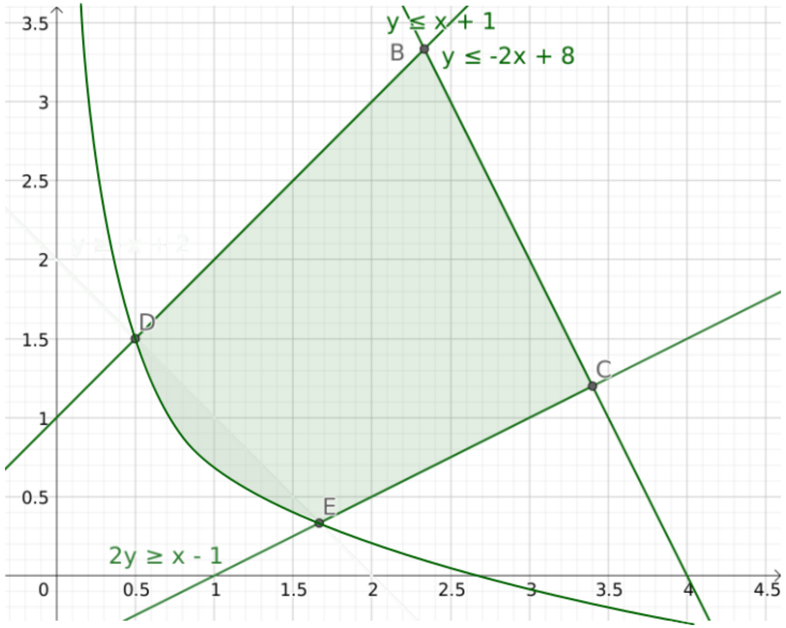
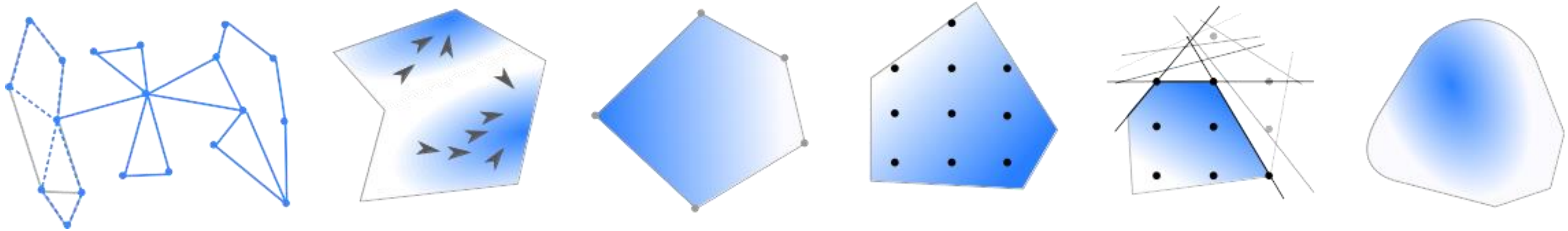
Construction Algorithms

COMP4691 / 8691





Previously on COMP4691/8691

Construction

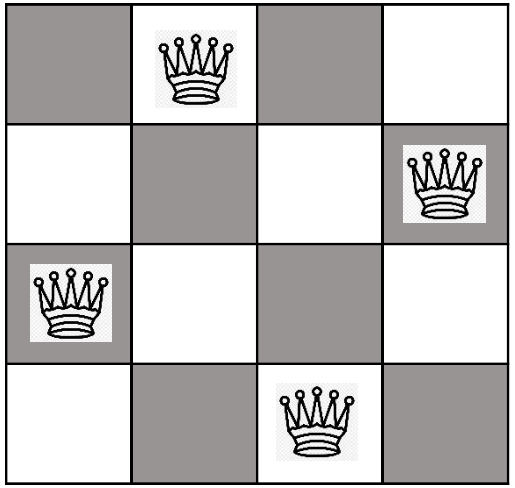
- Greedy construction
- Regret
- Bespoke

Today:

- **Improve – locally**

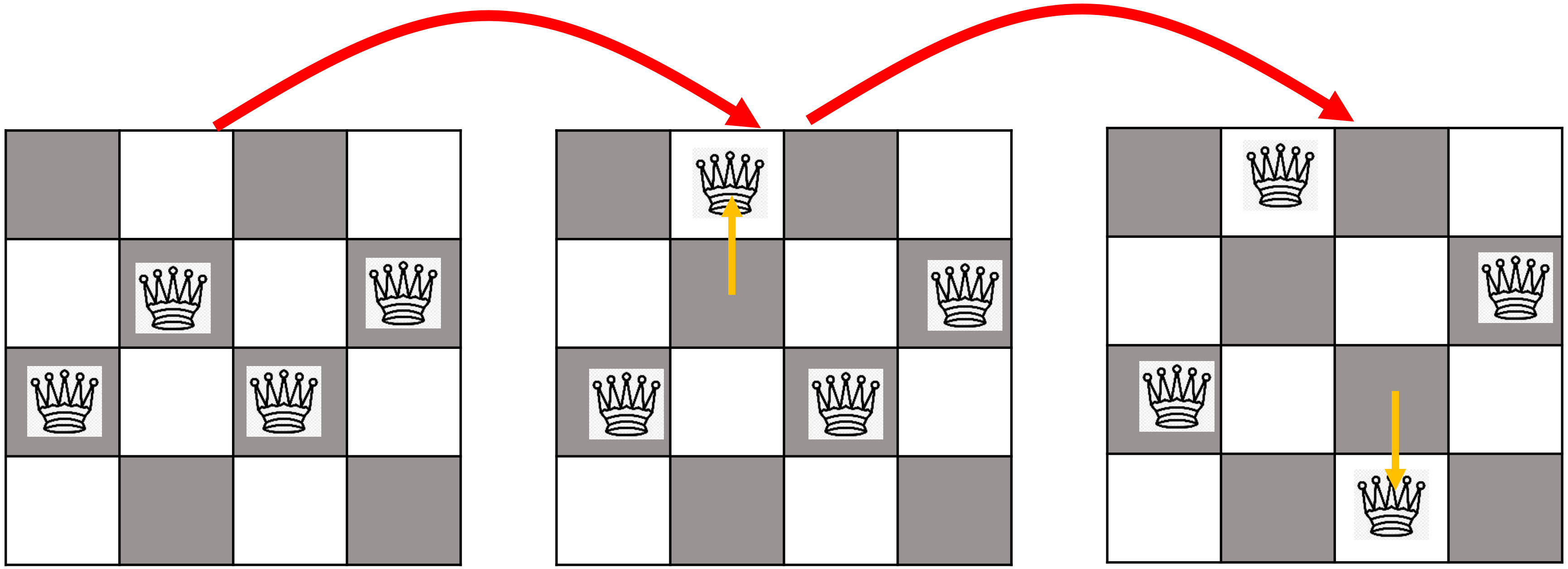
Local Search

- Move from a solution to its *neighbour*
- Neighbour defined by an operator
- E.g. n-queens problem
  - Place *n* queens on an *n* x *n* board, so that none can attack another (i.e., no two on the same row, column, or diagonal)



n-queens

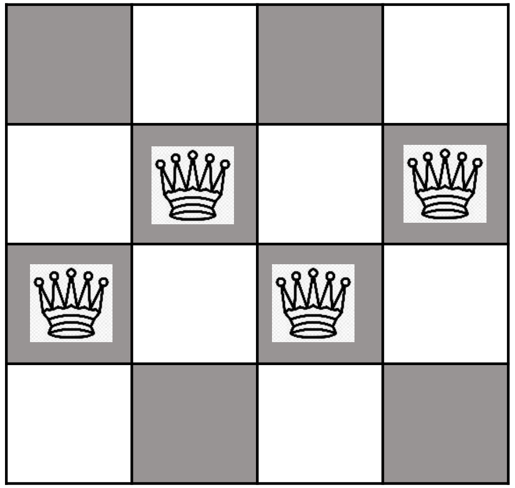
- Neighbourhood: All solutions that can be achieved by moving one queen to a different row

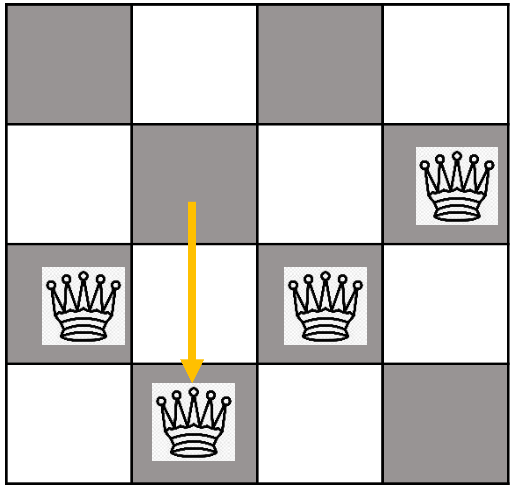


- Solution in 2 moves

n-queens

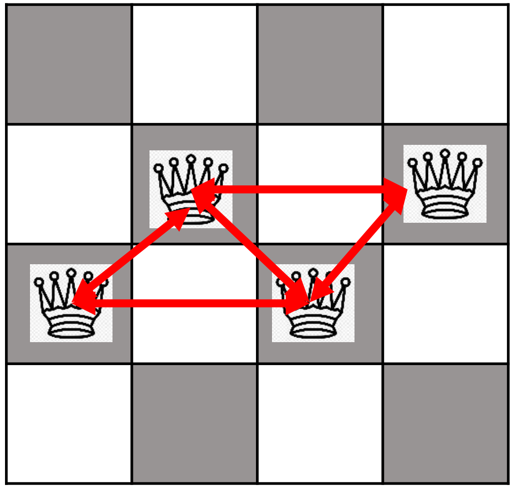
- Alg 1: Random (random walk)
  - Choose a random move; execute; repeat
  - Asymptotically complete (eventually, you visit every state)





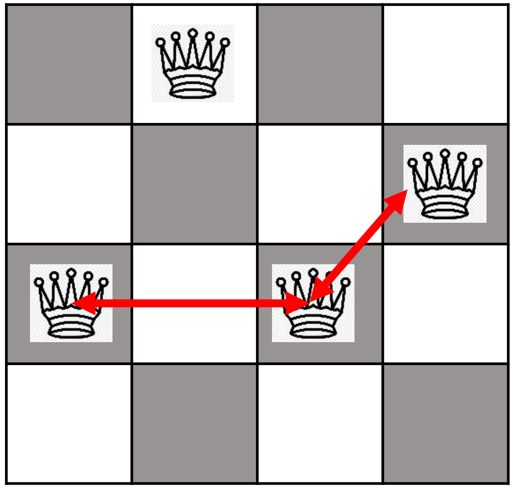
n-queens

- Currently 5 attacks, so give this score 5
- We want score 0



n-queens

- This neighbour has score 2



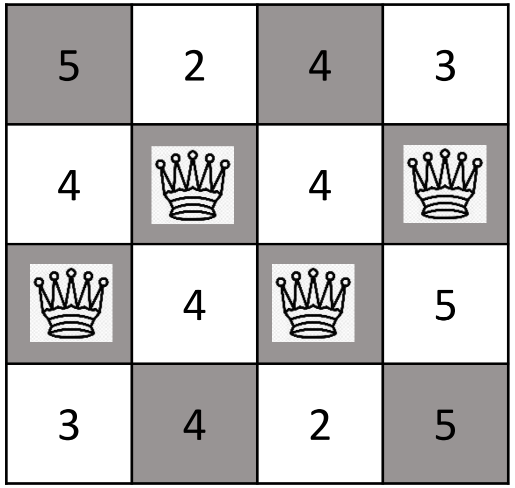
n-queens

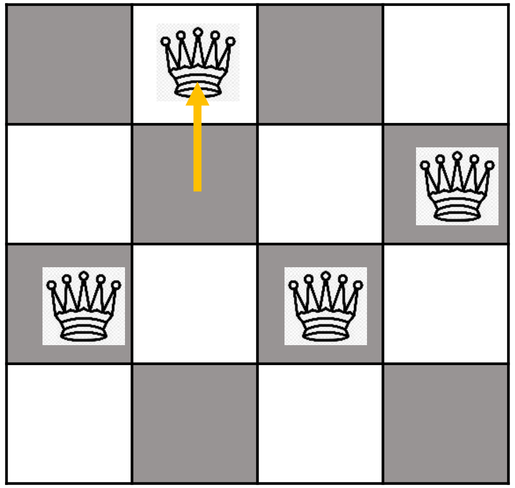
- We can label all squares with their score

5	2	4	3
4	♕	4	♕
♕	4	♕	5
3	4	2	5

n-queens

- Alg 2: Greedy (a.k.a Hill Climbing)
  - Choose the best move / one of the best moves
  - Requires us to evaluate the entire Neighbourhood



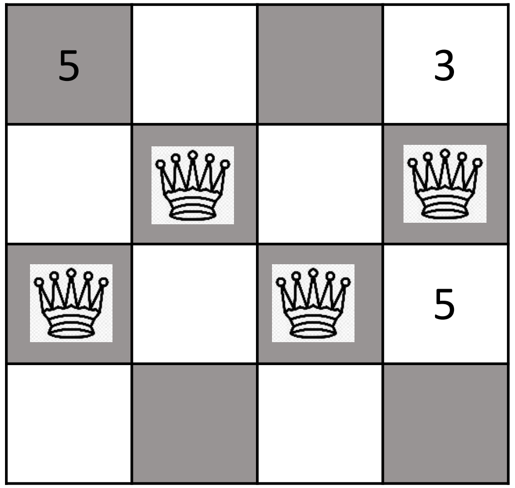


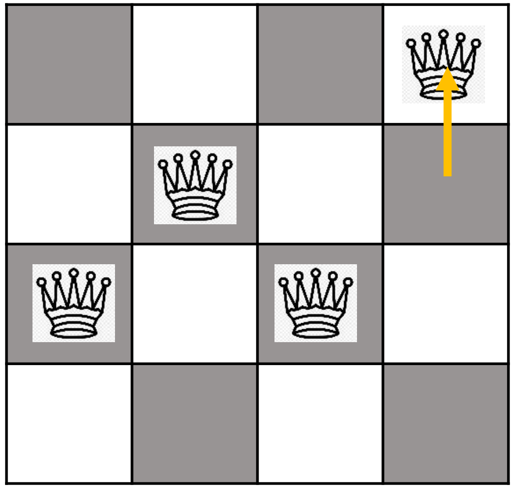
Hill Climbing has been compared to climbing Mt Everest … in thick fog … and while suffering from amnesia.

n-queens

- Alg 3: First found
  - Randomise neighbourhood evaluation
  - Make first improving move

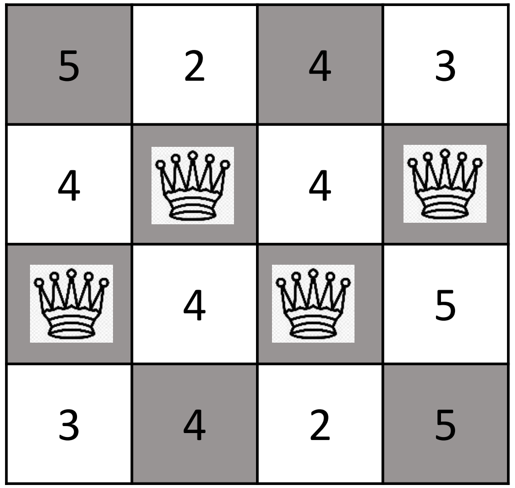
Computational evidence favours first-found over greedy

