Self-Organising Systems

Rudolf Mayer

Institut für Softwaretechnik & Interaktive Systeme

(mayer@ifs.tuwien.ac.at)

SBA Research

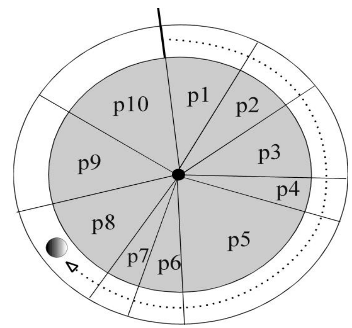
(www.sba-research.org)

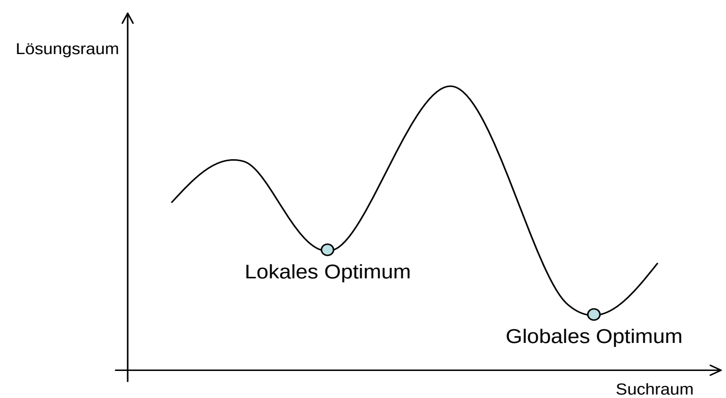
Agenda

- Brief Recap & Genetic algorithms, continued
- Genetic Programming
- Ant Colony Optimisation
- Agents

Recap: Genetic Algorithms

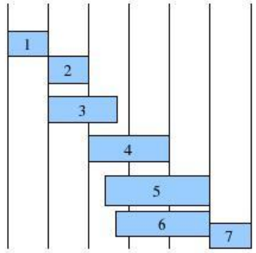
- Genetic Algorithms
  - Coding – binary, gray code, ...
  - Starting population – initialisation methods
  - Genetic operators: reproduction / crossover / mutation
  - Fitness function
  - Selection (Roulette wheel)
  - *Population diversity*
    - *Tournament selection*
    - *Crowding, Fitness sharing*
    - *Adaptation of mutation & crossover*



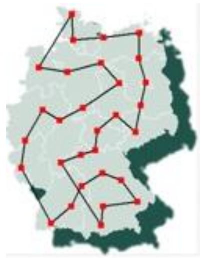


Recap: Genetic Algorithms

- Genetic Algorithms Applications
  - Generally heuristic optimisation / search tasks
    - Often NP (hard) problems
  - Scheduling (job scheduling) – ideal assignments of jobs to resources at particular times



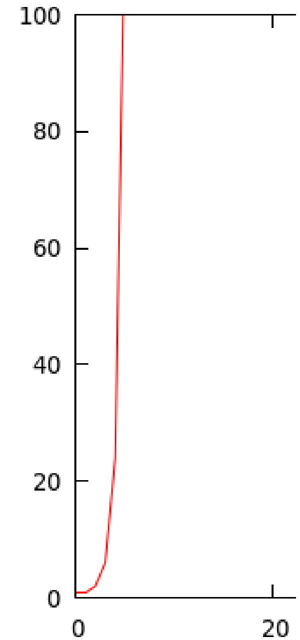
  - Timetabling
  - Travelling Salesman – finding the optimal route



  - Economics (e.g. supply/demand model)

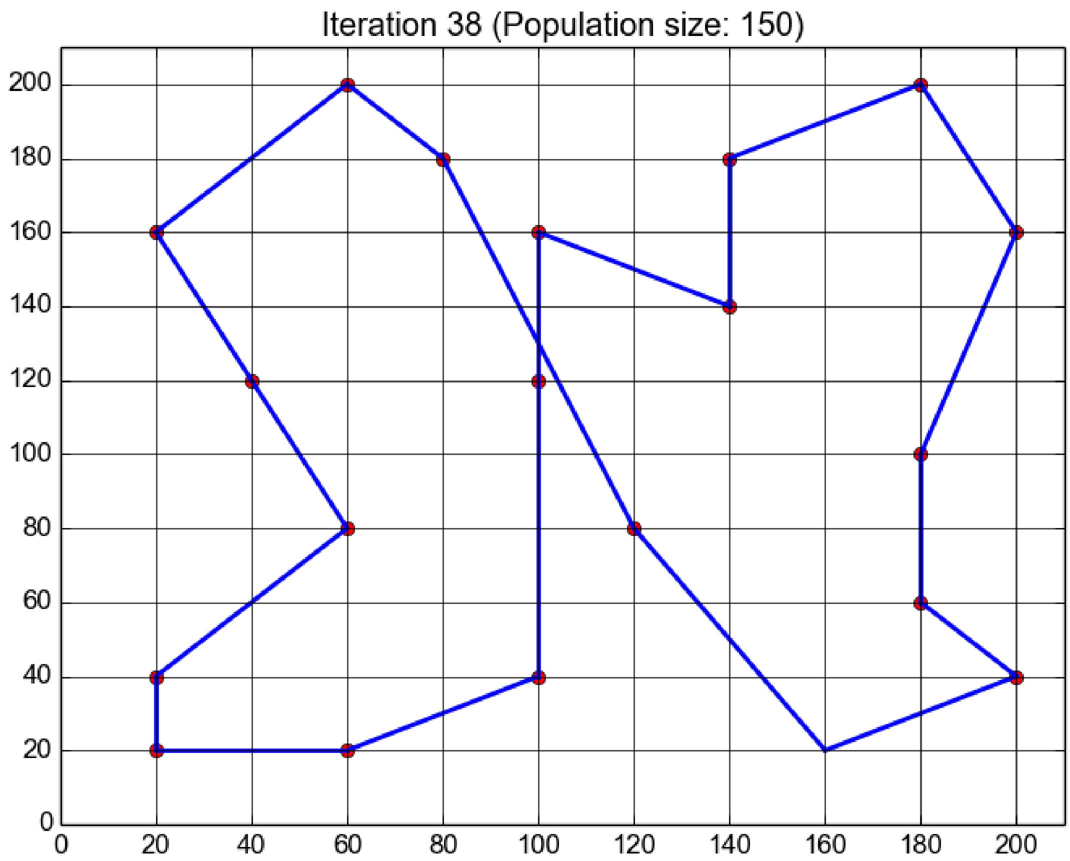
Recap: GA: Travelling salesman

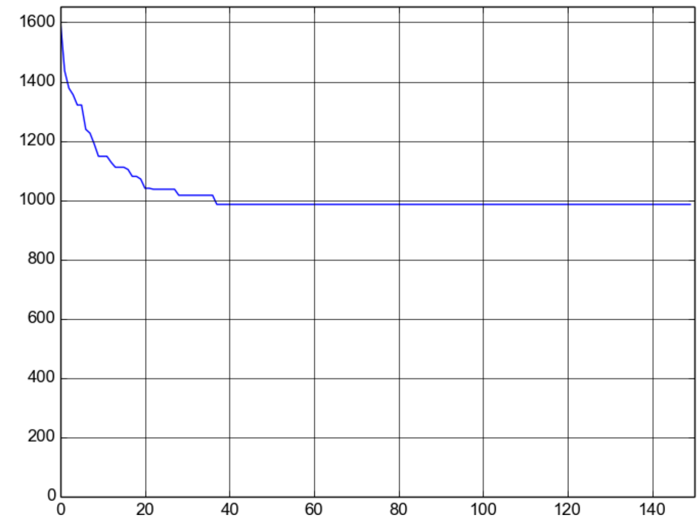
- Given a set of locations, visit all locations in the optimal route
- Simple example: 3 cities
  - E.g. Vienna, Graz, Linz
- *How many possible routes?*
  - 6 different routes (be less if starting & end city are fixed)
- *General number of routes?*
  - Factorial: n! = n × (n−1)!
    - 3! = 3 x 2 x 1 = 6



Recap: GA: Travelling salesman

- Run 1
  - Population size: 150, Iterations: 150
  - Final solution in Iteration 38

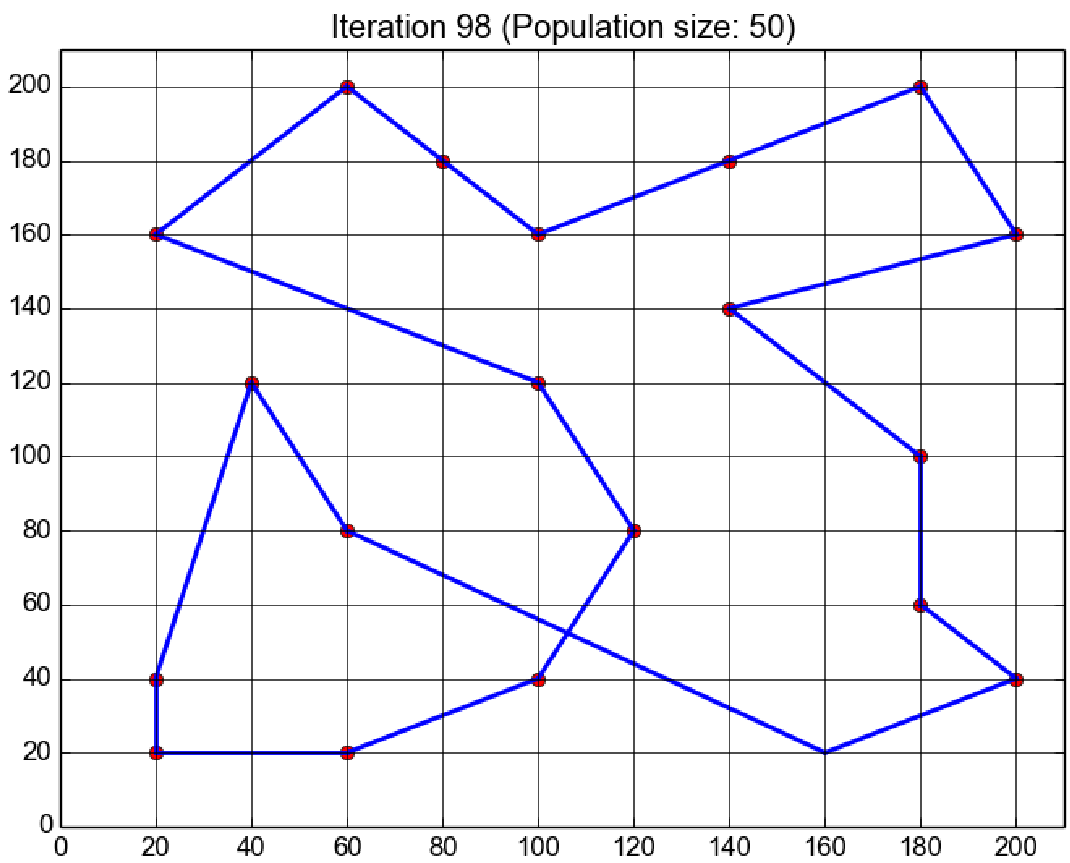


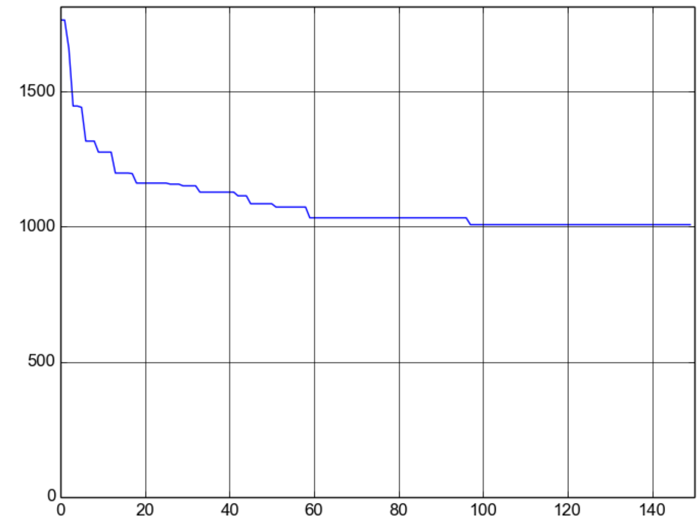


Fitness (distance)

Recap: GA: Travelling salesman

- Run 2
  - Population size: 50, Iterations: 150
  - Final solution in Iteration 98



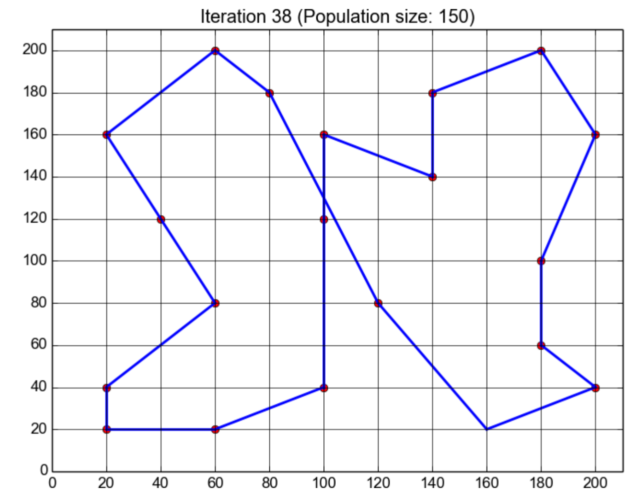


Fitness (distance)

Recap: GA: Travelling salesman

- Mind: population size vs. phenotype (solution) size

- In the previous example
  - 20 cities
    - Phenotype / solution contains 20 indices
  - Population size 150
    - At each iteration (generation): create 150 ***candidate*** solutions
    - In the example: always shown the best solution in that iteration



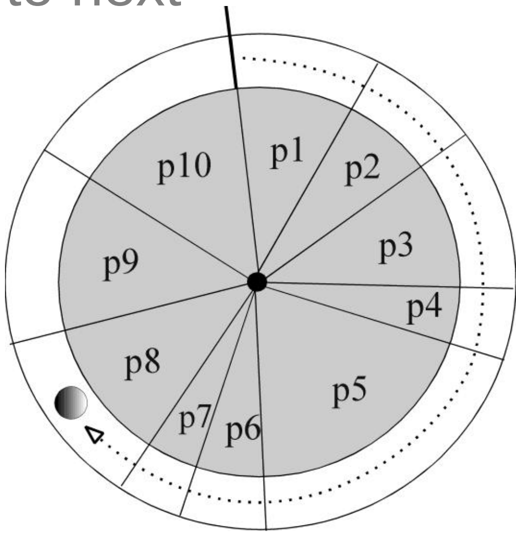
Recap: GA: Travelling salesman

- Encoding
- Adaptations of mutation & crossover operators
  - To ensure valid solutions
- Relatively quick convergence
- *Optimal solution found?*

Selection

- Models the principle of „surviving the fittest“ observed in nature
- Simple approach: only select the ***n*** fittest
  - Favours the fitter individuals, suppresses the weak ones
  - Each individual should have some chance to pass to next generation
- Roulette wheel
  - Probability P_i of selecting *i*-th individual is proportional to its fitness f_i
    - *(Fitness proportionate selection)*

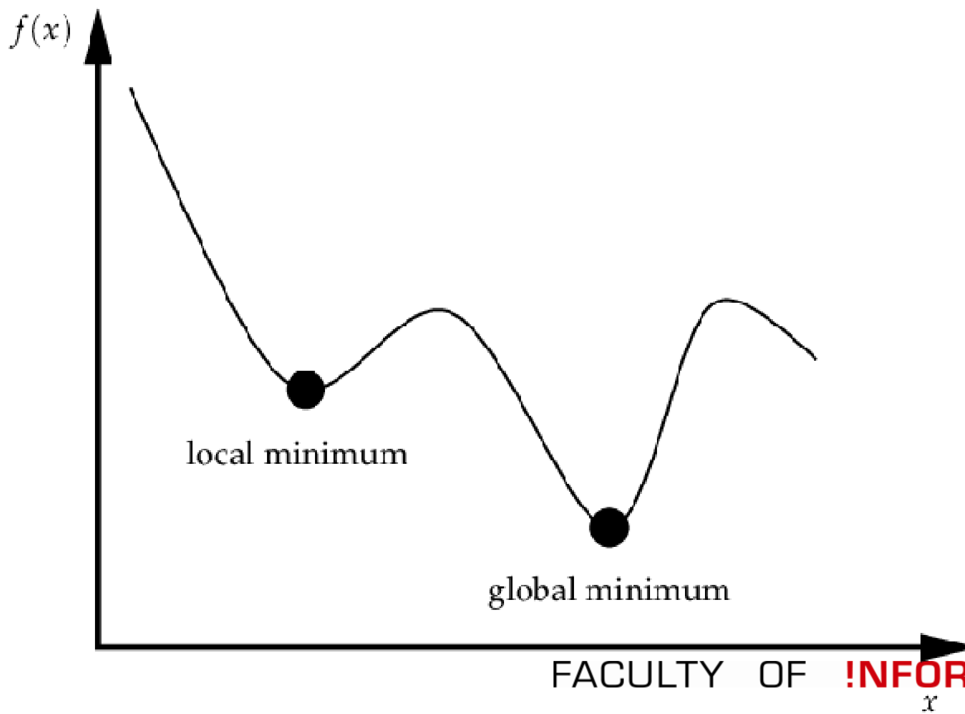
$$P_i = \frac{f_i}{\sum_{j=1}^{PopSize} f_j}$$



- Other selection schemes
  - Stochastic universal sampling, ***Tournament selection***, Reminder stochastic sampling, Rank-based selection

Population diversity

- Diversity is crucial
  - Early homogenization of genetic material causes premature convergence
    ➔ Disables valuable exploration of the search space
    ➔ May lead to local optimum



Population diversity

- Crowding, Fitness sharing
  - The more dense a region of the search space (more individuals of the current population) ➔ the less fit they become

- Modified **selection**, e.g. restricted tournament selection
  - Select individual from various subsets
    - Subsets contain only similar individuals, achieved e.g. by clustering
    - More diversity, as also individuals from sub-optimal groups get chosen

- Adaptation of **mutation** & **crossover** genetic operators
  - When the population converges: change (increase) mutation and/or crossover parameters and/or recombination operators

