vs. $f(x_2)$
- ✓ can compare and manipulate million pairs per second

What can you tell a computer to do?

Exercise: imagine an algorithm to find the minimum.
An exact algorithm

Finding the minimum of function $f(x)$ with $x \in \{0 \ldots 100\}$

$x^* \leftarrow 0$
$x \leftarrow 1$

while $x \leq 100$ do
    if $f(x) < f(x^*)$ then
        $x^* \leftarrow x$
    end if
    $x \leftarrow x + 1$
end while
Complexity of algorithms

Finding the minimum of function $f(x)$ with $x \in \{a \ldots b\}$

\begin{verbatim}
x* ← a
x ← a
while $x \leq b$ do
  if $f(x) < f(x^*)$ then
    x* ← x
  end if
  x ← x + 1
end while
\end{verbatim}

How long does the algorithm take?

Exercise: how many steps are done when looking for $x^*$ in $\{a \ldots b\}$

Can we be less accurate but faster?

Exercise: imagine a faster algorithm to find a "good" (low enough) value.
A heuristic algorithm

Finding a "good" point of function $f(x)$ with $x \in \{a \ldots b\}$

$x \leftarrow \text{random integer in } \{a \ldots b\}$

for $i \leftarrow 1$ to $10$ do
  if $f'(x) < 0$ then
    $x \leftarrow x + 1$
  else
    $x \leftarrow x - 1$
  end if
end for

How long does the algorithm take?

Exercise: how many steps are done when looking in $\{a \ldots b\}$

Accuracy and complexity of a heuristic

$\times$ the minimum is rarely found

✓ (very) fast
**Definition**

Algorithm: a succession of operations performing data manipulation.

These data can be:

- **Constants**
  - Cannot be modified
  - Typically: problem input

- **Variables**
  - Can be modified
  - Useful to store computation results

- Both kinds can be arranged into **data structures**
  - Example 1: Lists
  - Example 2: Arrays
Assignment: setting the value of a variable. In this course, the assignment is represented with the following arrow operator: $\leftarrow$

Exercise: what is the value of variable $z$ at the end of this algorithm?

\[
\begin{align*}
x & \leftarrow 3 \\
y & \leftarrow 4 \\
z & \leftarrow \sqrt{x^2 + y^2}
\end{align*}
\]

Answer: 5
Conditionals allow to perform parts of an algorithm only if certain conditions are met. In this course, this is achieved with the keywords `if`, `then` and `else`.

Exercise: what is the value of variable $z$ at the end of this algorithm?

```plaintext
x ← 3
y ← 4
if x < y then
  z ← 1
else
  z ← 2
end if
```

Answer: 1
Algorithms: basic operations

Loops

Repeatedly perform similar operations

Loops allow to perform the same part of an algorithm several times until a given condition is met. In this course, this is achieved with the keywords while, for, do, repeat and until.

Exercise: what is the value of variable \( z \) at the end of this algorithm?

\[
\begin{align*}
x & \leftarrow 0 \\
z & \leftarrow 0 \\
\textbf{while } x < 10 \textbf{ do} \\
& \quad z \leftarrow z + 2 \\
& \quad x \leftarrow x + 1 \\
\textbf{end while}
\end{align*}
\]

Answer: 20
Note: the following algorithm performs an equivalent task on $z$

\[
z \leftarrow 0
\]

\[
\text{for } x \leftarrow 1 \text{ to } 10 \text{ do}
\]
\[
z \leftarrow z + 2
\]

\[
\text{end for}
\]

Exercise: write an equivalent algorithm using `repeat (...) until`

\[
x \leftarrow 0
\]
\[
z \leftarrow 0
\]

\[
\text{repeat}
\]
\[
z \leftarrow z + 2
\]
\[
x \leftarrow x + 1
\]

\[
\text{until } x = 10
\]
Outline

1 Algorithms
   • Overview
   • How to write an algorithm

2 Complexity theory

3 3 NP-hard problems - Heuristics needed
   • The Traveling Salesman Problem
   • The Parallel Machines Scheduling Problem
   • The Warehouse Location Problem

