Combinatorial Optimisation and Heuristics for Getting Around --PART 1--

Slides are (mostly) by: Phil Kilby
Speaker today is: Charles Gretton

Outline

PART 1

- Combinatorial Optimisation
- The vehicle routing problem (VRP)
  - VRP Variants
- Solving the Combinatorial Optimisation problems
  - Exact methods
  - Heuristic Construction
  - Auction
  - Local Search
  - Meta-heuristics

PART 2

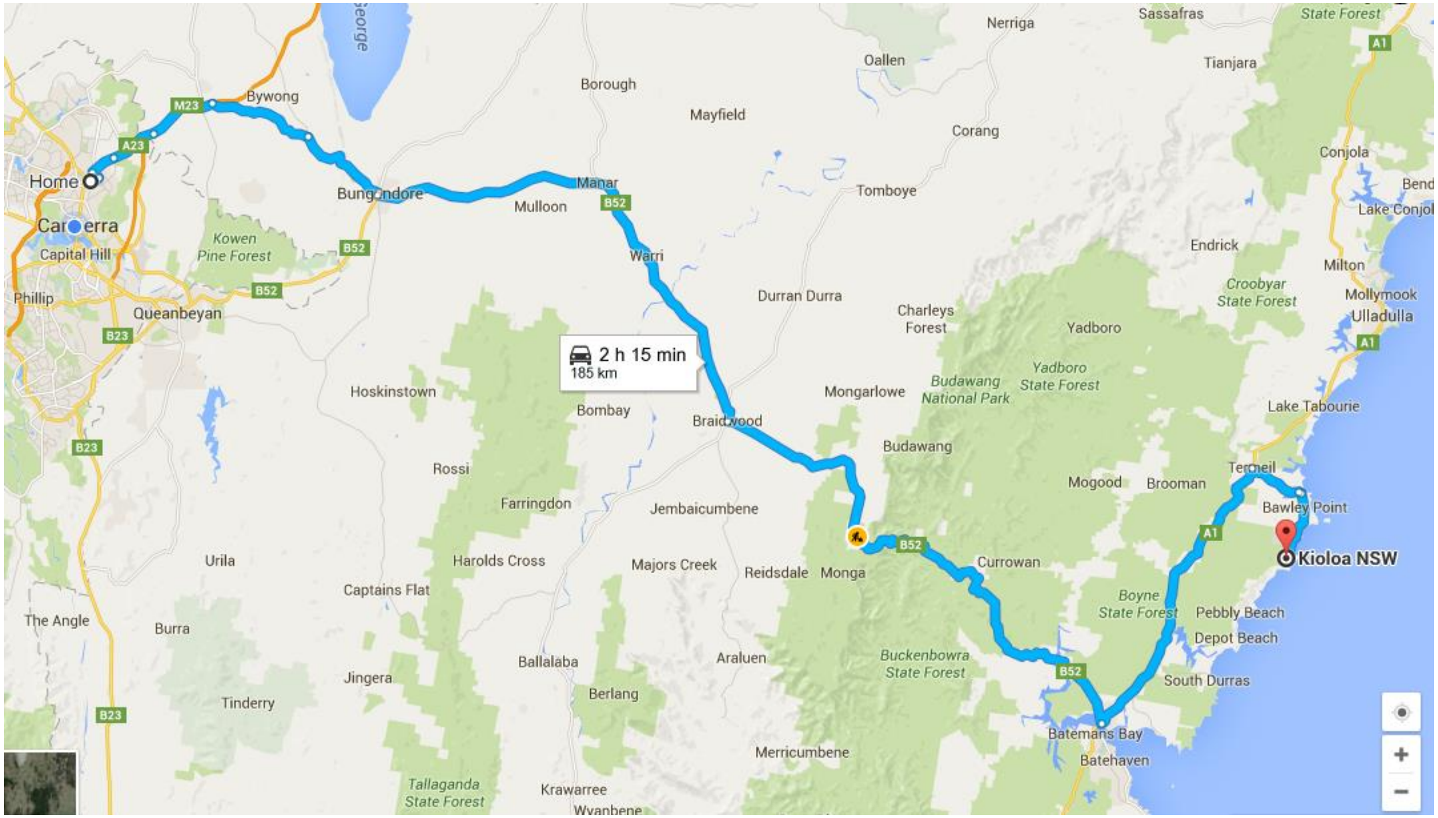
- Large Neighbourhood Search
- A CP model for the VRP
- Propagation
- Large Neighbourhood Search revisited

What is Combinatorial Optimisation?

- What is art?
- Problem will be described using language like:
  - Variables, assignments, constraints, **objective criterion**
  - Sets, subsets, permutations and combinations
  - Vertices, nodes, edges, cliques, paths, cycles
- Solutions are usually finite discrete objects
- Usually interested in problems where exhaustive search is infeasible
- Prototypical deep learning solutions were first described in 1985 (Hopfield and Tank)

