Variable Neighborhood Search

Nuno Antunes Ribeiro

Assistant Professor

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

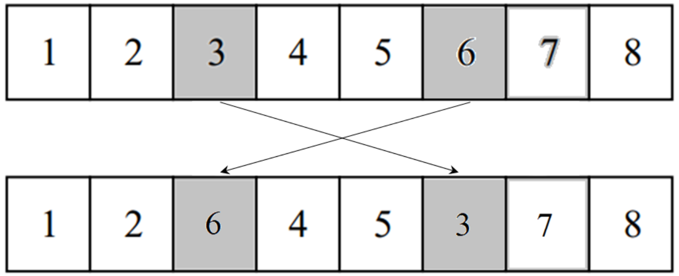
Engineering Systems and Design

Variable Neighbourhood Search

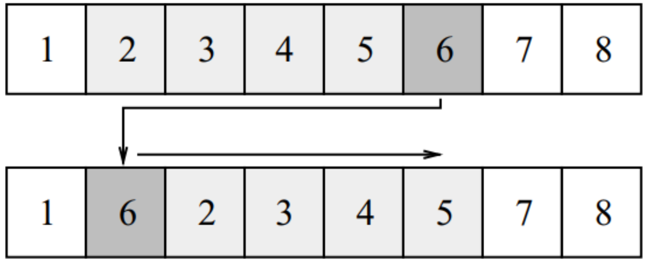
- The basic idea of Variable Neighbourhood Search VNS is to successively **explore a set of predefined neighbourhoods** to provide a better solution.
- It explores either at random or systematically a set of neighbourhoods to get different local optima and to escape from local optima.
- VNS exploits the fact that using various neighbourhoods in local search may generate different local optima and that the global optima is a local optima for a given neighbourhood.
- Different neighbourhoods generate different landscapes

Different Neighbourhoods

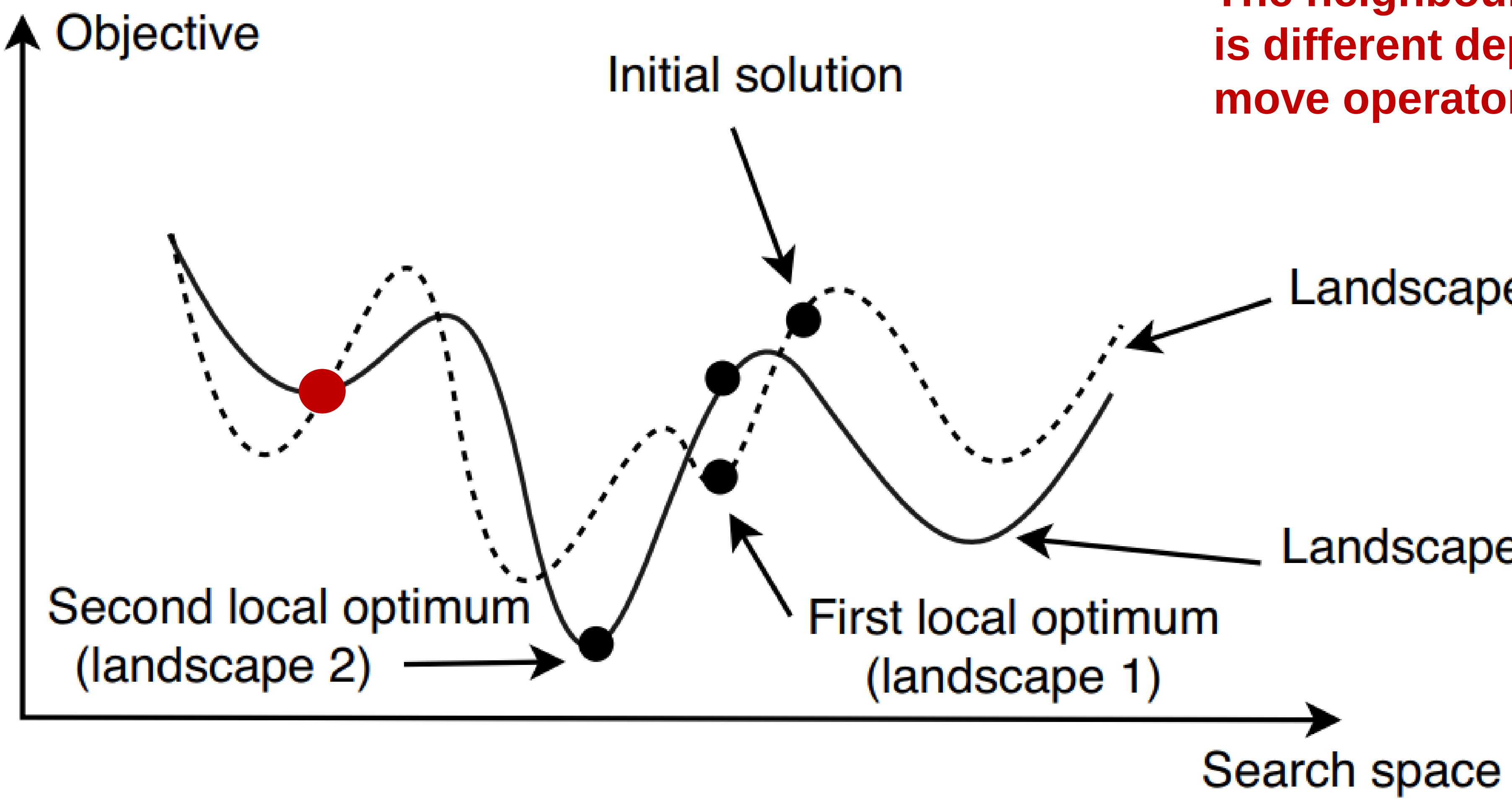
- Current Solution: [1,2,3,4,5,6,7,8]
  - We randomly select 6 to be moved using an operator
    - Swap Operator Neighbourhood
      - [6,2,3,4,5,1,7,8]
      - [1,6,3,4,5,2,7,8]
      - [1,2,6,4,5,3,7,8]
      - [1,2,3,6,5,4,7,8]
      - [1,2,3,4,6,5,7,8]
      - [1,2,3,4,5,7,6,8]
      - [1,2,3,4,5,8,7,6]



    - Insertion Operator Neighbourhood
      - [6,1,2,3,4,5,7,8]
      - [1,6,2,3,4,5,7,8]
      - [1,2,6,3,4,5,7,8]
      - [1,2,3,6,4,5,7,8]
      - [1,2,3,4,6,5,7,8]
      - [1,2,3,4,5,7,6,8]
      - [1,2,3,4,5,7,8,6]



Variable Neighbourhood Search



The neighbourhood of a solution is different depending on the move operator used

Search Operator Selection Procedure

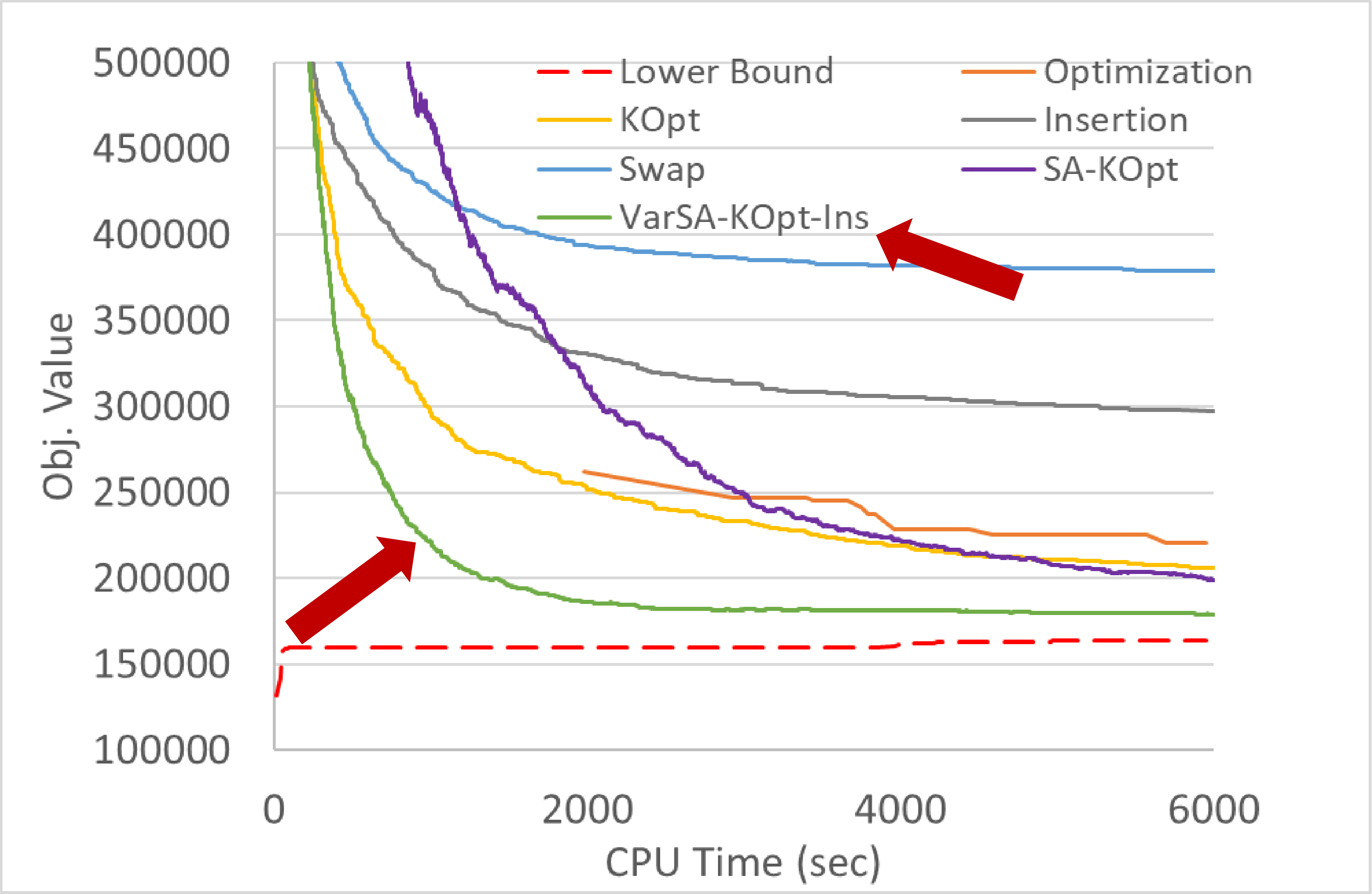
- The design of the VND algorithm is mainly related to the **selection of neighbourhoods and corresponding search operators.**
- Different strategies can be used:
  - **Exhaustive Selection** – At each iteration, **all the neighbourhoods are investigated**. The best solution found is selected for evaluation (through hill climbing, simulated annealing, others)
  - **Rank-based Selection** - **The different neighbourhoods are ranked by user** preference or complexity - e.g. size of the neighbourhood. At each iteration, the 1st ranked neighbourhood is evaluated. If a better solution is found, the algorithm moves to the next iteration; otherwise it explore the 2nd best neighbourhood. This procedure is repeated until all neighbourhoods are explored. The best solution found is selected for evaluation (through hill climbing, simulated annealing, others)
  - **Probabilistic Selection** – **A probability is given to each neighbourhood**. At each iteration a neighbourhood is randomly selected. The best solution found is selected for evaluation (through hill climbing, simulated annealing, others)

Variable Neighbourhood Search

1. **Initialize**: Generate random initial solution, p_{best},
2. **While** (termination criteria is not met)
   3. Apply search operator selection procedure
   4. Generate a new solution (or a set of new solutions) p_{new} by applying a the search operator selected to p_{best}
   5. If p_{new} is better than p_{best}, than p_{best}= p_{new}
   6. Go back to 2, until termination criteria is met

```
while swap_it<no_swap:
    if random.random()<0.5:
        k_opt(Solution_i)
    else:
        insert_random(Solution_i)
    swap_it=swap_it+1
```

TSP Example


Lower Bound
Optimization
KOpt
Insertion
Swap
SA-KOpt
VarSA-KOpt-Ins
500000
450000
400000
350000
300000
250000
200000
150000
100000
Obj. Value
0
2000
4000
6000
CPU Time (sec)

Multistage Local Search

Nuno Antunes Ribeiro

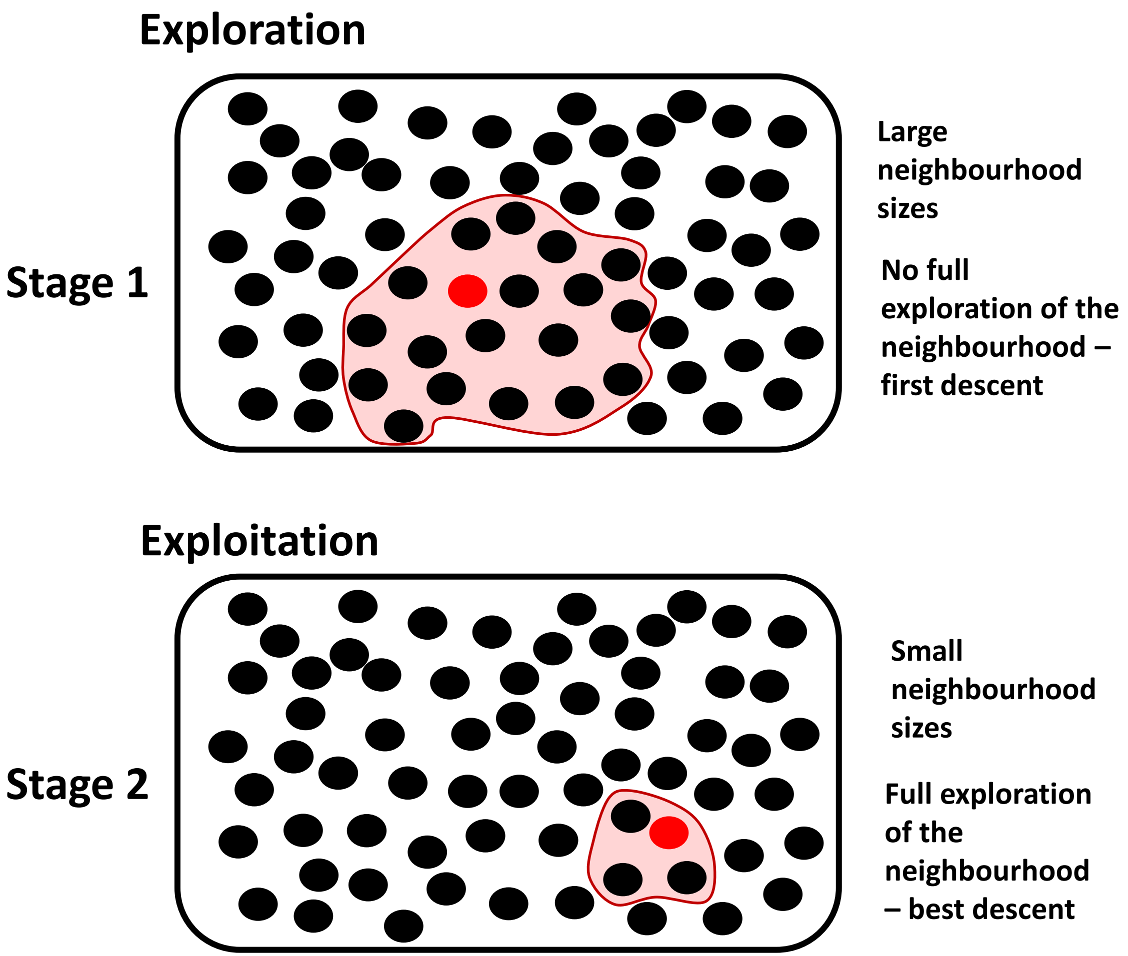
Assistant Professor

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Engineering Systems and Design

Multistage Local Search

- The basic idea of Multistage Local Search is having successive stages where different metaheuristic strategies are applied.
- For instance:
  - Having different search operators (e.g. swap operator followed by 2-opt operator)
  - Having different **neighbourhood explorations** (e.g. first descent, followed by best descent)
  - Having different metaheuristic approaches (e.g. genetic algorithm followed by simulated annealing)



Greedy Constructive Heuristics

Nuno Antunes Ribeiro

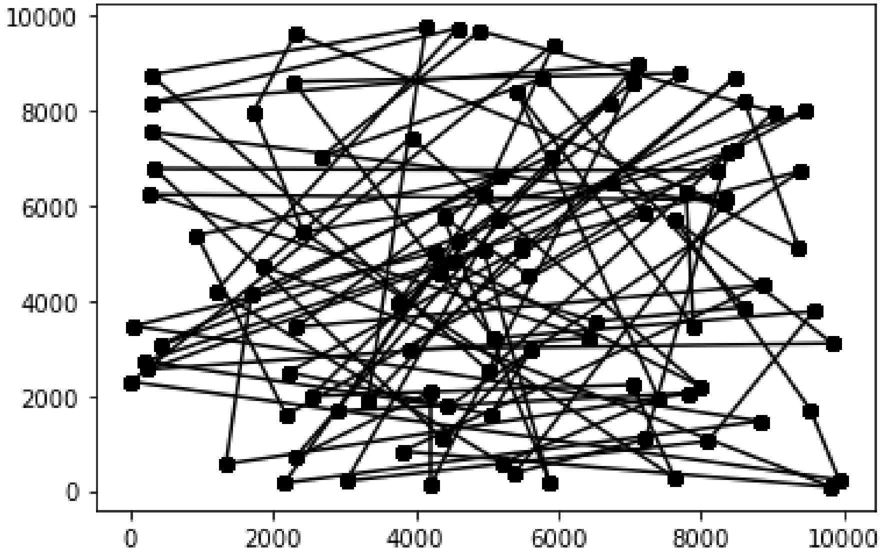
Assistant Professor

Engineering Systems and Design

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Initial Solution

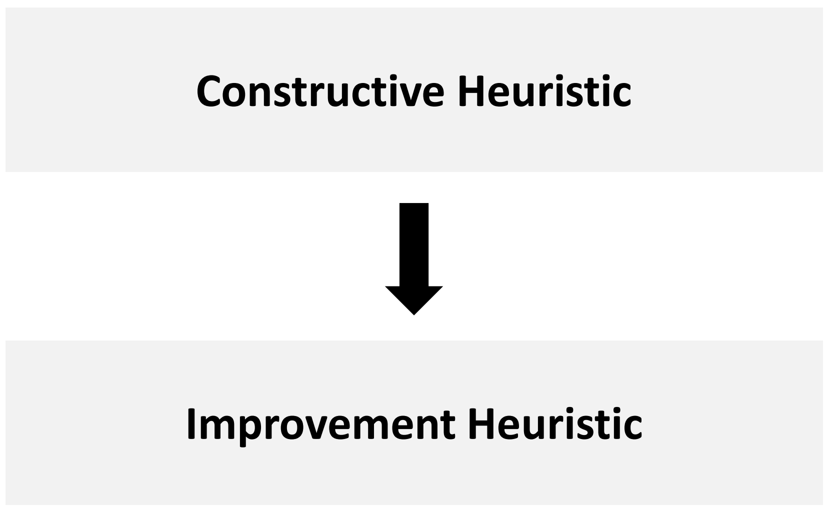
- Generating a random initial solution is quick, however the metaheuristic may take a large number of iterations to converge.
- To speed up the search, a **greedy constructive heuristic** may be applied.
- In greedy heuristics, we **start from scratch (empty solution) and construct a solution by assigning values to one decision variable at a time**, until a complete solution is generated.
- Many optimization problems have good greedy algorithms available, easy to design and implement.
- Note however that it does not necessarily mean that starting with a better initial solution always lead to better solutions.

