Heuristics an overview

Patrick De Causmaecker
CODeS
Katholieke Universiteit Leuven
Belgium

Deel heuristieken
Patrick De Causmaecker

- Overzicht van metaheuristieken
  - Simulated annealing, Tabu Search, Variable Neighborhood Search, Hyperheuristieken,…
- Link met data
  - Parameter tuning, Algoritme configuratie, Stochastische kosten

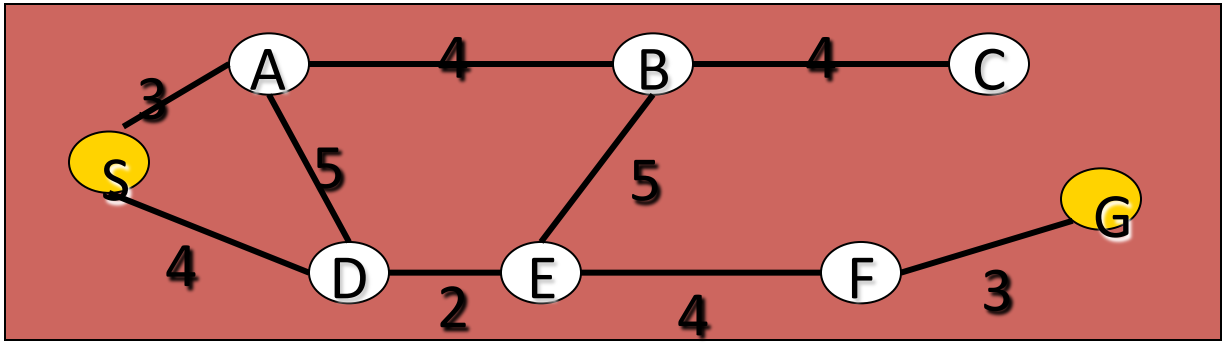
Content (lecture 1)

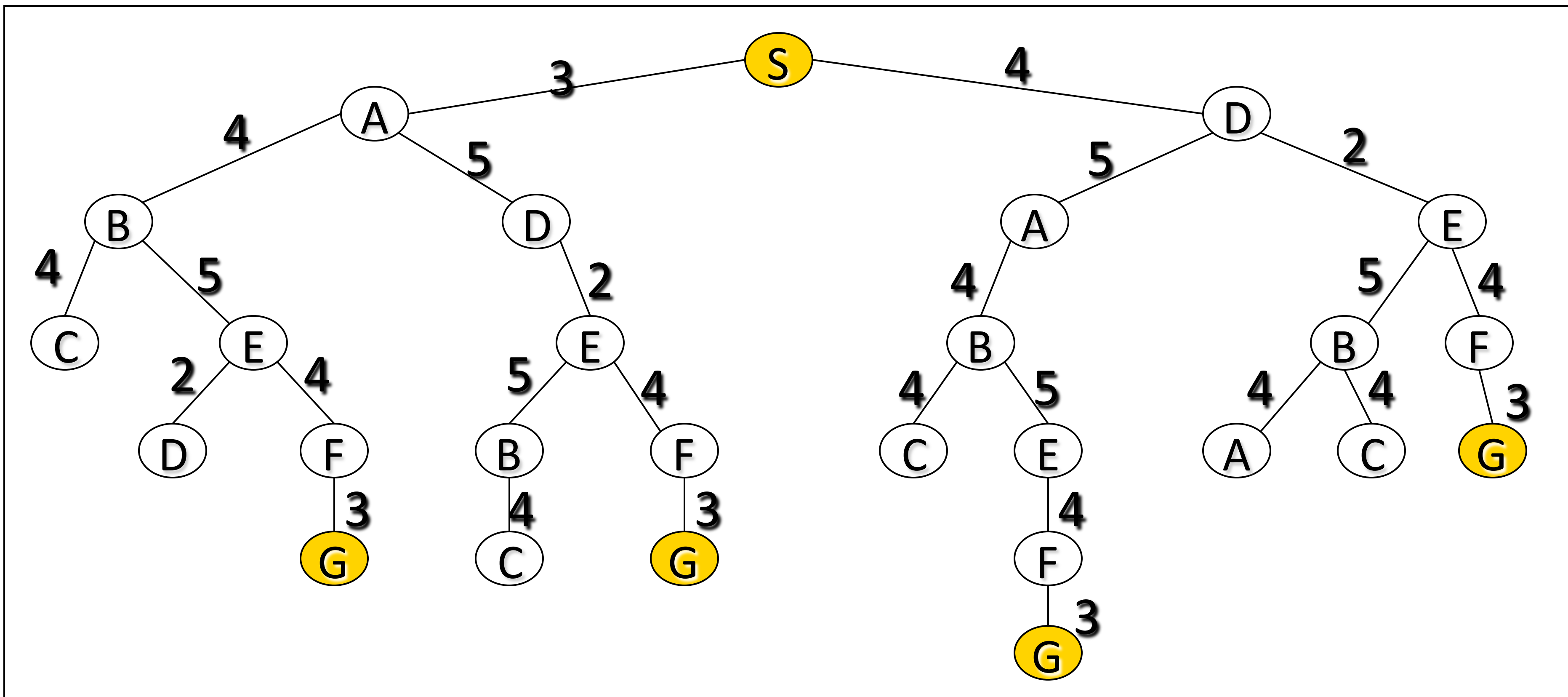
- An early example (what this talk is not about)
- What is may be about
- What it is about
- Genetic algorithms
- Tabu search
- Variable neighborhood search
- Hyper heuristics

Content (lecture 2)

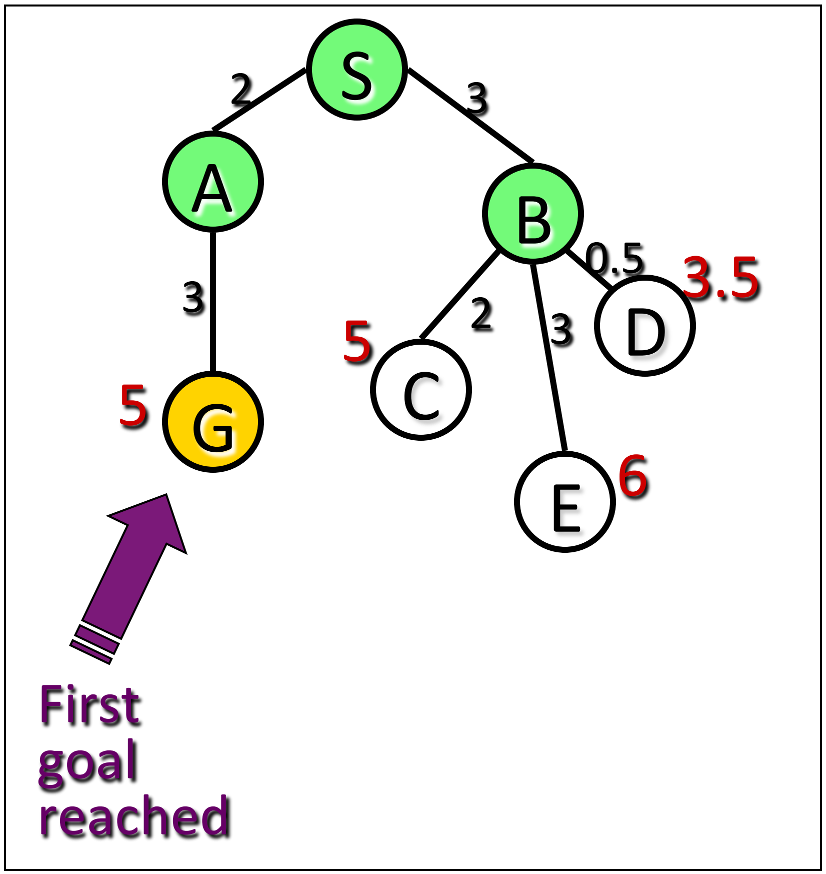
- Examples of real applications
- Detailed solution procedure

An early Example (what it is not about)