  ➔ Exploration is increased

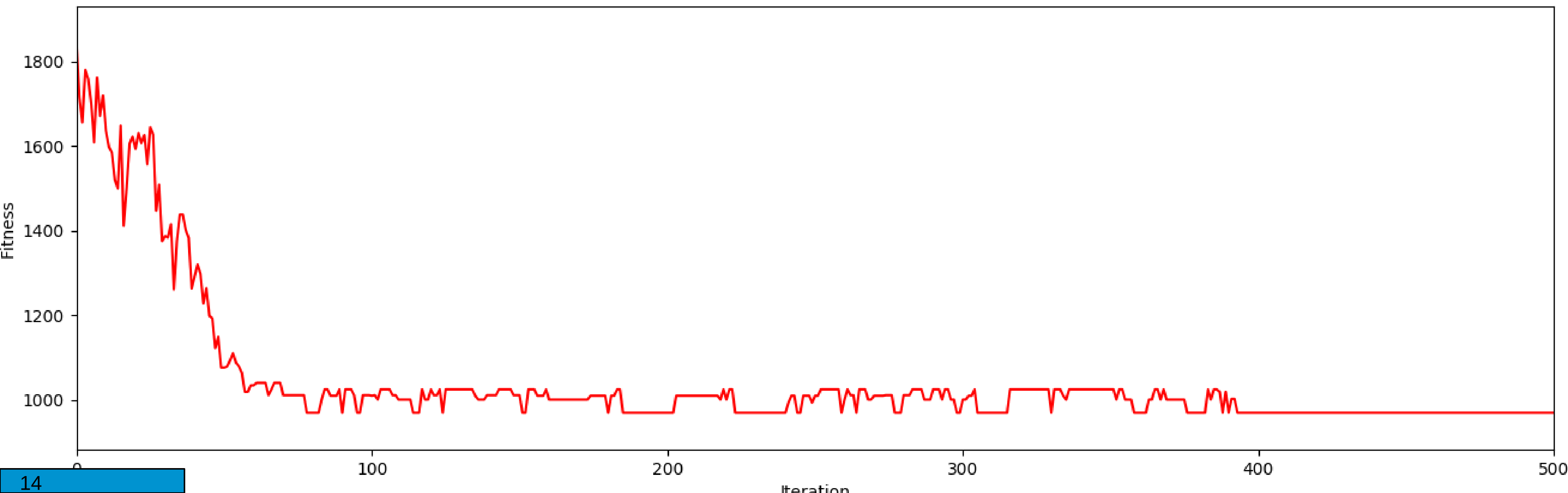
GAs: Recap general ideas

- Ideas are:
  - Adaption
  - Bring new „ideas“ into a system (Mutation)
  - Select among competitors (Selection)
  - Bring together successful individuals (Mating)

„It‘s Alive“: The Coming Convergence of Information, Biology, and Business“, Christoph Meyer, Stan Davis

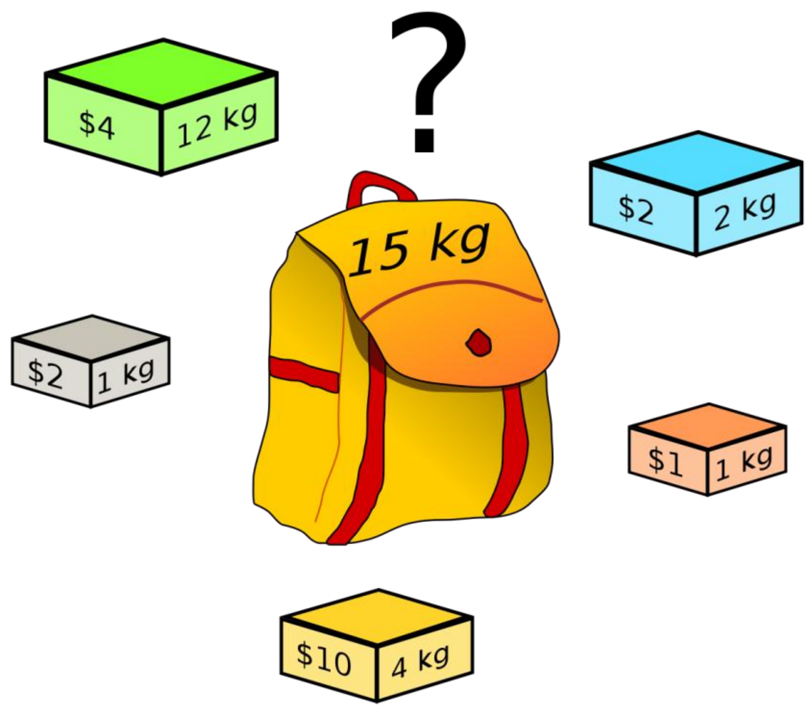
GA: elitist algorithm

- Elitist algorithm variation:
  - During selection: always keep best-known solution so far
  - *Why?*
    - Avoid regress in optimal solution found so far
    - If population big enough: not influencing population diversity too much
    - Alternative: Pocket algorithm – keep best known solution as ***result***



GA: Other example

- Given a set of items, each with a weight and a value
- Determine which items to include in a collection so
  - That the total weight is less than or equal to a given limit
  - And the total value is maximised



- *How to represent as GA?*

Agenda

- Recap
- Genetic Programming
- Ant Colony Optimisation
- Agents

Genetic Programming (GP)

- Evolutionary Algorithm, similar to Genetic Algorithms
- Structures that undergo adaptation are **trees**
  - Variable size and shape

- Trees represent programs / ***instructions***
  - Composed of functions (inner nodes) and terminals (leaf nodes) chosen for the problem at hand:
    - Terminals T – operands: input variables of the program
    - Functions F – operations on the data

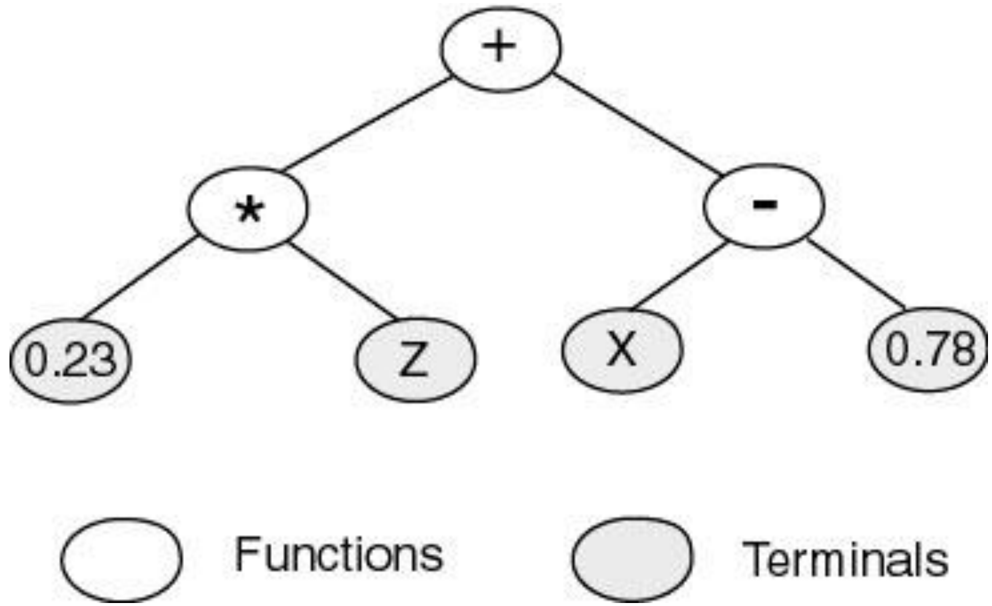
- Tree evaluated in a recursive manner

Genetic Programming (GP)

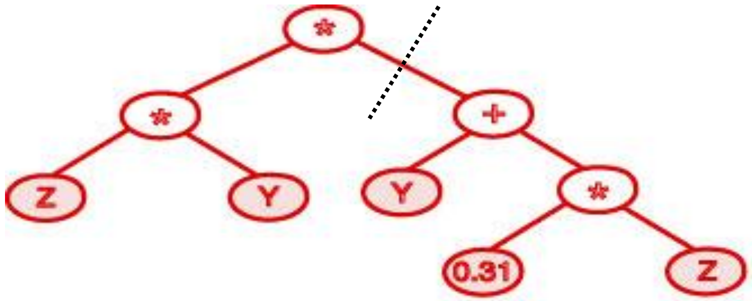
- Trees composed of functions and terminals
  - Terminals T – input variables
    - real-valued, integer or logical constants, functions w/o argument
  - Functions F
    - arithmetic functions (+, -, *, /)
    - algebraic functions (sin, cos, exp, log)
    - logic functions (AND, OR, NOT)
    - conditional operators (If-Then-Else, cond?true:false)
    - other problem specific operations
- Closure condition
  - It is necessary that the output of an arbitrary function or terminal can be an input of any other function
    - Favours languages like LISP & generally functional languages

GP: Representation

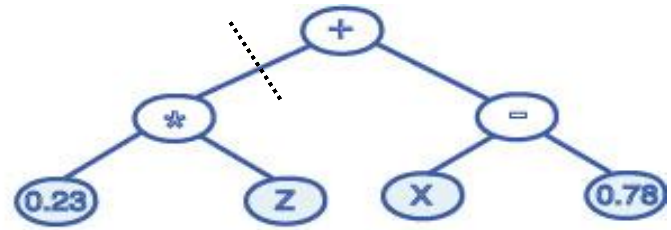
Ex.: Tree representation of expression **0.23 * Z + X - 0.78**



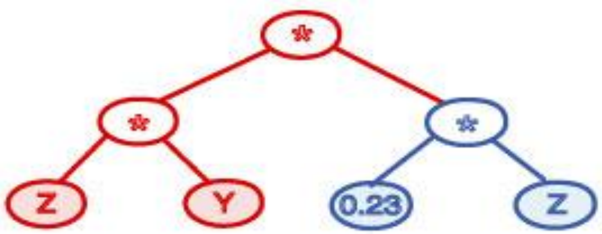
GP: Crossover



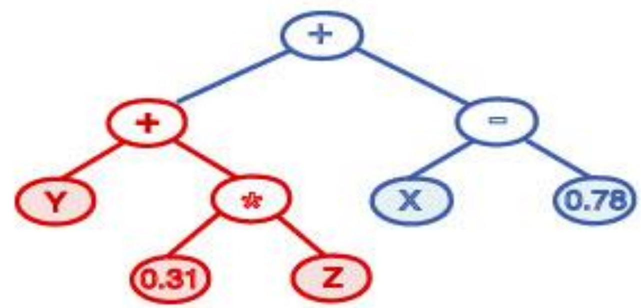
Parent 1: Z * Y * (Y + 0.31 * Z)



Parent 2: 0.23 * Z + X - 0.78



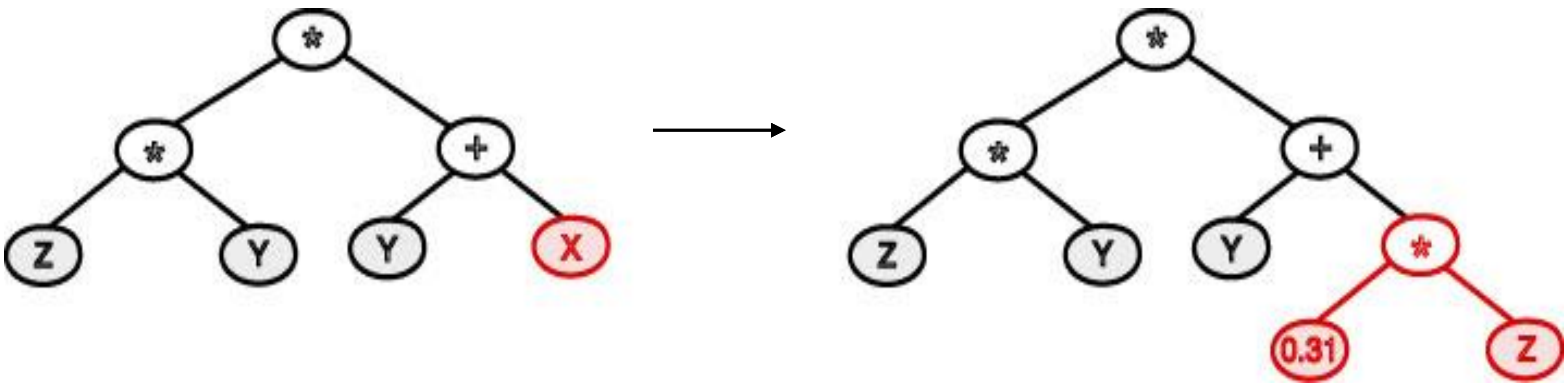
Child 1: 0.23 * Y * Z^2



Child 2: Y + 0.31 * Z + X - 0.78

GP: Mutation and Others

- Mutation
  - replaces selected sub tree with a randomly generated new one



  - Other operators proposed, e.g. permutation

GP: problem definition

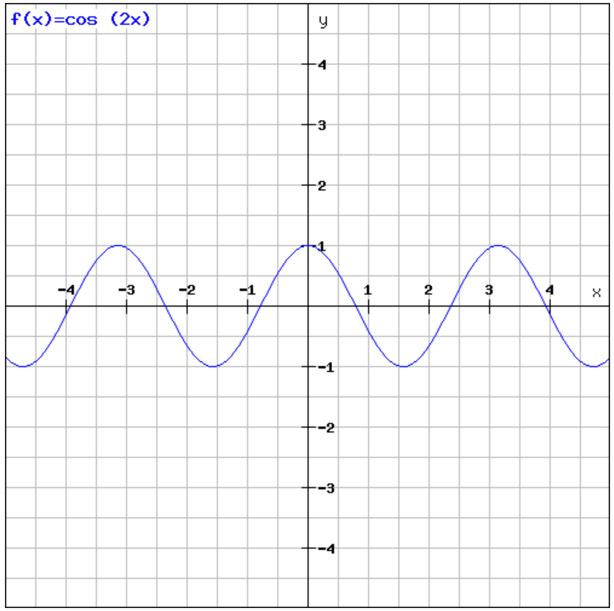
- Goal: find a tree that represents an (optimal) program

- Challenges
  - How to evaluate optimality?
    - For which types of programs is this doable?
  - Size of search space

- Example: finding identical mathematical functions

GP Example: Finding a trigonometric identity

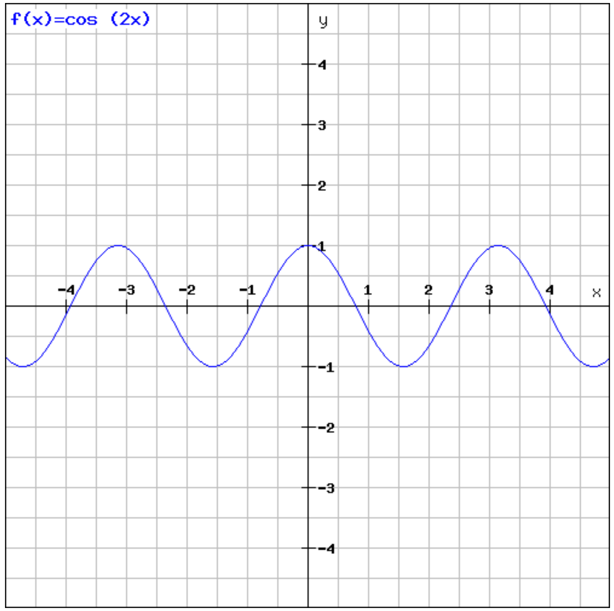
- Goal is to find a solution for the right side of the equation $\boxed{\cos 2x \equiv ?}$



- Terminals: T = {X, constant 1.0}
- Functions: F = {+, -, $*$, %, SIN}
- Fitness: Sum of absolute difference between y_i and the value generated by the tested expression for given x_i

GP Example: Finding a trigonometric identity

- Goal is to find a solution for the right side of the equation $\boxed{\cos 2x \equiv ?}$



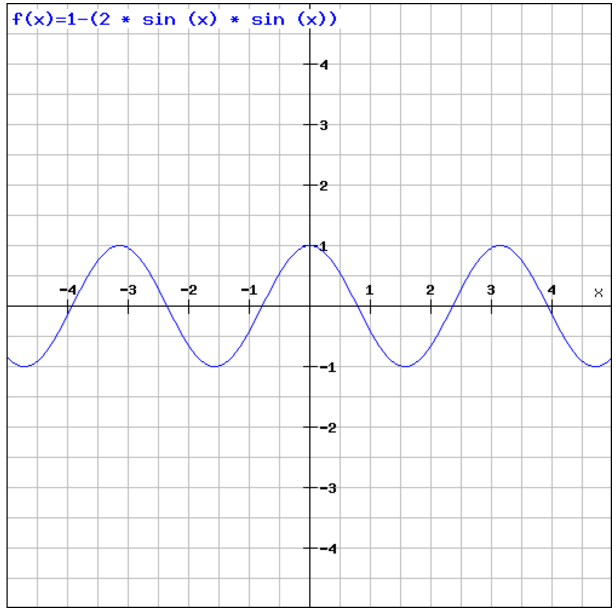
- Stopping criterion:
  solution with fitness (error) < 0,01

- Test cases: 20 pairs (x_i, y_i) (population size)
  - x_i randomly chosen from the interval $\langle 0, 2\pi \rangle$
  - $y_i = \cos(2 x_i)$ (groundtruth)

GP Example: Results

- 1^{st} test run
  - 13^{th} generation: following solution found (prefix notation)

    (- (- 1 (* (*sin X*) (*sin X*)))) (* (*sin X*) (*sin X*)))



  - After reformulating: **$1 - 2\, sin^2\, (x)$**
    - $1 - 2\, \sin^2 (x)$ is a *known identity* of cos(2x):
    - $\cos(2x) = \cos^2 (x) - \sin^2 (x) = 2 \cos^2 (x) - 1$
      $= \mathbf{1 - 2 \sin^2 (x)}$

- 2^{nd} test run
  - in 34^{th} generation: solution (- 1 (* (* (*sin X*) (*sin X*)) 2))
    - the same expression as in the first run

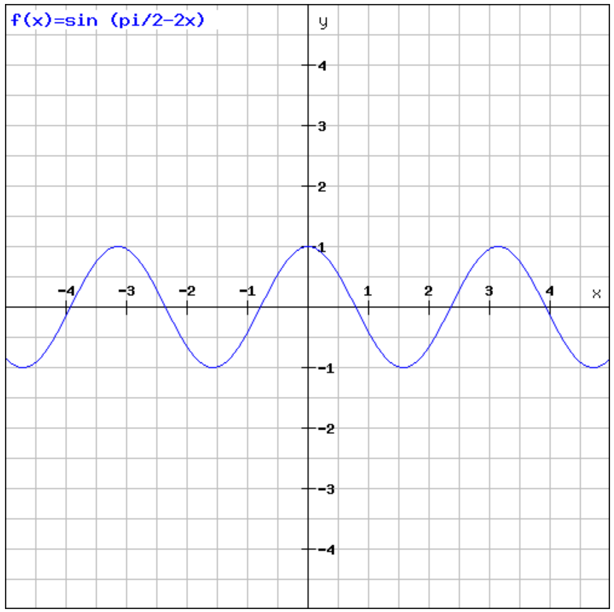
GP Example: Results

- 3rd test run

```
(sin (- (- 2 (* X 2))
        (sin (sin (sin (sin (sin (sin (* (sin (sin 1))
                  (sin 1))
        )))))))
))
```

$\boxed{\approx \frac{\pi}{2}}$

  - Discovered identity:
    cos 2x = sin (π/2 – 2x)



GP: properties

- Learns an equation/program
- Most often represented as tree
  - Closure condition

- Bloating effect – individuals would be reducible to shorter (more readable) form (cf. “cos 2x” example earlier)
  - Difficult to interpret
  - Approaches:
    - Limit allowed depth of individuals
    - Punish individuals with large size
    - Combination of both

- Requires the goodness of the program to be easily **evaluated** by fitness function

GP: Further readings

- http://www.human-competitive.org/awards
  - Papers awarded in the “human-competitive results” challenge
  - (Results were produced by any form of genetic and evolutionary computation, regularly contains GP results)

- Further material:
  - http://www.genetic-programming.com/
    (outdated)
  - http://geneticprogramming.com (updated more recently)

Agenda

- Recap
- Genetic Programming
- Ant Colony Optimisation
- Agents

Ant Colony Optimisation

- Representative of swarm intelligence algorithsm
  - *More algorithms to follow in upcoming lectures*

- Basic algorithm of artificial ant colonies

- Extensions to these algorithms

- Initial Application areas

Why ants?