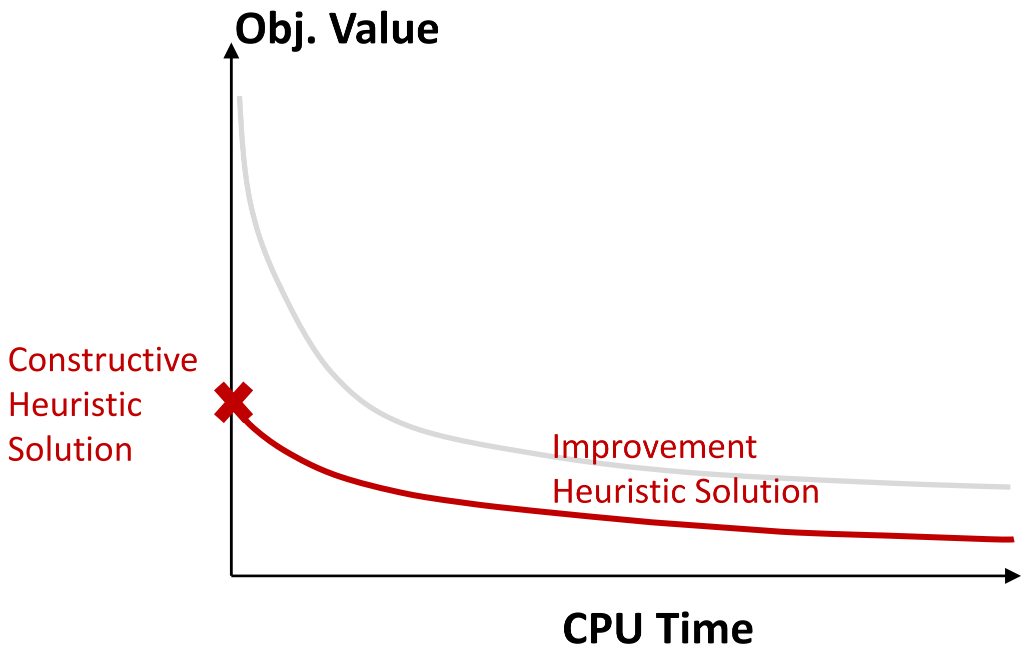


Random Solution

Constructive + Improvement Heuristics

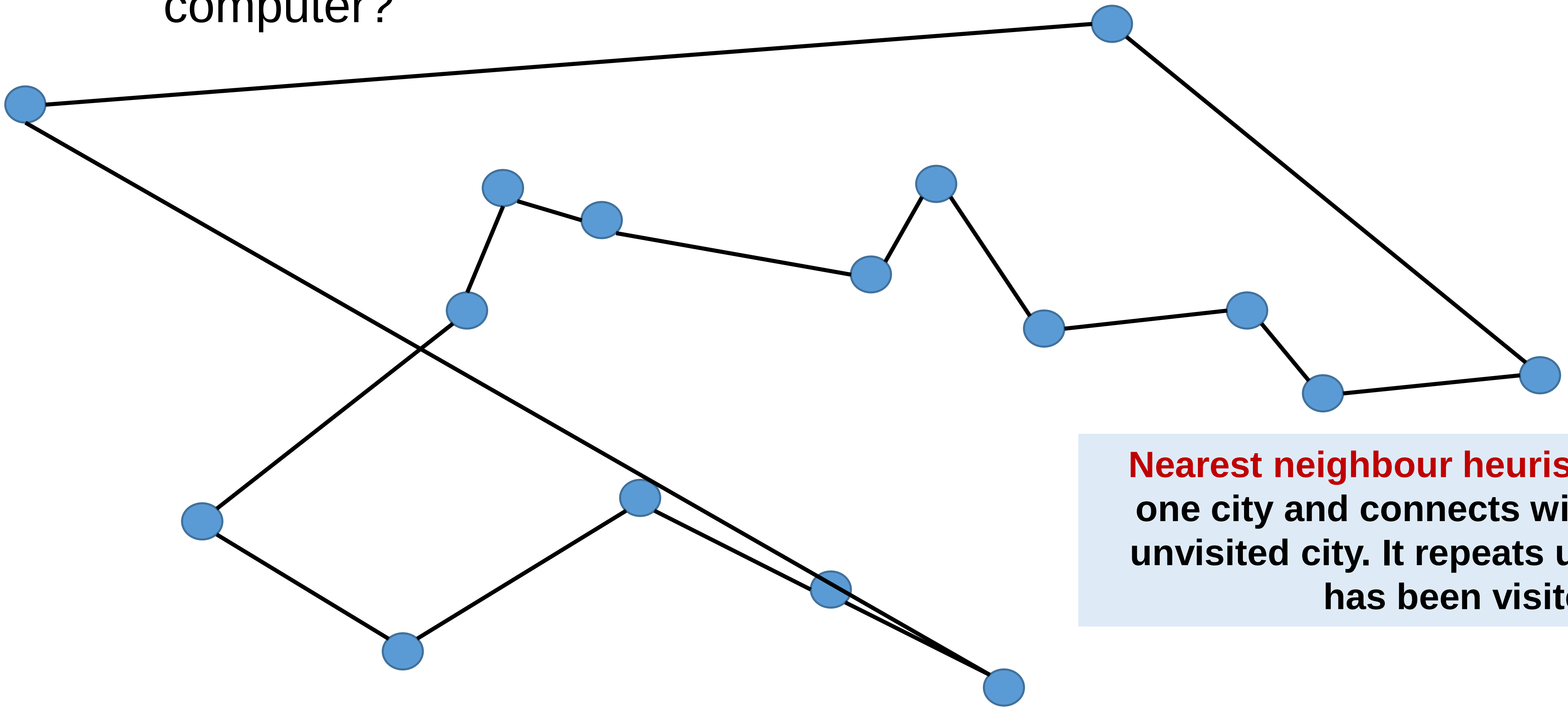
- **Constructive Heuristic** – fast method used to generate an initial feasible solution. It starts with an empty solution and repeatedly extends this solution until a complete solution is obtained - **problem specific**
- **Improvement Heuristic** – method used to improve an existing solution by performing local adjustments – **concept of metaheuristics**





Greedy Heuristic for the TSP

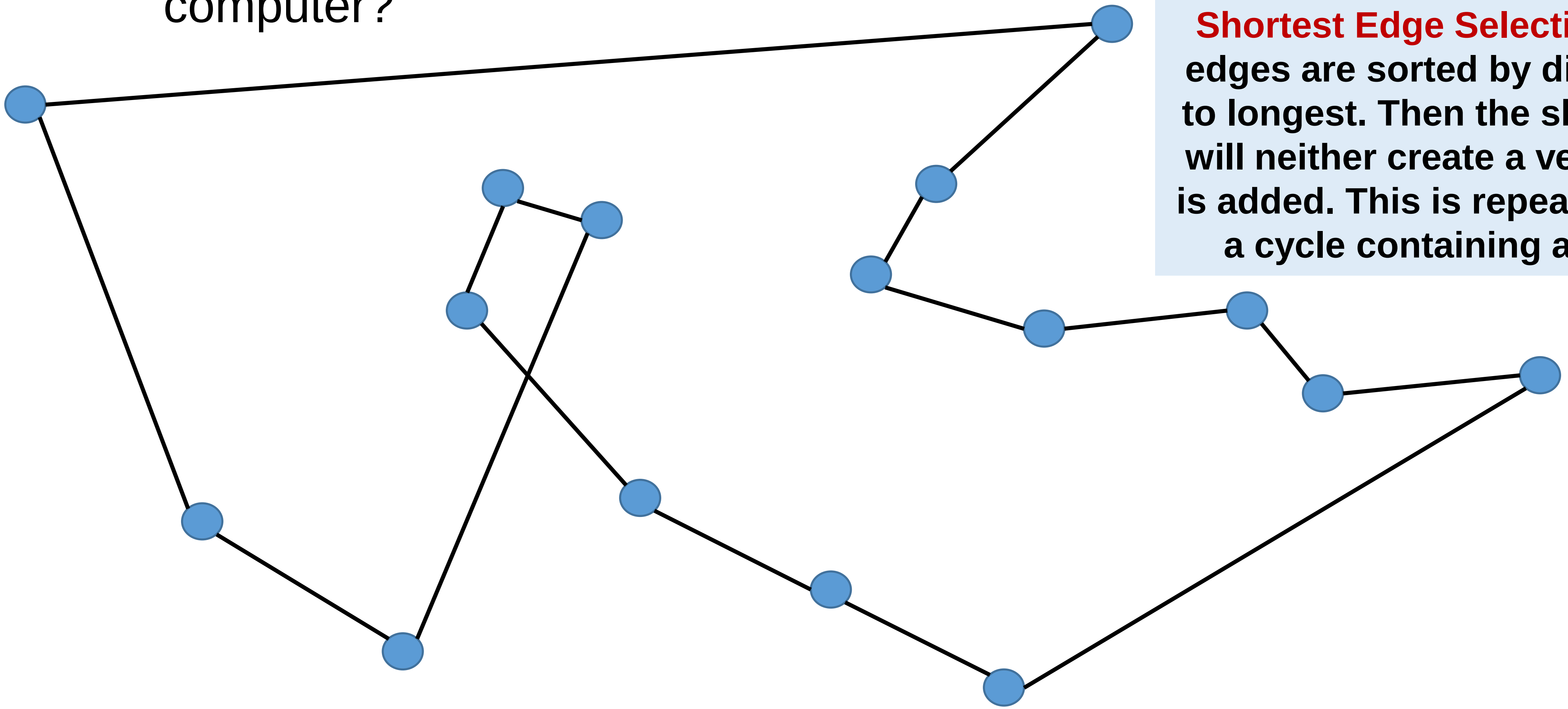
- How would you optimize the TSP if you did not have available a computer?



Nearest neighbour heuristic: It starts at one city and connects with the closest unvisited city. It repeats until every city has been visited.

Greedy Heuristic for the TSP

- How would you optimize the TSP if you did not have available a computer?



Shortest Edge Selection: All possible edges are sorted by distance, shortest to longest. Then the shortest edge that will neither create a vertex, nor a cycle is added. This is repeated until we have a cycle containing all of the cities.

Greedy - Shortest Edge Selection

- Distance Matrix

C-D

A-C-D

A-C-D-B

A-C-D-B-A ✗ Cycle

E-A-C-D-B

E-A-C-D-B-E

	A	B	C	D	E
A	999	999	20	999	999
B	999	999	999	999	90
C	999	999	999	10	999
D	999	30	999	999	999
E	70	999	999	999	999

Greedy - Shortest Edge Selection

```
while np.min(distancelct)!=99999:
    nextmin=np.where(distancelct==np.min(distancelct))
    p2_value=0
    if len(nextmin[0])>1:
        p1=nextmin[0][0]
        p2=nextmin[1][0]
    else:
        p1=nextmin[0]
        p2=nextmin[1]
    pcycle=np.copy(p2)
    while np.where(pcycle==linkindex_p1)[0].size==1:
        pcycle_index=np.where(pcycle==linkindex_p1)[0]
        pcycle=linkindex_p2[np.where(pcycle==linkindex_p1)[0]]
    if pcycle== p1:
        distancelct[p1,p2]=99999
    else:
        distancelct[p1,:]=99999
        distancelct[:,p2]=99999
        distancelct[p2,p1]=99999
        linkindex_p1=np.append(linkindex_p1, p1)
        linkindex_p2=np.append(linkindex_p2, p2)
        linkindex_p1=linkindex_p1.astype(int)
        linkindex_p2=linkindex_p2.astype(int)
```

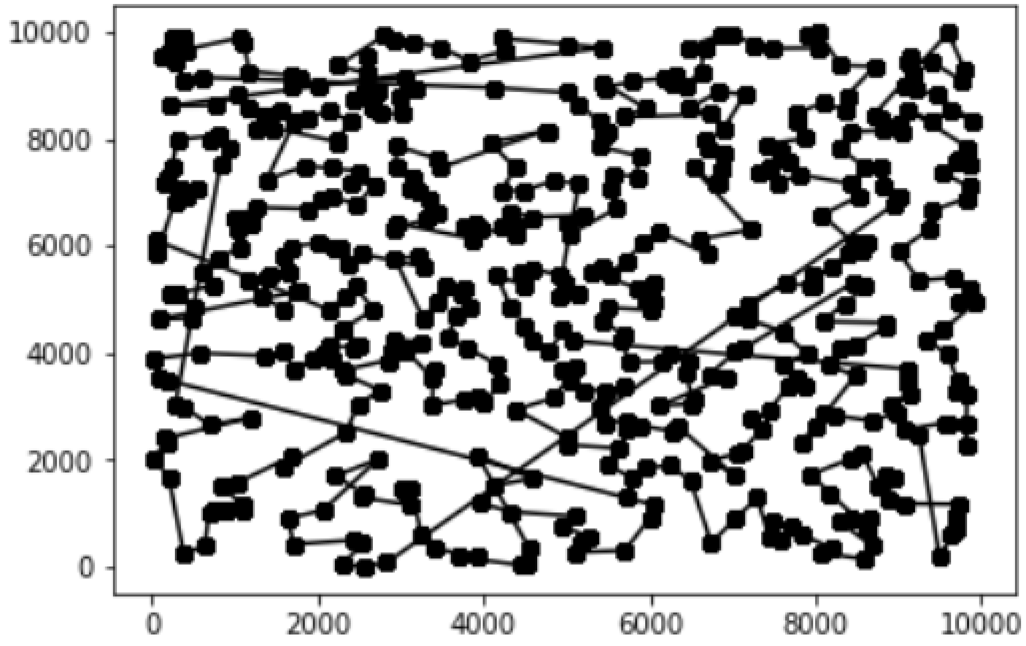
Stop when all distances are 999

Select minimum distance

Check if cycle

Update distance table and solution variables

Greedy - Shortest Edge Selection



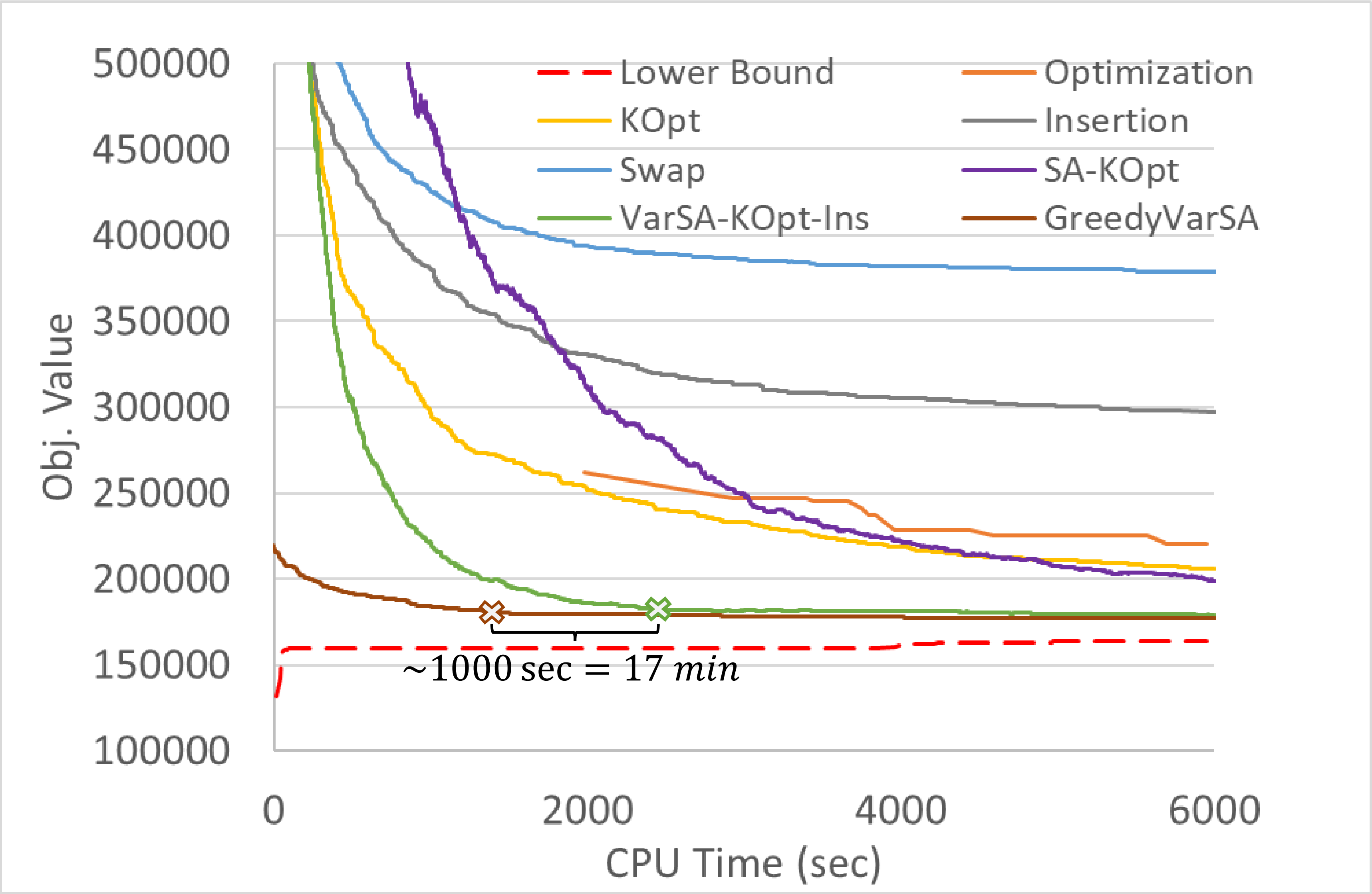
Obj. Value = 219,336.56 (Less than 1 second)

