Guest lecture for Transport Engineering

Sebastian Iwanowski
FH Wedel, University of Applied Sciences

The Vehicle Routing Problem – How to solve this with Ant Colony Optimisation

VRP – How to solve this with ACO

1. Problem Definitions: VRP, TSP, ACO
2. Solving TSP with ACO – simple ideas
3. Solving VRP with ACO
4. Example from Practice
5. Complexity issues – what makes TSP like problems prone to ACO?
6. Another example from practice elaborated at FH Wedel

What is VRP (Vehicle Routing Problem)?

Supplier has set of trucks.

Supplier must deliver goods to certain customers.

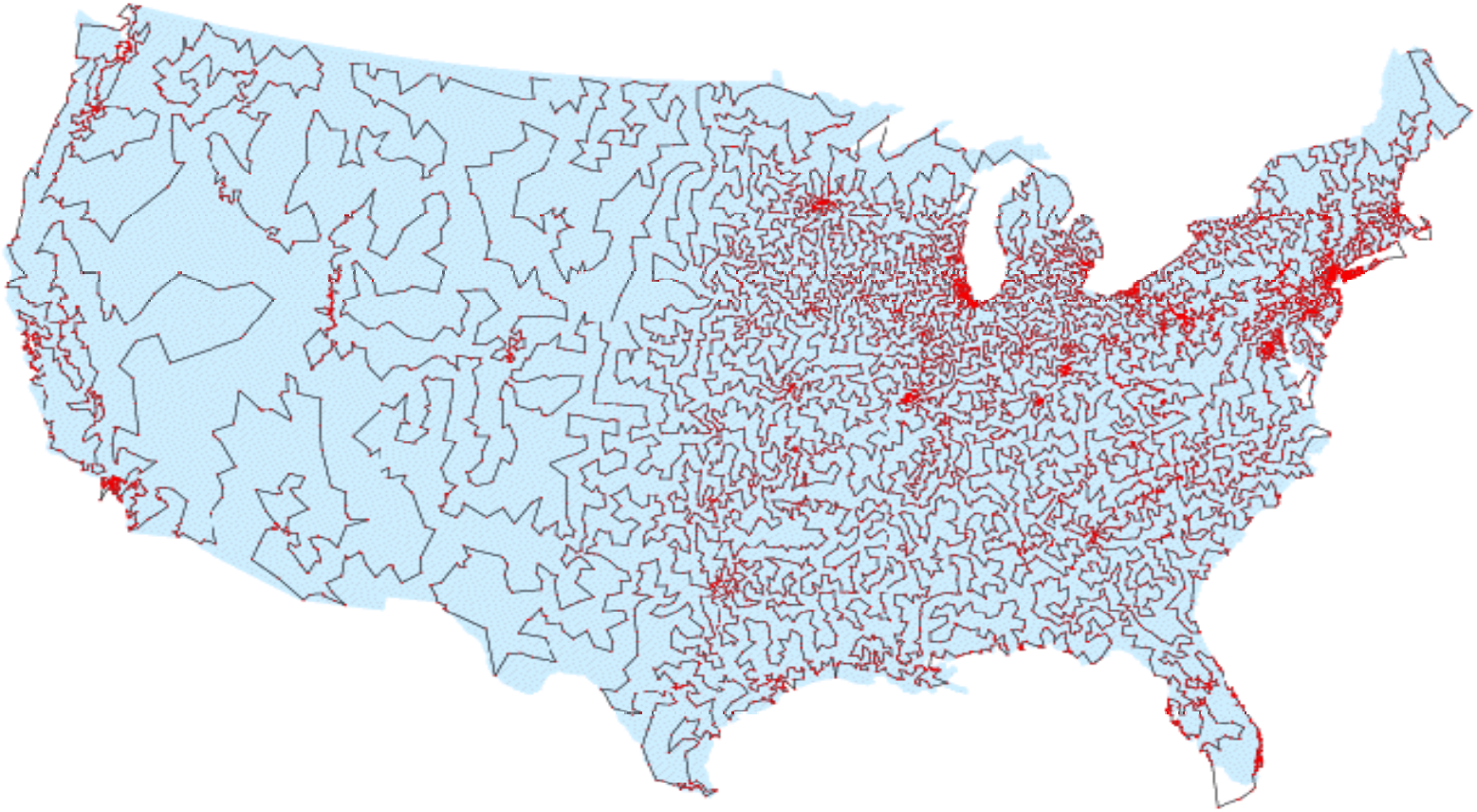
The set of customers is assigned to set of trucks such that each customer is supplied by exactly one truck.