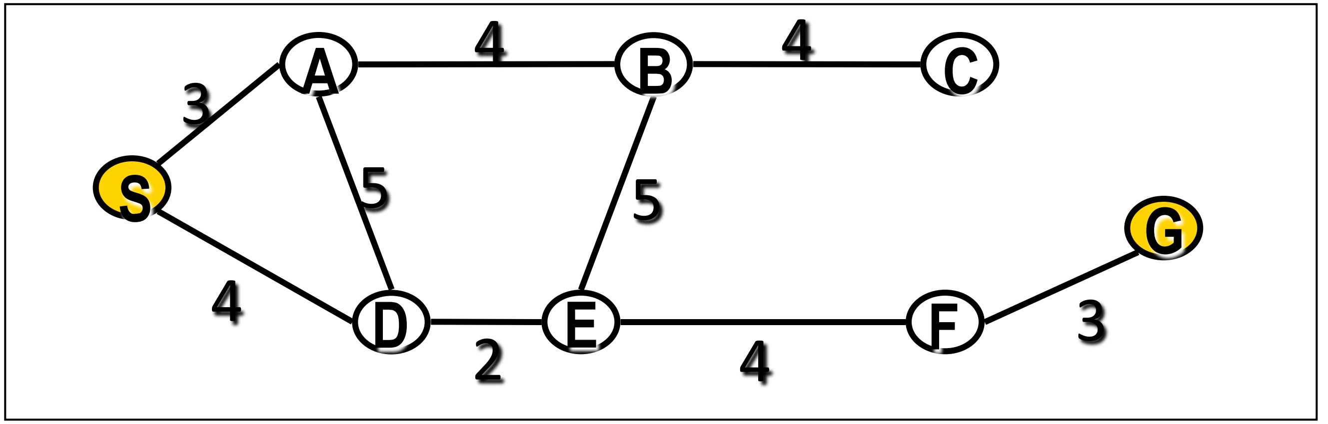


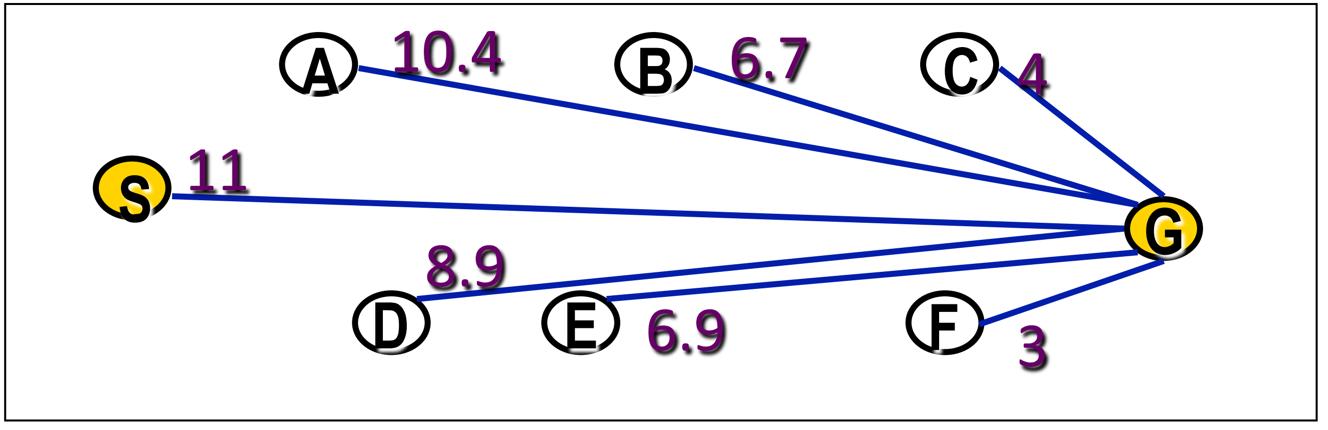
Branch and bound



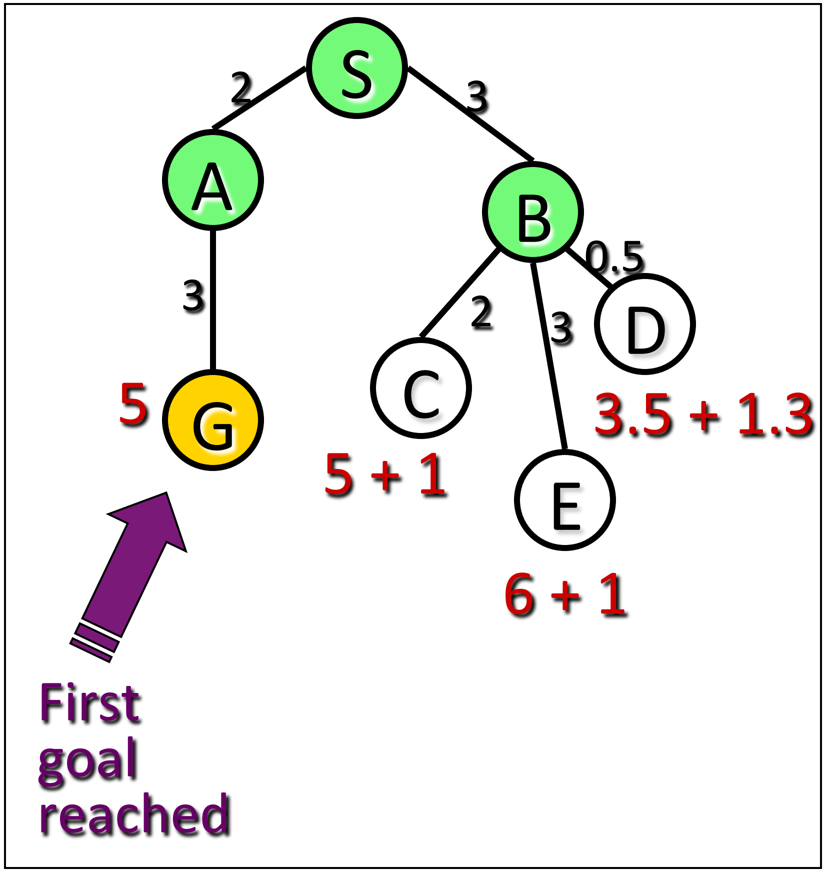
- Use any (complete) search method to find <u>a path</u>.
- Remove all partial paths that have an accumulated cost larger or equal than the found path.
- Continue search for the next path.
- Iterate.

Heuristic estimates





Search using heuristic estimates



- Use any (complete) search method to find <u>a path</u>.
- Remove all partial paths that have an ***estimated*** accumulated cost larger or equal than the found path.
- Continue search for the next path.
- Iterate.

- Finds the optimal path if the heuristic is an **underestimate**

Another example 8-puzzle

- f1(T) = the number correctly placed tiles on the board:

$$f1 \left(\begin{array}{|c|c|c|} \hline \boxed{1} & 3 & 2 \\ \hline \boxed{8} & & \boxed{4} \\ \hline 5 & \boxed{6} & 7 \\ \hline \end{array} \right) = 4$$

◎ **f2(T) = number of incorrectly placed tiles on board:**

➔ gives (rough!) estimate of how far we are from goal

$$f2 \left(\begin{array}{|c|c|c|} \hline 1 & \boxed{3} & \boxed{2} \\ \hline 8 & & 4 \\ \hline \boxed{5} & 6 & \boxed{7} \\ \hline \end{array} \right) = 4$$

Most often, 'distance to goal' heuristics are more useful !

Another example: 8 puzzle

- f3(T) = the sum of (the horizontal + vertical distance that each tile is away from its final destination):
  - gives a better estimate of distance from the goal node

$$f2 \begin{pmatrix} 1 & \boxed{3} & \boxed{2} \\ 8 & & 4 \\ \boxed{5} & 6 & \boxed{7} \end{pmatrix} = 1 + 1 + 2 + 2 = 6$$

Linear programming

$$\textit{maximize} \quad 3x_1 + 5x_2 + 4x_3$$

subject to

$$\begin{aligned}
7.5x_1 + 8x_2 + 4x_3 &\leq 10000 \\
12x_1 + 9x_2 + 8x_3 &\leq 18000 \\
3x_1 + 4x_2 + 2x_3 &\leq 9000 \\
x_1 &\geq 1000
\end{aligned}$$

General (Canonical) form

$$max(min) \sum_{i=1}^{n} c_i x_i$$

subject to

$$\sum_{i=1}^{n} a_{1i} x_i \sim b_1$$

$$\vdots$$

$$\sum_{i=1}^{n} a_{mi} x_i \sim b_m$$

$$x_i \geq 0$$

$$max\ CX$$

subject to

$$AX = b$$

$$X \geq 0$$

Duality

$$\max\ CX$$

subject to

$$\begin{aligned} AX &= b \\ X &\geq 0 \end{aligned}$$

$$\min\ b^T Y$$

subject to

$$\begin{aligned} A^T Y &\geq C^T \\ Y &\geq 0 \end{aligned}$$

Facility location: mixed integer problem (MIP)

$$
\begin{array}{rl}
min & 52.(25x_{11} + 20x_{12} + 15x_{13}) \\
+ & 25x_{11} + 20x_{12} + 15x_{13} \\
+ & 15x_{21} + 25x_{22} + 20x_{23} \\
+ & 20x_{31} + 15x_{32} + 25x_{33}) \\
+ & 500000.(y1 + y2 + y3)
\end{array}
$$

subject to

$$
\begin{aligned}
x_{11} + x_{12} + x_{13} &= 1000 \\
x_{21} + x_{22} + x_{23} &= 1000 \\
x_{31} + x_{32} + x_{33} &= 500
\end{aligned}
$$

$$
\begin{aligned}
x_{11} + x_{21} + x_{31} &\leq 1500y_1 \\
x_{12} + x_{22} + x_{32} &\leq 1500y_1 \\
x_{13} + x_{23} + x_{33} &\leq 1500y_1
\end{aligned}
$$

$$
\begin{aligned}
x_{ij} &\geq 0 \\
y_j &\in \{0, 1\}
\end{aligned}
$$

Relaxation

- Remove integer constraint from the MIP

$$y_j \in \{0, 1\}$$

- The new problem (LR) is 'easier', a (linear) relaxation
  - Optimal value is 'better'
  - LR infeasible -> MIP is infeasible
  - Optimal solution of LR with integer values is optimal for MIP
  - Floor/ceiling property (min problem with integer c)

$$\lceil LR \rceil \leq MIP$$

Optimum of LR for the facility location

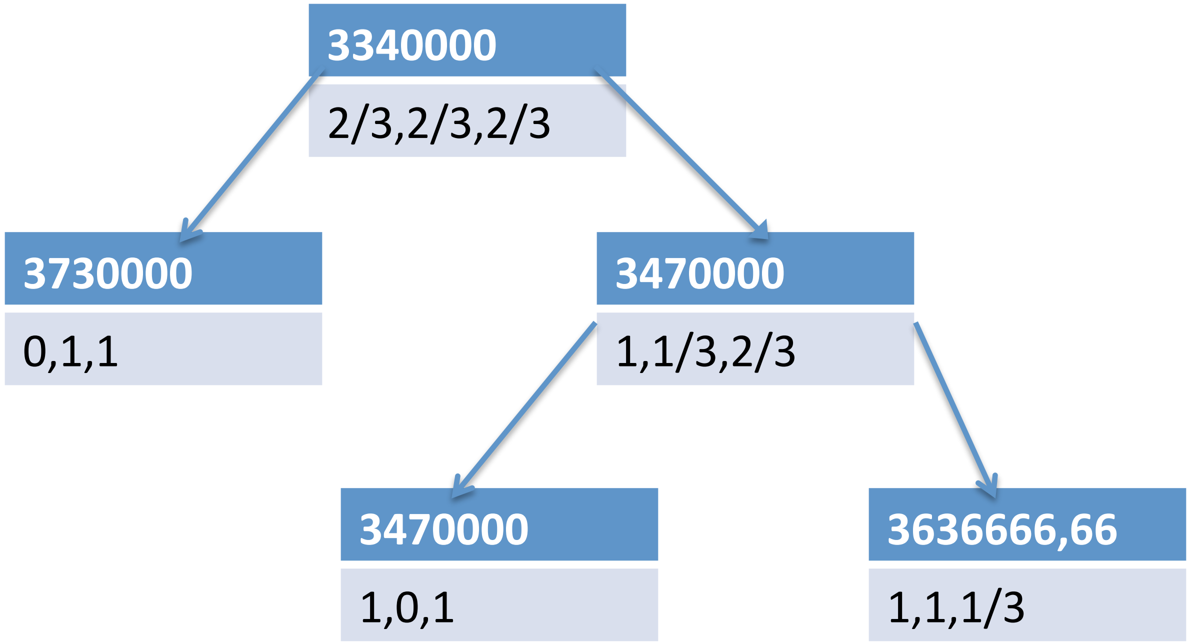
$$y_1 = \frac{2}{3} = y_2 = y_3$$

$$Optimum = 3340000$$

$$Round\ to\ y_1 = 1 = y_2 = y_3 (feasible!) \rightarrow 3840000$$

- Rather good, but is it optimal?
- Hence: branching: $IP(y_1 = 1)$ & $IP(y_1 = 0)$
- This has lead to a rich research area with remarkable advances in both theory and practice.

Full tree



3340000
2/3,2/3,2/3
3730000
0,1,1
3470000
1,1/3,2/3
3470000
1,0,1
3636666,66
1,1,1/3

Heuristics in complete search

- Bring in **domain/expert** knowledge
- More accurate heuristics deliver better
- Reduce the search time/do not reduce worst case complexity

Combinatorial problems

- Hamilton cycle
- Set covering
- Knapsack
- Assignment
- Bin packing
- Scheduling

- Travelling salesperson
- Knapsack
- Assignment
- Max clique
- Shortest path
- Bin packing
- Vehicle routing
- Scheduling

Content

- An early example (what this talk is not about)
- **What it may be about**
- What it is about
- Genetic algorithms
- Tabu search
- Variable neighborhood search
- Hyper heuristics

Heuristic search

- Nearest neighbor for TSP and Hamilton
- New-best-in heuristic for max clique
- Best fit for bin packing
- Worst fit for bin packing
- Earliest due date for scheduling
- Shortest job for scheduling
- …

Complex? Complete/Optimal?