Best Solution Found SA = 198,575.60 (After 6000 seconds)

Best Solution Found OPT = 220,704.05 (After 6000 seconds)

Best Lower Bound OPT = 163,571.47

Greedy integration with SA

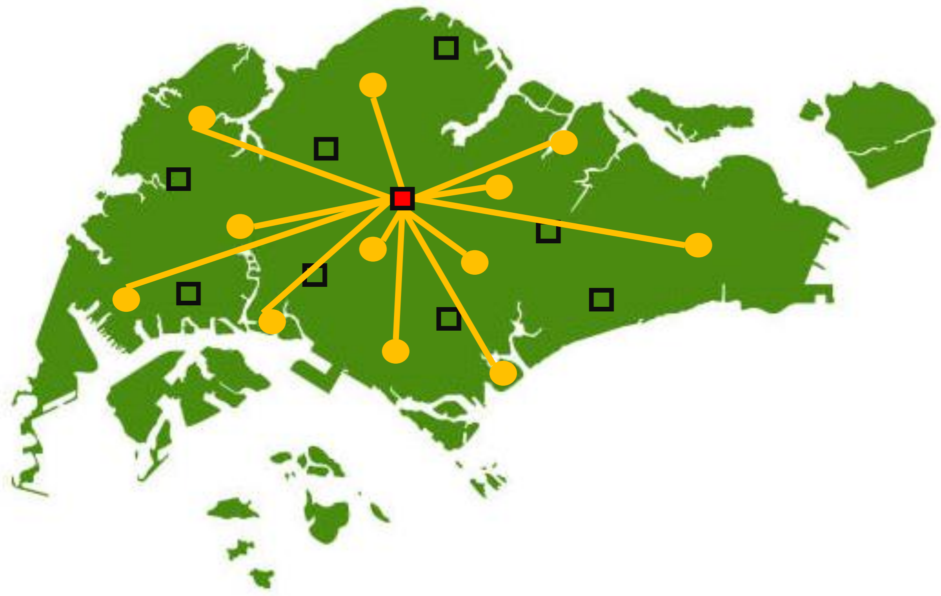


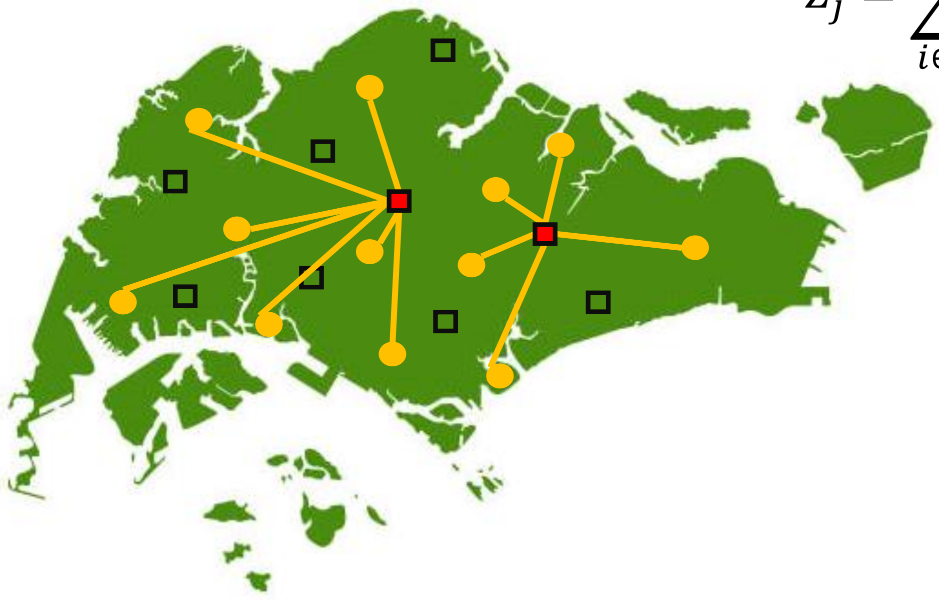
Other Greedy Heuristics

- P-Median - Greedy-add algorith,
  - Firstly, a facility is located in such a way as to minimize the total cost for all customers. Facilities are then added one by one until p is reached.

Cost

$$Z_j = \sum_{i \in I} d_j c_{ij} x_{ij} \ \forall j \in I$$



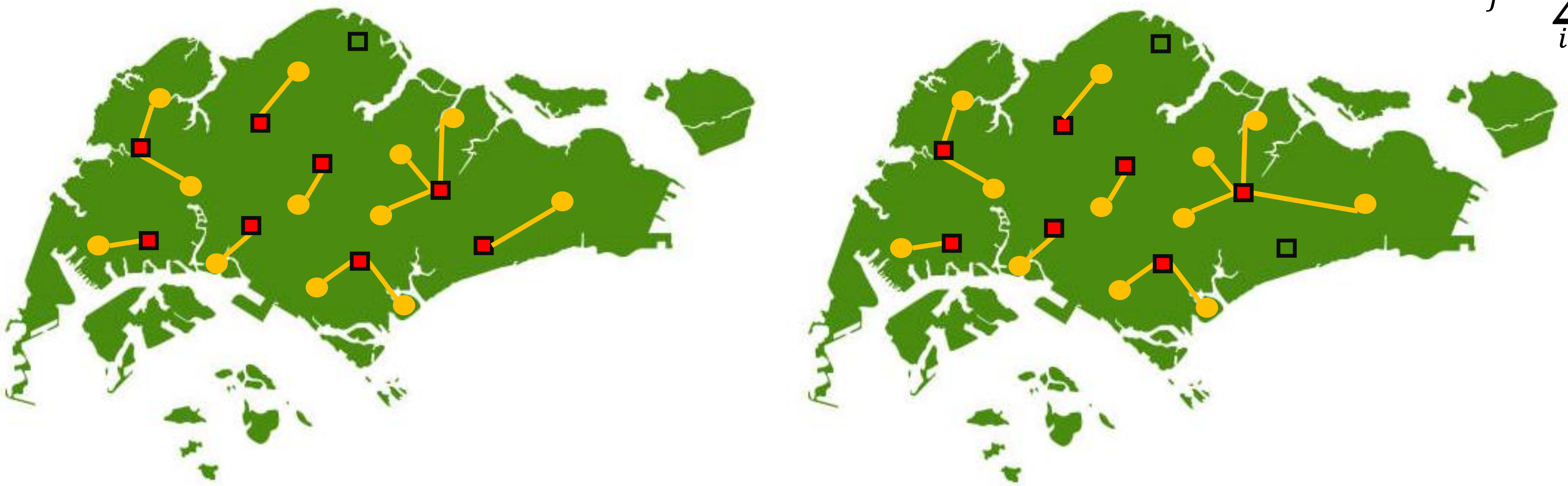


Other Greedy Heuristics

- P-Median - Greedy-drop algorithm
  - Starts with facilities located at all potential facility sites and then eliminate (drop) the facility that has the least impact on the objective function.

Cost

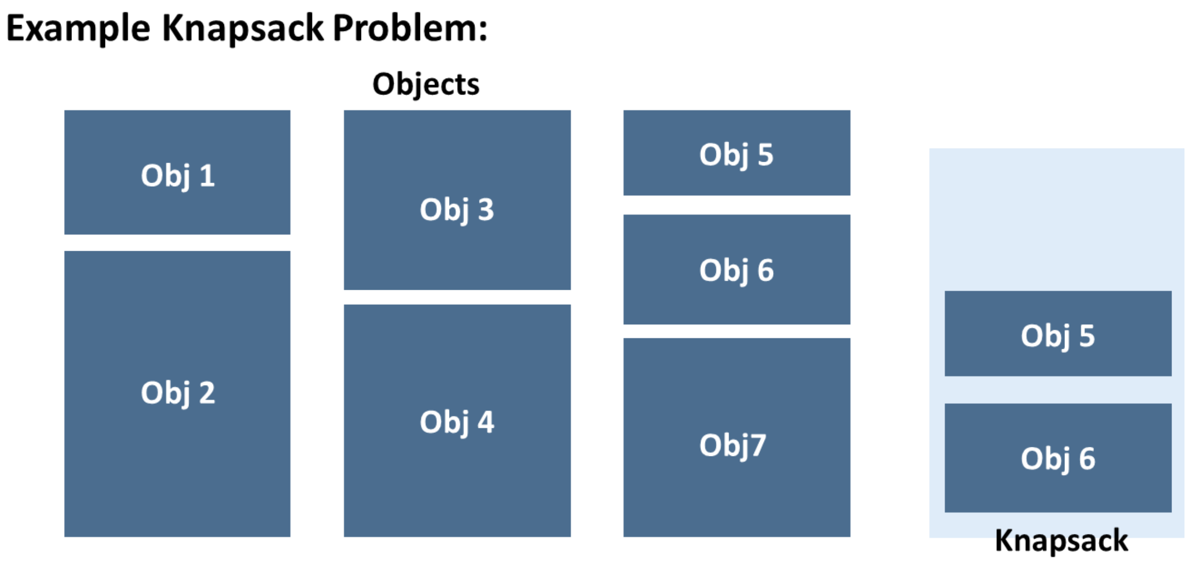
$$Z_j = \sum_{i \in I} d_j c_{ij} x_{ij} \ \forall j \in I$$



Other Greedy Heuristics

- Knapsack Problem - profit/weight greedy algorithm
  - Select always the object with the highest profit/weight value

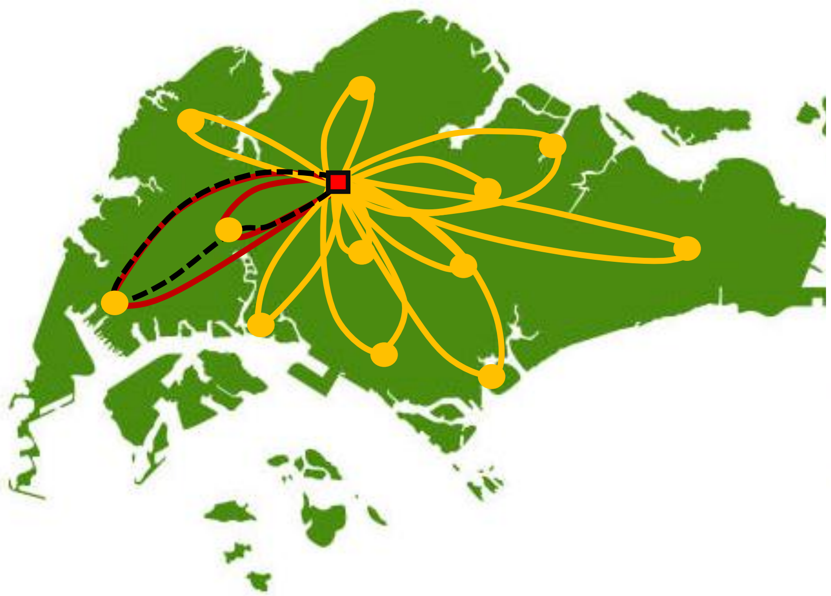
	w_i	f_i	f_i/w_i
Obj 1	2	5	2.5
Obj 2	3.75	7	1.87
Obj 3	2.5	3	1.2
Obj 4	3	5	1.67
Obj 5	1	4	4
Obj 6	1.5	8	5.33
Obj 7	2.75	7	2.54
cap	4		

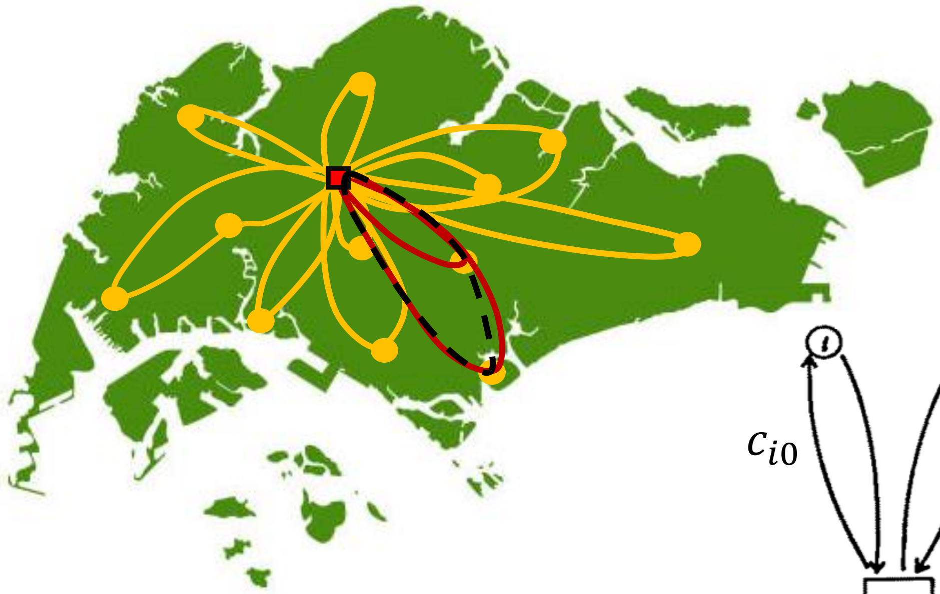


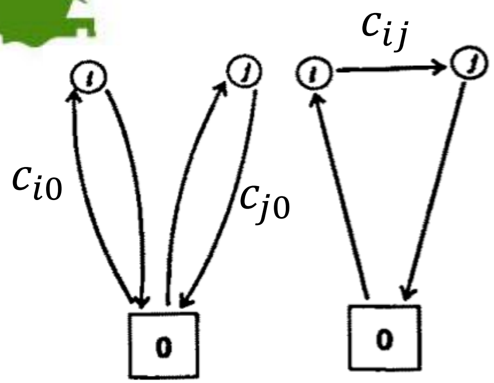
Profit=12
Weight=2.5

Other Greedy Heuristics

- Vehicle Routing - Clarke-Wright greedy algorithm
  - The procedure starts with each costumer being served by a single tour.
  - Cost savings $S_{ij} = c_{i0} + c_{j0} - c_{ij}$ can be obtained assuming c_{ij} is the cost of travelling from customer i to j (where $j = 0$ is the depot)
  - The savings are sorted in decreasing order. The procedure merges costumers i and j corresponding to the highest savings S_{ij} , without violating the capacity restrictions