- Ants are relatively simple individuals
- Characteristics
  - De-centralised decision making
  - Feedback, random behaviour, ...
  - Communication between ants, via pheromones (trail-laying / trail-following behaviour)

➔ Capabilities to solve complex problems
  - Building of geometrically complex mounds
  - **Finding of shortest paths to food sources**

- Self-organising system

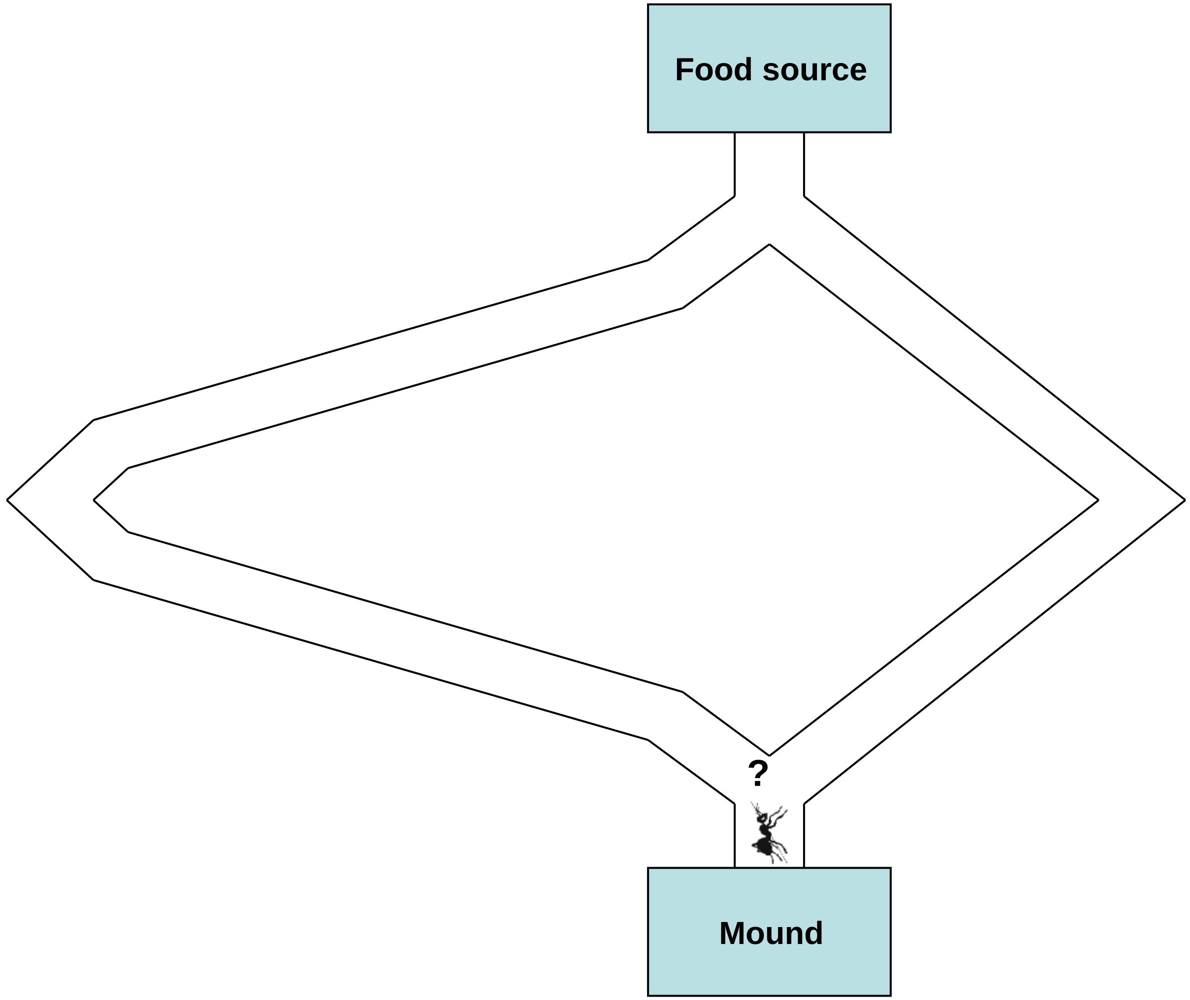
Finding the shortest path

- Ants wander (randomly) to find food
- Once found, return to the mound
  - Leaving pheromone trail

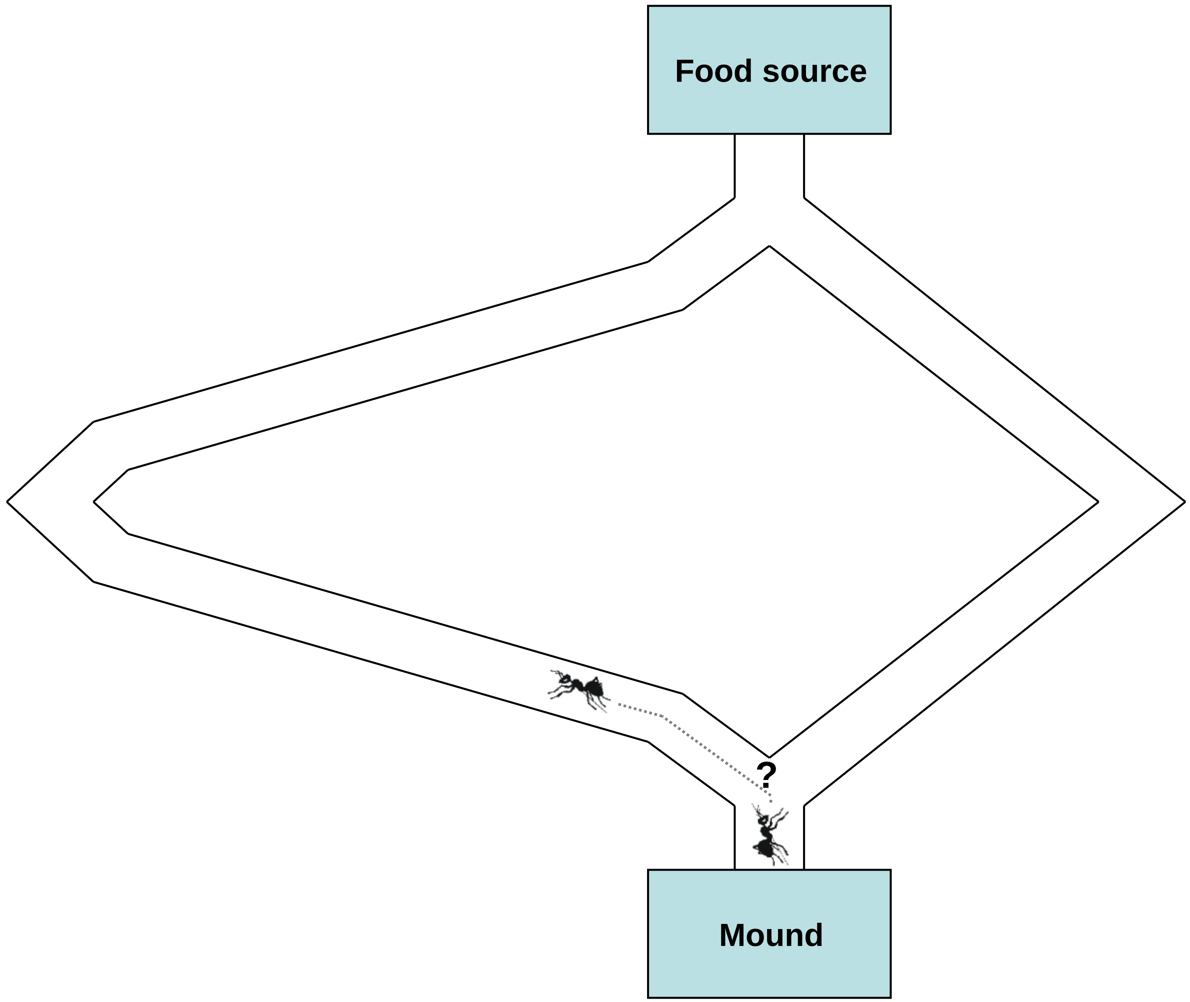
- Subsequent ants will likely follow that trail

- If they also find food, they will reinforce that trail (“solution“)