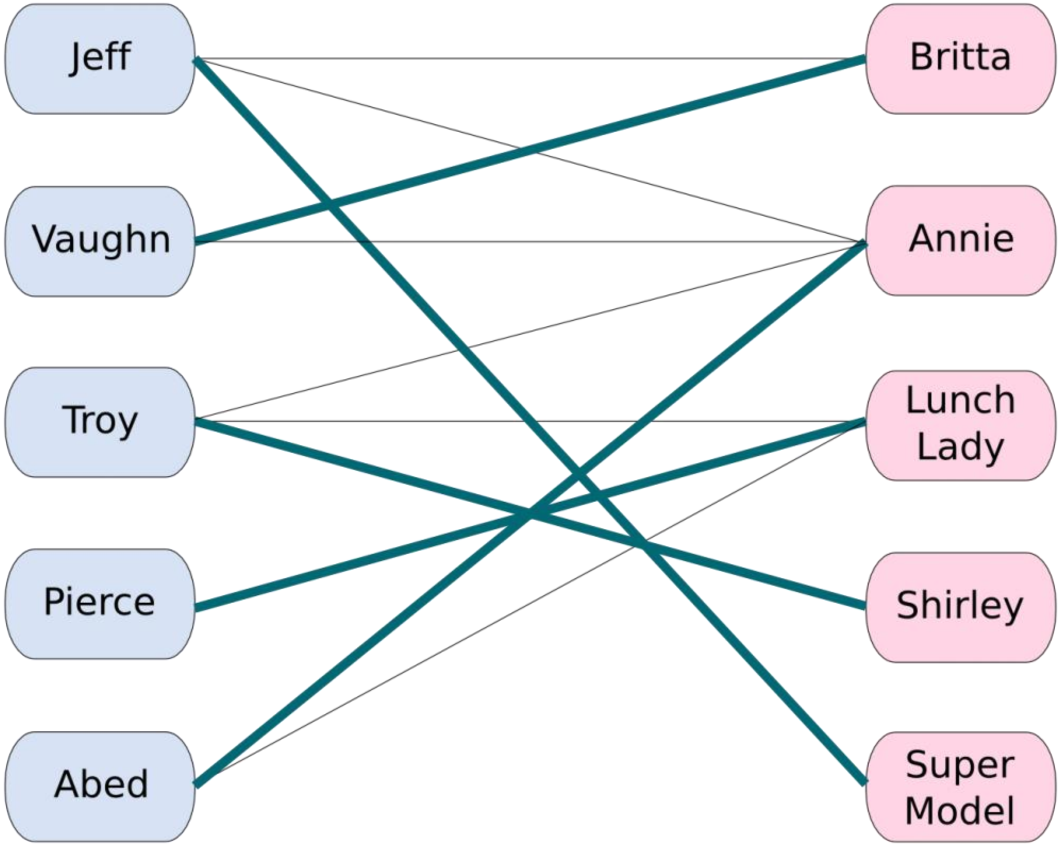
Shortest Path Problem

Find shortest path between two nodes



Bipartite Matching

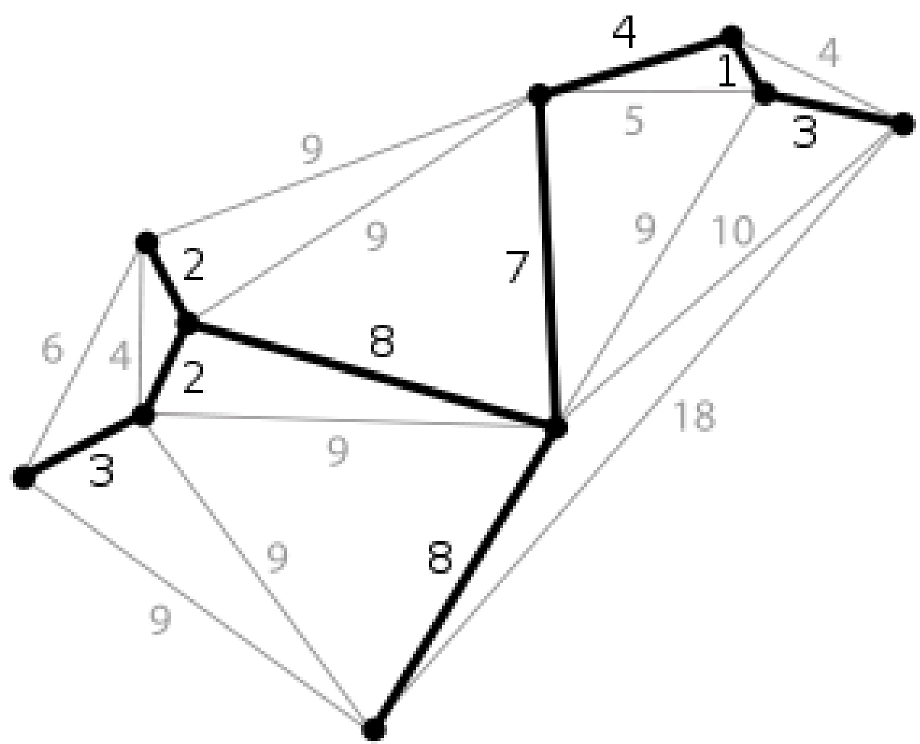
Find the pairing with minimum cost



Caveat : Example pre-dates Australia's 2018 marriage equality laws

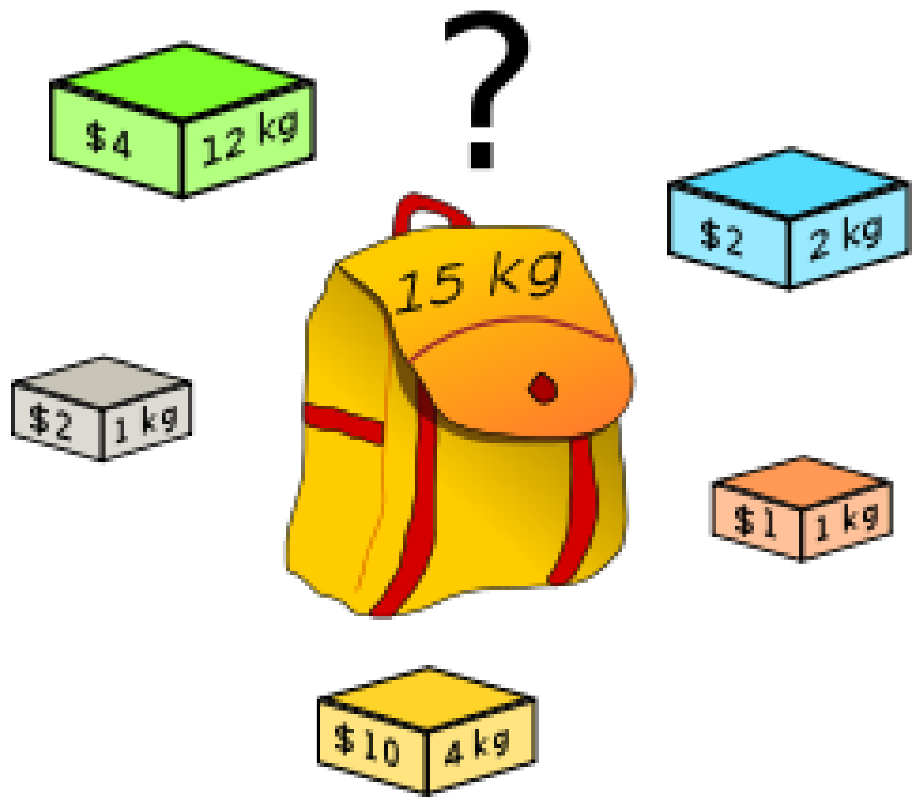
Minimum Spanning Tree

Find the spanning tree of minimum weight



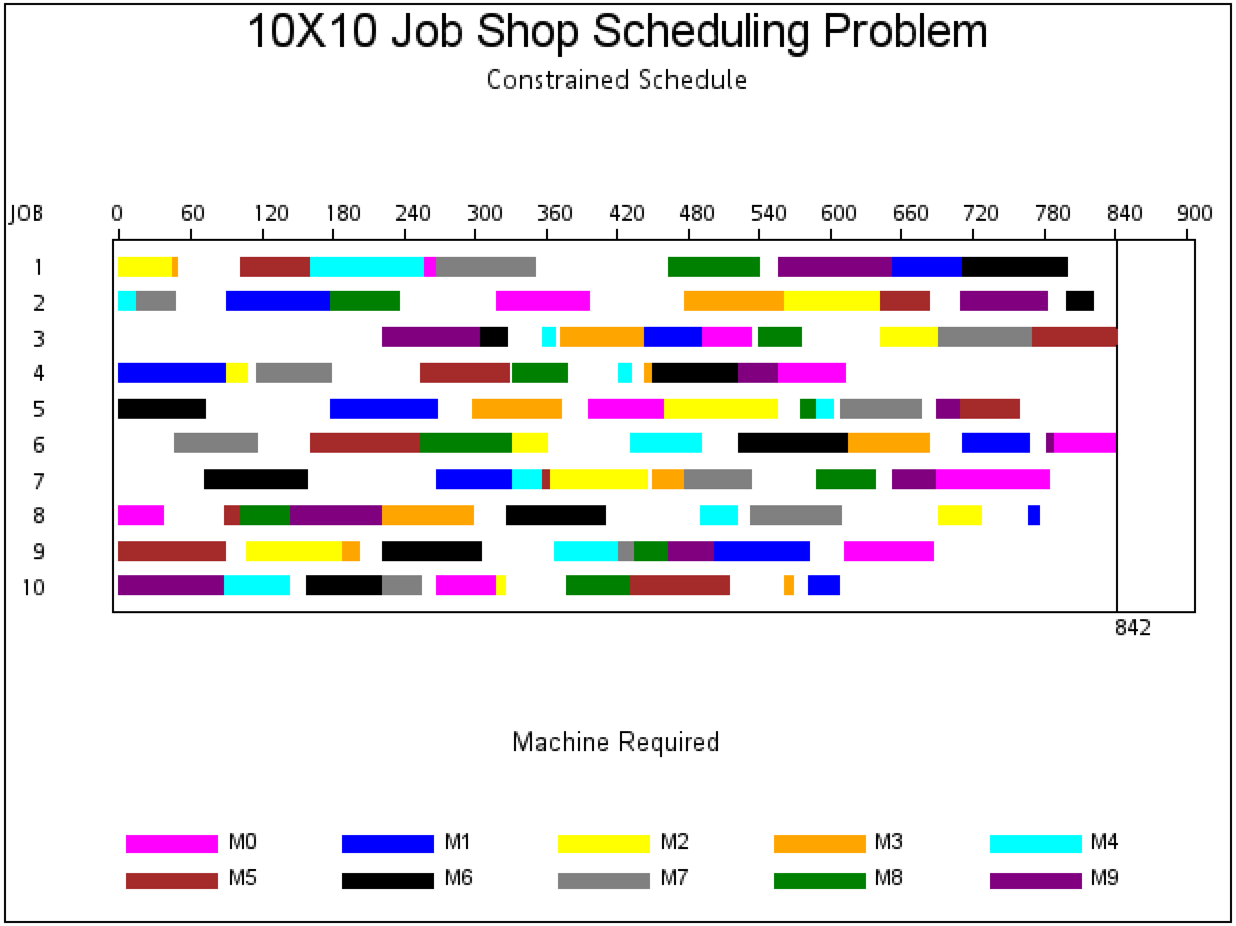
Knapsack / Bin Packing problem

Choose items to give the maximum value within a given capacity.



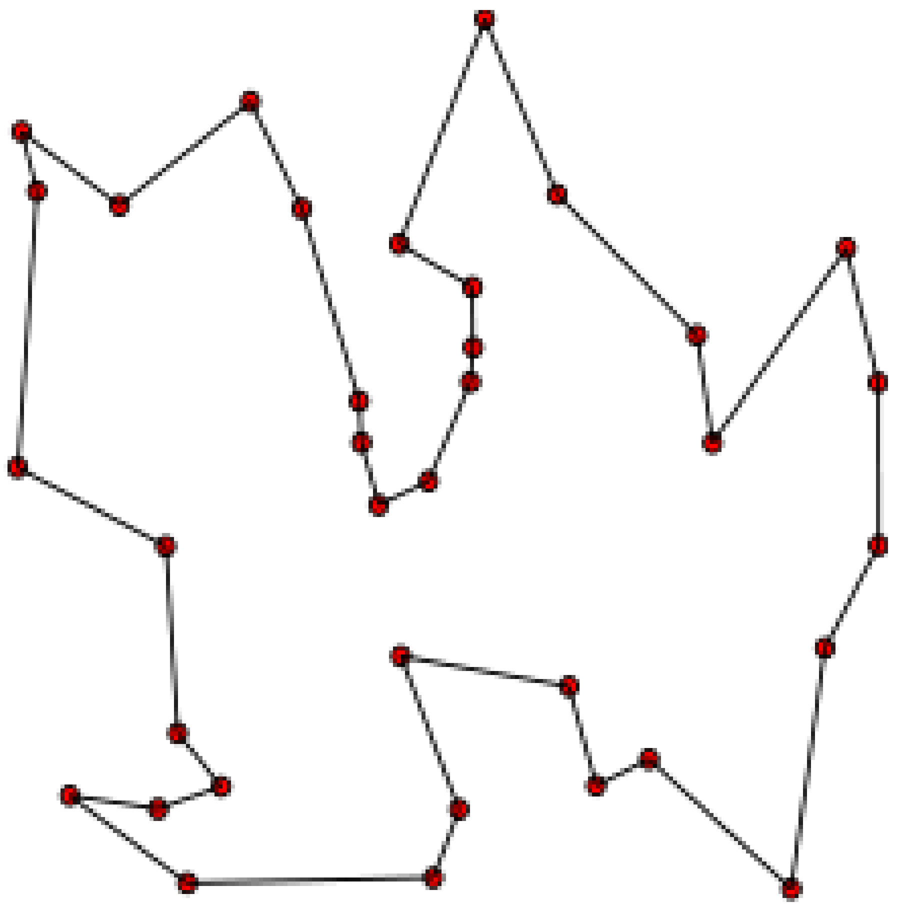
Jobshop Scheduling

Schedule jobs on machines to minimize total time



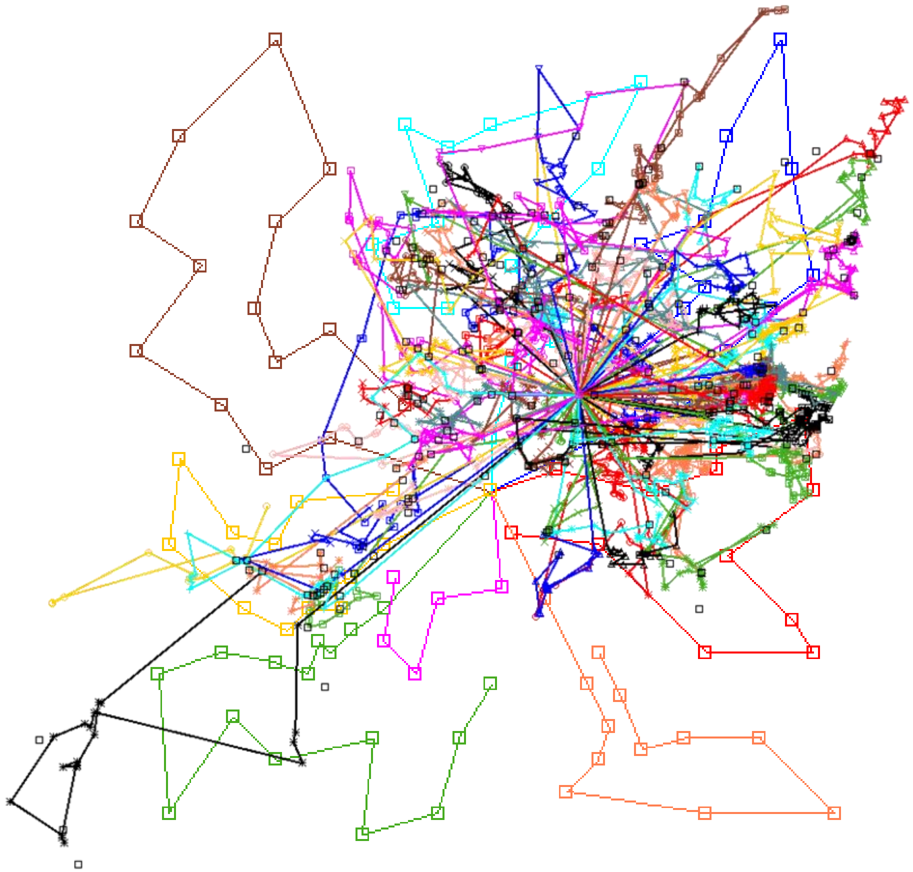
Travelling Salesman Problem

Find the tour of minimum cost that visits all cities



Vehicle routing problem

Given a set of customers, and a fleet of vehicles to make deliveries, find a set of routes that services all customers at minimum cost



Combinatorial Optimisation

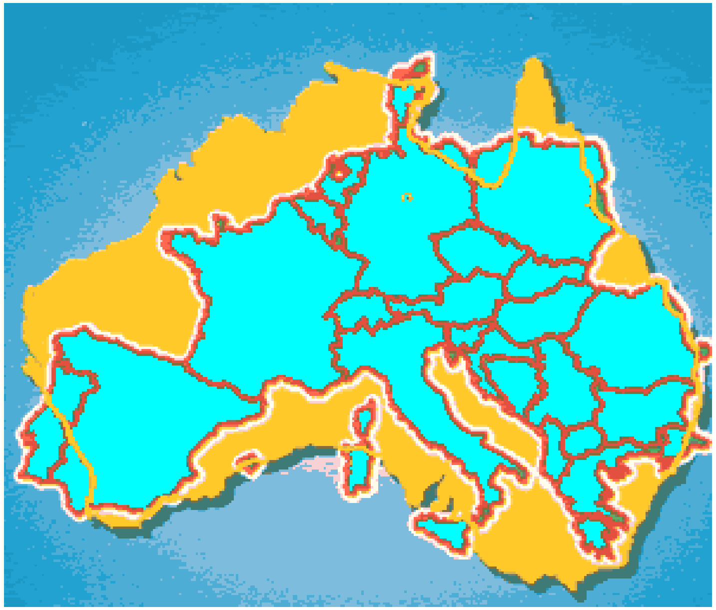
- Usually a graph or set representation

- Some easy (Polynomial)
  - SPP (sortest path problem),
  - MST (minimum spaning tree),
  - Matching (min/max cost/revenue matching)
- Some hard (NP Hard)
  - ESSPWRC (Elementary shortest path with resource constraints),
  - TSP (travelling sales person),
  - VRP (vehicle routing problem)
- (Some in-between: Knapsack: “Easy NP”)

- Many real-world applications
  - Add: Garry is not allowed to see Steve while he is driving his truck
  - Add: Anthony likes his sleep

Why study the VRP?