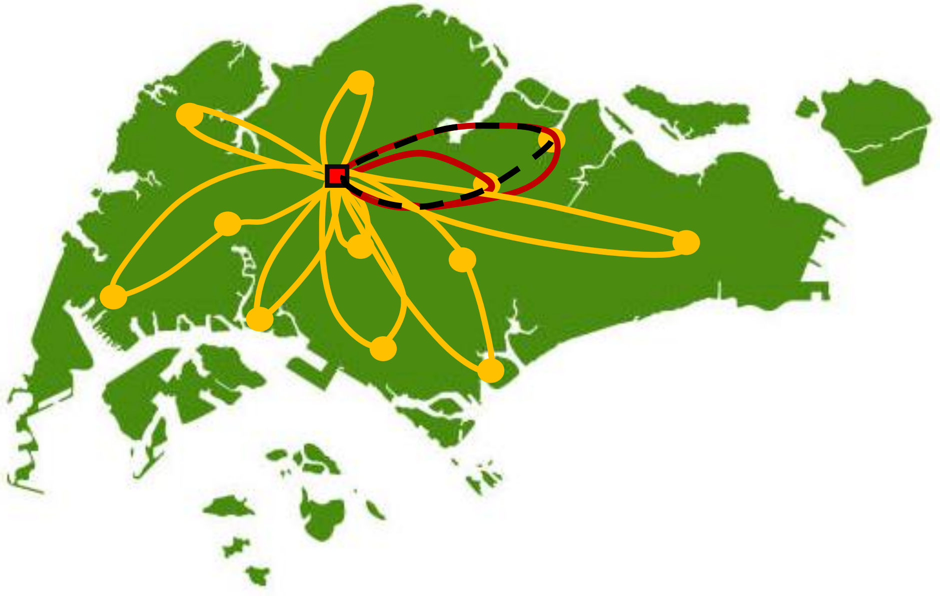


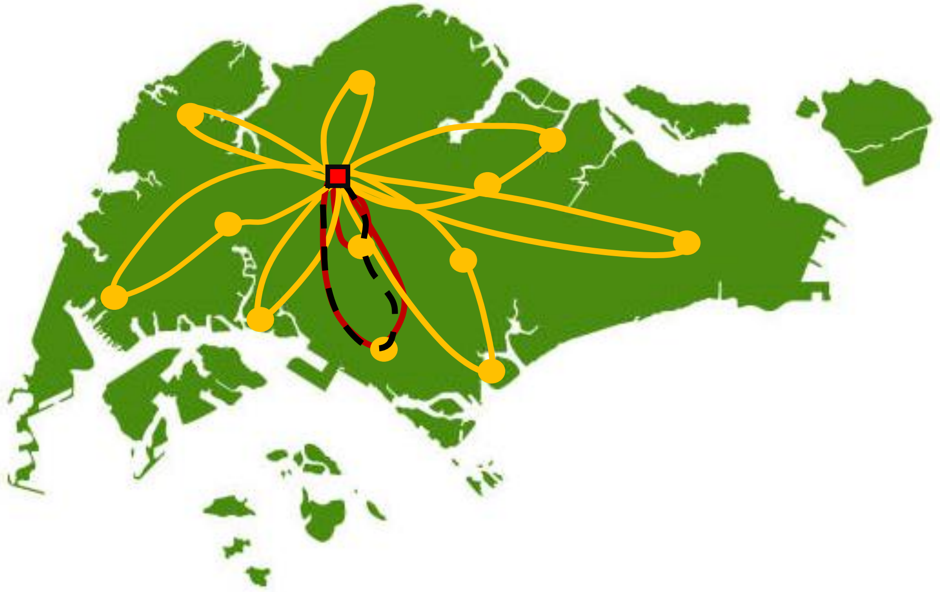




Other Greedy Heuristics

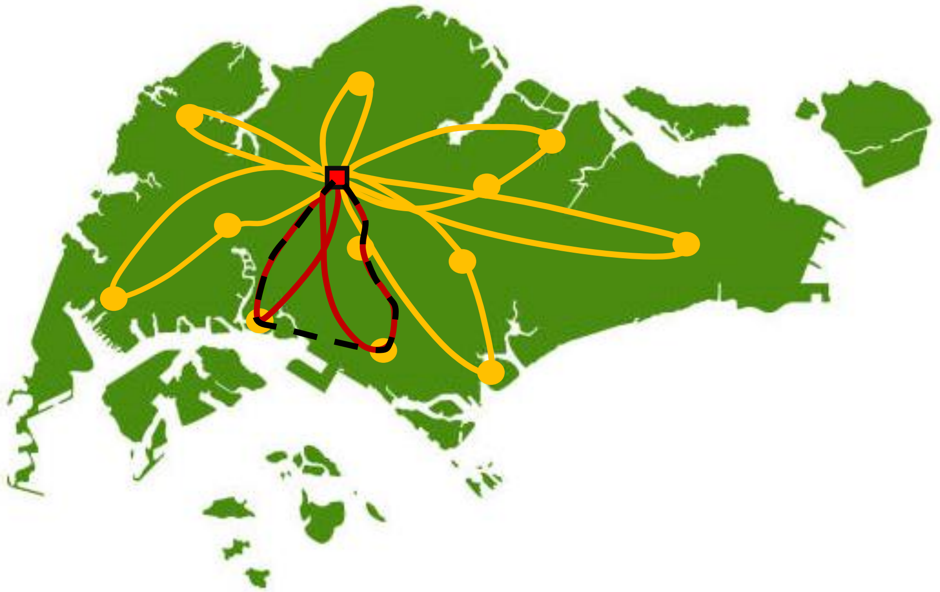
- Vehicle Routing - Clarke-Wright greedy algorithm
  - The procedure starts with each costumer being served by a single tour.
  - Cost savings $S_{ij} = c_{i0} + c_{j0} - c_{ij}$ can be obtained assuming c_{ij} is the cost of travelling from customer i to j (where $j = 0$ is the depot)
  - The savings are sorted in decreasing order. The procedure merges costumers i and j corresponding to the highest savings S_{ij} , without violating the capacity restrictions

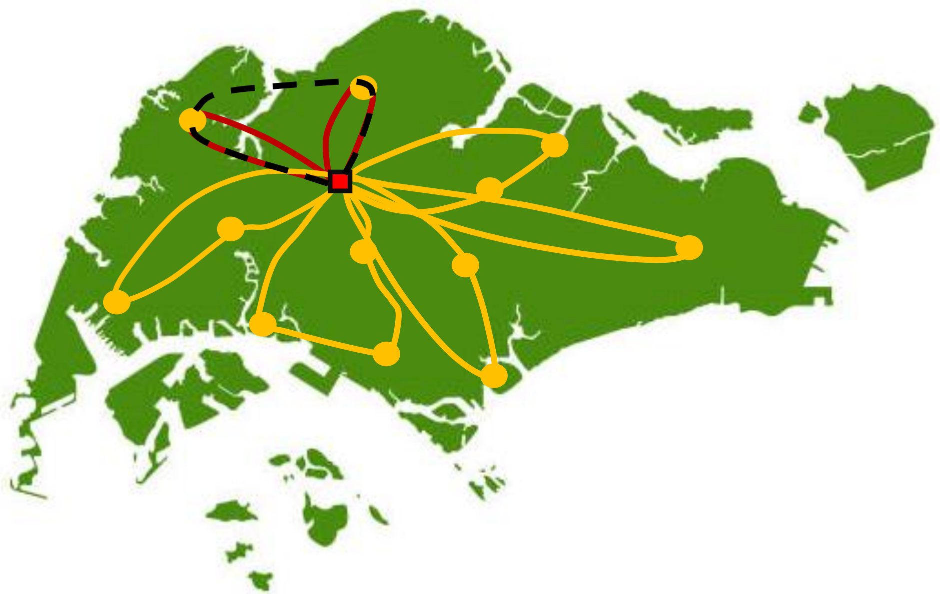




Other Greedy Heuristics

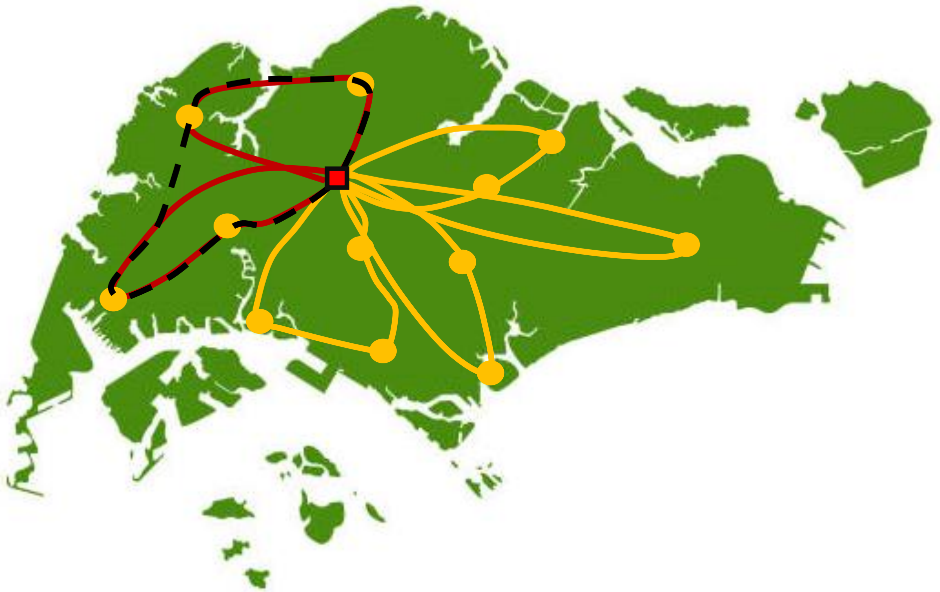
- Vehicle Routing - Clarke-Wright greedy algorithm
  - The procedure starts with each costumer being served by a single tour.
  - Cost savings $S_{ij} = c_{i0} + c_{j0} - c_{ij}$ can be obtained assuming c_{ij} is the cost of travelling from customer i to j (where $j = 0$ is the depot)
  - The savings are sorted in decreasing order. The procedure merges costumers i and j corresponding to the highest savings S_{ij} , without violating the capacity restrictions

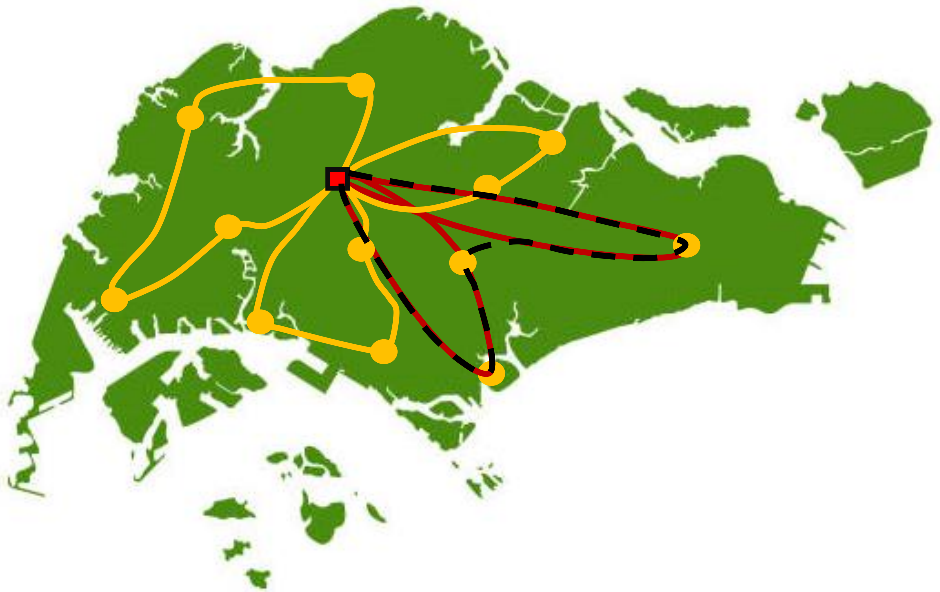




Other Greedy Heuristics

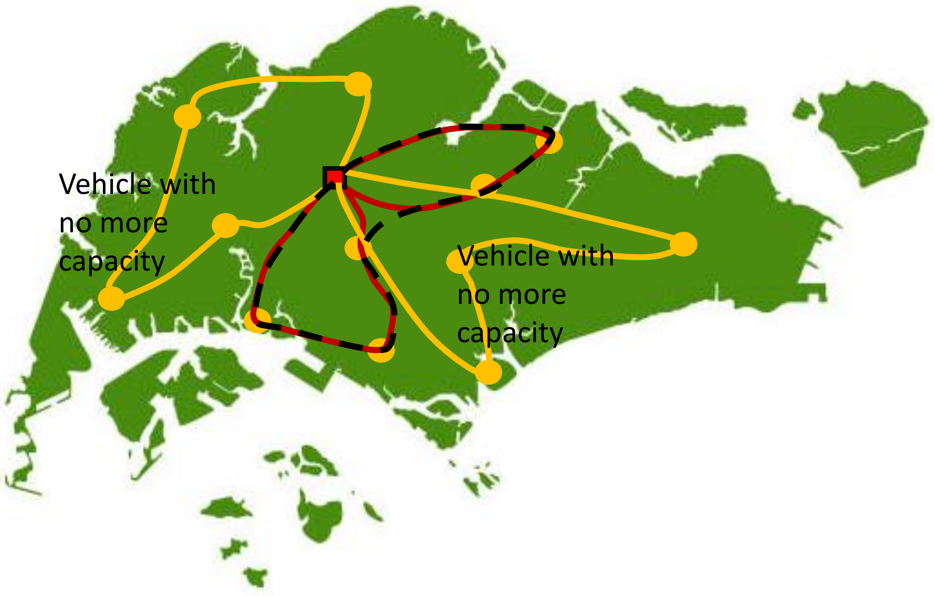
- Vehicle Routing - Clarke-Wright greedy algorithm
  - The procedure starts with each costumer being served by a single tour.
  - Cost savings $S_{ij} = c_{i0} + c_{j0} - c_{ij}$ can be obtained assuming c_{ij} is the cost of travelling from customer i to j (where $j = 0$ is the depot)
  - The savings are sorted in decreasing order. The procedure merges costumers i and j corresponding to the highest savings S_{ij} , without violating the capacity restrictions

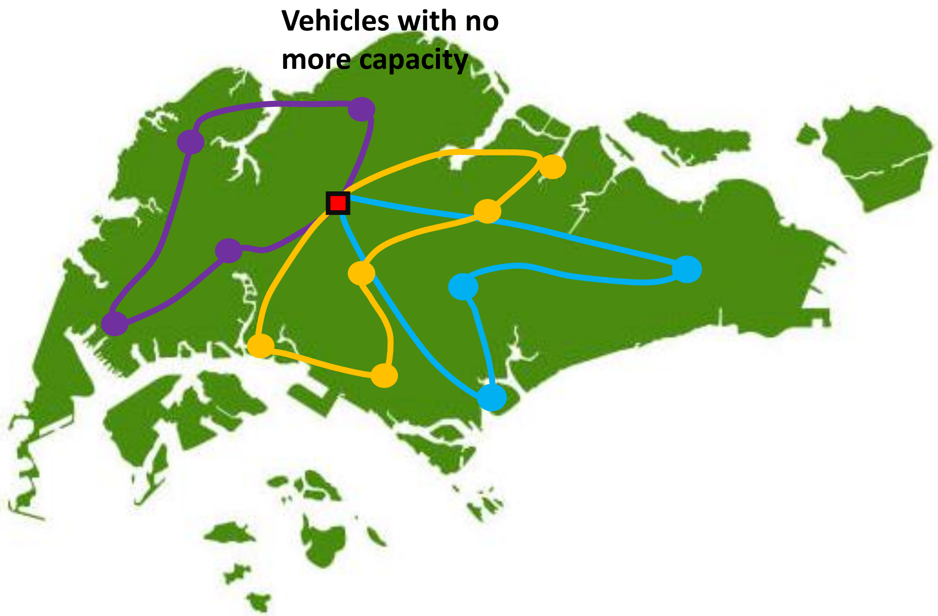




Other Greedy Heuristics

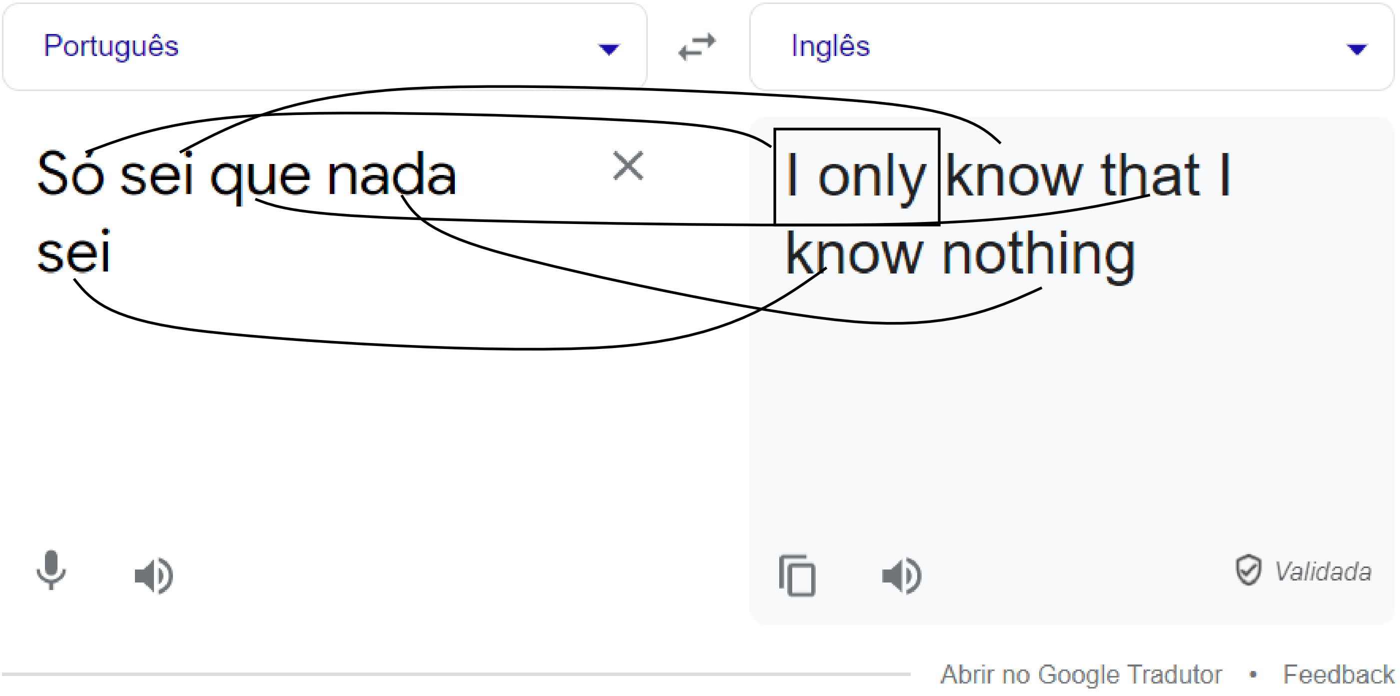
- Vehicle Routing - Clarke-Wright greedy algorithm
  - The procedure starts with each costumer being served by a single tour.
  - Cost savings $S_{ij} = c_{i0} + c_{j0} - c_{ij}$ can be obtained assuming c_{ij} is the cost of travelling from customer i to j (where $j = 0$ is the depot)
  - The savings are sorted in decreasing order. The procedure merges costumers i and j corresponding to the highest savings S_{ij} , without violating the capacity restrictions