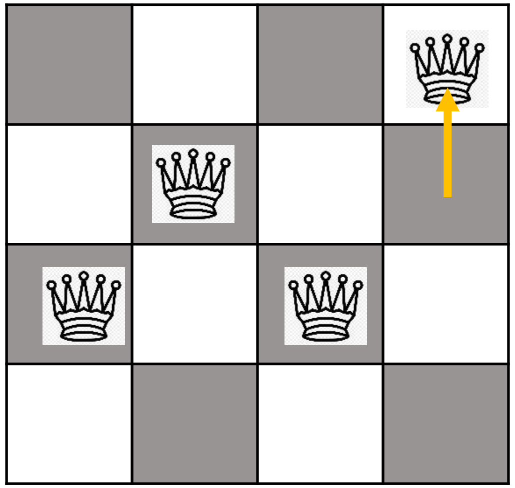




n-queens

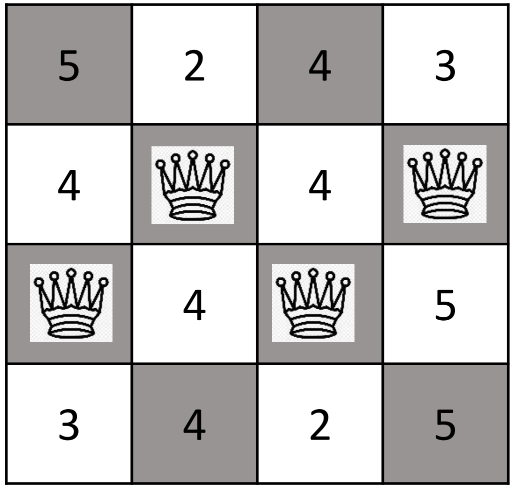
- Greedy search is incomplete
- Alg 4: Randomised Greedy
  - With probability *p* do greedy/first found move
  - With otherwise do a random move
    - Asymptotically complete

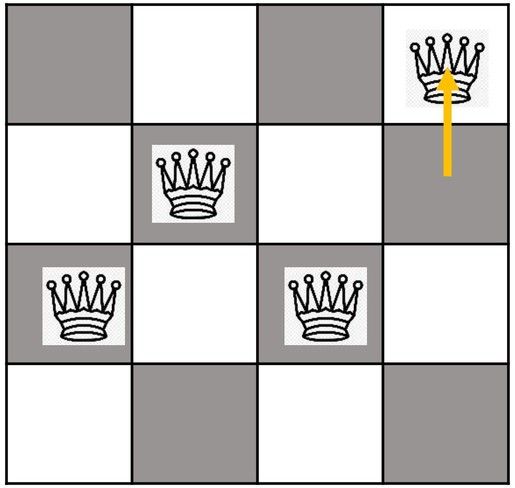




n-queens

- Alg 5: Biased
  - Choose a move with probability (inversely) proportional to score
  - (Twice as likely to choose a ‘2’ move than a ‘4’)

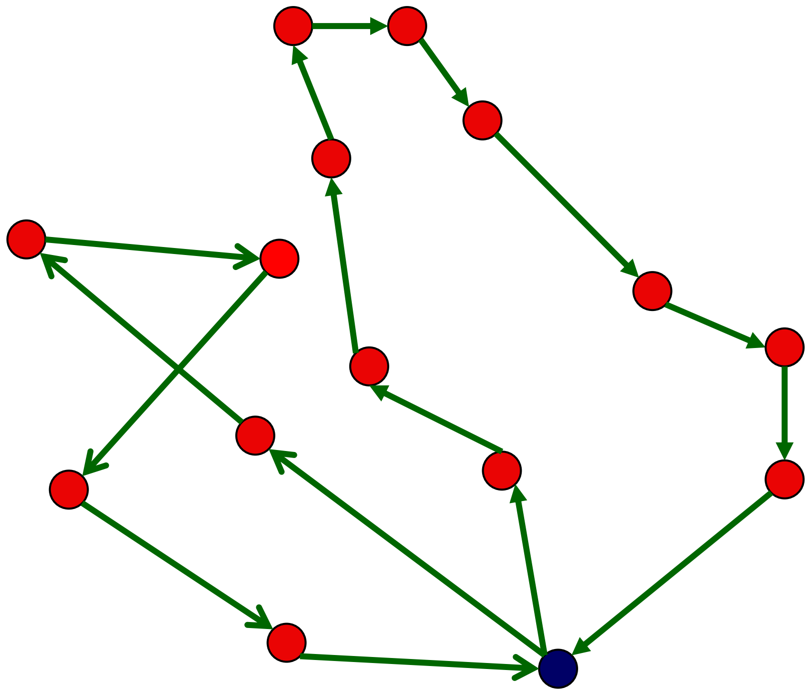




VRP

Local Search in VRP

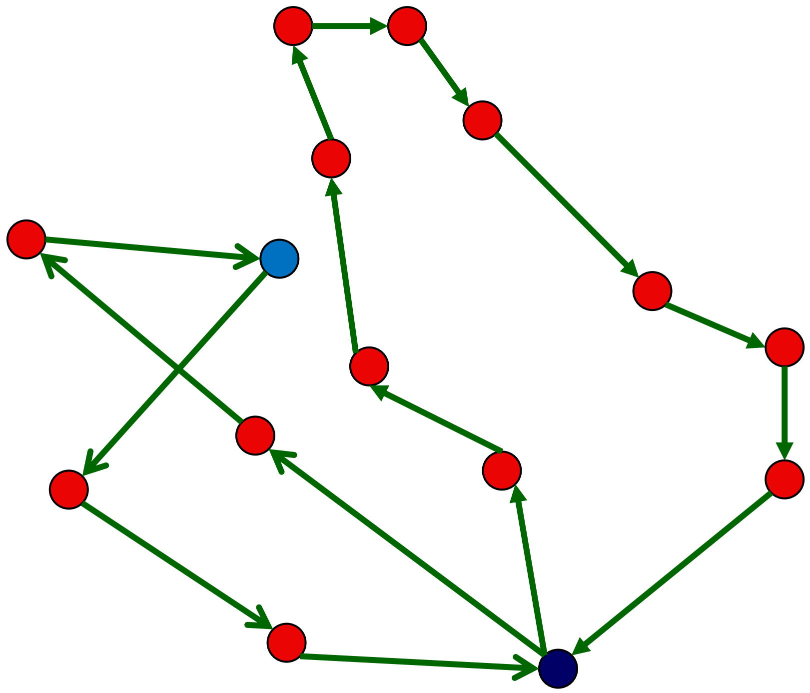
- More complex operators



VRP

Local Search in VRP

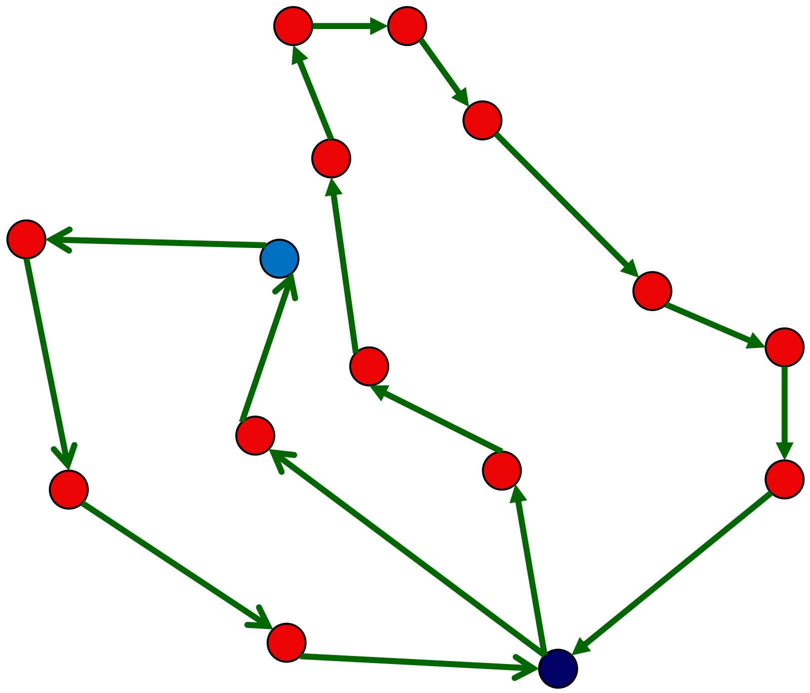
- More complex operators
  - 1-move



VRP

Local Search in VRP

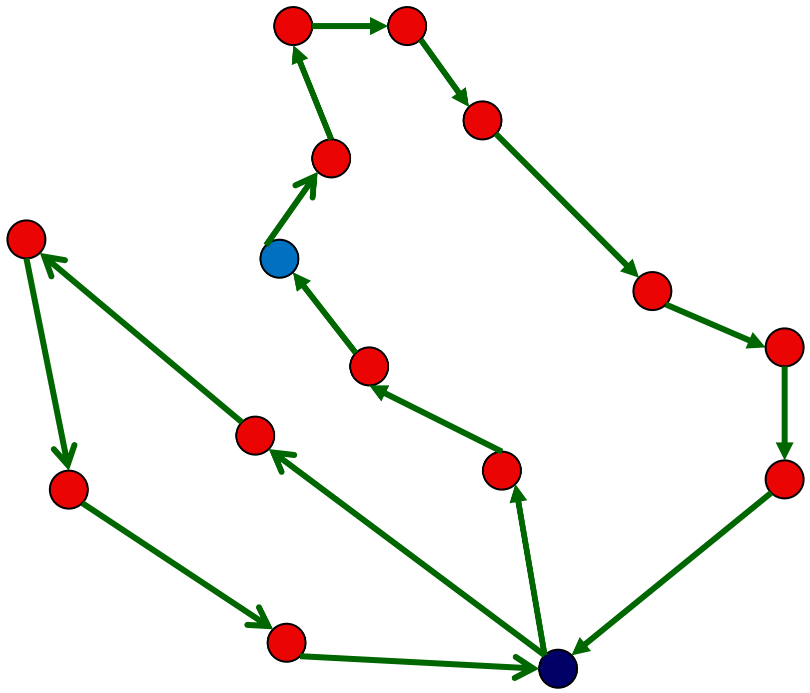
- More complex operators
  - 1-move



VRP

Local Search in VRP

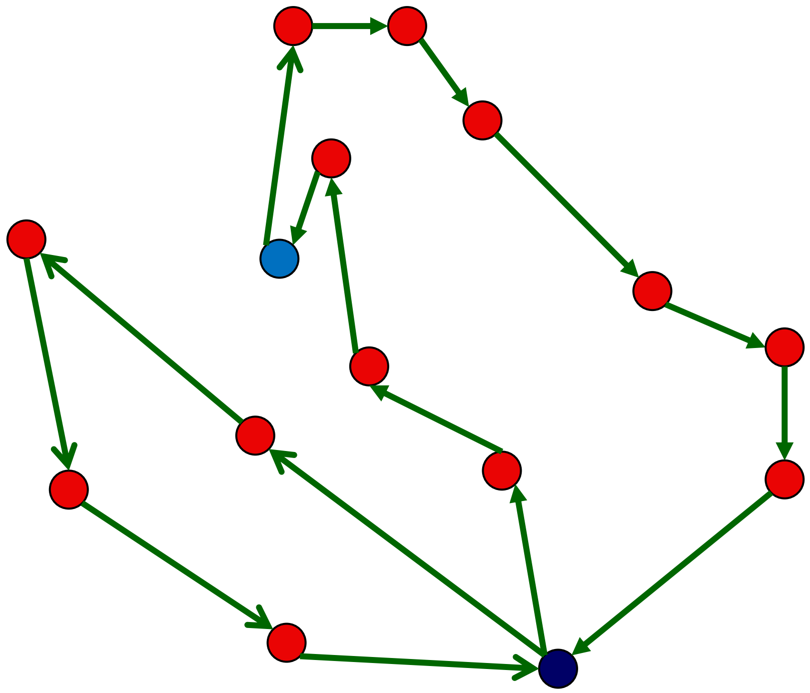
- More complex operators
  - 1-move



VRP

Local Search in VRP

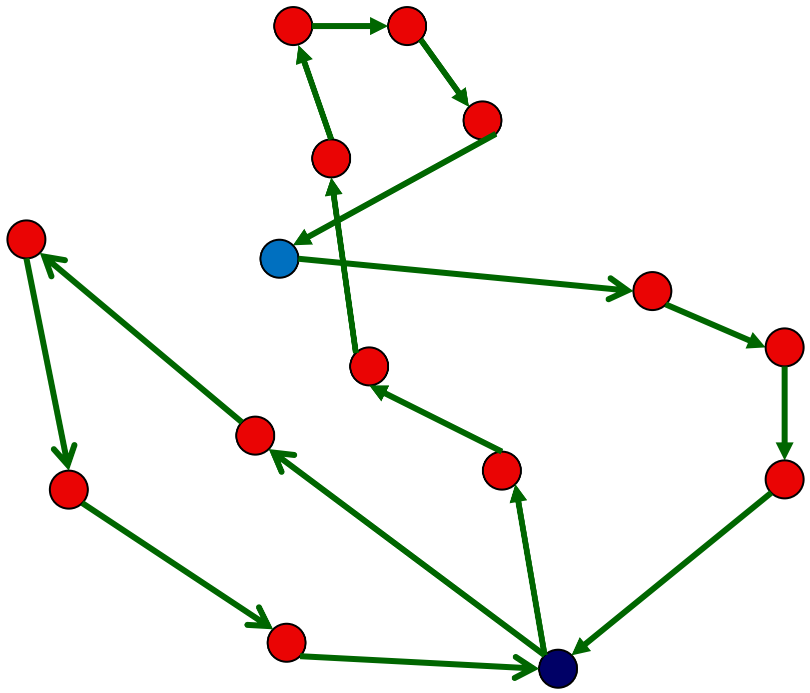
- More complex operators
    - 1-move



VRP

Local Search in VRP

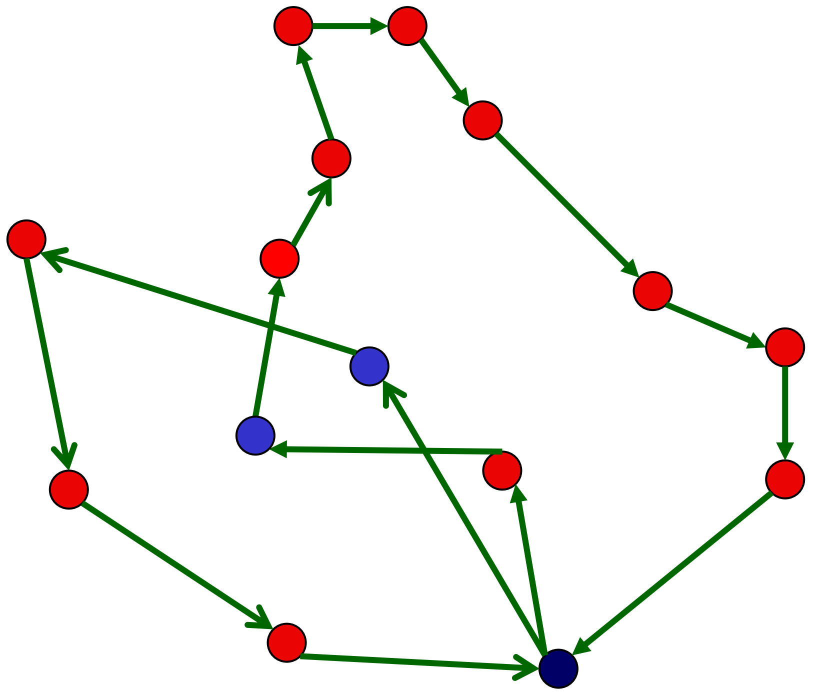
- More complex operators
  - 1-move



Local Search

Other Neighbourhoods for VRP:

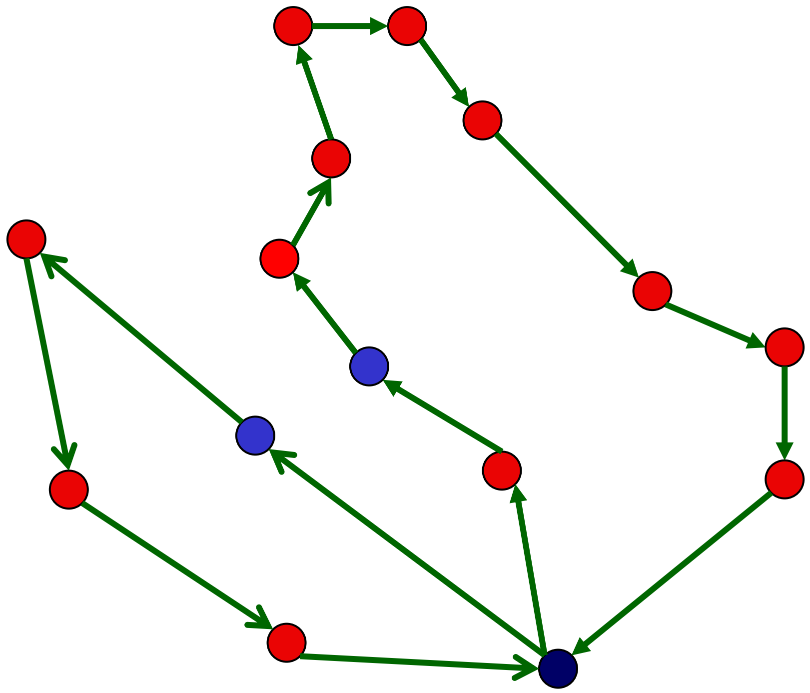
- Swap 1-1



Local Search

Other Neighbourhoods for VRP:

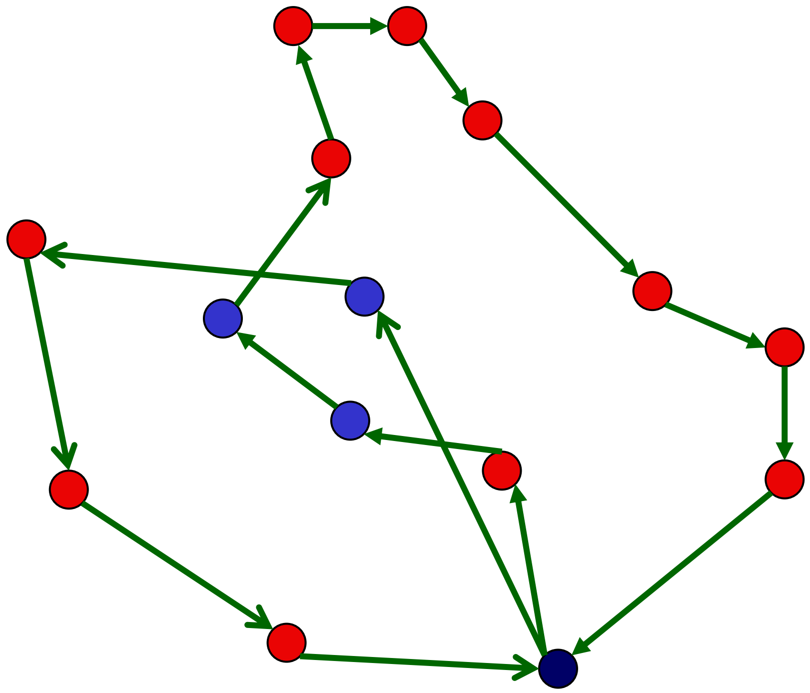
- Swap 1-1



Local Search

Other Neighbourhoods for VRP:

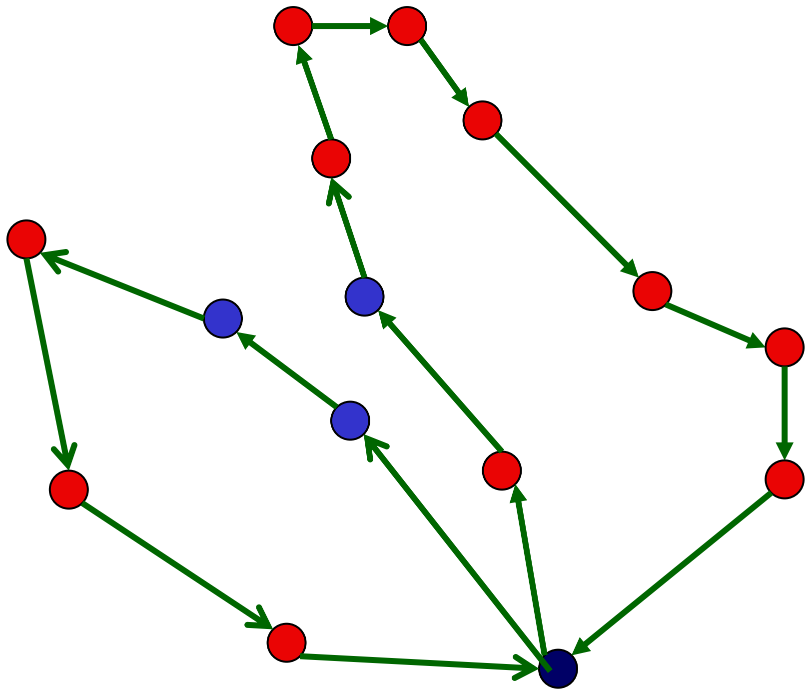
- Swap 2-1



Local Search

Other Neighbourhoods for VRP:

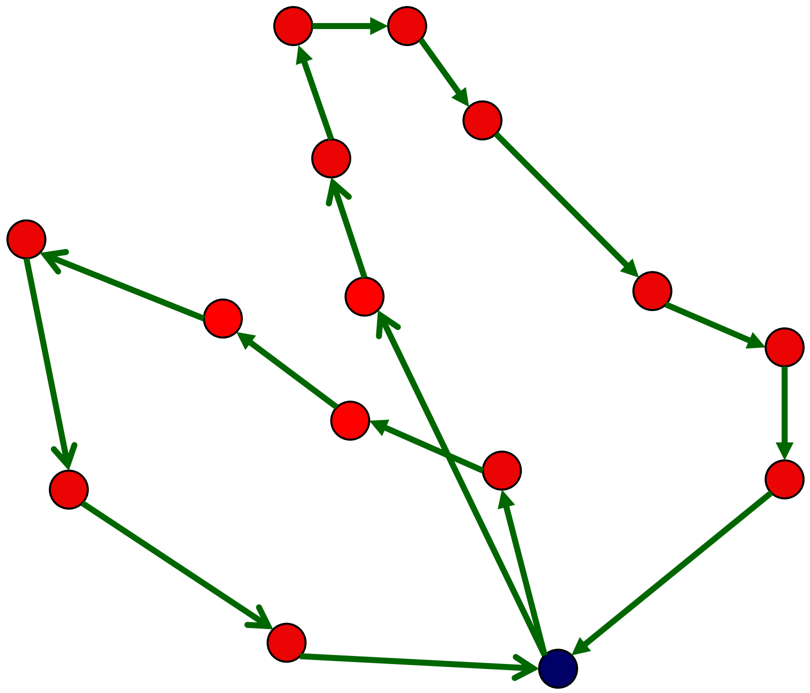
- Swap 2-1



Local Search

Other Neighbourhoods for VRP:

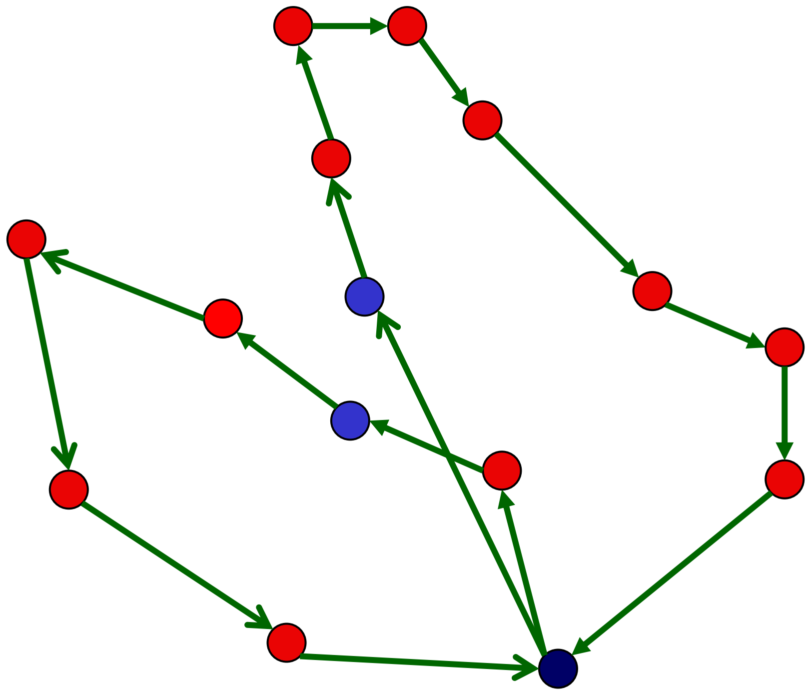
- Swap tails



Local Search

Other Neighbourhoods for VRP:

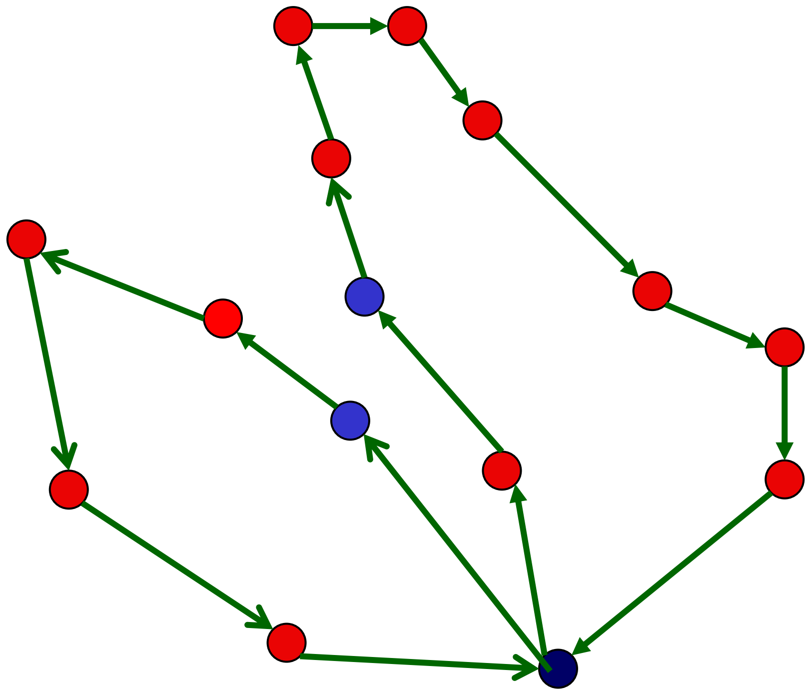
- Swap tails



Local Search

Other Neighbourhoods for VRP:

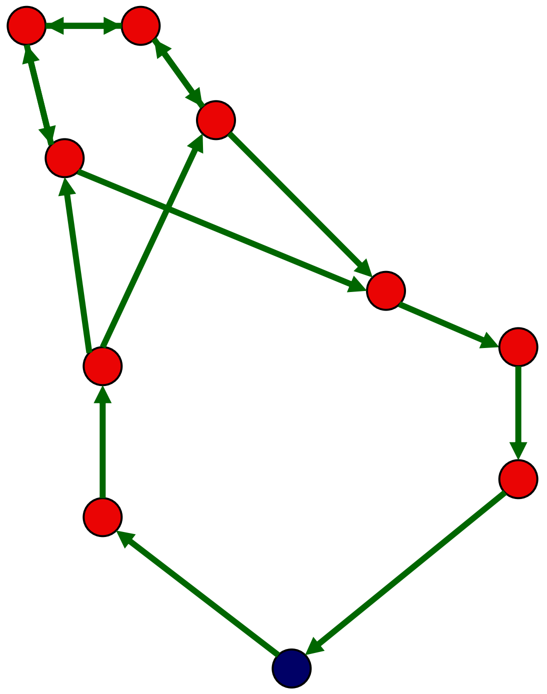
- Swap tails



VRP specific operators

2-opt (3-opt, 4-opt…)