- Fast: typically O(n^2)
- Not exactly optimal, e.g. TSP with NN:
  - N cities randomly distributed on a plane:
    - average = 1.25 * exact_shortest_length
  - Special constructions
    - Worst route can be found
- If the triangle inequality is satisfied:
    - NN: *O(log|V|) (Rozenkrantz, 1977)*
    - Christofides algorithm guarantees 1.5*exact_shortest_length

Are exact/complete solutions feasible? Take TSP

- Enumeration (*$O(n!)$*)
- Dynamic programming (*$O(n^2*2^n)$*) *(exponential space)*
- Branch and bound (40-60 cities)
- Lagrangian relaxation (100)
- Linear/integer programming (200)
- Benchmarks: TSPLIB
  - 15112 German towns: 22.6 years, 100 processors, 2001
  - 24978 Swedish cities: 2004
  - Recent: 33810 (2005), 85900 (2006) cities…

Live with (incomplete/non exact) algorithms/heuristics
But there are some problems

- Problem specific:
  - No generality
  - Application to real world problems?
- Improvement
  - Heuristics often run in a fixed time,
    - giving it more time does not result in better results
  - Stochastic behavior delivers better results faster.
    - Heuristics are often deterministic in nature. If there is a stochastic element, it is localized and it is hard to manipulate its impact

Content

- An early example (what this talk is not about)
- What it may be about
- **What is it about**
- Genetic algorithms
- Tabu search
- Variable neighborhood search
- Hyper heuristics

Move from problem specific heuristics to general heuristic schemata

- 1986: ‘metaheuristic’ (Fred Glover)
- **Recipe** to build a heuristic
- developer can insert problem specific information at certain points
- Attempt to find a **general** approach that works for a large set of problems
- Bears on general principles and ideas
  - Sometimes with hard mathematical underpinning
- Can be considered **designer guidelines**

Some examples

- **Genetic algorithms**
- Evolutionary algorithms
- **Tabu search**
- **Simulated annealing**
- Great deluge
- **Hyperheuristics**
- **Variable neighbourhood search**
- Late acceptance
- Extreme optimisation
- Neural networks
- Electrostatic potential

METAPHORS

Content

- An early example (what this talk is not about)
- What is may be about
- What it is about
- **Genetic algorithms**
- Tabu search
- Variable neighborhood search
- Hyper heuristics

Genetic algorithms

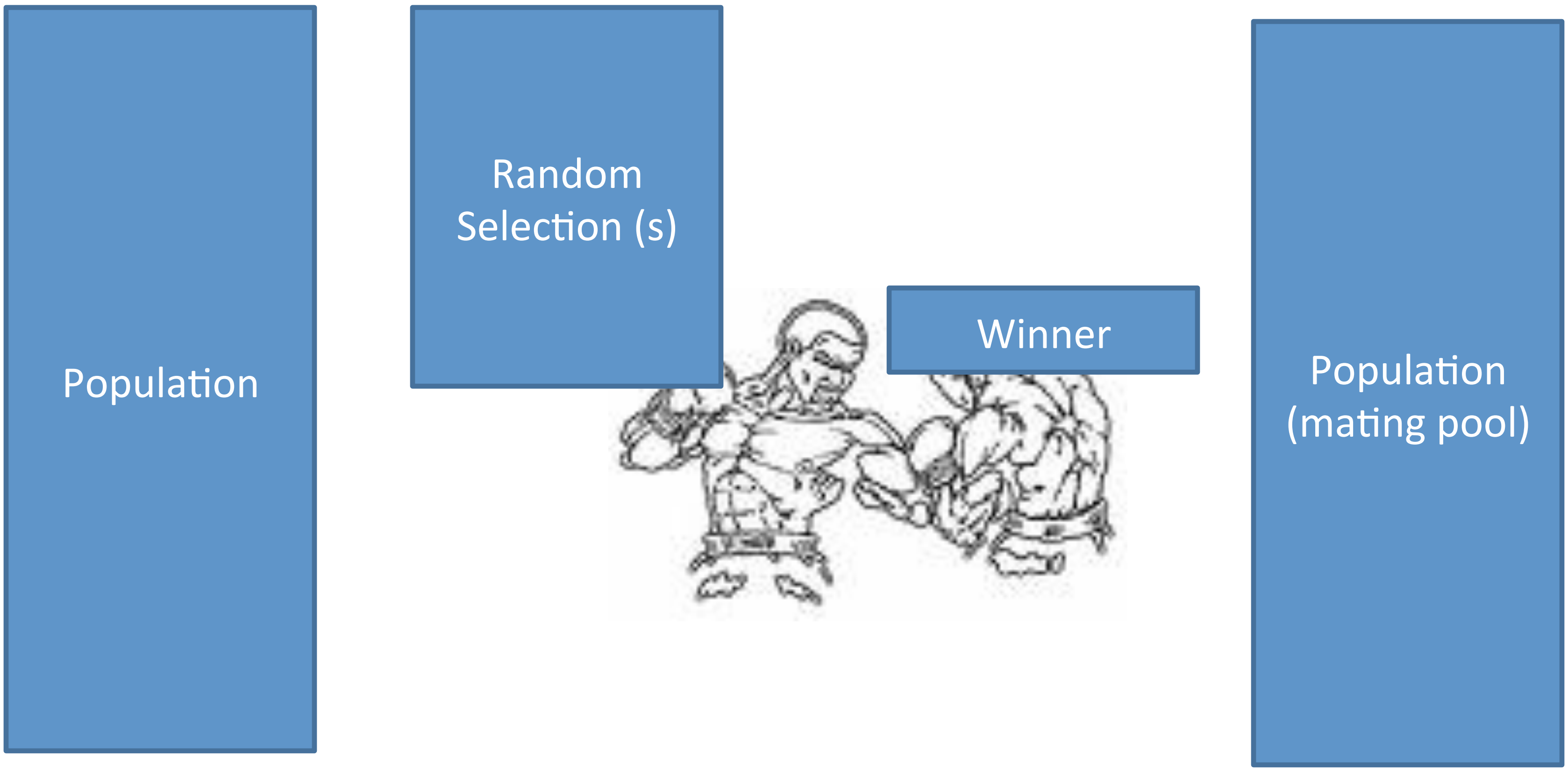
- **Initialisation**: Set up a population of individuals
  - Random across search space or guided by domain knowledge
- **Evaluation**:
  - fitness values of the candidate solutions
- **Selection**:
  - favour the better solutions
- **Recombination**:
  - combine parts of solutions
- **Mutation**:
  - random changes in each recombination
- **Replacement**:
  - offspring replaces parental

Genetic algorithms

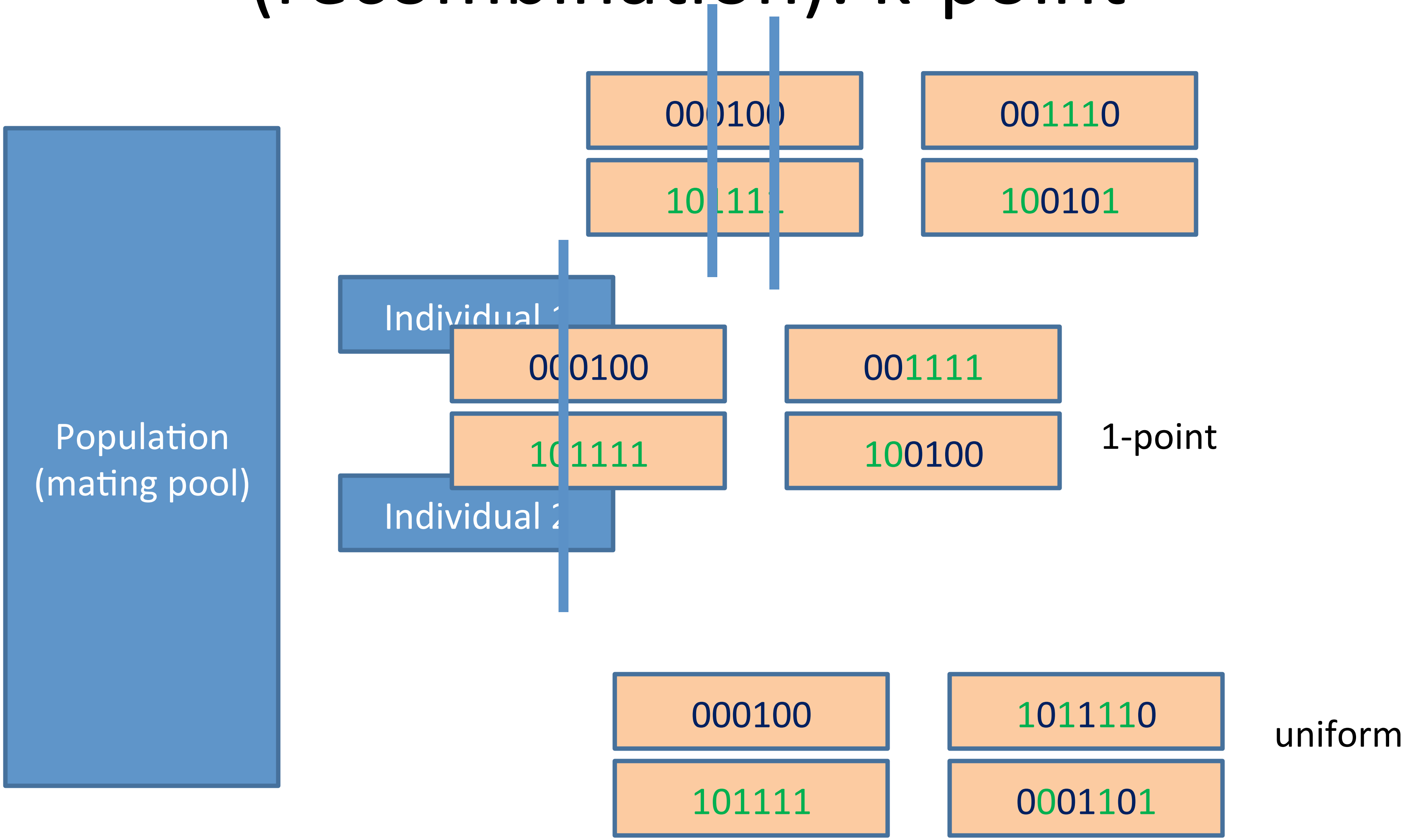
- Selection
  - Fitness proportional, e.g. biased roulette wheel