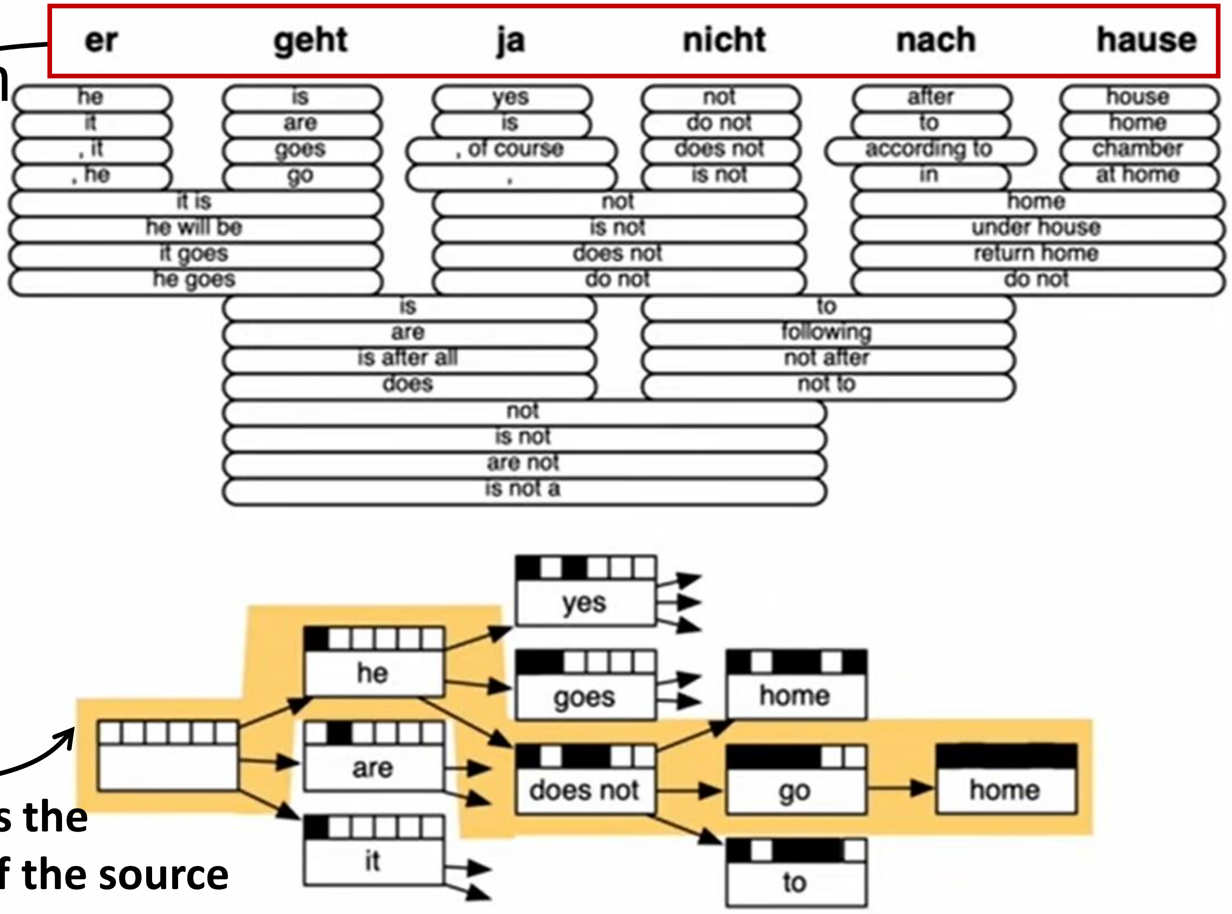
Machine Translation

- Machine Translation - Beam Search
  - Machine Translation is the task of translating a sentence *x* from one language (source language) to a sentence *y* in another language (the target language)



Machine Translation

- Machine Translation



Sentence to Translate

Trained Neural Machine Translation

Neural Network sequentially computes the probability of each word conditional of the source sentence

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}_{\text{Probability of next target word, given target words so far and source sentence } x}$$

Source: "Statistical Machine Translation", Chapter 6, Koehn, 2009.
https://www.cambridge.org/core/books/statistical-machine-translation/94EADF9F680558E13BE759997553CDE5

Source: https://www.youtube.com/watch?v=XXtpJxZBa2c

Beam Search

- Beam Search Greedy Algorithm: On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
  - K is the beam size (usually between 5 to 10)

- A hypotheses $y_1, \ldots, y_t$ has a score which is its log probability

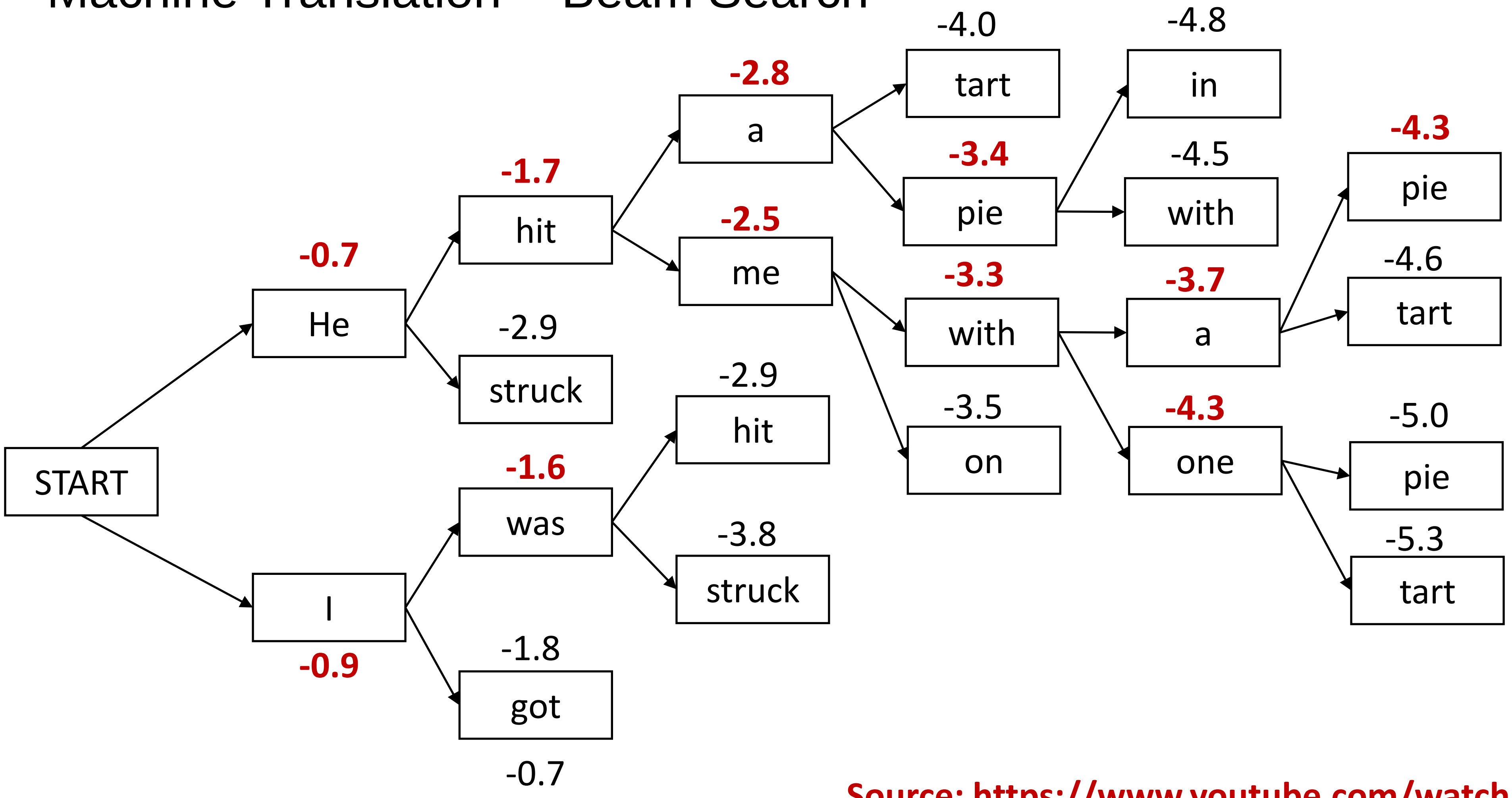
$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t|x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses

Source: https://www.youtube.com/watch?v=XXtpJxZBa2c

Other Greedy Heuristics

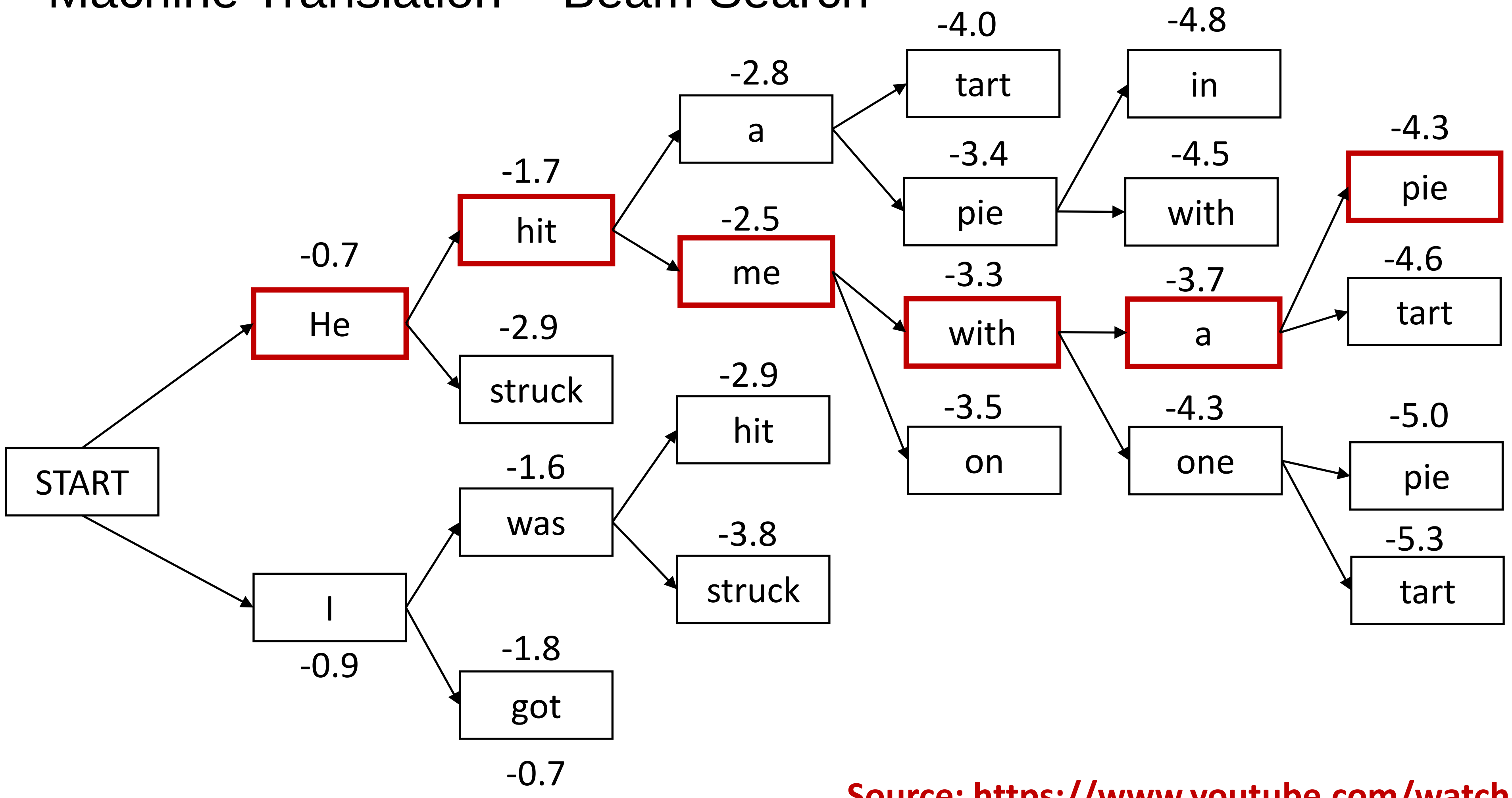
- Machine Translation - Beam Search



Source: https://www.youtube.com/watch?v=XXtpJxZBa2c

Other Greedy Heuristics

- Machine Translation - Beam Search

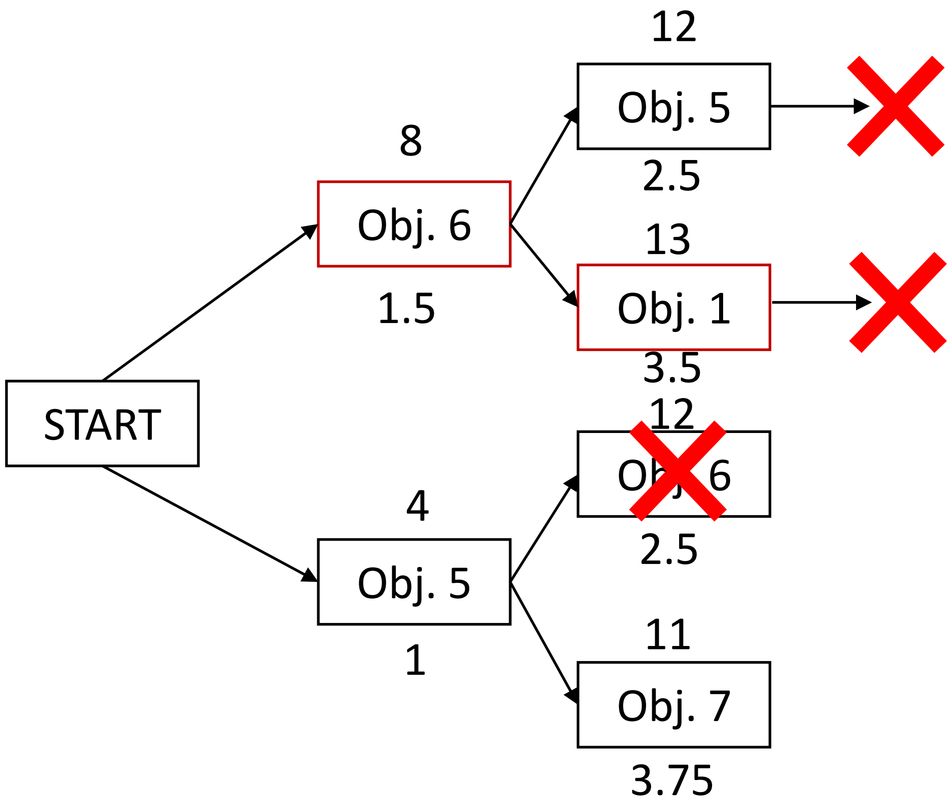


Source: https://www.youtube.com/watch?v=XXtpJxZBa2c

Other Greedy Heuristics

- Knapsack Problem - Beam Search

	w_i	f_i	f_i/w_i
Obj 1	2	5	2.5
Obj 2	3.75	7	1.87
Obj 3	2.5	3	1.2
Obj 4	3	5	1.67
Obj 5	1	4	4
Obj 6	1.5	8	5.33
Obj 7	2.75	7	2.54
cap	4		



Greedy Randomized Adaptive Search Procedure (GRASP)

Nuno Antunes Ribeiro

Assistant Professor

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Engineering Systems and Design

GRASP

- The GRASP metaheuristic is an iterative greedy heuristic to solve combinatorial optimization problems.
- Each iteration of the GRASP algorithm contains two steps: construction and local search
- In the construction step, a feasible solution is built using a **randomized greedy algorithm**, while in the next step a local search heuristic is applied from the constructed solution.
- The greedy algorithm must be randomized to be able to generate various solutions. Otherwise, the local search procedure can be applied only once.
- This approach is efficient if the **constructive heuristic designs different promising regions of the search space** that makes the different local searches generating different local optima of “good” quality

Restricted Candidate List

- In the constructive heuristic, at each iteration the elements that can be included in the partial solution are ordered in the list
- From this list, a subset is generated that represents the **restricted candidate list (RCL)**
- The RCL list is the key component of the GRASP metaheuristic. It represents the probabilistic aspect of the metaheuristic

GRASP

- Knapsack Problem - profit/weight randomized greedy algorithm
  - RCL includes the **top n objects** with the highest profit/weight value

	w_i	f_i	f_i/w_i
Obj 1	2	5	2.5
Obj 2	3.75	7	1.87
Obj 3	2.5	3	1.2
Obj 4	3	5	1.67
Obj 5	1	4	4
Obj 6	1.5	8	5.33
Obj 7	2.75	7	2.54
cap	**8**		

$n = 3$

$RCL_1 = \{Obj\ 6\ ;\ Obj\ 5\ ;\ Obj7\} \longrightarrow$ Randomly selected $Obj7$

$Profit = 0$

$Weight = 0$

GRASP

- Knapsack Problem - profit/weight randomized greedy algorithm
  - RCL includes the **top n objects** with the highest profit/weight value

	w_i	f_i	f_i/w_i
Obj 1	2	5	2.5
Obj 2	3.75	7	1.87
Obj 3	2.5	3	1.2
Obj 4	3	5	1.67
Obj 5	1	4	4
Obj 6	1.5	8	5.33
Obj 7	2.75	7	2.54
cap	**8**		

$n = 3$

Randomly selected

$RCL_1 = \{Obj\ 6\ ;\ Obj\ 5\ ;\ Obj7\} \longrightarrow Obj7$

$RCL_2 = \{Obj\ 6\ ;\ Obj\ 5\ ;\ Obj1\} \longrightarrow Obj5$

$Profit = 7$

$Weight = 2.75$

GRASP

- Knapsack Problem - profit/weight randomized greedy algorithm
  - RCL includes the **top n objects** with the highest profit/weight value

	w_i	f_i	f_i/w_i
Obj 1	2	5	2.5
Obj 2	3.75	7	1.87
Obj 3	2.5	3	1.2
Obj 4	3	5	1.67
Obj 5	1	4	4
Obj 6	1.5	8	5.33
Obj 7	2.75	7	2.54
cap	**8**		

$n = 3$

Randomly selected

$RCL_1 = \{Obj\ 6\ ;\ Obj\ 5\ ;\ Obj7\} \longrightarrow Obj7$

$RCL_2 = \{Obj\ 6\ ;\ Obj\ 5\ ;\ Obj1\} \longrightarrow Obj5$

$RCL_3 = \{Obj\ 6\ ;\ Obj\ 1\ ;\ Obj\ 2\} \longrightarrow Obj\ 1$

$Profit = 11$

$Weight = 3.75$

GRASP

- Knapsack Problem - profit/weight randomized greedy algorithm
  - RCL includes the **top n objects** with the highest profit/weight value

	w_i	f_i	f_i/w_i
Obj 1	2	5	2.5
Obj 2	3.75	7	1.87
Obj 3	2.5	3	1.2
Obj 4	3	5	1.67
Obj 5	1	4	4
Obj 6	1.5	8	5.33
Obj 7	2.75	7	2.54
cap	**8**		