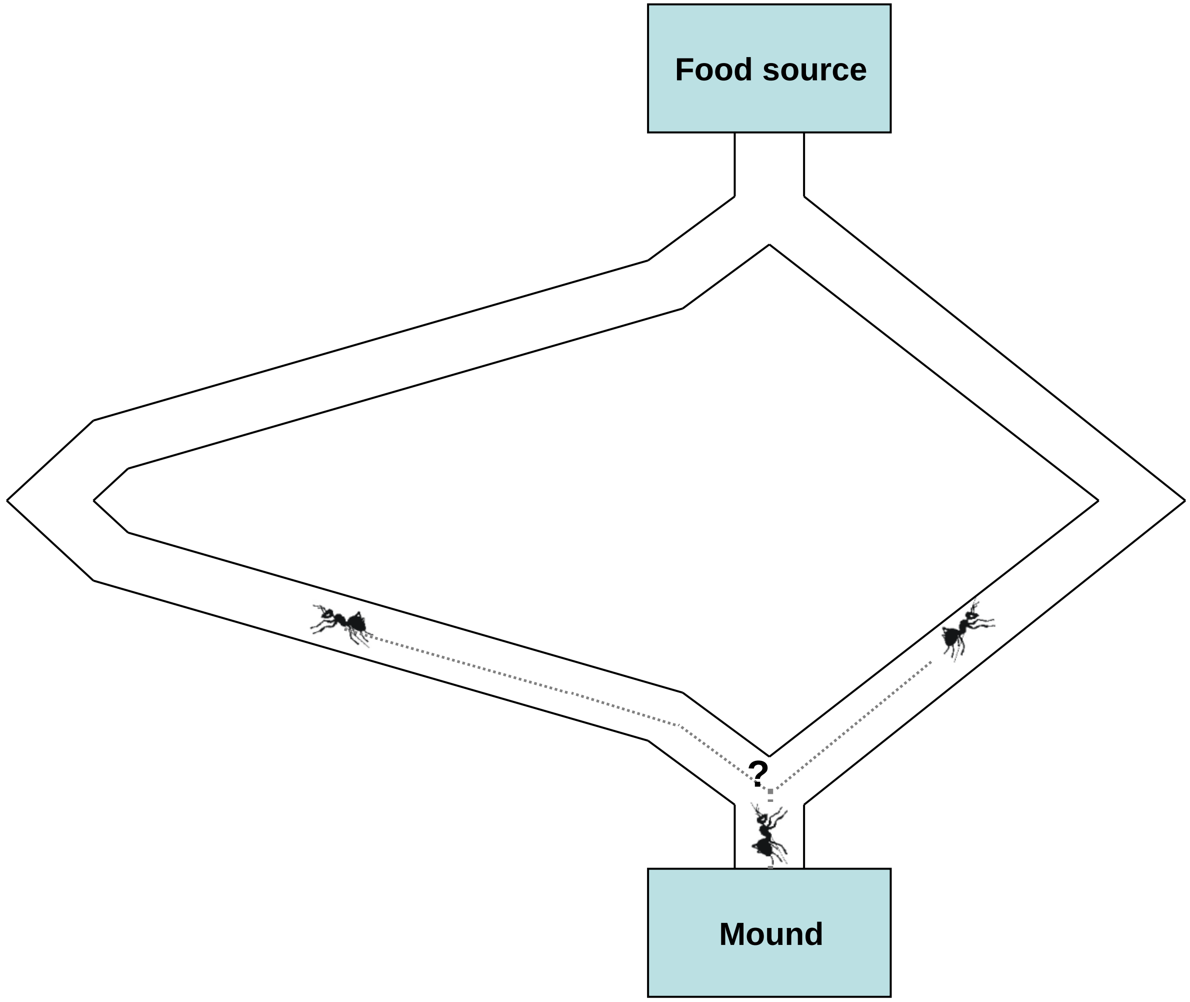
Finding the shortest path



Finding the shortest path

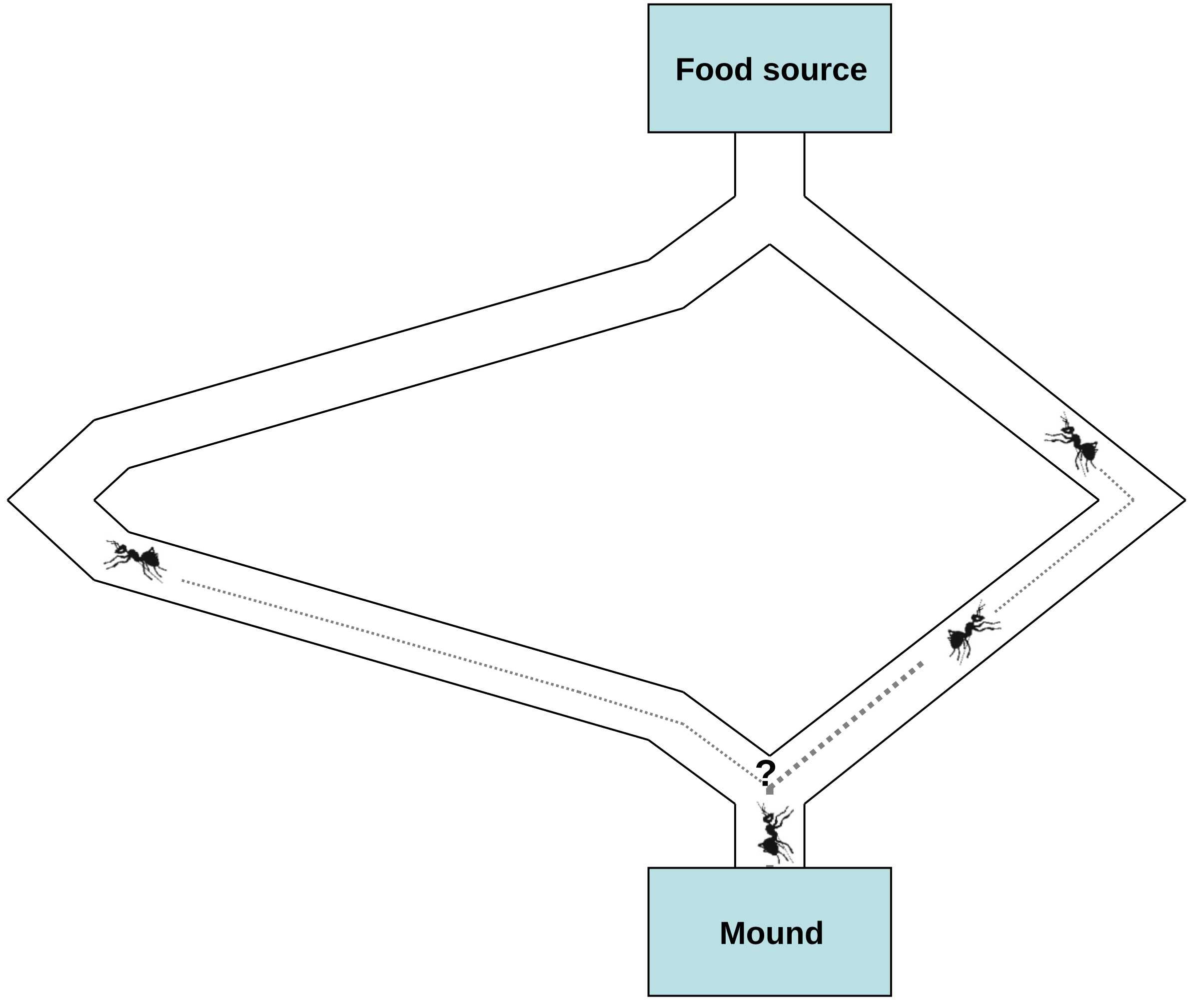


Finding the shortest path

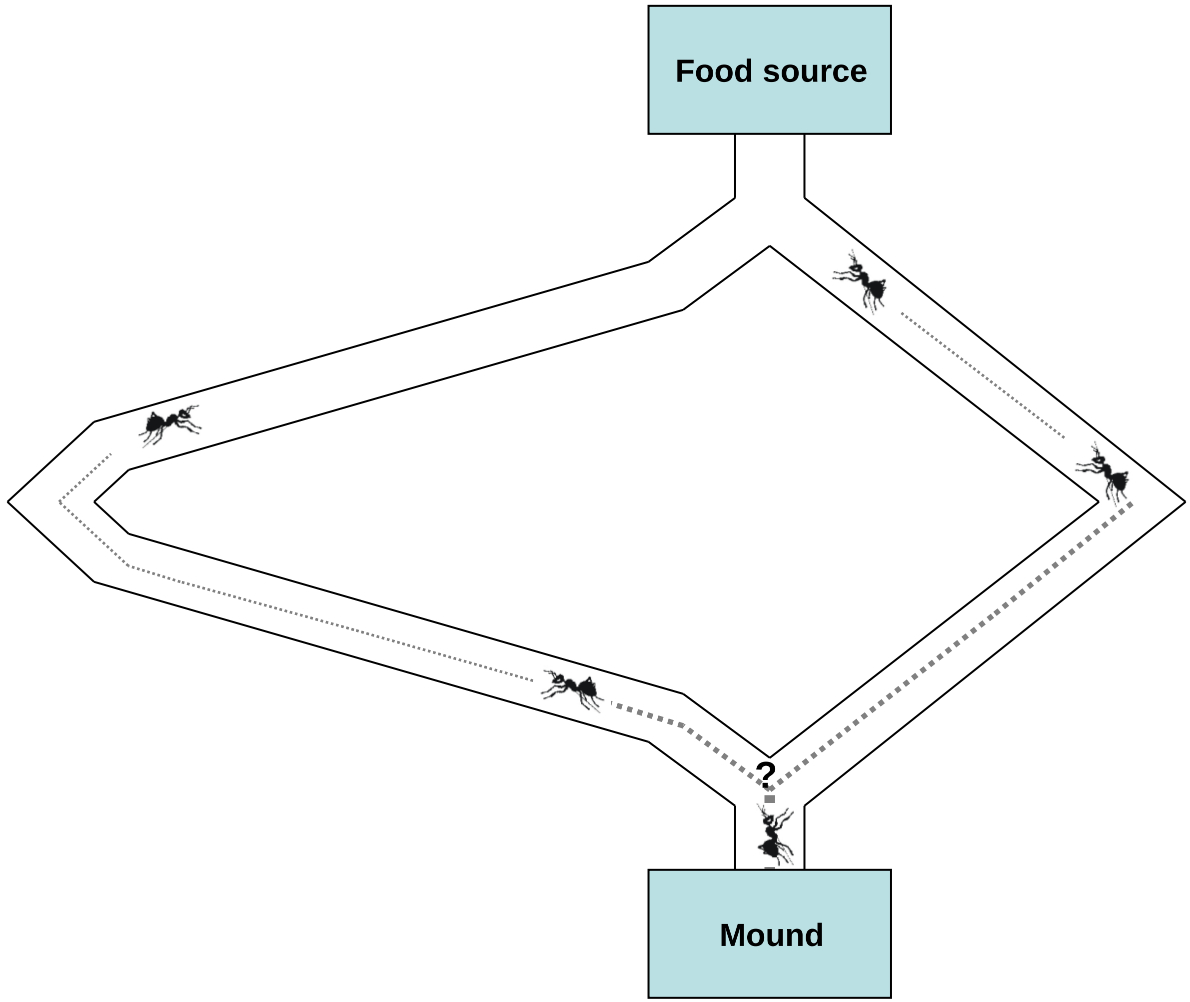


FACULTY OF !NFORMATICS

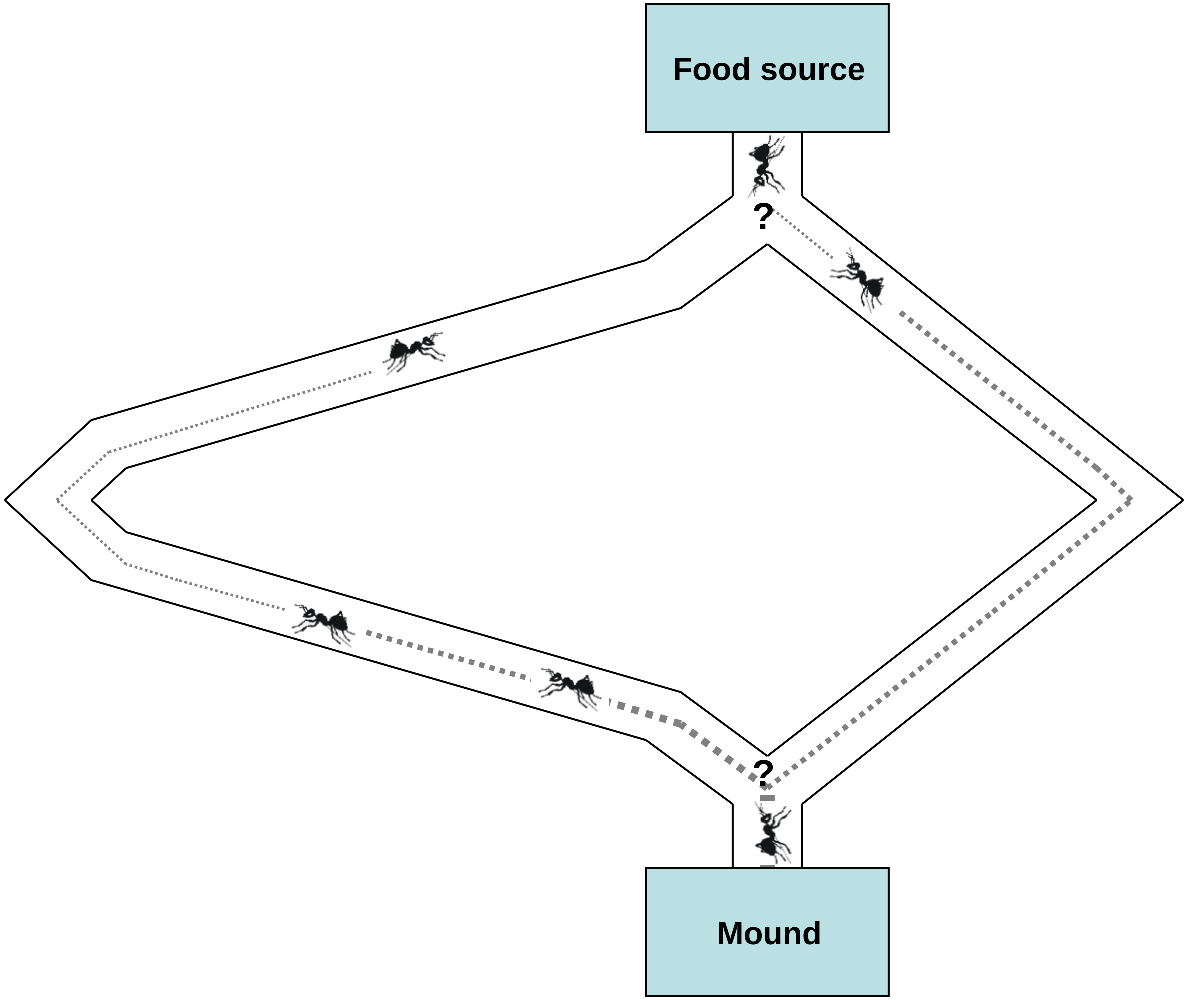
Finding the shortest path



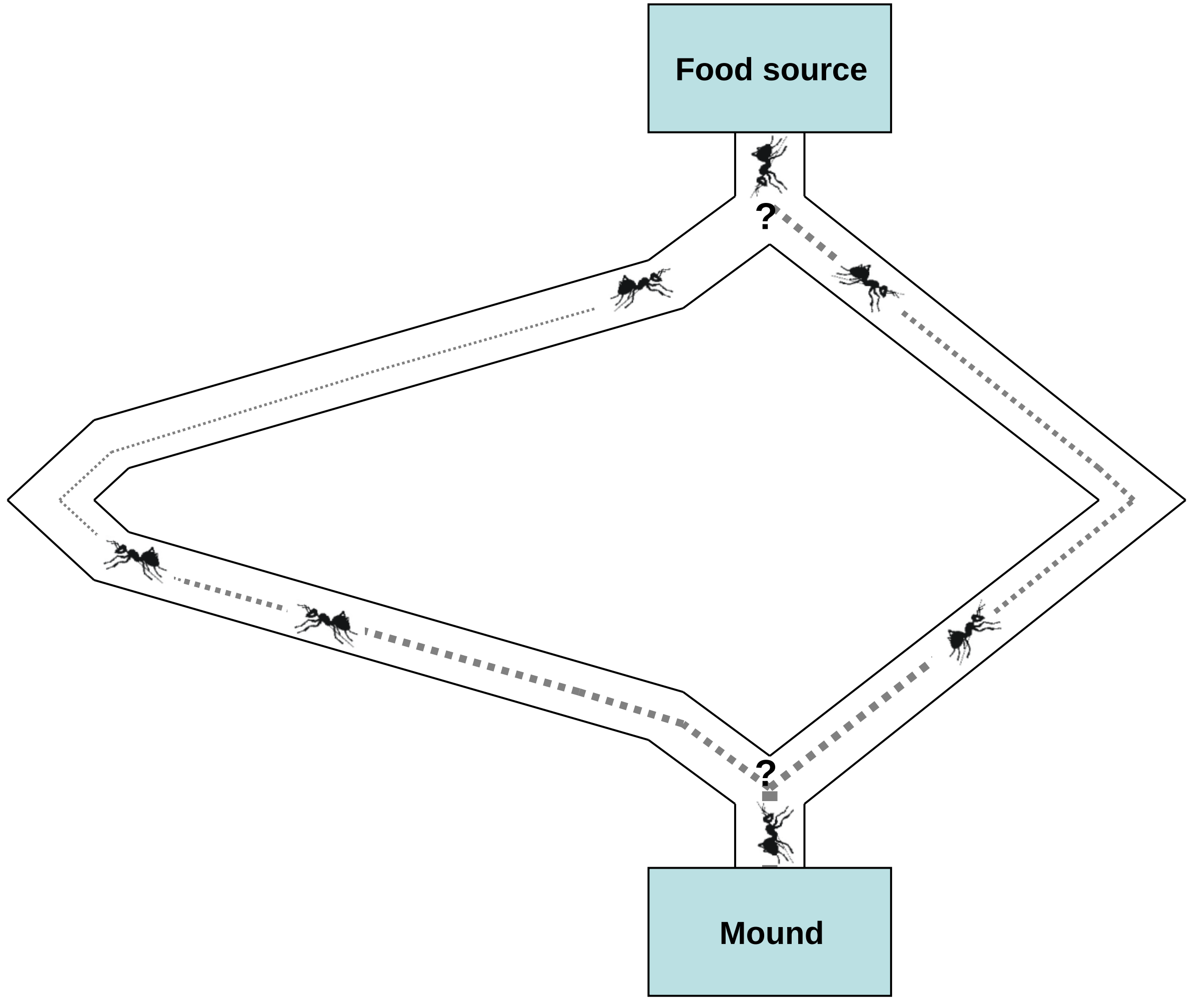
Finding the shortest path



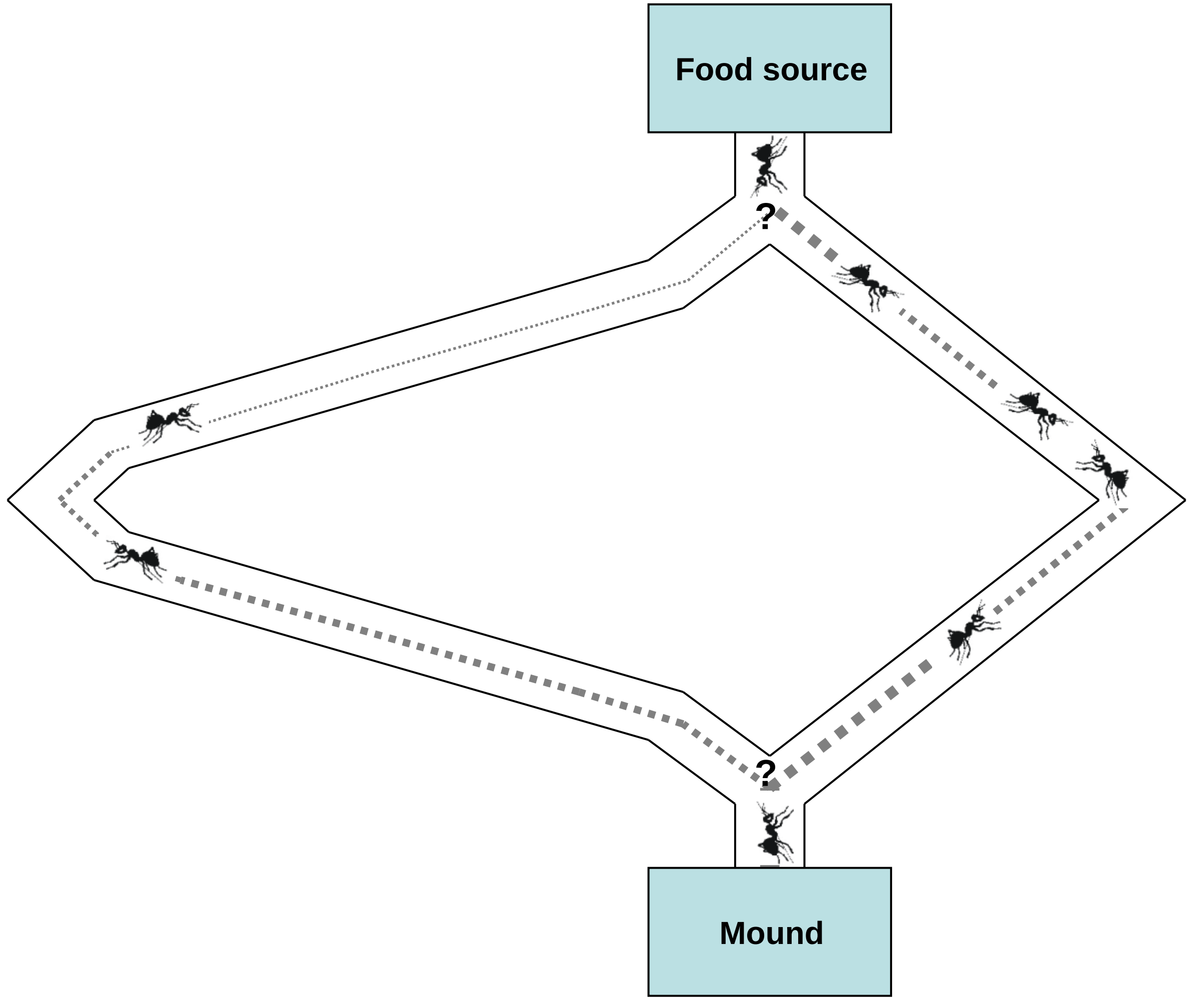
Finding the shortest path



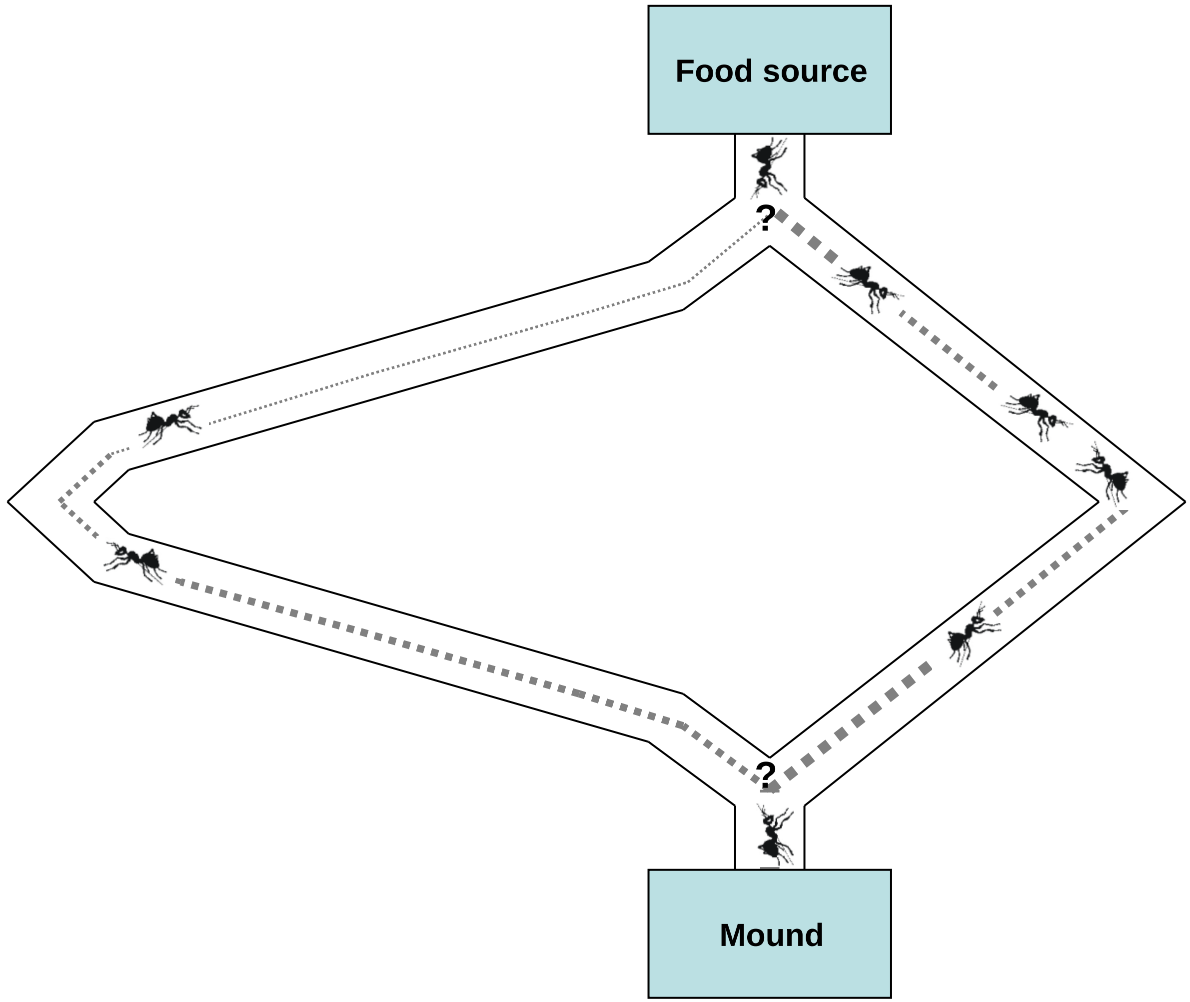
Finding the shortest path



Finding the shortest path

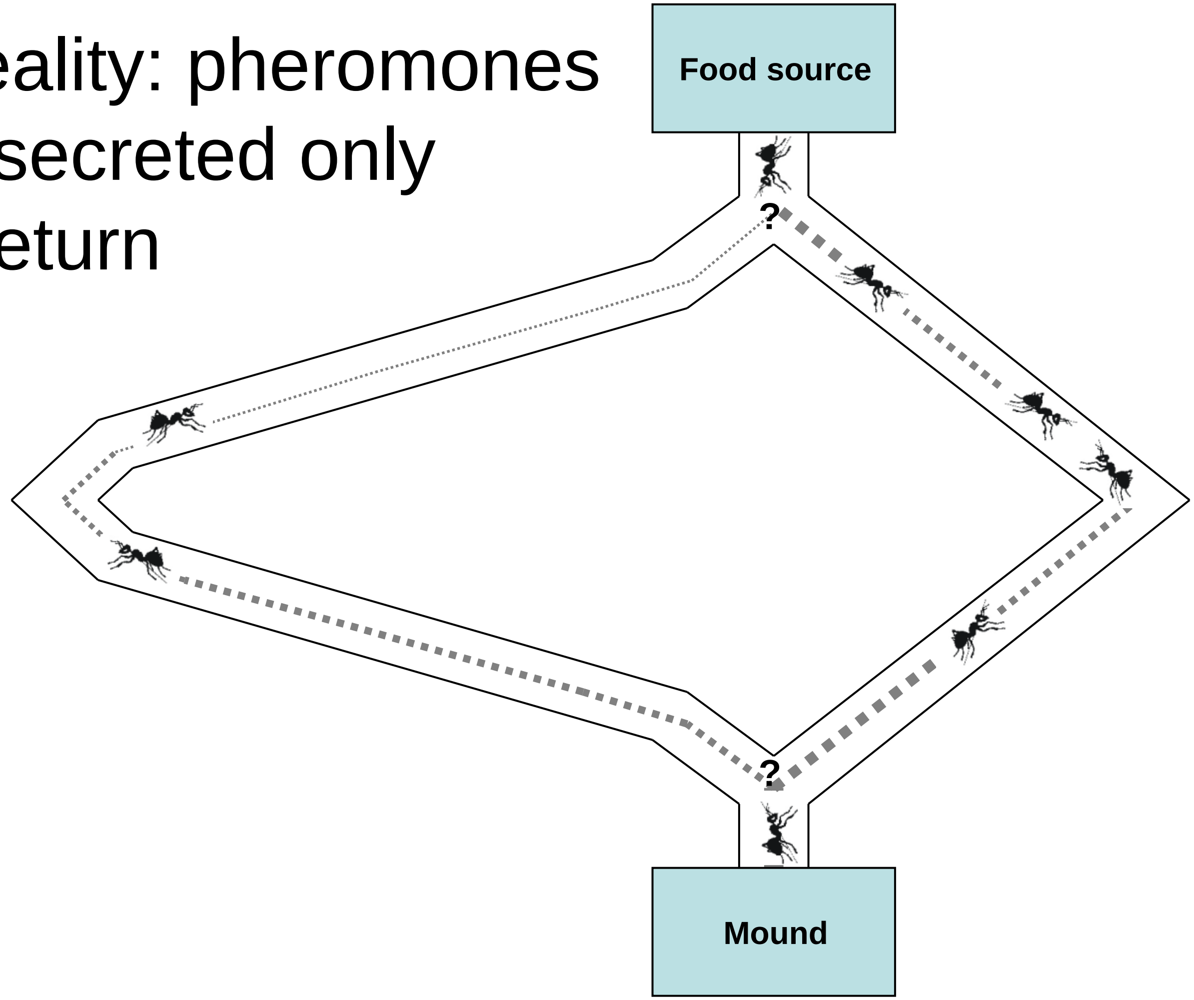


Finding the shortest path



Finding the shortest path

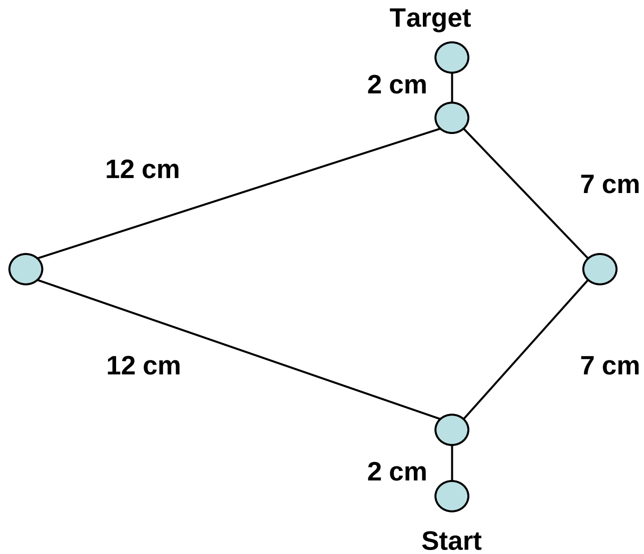
- In reality: pheromones are secreted only on return



Ant algorithms

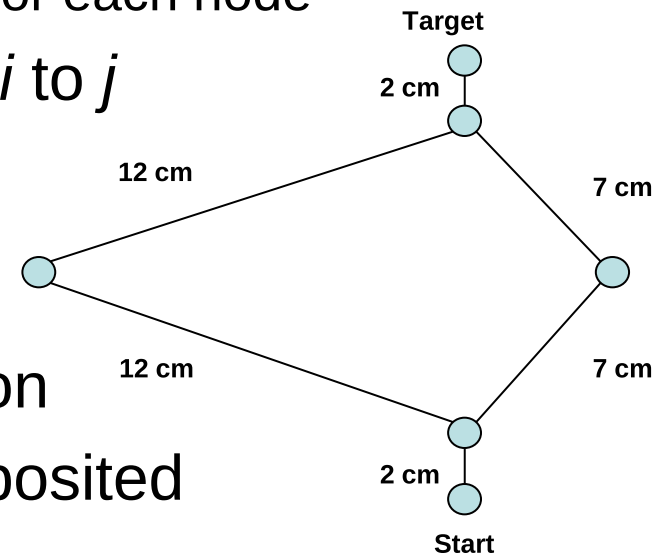
- Initial Algorithm: Ant System
  Colorni, Dorigo and Maniezzo, 1991

- Differences zu natural ants:
  – Artificial ants move only in one direction
  – Rule for decision
  – Pheromones are secreted after end of one iteration
  – *Natural ants don‘t optimise only by shortest path*



Ant System

- State transition rules
  - Decide to which node an ant will move
  - Prevent cyclic paths
    - keeping a list of potential transition nodes for each node
  - ➔ Probability *p* of *k-th* ant to move from *i* to *j*

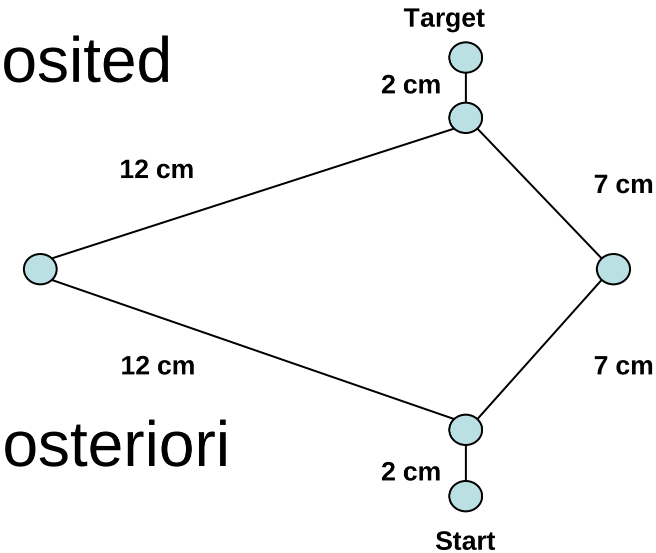


- Information available
  - **Local** ($\eta_{i,j}$): desirability of state transition
  - **Global** ($\tau_{i,j}$): amount of pheromone deposited

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}}$$

Ant System

- Information available
  - Local ($\eta_{i,j}$): desirability of state transition
    - A priori knowledge about the problem domain
    - Typically: length of path, i.e. $\eta_{ij} = \frac{1}{d_{ij}}$
  - Global ($\tau_{i,j}$): amount of pheromone deposited