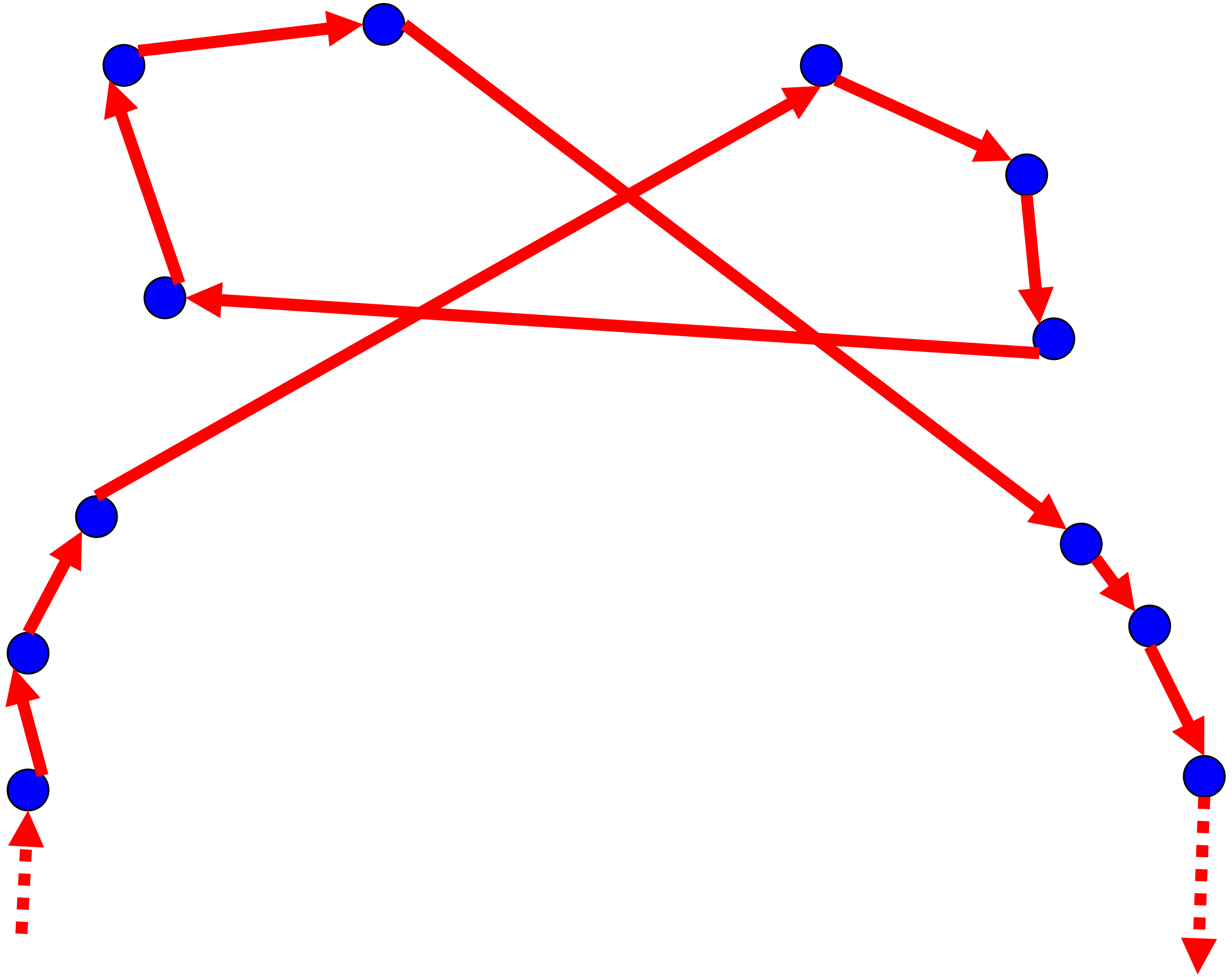
- Remove 2 arcs
- Replace with 2 others



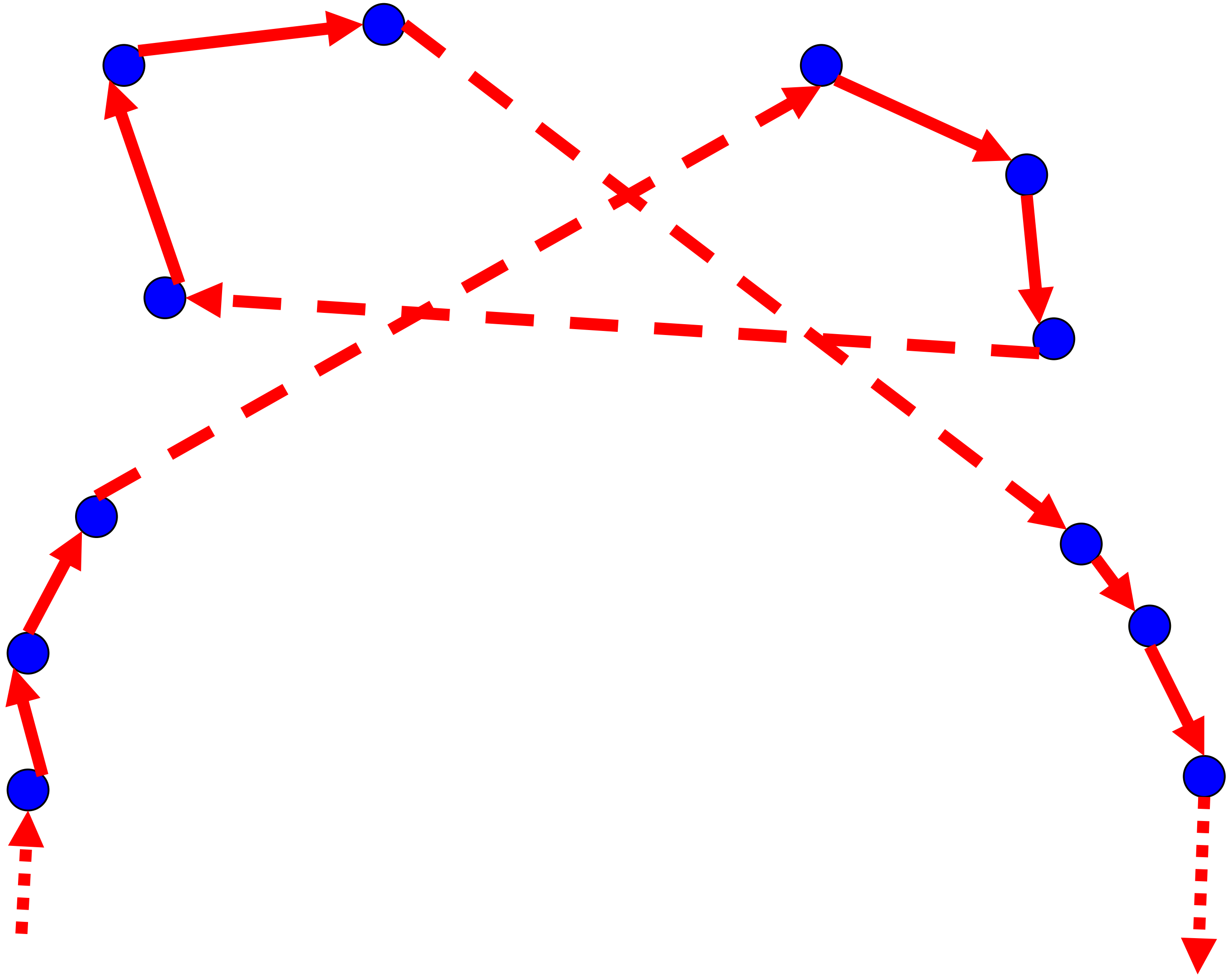
3-opt exchange

- Select three arcs
- Replace with three others
- 2 orientations possible

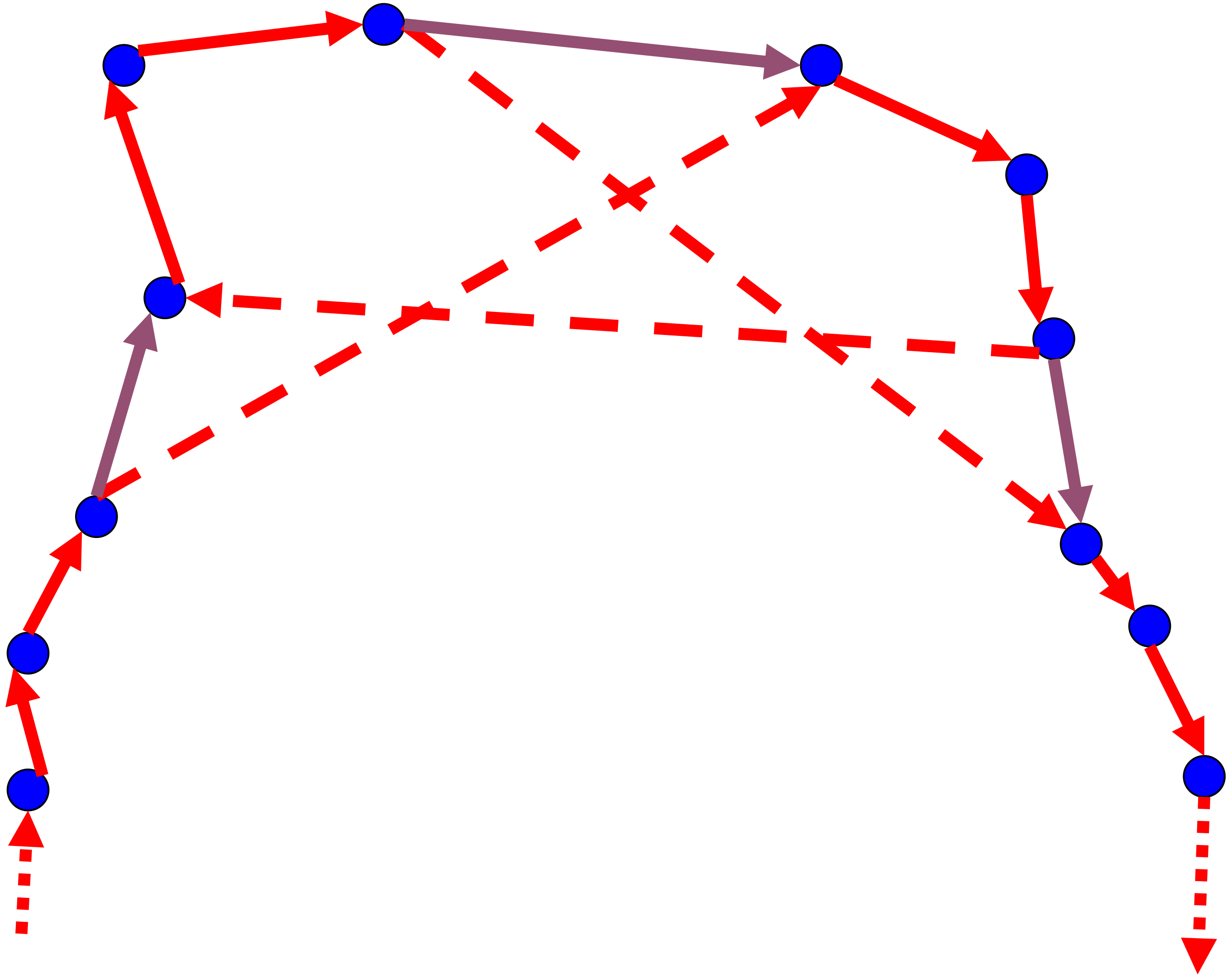
3-opt exchange



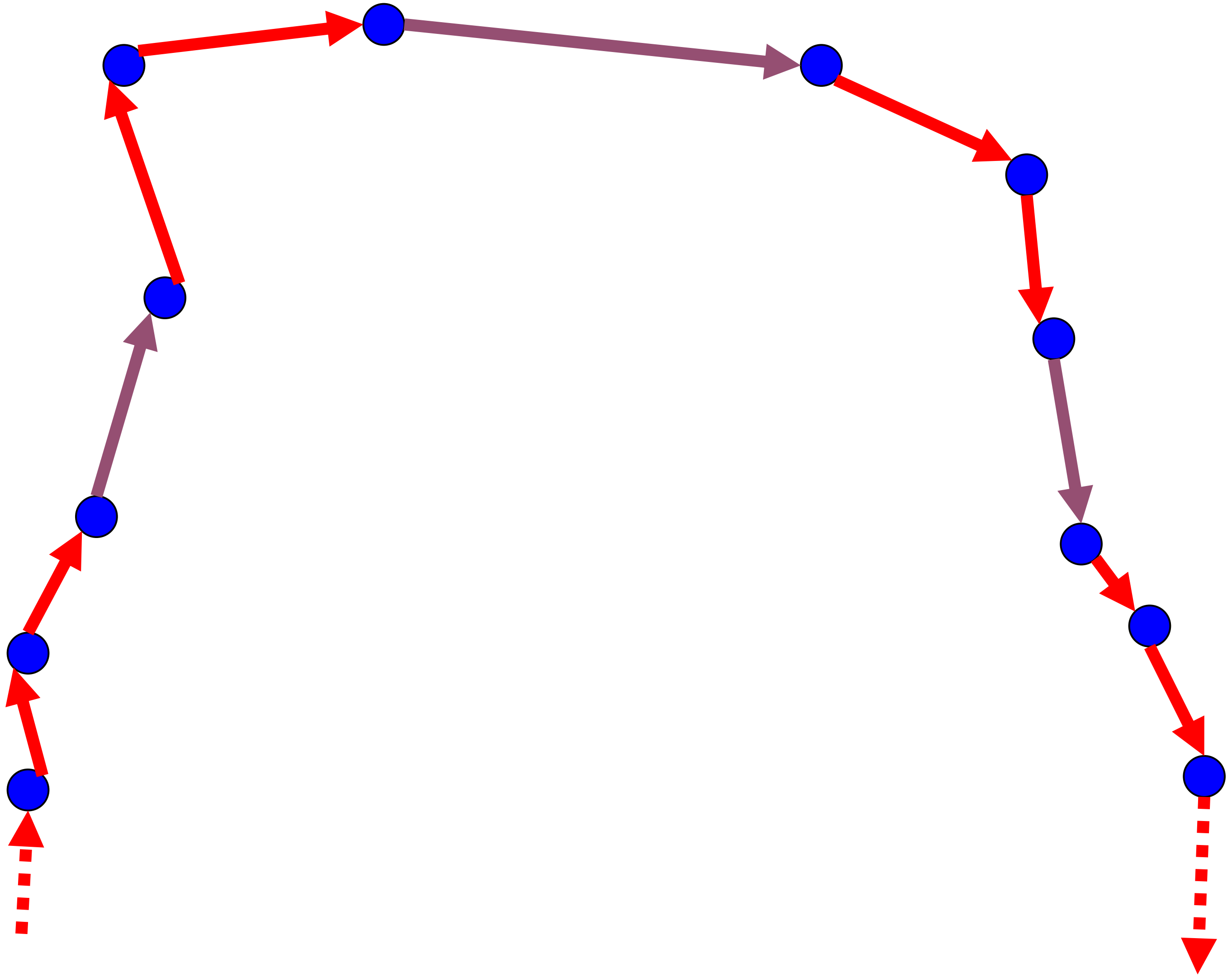
3-opt exchange



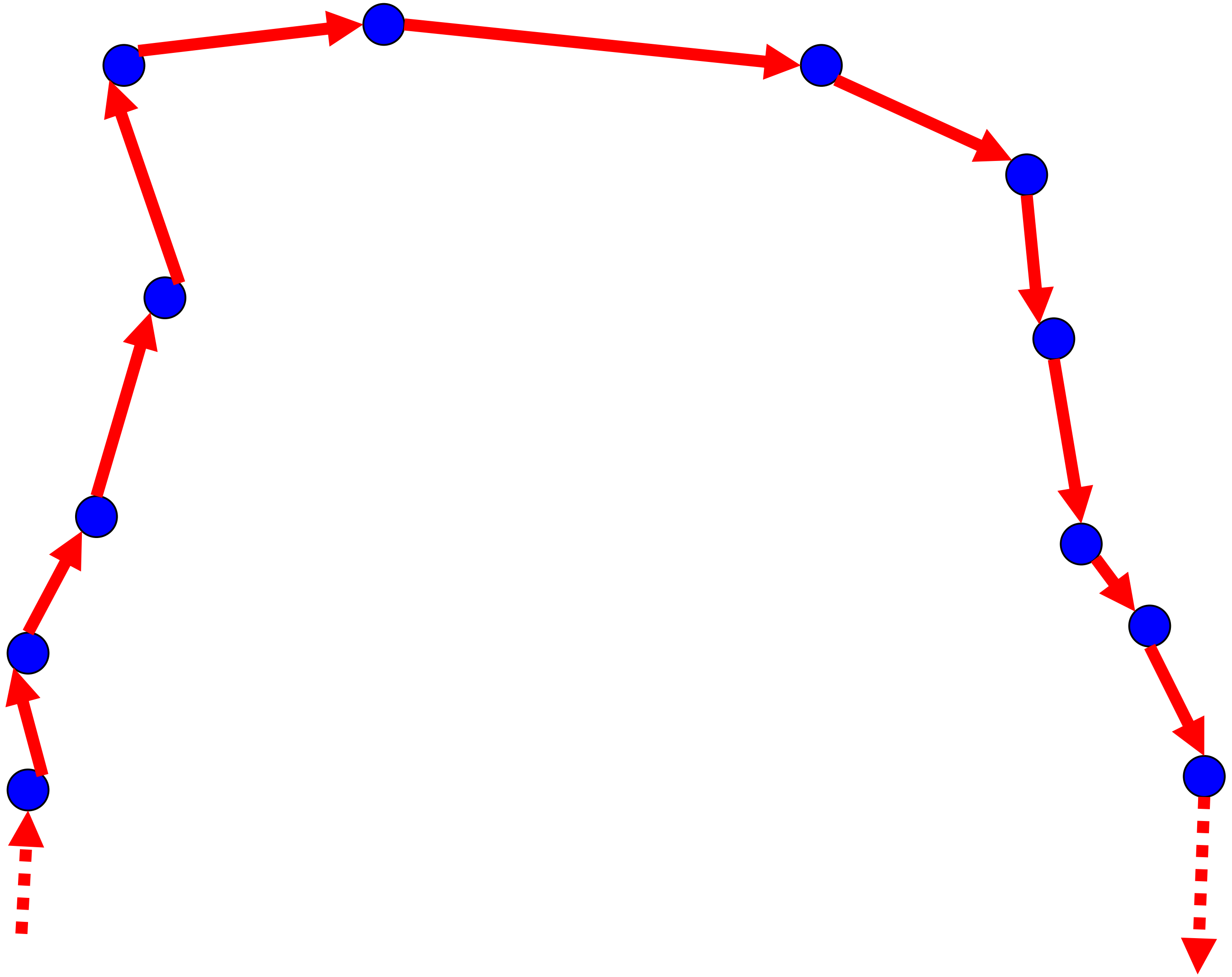
3-opt exchange



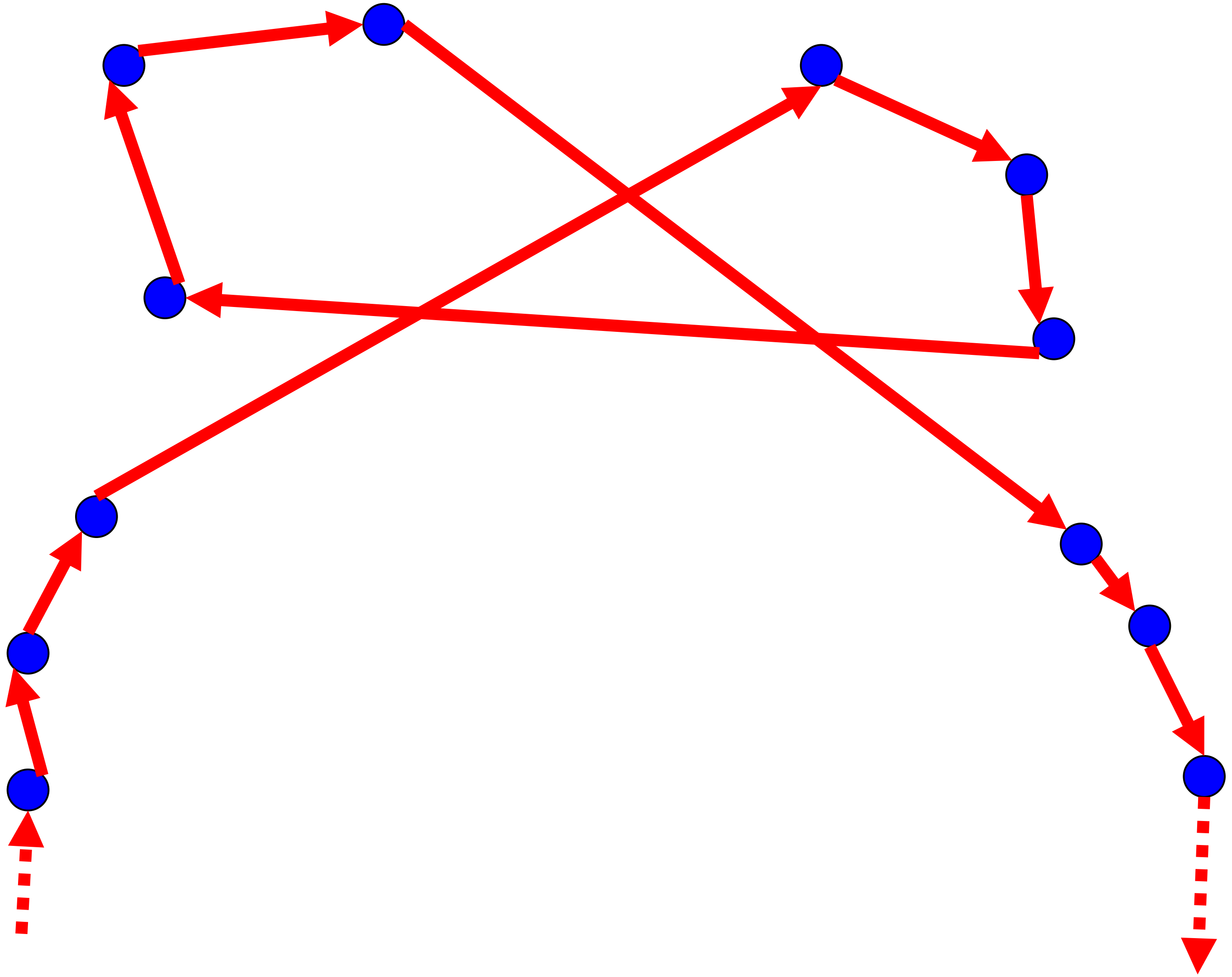
3-opt exchange



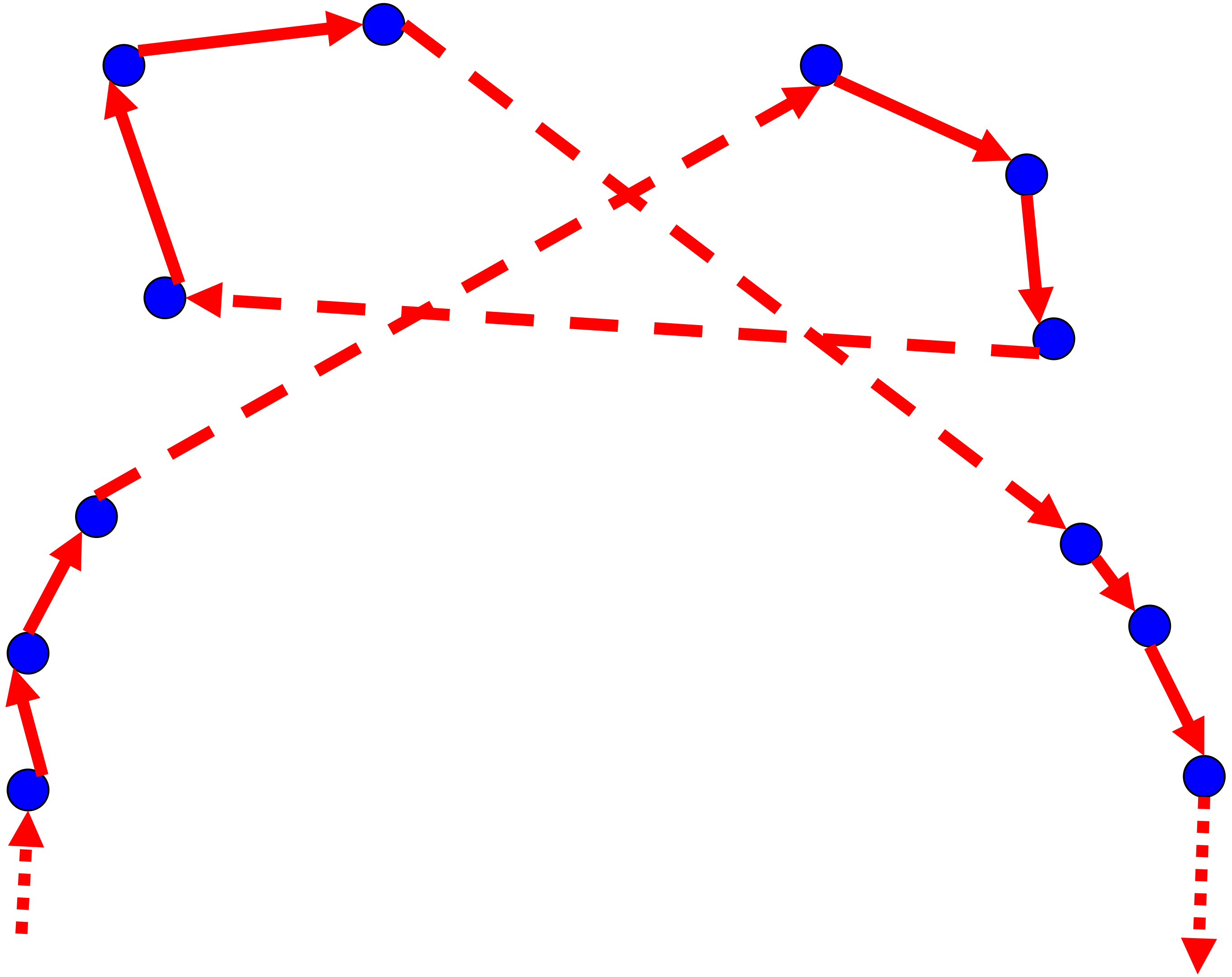
3-opt exchange



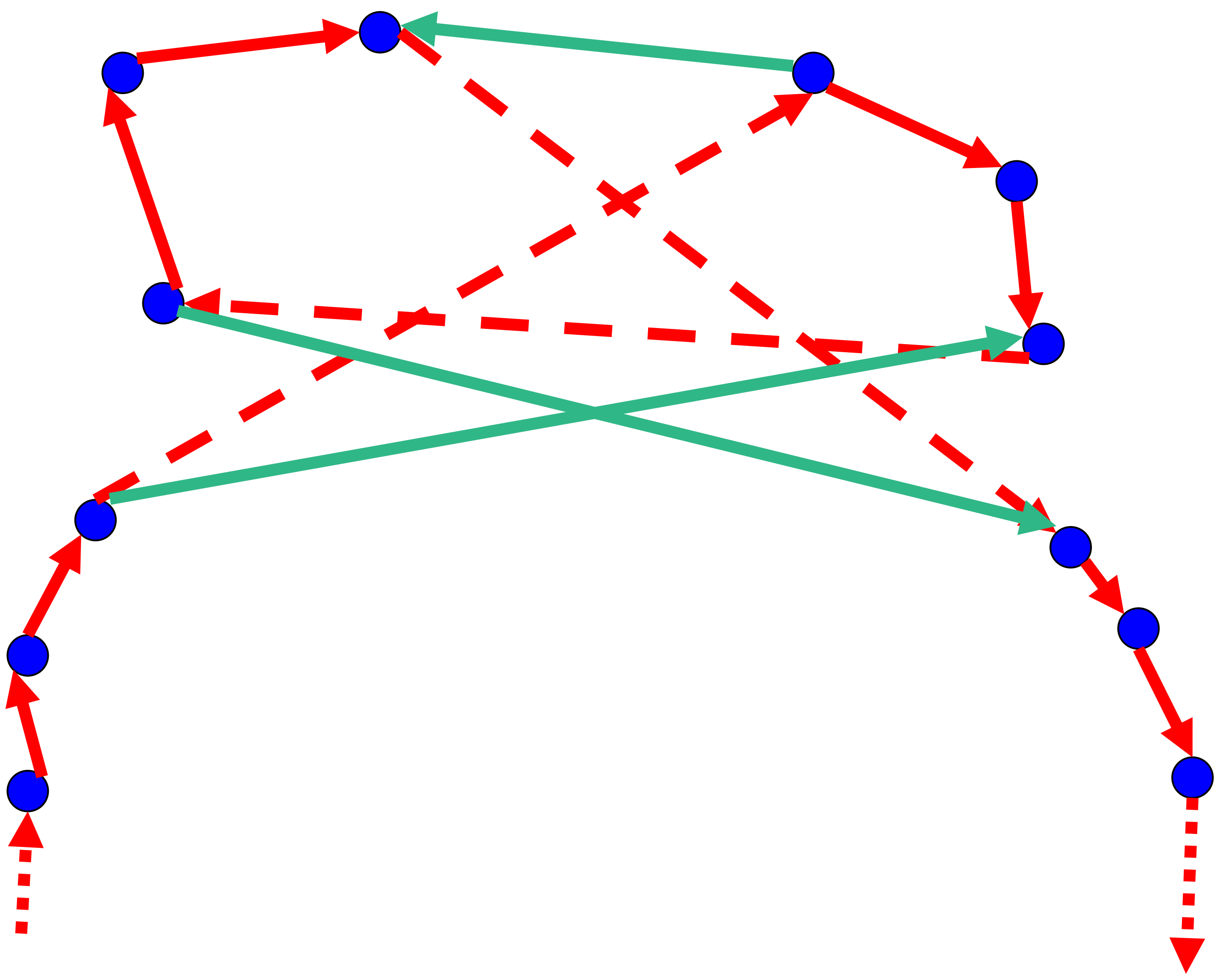