$n = 3$

Randomly selected

$RCL_1 = \{Obj\ 6\,;\ Obj\ 5\,;\ Obj 7\} \longrightarrow Obj 7$

$RCL_2 = \{Obj\ 6\,;\ Obj\ 5\,;\ Obj 1\} \longrightarrow Obj 5$

$RCL_3 = \{Obj\ 6\,;\ Obj\ 1\,;\ Obj\ 2\} \longrightarrow Obj\ 1$

$RCL_4 = \{Obj\ 6\,\} \longrightarrow Obj\ 6$

$Profit = 16$

$Weight = 5.75$

GRASP

- Knapsack Problem - profit/weight randomized greedy algorithm
  - RCL includes the **top n objects** with the highest profit/weight value

	w_i	f_i	f_i/w_i
Obj 1	2	5	2.5
Obj 2	3.75	7	1.87
Obj 3	2.5	3	1.2
Obj 4	3	5	1.67
Obj 5	1	4	4
Obj 6	1.5	8	5.33
Obj 7	2.75	7	2.54
cap	**8**		

$n = 3$

Randomly selected

$RCL_1 = \{Obj\ 6\ ;\ Obj\ 5\ ;\ Obj7\} \longrightarrow Obj7$

$RCL_2 = \{Obj\ 6\ ;\ Obj\ 5\ ;\ Obj1\} \longrightarrow Obj5$

$RCL_3 = \{Obj\ 6\ ;\ Obj\ 1\ ;\ Obj\ 2\} \longrightarrow Obj\ 1$

$RCL_4 = \{Obj\ 6\ \} \longrightarrow Obj\ 6$

$Profit = 24$

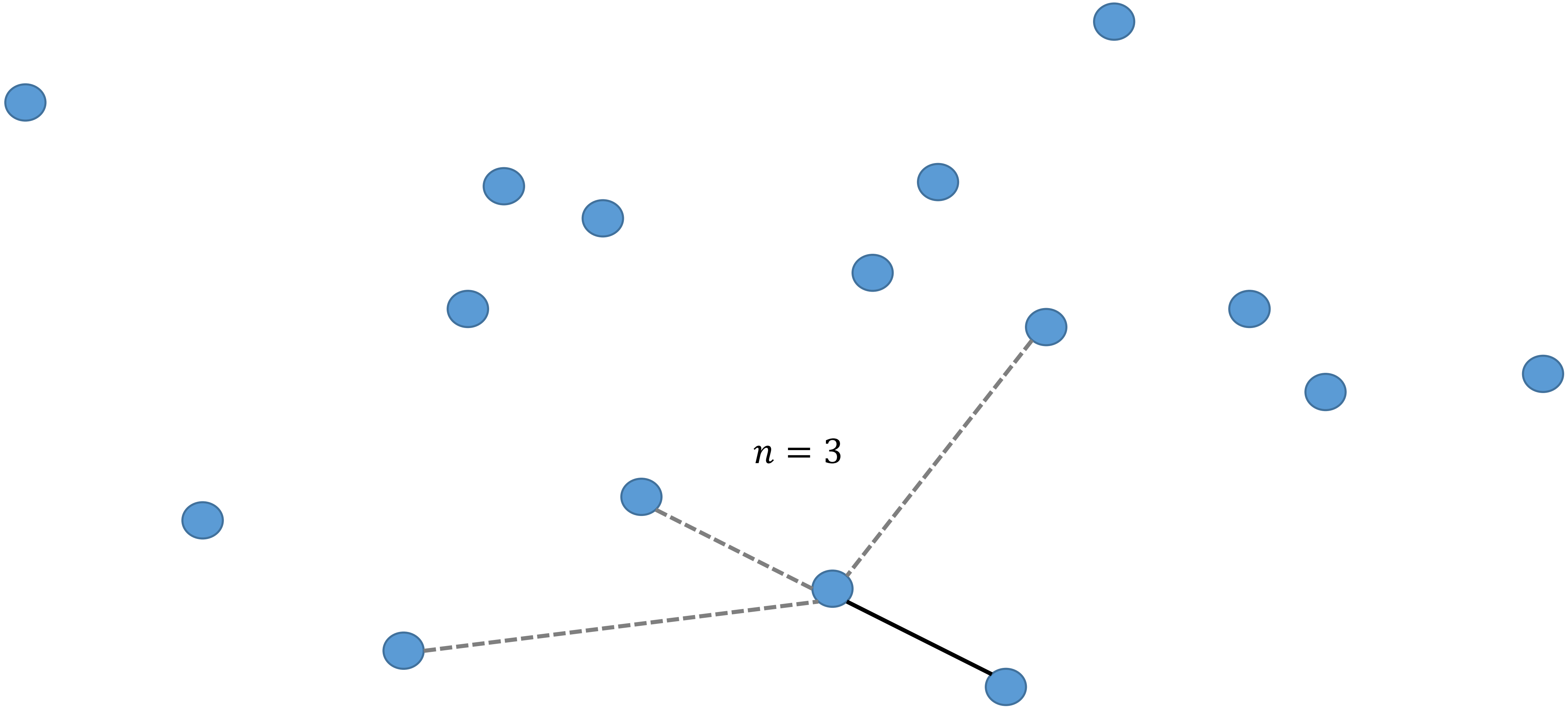
$Weight = 7.25$

Restriction Criteria

- The restriction criteria:
  - Cardinality-based criteria: The RCL list is made of the n best elements in terms of the incremental cost, where the parameter p represents the maximum number of elements in the list.
  - Value-based criteria: It consists in selecting the solutions that are better than a given threshold value.

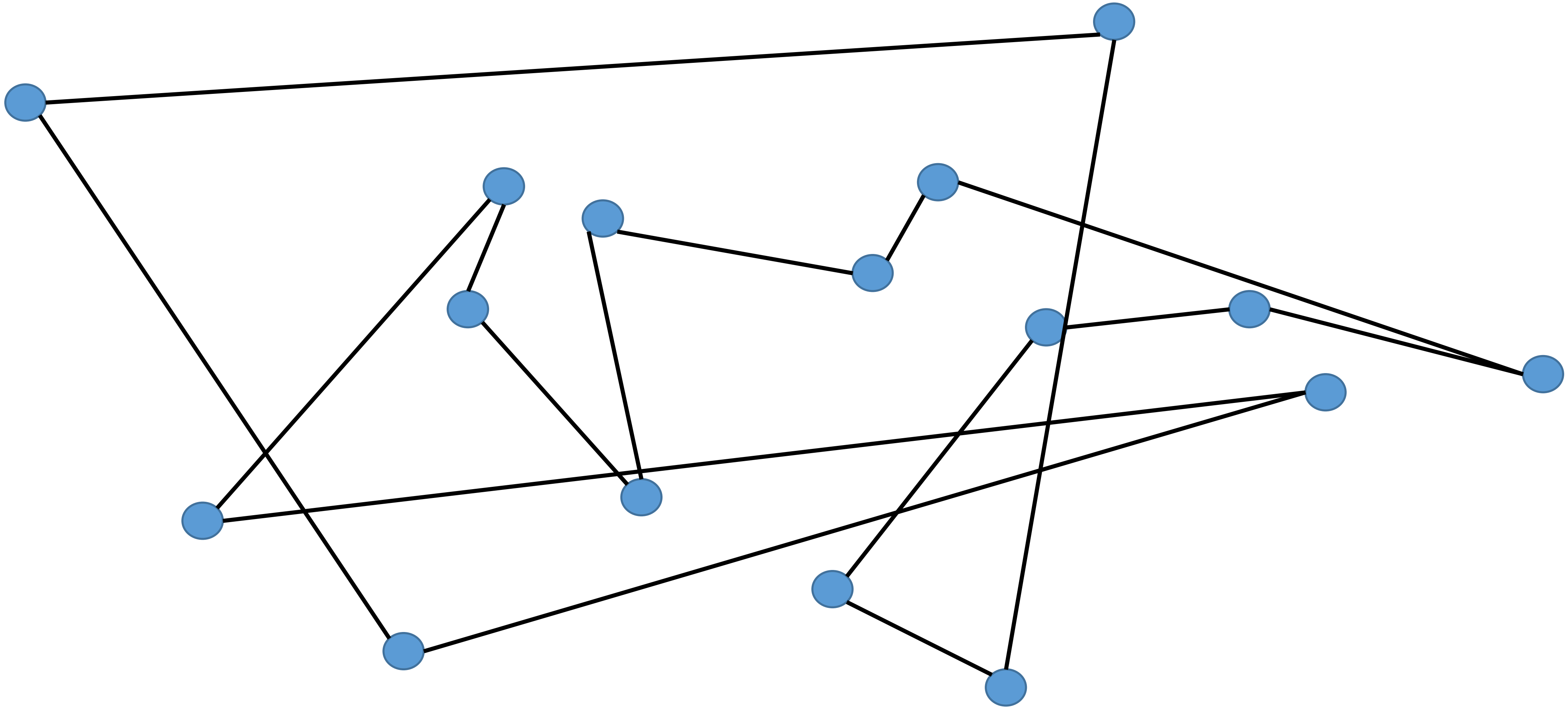
GRASP

- Randomized Nearest Neighbour Search



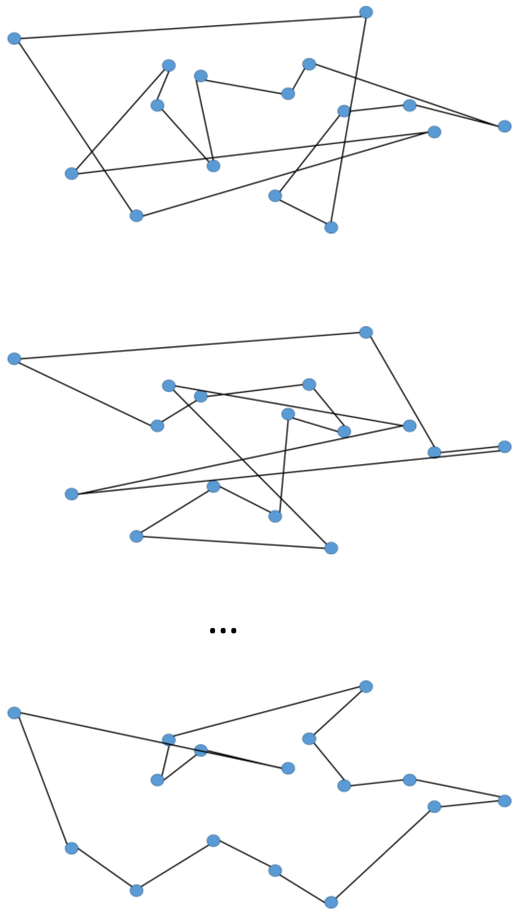
GRASP

- Randomized Nearest Neighbour Search

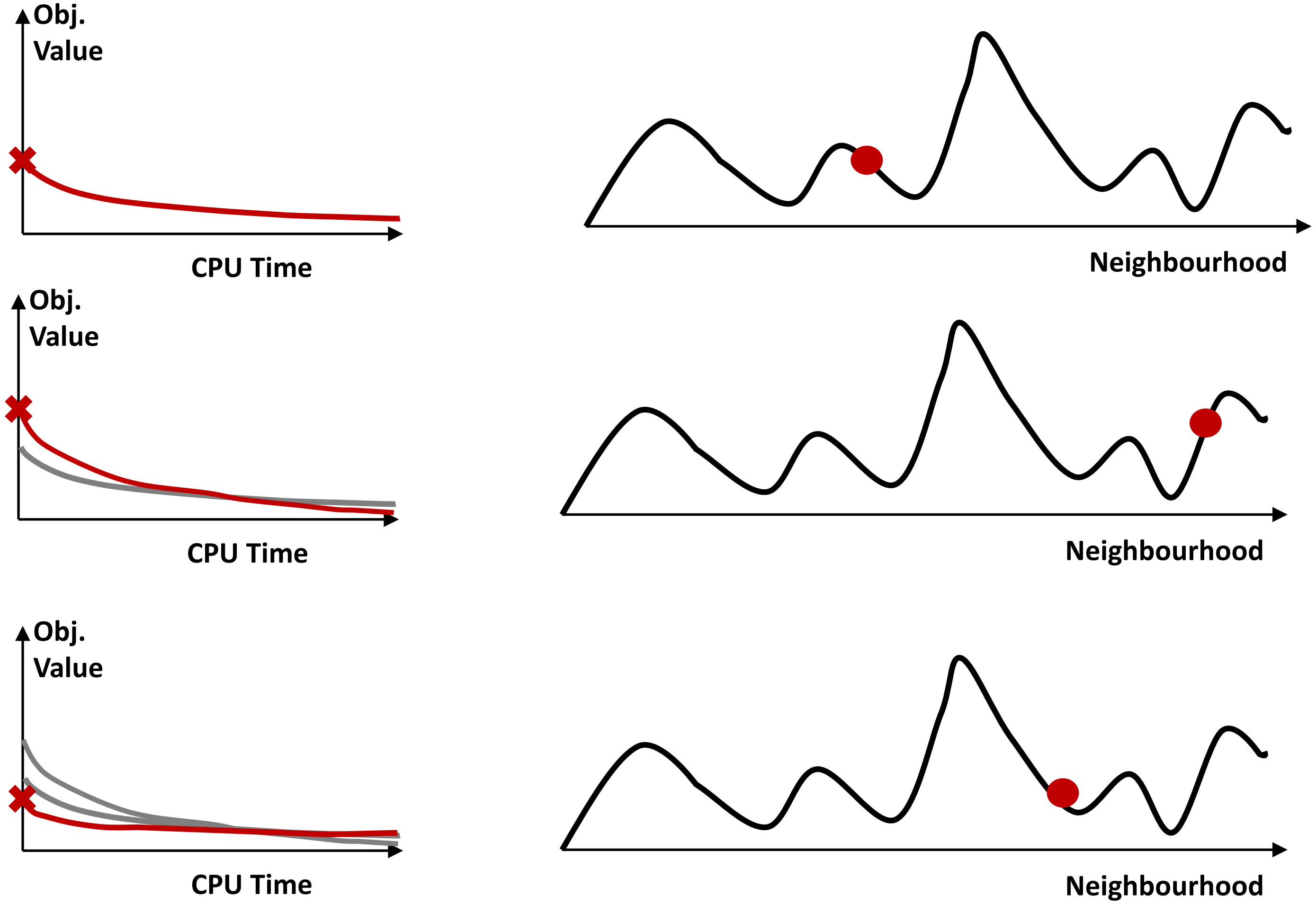


Grasp

Constructive Heuristic



Improvement Heuristic



Iterations

Project Guidelines and Ideas

Nuno Antunes Ribeiro

Assistant Professor

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN | Engineering Systems and Design

Project Paper

- **Project:** The objective of the project is to enable students to design, propose and adapt metaheuristic solutions to solve a real-world challenge. The topic can be selected according to the students' interest. A **maximum of three students** per project team is allowed. Project deliverables include **a research paper (5-15 pages):**
  1. Abstract – 1 paragraph
  2. Introduction / literature review – 1 to 2 pages
  3. Optimization problem formulation – 1 to 2 pages
  4. Solution encoding and search operators 1 – 3 pages
  5. Summary of the metaheuristics used, and heuristic rules applied 2 – 4 pages
  6. Results 1 – 3 pages

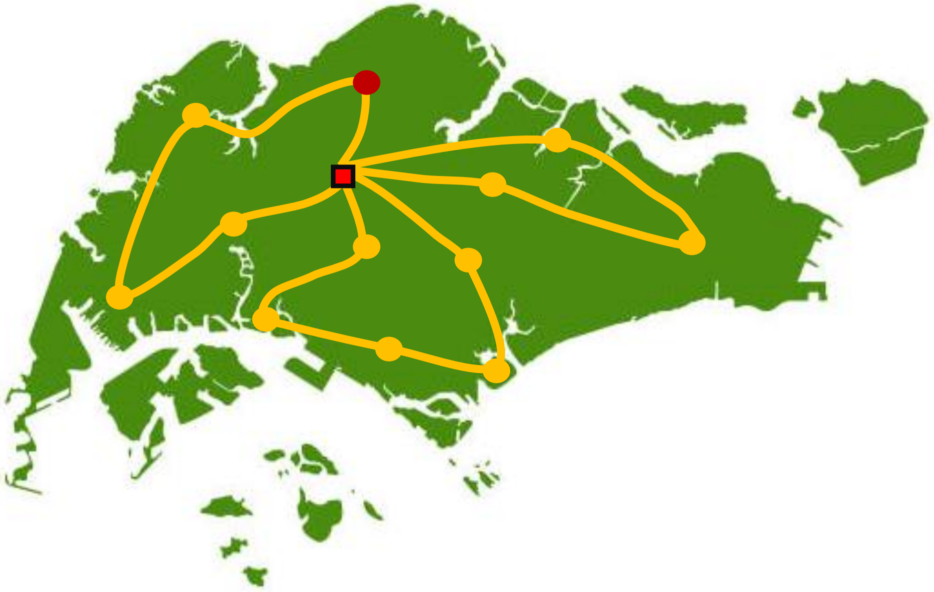
Grading Rubrics

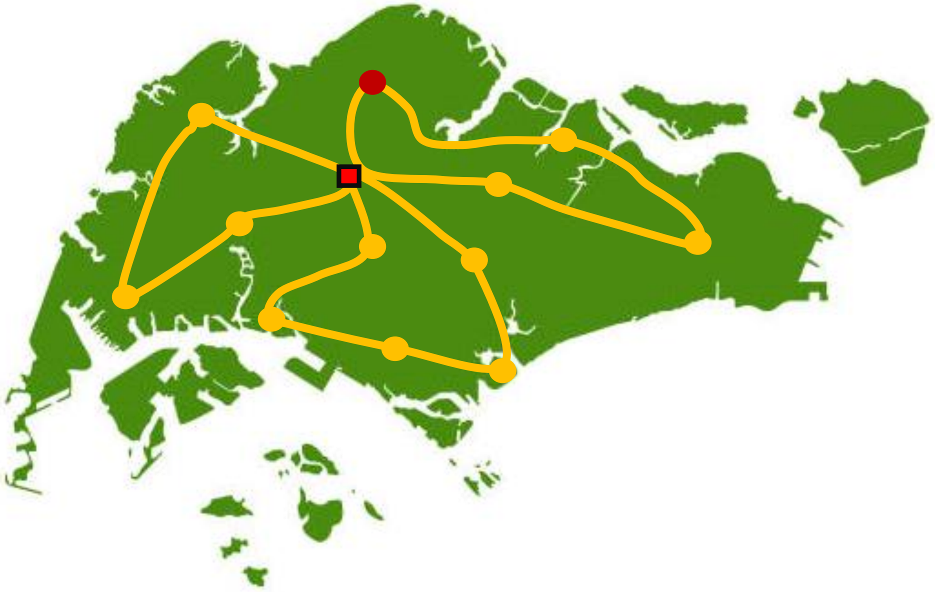
- Quality of the research paper
- Complexity of the search operators and solution encoding
- Number of metaheuristics implemented
- Complexity of the heuristic rules implemented within each metaheuristic
- Discussion of the results

Routing Problems

Many different Search Operators have been proposed for the Vehicle Routing Problem

- **Relocate:** This operator consists in relocating one customer from one route to another.