Each vehicle starts at the supplier‘s depot and visits each customer assigned to itself exactly once, delivers the required goods and finally returns to the depot (TSP)

What is TSP (Traveling Salesman Problem)?

Given certain targets on a map, find shortest round trip meeting each target exactly once.

The order of the targets visited does not matter (in the original TSP).

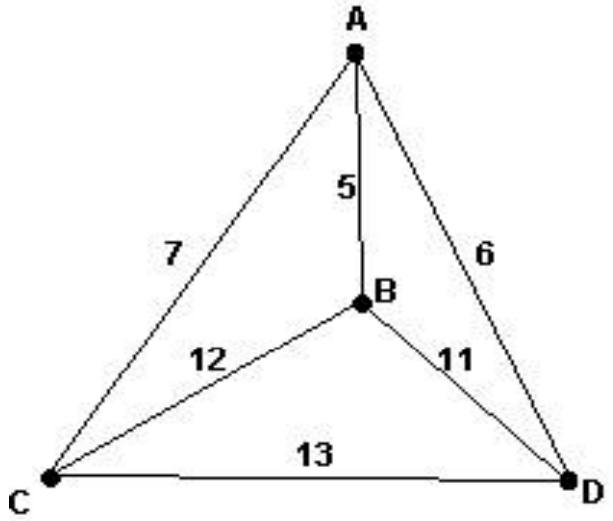


TSP on a Graph

Given Graph $G = (N, E)$.

N is the set of nodes, E is the set of edges.

Each edge $(i, j) \in N$ has a length d_{ij}. $(i, j \in N)$



Solution for the shortest path: ACBDA (36)

What is ACO (Ant Colony Optimisation)?

Ant Colony Optimization (ACO) is a class of algorithms for combinatorial optimisation problems resembling the way ants would solve that problem.

Models in nature are ants seeking food.

ACO principles

Ants mark their paths by pheromones:
The more ants are walking, the higher is the pheromone intensity

Other ants follow preferably more intense pheromone traces.

Short paths will be used more often which makes their pheromone track more intense
→ Efficient paths are more attractive.

Not all ants follow the old tracks
→ New (better) paths have chances to be found.

VRP – How to solve this with ACO

1. Problem Definitions: VRP, TSP, ACO
2. Solving TSP with ACO – basic ideas
3. Solving VRP with ACO
4. Example from Practice
5. Complexity issues – what makes TSP like problems prone to ACO?
6. Another example from practice elaborated at FH Wedel

ACO main procedure for TSP

1. Initialise parameters und pheromones.

2. Repeat as long as termination criterion is not satisfied:
   I. Generate ants and let them find a complete tour considering the current pheromone distribution.
   II. Do some optimising work
   III. Update pheromones.

1. TSP with ACO: Initialisation

Distribute pheromone intensities uniformly **to all edges** of the network.

The ants following will drop their pheromones slightly less intense than deposited in this initial phase:

• If the initial pheromone intensities are too weak, subsequent ants are too much biased by the first tour.

•If the initial pheromone intensities are too strong, subsequent ants are too little influenced by the first ant scouts at all.

Work with m ants on n nodes (m >> n).

2. TSP with ACO: Construction of Tour

Outline

1. Every ant starts at its initialisation node and visits adjacent nodes subsequently until it has visited all nodes.

2. Finally, every ant returns to its initialisation node.

3. At the end, the tour of every ant may be optimised.

2. TSP with ACO: Construction of Tour

Pheromone biased search

Formula for the probability that using the edge (i,j) is a good choice for the tour:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, \quad \text{if } j \in \mathcal{N}_i^k$$

τ_{ij} : pheromone intensity.