Genetic algorithms

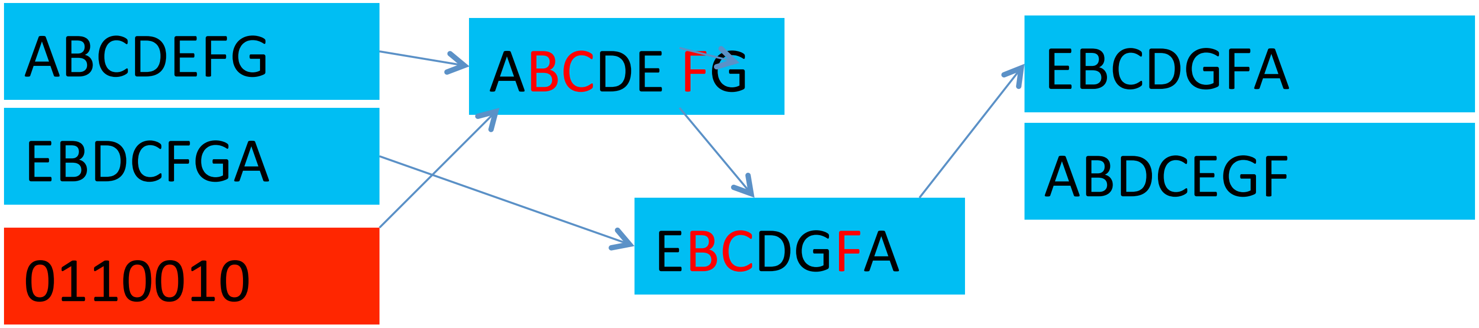
- Tournament selection



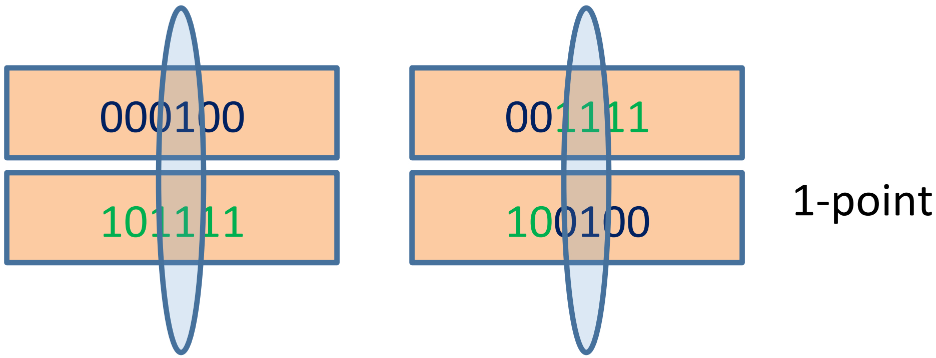
Genetic algorithms: Crossover (recombination): k-point



Genetic algorithms: crossover permutations: uniform order based



Genetic algorithms
Mutation



Some genes never change

Diversity

Mutation (at a low probability when compared to crossover)

Flipover or problem specific

Genetic algorithms
Replacement

- **Delete all**
  - Remove the whole population and replace it by the offspring
- **Steady-state**
  - Delet n old members and replace them with offspring members
  - Best, parents,...
- **Steady-state no duplicates**
  - Better coverage at a computational cost

Genetic algorithms other topics

- **Hybridization**
  - **Combing** GA with other technique, e.g. local search
    - Produce stronger results
    - **Memetic** algorithms
  - Repair, initialisation, case based memory, heuristics
- **Applications: ample**
  - Machine scheduling, electrical power systems, sports scheduling, nuclear power plants, airline scheduling

Genetic algorithms: tricks

- **Off the shelf**
- **Software package** (GA-LIB)
- **Representation**
  - (bit-representation <-> complex data structures)
- **Minimum**: evaluation function
  - Problem specific data (intelligent crossover operators, heuristic initial population)
- **Hybridize**

Genetic algorithms: parameters

- Population size
- Mutation probability
- Crossover probability
  - -> Experiments (use statistics and ev. experimental design)
  - -> consider dynamic parameter settings (in fact introduces extra parameters)
  - -> Go to the literature

Some examples

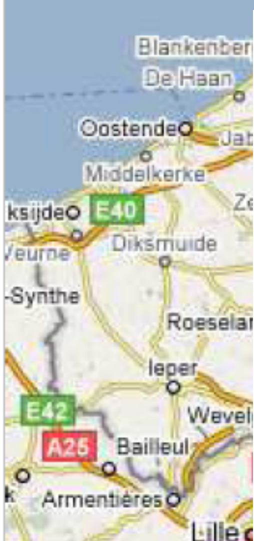
- **Genetic algorithms**
- Evolutionary algorithms
- **Tabu search**
- **Simulated annealing**
- Great deluge
- **Hyperheuristics**
- **Variable neighbourhood search**
- Late acceptance
- Extreme optimisation
- Neural networks
- Electrostatic potential

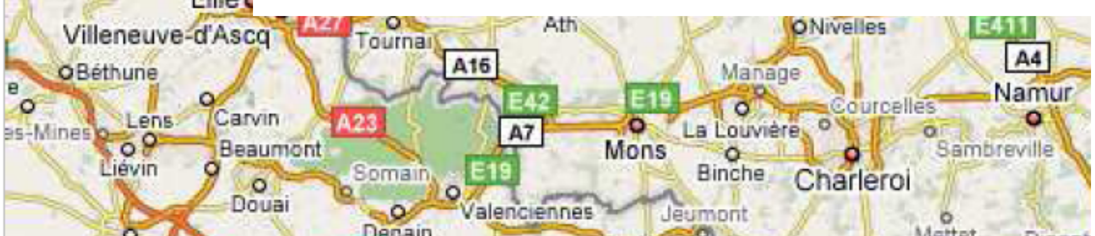
METAPHORS

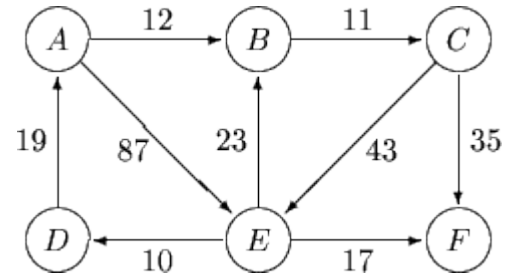
Tabu Search: background

- **Search space** / landscape
- Heuristic **moves** through the landscape
- **Hill climber**
  - tries moves in the direction of (the fastest/some) improvement
- **local optimum**
  - **Hill climber** and has no way of **escaping** it
- 1983: simulated annealing (Kirkpatrick et al.)
  - 1986 tabu search (Glover), 1992 ant systems (Dorigo), 1975 genetic algorithms (Holland)
  - > field of metaheuristics (Glover)

Example problem







- *Given a weighted graph, (V,E), and two vertices, 'source' and 'destination', find the path from source to destination with the smallest accumulated weight.*
- Dijkstra: $O(|E|^2 + |V|)$ (for sparse graphs $O(|E|\log|V|)$)

Optimising what?

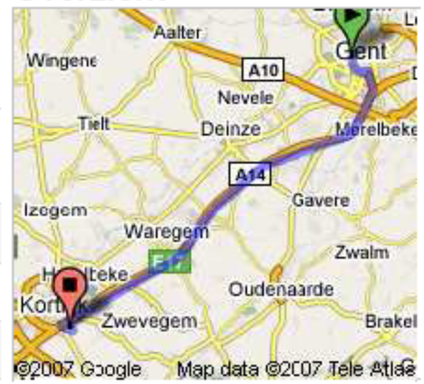
Bargiekaai 1, 9000 Gent naar Etienne Sabbelaan 53, 8500 Kortrijk - ...   http://maps.google.be/maps?f=d&hl=nl&geocode=&saddr=bargiekaa...

Beginpunt **Bargiekaai 1**
9000 Gent
Eindpunt **Etienne Sabbelaan 53**
8500 Kortrijk
Rei **48.9 km – ca 34 min.**

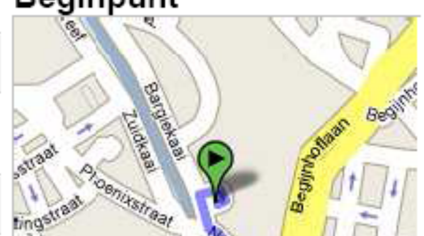
Bargiekaai 1
9000 Gent
Met de auto: 48.9 km – ca. 34 min.

1. Vertrek in **noordoostelijke** richting op de **Bargiekaai**	67 m
2. Sla **linksaf** om op de **Bargiekaai** te blijven	58 m
3. Sla **linksaf** bij **Noordstraat**	30 m
4. Flauwe bocht naar **rechts** bij **Waldamkaai**	0.1 km
5. Ga verder op **Coupure Rechts**	0.6 km 2 min.
6. Sla **rechtsaf** bij **Papegaaistraat**	50 m
7. Sla **linksaf** bij **Coupure Links**	1.0 km 2 min.
8. Sla **linksaf** bij **Nederkouter**	0.1 km