3-opt exchange



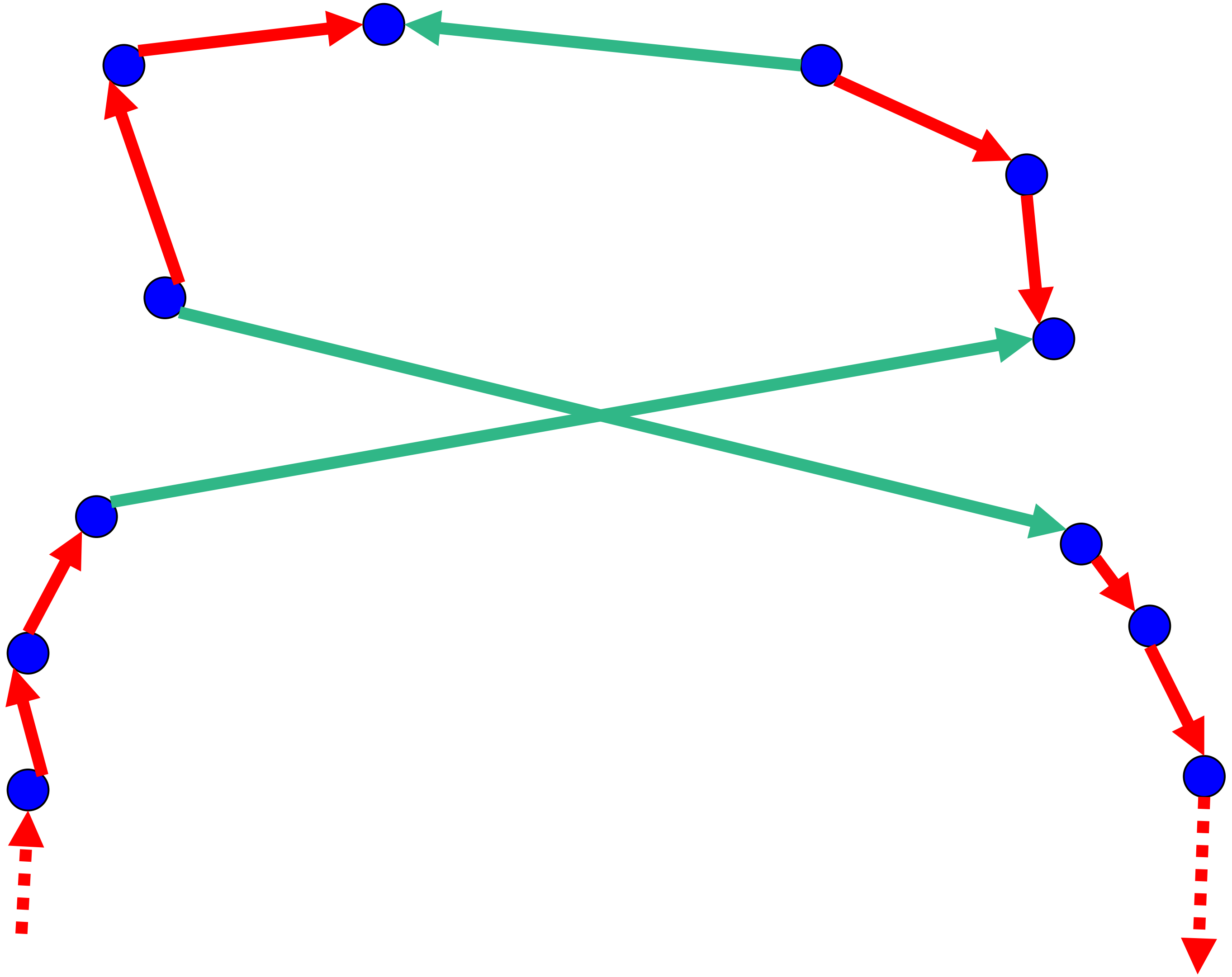
3-opt exchange



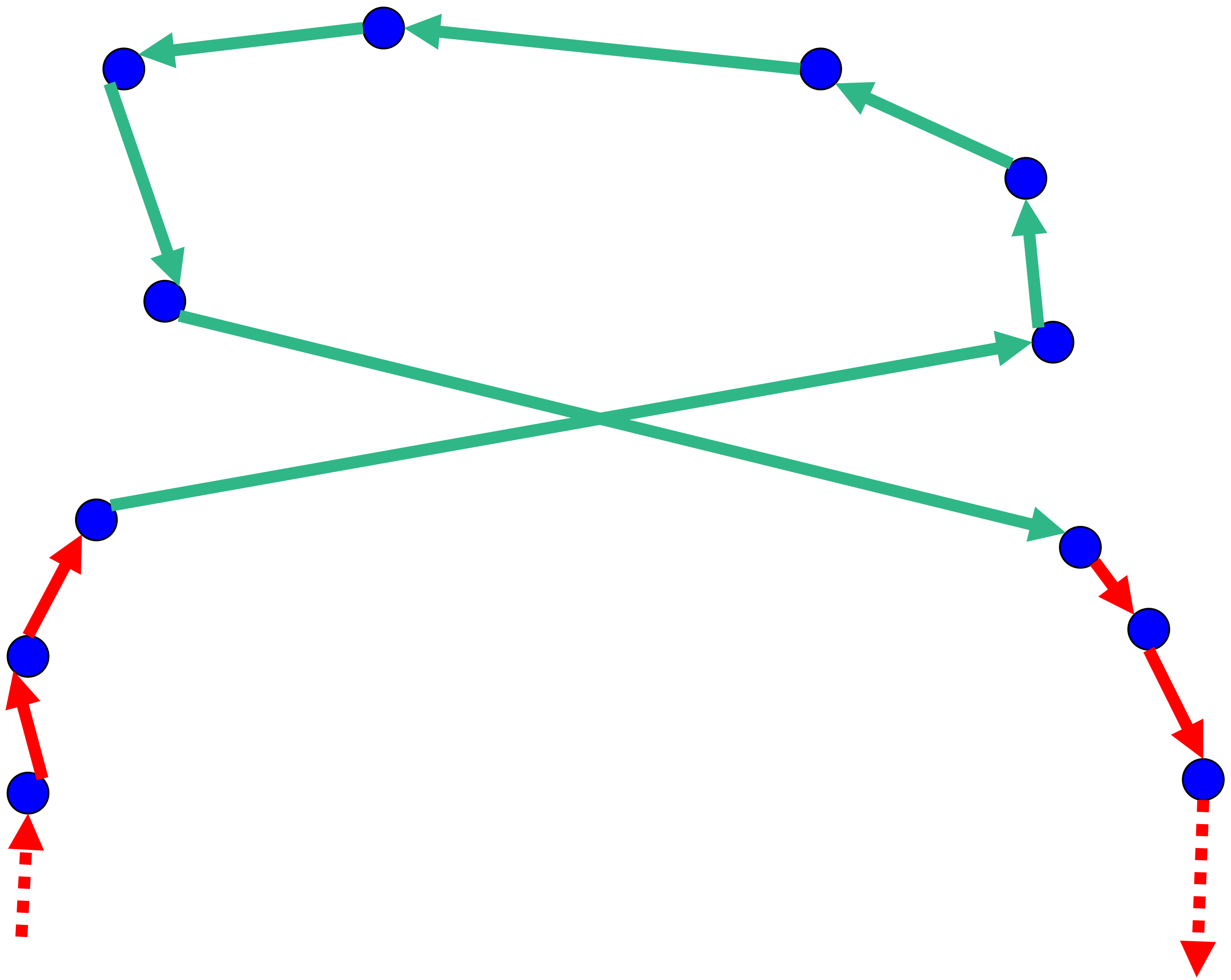
3-opt exchange



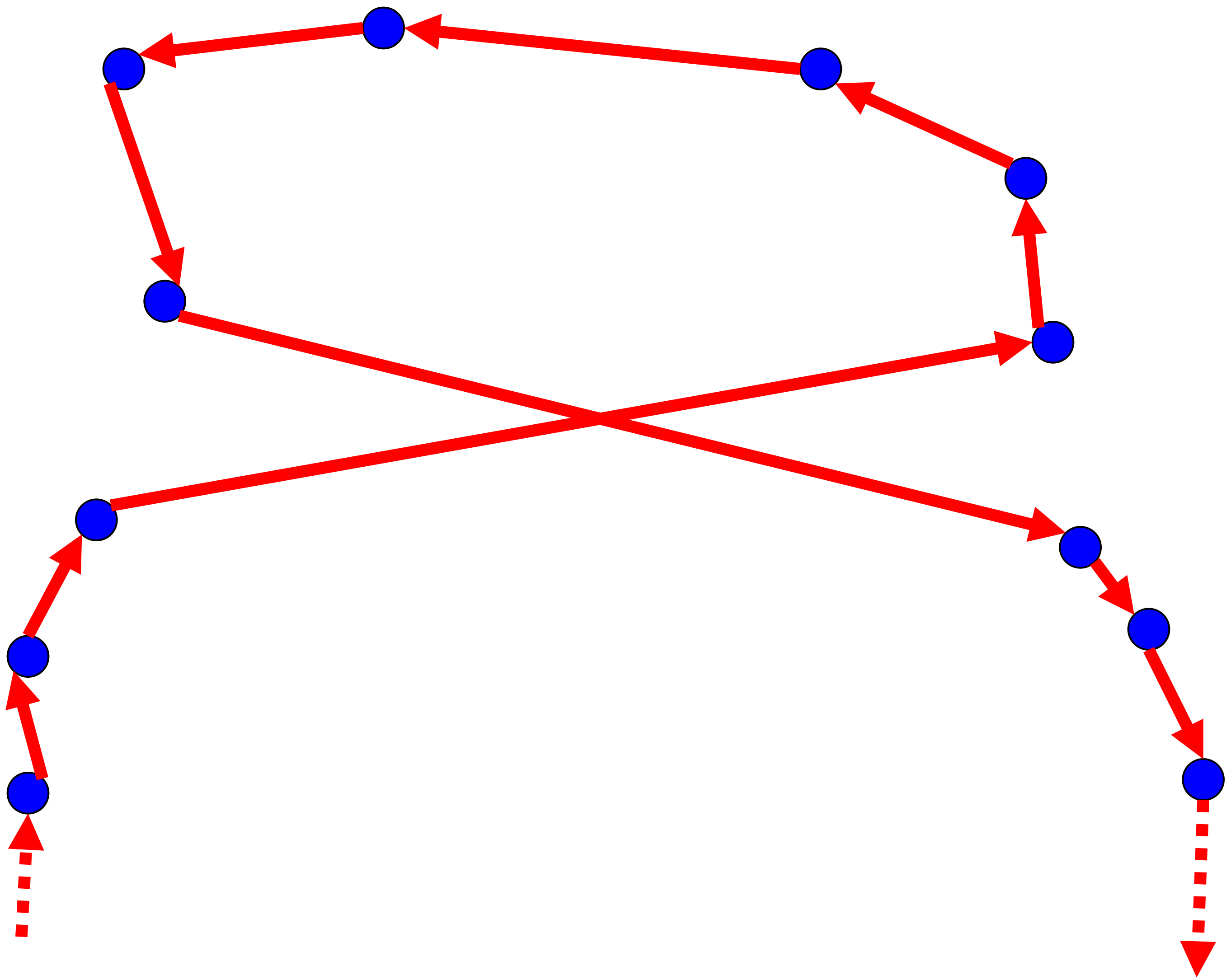
3-opt exchange



3-opt exchange



3-opt exchange



Other problems

Neighbourhoods for other problems

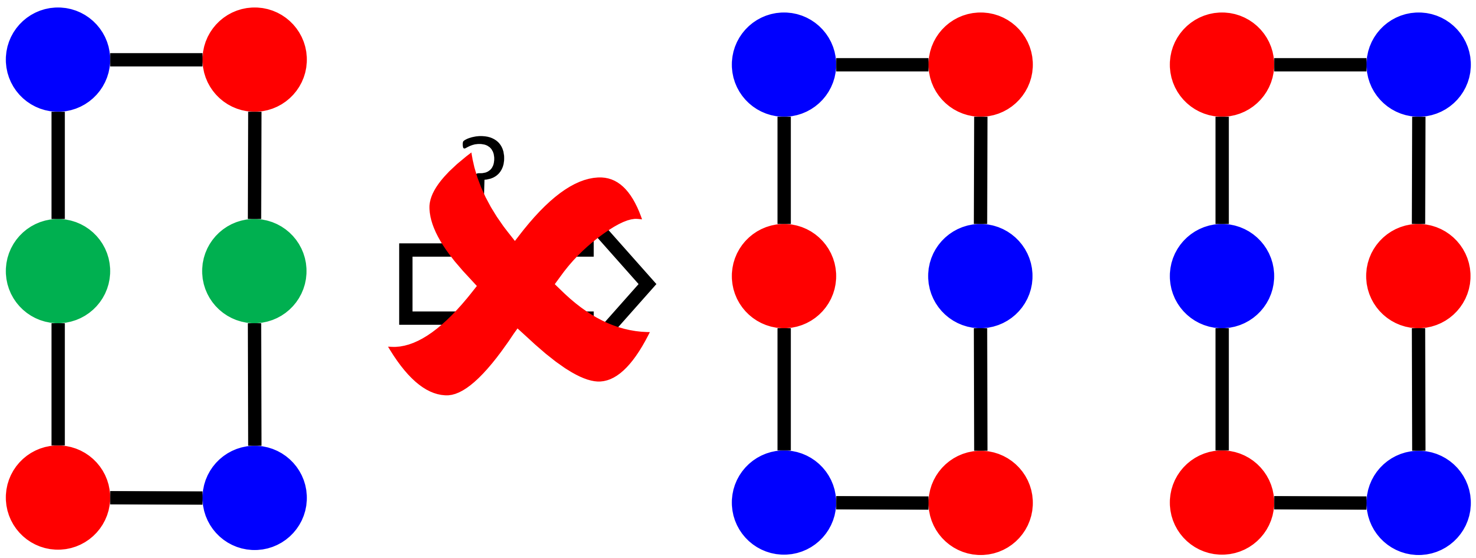
- Knapsack
  - Swap 2 items
  - Swap 1 item with multiple items of equal size

- Scheduling
  - Swap jobs between machines
  - Swap order of jobs

Neighbourhood

- E.g. Map Colouring (*k*-colouring)