(What was experienced in the past ?)

η_{ij} : heuristic information: $1/d_{ij}$

(How good is this edge normally ?)

α und β: internal parameters for adjusting.

$\mathcal{N}_i^k$: Set of nodes being candidates for a visit next.

2.3 TSP with ACO: Updating the Pheromones

Principle

Updating the pheromones starts after all ants have visited all nodes and returned to their initialisation node.

The strength of the new pheromones for edges used by a tour should depend on the quality of the tour discovered.

2.3 TSP with ACO: Updating the Pheromones

Details: Evaporation

All pheromones are diminished by a constant number (Pheromone Evaporation Phase).

This makes edges on bad tour less attractive.

Evaporation formula:

$$\tau_{ij} \longleftarrow (1 - p)\ \tau_{ij}\ ,\ \ \forall\ (i, j) \in E$$

p: fixed network evaporation rate $0 < p \leq 1$

2.3 TSP with ACO: Updating the Pheromones

Details: Enforcement

After evaporation phase, all trails used are enforced.

Every ant raises the pheromone on each edge it used for the tour:

$$\forall (i, j) \in E \qquad \tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k},$$

If L^k is the discovered length of the tour k, $\Delta\tau^k_{ij} = 1/L^k$, if edge (i, j) was used by tour k.

If edge (i, j) was not used by tour k, $\Delta\tau^k_{ij} = 0$.

TSP with ACO: Remarks on Feasibility

For big networks, it is infeasible to compare all nodes of the network where to go next.

Nearest Neighbour Lists are used instead:

An ant will only decide between nodes of the Nearest Neighbour List.

TSP with ACO: Remarks on Further Optimisation

Not all ants are equal: Some ants are elected to be elite ants:

After end of all tours, all lengths are compared.

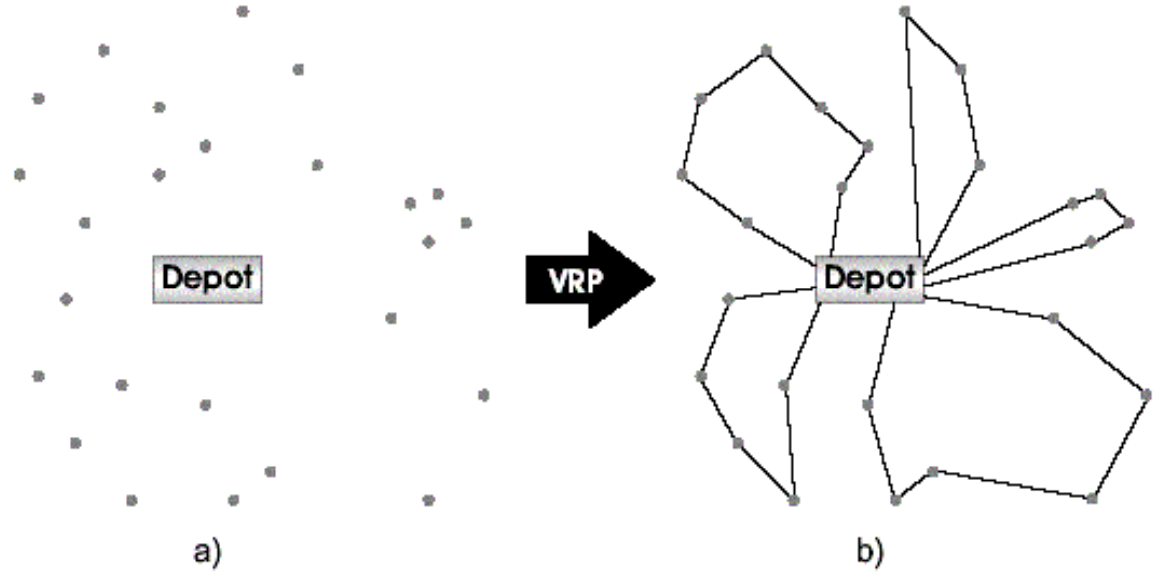
Only the best ants are allowed to deposite new pheromones.