- It's hard: it exhibits all the difficulties of comb. opt.
- It's useful:
  - The logistics task is 9% of economic activity in Australia
  - Logistics accounts for 10% of the selling price of goods



Vehicle Routing Problem

For each customer, we know

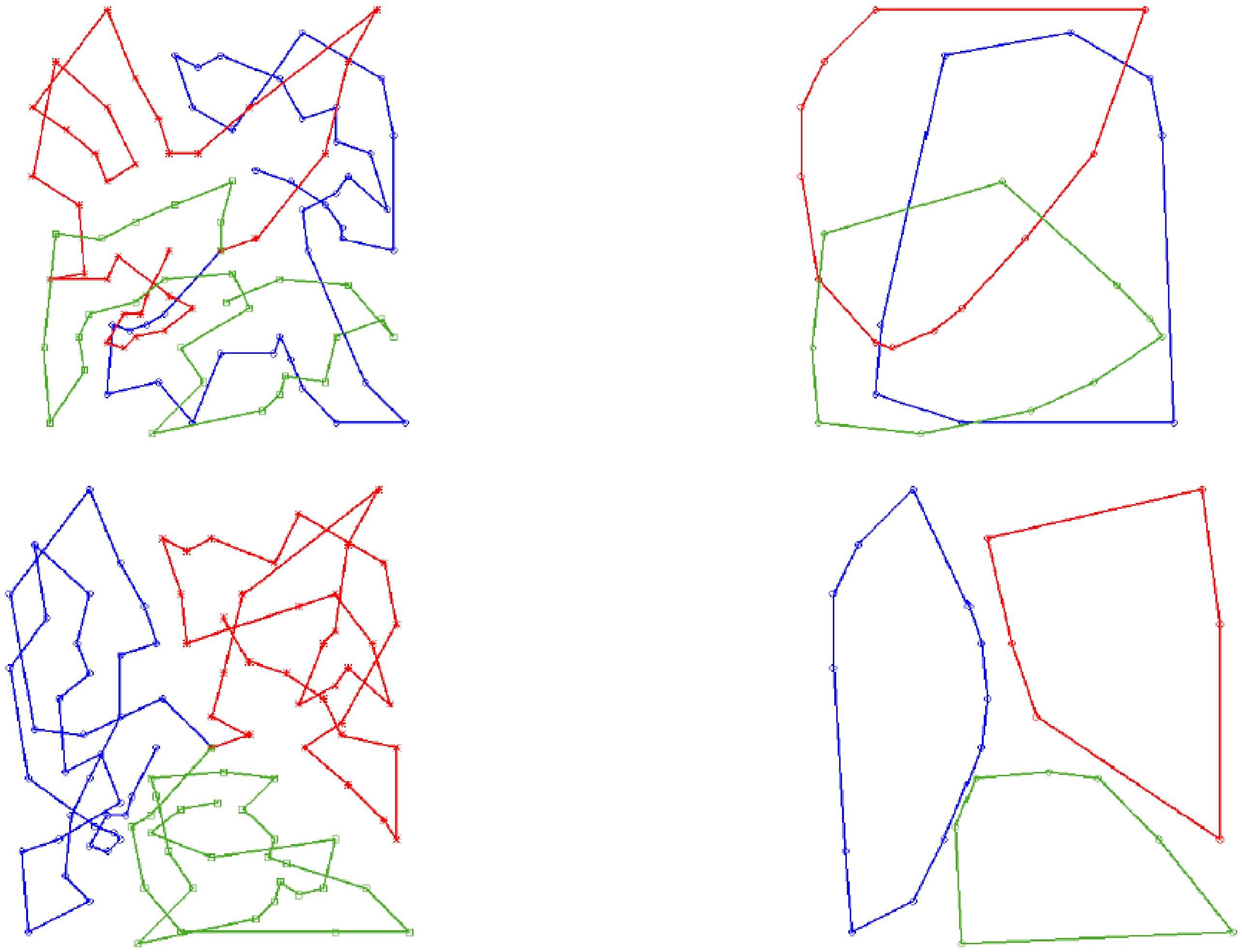
- Quantity required
- The cost to travel to every other customer

For the vehicle fleet, we know

- The number of vehicles
- The capacity

We must determine which customers each vehicle serves, and in what order, to minimise cost

Vehicle Routing Problem



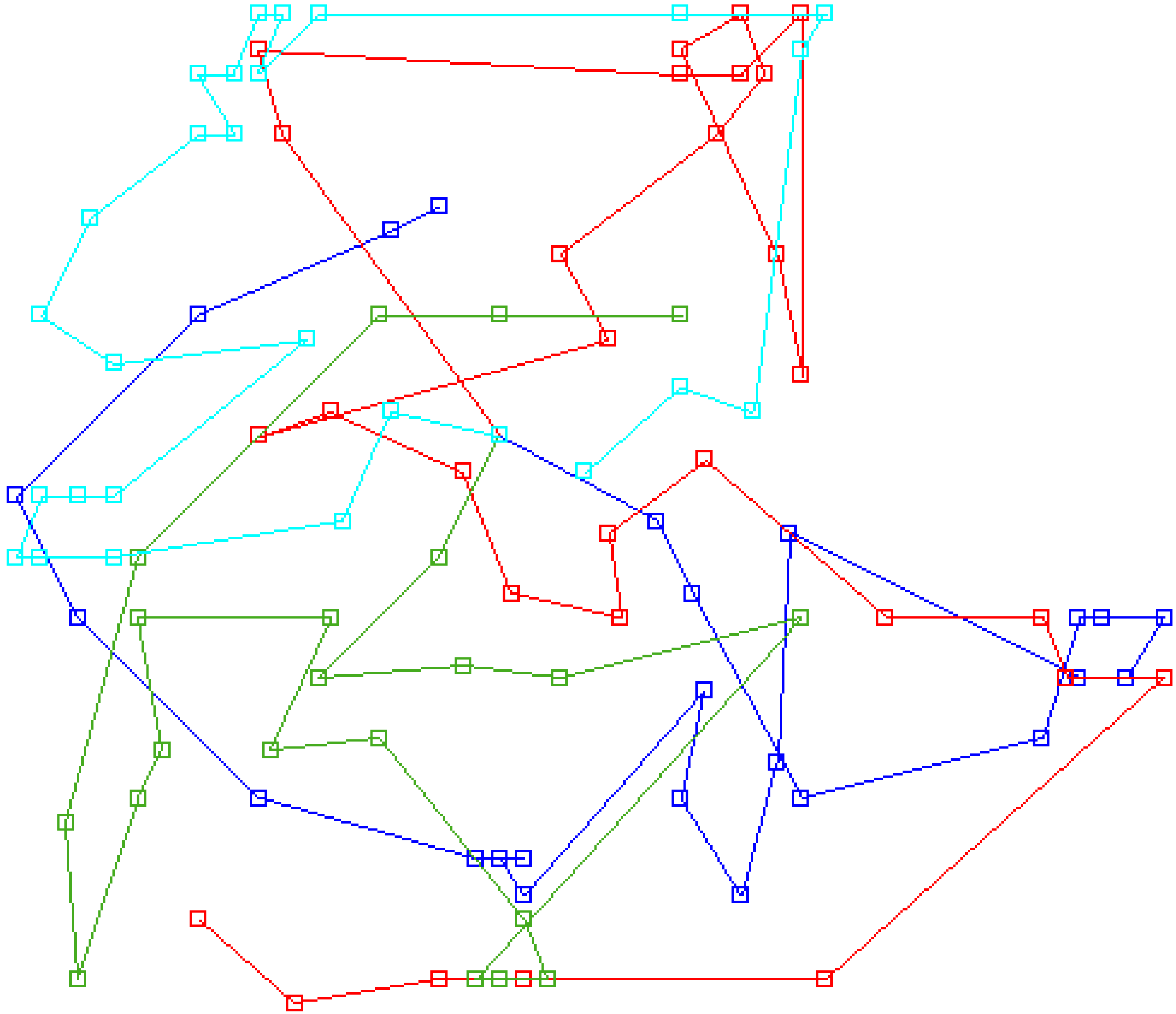
Vehicle Routing with constraints

- Time Window constraints
  - A window during which service can start
  - E.g. only accept delivery 7:30am to 11:00am

  - Additional input data required
    - Duration of each customer visit
    - Time between *each pair* of customers

    - (Travel time can be vehicle-dependent or time-dependent)
  - Makes the route harder to visualise

Time Window constraints



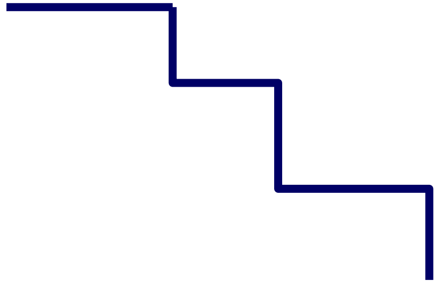
Pickup and Delivery problems

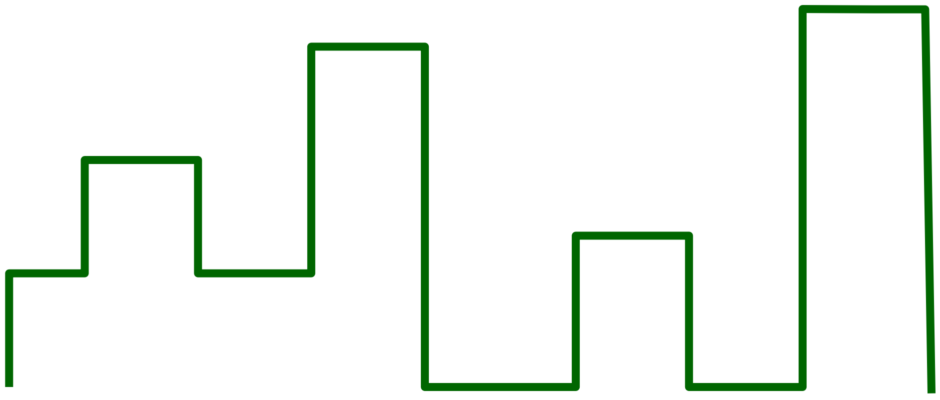
•Most routing considers delivery to/from a depot (depots)

•Pickup and Delivery problems consider FedEx style problem:

pickup at location A, deliver to location B

Load profile:

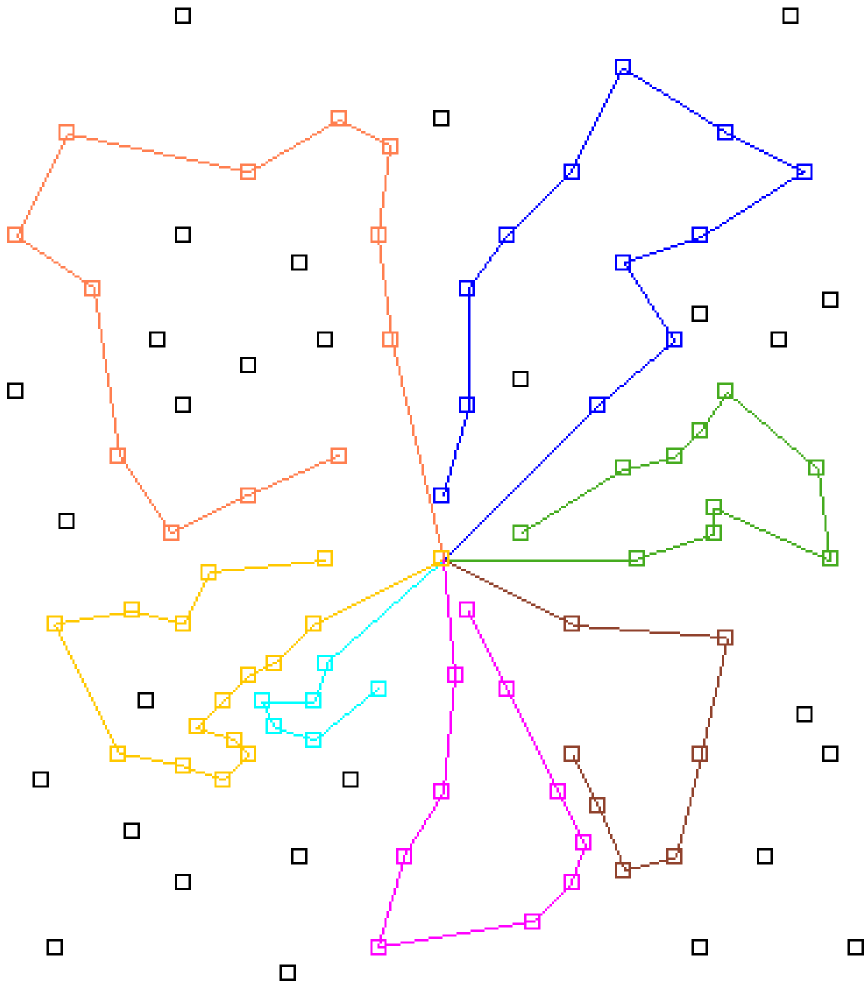




Other variants

Profitable tour problem

- Not all visits need to be completed
- Known profit for each visit
- Choose a subset that gives maximum return
(*profit from visits – routing cost*)