    - A posteriori knowledge



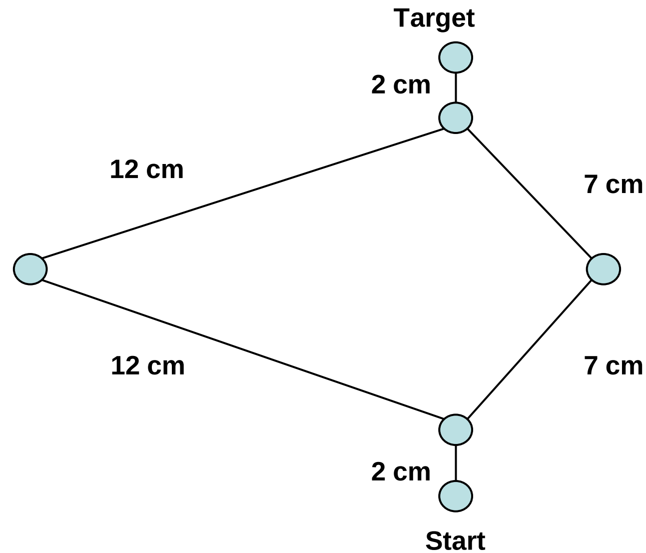
- Other parameters
  - α,β: control influence of a-priori vs. a-posteriori

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}}$$

Ant System

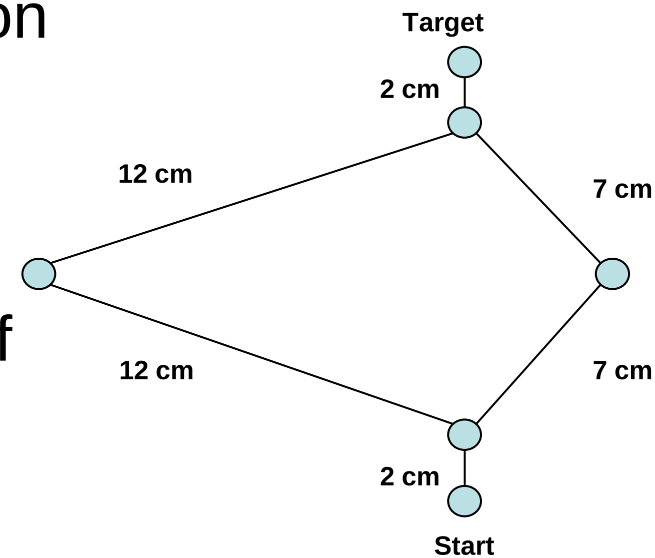
$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}}$$

1. Compute for every possible transition
2. Select according to probabilities



Ant System

- Update of pheromones
  - When all ants reached the target: pheromone trails are updated
  - Pheromones deposited on each edge used by an ant
  - Amount depending on quality of solution
    - Typically on length of path (L)
    - *Compare to fitness function in GA*



  - After each iteration a certain amount of pheromones vaporises: ρ

$$\tau_{ij,t+1} = (1-\rho)\cdot\tau_{ij,t} + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

(Sum of all newly deposited pheromones by all ants 1..m)

$$\Delta\tau_{ij}^{k} = Q / L^{k}$$

Q: Constant

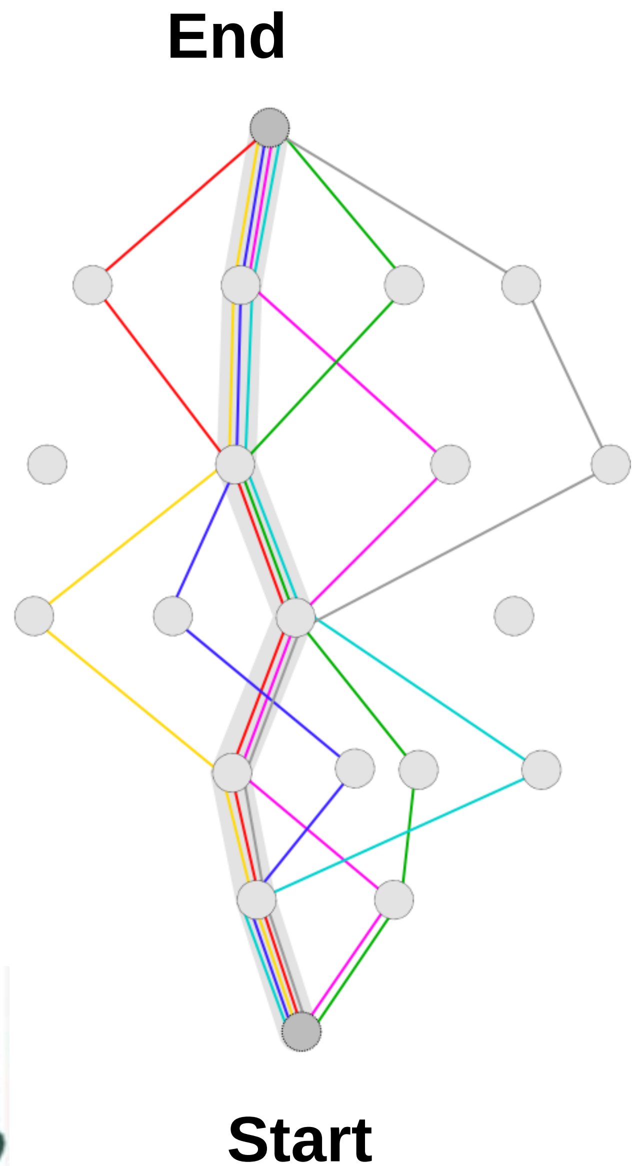
Ant System: Algorithm Recap

- Initialise all edges
- For t = 1 to tmax *(iterations)*
  - For k = 1 to m *(ants)*
    - Node i = Start node
    - While Node i not = Target node
      - Apply transition rule → move to next node
    - End While
  - End For
  - Compute path length of all ants, store shortest path
  - Apply pheromone update rule on edges
- End For

Ant System

- Often used to solve shortest-path problems

- Popular example: Travelling sales man
  - Ants don‘t travel from pre-defined start to end node
  - Goal reached when all nodes visited



Ant System

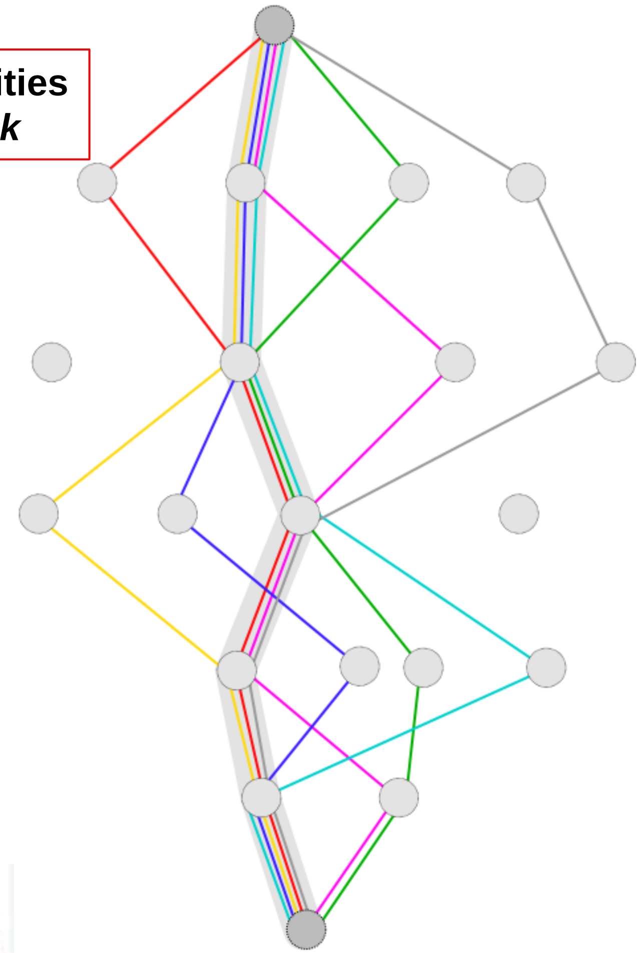
- Travelling sales man
  - Adaption of transition probability

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}} & \text{if} \quad j \in C_k(i) \\ 0 & \text{otherwise} \end{cases}$$