VRP – How to solve this with ACO

1. Problem Definitions: VRP, TSP, ACO
2. Solving TSP with ACO – simple ideas
3. Solving VRP with ACO
4. Example from Practice
5. Complexity issues – what makes TSP like problems prone to ACO?
6. Another example from practice elaborated at FH Wedel

From TSP to VRP

VRP unites several TSPs:



This suggests to generalise TSP algorithms:
Use parallel ant swarms instead of single ants.

But in most cases, there are further constraints:
- Capacity is limited
- Time windows for delivery have to be observed.

Capacitated Vehicle Routing Problem

CVRP is the basic practical VRP. This was the first VRP for which ant algorithms were designed.

Ants start independently from each other and choose next node subsequently just as in TSP.

If capacity is used up or if maximum route length is obtained
→ Ant has to go back to the depot.

VRP with Time Windows

Most typical VRP in reality.

Crucial difference: Time Windows - Customers cannot be served at any time

Two optimisation criteria:

- Minimisation of parallel routes (trucks)
- Minimisation of overall delivery time.

Normally, minimisation of trucks gets priority.

VRP with Time Windows

Most efficient ACO algorithm for VRPTW:

MACS-VRPTW (Multi-Ant-Colony-System)

1999: invented by Gambardella / Taillard / Agazzi

Central idea: Two different ant colonies.

One colony minimises route number (ACS-VEI)

Another colony minimises overall time (ACS-TIME).

Both colonies operate independently from each other: They are using different pheromones.

ACS-TIME is working with route numbers discovered by ACS-VEI only.

VRP with Time Windows

MACS-VRPTW Outline:

1. Start with an initial solution using a certain number of routes and needing a certain overall time.
2. ACS-VEI tries to find a solution with one route less.
3. ACS-TIME tries to find a better solution with this number of routes.
4. If ACS-VEI found a better solution, continue at step 2.

VRP – How to solve this with ACO

1. Problem Definitions: VRP, TSP, ACO
2. Solving TSP with ACO – simple ideas
3. Solving VRP with ACO
4. Example from Practice
5. Complexity issues – what makes TSP like problems prone to ACO?
6. Another example from practice elaborated at FH Wedel

VRPTW Application

Big grocery market chain in Switzerland.

More than 600 stores to be served.

Different truck types: Different capacities, but not all trucks can serve all stores.

Number of trucks is limited.

Each tour must be completed within one day.

Solution

ACO algorithm ANTROUTE
which is a slight modification of MACS-VRPTW algorithm.

At start, ants decide randomly for a truck type.

Waiting penalty costs try to avoid an early (out-of-time-window) arrival.

Two configurations:

AR-RegTW observes regional requests and leaves one hour for unloading for each store.

AR-Free relaxes this restriction if necessary.

Test Scenario and Results

•Within 20 days, 52.000 Pallets had to be distributes to 6800 stores.

•Each day, ANTROUTE received the respective working tasks and gave a good solution as output within 5 minutes.

•Human planners of the company needed 3 hours to find a good solution.

Quality improvements:

	Planning by man	AR-RegTW	AR-Free	AR-RegTW vs. Planer	AR-Free vs. Planer
Number of tours	2.056	1.807	1.614	12,11%	21,50%
Total distance in km	147.271	143.983	126.258	2,23%	14,27%
Avarage degree of capacity utilisation	76,91%	87,35%	97,81%	10,44%	20,9%

VRP – How to solve this with ACO

1. Problem Definitions: VRP, TSP, ACO
2. Solving TSP with ACO – simple ideas
3. Solving VRP with ACO
4. Example from Practice
5. Complexity issues – what makes TSP-like problems prone to ACO?
6. Another example from practice elaborated at FH Wedel

Rating Algorithms

What is measured ?

- Time required
- Space required

Which properties do we require from a rating?

- independent of implementation
- independent of hardware

How can this be achieved?

Principle:

- Neglect constants not depending on the problem size!
- Implementations are considered „equally good“,
  if they only differ by a constant factor.