4 Introducing Heuristics
   • Introduction
   • Construction and Improvement
   • Limitations of “simple” heuristics

5 Introducing Metaheuristics
   • Etymology and Historical Aspects
   • Evaluating Metaheuristics
A decision problem: the Hamiltonian Cycle Problem
Complexity Theory

A decision problem: the Hamiltonian Cycle Problem

Diagram: A graph with nodes A, B, C, D, E, and F connected by edges.
A decision problem: the Hamiltonian Cycle Problem

Complexity Theory

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
A decision problem: the Hamiltonian Cycle Problem

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
Complexity Theory

A decision problem: the Hamiltonian Cycle Problem

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
Complexity Theory

A decision problem: the Hamiltonian Cycle Problem

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
A decision problem: the Hamiltonian Cycle Problem

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
Complexity Theory

A decision problem: the Hamiltonian Cycle Problem

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
A decision problem: the Hamiltonian Cycle Problem

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
A decision problem: the Hamiltonian Cycle Problem

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
A decision problem: the Hamiltonian Cycle Problem

How can you tell a computer to find a Hamiltonian cycle?

Exercise: draw a tree showing how an algorithm tries all possibilities until a Hamiltonian cycle is found.
Complexity Theory

The Turing Machine

[Diagram of a Turing machine with labels for parts such as Tractor motor, Print motor, Offset print raised for a "mark", Electric eye looking at tape square, Eraser, Offset-printing + eraser roller, Tractor roller, TAPE, HEAD, Control unit, and a table with states and actions for different tape symbols.]
Complexity Theory

Church-Turing’s thesis

Every problem for which we know a solving way (algorithm) can be solved by a Turing Machine.
Complexity Theory

The Deterministic Turing Machine (like real computers)

Deterministic means both

- it cannot guess the right choice,
- it is in one state at once.

The solution tree of our problem is explored branch by branch:
Complexity Theory

The Non-deterministic Turing Machine (imaginary computer)

Non-deterministic means either
- it can be in several states at once,
- or it can guess the right choice.

So, at a given level, either
- for every node, in parallel, it deploys its branches;
- for the correct node, it chooses among branches the next correct node.
NP is the set of decision problems which can be solved in polynomial time by a non-deterministic machine.
Complexity Theory

Class P

$P$ is the set of decision problems which can be solved in polynomial time by a deterministic machine.
Cook-Levin theorem

Every problem in NP can be reduced to SAT in polynomial time (SAT can solve it).

\[ \text{so SAT is } \text{NP-complete}. \]
Complexity Theory

**Other NP-complete problems**

Many problems are also NP-complete, because

- SAT is reducible to them in polynomial time (they can solve SAT).

![Diagram showing relationships between NP-C, SAT, and other problems]

- Independent Set
- Vertex Cover
- Set Cover
- 3-SAT
- Hamiltonian Cycle
- TSP
- 3-Coloring
- Coloring
- Subset Sum
- Knapsack
- Graph Isomorphism
- Nash Equilibrium
- Factorization
- Spanning Tree
- Max Flow
- 2-SAT
- Assignment
- Primes
- Linear Programming

- P
Other NP-complete problems

Many problems are also NP-complete, because

- SAT is reducible to them in polynomial time (they can solve SAT).
NP-hard problems

- They are not all decisions problems: NP-hard is not in NP.
NP-hard problems

- They are not all decisions problems: NP-hard is not in NP.
- SAT is reducible to any NP-hard problem.
NP-hard problems

- They are not all decisions problems: NP-hard is not in NP.
- SAT is reducible to any NP-hard problem.
- So any problem of NP is reducible to a NP-hard problem.
Proving that your problem is NP-hard = doing reduction

- Let’s consider the Vehicle Routing Problem (VRP).
- Is it NP-hard?
Complexity Theory