VRP meets the real world

Many groups now looking at real-world constraints

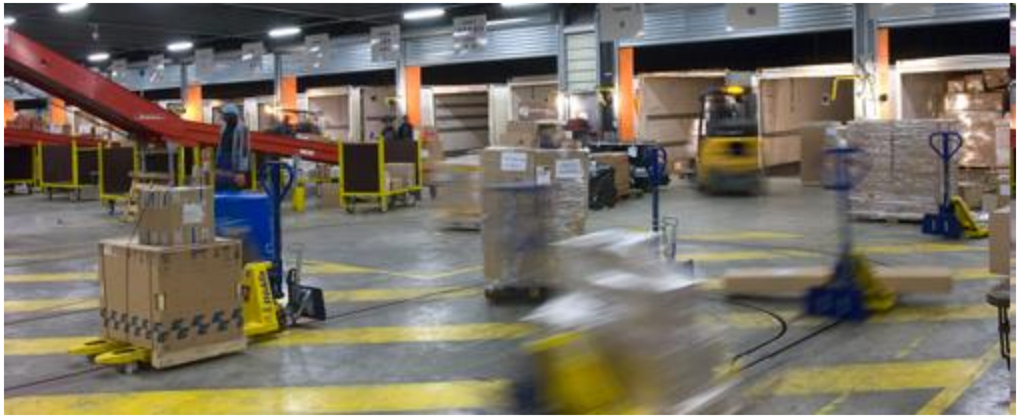
Rich Vehicle Routing Problem

- Attempt to model constraints common to many real-life enterprises
  - Multiple Time windows
  - Multiple Commodities
  - Multiple Depots
  - Heterogeneous vehicles
  - Compatibility constraints
    - Goods for customer *A* {must | cannot} travel with goods from customer *B*
      - *Cardboard and Glass*
    - Goods for customer *A* {must | cannot} travel on vehicle *C*
      - *Ambient v.s. Chilled*

VRP meets the real world

Other real-world considerations

- Fatigue rules and driver breaks
- Vehicle re-use
- Ability to change vehicle characteristics (composition)
  - Add trailer, or move compartment divider
- Use of limited resources
  - e.g limited docks for loading, hence need to stagger dispatch times
- Variable loading / unloading times



Solving Combinatorial Problems

Solution Methods

Exact:

- Bespoke Method (e.g. “Hungarian” method for matching)
- Integer Programming or Mixed Integer Programming
- Constraint Programming

Heuristic:

- Construct
- Improve
    - Local Search
    - Meta-heuristics

Exact Methods

VRP:

- MIP: Can only solve problems with 50-100 customers
- CP: Similar size
  - Session 2

ILP

$$\text{minimise} \sum_{ij} c_{ij} \sum_{k} x_{ijk}$$

$$\text{subject to} \sum\nolimits_i \sum\nolimits_k x_{ijk} = 1, \qquad \forall j$$

$$\sum\nolimits_j \sum\nolimits_k x_{ijk} = 1, \qquad \forall i$$

$$\sum\nolimits_j \sum\nolimits_k x_{jhk} - \sum\nolimits_j \sum\nolimits_k x_{hjk} = 0, \qquad \forall kh$$

$$\sum\nolimits_i q_i \sum\nolimits_j x_{ijk} \leq Q_k, \qquad \forall k$$

$$\sum\nolimits_{x_{ijk} \in S} x_{ijk} = |S| - 1, \qquad S \subseteq P(N), 0 \notin S$$

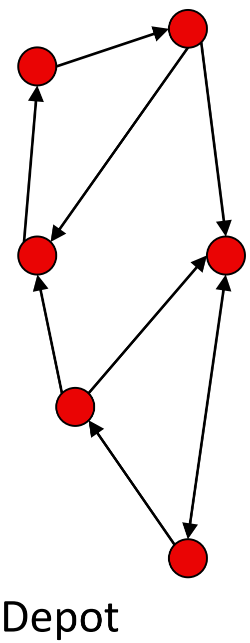
$$x_{ijk} \in \{0, 1\}$$

Advantages

•Can find optimal solution

Disadvantages

- Only works for small problems
- One extra constraint → back to the drawing board
- S is huge



ILP – Column Generation

Columns represent routes

Column/route cost c_k

Rows represent customers

Array entry $a_{ik} = 1$ iff customer i is covered by route k

	89	*76*	*99*	*45*	*32*	
1	1	1	1	0	0	…
2	0	1	1	0	1	…
3	0	0	0	0	0	…
4	1	0	1	1	0	…
5	1	0	0	0	0	…

Column Generation – The Master

Want more details: see 2010 Tutorial by Dominique Feillet

- Decision var x_k: Use column k?
- Column only appears if feasible ordering is possible
- Cost of best ordering is c_k
- Best order stored separately
- Master problem at right

$$\textit{minimise} \quad \sum_k c_k x_k$$

$$\textit{subject to}$$

$$\lambda_i \qquad \sum_k x_k a_{ik} \geq 1, \qquad \forall i$$

Dual Solution → λ_i

Column Generation – The "Subbie"

Want more details: 2005 Textbook by Stefan Irnich and Guy Desaulniers

Subproblem: add a column to the *basis* with negative *reduced costs*

$$\min \sum_{ij} x_{ij}(c_{ij} - \lambda_i)$$

- i.e. above equation is the objective of a shortest path problem
- Elementary shortest path problem with **resource constraints**
- If subproblem value –ve add column (path) to master problem
    - Subproblem is intractable,
    - DP solutions are state-of-the-art,
    - worth considering CP for realistic problems with unusual constraints

Heuristic Methods

Construction:

- Start with an empty solution
- Add one element to the solution at a time
- “Greedy” methods
  - Look around and find the “best” move locally
  - Do it
  - Repeat

- e.g. Knapsack: Insert items with best value/volume ratio

Heuristics for the VRP

Construction by Insertion

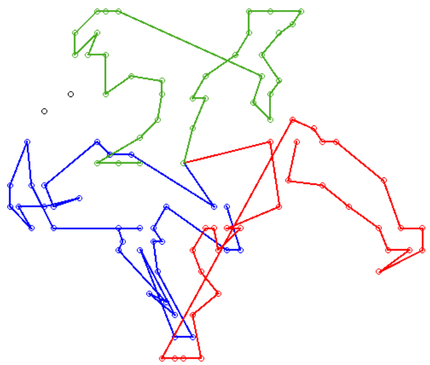
- Start with an empty solution
- Repeat
  - Choose which customer to insert
  - Choose where to insert it

E.g. (Greedy)

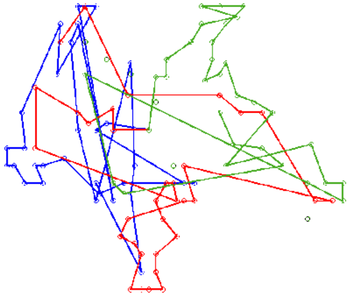
- Choose the customer that increases the cost by the least
- Insert it in the position that increases the cost by the least

Solving the VRP the easy way

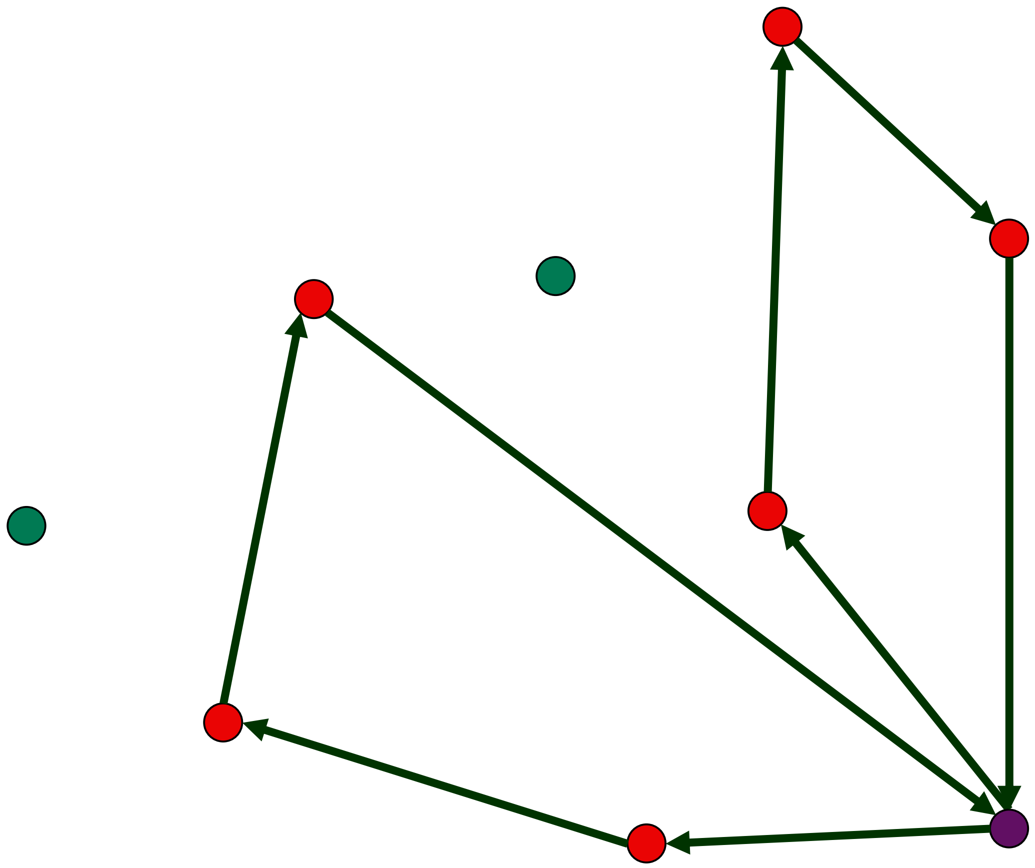
Insert methods



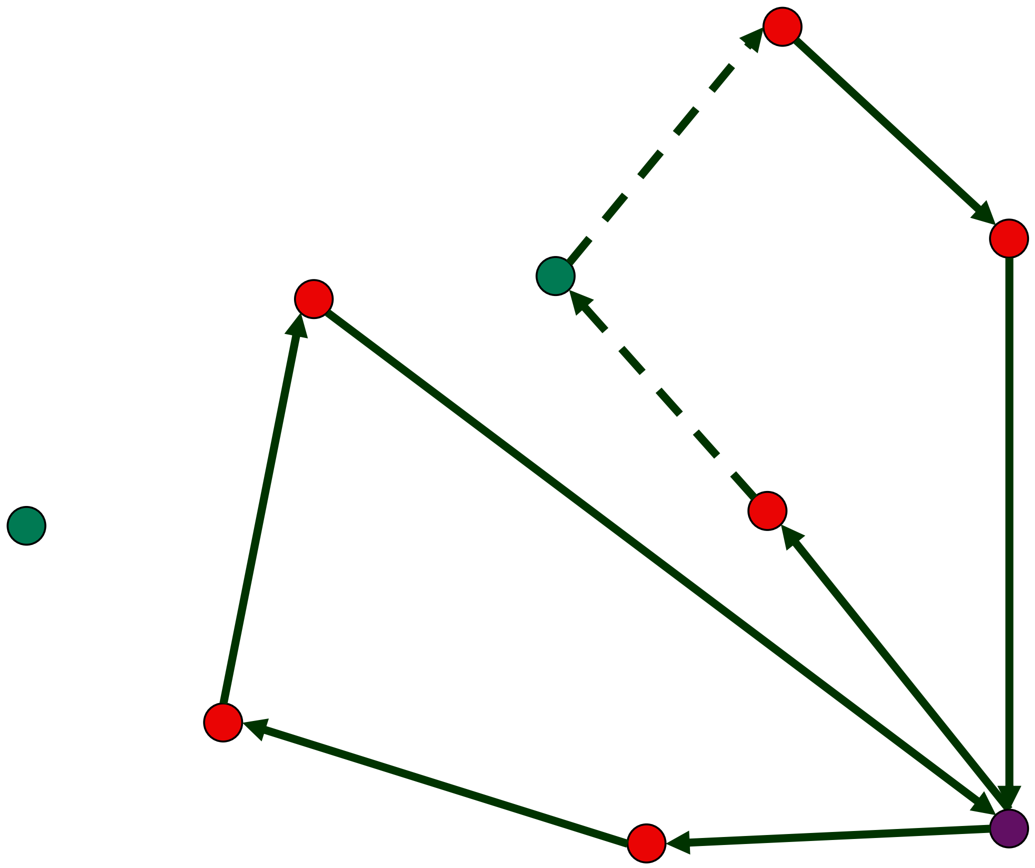
Order is important:



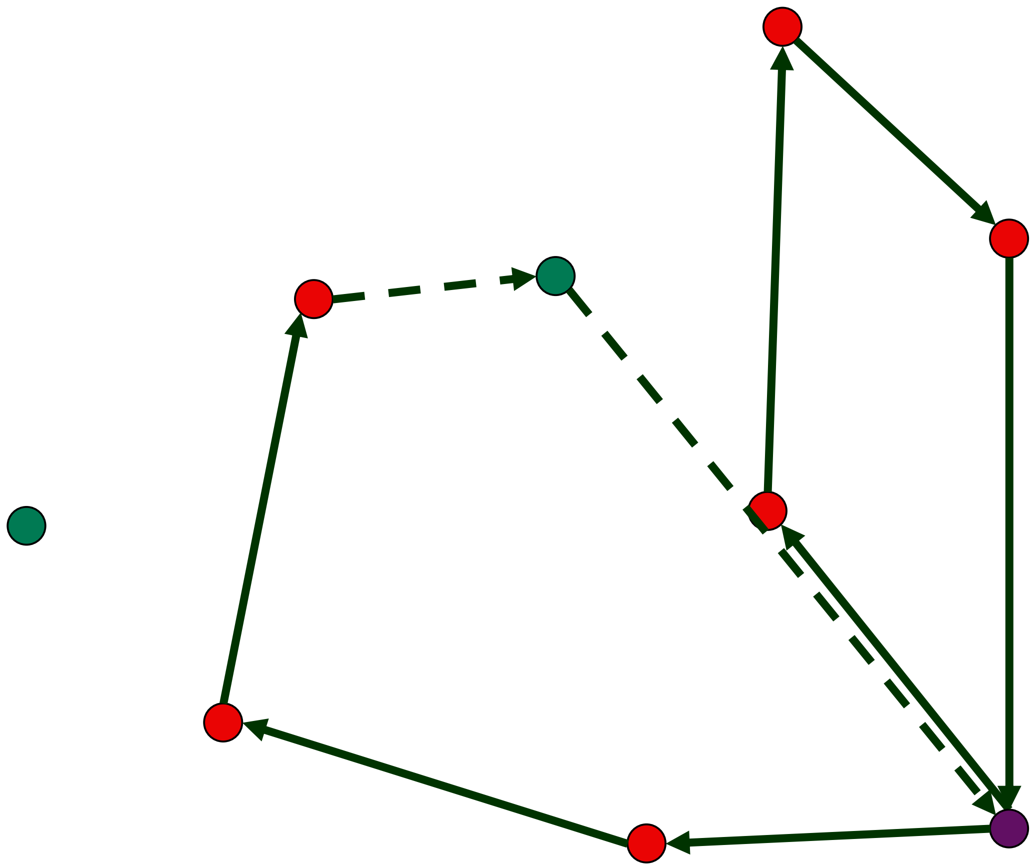
Regret



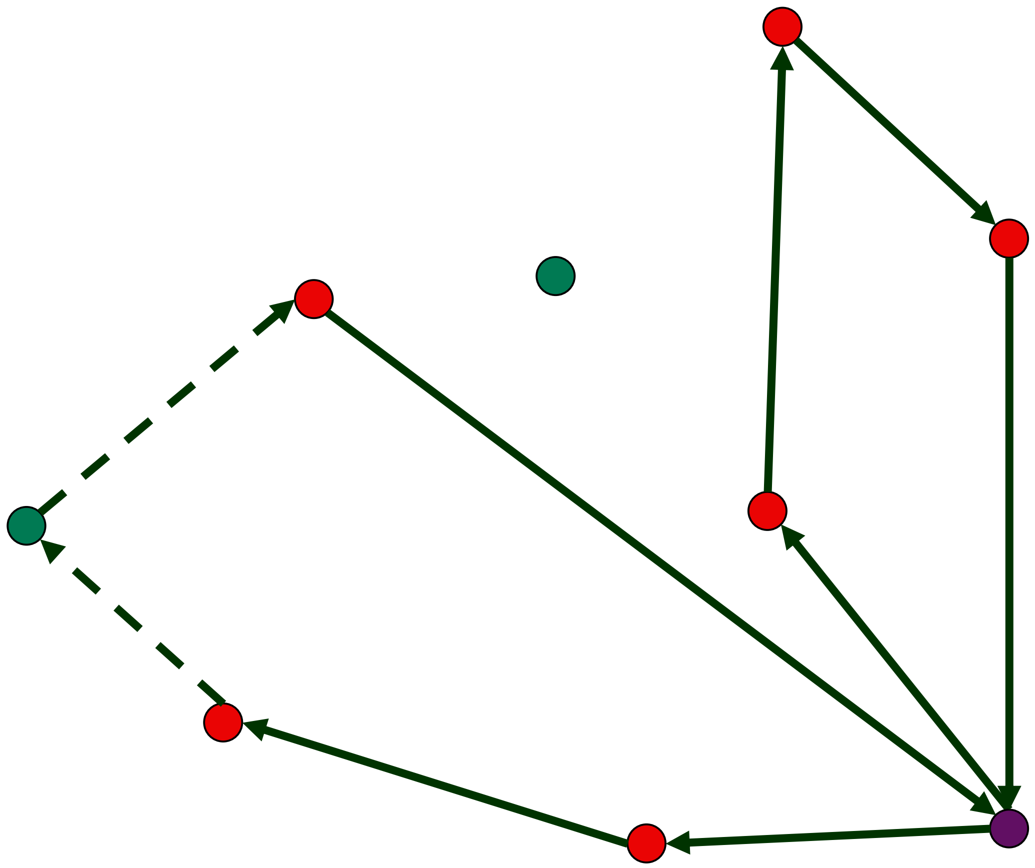
Regret



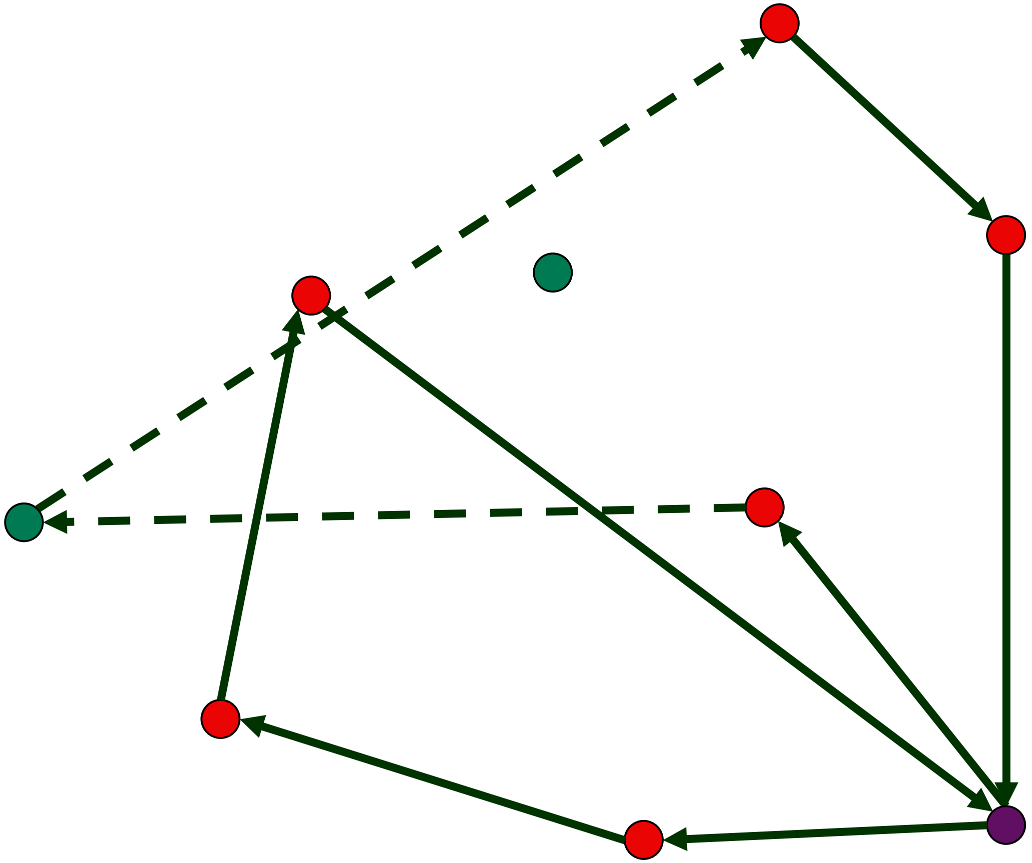
Regret



Regret



Regret



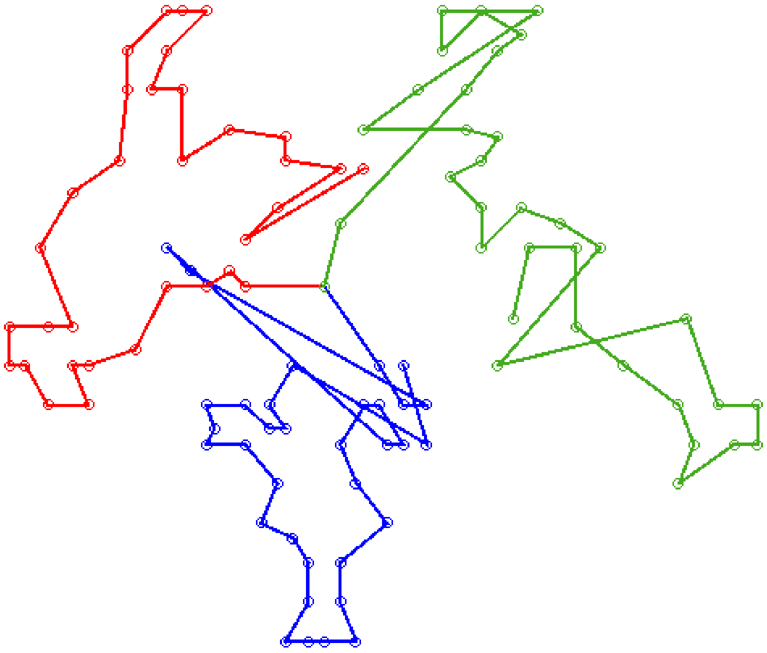
Regret

Regret = C(insert in 2nd-best route) – C(insert in best route)

$= f(2,i) - f(1,i)$

K-Regret = $\sum_{k=1,K} (f(k,i) - f(1,i))$

Insertion with Regret



Auction Algorithm

Constrained allocation e.g. Bipartite Matching

- Cost c_{ij} to allocate thing i to person j
- Find the matching that minimises the sum of costs
- x_{ij} = 1 if i is assigned to j, 0 otherwise

$$\min \sum c_{ij} x_{ij}$$

Subject to

$$\sum_j x_{ij} \leq 1 \;\forall i$$

$$\sum_i x_{ij} = 1 \;\forall j$$

Auction Algorithm

Start with Greedy Allocation:

- Give each person the lowest cost thing

Repeat

- If no thing is over-subscribed,
  - We have a solution! Stop
- Else
  - Find the least cost that will make someone flip their choice *amongst all oversubscribed things*
  - Increase the cost of the thing ***for everybody*** by that much
  - Give each person the lowest cost thing (using new costs)

Auction Algorithm

$$\min \sum c_{ij} x_{ij}$$

Subject to

$$\sum_j x_{ij} = 1 \ \forall i \qquad \lambda_i$$

$$\sum_i x_{ij} = 1 \ \forall j$$

Exercise

Solve a constrained matching problem using the Auction Algorithm

1. Score your preference for each sweet – **lower is better**!
2. Scores must sum to 100
3. Write your score on a sheet of paper
4. We will run an Auction algorithm to allocate the sweets

Choices, choices…

Full size bar!

Choices, choices…

Choices, choices…

Choices, choices…

Choices, choices…

~50cm!

Choices, choices…

5 snakes!