Set of remaining cities (nodes) for ant *k*

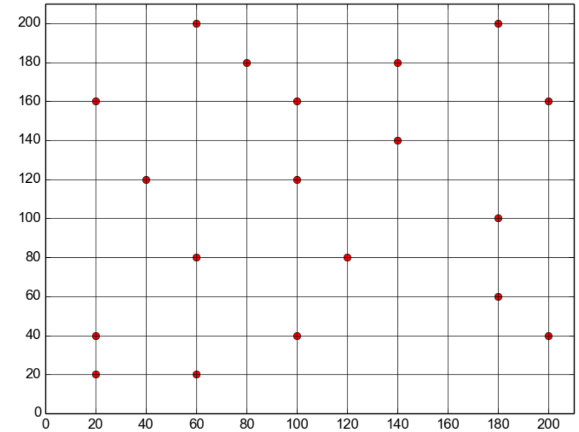
  - Goal reached when all nodes visited

End

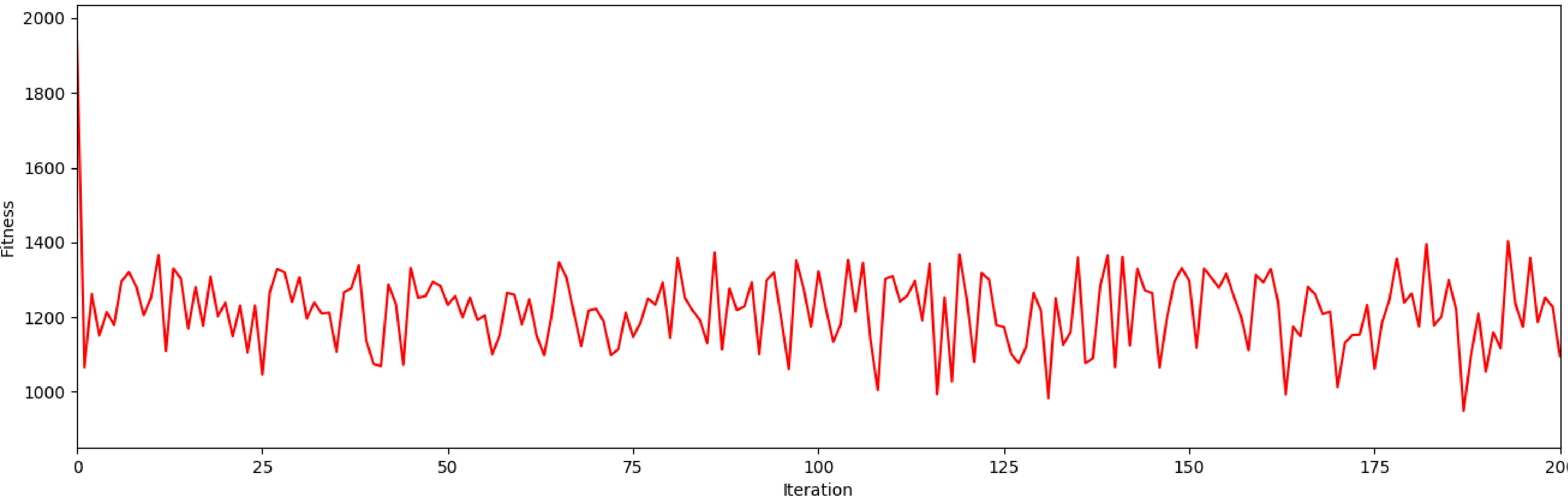


Start

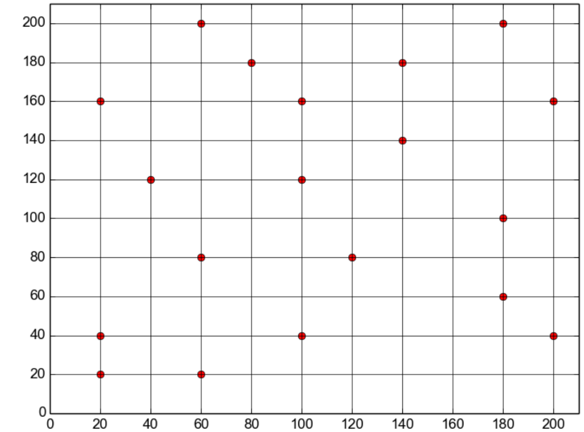
Ants on TSP problem



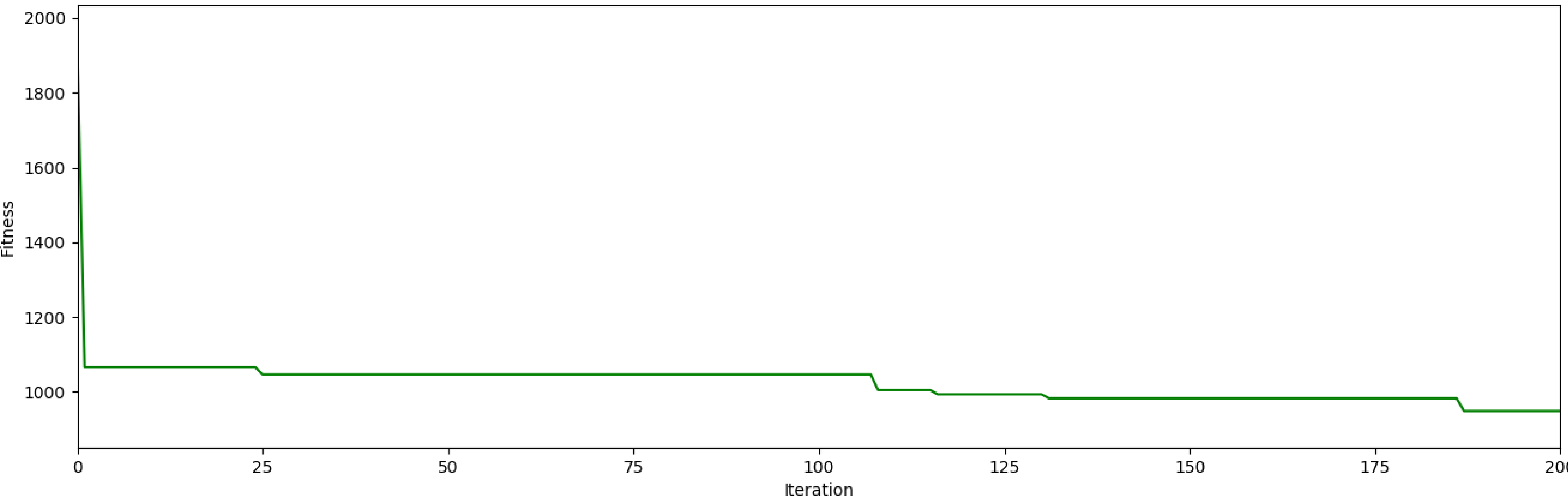
- Same optimisation example as with GA
  - Finds **good** solution very **fast**
  - Best solution fluctuates a lot over iterations
  - Relatively **slow improvements** from initial good solution



Ants on TSP problem



- Same optimisation example as with GA
  - Finds **good** solution very **fast**
  - Best solution fluctuates a lot over iterations
  - Relatively **slow improvements** from initial good solution
    - (only showing best solution)

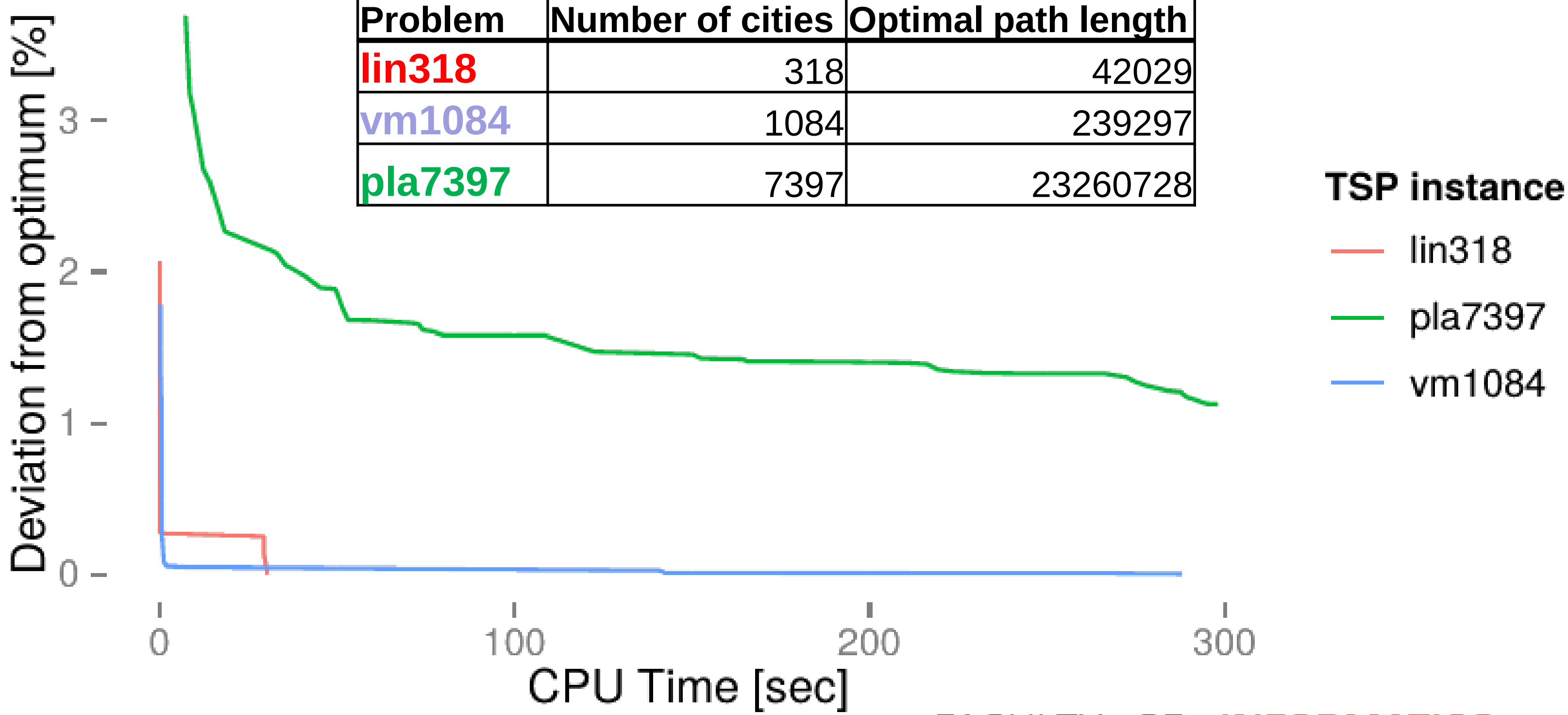


Ant System

- Problem:

  Works well for smaller tasks, not so good for larger problems with many nodes and edges

Problem	Number of cities	Optimal path length
lin318	318	42029
vm1084	1084	239297
pla7397	7397	23260728



Ant System

- Problem:

  Works well for smaller tasks, not so good for larger problems with many nodes and edges

→ Several extensions to the original algorithm:

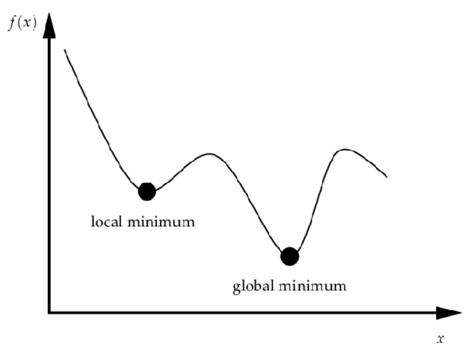
- AS with Elitist Strategy
- Ant Colony System
- Hybrid Ant System
- MAX-MIN Ant System

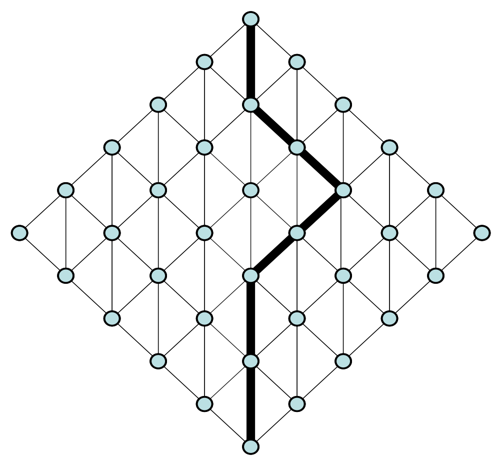
Ant System with „Elitist Strategy“

- Proposed by Colorni, Dorigo and Maniezzo
- Idea:
  - Currently best path contains edges of eventual optimal path
- Extension:
  - *e* ants additionally support the current best path

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij} \boxed{+\, e\cdot\Delta\tau^{e}_{ij}}$$

- Problem:
  - Too high value of e: → local optimum





Ant Colony System

- Proposed by Dorigo & Gambardella

- Idea:
  - Reduce effort for exploring new ways
  - But do that more efficiently

- Changes:
  - New transition rule, favouring current best solution
  - New pheromone update rule
    - Additionally: local pheromone update rule
  - Using candidate lists

Ant Colony System

- Pheromone update rule
  – In each iteration only the currently best solution is reinforced
    → Search focussed more on this neighbourhood

- Local pheromon update rule
  – After each state transition: pheromone trail of this edge is reduced
    → Currently used edge becomes less attractive, ants search elsewhere
    → Search space is expanded

Ant Colony System

- Transition rule:
  – Explore new paths

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}}$$

  – Utilise existing results

$$\arg\max_{u \in J_i^k} \left\{ [\tau_{iu}] \cdot [\eta_{iu}]^{\beta} \right\}$$

$$j = \begin{cases} \arg\max_{l \in J_i^k} \left\{ [\tau_{il}] \cdot [\eta_{il}]^{\beta} \right\} & \text{for } q \leq q_0 \\ S & \text{for } q > q_0 \end{cases}$$

- q: random number 0..1
- q_0: parameter (0..1): guide amount of exploration
  – *close to 0? ➔ little exploration, likely stuck in local optimum*
  – *close to 1? ➔ very similar to original ant system*

Ant Colony System

- Local update rule: applied after each state transition

  k: parameter, 0<*k*<1

$$\tau_{ij} \leftarrow (1-\kappa)\cdot\tau_{ij} + \kappa\Delta\tau_{ij}$$

- Global update rule: after all ants have completed route

  p: pheromone decay

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \boxed{\rho}\Delta\tau_{ij}$$

$$\boxed{\Delta\tau_{ij} = \begin{cases} (L_{gb})^{-1} & \text{if } (i,j)\in \text{global-best-tour} \\ 0 & \text{otherwise} \end{cases}}$$

L_{gb}: length of global-best-tour (from trail beginning)

Ant Colony System

- Candidate lists
  - For large-scale problems

  - Each node has a list of candidate solutions
    - E.g. list of preferred cities to visit
  - List contains the *cl* closes nodes, ordered by distance
    - E.g. 15-20 nodes

  - Transition rule applied only for the nodes in the candidate list
    - Unless?
    - No node from the candidate list can be visited

Hybrid Ant System

- Propose by Gambardella, Taillard & Dorigo
- Idea
  - Ant algorithms arrive relatively quickly at a good solution, but take long to find the optimum
  - Local search methods take relatively long for a good solution, but then find the optimum relatively quickly

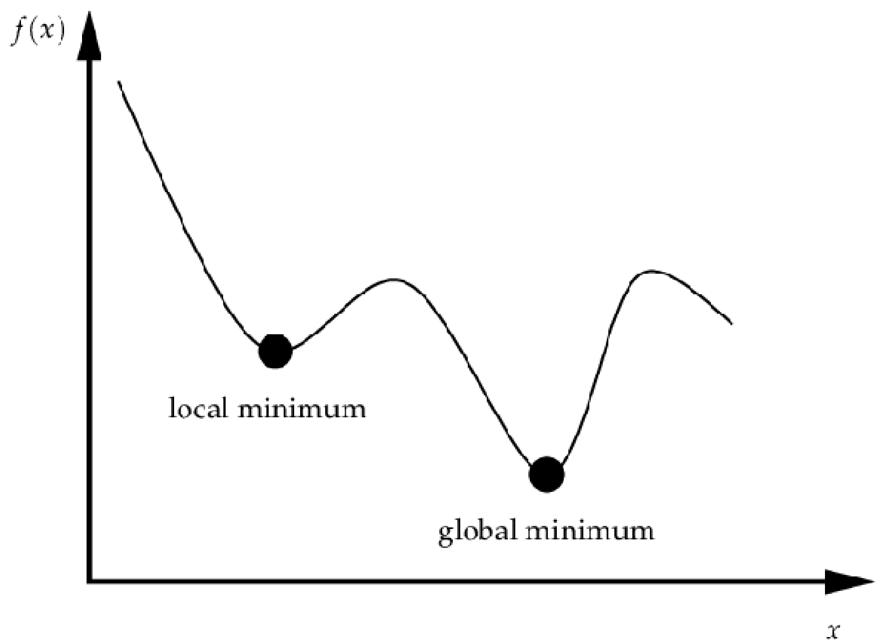
    → Combine both methods

- Extensions:
  - Integrate a local search method to ant system algorithsm

- Shows strong performance improvements

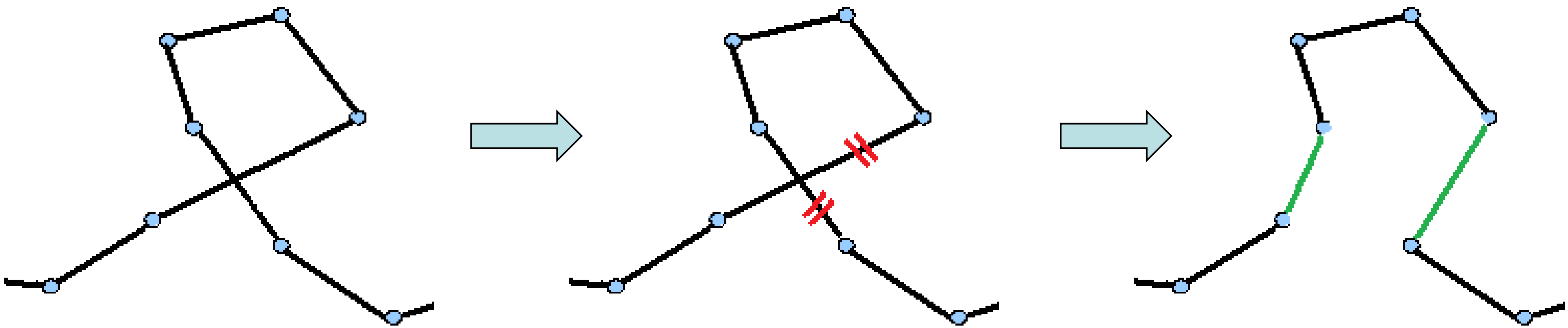
Local search

- Existing solution improved by iterative, small changes
- If no improvement possible – method stops
- Normally only discovers local optima



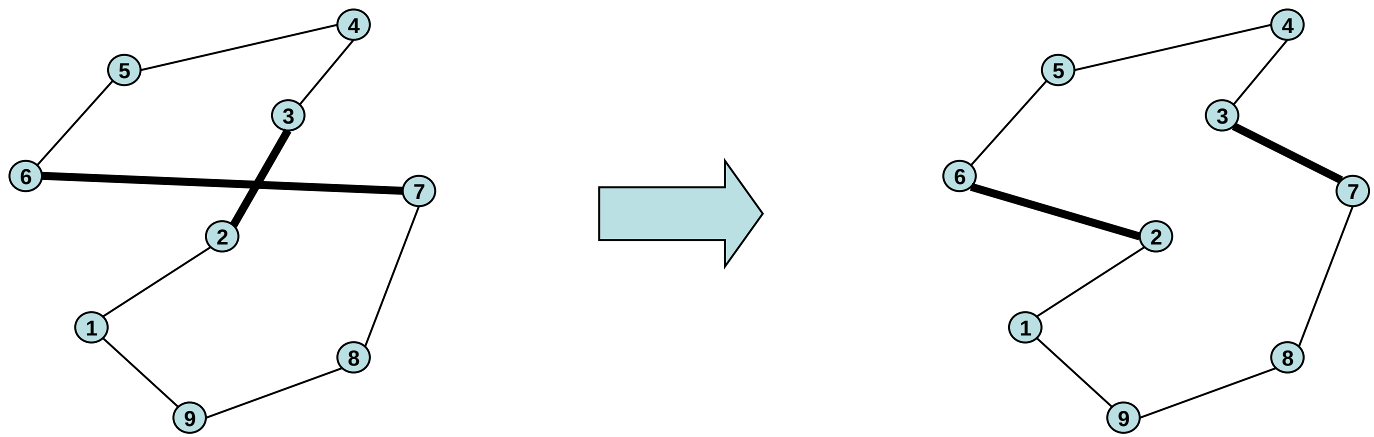
Local search

- Methods typically used with ant colony algorithms
  - 2-opt-Methode: reorder route so it doesn‘t cross itself
    - Delete two edges, and re-add them crossed-over



Local search

- Methods typically used with ant colony algorithms:
  - 2-opt-Methode: reorder route so it doesn‘t cross itself
    - Delete two edges, and re-add them crossed-over