Proving that your problem is NP-hard = doing reduction

By definition, if

- a NP-complete problem is reducible to VRP

then VRP is NP-hard.
Proving that your problem is NP-hard = doing reduction

By definition, if

- a NP-complete problem is reducible to VRP
- or a NP-hard problem is reducible to VRP

then VRP is NP-hard.
Proving that your problem is NP-hard = doing reduction

By definition, if

- a NP-complete problem is reducible to VRP
- or a NP-hard problem is reducible to VRP

then VRP is NP-hard.
1. Algorithms
   - Overview
   - How to write an algorithm

2. Complexity theory

3. 3 NP-hard problems - Heuristics needed
   - The Traveling Salesman Problem
   - The Parallel Machines Scheduling Problem
   - The Warehouse Location Problem

4. Introducing Heuristics
   - Introduction
   - Construction and Improvement
   - Limitations of “simple” heuristics

5. Introducing Metaheuristics
   - Etymology and Historical Aspects
   - Evaluating Metaheuristics
Imagine a problem with...

- Some operational constraints
  - Example: visit all customers
- At least one optimal solution
  - Optimise = Minimise or Maximise an Objective function (a.k.a. Fitness)
  - Example: Minimise the sum of operational costs

Such that finding this optimal solution is too hard...

- Even with the best programmer in the world
- Even with the best programming language
- Even with the best operating system
- Even with the fastest computer
- Even 20 years in the future
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...

- Given data: distances using the road network
- Constraints: starting from a given depot, visit all \( n \) customers
- Decision: choose visiting order
- Objective: minimise total distance

An example!
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...

- Given data: distances using the road network
- Constraints: starting from a given depot, visit all $n$ customers
- Decision: choose visiting order
- Objective: minimise total distance

An example!
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...
- Given data: distances using the road network
- Constraints: starting from a given depot, visit all $n$ customers
- Decision: choose visiting order
- Objective: minimise total distance

An example!
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...

- Given data: distances using the road network
- Constraints: starting from a given depot, visit all \( n \) customers
- Decision: choose visiting order
- Objective: minimise total distance

An example!
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...

- Given data: distances using the road network
- Constraints: starting from a given depot, visit all \( n \) customers
- Decision: choose visiting order
- Objective: minimise total distance

An example!
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...
- Given data: distances using the road network
- Constraints: starting from a given depot, visit all \( n \) customers
- Decision: choose visiting order
- Objective: minimise total distance

An example!
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...  
- Given data: distances using the road network  
- Constraints: starting from a given depot, visit all $n$ customers  
- Decision: choose visiting order  
- Objective: minimise total distance

An example!
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...

- Given data: distances using the road network
- Constraints: starting from a given depot, visit all $n$ customers
- Decision: choose visiting order
- Objective: minimise total distance

An example! Basic approach: enumerate all solutions
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...
- Given data: distances using the road network
- Constraints: starting from a given depot, visit all $n$ customers
- Decision: choose visiting order
- Objective: minimise total distance

An example! Basic approach: enumerate all solutions
The Travelling Salesman Problem (TSP)

A Travelling Salesman wants to sell his goods to customers...

- Given data: distances using the road network
- Constraints: starting from a given depot, visit all $n$ customers
- Decision: choose visiting order
- Objective: minimise total distance

Until which size can a computer enumerate all solutions?
The Traveling Salesman Problem (TSP)

Exercise: how many possible ways to visit \( n \) customers?

- A solution is a permutation!
- Number of solutions = number of permutations = \( n! \)
  \( \left( \frac{n!}{2} \right) \) if the distances are symmetric

Suppose our recent computer can process 2,000,000 solutions per second...

- 10 customers: 1.8 s
- 12 customers: 4 m
- 15 customers: 7.5 d
- 20 customers: 38,547 years
- 99 customers: \( 1.48 \cdot 10^{142} \) years! (Universe is \( 5 \cdot 10^9 \) y.o.)