  - Colour a map (graph) so that no two adjacent countries (nodes) are the same colour
  - Either:
    - Use at most *k* colours
    - Minimize number of colours

Map Colouring



Starting sol

Two optimal solutions

Define neighbourhood as:

Change the colour of at most one vertex

Make k-colour constraint soft fixes this issue

Neighbourhood

- Hard constraints create impenetrable mountain ranges in solution space
- Soft constraints allow passes through the mountains

Strongly connected:

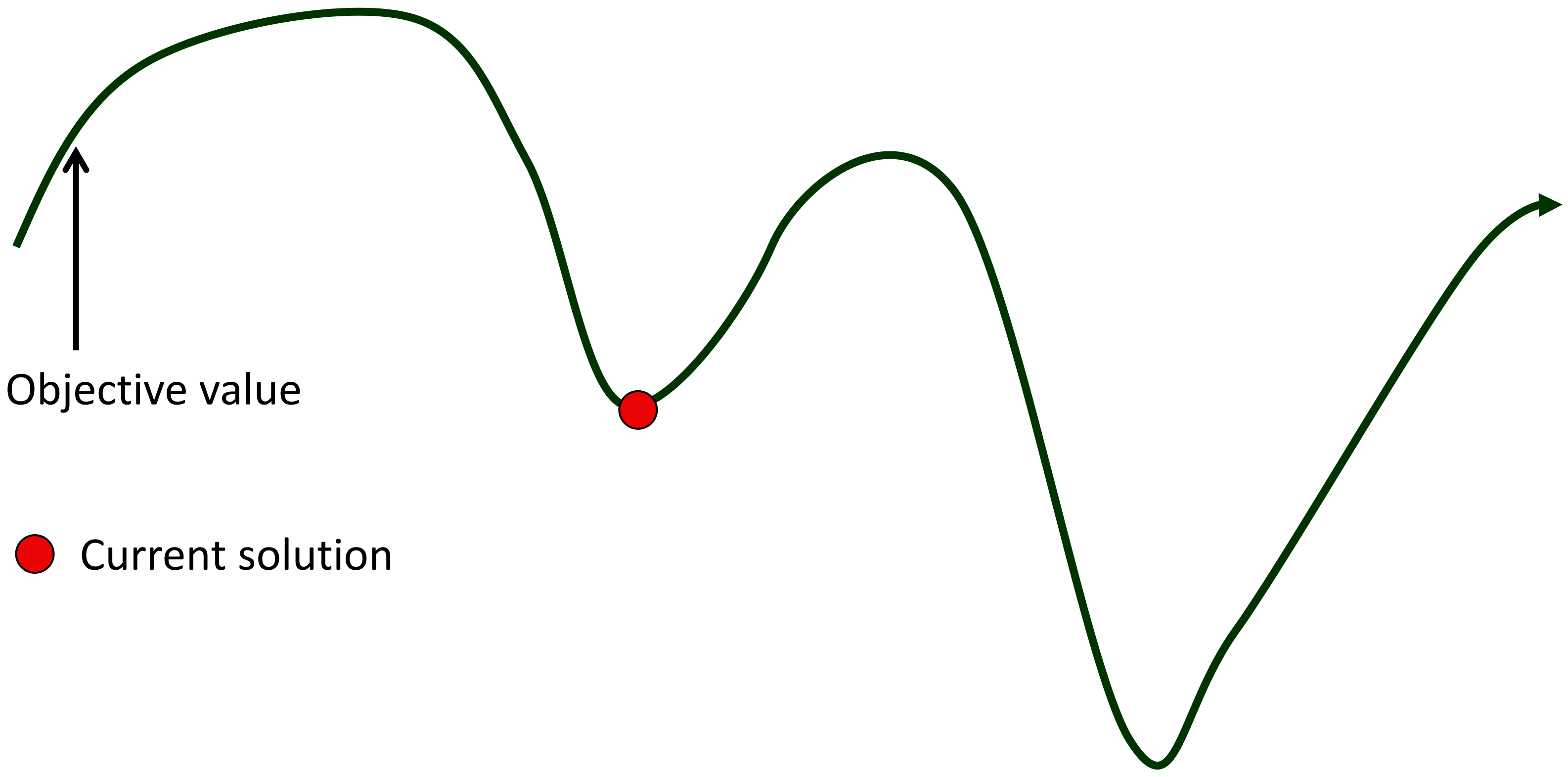
- Any solution can be reached from any other (e.g. 2-opt)