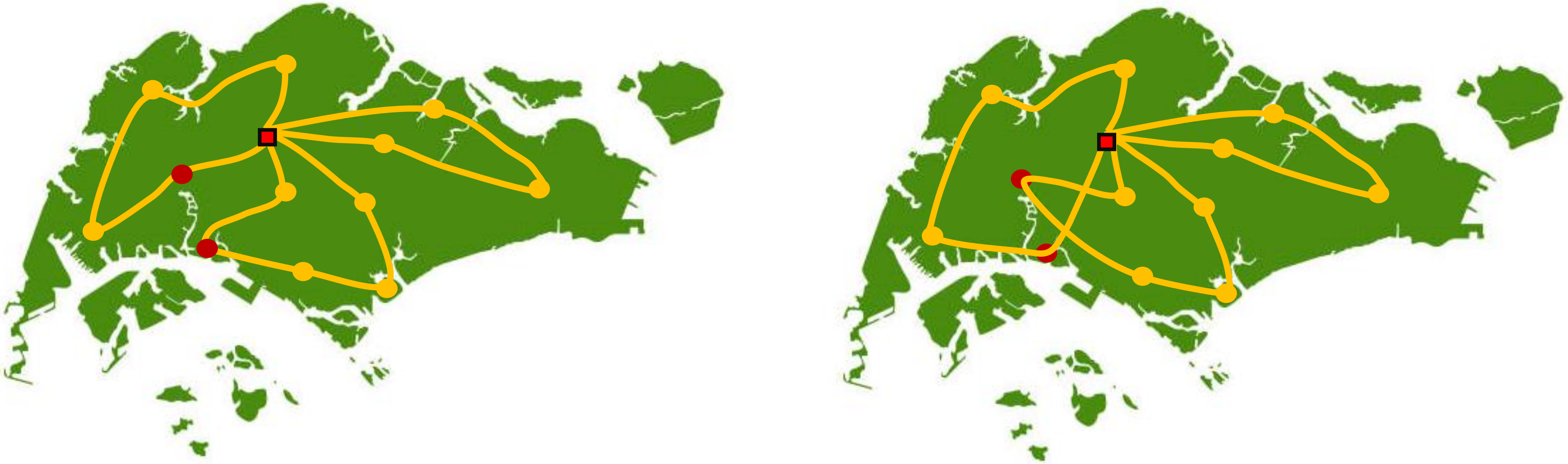


Data Instances: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Routing Problems

Many different Search Operators have been proposed for the Vehicle Routing Problem

- **Swap:** This operator consists in swapping two customers from different routes. It can be seen as a double relocation in which the customers are inserted at their counterpart's position in the route.

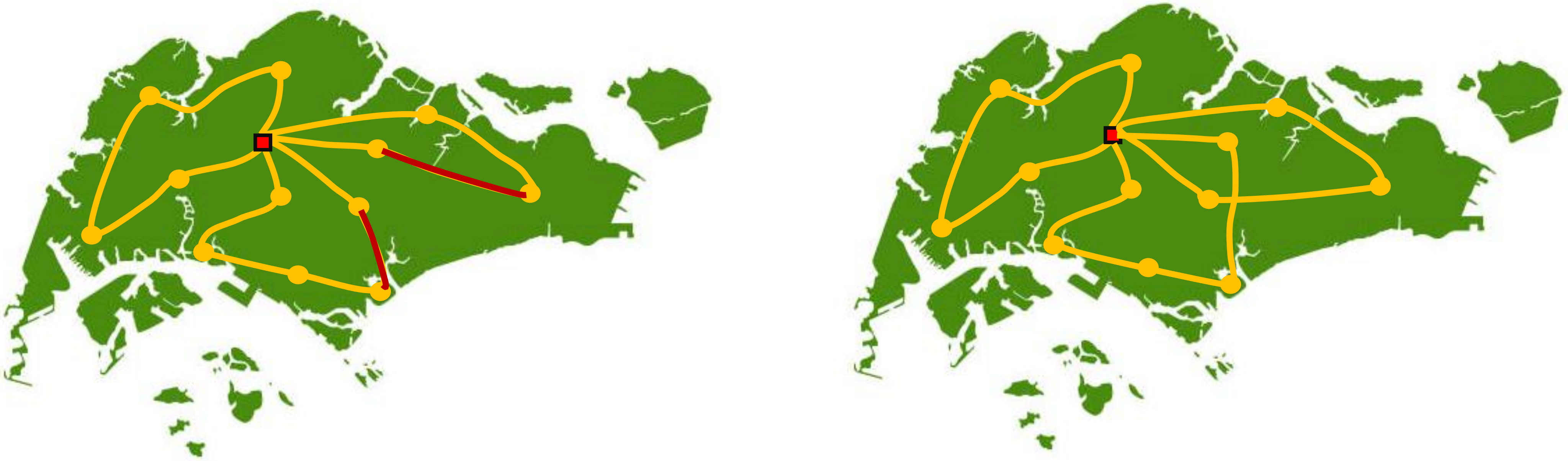


Data Instances: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Routing Problems

Many different Search Operators have been proposed for the Vehicle Routing Problem

- **K-Opt:** This operator, which is also called k-opt, consists in dropping k edges in the same route and then reconnecting the resulting segments by other edges.

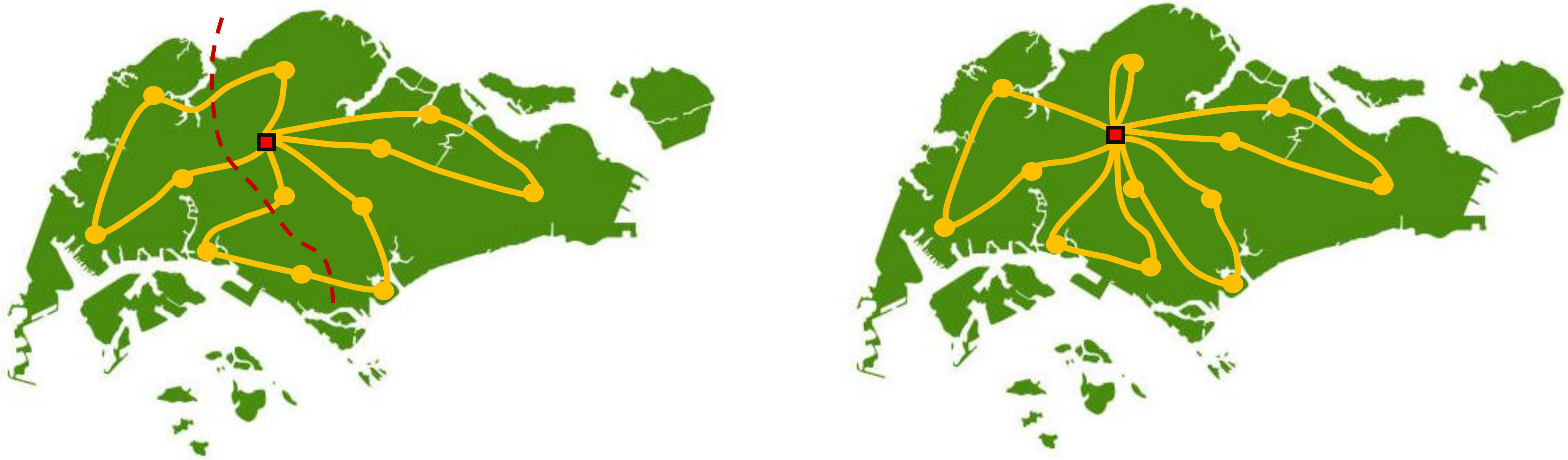


Data Instances: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Routing Problems

Many different Search Operators have been proposed for the Vehicle Routing Problem

- **Cross-Operator:** This operator cuts two different routes in two parts and recompose them with crossing edges.

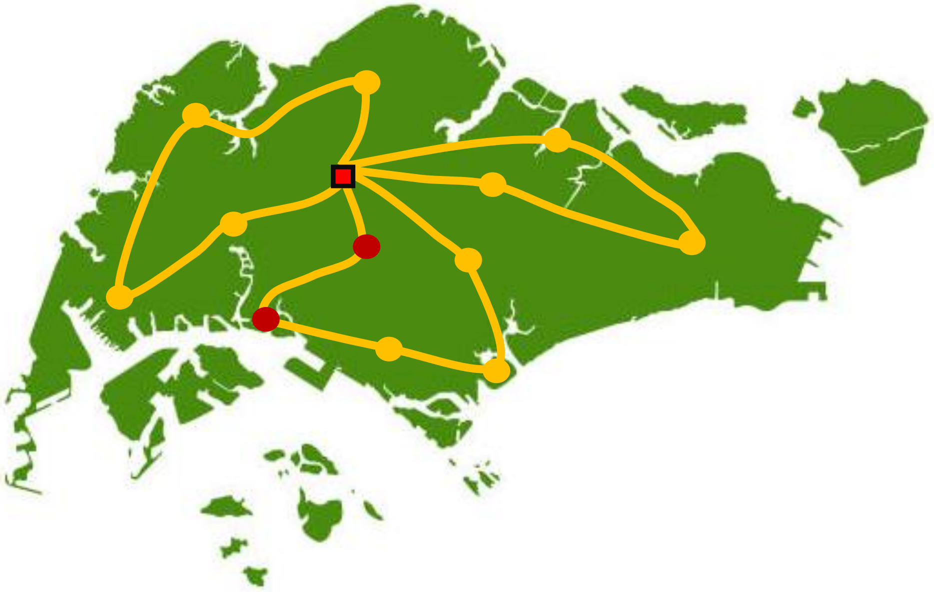


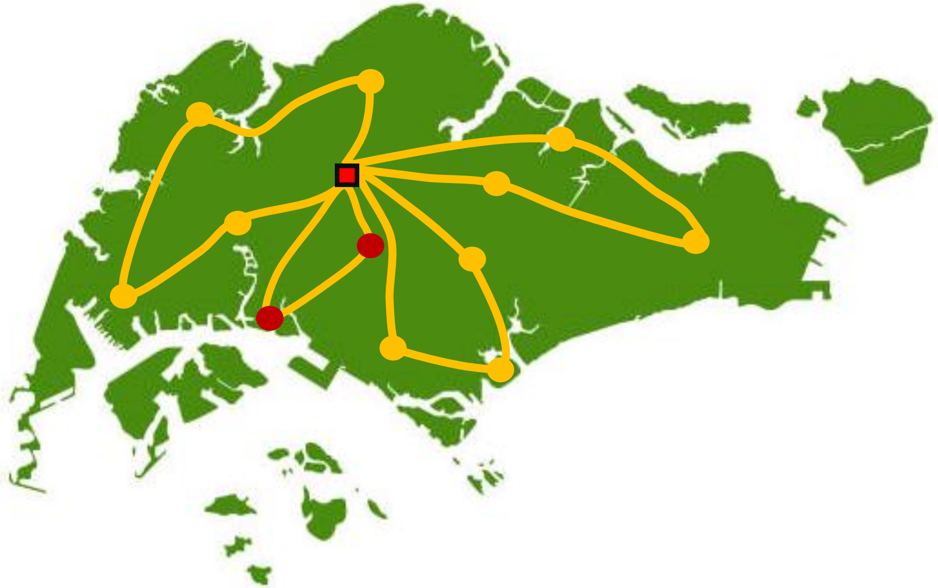
Data Instances: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Routing Problems

Many different Search Operators have been proposed for the Vehicle Routing Problem

- **Split-to-single:** A pair of demand centers are selected and combined to create a new route





Data Instances: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Routing Problems

TSP/Vehicle Routing Extensions

- Demand centres may have time window constraints
- Consideration of pick-up and deliveries
- Multi depot
- Bin packing (see next slide)

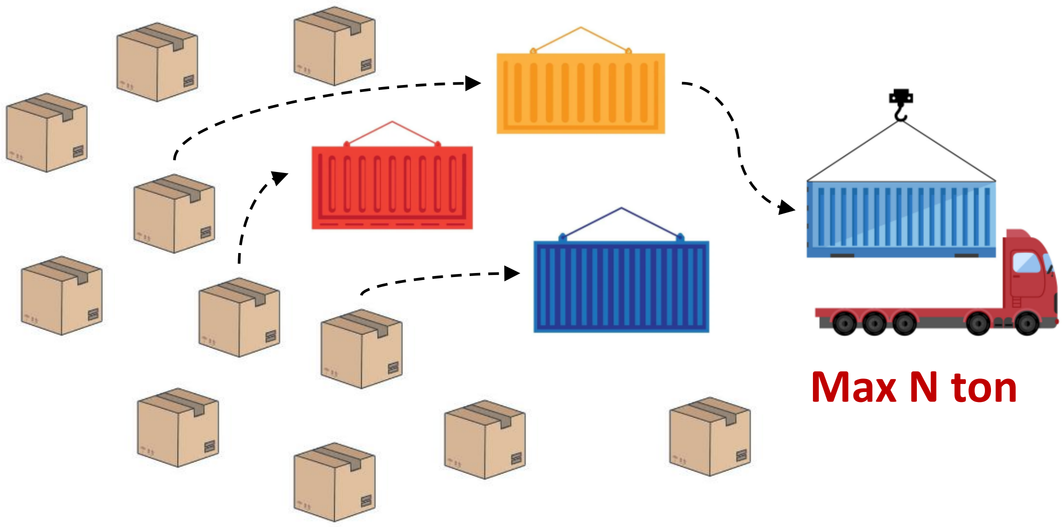
Data Instances: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

+ Bin Packing Problem

Possible metaheuristic procedure

1. Initial solution generated using greedy algorithm
2. Each bin of the current solution is successively eliminated and its items are redistributed to other bins. If the new solution is feasible, we move to that solution. Otherwise a metaheuristic method is used to reduce infeasibility
3. Metaheuristic: Items from a bin are removed and moved to other bins. Whenever a feasible solution is obtained, return to step 2.

Objective minimize the number of bins required

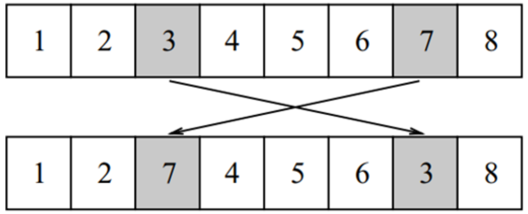


Scheduling Problems

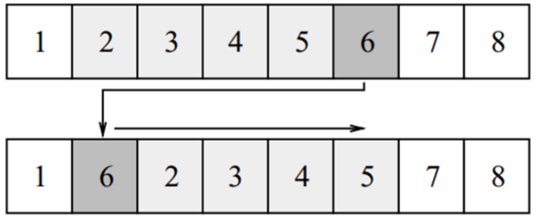
- **Job Shop Scheduling**

Machine 0		Machine 1			Machine 2		
1	0	2	1	0	1	2	0

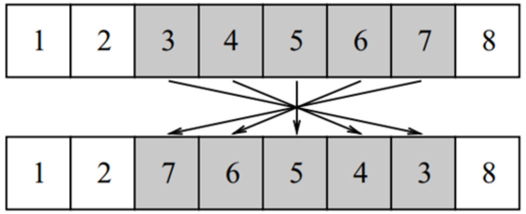
- Swap/Exchange Operator

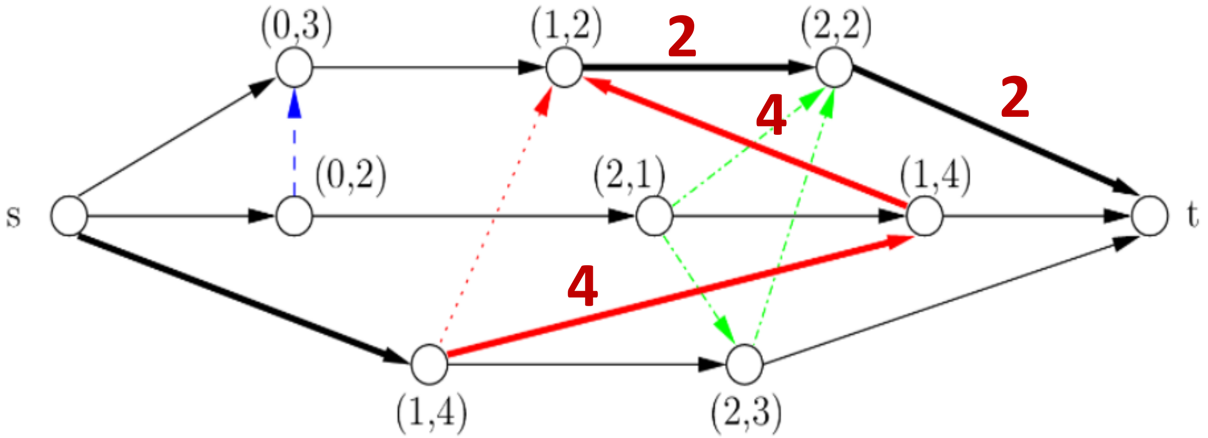


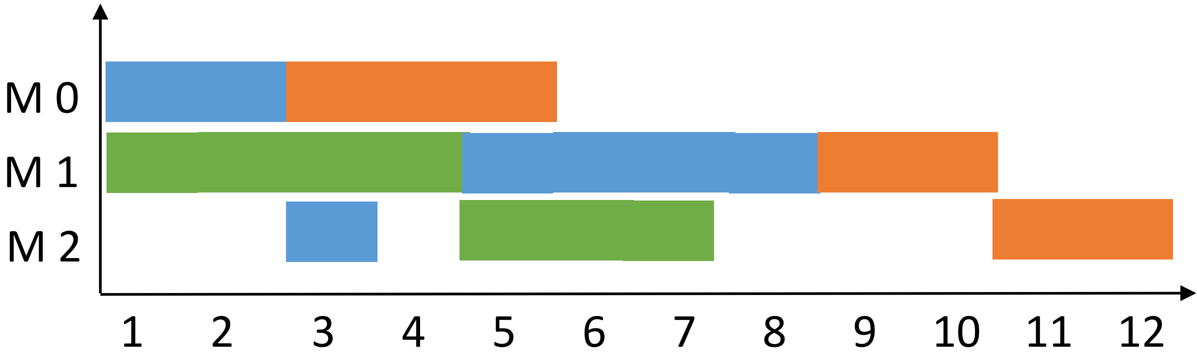
- Insertion Operator



- Inversion Operator







Data Instances: http://optimizizer.com/DMU.php

Scheduling Problems

- **Sport's Scheduling**
- Inputs:
  - n teams
  - A matrix d of distances between teams
- Constraints:
  - Teams play with each other team twice (home or away)
  - Teams cannot play more than 3 consecutive games at home or away
  - Two teams cannot play with each other in consecutive weeks (i.e. a game a @ b cannot be followed by a game b @ a
- Objective
  - Minimize travel distance