Now suppose we have a computer a million times faster:

- 20 customers: 2 weeks
- 22 customers: 18 years
- 25 customers: 245,760 years
The Parallel Machines Scheduling Problem \((P||C_{max})\)

We want to perform \(n\) production jobs, we have \(m\) “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example!
The Parallel Machines Scheduling Problem \( (P||C_{\text{max}}) \)

We want to perform \( n \) production jobs, we have \( m \) “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example!
The Parallel Machines Scheduling Problem ($P||C_{max}$)

We want to perform $n$ production jobs, we have $m$ “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example!
The Parallel Machines Scheduling Problem ($P||C_{max}$)

We want to perform $n$ production jobs, we have $m$ “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example!

```
18
42
56
89
15
77
43
57
11
98
```

```
<table>
<thead>
<tr>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>42</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>57</td>
<td>11</td>
</tr>
<tr>
<td>89</td>
<td></td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
The Parallel Machines Scheduling Problem ($P||C_{max}$)

We want to perform $n$ production jobs, we have $m$ “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example!
The Parallel Machines Scheduling Problem \((P||C_{max})\)

We want to perform \(n\) production jobs, we have \(m\) “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example!
The Parallel Machines Scheduling Problem (\(P|\|C_{max}\))

We want to perform \(n\) production jobs, we have \(m\) “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example!

\[
\begin{array}{cccc}
18 & 42 & 56 & 89 \\
15 & 77 & 43 & 57 \\
89 & 56 & 11 & 98 \\
77 & & & \\
\end{array}
\]
The Parallel Machines Scheduling Problem ($P||C_{max}$)

We want to perform $n$ production jobs, we have $m$ “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

An example! Basic approach: enumerate all solutions
The Parallel Machines Scheduling Problem \((P||C_{\text{max}})\)

We want to perform \(n\) production jobs, we have \(m\) “parallel” identical machines...

- **Given data:** for each job, its duration
- **Constraint:** perform all jobs
- **Decision:** assign jobs to machines
- **Objective:** end the last job as early as possible

An example! Basic approach: enumerate all solutions
The Parallel Machines Scheduling Problem \((P||C_{max})\)

We want to perform \(n\) production jobs, we have \(m\) “parallel” identical machines...

- Given data: for each job, its duration
- Constraint: perform all jobs
- Decision: assign jobs to machines
- Objective: end the last job as early as possible

Until which size can a computer enumerate all solutions?
The Parallel Machines Scheduling Problem ($P||C_{max}$)

Exercise: how many possible solutions? ($m$ machines, $n$ jobs)

- Let $C_p$ be the number of solutions for $p$ jobs
- Each job can be assigned to any of the $m$ machines
- Therefore $C_1 = m$
- Now for all of these $m$ solutions, we can assign a second job to any of the $m$ machines, so $C_2 = m \cdot C_1 = m^2$
- In a similar way, $C_n = m \cdot C_{n-1} = m \cdot m \cdot C_{n-2}$
- Total: $m \cdot m \cdot m \cdot \ldots \cdot m = m^n$

Suppose our computer can process 500,000 solutions per second...

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10</td>
<td>2 s</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>17 h</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>232 years</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td>$2.39 \cdot 10^9$ years</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
<td>$6.34 \cdot 10^{16}$ years</td>
</tr>
</tbody>
</table>
We are a company with \( m \) warehouses and \( n \) customers. . .

- Given data: cost induced by servicing customers through warehouses (e.g. distance)
- Constraint 1: No more than \( p \) warehouses may be open
- Constraint 2: Each customer must be serviced by a warehouse
- Decision 1: which warehouses should we open?
- Decision 2: which open warehouse should service which customer?
- Objective: minimise total service cost
Warehouse Location (a.k.a. $p$-median Problem)

An example! ($m = 10, p = 5$)
Warehouse Location (a.k.a. $p$-median Problem)

An example! ($m = 10$, $p = 5$)
Warehouse Location (a.k.a. $p$-median Problem)

An example! ($m = 10, p = 5$)  Basic approach: enumerate
Warehouse Location (a.k.a. $p$-median Problem)

An example! ($m = 10, p = 5$) Basic approach: enumerate
Warehouse Location (a.k.a. $p$-median Problem)

Until which size can a computer enumerate all solutions?
Exercise: how many possible solutions?