(colours may vary)

4 X

14 X

6 X

6 X

4 X

10 X

Auction Algorithm

Start with Greedy Allocation:

- Give each person the lowest cost thing

Repeat

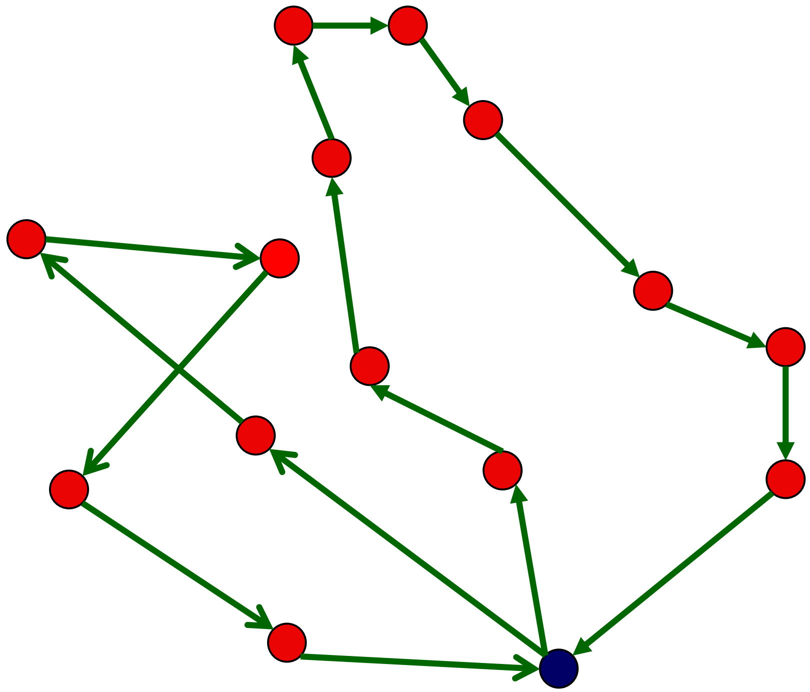
- If no thing is over-subscribed,
  - We have a solution! Stop
- Else
  - Find the least cost that will make someone flip their choice *amongst all oversubscribed things*
  - Increase the cost of the thing ***for everybody*** by that much
  - Give each person the lowest cost thing (using new costs)

Local Search

Improvement Methods

Local Search

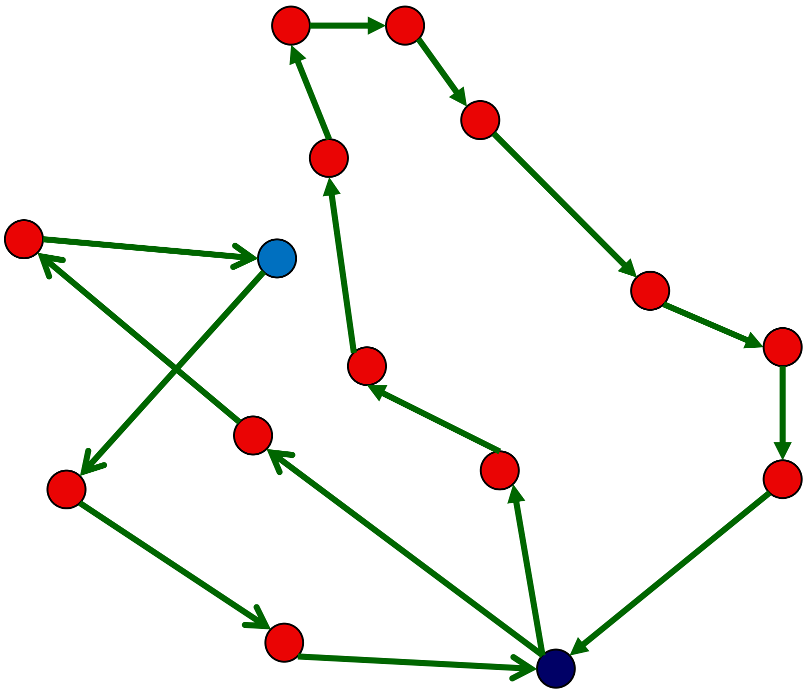
- Often defined using an “operator”



Improvement Methods

Local Search

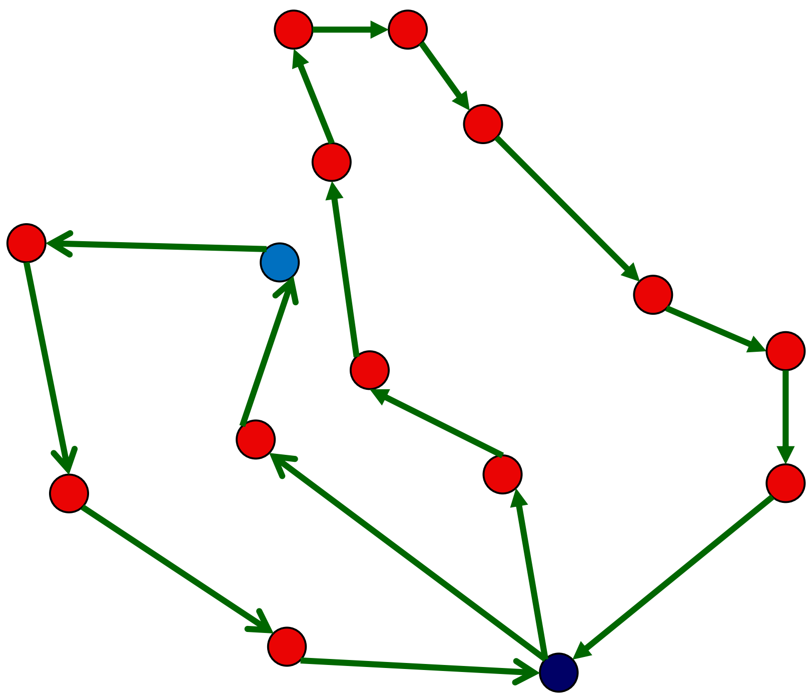
- Often defined using an "operator"
  - e.g. 1-move



Improvement Methods

Local Search

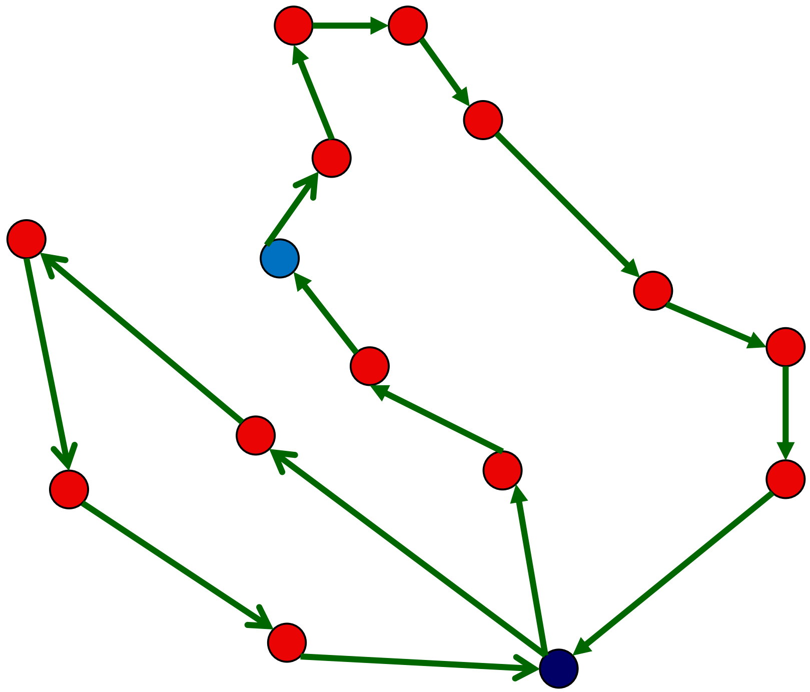
- Often defined using an "operator"
  - e.g. 1-move



Improvement Methods

Local Search

- Often defined using an "operator"
  - e.g. 1-move

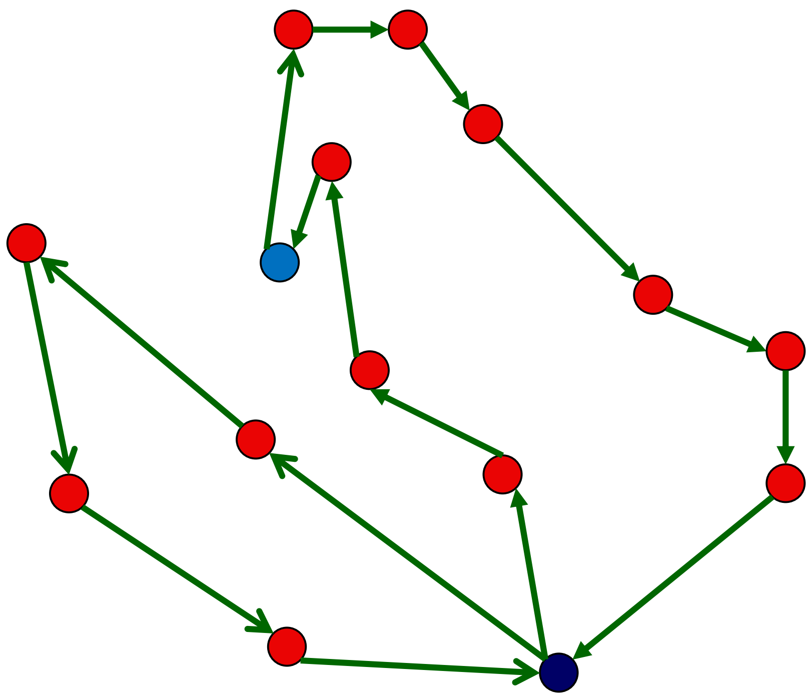


Improvement Methods

Local Search

•Often defined using an "operator"

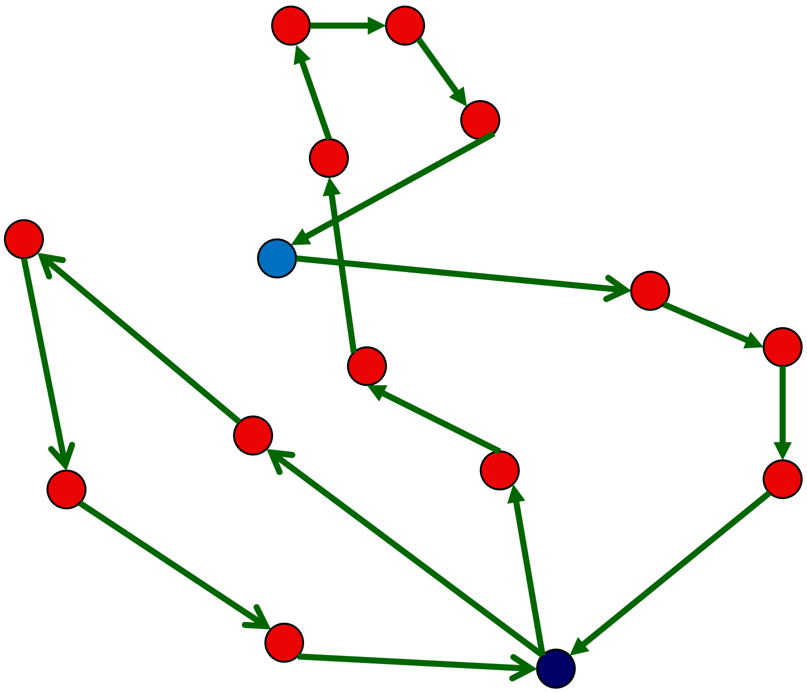
- e.g. 1-move



Improvement Methods

Local Search

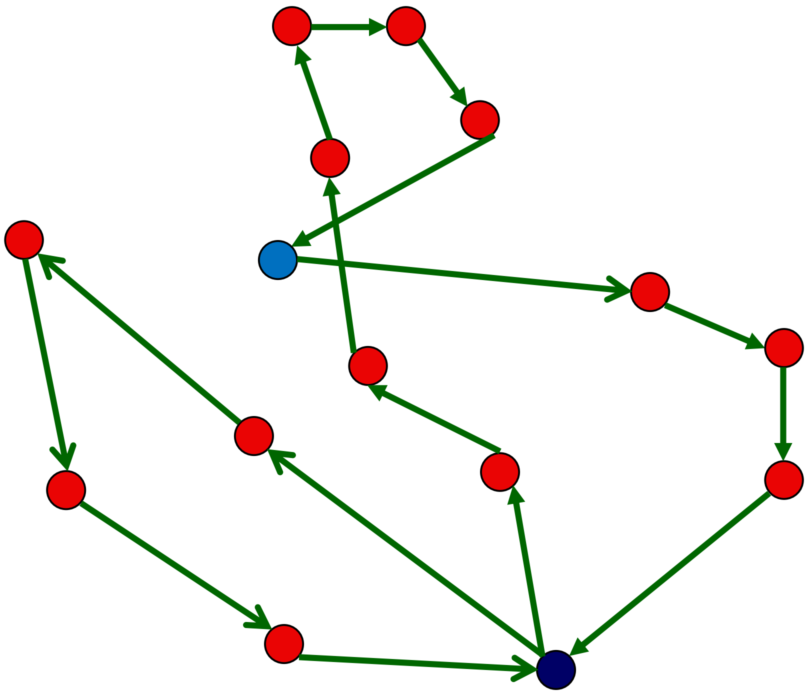
- Often defined using an “operator”
  - e.g. 1-move