Overzicht

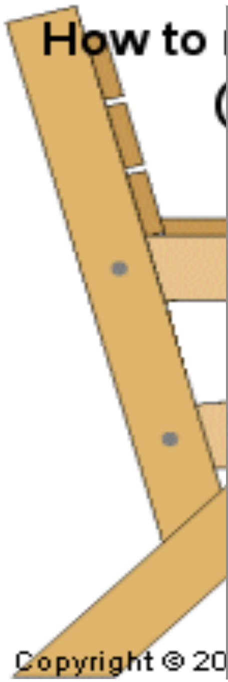


Beginpunt

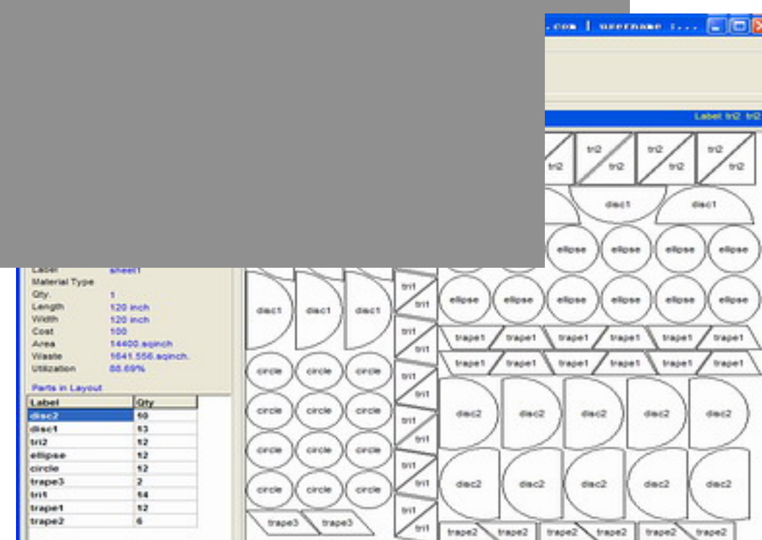


Stock cutting

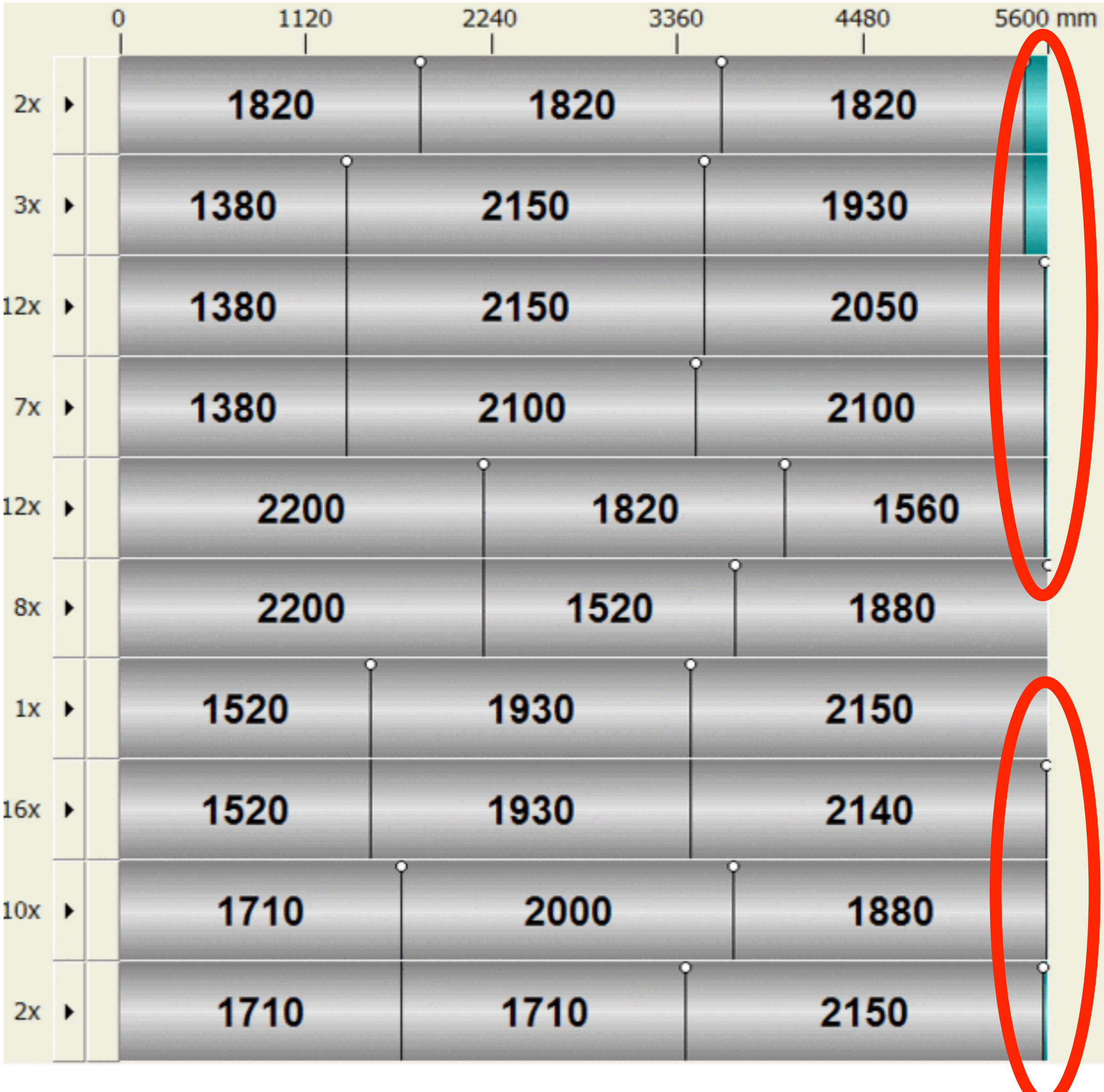
X 10000



- *NP-complete, even in one dimension (pipe-cutting)*
- *Integer programming, branch and bound, dynamic programming,…*
- *Heuristics apply*
  (E.K.Burke, G.Kendall and G.Whitwell, A New Placement Heuristic …, Operations Research Volume 52 Number 4, 2004)



Optimising what?



Optimising what?

- Goal function: measure of optimality
  - -> (only) guide through the space
- Shortest path problem
  - Find the path between source and destination of minimum total weight
- Stock cutting problem
  - Find the layouts that minimise the waste produced

Another example

- The Employee Timetabling Problem (ETP)
  - *An ETP consists of assignments of employees to working shifts while satisfying all constraints*
  - *Constraints:*
    - ***Shifts have start times and end times***
    - ***Employees have a qualification***
    - ***A required capacity per shift is given per qualification***
    - ***Employees can work subject to specific regulations***
    - ***…***

Shift	1	2	3	4	5	7	1	2	3	C
Pjotr	A	A	A			C	T	T	F	1
Ludwig	C	C			R	R	T	T	T	0
Clara	T	T	C		R	R	F	T	T	1
Hildegard			A	A	A		T	T	T	0
Johann			C	C			T	T	F	1
Wolfgang		C	T	T	C		T	T	T	0
Guiseppe	R	R			A	A	F	T	T	1
Antonio	R	R			C	C	F	T	T	1
Arranger	2	2	2	1	0	0				
Tonesetter	1	2	1	1	0	0				
Composer	1	1	1	1	2	0				
Reader	3	1	1	1	2	0				

ETP: Optimise what?

- Just solve the problem (CP)
- (Weighed) number of constraints violated
- Weighted number of constraints violated
- Amount under assignment
- Amount over assignment
- This may lead to the definition of a **goal function** (representing a lot of **domain** information)

Shift	1	2	3	4	5	7	1	2	3	C
Pjotr	A	A	A			C	T	T	F	1
Ludwig	C	C			R	R	T	T	T	0
Clara	T	T	C		R	R	F	T	T	1
Hildegard			A	A	A		T	T	T	0
Johann			C	C			T	T	F	1
Wolfgang		C	T	T	C		T	T	T	0
Guiseppe	R	R			A	A	F	T	T	1
Antonio	R	R			C	C	F	T	T	1
Arranger	2	2	2	1	0	0				
Tonesetter	1	2	1	1	0	0				
Composer	1	1	1	1	2	0				
Reader	3	1	1	1	2	0				

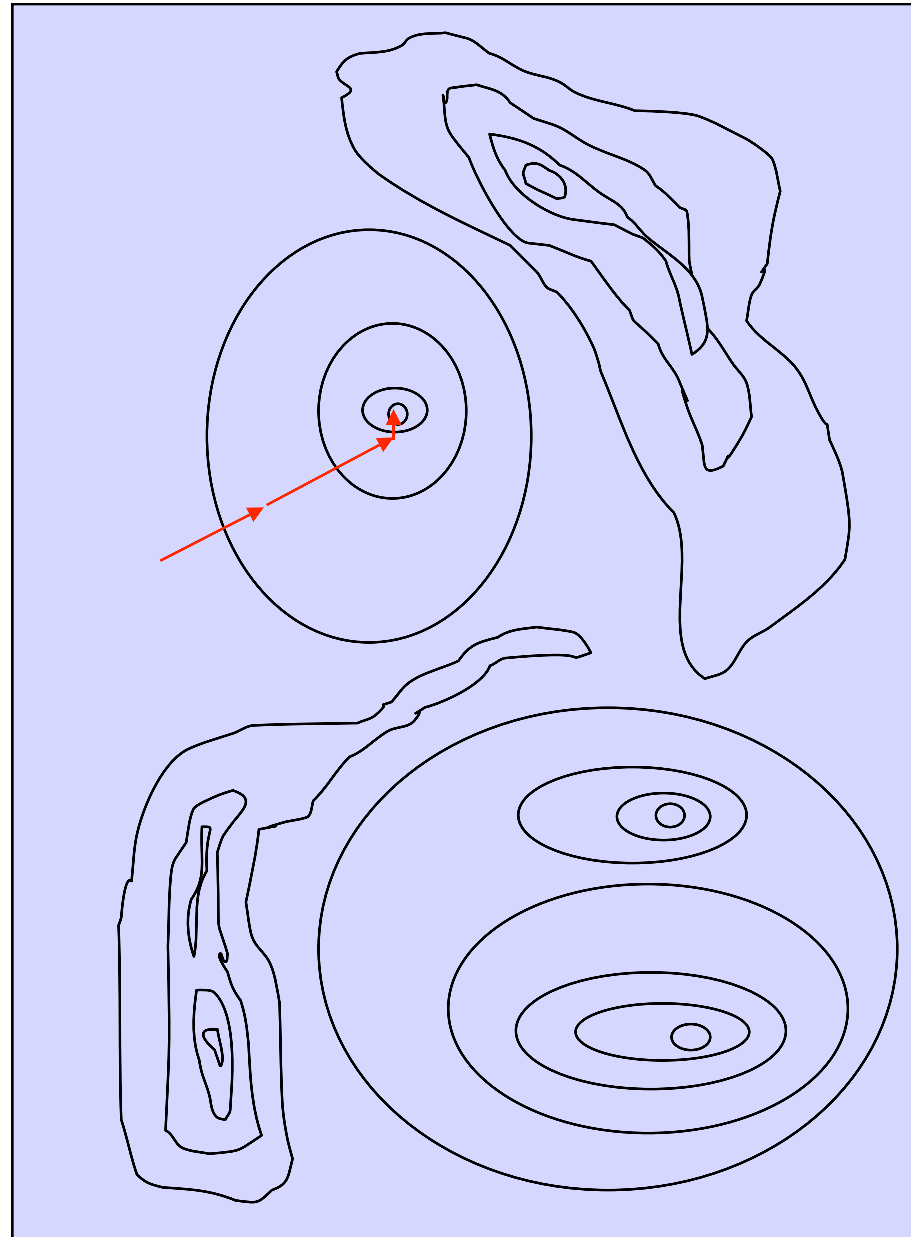
Heuristics for ETP

- One can easily think of
  - Swaps
  - Removals
  - Insertions
  - ‘Large Swaps’
- These ‘easy’ options do depend on the **domain**
- They define ‘steps’ in a ‘solution space’ with an associated change in the goal function.
- One obvious heuristic is a **hill-climber** based on a selection of these possible steps.

Shift	1	2	3	4	5	7	1	2	3	C
Pjotr	A	A	A			C	T	T	F	1
Ludwig	C	C			R	R	T	T	T	0
Clara	T	T	C		R	R	F	T	T	1
Hildegard			A	A	A		T	T	T	0
Johann			C	C			T	T	F	1
Wolfgang		C	T	T	C		T	T	T	0
Guiseppe	R	R			A	A	F	T	T	1
Antonio	R	R			C	C	F	T	T	1
Arranger	2	2	2	1	0	0				
Tonesetter	1	2	1	1	0	0				
Composer	1	1	1	1	2	0				
Reader	3	1	1	1	2	0				