- A solution = which depots should be open
- (then we can assign to each customer its closest depot)
- Number of solutions using exactly $p$ warehouses: $\binom{m}{p}$
- There are solutions using $p$ warehouses,
- Some other using $p - 1$ warehouses, and so on down to 1.
- Total quantity of solutions: $\binom{m}{p} + \binom{m}{p-1} + \ldots + \binom{m}{1}$
- Which is equal to: $\sum_{k=1}^{p} \binom{m}{k}$
Suppose our recent computer can process 4,000 solutions per second...

<table>
<thead>
<tr>
<th>$m$</th>
<th>$p$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>0.15 s</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>2.5 m</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>3 days</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>5 years</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>904 years</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>$5 \cdot 10^6$ years</td>
</tr>
</tbody>
</table>

Some insight

- **Question:** Evaluation is quite time-consuming. Why?
- **Answer:** Given a solution, finding the best open depot for each customer takes time.
Solving NP-hard problems

**Bottom line**

NP-hard problems are hard to solve
Enumeration is not the most clever approach...

There are alternatives:

- Exact methods
  - Example 1: Constraint Programming
  - Example 2: Linear Programming
  - May allow to solve bigger problems than with enumeration
  - May also be extremely long: a TSP with 15,112 nodes in 22.6 years on a computer from 2001
  - Size limitations occur sooner or later

- Heuristics

- Metaheuristics (general-purpose heuristic-based methods)
Heuristic approaches

A metaheuristic is a special kind of heuristic method...

(personal) Definition

Heuristic: a method based on common sense, empirical knowledge, and intuition.

- This is neither a common sense lecture...
- ...nor an intuition lecture.
- We will work on method
- You will gain specific knowledge in the next course
Heuristic approaches

A metaheuristic is a special kind of heuristic method...

**Definition**

Heuristic: a method based on common sense, empirical knowledge, and intuition.

- This is neither a common sense lecture...
- ...nor an intuition lecture.
- We will work on method
- You will gain specific knowledge in the next course
A metaheuristic is a special kind of heuristic method...

**Definition**

Heuristic: a method based on common sense, empirical knowledge, and intuition.

- This is neither a common sense lecture...
- ...nor an intuition lecture.
- We will work on method
- You will gain specific knowledge in the next course
Heuristic approaches

A metaheuristic is a special kind of heuristic method...

(perso[nal]) Definition

**Heuristic:** a method based on **common sense**, empirical knowledge, and **intuition**.

• This is neither a common sense lecture...
• ...nor an intuition lecture.
• We will work on method
• You will gain specific knowledge in the next course
A metaheuristic is a special kind of heuristic method...

**Definition**
Heuristic: a method based on common sense, empirical knowledge, and intuition.

- This is neither a common sense lecture...
- ...nor an intuition lecture.
- We will work on method
- You will gain specific knowledge in the next course
Construction Vs. Improvement

Two critical questions

1. How do we construct a solution?
2. How do we improve a constructed solution?

Different questions call for different answers...

Two possible answers

1. With a construction heuristic
2. With an improvement heuristic
Constructive heuristics

Idea: from a problem input, construct a solution to this problem.

**Example: greedy algorithms**
- Iteratively satisfy problem constraints
- Use a *greedy criterion* to achieve this
- Stop when all constraints are satisfied
- Remark: The greedy criterion should take the objective into account

**Exercise: design a greedy heuristic for the TSP**
1. Which constraints do we iteratively satisfy?
2. What is the greedy criterion?
**Greedy heuristic for the TSP**

**Distance matrix**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>28</td>
<td>82</td>
<td>-</td>
<td>99</td>
<td>52</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>-</td>
<td>42</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>21</td>
<td>46</td>
<td>54</td>
<td>58</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>27</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>28</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335
But there exists a better solution, with cost = 308 (9% better)
So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>50</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>
```