Weakly optimally connected

- The optimum can be reached from any starting solution

Problems with Local Search I

Local minima

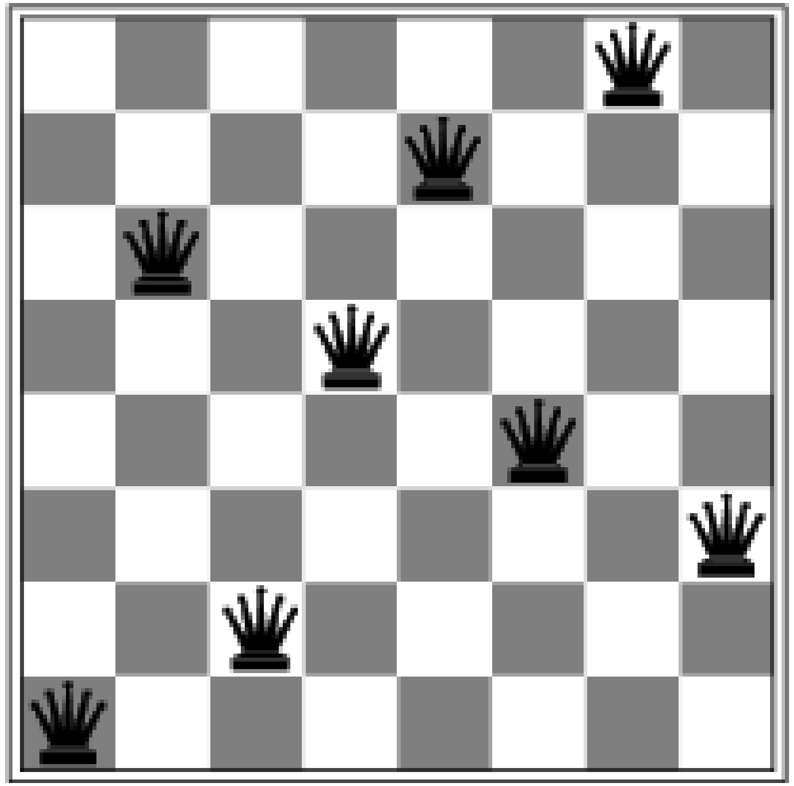


8 Queens

Hill-climbing search for 8-Queens

- Randomly generated 8-queens starting states…
- 14% the time it solves the problem
- 86% of the time it get stuck at a local minimum

- However…
  - Takes only 4 steps on average when it succeeds
  - And 3 on average when it gets stuck
  - (for a state space with 8^8 =~17 million states)

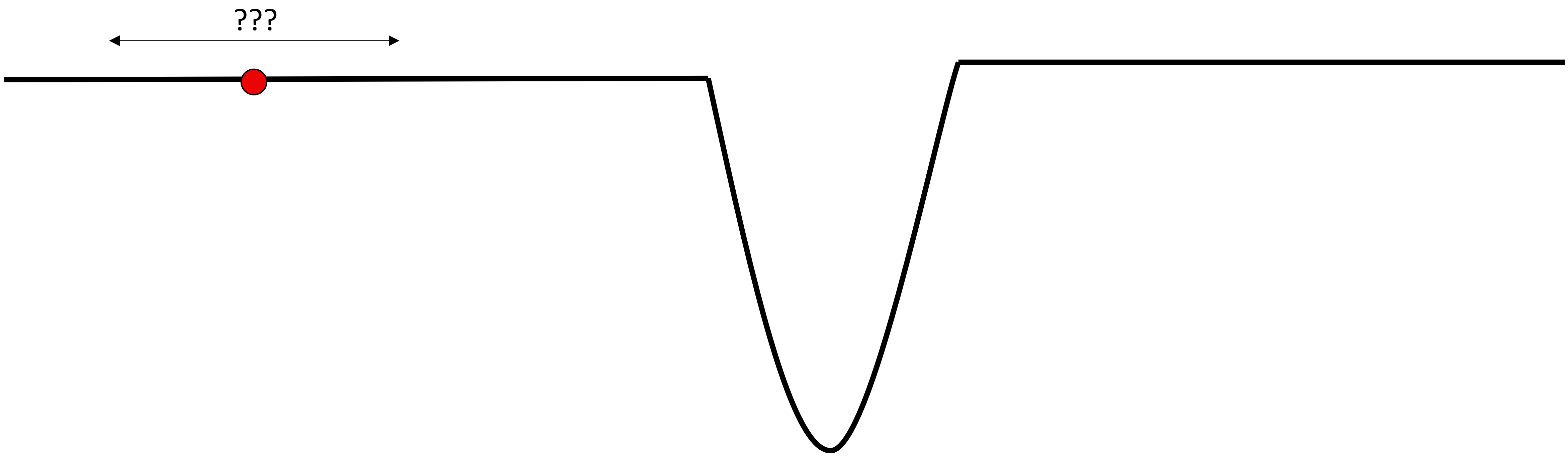


Escaping local minima

- Solution 1: Random Restarts
  - Whenever you hit a local minimum, generate a new {random / randomised} starting solution
  - This has proved very powerful on some problems
  - Particularly powerful is combination of randomised greedy + random restarts

Problems with Local Search II

- Plateaux



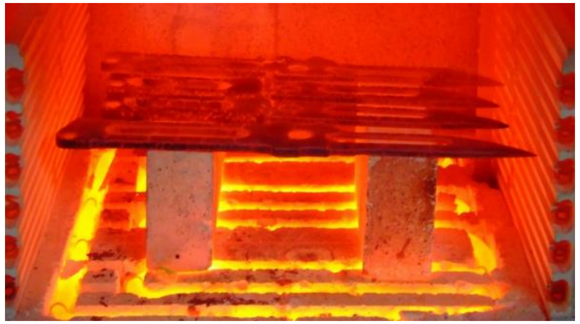
Escaping Plateaux (Shoulders)

- If no downhill (uphill) moves, allow sideways moves in hope that algorithm can escape
  - Need to place a limit on the possible number of sideways moves to avoid infinite loops
- For 8-queens
  - Now allow sideways moves with a limit of 100
  - Raises percentage of problem instances solved from 14% to 94%
- However….
  - 21 steps for every successful solution
  - 64 for each failure

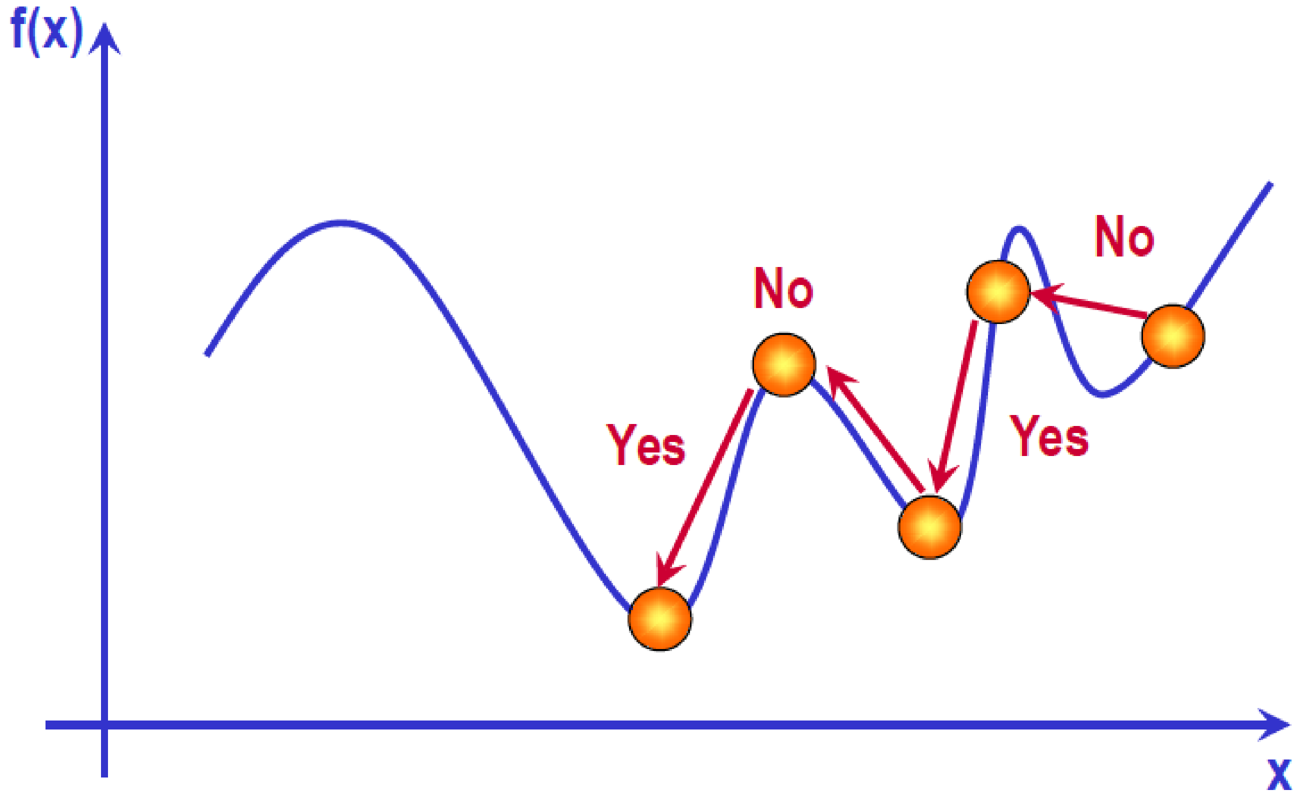
Escaping local minima II

Solution 2: Simulated Annealing

- Based on manner in which crystals are formed
  - At high temperatures, molecules move freely
  - At low temperatures, molecules are "stuck"
- If cooling is slow
  - A low energy, organized crystal lattice formed
- Minimise energy in crystal ⇔ Minimise objective

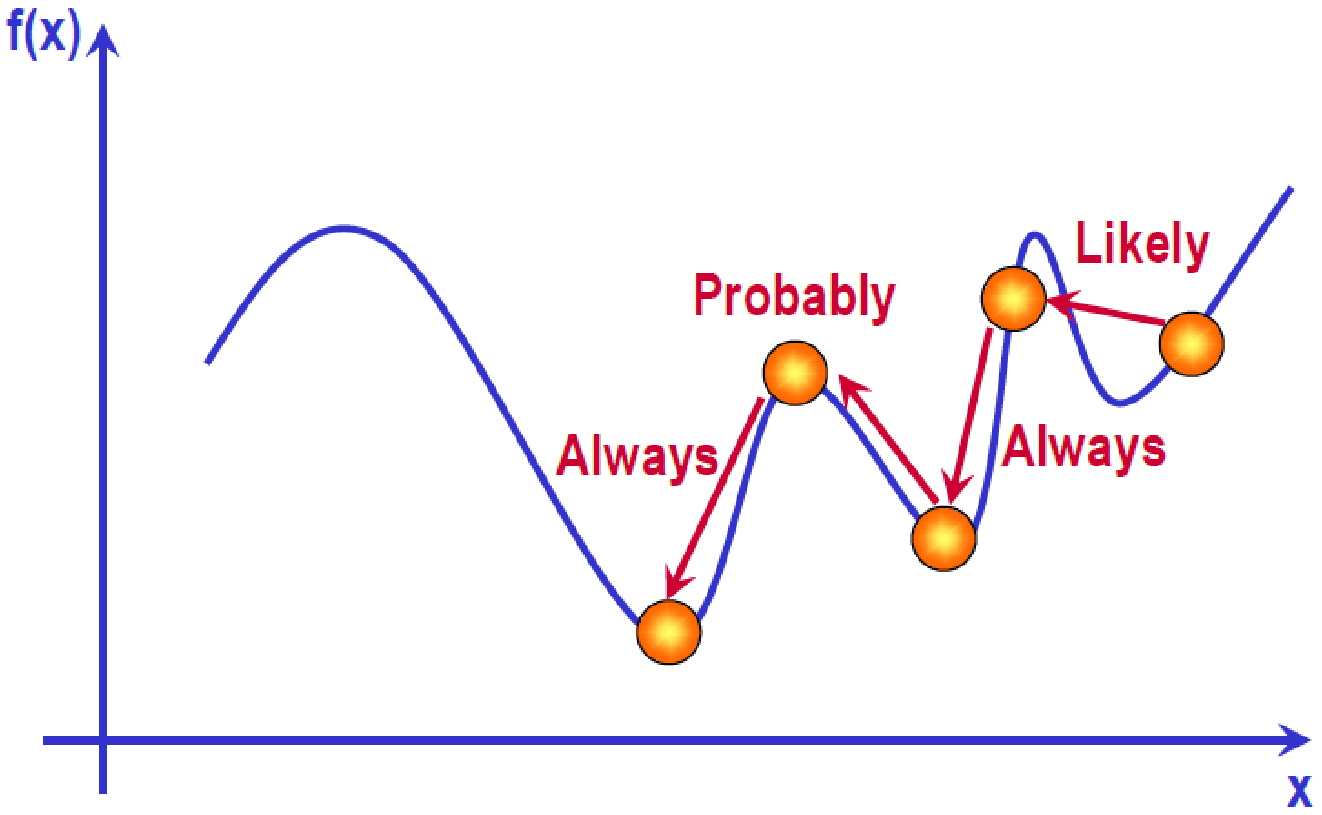


Simulated Annealing



Greedy local search reduces objective at every iteration

Simulated Annealing



Simulated Annealing accepts/rejects solution candidate based on probability

Simulated Annealing

If candidate solution reduces the objective,

- always accept the change (c.f. first found)

If candidate solution has higher objective,

- system parameter *T* *("Temperature")* controls probability of acceptance

$$P(accept\, increase\, \Delta) = e^{-\Delta/T}$$

- *T* reduces as method proceeds
- As *T* → 0, only improving moves accepted

Simulated Annealing

- Nice theoretical result:

  As number of iters $\rightarrow \infty$, probability of finding the optimal solution $\rightarrow 1$

- Experimental confirmation: On many problem, long runs yield good results
- Weak optimal connection required

Simulated Annealing

Initial T

- Set equal to max [acceptable] Δ

Updating T

- Geometric update: $T_{k+1} = \alpha\ T_k$
- α usually in [0.9, 0.999]

Don't want too many changes at one temperature (too hot):

```
If (numChangesThisT > maxChangesThisT)
    updateT()
```