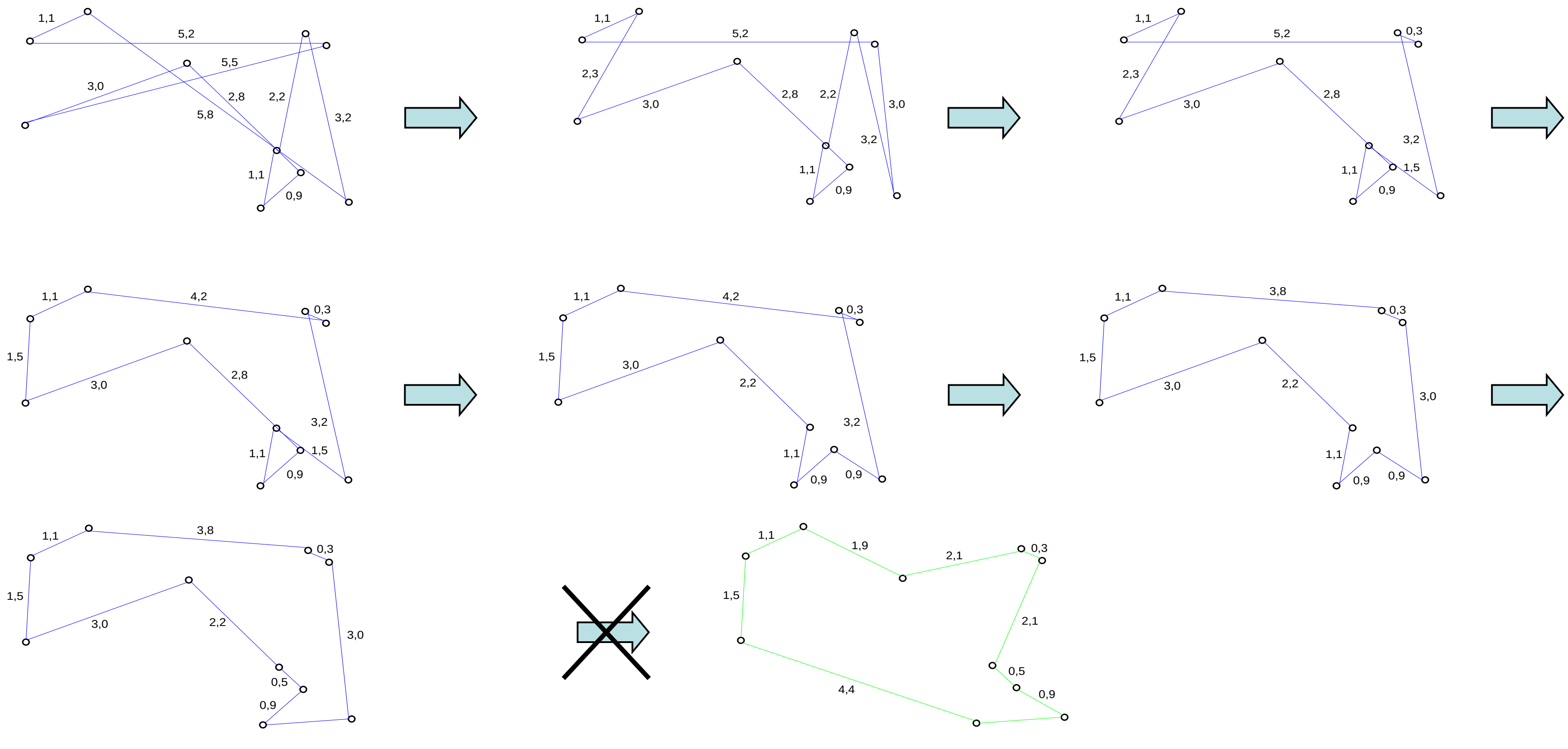


Local search

- Methods typically used with ant colony algorithms:
  - 2-opt method
    - Delete two edges, and re-add them crossed-over
  - 3-opt method
    - Delete three edges at a time
    - Reconnect in all possible ways
    - Chose best solution
  - Lin-Kernighan method
    - Decide number of replacements at each step

Local search

- Example: TSP



Length: 17,7
(found with local search)

Length: 16,2
(optimum)

Hybrid Ant System

- Application of local search variants: at various points and to various degrees possible
  - After each ant found a solution
  - Only for a specific percentage of solutions
  - Only on the best solution of a iteration

MAX-MIN Ant System (MMAS)

- Proposed by Stützle & Hoos
- Idea
  - Allowing only the best ant to update the trail can enhance solution-finindg performance
  - However, can lead to early stagnation

➔ Prevent too large or too small amounts of pheromones on the edges

- Too large amount ➔ local optimum
- Too small amount ➔ edge is never used in solution space

➔ No edge that is never / always chosen

MAX-MIN Ant System (MMAS)

- Extension:
  - Introduce explicit maximum and minimum edge strength
    - Parameters τ_{min} and τ_{max}
    - Depending on the average edge length
  - Initialise all edges with τ_{max}
    - Evaporation/decay after each iteration
  - Only edges along the best path are reinforced
  - Can still lead to a local optimum

    ➔ Integration of local search

MAX-MIN Ant System (MMAS)

- Can still lead to a local optimum
  - ➔ Integration of local search
  - – When trail branching is low (limited exploration)
    - ➔ Adjust trail (edge) intensities (“smoothing“)
    - • Proportional to the difference between τ_{ij} and τ_{max}
- Empiric evaluation: results compared to ACS
  - – MMAS doesn‘t always produce ***best*** result
  - – But in ***average***: results are better

Problem	ACS best	ACS avg	MMAS best	MMAS avg
eil51.tsp	426	428	426	426.7
kroA100.tsp	21282	21420	21282	21302.8
D198.tsp	15888	16054	15963	16048
ftv170.atsp	2774	2826.5	2787	2807.75
kro124.atsp	36241	36857	36416	36572.9

ACO-Algorithms

- Ant Colony Optimization
  – „Ant Colony Optimization“ introduced as term by Dorigo and Di Caro
  – **Umbrella** term for all ant systems, meta-heuristics

- „Most successful“ ACO algorithms:
  – ACS
  – MMAS

- Extensions can overcome some limitations, but – introduce new parameters that need to be carefully set

Application areas for ACO algorithms

- Combinatorial optimization problems
  - Scheduling
  - Vehicle routing problems: minimum cost set of routes
    - Serving all customers
    - Demand of each customer < vehicle capacity
    - Visited exactly once
  - Multi-depot vehicle routing problem (mutli depots to start from)
  - Periodic vehicle routing problem
  - Split delivery (demand can be > capacity)

Application areas for ACO algorithms

- *Generally: heuristic optimisation for NP-hard problems*
  - *Complexity rises exponentially with number of nodes*
  - *Solution can not be found with exthaustive search*
  - *Solution goodness can be validated*

- ACO: problem needs to be formulated as ***graph***

Dynamic Problems

- Structure changes during learning phase
  – Nodes and/or edges get deleted
  – Node and/or edges get added
  – Loacl information changed

- Examples:
  – Dynamic TSP
    - Construction sites, traffic conditions → edge value/costs **change**
    - Update of targets: **new** target nodes (new customers), **removed** targets (customer cancelled order)

Dynamic Problems

- After change of problem structure:
  one time update of pheromone amounts

- Different strategies for update
  - Proximity to changed node/edges not considered
  - Proximity to changed node/edges considered according to local information
  - Proximity to changed node/edges considered according to global information
- Amount of update
  - Too little: stay in local optimum
  - Too much: loss of performance

Example System: AntNet

- Di Caro & Dorigo, 1997

- Optimal routing of data packets in a network, considering the current load
- → distributed, dynamic problme

- Information for each node i:
  - Routing table, containing state transition probabilities $P^{i}_{j,z}(t)$
  - Matrix Γ_i, containing
    - estimated average time $\mu_{i \rightarrow z}$
    - variance $\sigma^2_{i \rightarrow z}$

Example System: AntNet

- Initialisation phase

  Ants → initialise routing table and matrix
- Main phase

  Ants → update routing table and matrix

  Data packets routed according to probabilities

- Types of ants:
  - Forward ants
  - Backward ants (have priority)

Example System: AntNet

- Algorithm
  - Peridocially on each node a “forward ant“ is placed, with a random target node
  - Ants decide upon routing table and local information on load of the next node
  - Each ant has table $S^k_{s \rightarrow z}$, which records after each transition the node number and the time of leaving starting node
  - Once at the target node → a “backward ant“ with table $S^k_{s \rightarrow z}$ is generated and sent to the starting node
  - Ant updates routing table and matrix on each node, using information from $S^k_{s \rightarrow z}$

  → Updates information on current load on the network

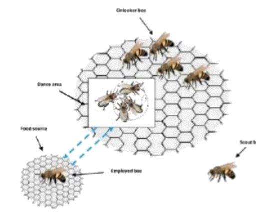
  More details in upcoming lecture

ACO: properties

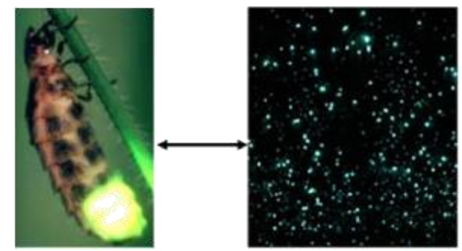
- Advantages of ACO algorithms
  - Can be adapted to various tasks
  - Can solve dynamic problems
  - Decentralised, with global state
  - Can find good solutions relatively quickly

- Disadvantages
  - Takes relatively long to find best solution
    → combination with local search, if applicable

Other swarm methods



- Particle swarm optimisation (~1995)
  - Particles moved around in search space
  - According to local knowledge and best global solution
- Artificial bee colony (~2000)
  - Example: Dynamic Server Allocation

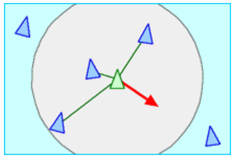


- Glow worm swarm optimisation (~2005)
  - Glow worms light according to their fitness; attract others
- Intelligent water drops (2007)
- Multi-swarm optimisation (multiple particle swarms)

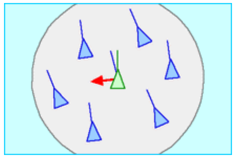
More in depth: upcoming lectures

Other swarm methods

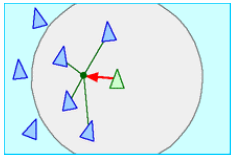
- Applications
  - Problem solving
    - Heuristic optimisation
  - Simulate (natural) behaviour
    - E.g. “Boids” (1986): simulation of bird flock behaviour
      - “Operations”
        - » Steer to avoid local overcrowding



        - » Steer towards direction of birds in proximity



        - » Steer towards average position of birds in proximity



      - Algorithm used e.g. in games & CGI film sequences
      - http://www.red3d.com/cwr/boids/

Questions

Agenda

- Recap
- Genetic Programming
- Ant Colony Optimisation
- Agents

Agent?

- Not a spy ☺
- Agente (italian; from agere (latin))
- Entities that
  - Live in the same environment with other agents
  - Communicate with them
  - Act targeted
  - Are autonomous and independent
  - *Can change their behaviour*

Agent Based Modelling

- Comparably young bottom-up modeling technique
  - Term ABM is only agreed upon since a couple of years
    - Other terms used (partially) synonymously: Artificial Life, Artificial Societies, Multi-Agent-Based System, Evolutionary Agent Models, Individual-Based Modeling, (Multi-Agent Systems), ...
    - Early example: Schelling's Segregation Model
  - Earlier models often were rather understood as extensions of other techniques, rather than being new (at the time)
    - Migration Models, Extended CA's, Traffic Microsimulation, Microsimulation with Interacting Agents, GIS with Intelligent Agents, …
  - Encompasses wide range of different computational models
    - Very different kinds of simulations with very different foci
    - Few Elements in common: Agents w. Interaction, Environment

Agent Based Modelling

- Goals: often simulation of systems
  - Robustness of systems is of prime interest
  - Ability to adapt to changed environment
- Sometimes used as umbrella term for various micro-scale methods
  - E.g. CAs, Game of Life, Ants/Swarms, ...
  - Agents: often no spatial restrictions
- Applications in biology, economics, social sciences, ...
- Several tools with integrated GUI: Anylogic, NetLogo, ...

Agent Based Modelling

- Agent is an autonomous, decision-making entity
  - Potentially individual rules
- “Bottom-up" approach

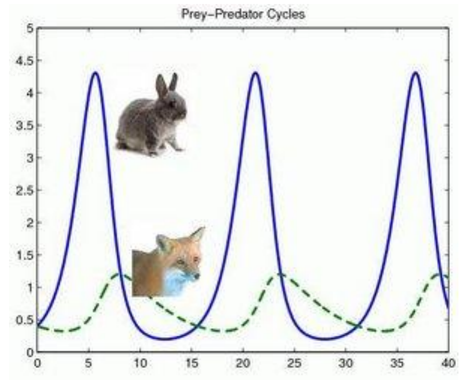
- Simulation / description of the behavior of (many) heterogeneous agents
- System behavior is not determined prior to simulation
  - Illustration of the aggregate behavior as an **emergent** result of the behavior of the agent population
  - Interaction between agents of main interest (often also the structure of this interaction → networks)

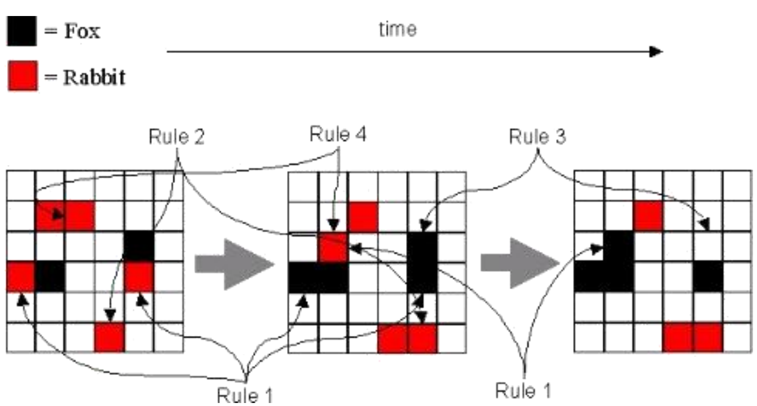
Agent Based Modelling

- Application scenarios
  - Modeling epidemics, analyzing different strategies like closing schools
  - Analyzing the consumer behavior and the influence of promotion campaigns
  - Social networks, e.g. spread of a rumor
  - Simulating traffic jams, airports, logistics

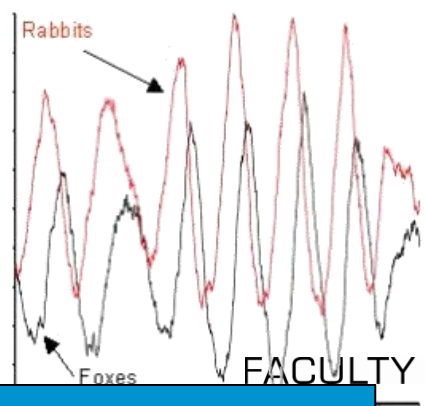
Agent Based Modelling vs. “traditional”

- Example: Predator-prey
  - Modelling approach via differential equations, e.g. Lotka-Volterra
    - Periodic, recurring activity
  - Modelling approach via ABM?
    - Using lattice as “living space”
    - Update rules
    - Occupy space
      - *Compare e.g. 2D CA* cell automata





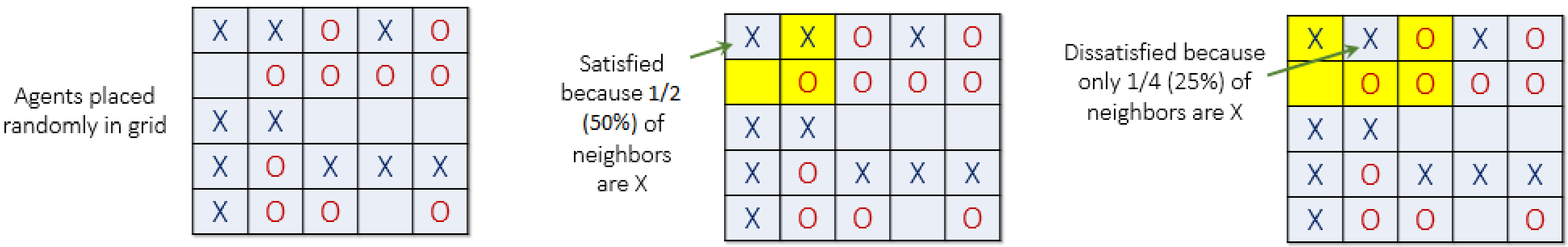
- Similar patterns
  - Less smooth
  - Can exhibit extreme cases



Agent Based Modelling – example

- Schelling's Segregation Model
  - Inspired by observed Community Segregation (esp. in US)
  - Tries to give a simple explanation for segregation
  - Simple assumption to be tested
    - Individuals like to have (some) similar neighbors

  - Rules:
    - Stay if at least a third of your neighbours are similar
    - Otherwise: move to a random new location (unoccupied)



Agent Based Modelling – example

- One iteration:

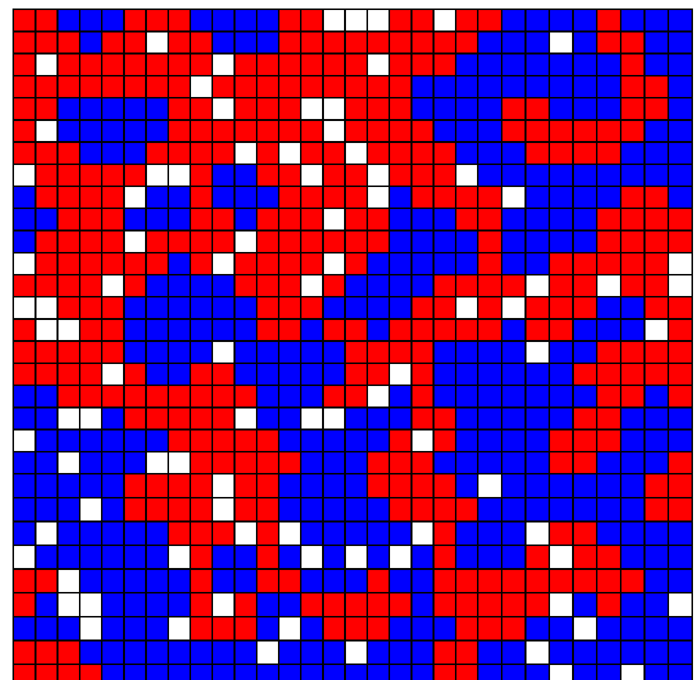
Dissatisfied agents marked with *

X	X*	O	X*	O
	O	O	O	O
X	X			
X	O*	X	X	X
X	O	O		O*

All dissatisfied agents relocated

X		O		O
O	O	O	O	O
X	X	X		X
X		X	X	X
X	O	O	O	

- Example on 30x30 grid, 10% empty cells



- *Is this a CA?*

Multi Agent Systems (MAS)

- A computerised system that is composed of multiple interacting intelligent agents [1]

- *Difference to what we talked about just before?*
  - Intelligence
  - Agents can have different behaviour / goals / ***owners***

- *MAS vs. Agent Based Modelling (ABM)?*
  - *Not a very clear definition / distinction, but:*
  - ABM is often interested in simulation; rather similar agents
  - MAS can also be directed towards solving specific problems

[1] Wooldridge, M. “Introduction to Multiagent Systems”. John Wiley and Sons, 2002.