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>38</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>28</td>
<td>39</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308

(9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP
Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>82</td>
<td>-</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335
But there exists a better solution, with cost = 308
(9% better)
So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>82</td>
<td>-</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>88</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>15</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

Total cost = 335

But there exists a better solution, with cost = 308 (9% better)

So it would be nice to improve this tour...
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

- Total cost = 335
Greedy heuristic for the TSP
Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

- Total cost = 335
- But there exists a better solution, with cost = 308 (9% better)
Greedy heuristic for the TSP

Nearest neighbour construction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76</td>
<td>7</td>
<td>73</td>
<td>69</td>
<td>79</td>
<td>54</td>
<td>58</td>
<td>38</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>82</td>
<td>108</td>
<td>16</td>
<td>52</td>
<td>99</td>
<td>51</td>
<td>42</td>
<td>72</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>-</td>
<td>69</td>
<td>74</td>
<td>81</td>
<td>50</td>
<td>60</td>
<td>42</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>92</td>
<td>68</td>
<td>19</td>
<td>57</td>
<td>73</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>74</td>
<td>92</td>
<td>-</td>
<td>36</td>
<td>84</td>
<td>35</td>
<td>31</td>
<td>60</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>81</td>
<td>68</td>
<td>36</td>
<td>-</td>
<td>68</td>
<td>21</td>
<td>46</td>
<td>60</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>50</td>
<td>19</td>
<td>84</td>
<td>68</td>
<td>-</td>
<td>52</td>
<td>60</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>60</td>
<td>57</td>
<td>35</td>
<td>21</td>
<td>52</td>
<td>-</td>
<td>28</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>42</td>
<td>73</td>
<td>31</td>
<td>46</td>
<td>60</td>
<td>28</td>
<td>-</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>22</td>
<td>52</td>
<td>60</td>
<td>60</td>
<td>35</td>
<td>39</td>
<td>30</td>
<td>-</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>40</td>
<td>52</td>
<td>45</td>
<td>41</td>
<td>40</td>
<td>20</td>
<td>21</td>
<td>19</td>
<td>-</td>
</tr>
</tbody>
</table>

- Total cost = 335
- But there exists a better solution, with cost = 308 (9% better)
- So it would be nice to improve this tour...
Improvement heuristics

**Definition**

**Neighbourhood**: the set of solutions that can be obtained by applying a given operator to a given solution.

Solutions in the neighbourhood of $x$ are its neighbours.

- Operator example for the TSP: move one node to another position in the tour.
- Exercise: find other neighbourhoods for the TSP.

**Definition**

**Local Search**: a heuristic aiming at improving a solution by performing neighbourhood exploration.
Improvement heuristics for the TSP
An example: the “node move” neighbourhood

Suppose that visiting order starts with node 2... We can:
Suppose that visiting order starts with node 2... We can:

1. Move 2 at the end of the tour

For this solution, cost = 326. We are still far from 308...
Suppose that visiting order starts with node 2... We can:

1. Move 2 at the end of the tour
Suppose that visiting order starts with node 2... We can:

1. Move 2 at the end of the tour
2. Move 6 after 3
Improvement heuristics for the TSP
An example: the “node move” neighbourhood

Suppose that visiting order starts with node 2. . .
We can:

1. Move 2 at the end of the tour
2. Move 6 after 3

For this solution, cost = 326
We are still far from 308. . .
Improvement heuristics for the TSP
An example: the “node move” neighbourhood

Suppose that visiting order starts with node 2...
We can:

1. Move 2 at the end of the tour
2. Move 6 after 3
3. Move 10 after 8
Suppose that visiting order starts with node 2...

We can:

1. Move 2 at the end of the tour
2. Move 6 after 3
3. Move 10 after 8
Improvement heuristics for the TSP
An example: the “node move” neighbourhood

Suppose that visiting order starts with node 2... We can:

1. Move 2 at the end of the tour
2. Move 6 after 3
3. Move 10 after 8

Then there is no more improvement!
Suppose that visiting order starts with node 2.