Improvement Methods

Local Search

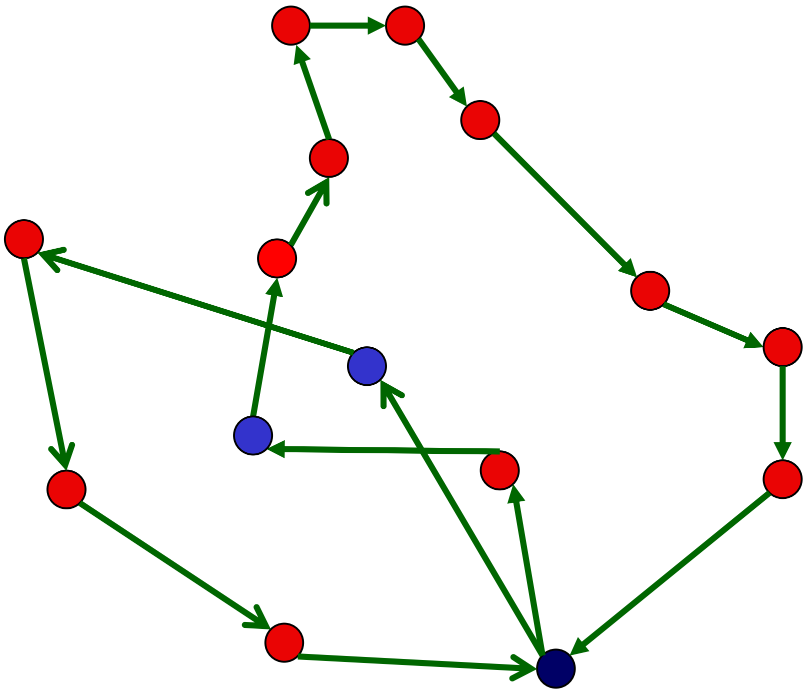
- Often defined using an “operator”
  - e.g. 1-move
- Operators determine the search "neighbourhood"
- Local Search explores the neighbourhood of
  the current incumbent solution



Local Search

Other Neighbourhoods for VRP:

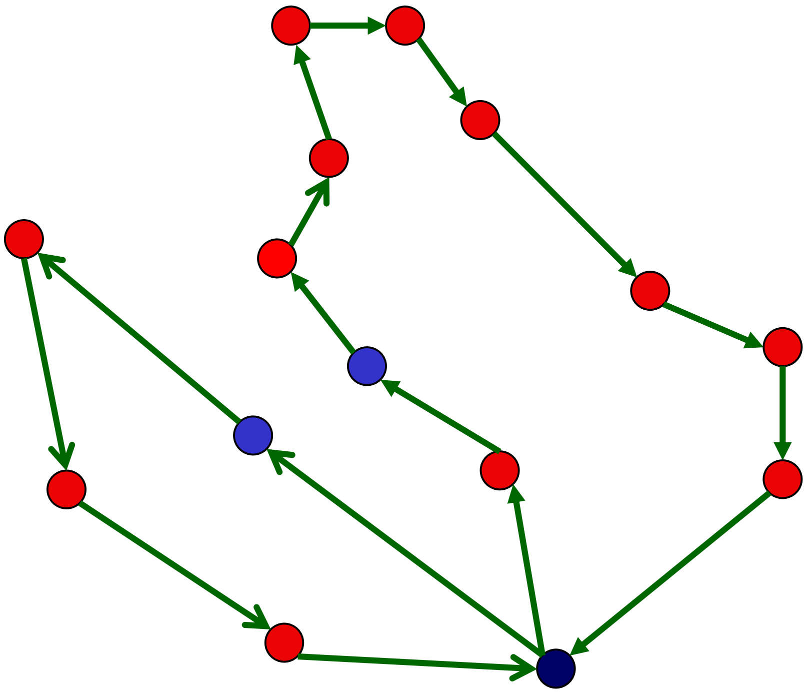
Swap 1-1



Local Search

Other Neighbourhoods for VRP:

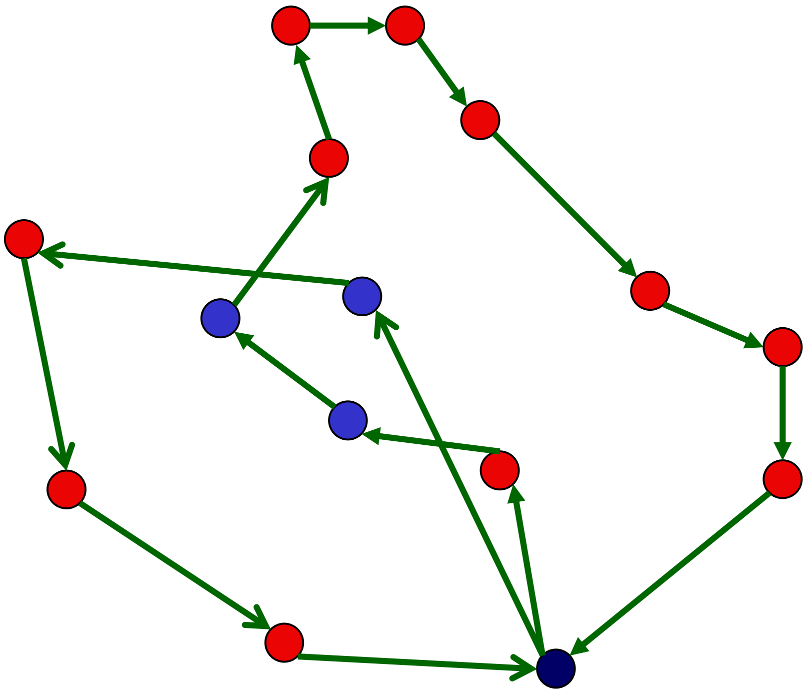
Swap 1-1



Local Search

Other Neighbourhoods for VRP:

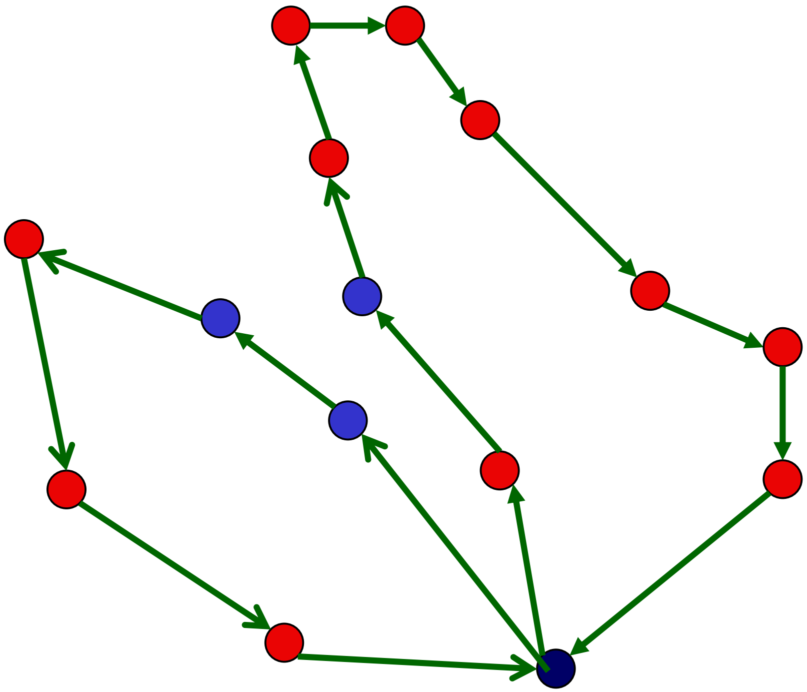
Swap 2-1



Local Search

Other Neighbourhoods for VRP:

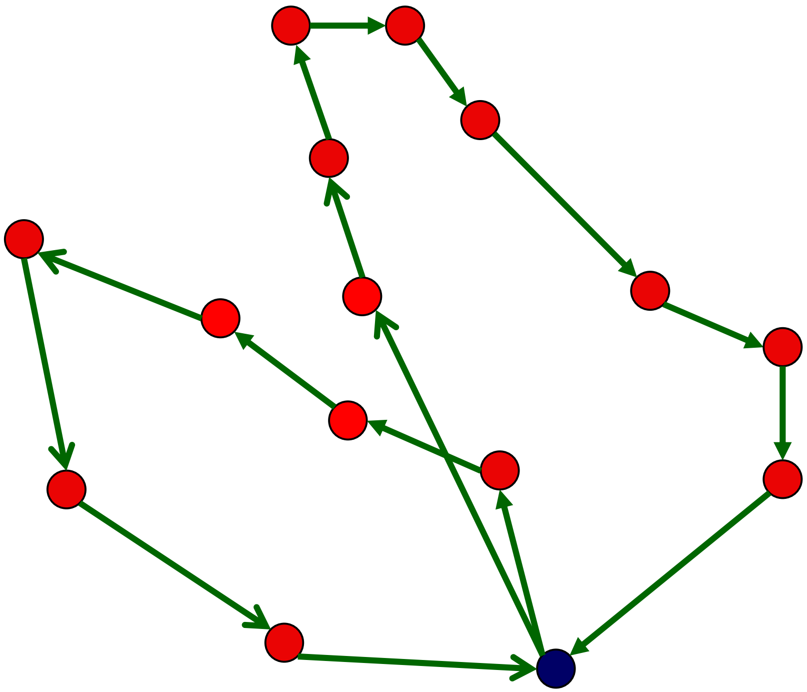
Swap 2-1



Local Search

Other Neighbourhoods for VRP:

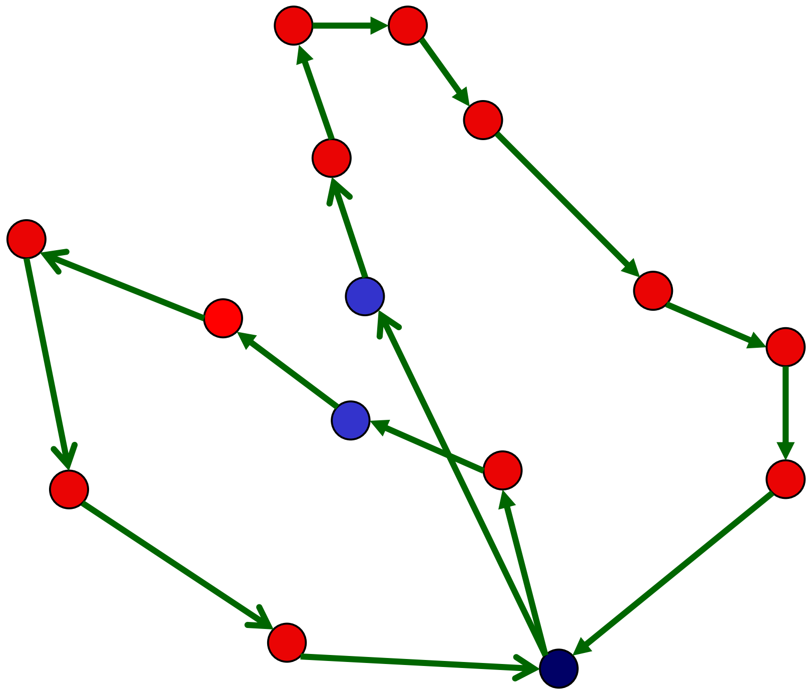
Swap tails



Local Search

Other Neighbourhoods for VRP:

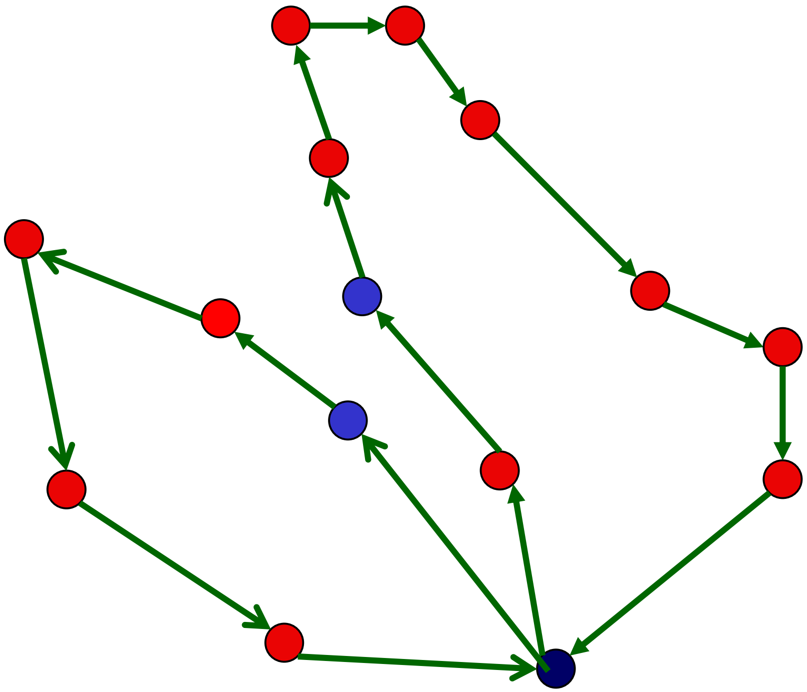
Swap tails



Local Search

Other Neighbourhoods for VRP:

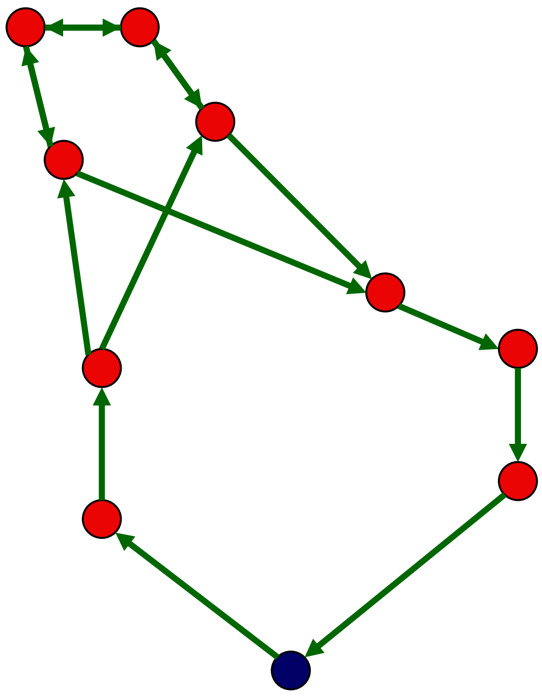
Swap tails



Improvement Methods

2-opt (3-opt, 4-opt…)

- Remove 2 arcs
- Replace with 2 others



Other problems

Neighbourhoods for other problems

Knapsack

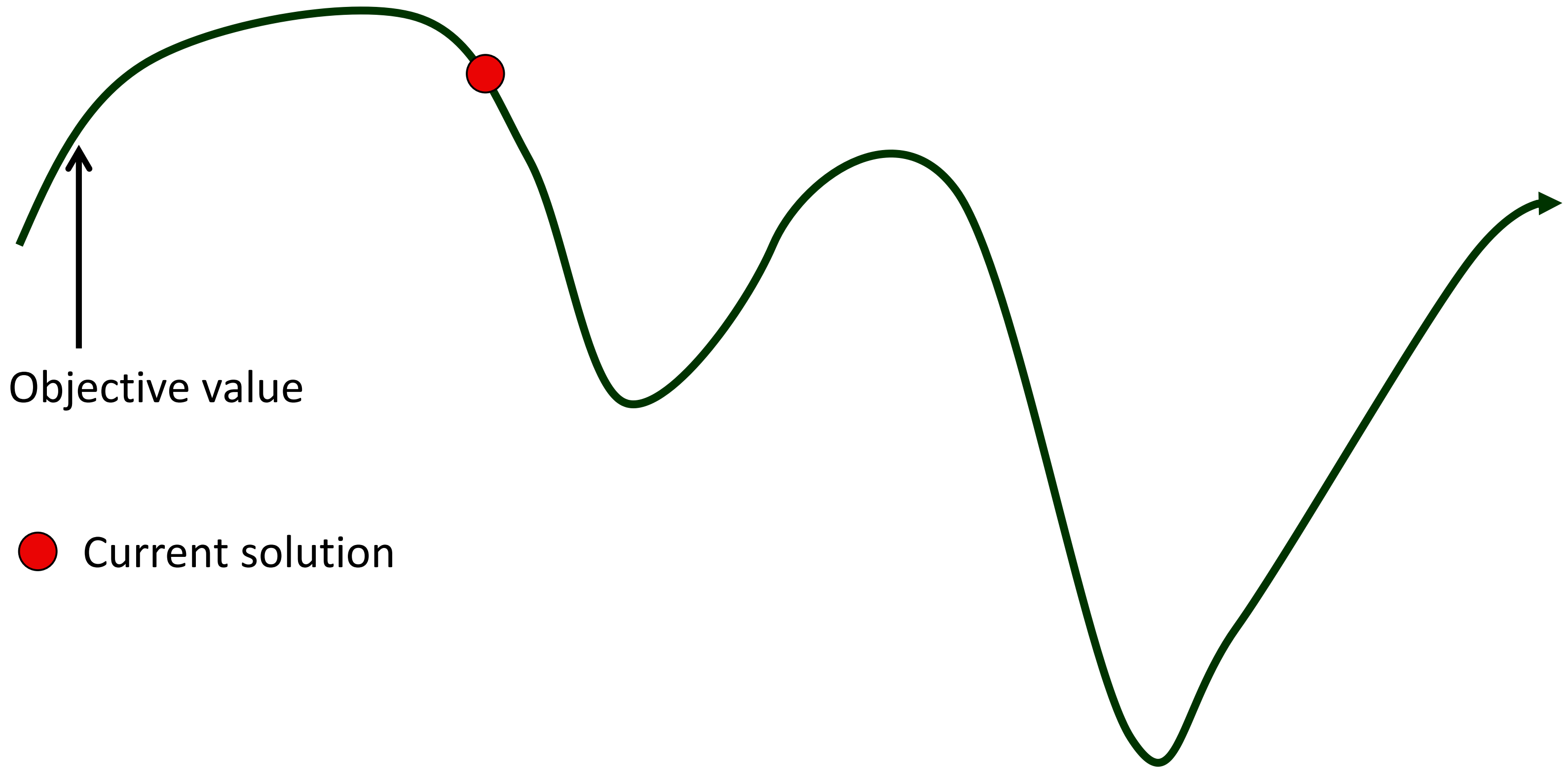
- Swap 2 items (in/out)
- Swap 1 item with multiple items of equal size

Scheduling

- Swap jobs between machines
- Swap order of jobs

Local Search

Local minima



Local Search

Escaping local minima

Meta-heuristics

- Heuristic way of combining heuristics
- Designed to escape local minima

Local Search

Escaping local minima

Define more (larger) neighbourhoods

- 1-move (move 1 visit to another position) — $O(n^2)$
- 1-1 swap (swap visits in 2 routes) — $O(n^2)$
- 2-2 swap (swap 2 visits between 2 routes) — $O(n^2)$
- 2-opt — $O(n^2)$
- 3-opt — $O(n^3)$
- Or-opt size 2 (move chain of length 2 anywhere) — $O(n^2)$
- Or-opt size 3 (chain length 3) — $O(n^2)$
- Tail exchange (swap final portion of routes) — $O(n^2)$

Local Search

Variable Neighbourhood Search

- Consider multiple neighbourhoods
  - 1-move (move 1 visit to another position)
  - 1-1 swap (swap visits in 2 routes)
  - 2-2 swap (swap 2 visits between 2 routes)
  - 2-opt
  - Or-opt size 2 (move chain of length 2 anywhere)
  - Or-opt size 3 (chain length 3)
  - Tail exchange (swap final portion of routes
  - 3-opt
- Explore one neighbourhood completely
- If no improvement found, advance to next neighbourhood
- When an improvement is found, return to level 1

Local Search

Tabu Search

- Find Local minimum
  - Explore each neighbourhood
  - If an improving move is found, make it
  - Repeat until local minimum is found
- Choose a cost-increasing move
- Make the cost-increasing move
- Make its reversal "tabu"
- Repeat
- Limit size of tabu list
- Bad moves are eventually reversed

Local Search

Simulated Annealing

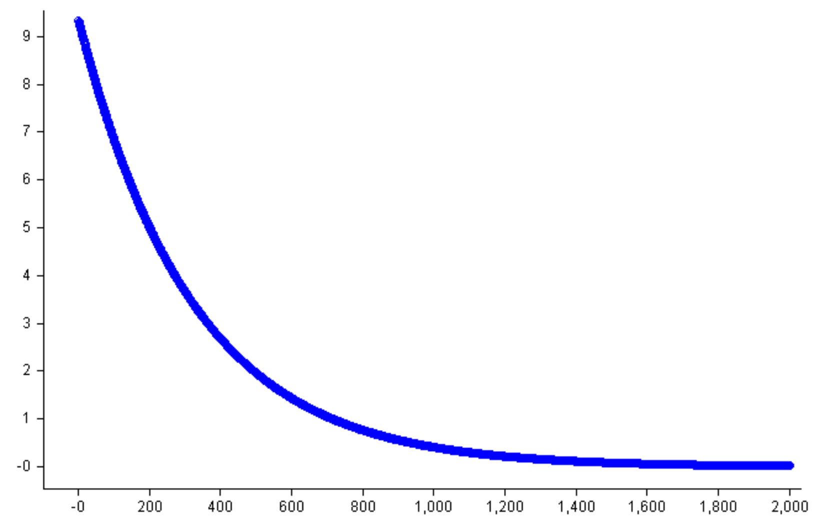
•Reflects "annealing" of a crystal

•Minimise energy in crystal ⟺ Minimise objective

•*"Temperature"* T controls

- How large an increase (Δ) will be accepted
- Probability of acceptance

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



Temperature

•*As* $T \downarrow 0$, only improving moves accepted

Local Search

Large Neighbourhood Search

= Destroy & Re-create

- Destroy part of the solution
  - Remove visits from the solution
- Re-create solution
  - Use favourite construct method to re-insert customers
- If the solution is better, keep it
- Repeat

Local Search

Genetic Algorithms

- •Generate a population of solutions (construct methods)
- •Evaluate fitness (objective)
- •Create next generation:
  - – Choose two solutions from population
  - – Combine the two (two ways)
  - – (Mutate)
  - – Produce offspring (calculate fitness)
  - – (Improve)
  - – Repeat until population doubles
- •Apply selection:
  - – Bottom half "dies"
- •Repeat

Local Search

.. and the whole bag of tricks

- Ants
- Bees
- Moths
- Particle Swarms
- ….

Review

Solving VRPs

- Exact
- Heuristic

Local Search

- “Neighbourhood”
- Neighbourhood-based local search
- Metaheuristics
  - Variable Neighbourhood Search
  - Large Neighbourhood Search
- Applied these to the VRP

Presenter’s Transportation Publications

- H. Aziz, C. Cahan, **C. Gretton, P. Kilby**, N. Mattei and T. Walsh. *A Study of Proxies for Shapley Allocations of Transport Costs*. Journal of Artificial Intelligence Research 56:573-611, 2016.
- H. Grzybowska, **C. Gretton, P. Kilby**, S. T. Waller. A Decision Support System for a Real-Time Field Service Engineer Scheduling Problem with Emergencies and Collaborations. Journal of the Transportation Research Board 2497:117-123*. 2015.*tificial Intelligence Research 56:573-611, 2016.
- **C. Gretton and P. Kilby**. A Study of Shape Penalties in Vehicle Routing. TRISTAN VIII, 2013.

Presenter's Local Search Publications

- D. Pham, J. Thornton, **C. Gretton**, and A. Sattar. *Combining Adaptive and Dynamic Local Search for Satisfiability*. Journal on Satisfiability, Boolean Model Checking, and Computation, 2008.
- S.Richter, M.Helmert and **C.Gretton**. *A Stochastic Local Search Approach to Vertex Cover*. Proceedings of the 30th German Conference on Artificial Intelligence (KI-2007), 2007.