Rating Algorithms

Definition of a complexity class:

An algorithm belongs to complexity class **O(f(n))** w.r.t. time (or space) iff there is a constant c such that:

Time (Space) for a problem of size n needs at most c·f(n) calculation steps (storage units).

- c must not depend on the problem size.
- c may depend on the algorithm.
- c may depend on the implementation or the actual computer used.
- O is called the Landau symbol (Edmund Landau, 1877-1938)

Typical complexity classes:

O(1), O(log n), O(n), O(n log n), O (n^2), O(P(n)), where P is a polynomial

O(2^n), O(exp(n))

Rating Algorithms

Beispiel für einen 500 Mhz-Rechner mit Rechenzeit $2ns$ pro Taktzyklus:

	Wert von n						
$O(\ldots)$	2	4	8	16	32	64	128
$O(\log_2 n)$	$2ns$	$4ns$	$6ns$	$8ns$	$10ns$	$12ns$	$14ns$
$O(n)$	$4ns$	$8ns$	$16ns$	$32ns$	$64ns$	$128ns$	$256ns$
$O(n \log_2 n)$	$4ns$	$16ns$	$48ns$	$128ns$	$320ns$	$768ns$	$1792ns$
$O(n^2)$	$8ns$	$32ns$	$128ns$	$512ns$	$2\mu s$	$8\mu s$	$32\mu s$
$O(n^3)$	$16ns$	$128ns$	$1\mu s$	$8\mu s$	$65\mu s$	$524\mu s$	$4ms$
$O(2^n)$	$8ns$	$32ns$	$512ns$	$131\mu s$	$8.59s$	$1169a$	$2 \cdot 10^{22}a$
$O(3^n)$	$18ns$	$162ns$	$13\mu s$	$86ms$	$42.89d$	$2 \cdot 10^{14}a$	$7.5 \cdot 10^{44}a$
$O(n!)$	$4ns$	$48ns$	$81\mu s$	$11.6h$	$1.67 \cdot 10^{28}a$	$1.9 \cdot 10^{74}a$	$2 \cdot 10^{214}a$

Tabelle 24: Berechnungsdauer von n Rechenschritten nach Komplexitätsklasse

Complexity classes

Rating Algorithms

Given an algorithms:
Find the best complexity class with respect to **time** or space:

- **in worst case (w.c.)**
- In average case (a.c.)

Examples:

	Time:	Space:
Linear search:	O(n)	O(n)
Binary search:	O(log n)	O(n)
Selectionsort:	$O(n^2)$	O(n)
Mergesort:	O(n log n)	O(n)
Quicksort:	$O(n^2)$ **w.c.** O(n log n) **a.c.**	O(n)

Complexity classes

Rating Problems

Given a problem:
Find the best complexity class for which an algorithm exists solving the problem in general.

A problem belongs to complexity class **Ω(f(n))** w.r.t. time (or space), iff there is a constant c such that:

Time (or space) of ***every*** algorithm for a the solution of the problem of size n required is ***at least*** c·f(n) computation steps (storage units).

Beispiele:

	Time:	Space:
Searching in an array:	Ω(n)	Ω(n)
Sorting an array:	Ω(n log n)	Ω(n)

Complexity classes

How do we make complexity classes independent of the computer?

The crucial idea: The Turing Machine

- Theoretical ancestor of all computers defined by Alan Turing in 1937

For every computer known to these days holds:

- If a problem of size n is solvable on a computer with f(n) computing steps, then it is also solvable in O(P(f(n))) computing steps
  for a polynomial P not depending on the specific computer.

NP-completeness

NP complete problems are characterised as follows:

- The problem is solved on a **nondeterministic** Turing Maschine in polynomial time.

  These are problems where the solution can be verified on a deterministic (normal) Turing Machine in polynomial time.

- An NP-complete problem is solvable on a deterministic Turing Machine in polynomial time, **iff *all*** NP-complete problems are solvable on a deterministic Turing Machine in polynomial time.

  An NP-complete problem is a most difficult problem among those whose solution is verifiable on a deterministic Turing Machine in polynomial time.