We can:

1. Move 2 at the end of the tour
2. Move 6 after 3
3. Move 10 after 8

Then there is no more improvement!

- For this solution, cost = 326
Suppose that visiting order starts with node 2…
We can:

1. Move 2 at the end of the tour
2. Move 6 after 3
3. Move 10 after 8

Then there is no more improvement!

- For this solution, cost = 326
- We are still far from 308…
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space.
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space.
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space.

Note that the global optimum is also a local optimum.
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space. . .
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space...

Fitness

Solutions

No more improvement in the neighbourhood!
The local optimum issue
Assume neighbour solutions are adjacent points in this Solution Space.

No more improvement in the neighbourhood!
Note that the global optimum is also a local optimum
Simple heuristics: limitations

A heuristic typically exploits problem structure

- Good news: using information from the problem really helps
- Bad news: heuristic designed for problem $X$ cannot be applied to problem $Y$
- Bad news: sometimes a heuristic is “just not enough”
  - Local optima
  - How do we explore a neighbourhood?

Metaheuristics aim at dealing with these bad news!
Outline

1. Algorithms
   - Overview
   - How to write an algorithm

2. Complexity theory

3. 3 NP-hard problems - Heuristics needed
   - The Traveling Salesman Problem
   - The Parallel Machines Scheduling Problem
   - The Warehouse Location Problem

4. Introducing Heuristics
   - Introduction
   - Construction and Improvement
   - Limitations of “simple” heuristics

5. Introducing Metaheuristics
   - Etymology and Historical Aspects
   - Evaluating Metaheuristics
Etymology

There is no commonly agreed definition!

(personal) Definition

**Metaheuristic**: a general-purpose heuristic method manipulating other, often problem-specific, heuristic methods.

Other definitions:

- A top-level general strategy which guides other heuristics to search for feasible solutions in domains where the task is hard. (various sources on the web)
- A set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. (Metaheuristics Network)

Most metaheuristics include **stochastic** components (a.k.a. **non-deterministic**; e.g. randomness is involved in the process)
A few milestones

Term introduction

- This paper is about **Tabu Search**, a now well-known metaheuristic.
- (but there were metaheuristics before that...) 

A few dates

- 1965: first **Evolution Strategy**
- 1975: first **Genetic Algorithms**
- 1983: **Simulated Annealing**
- 1986: **Tabu Search**
- 1991: **Ant Colony Optimisation**
- 1997: **Variable Neighbourhood Search**
Evaluating Metaheuristics

- There is no such thing as a “Best Metaheuristic”
- Some theoretical results back this claim
- But there are still relevant questions!

### Relevant questions

1. Is the metaheuristic well-designed?
   - → Qualitative analysis

2. Once applied to a given problem, does it behave well?
   - → Quantitative analysis
Qualitative analysis

Again, no common agreement; personal suggestions:

**Suggestion 1: Clarity, Modularity, Simplicity**
A metaheuristic should be easy to apply to any given hard problem!

**Suggestion 2: Intensification**
- The global optimum is also a local optimum
- A good metaheuristic should be good at finding local optima!

**Suggestion 3: Diversification**
- Interesting solutions might reside in different “regions”
- A good metaheuristic should provide a broad exploration of the solution space!

Intensification and Diversification can be seen as opposite goals.
Quantitative analysis

Goal: once the metaheuristic is implemented, evaluate it.

Interesting criteria include:

- Ability to find good/optimal solutions
  - Comparison with optimal solutions (if available)
  - Comparison with Lower/Upper Bounds (LB/UB, if available)
  - Comparison with other metaheuristics

- Reliability and stability over several runs
  - Homemade indicators (e.g. Average gap to the optimum)
  - Traditional statistical tools (standard deviation etc)

- Reasonable computational time
  - “Reasonable” → highly subjective
  - Context-dependent
  - Usually: between a few seconds and 1 hour

Keep this in mind for in a few weeks!