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:

- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Objective function:

$$d_{12} + d_{21} + d_{15} + d_{54} + d_{43} + d_{31} + d_{16} + d_{61}$$

$$+ \ldots +$$

$$d_{61} + d_{14} + d_{45} + d_{56} + d_{63} + d_{36} + d_{62} + d_{26}$$

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	@4	3	6	4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:

- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:
- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:
- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Repair Procedure

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:
- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Repair Procedure

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	5	4	3	@2	@4	@3	2	@5	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	@2	@5	@1	6	5	1	@6	2	4
4	3	6	@1	2	5	1	@2	5	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	2	@5	@3	@2	5	3	4	1

Search Operators:

- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Repair Procedure

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@6	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@1	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:

- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@1	@5	4	3	6	@4	@6	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@3	@6	@2	1	5	2	@1	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:

- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Sport's Scheduling

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@1	@5	4	3	6	@4	@6	@3
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@3	@6	@2	1	5	2	@1	@4
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Search Operators:

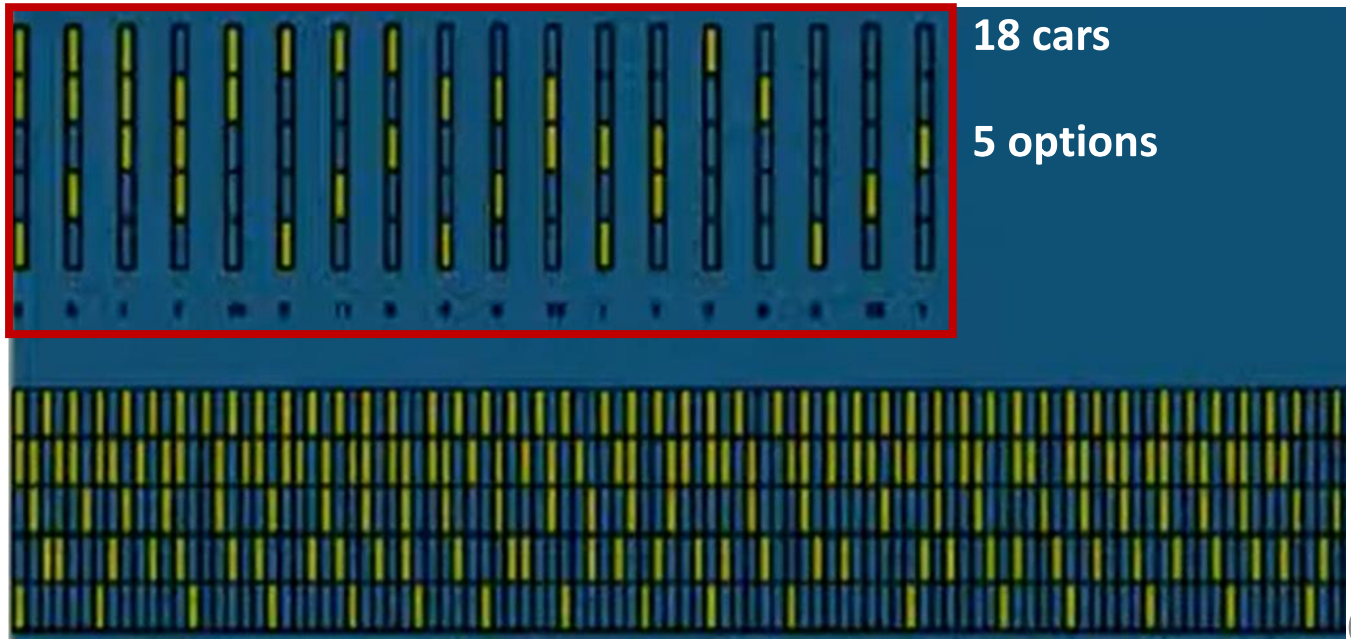
- Swap homes
- Swap teams
- Partial swap teams
- Swap rounds
- Partial Swap teams

Source: https://www.youtube.com/watch?v=MqSyOP-TpCs&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=26

Data Instances: https://mat.tepper.cmu.edu/TOURN/

Assembly Line

- Example: Car Sequencing Problem
  - Cars require different options: leather seats, moonroof etc.
  - Capacity constraints on the production units (e.g. a t most 2 out of 5 successive cars can require a moon roof
  - Objective: sequence all the cars such that the capacity constraints are satisfied



Assembly Line

Options Car	1	2	3	4	5	Demand
1	Yes		Yes	Yes		1
2				Yes		1
3		Yes			Yes	2
4		Yes		Yes		2
5	Yes		Yes			2
6	Yes	Yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Source: https://www.youtube.com/watch?v=yxiTNuqwjZM&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=11

Assembly Line

Slots	1	2	3	4	5	6	7	8	9	10	Demand
1											1
2											1
3											2
4											2
5											2
6											2

Setup	1	2	3	4	5	6	7	8	9	10	Demand
Opt 1											1
Opt 2											1
Opt 3											2
Opt 4											2
Opt 5											2

Options Car	1	2	3	4	5	Demand
1	Yes		Yes	Yes		1
2				Yes		1
3		Yes			Yes	2
4		Yes		Yes		2
5	Yes		Yes			2
6	Yes	Yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Source: https://www.youtube.com/watch?v=yxiTNuqwjZM&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=11

Assembly Line

Options Car	1	2	3	4	5	Demand
1	Yes		Yes	Yes		1
2				Yes		1
3		Yes			Yes	2
4		Yes		Yes		2
5	Yes		Yes			2
6	Yes	Yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Slots	1	2	3	4	5	6	7	8	9	10	Demand
1											1
2											1
3											2
4											2
5											2
6											2

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Opt 1											1/2
Opt 2											2/3
Opt 3											1/3
Opt 4											2/5
Opt 5											1/5

Constraint Violations

3

2

2

2

3

Source: https://www.youtube.com/watch?v=yxiTNuqwjZM&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=11

Assembly Line

Options Car	1	2	3	4	5	Demand
1	Yes		Yes	Yes		1
2				Yes		1
3		Yes			Yes	2
4		Yes		Yes		2
5	Yes		Yes			2
6	Yes	Yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Slots	1	2	3	4	5	6	7	8	9	10	Demand
1											1
2											1
3											2
4											2
5											2
6											2

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Opt 1											1/2
Opt 2											2/3
Opt 3											1/3
Opt 4											2/5
Opt 5											1/5

Constraint Violations

3

2

2

2

3

Source: https://www.youtube.com/watch?v=yxiTNuqwjZM&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=11

Assembly Line

Options Car	1	2	3	4	5	Demand
1	Yes		Yes	Yes		1
2				Yes		1
3		Yes			Yes	2
4		Yes		Yes		2
5	Yes		Yes			2
6	Yes	Yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Slots	1	2	3	4	5	6	7	8	9	10	Demand
1											1
2											1
3											2
4											2
5											2
6											2

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Opt 1											1/2
Opt 2											2/3
Opt 3											1/3
Opt 4											2/5
Opt 5											1/5

Constraint Violations

1

1

0

2

0

Source: https://www.youtube.com/watch?v=yxiTNuqwjZM&list=PLNMgVqt8MREx6Nex1Q9003vrZem-JXNvX&index=11

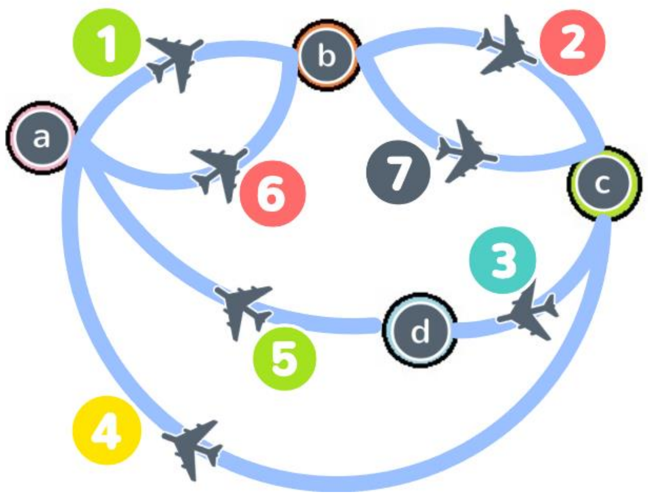
Airline Crew Scheduling

- The aim of crew pairing is to find a minimum cost set of pairings that cover all flights for a scheduling period (usually one month).
- Each pairing must satisfy all relevant regulations, for instance:
  - A pairing must begin and end at the same city
  - A pairing shall not comprise more than 48 hours
  - The number of flights in a pairing shall not be >4 times

Flight Schedule

1. City A -> City B	08:00 – 09:00
2. City B -> City C	10:00 – 11:00
3. City C -> City D	13:00 – 14:00
4. City C -> City A	07:00 – 08:00
5. City D -> City A	07:00 – 08:00
6. City A -> City B	10:00 – 11:00
7. City B -> City C	11:00 – 12:00

Flight Network



Possible Pairings

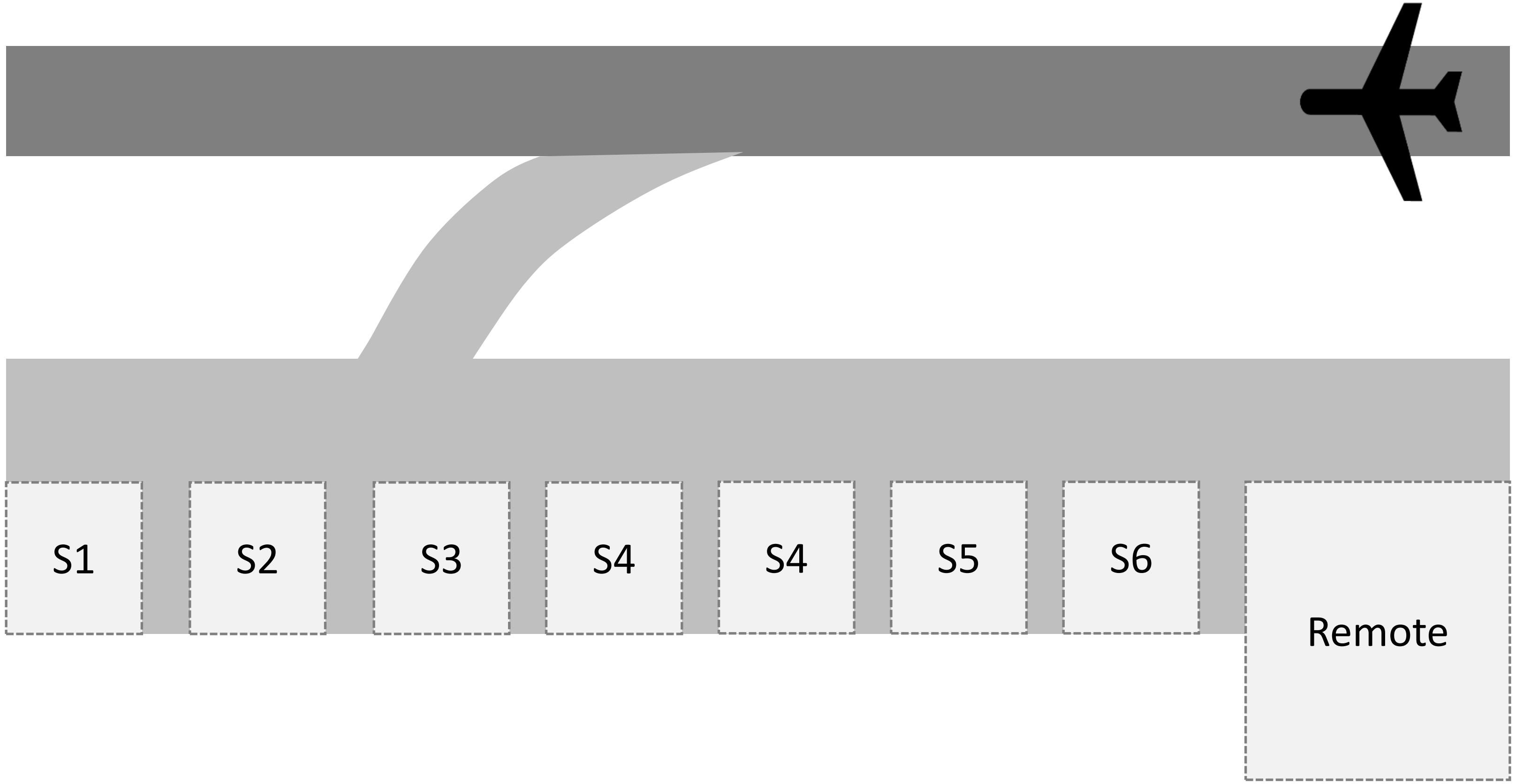
P1 = { 1 7 4 }

a → b → c → a

P2 = { 2 3 5 6 }

b → c → d → a → b

Aircraft Stand Allocation



Aircraft	Arr.	Dep.	Stand
1	0800	1000	
2	0810	0945	
3	0813	0910	
4	0815	1415	?
5	0818	1120	
…	…	…	
1000	2347	0815	

- Optimize allocation of stands to aircraft, aiming at minimizing taxi times and aircraft allocated to remote stands

Feature Selection in Machine Learning

- Apply metaheuristics for optimal selection of features in machine learning models
- Example: Machine learning model to predict flight delays at Changi Airport
  - Explanatory Variables

    Number of flights

    Lagged number of flights

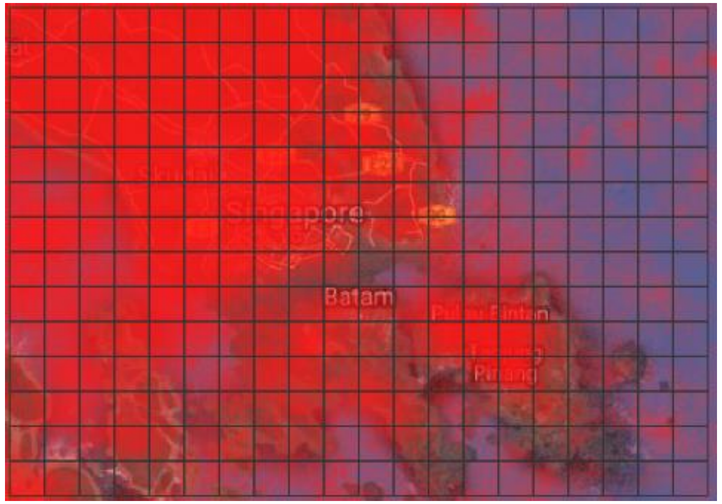
    Time of the day

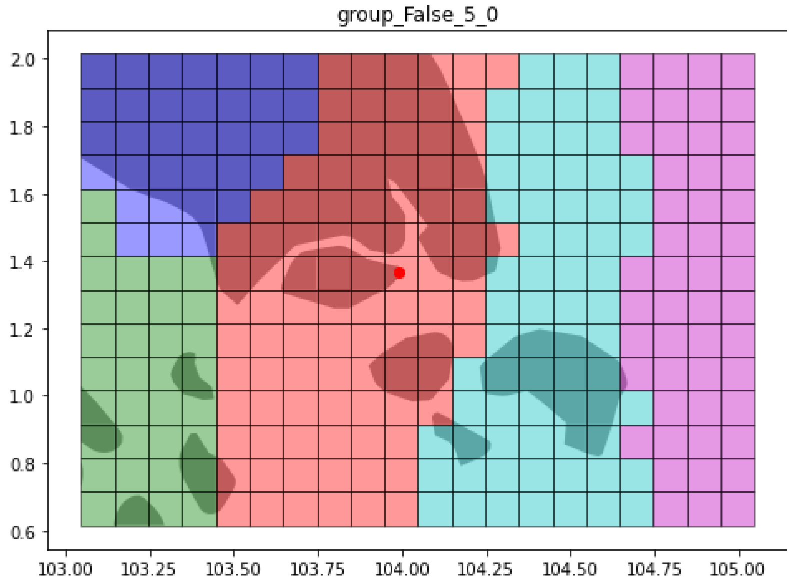
    % Arrivals and Departures

    Wind conditions

    Cluster-based lightning variables

    etc.





Machine Translation

- Applying a GRASP procedure instead of a pure greedy beam search approach
- Applying other metaheuristic procedures