Local search based metaheuristics

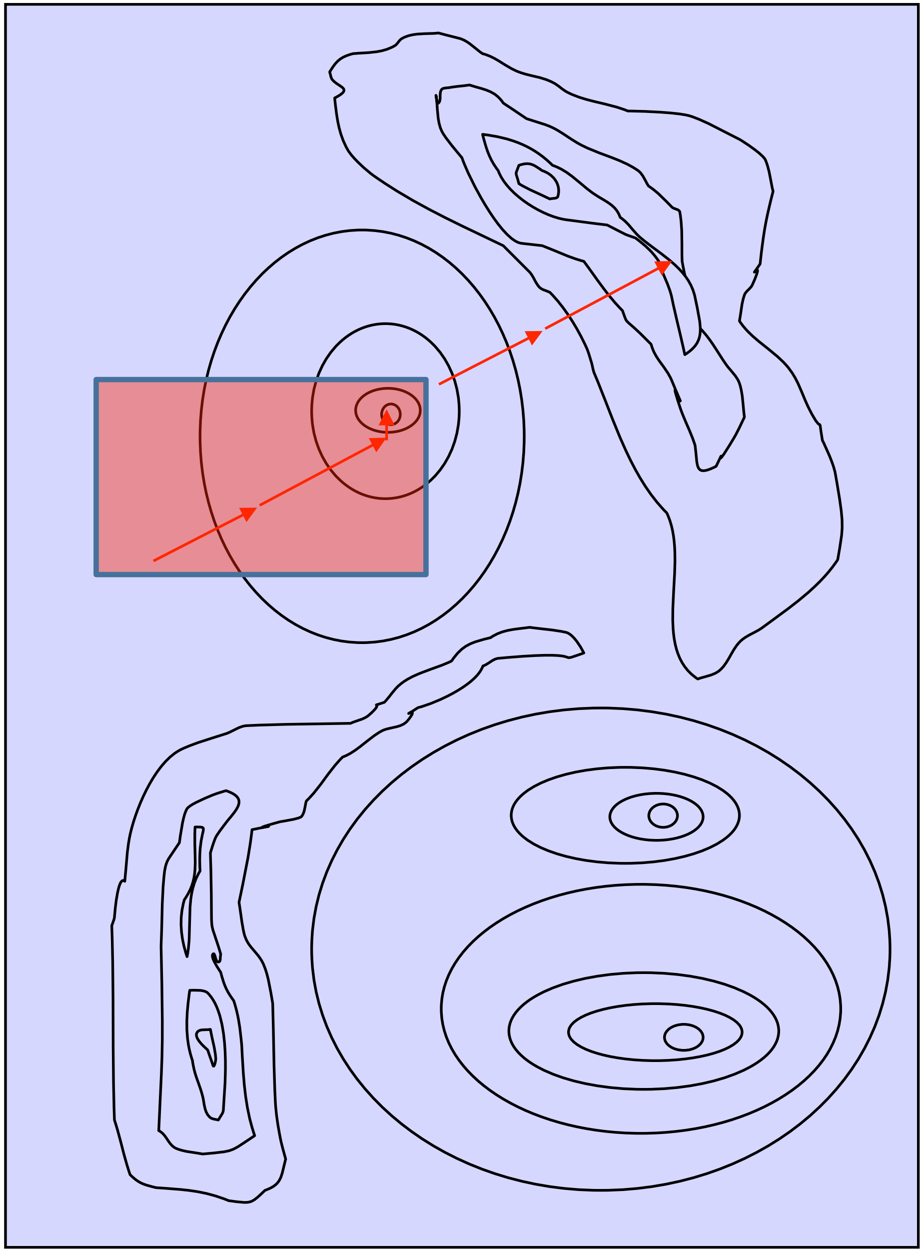
- Hill-climbing allows to introduce domain specific elements through
  - The goal function
  - The selection of the steps
- It describes a procedure to use these elements in a procedure that eventually produces an answer
- It does not guarantee optimality
- This is a local search heuristic



How to escape: Tabu

- While moving through the space, regions become forbidden
- This allows to escape from the local
- After a while, we may revisit

- Tabu supports diversification of the search
- Local search intensifies
- These phases alternate



Example ETP

- Initialise to satisfy the required amounts
- Allow only vertical swaps *(neighbourhood)*
- If a swap has influenced a certain region of the timetable, do not allow any other swap to influence this region for a specified number of moves *(Tabu list, Tabu attributes)*
- The search has now been parametrised

Shift	1	2	3	4	5	7	1	2	3	C
Pjotr	A	A	A				T	T	F	1
Ludwig	C	C			R		T	T	T	0
Clara	T	T	R		R		F	T	T	1
Hildegard	A		A	A			T	T	T	0
Johann		A	C	C			T	T	F	1
Wolfgang	R	T	T	T	C		T	T	T	0
Guiseppe	R						F	T	T	1
Antonio	R	R		R	C		F	T	T	1
Arranger	2	2	2	1	0	0				
Tonesetter	1	2	1	1	0	0				
Composer	1	1	1	1	2	0				
Reader	3	1	1	1	2	0				

Tabu search: tricks

- We introduced a number of parameters
- -> tuning
  - Neighbourhoods
  - List size
  - Aspectratio
  - Diversification/intensification
- Tabu search allows to introduce domain information through
  - the solution approach
  - the steps
  - the neighborhoods
  - the Tabu attributes

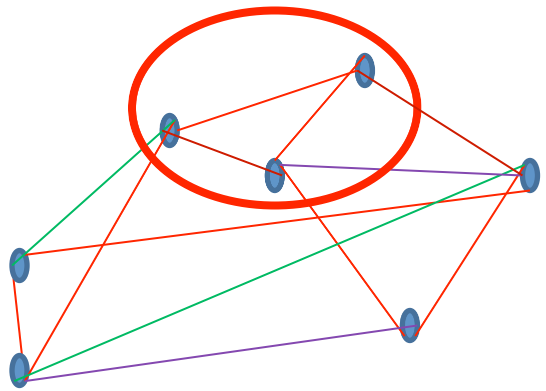
benchmarks

Important sector requiring a lot of effort

Examples

- Nurse rostering
- Multimodal transportation
- Data mining
- Clustering
- Gene prioritisation
- Tourist guide

Another example: another neighborhood: 2-change in TSP



- (n-1)! or (n-1)!/2 Routes
- 2-change connects them all
  - -> connectedness of search space
- Lin, Lin and Kernighan: k-opt

Some examples

- **Genetic algorithms**
- Evolutionary algorithms
- **Tabu search**
- **Simulated annealing**
- Great deluge
- **Hyperheuristics**
- **Variable neighbourhood search**
- Late acceptance
- Extreme optimisation
- Neural networks
- Electrostatic potential

METAPHORS

Annealing

- Solid state physics
  - Increase the temperature above melting point
  - Decrease carefully the temperature until the particles arrange in the gound state
- Ground state: **minimal** energy
- **1953** (Metropolis)
  - Simulate annealing in **materials** by Monte Carlo techniques

Simulated annealing

- Use **perturbation** to generate a new state *j* (energy E_j) from an existing state I (energy E_i)
- If $E_j - E_i < 0$ accept state *j*
- If $E_j - E_i > 0$ accept state *j* with probability
- (T is the temparature, k_B is the Boltzmann constant)

$$e^{\frac{E_i - E_j}{k_B T}}$$

Simulated annealing

- If the temperature decreases sufficiently slowly, the solid reaches ***Thermal equilibrium*** at each temperature.
- The Metropolis algorithm generates a large number of transitions at a given temperature.
- ***Thermal equilibrium:***

$$P_T[X = i] = \frac{\exp\left(-E_i/k_B T\right)}{\sum_j \exp\left(-E_j/k_B T\right)}$$

Simulated annealing

- Solutions combinatorial optimisation problem
  - <-> states of the physical system
- Cost of a solution
  - <-> energy of a state
- Control parameter c
  - <-> temperature * Boltzmann constant
- -> basic simulated annealing algorithm (Kirkpatrick 1983)

Simulated annealing algorithm

- Initialize (i_{start}, c_0,L_0)
- Let $k = 0$, $i = i_{start}$
- Repeat until stopcriterion is met
  – For $l=1$ to L_k do
    - Generate (j from S_i)
    - If $f(j) < f(i)$ then $i = j$
    - Else if $\exp(f(i)-f(j)/ck) > \text{random}[0,1)$ then $i = j$
  – $k=k+1$
  – Compute length (L_k), control(c_k)

Simulated annealing Theorem (Aarts, Korst)

- S, f is an instance of an optimisation problem (S is the solution space, f is the goal function)
- A sufficiently large number of transitions in the SA algorithm leads to the probabilty of state i in S:

$$P_T[X = i] = \frac{\exp(-f(i)/c)}{N_0(c)} \qquad P_T[X = i] \equiv q(i)$$

$$N_0(c) = \sum_{j \ in \ S} \exp\left(-\frac{f(i)}{c}\right)$$

Simulated annealing Corollary(Aarts, Korst)

- It is not hard to prove that from the theorem follows: (S^* is the set of globally optimal solutions in S, X_A is the characteristic function on the set A)

$$\lim_{c \to 0} q_i(c) = \frac{1}{|S^*|} \chi_{S^*}(i)$$

Simulated annealing

- Is open to statistical (theoretical) analysis
- Works well for graph partitioning problems
- Produces competitive results for quadratic assignment, scheduling and graph colouring at a computational cost
- Underperforms with respect to tailored solutions for TSP, number partitioning, …

Simulated annealing and the real world

- VLSI, image processing, assembly
  - > no tailored algorithms are available, goal function is messy
- Better than time-equivalent iterative improvement. More substantial for larger problems.
- Depends on skill and effort! (neighborhood, Cooling,..)

Some examples

- **Genetic algorithms**
- Evolutionary algorithms
- **Tabu search**
- **Simulated annealing**
- Great deluge
- **Hyperheuristics**
- **Variable neighbourhood search**
- Late acceptance
- Extreme optimisation
- Neural networks
- Electrostatic potential

METAPHORS

Variable neighborhood search (Mladenovic and Hanssen 1997)

- Fact 1

  A local minimum with respect to one neighborhood structure is not necessarily so for another

- Fact 2

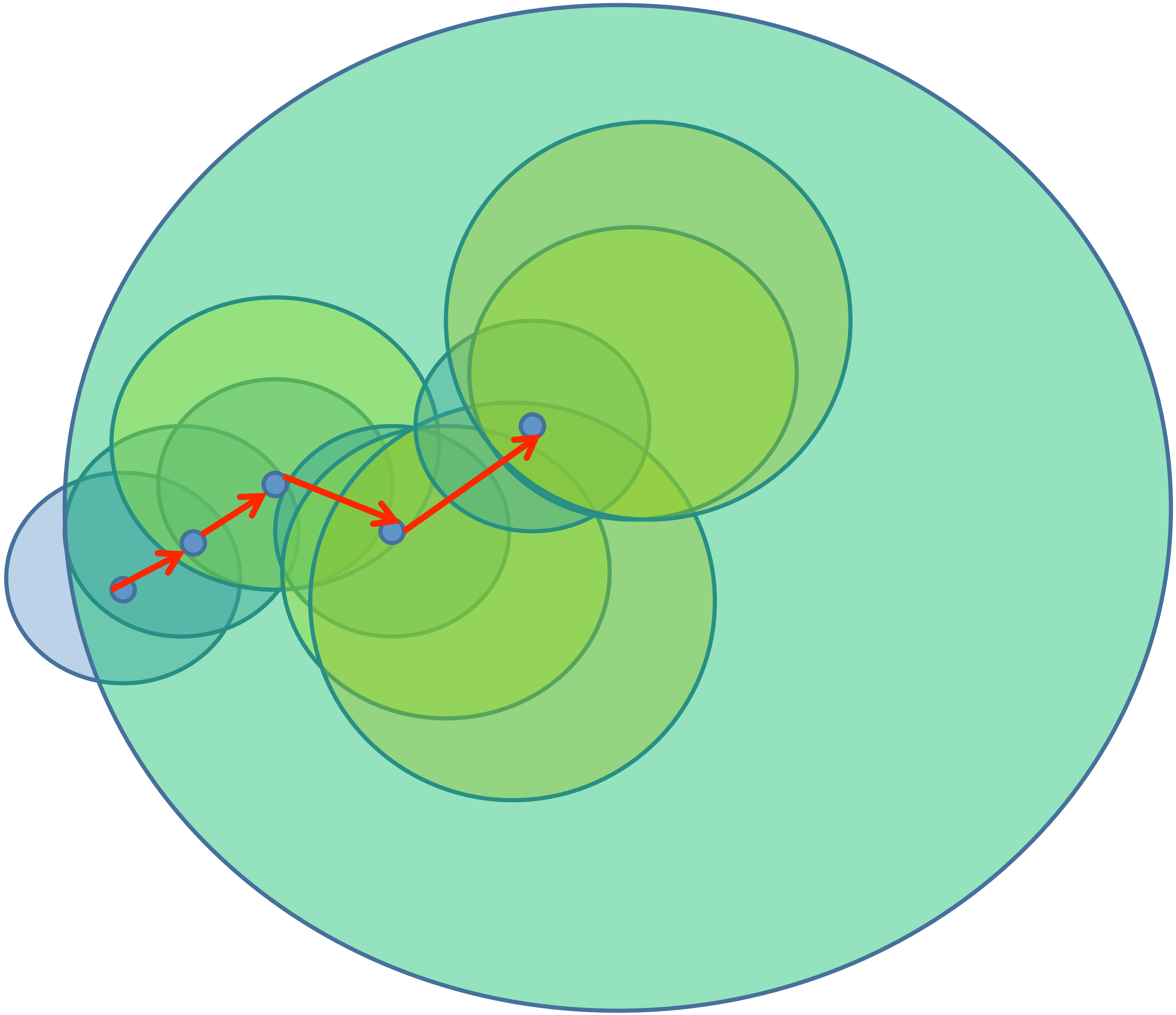
  A global minimum is a local minimum with respect to all neighborhood structures