Simulated Annealing

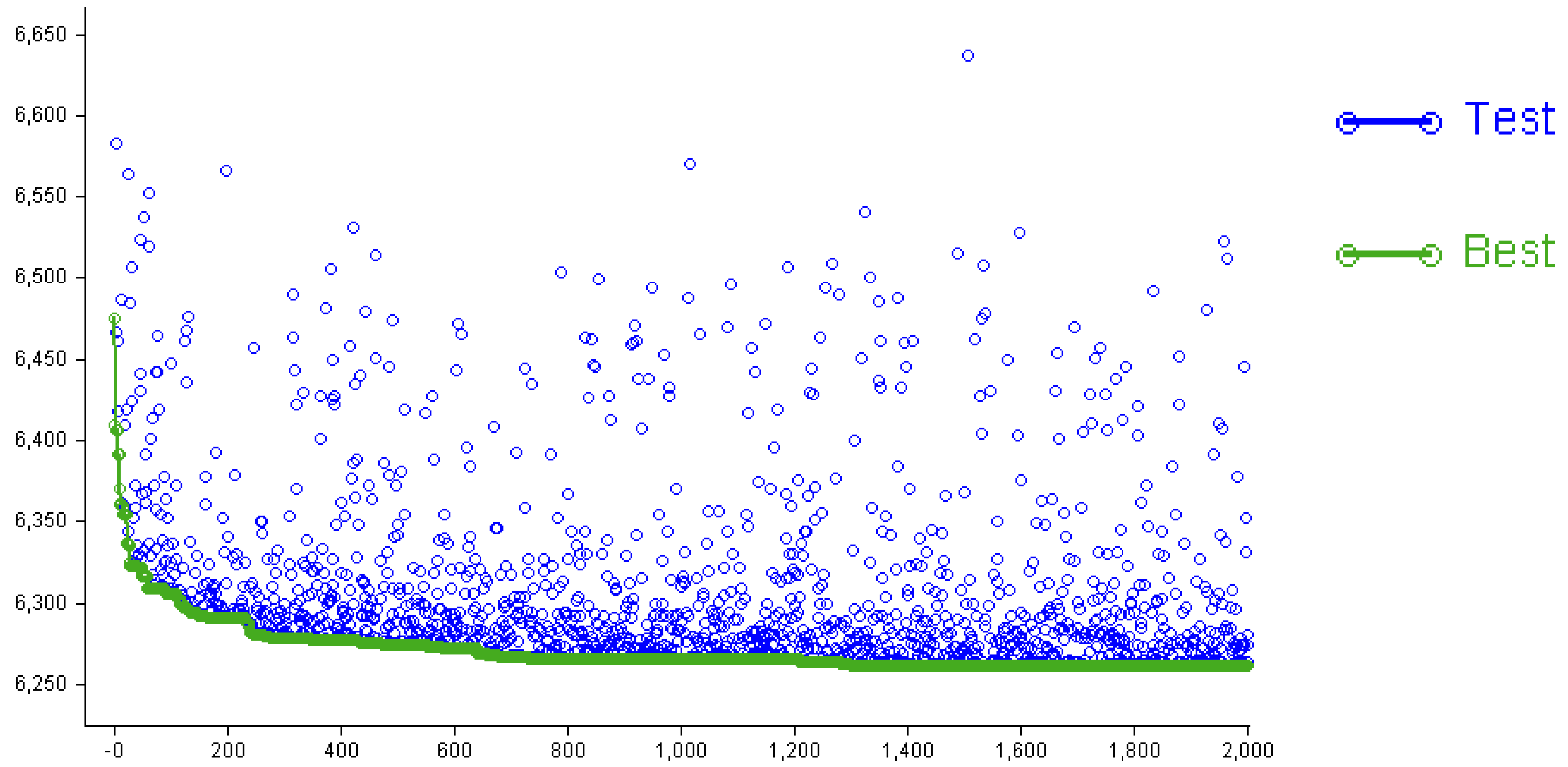
Updating T
- Many other update schemes

Re-boil (== Restart)
- Re-initialise T

0-cost changes
- Handle randomly

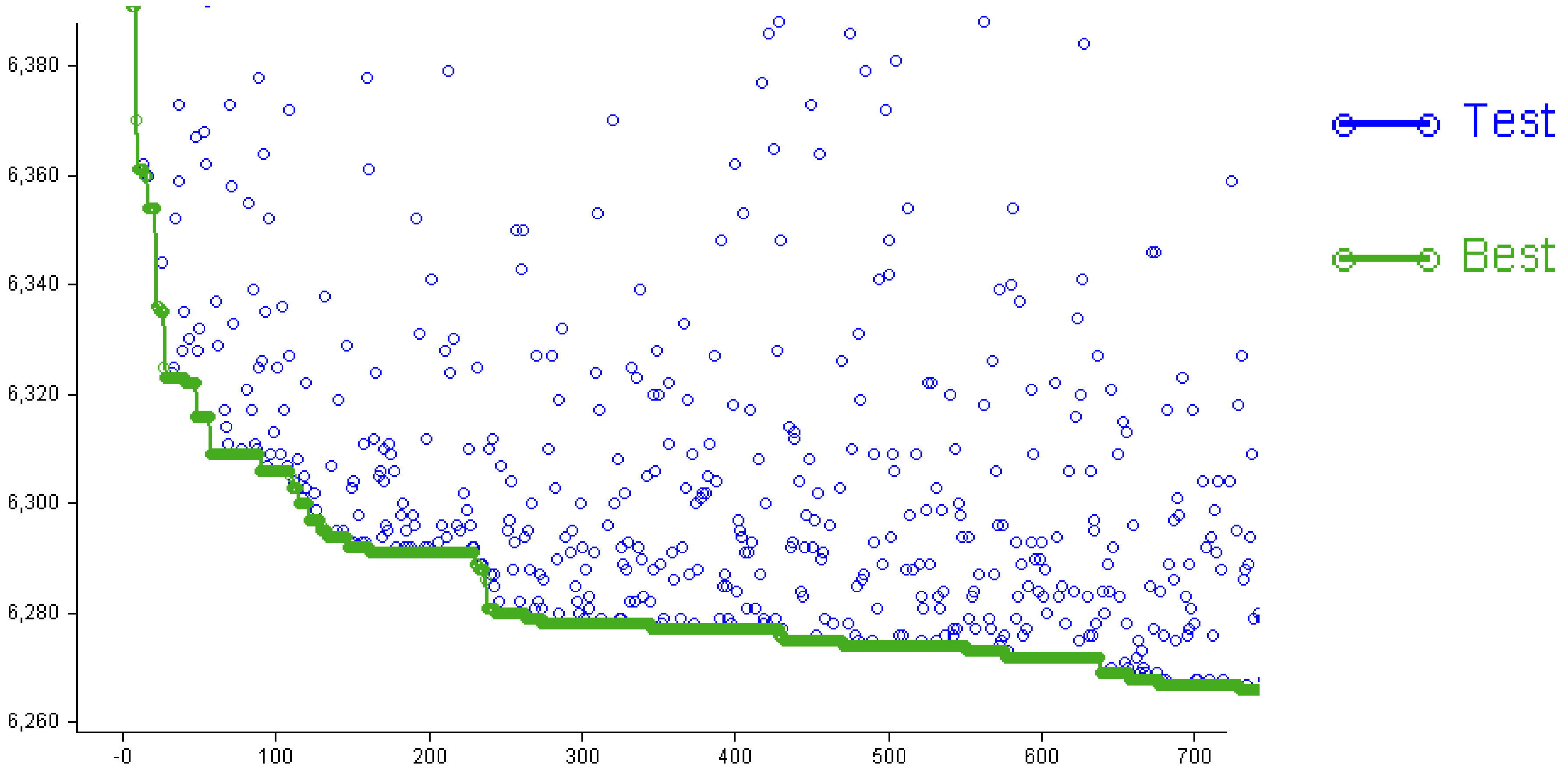
Adaptive parameters
- If you keep falling into the same local minimum,
  - `maxChangesThisT *= 2, or initialT *= 2`

VRP – Greedy Search

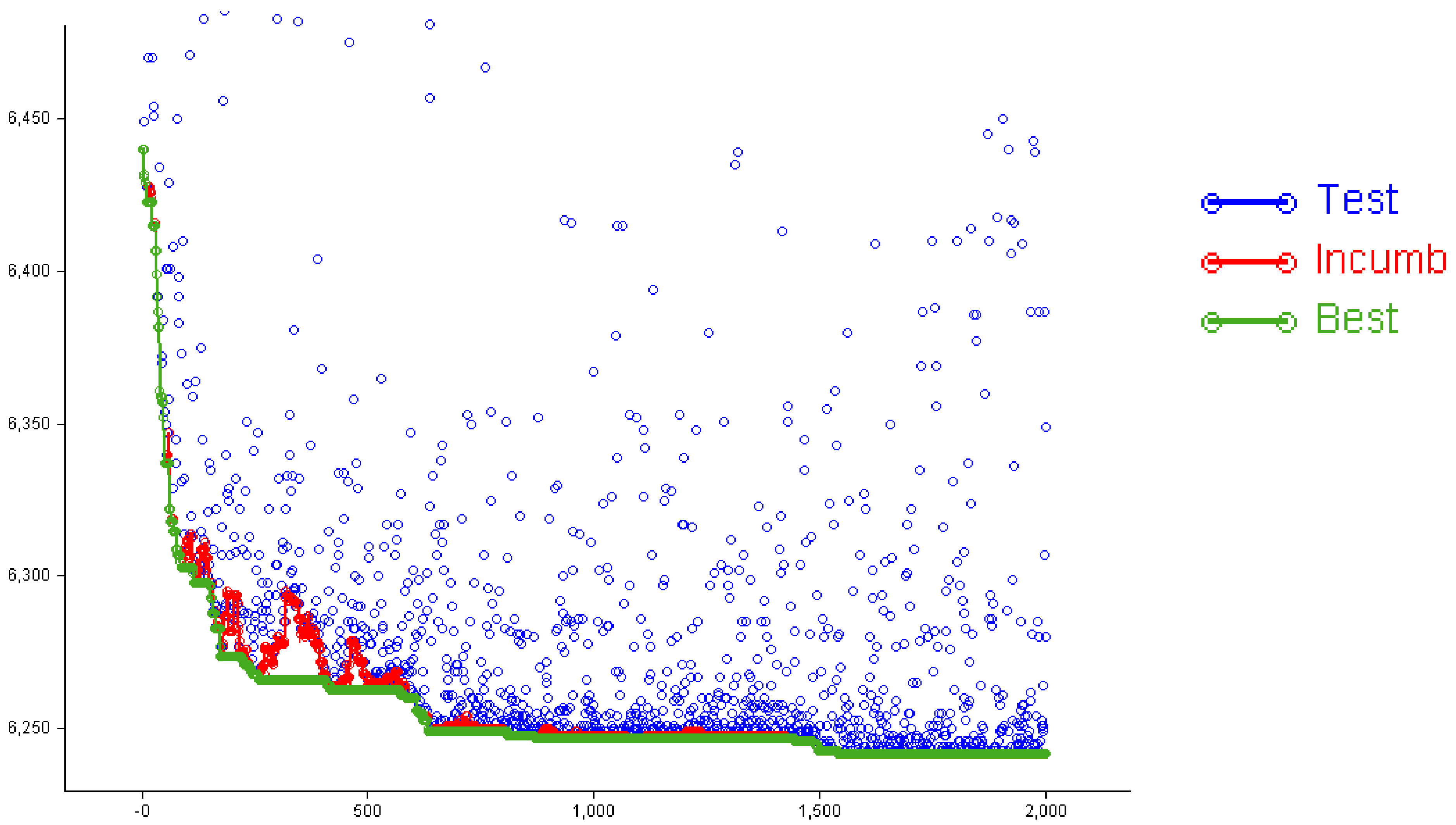


VRP – Greedy Search

Zoom in on the first 700 iterations

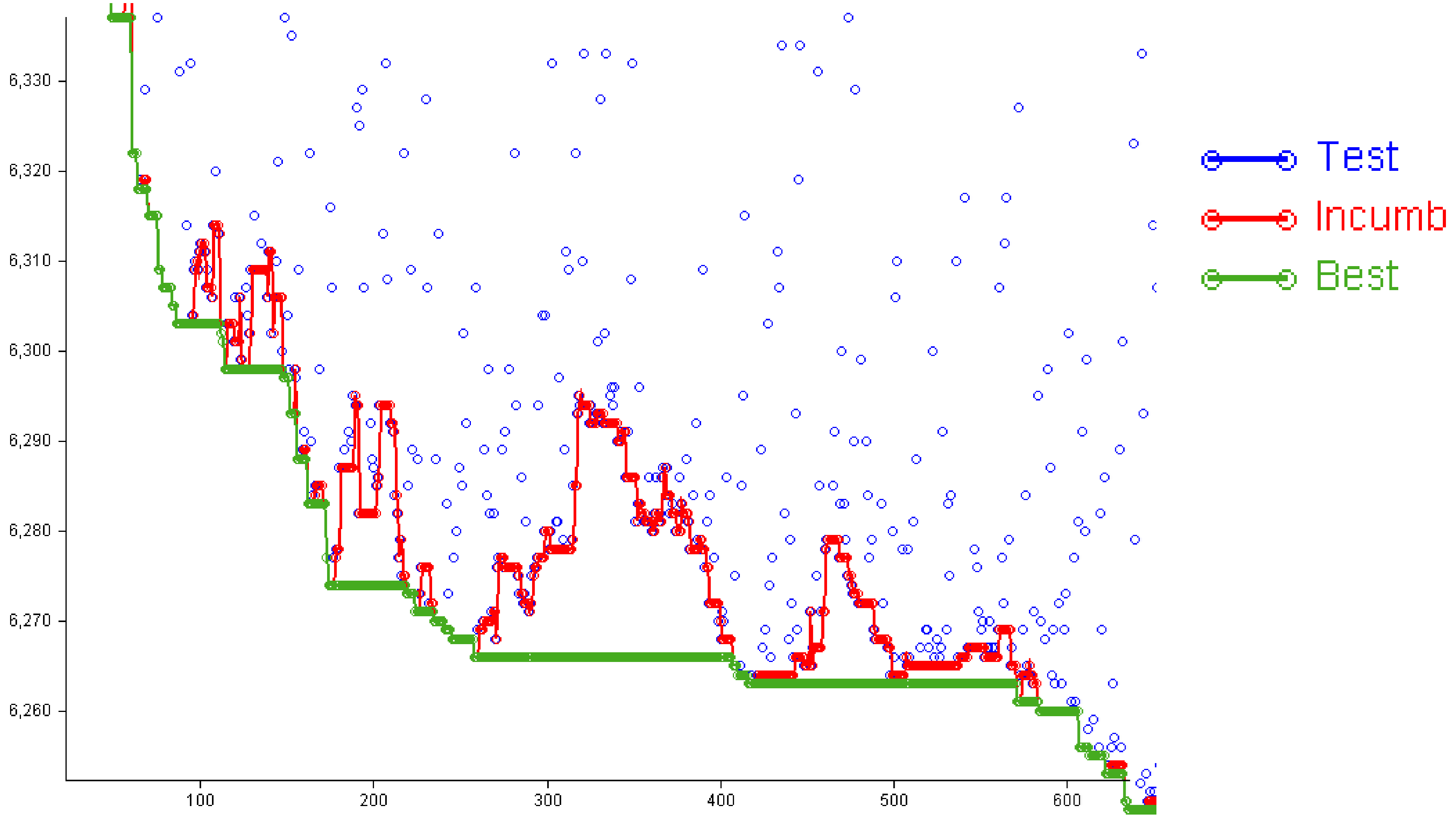


VRP – Simulated Annealing



VRP – Simulated Annealing

Zoom in on the first 600 iterations



Summary

- We have defined Local Search in terms of a *Neighbourhood* defined by an operator
- We have seen several ways of moving through the search space
  - Greedy, Random, Randomised Greedy, …
- We have seen some features of the solution space can mess up local search
  - Disconnected spaces, Shoulders, Local minima, …
- We have seen how to try to alleviate these problems
  - Sideways moves, Restarts, Simulated Annealing, …

Summary

Simulated Annealing is an example of a *meta-heuristic*

- A heuristic that applies to heuristics

Next week

- More meta-heuristics