Disambiguation

- Software Agents
  - Autonomous Programs (e.g. Bots)
    - May feature quite sophisticated AI logic
  - May communicate with identical or different programs
    - Part of **Multi Agent Systems**
    - But normally also a single Agent capable of work

- ABM – Agents
  - Only parts of programs/program constructs
    - Not individual programs
  - Normally restricted to a simulation model‘s environment
  - Usually employ very simple logic (heuristics, etc.)
  - Many Agents needed to reach useful results (Emergence)

Software Agents & Multi-Agent Systems

- An **agent** is a computer system that is capable of *independent action on behalf of its user / owner*
  - Figuring out what needs to be done to satisfy design objectives
  - Rather than being (constantly) instructed
  - Software, or also hardware (robots)

- A **multi-agent system** is one that consists of a number of agents, which *interact with each other*

- In the most general case, agents will be acting on behalf of **users** with different goals and motivations
  - *Represent their best interests while interacting with others*

Agents and Multi-Agent Systems

- MAS: distributed, artificial intelligence system
  - Embodies a number of **autonomous** agents within the same environment to achieve common goals
  - "Environment" e.g. physical environments for robotic agents, runtime environments for software agents, …
- Can be used to solve problems that are difficult or impossible for an individual agent/monolithic system to solve
  - E.g. optimisations in transport & logistics, film & media for CGI graphics

Software Agents vs. Objects

- Are software agents just objects by another name?
- Objects
  - encapsulates some state
  - communicates via message passing
  - has methods (operations performed on state)

- Main differences
  - *Agents are autonomous:* They decide for themselves whether or not to perform an action on request from another agent
  - *Agents are intelligent:* capable of flexible (reactive, pro-active, social) behavior
  - *Agents are active:* each agent is assumed to have active control

Characteristics of Intelligent Agents

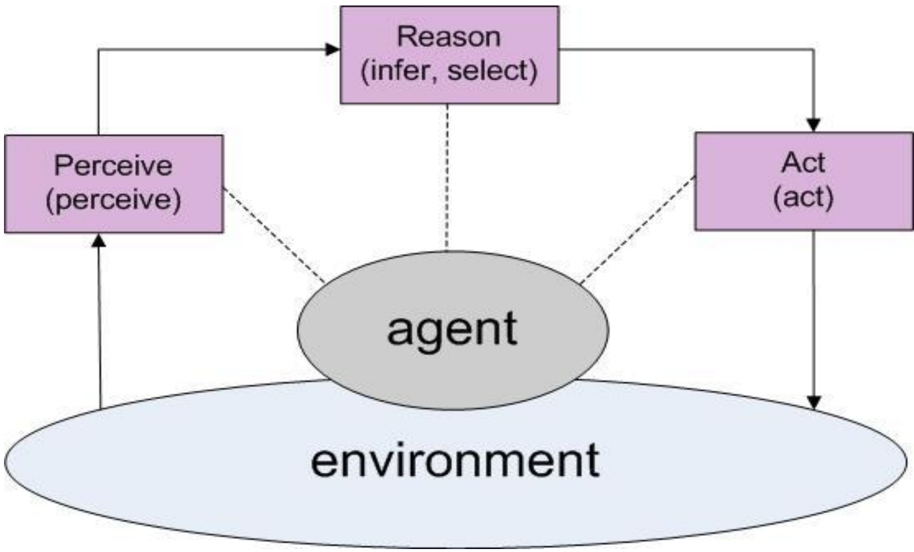
- Capable of **acting** in an environment
- **Communicate** directly with other agents
- Possess resources, skills and can offer services
- Capable of **perceiving** their environment (only partial / local)
- May be able to reproduce themselves
- Behaviour tends towards satisfying **objectives**

- *Autonomy: the agents are at least partially autonomous*
- *Local views: no agent has a full global view of the system*
- *Decentralization: there is no one controlling agent*

  ➔ **Self organisation**

Acting Cycle of intelligent Agents

- Agent starts reasoning process by interpreting arriving information (perceive)
- By combining this information with the existing knowledge and specified goals, the agent infers and selects possible actions (infer, select)



- Agent executes inferred and selected action (act) and changes the state of the environment

Physical Agents vs. Software Agents

- Physical agents or embodied agents
  - Interact with real world (sensors, actuators connected to real world)
  - Challenges: perception and action
  - Best known example: *Robots*

- Software agents: environment is virtual
  - Environment: single machine, intranet, internet, …
  - Interact with other software agents, with SW modules, services
  - Interact with humans through human interfaces

Software Agents vs. other methods

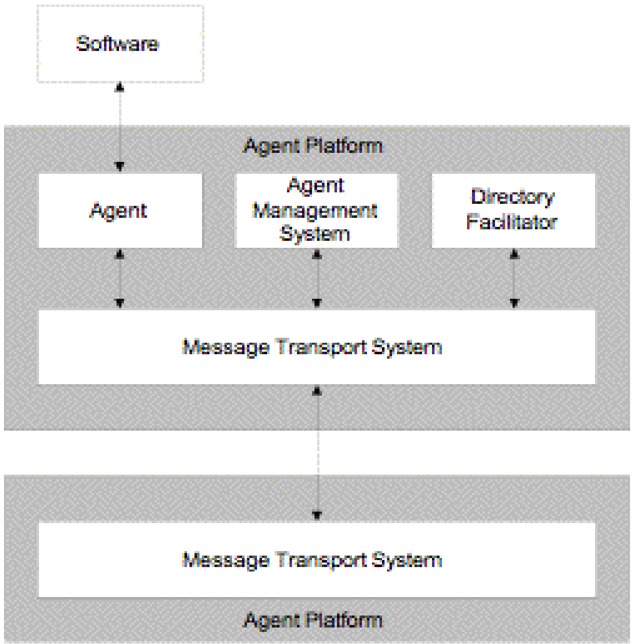
- CA/GA/Ants: homogenous
  - Generally one type of agent
  - Behaviour is predefined / the same for all agents
  - Behaviour generally doesn't change over time
  - Can have a spatial restriction (e.g. lattice in CAs)

- MAS: heterogenous
  - Agents can be of different type, change their behaviour, have different goals ➔ *what challenges does that entail?*

  ➔ Requires **organisation & governance of agents!**
    - Communication standards, protocols, ...
    - Middleware to support agent systems – e.g. FIPA standard
    - Cooperation between agents…

FIPA Standard

- "Foundation for Intelligent Physical Agents"
  - IEEE Computer Society standards organization
    - http://www.fipa.org
- Specifications for heterogeneous and interacting agents & systems
  - Coordination
  - Communication (protocols, syntax, …)
  - …



FIPA Standard

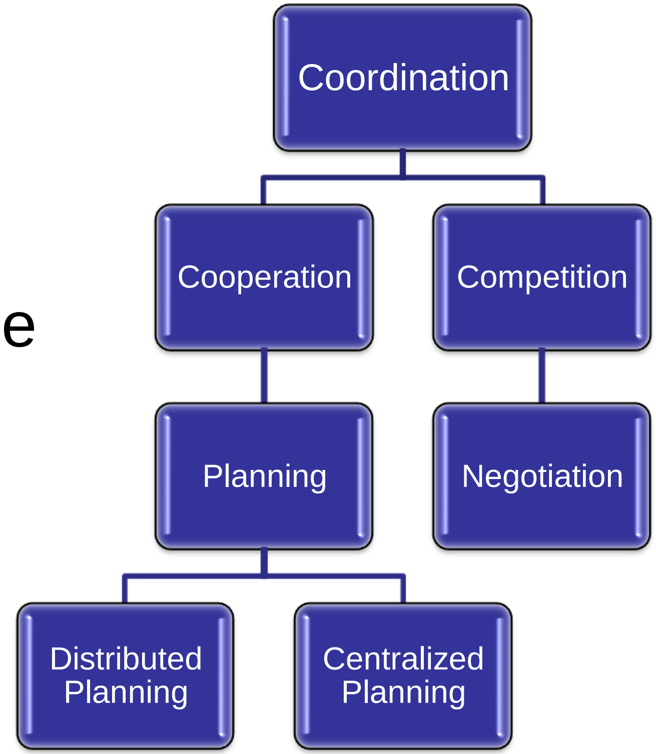
- Agent communication
  - Four levels in communication
    - Message Semantics
    - Message Syntax
      - Message structure: Agent Communication Language (FIPA standard)
        - » Defines a set of “performatives” (communicative verbs, e.g. ask, request)
      - Content codification: Content Language
    - Interaction protocol: How are conversations structured?

      ➔ Agent Communication Protocols

      e.g. ***Request / Request-When*** / ***Query*** / ***Propose / Contract***
    - Transport protocol: How messages sent & received by agents

Coordination

- Ability to communicate with other agents (*social ability)*
  - *Not only* to share information but to **coordinate actions in order to achieve goals**
- Degree of coordination depends on:
  - Inability of each individual agent to achieve whole task(s)
  - Need to reduce/optimize resource usage
  - Need to keep some conditions holding



- Coordination becomes critical when agents are **heterogeneous and autonomous**

Social Models for Coordination

- One source for inspiration to solve coordination problems are human societies

- There are several social abstractions that have been introduced in Multi-Agent Systems
  - *Trust and Reputation*
  - *Social Structures and Social Roles*
  - …

Coordination: Trust

- Different levels
  - User confidence: Can we trust the user behind the agent?
  - Trust of users in agents
  - Trust of agents in agents: Reputation mechanisms, Contracts, …

- *In an open environment trust is not easy to achieve; why?*
  - Agents introduced by system designer can be expected to be nice and trustworthy, but this cannot be ensured for other alien agents
  - Alien agents may give incomplete or false information to other agents or betray them to fulfill their individual goals

➔ Create competitive systems where each agent seeks to maximize its own expected utility at the expense of others

Coordination: How to compute trust

- Externally defined
  - By system designer: trust values are pre-defined
  - By human user: introduce trust values about humans behind other agents
- Inferred from existing representation about interrelations
  - Communication patterns, cooperation history, …
- Learnt from current and past experiences
  - In/decrease trust value for a specific agent if it behaves properly with / defects us
- Propagated or shared through a MAS
  - Reputation mechanisms

Other aspects of MA systems

- Communication methods
  - Blackboard systems
    - Agents communicate information through a common data structure, accessible by everybody
  - Message passing
    - Agents communicate directly by means of messages

- Coordination structures
  - Centralized: specific agent that ensures coordination
    - Some kind of control on other agents' goals / work assigned to agent
    - Coordinator becomes a single point-of-failure for the system
    - Other agents have loss of autonomy
  - Distributed: distribute work & control among all agents
    - *Internalize control in each agent*
      - Needs reasoning and social abilities to understand intentions and knowledge of other agents plus the global goal of the society
      - Becomes unrealistic when conflict resolution is difficult

Applications: Collective robotics

- Assembly of robots acting in cooperation in order to reach a goal

- ***Mobile robotics***
  - Uses at least two robots, which coordinate their movements to accomplish their tasks
  - MAS consists of robots which can be interpreted as agents

- ***"Cellular" robotics***
  - Construction of robots on a modular basis
    - Each module of the robot is considered as a part of the multi-agent system and regarded as an agent
  - Movement is result of the coordinative work of the agents building up the robot to fulfil the actual needed movement

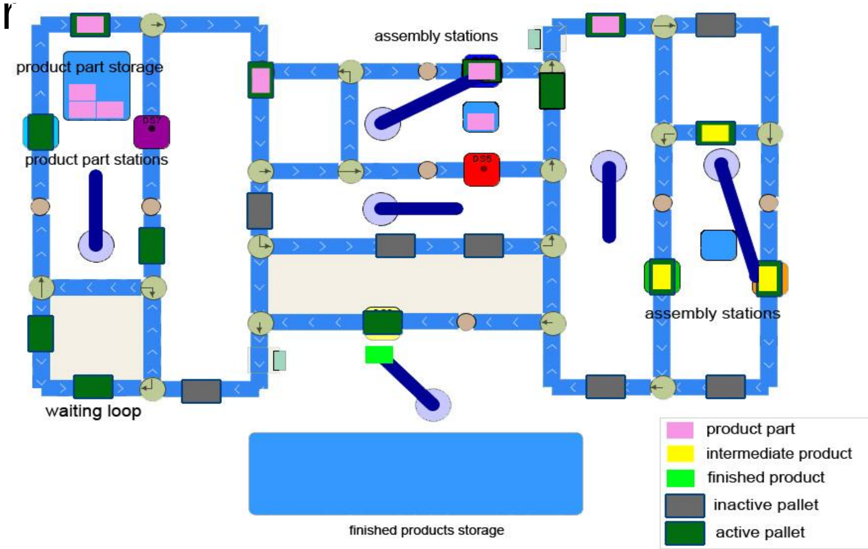
Applications: MAS Simulation

- Analysing the properties of theoretical models for real-life scenarios
  - Try to explain or forecast problems by testing the constructed or designed models in a virtual world
  - Compare and interpret the calculated results or phenomena

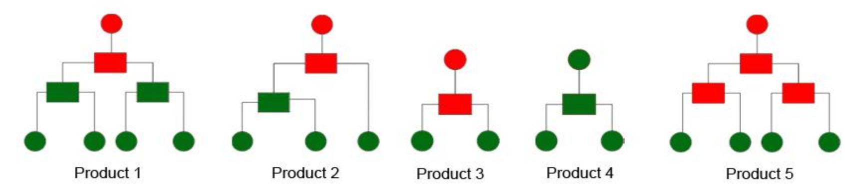
- E.g., simulation of production processes
  - MAS simulation allows a relatively simple reproduction of existing or planned assembly lines
  - Provides a practicable way to identify possible problems of the production plant design or improve the construction of the needed assembly line to optimize the throughput and production capacities

Applications: Assembly Workshops

- Analysing the properties of theoretical models for real-life scenarios
  - Explain/forecast problems
- E.g., simulation of production processes
  - MAS simulation allows a relatively simple reproduction of existing or planned assembly lines
  - Identify possible problems of the production plant design or optimize the throughput and production capacities



- Assembly Workshop consisting of heterogeneous components, agent-controlled
- Message-based coordination of agents to cope with failures or disturbances.
- Changing production requirements need workshop reconfiguration
  - Validation on hardware test bed is expensive
  - Mission-critical complex system: hard to model, predict and control due to dynamic and emerging system properties



Applications: Building artificial worlds

- Construction of synthetic worlds makes it possible to analyse certain interaction mechanisms in a detailed way

- Constructed worlds can be manipulated with different parameters to create the demanded environment
  – allows to analyze interesting behaviours and situations which correspond to the real world without impacts from the outside

- "Social Simulation"
  – modeling or simulation of social phenomena (e.g., cooperation, competition, markets, social networks dynamics, etc.)

Applications: CoGNETs

- Intelligent Infrastructure Management for autonomous IoT-to-Cloud computing
- Empowers devices to self-organise into dynamic swarm continuums for optimal data processing and service delivery
- Decentralised broker architecture and game-intelligent agents
- Use-cases: manufacturing, mobility, and health

- https://cognets.eu/

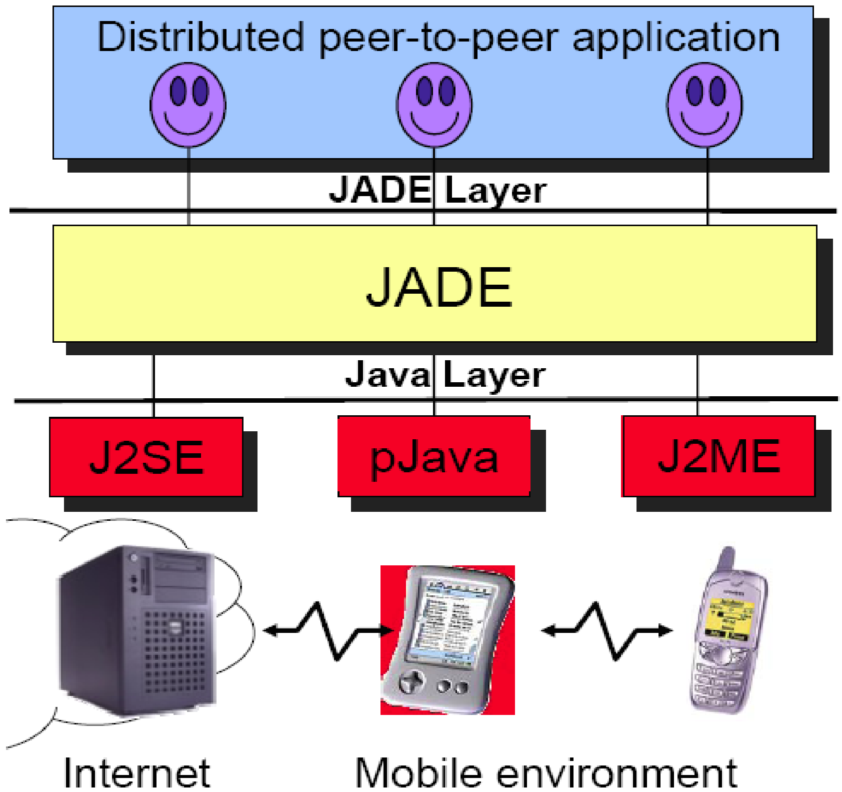
CoGNETs
Continuums of Game Nets

Applications: Crowd Simulation

- MASSIVE: http://www.massivesoftware.com/

Implementing FIPA: Middleware

- Simplifies implementation
  - JADE: development & runtime execution of peer-to-peer intelligent-agent applications
    - JADE Tutorial in TUWEL
    - http://jade.tilab.com/doc/index.html
    - API doc: http://jade.tilab.com/doc/api/index.html



  - Other implementations exist as well, e.g. SPADE (https://github.com/javipalanca/spade)

Different Views / Terminology / Definitions

- *Agents as a **tool for understanding** (human societies)*
  – Agent based modelling, understand various kinds of social processes
  – Similar to using neural networks to understand the human brain, rather to use it for e.g. prediction tasks

- *Agents as tool to solve specific task (e.g. optimisation)*

- *Agents as a paradigm for software engineering*
  – Software engineers have derived a progressively better understanding of the characteristics of complexity in software
    - Widely recognized that *interaction is probably the most important single* characteristic of complex software
    - Major research topic: development of tools & techniques to model, understand, and implement systems in which interaction is the norm
  – ***Agent-Oriented Software Engineering***

MAS as Computer Science progression

- Wooldridge: programming has progressed through:
  – machine code;
  – assembly language;
  – machine-independent programming languages;
  – sub-routines;
  – procedures & functions;
  – abstract data types;
  – objects;

to *agents*

[Wooldridge, M. “Introduction to Multiagent Systems”. John Wiley and Sons, 2002.]

AI Agents / Agentic AI

- AI Agents as term also popularised in the context of large-language models (LLMs)
  - “Autonomous software tools that perform tasks, make decisions, and interact with their environment intelligently and rationally. They use AI to learn, adapt, and take action based on real-time feedback and changing conditions. AI agents can work on their own or as part of a bigger system, learning and changing based on the data they process.”
  - Google Project Mariner, Perplexity Shopping Agent, …
  - Multi-agent systems: LLM agents that are specialised for certain tasks

AI Agents / Agentic AI

- Similar issues as before
  - Needs protocols to access data, for exchanging context, for cooperation, …
  - Agent Protocol (LangChain)
  - Model Context Protocol

Questions ?