- Fact 3

  For many problems local minima with respect to one or several neighbourhoods are relatively close to each other

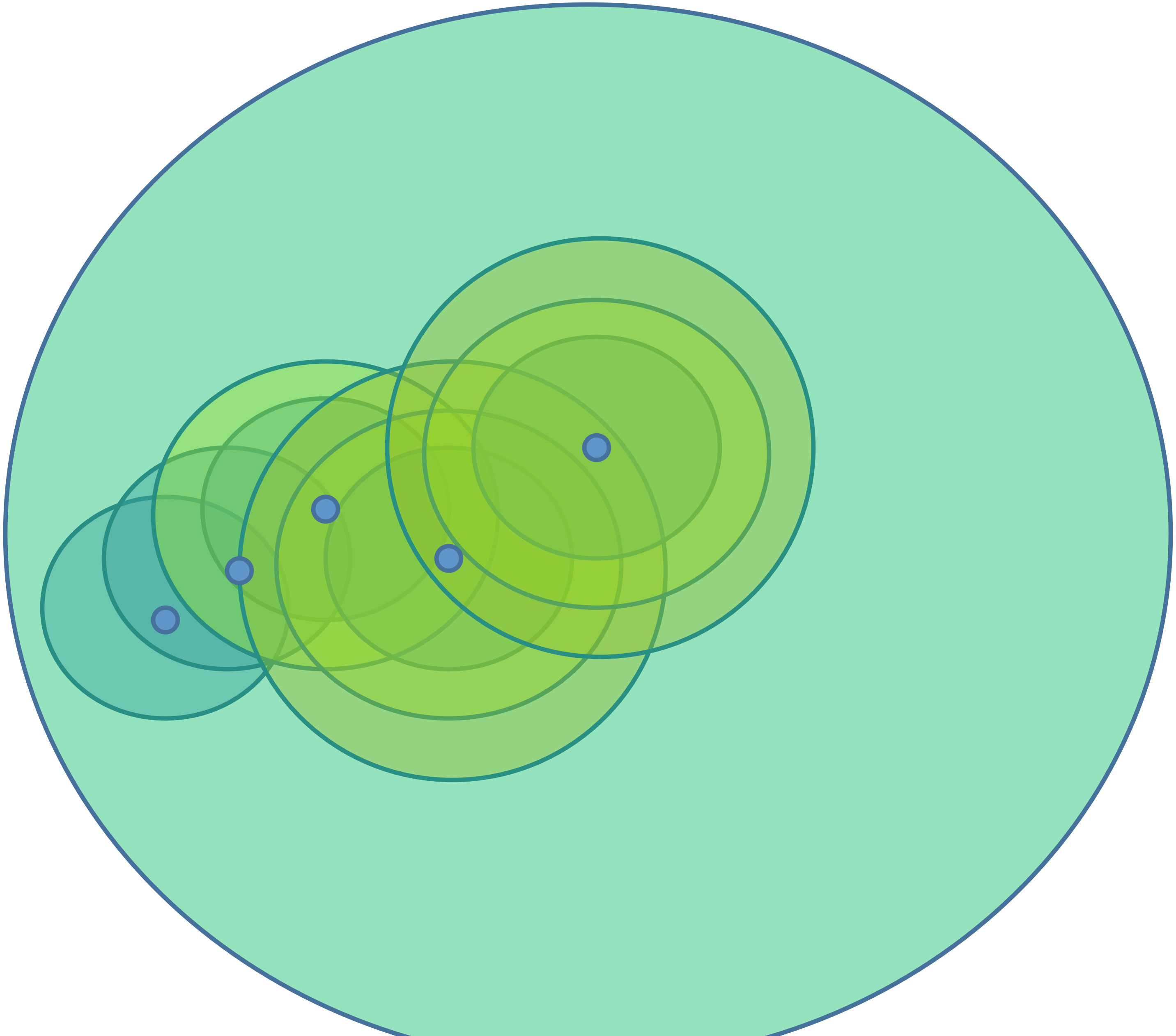
Variable neighbourhood search Descent

- Select a set of neighbourhood structures N_l (l = 1 to l_{max}) Find an initial solution x
- Set $l = 1$, repeat until $l = l_{max}$
  - Exploration:
    - find the best neighbour x' of x in $N_l(x)$
  - Acceptance
    - If the solution x' is better than x, let x = x', $l = 1$
    - else let $l = l+1$



Variable neighbourhood search Reduced

- Select a set of neighbourhood structures N_l ($l = 1$ to l_{max}) Find an initial solution x; choose a stoping condition
- Repeat until the stopping condition is met
- Let $l = 1$; repeat until $l = l_{max}$
  - Shake:
    - Generate a **random** neighbor x' of x in $N_l(x)$
  - Acceptance
    - If the solution x' is better than x, let x = x', l = 1
    - else let l = l+1

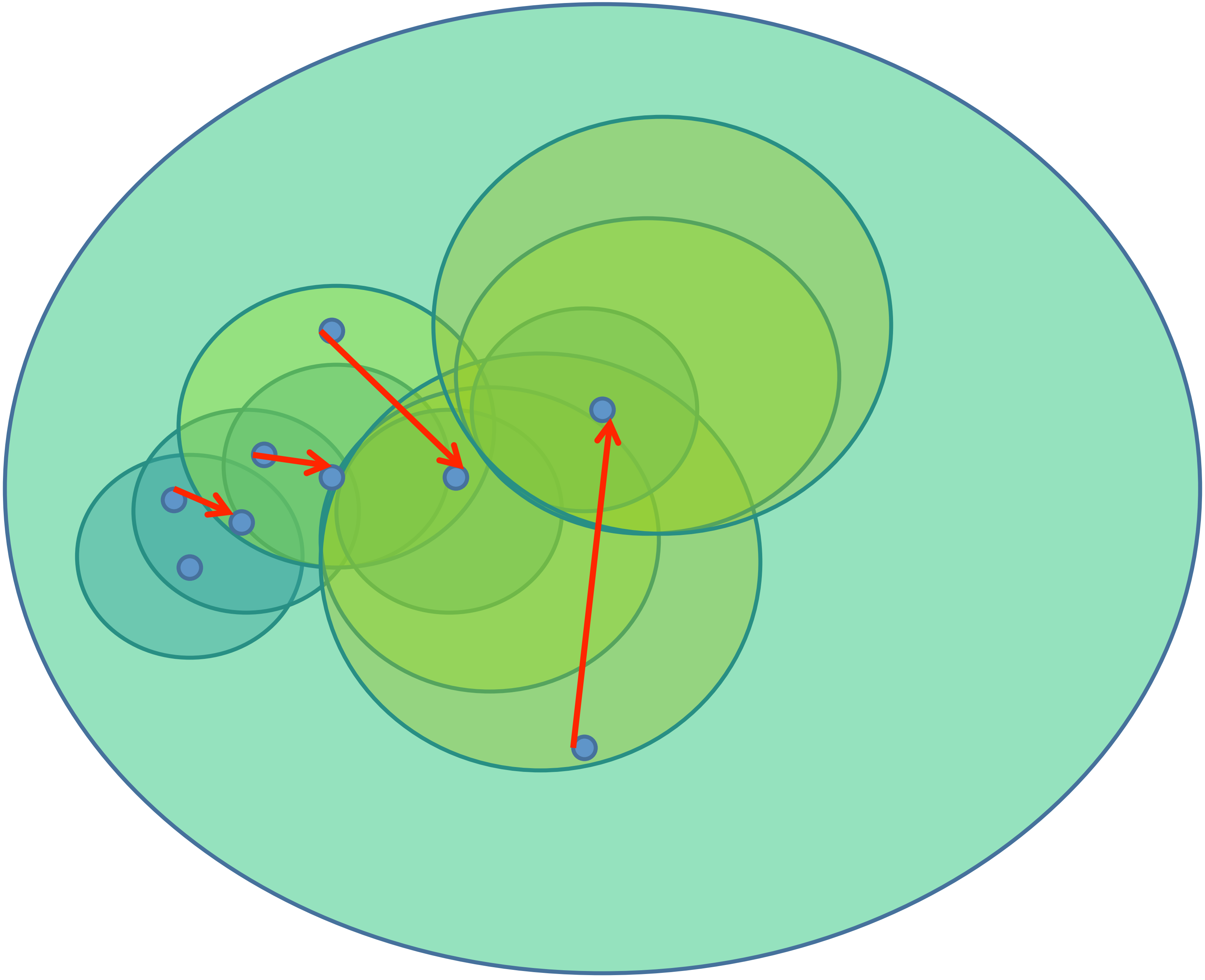


Variable neighbourhood reduced

- Neighbourhoods are often nested here, and in any case, lower l neighbourhoods should be smaller
- Results in an intensification/diversification behaviour of the algorithm
- This bears on Fact 3 (closeness of local optima) and Fact 2 (once we make it to the last (largest) neighbourhood we may be in a global optimum

Variable neighbourhood Basic

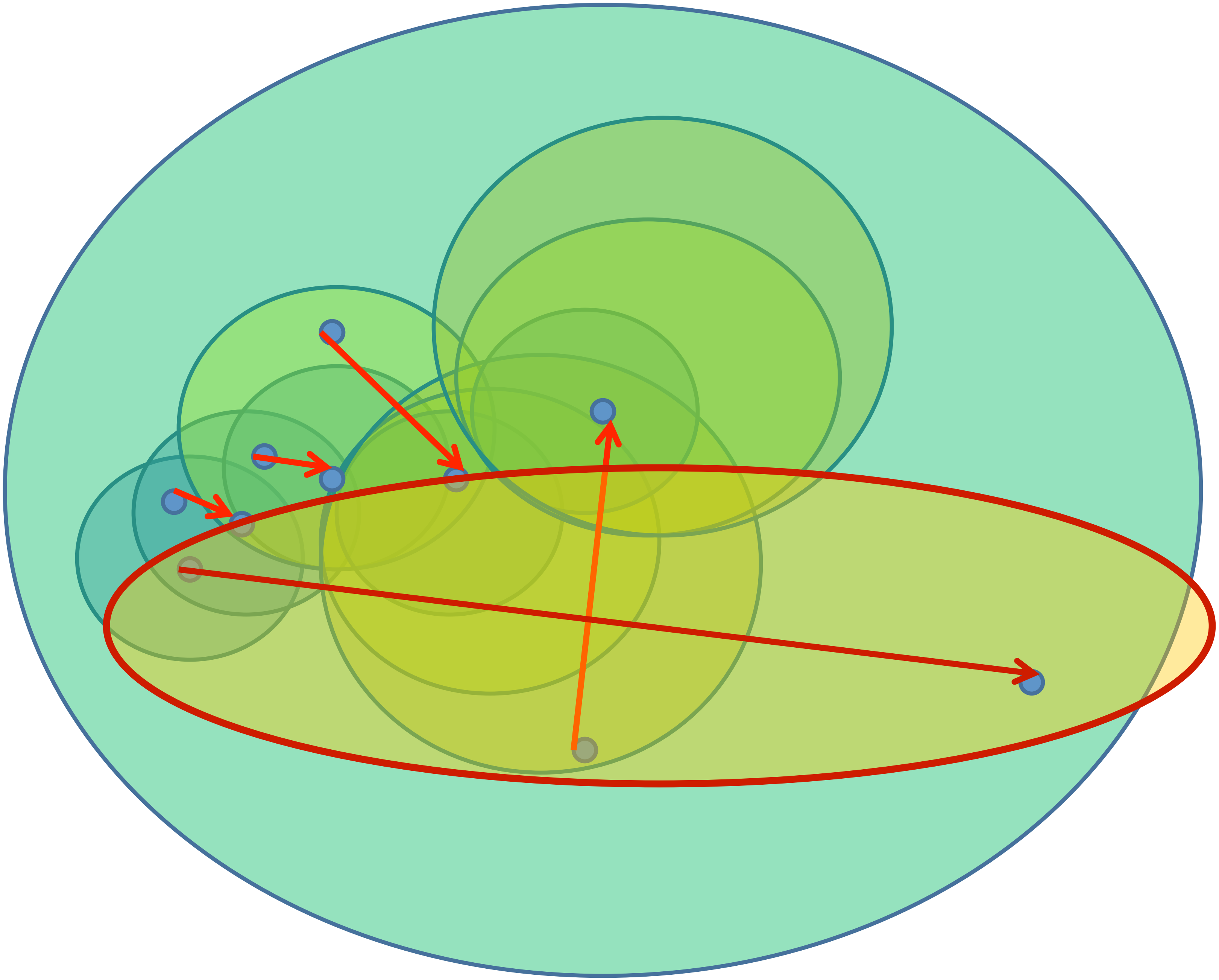
- Select a set of neighbourhood structures N_l(l = 1 to l_{max}) Find an initial solution x; choose a stoping condition
- Repeat until the stopping condition is met
- Let $l = 1$; repeat until $l = l_{max}$
  - Shake:
    - find a random neighbour x' of x in $N_l(x)$
  - Local search
    - Local search with x' as initial solution to find x''
  - Acceptance
    - If the solution x'' is better than x, let x = x'', $l = 1$
    - else let $l = l+1$



Variable neighbourhood search General

- Select a set of neighbourhood structures
  N_k ($k = 1$ to k_{max}) for shaking,
  N_l ($l = 1$ to l_{max}) for local search. Find an initial solution x and improve it by RVNS; choose a stoping condition
- Repeat until the stopping condition is met
- Let $k = 1$
- Repeat until $k = k_{max}$
  - Shake:
    - find a random neighbour x' of x in $N_k(x)$
  - Let $l = 1$; repeat until $l = l_{max}$
    - Explore:
      - find the best neighbour x'' of x' in $N_l(x')$
    - Acceptance
      - If x'' is better than x', let x' = x'', $l = 1$, else let $l = l+1$
  - Acceptance
    - If the local minimumb x'' is better than x, let x = x'', $k = 1$
    - else let $k = k+1$

~RVNS



Applications

- Graph theory
- Plant location
- Timetabling
- ETP
- …

Shift	1	2	3	4	5	7	1	2	3	C
Pjotr	A	A	A			C	T	T	F	1
Ludwig	C	C			R	R	T	T	T	0
Clara	T	T	C		R	R	F	T	T	1
Hildegard			A	A	A		T	T	T	0
Johann			C	C			T	T	F	1
Wolfgang		C	T	T	C		T	T	T	0
Guiseppe	R	R			A	A	F	T	T	1
Antonio	R	R			C	C	F	T	T	1
Arranger	2	2	2	1	0	0				
Tonesetter	1	2	1	1	0	0				
Composer	1	1	1	1	2	0				
Reader	3	1	1	1	2	0				

(Variable neighbourhood) Tricks,..

- Familiarization
- Read
- Test instances
- Data structure
- Initial solution
- Objective value calculation
- Shaking
- Local search
- First-Best improvement
- Reduce neighborhood
- Intensify shaking
- Neighborhood structures
- Parameter setting
  - 1 parameter in VNS: the number of neighborhoods

General important properties for metaheuristics

- Simplicity (clear principle)
- Precision (mathematical)
- Coherence (principle)
- Efficiency (near optimal)
- Effectiveness (computing time)
- Robustness (variety of instances)
- User friendliness (parameters)
- Innovation (new applications)

Some examples

- **Genetic algorithms**
- Evolutionary algorithms
- **Tabu search**
- **Simulated annealing**
- Great deluge
- **Hyperheuristics**
- **Variable neighbourhood search**
- Late acceptance
- Extreme optimisation
- Neural networks
- Electrostatic potential

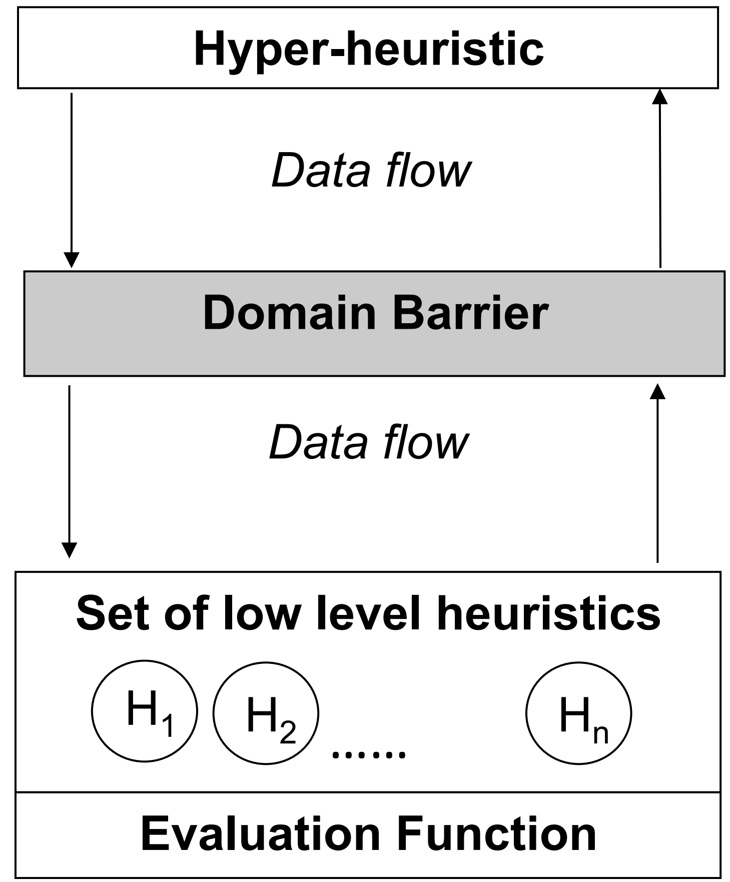
METAPHORS

Hyperheuristics

- Simple Idea: Heuristics to choose heuristics
- Operates on a search space of heuristics rather than directly on a search space of solutions

Hyperheuristics

- Metaheuristics tackle specific problems
- Are often tailored
- Can we increase the level of generality?
- Good enough, cheap enough, fast enough?



Hyperheuristics

- No ‘Superalgorithms’ (No Free Lunch)
- Intelligent schemes that function well on a number of problems
- Give the developer extra handles to bring in his domain expertise without requiring him to study the details of an advanced optimisation strategy

Hyperheuristics
Example

- Choice Function
- f1 + f2 + f3

- How well has each heuristic done +
- How well have pairs of heuristics done +
- Time since last called

- Applied to sales summit scheduling,
- nurse rostering, exam timetabling

Hyperheuristics

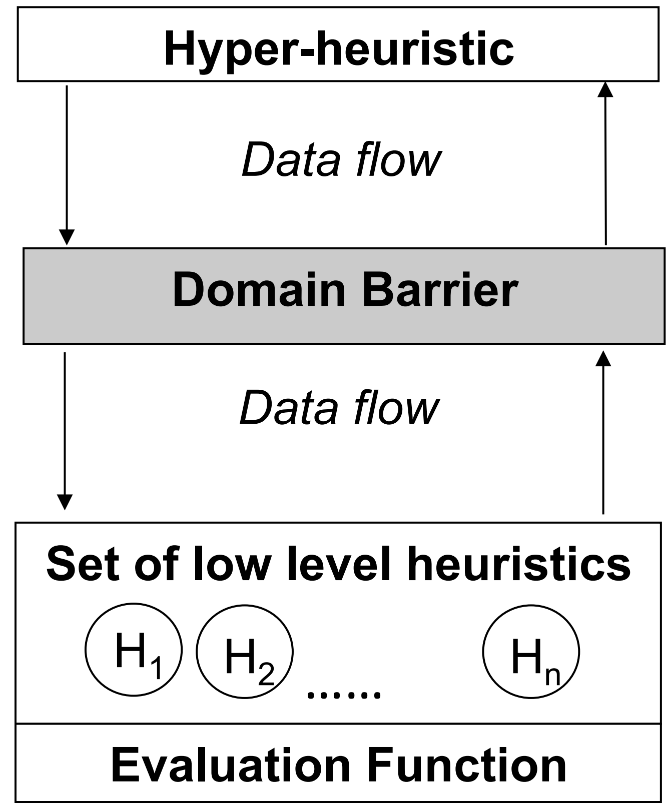
- Sales Summit Scheduling
- Low level heuristics are based on three types of neighborhood moves:
  - Add / Remove one delegate to / from the current solution
  - Add / remove a meeting to / from the current solution (random, 1st improving, best improving, etc)
  - Remove excess of meetings from an overloaded delegate

Hyperheuristics

- Nurse Rostering, Exam timetabling
  - Competitive with other results in the literature
  - Same algorithm worked across a range of problems

Hyperheuristic

- Above the domain barrier
  - Genetic algorithm
  - Genetic programming
  - Tabusearch
  - Simple random
- Room for machine learning
  - Learn the weights of the low level heuristics
  - Relate them to solution space features (> expert?!)
  - Reduce the number of parameters
- Hyper heuristic competition



Conclusions

- Combinatorial problems often require heuristic approaches
- Metaheuristics offer a framework to express domain knowledge
- Hyperheuristics separate domain expertise from heuristics expertise
- No single best one fits all solution
- Often a lot of tailoring is required

Heuristics and Learning

- Machine learning may help to reduce the number of parameters
- What should we look at
  - What are solution space features
  - How to recognise phases in the search
    - E.g. diversification/intensification
  - What does the heuristic really need to see?
    - Hyperheuristics going on performance alone (improvement, time taken,…)

Heuristics and learning

- Online/Offline learning
- Features
  - Expert specified
  - Automatically detected
  - Generically generated
- -> An example, off line, from nurse rostering