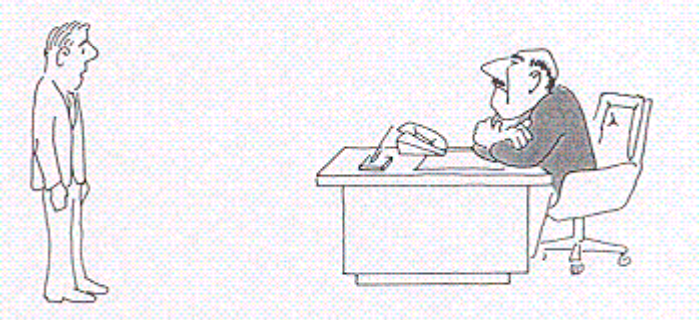
Open question of the century in computer science:

1) Are NP-complete problems solvable on a computer in polynomial time?

2) Or do they belong to a complexity class Ω(f(n)) where f(n) is growing faster than every polynomial P(n) which means they are intractable ?

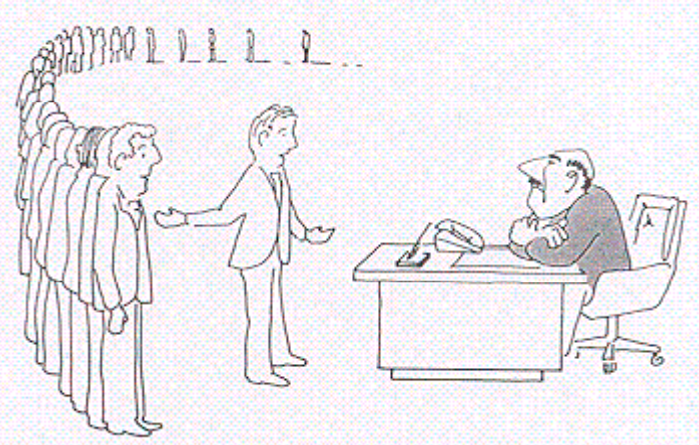
NP-completeness

What do you explain your superior
if you got assigned a problem you could not solve ?



I did not follow in class when we discussed
how to solve your assignment!

Your assignment is unsolvable!



I could not solve your assignment,
But all these famous scientists could solve it neither!

According to: Michael Garey / David Johnson:
Computers and Intractability, WH Freeman 1979

Relation to traffic issues

What does this complexity stuff have to do with TSP or VRP?

TSP is one of the NP-complete problems !

This makes it unlikely to find a feasible solution with an exact mathematically optimised algorithm

➔ Learn from the ants!

VRP – How to solve this with ACO

1. Problem Definitions: VRP, TSP, ACO
2. Solving TSP with ACO – simple ideas
3. Solving VRP with ACO
4. Example from practice
5. Complexity issues – what makes TSP like problems prone to ACO?
6. Another example from practice elaborated at FH Wedel

TSP ant algorithm in another practical application

Traveling Salesman (TSP) with Ant Algorithms (ACO)

The way this algorithm works

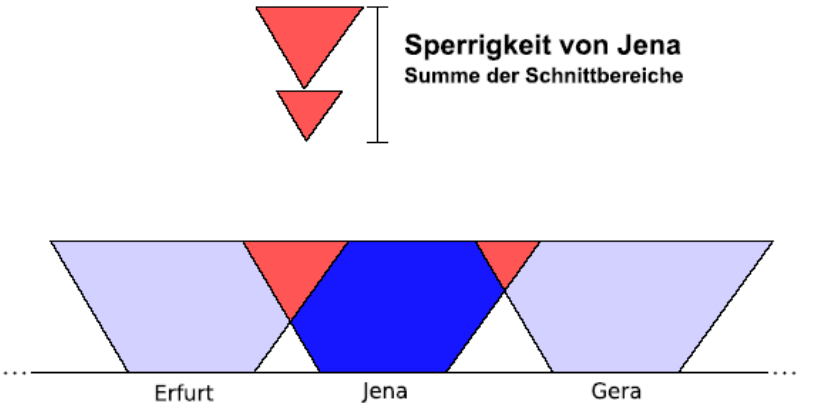
- Phermonomes indicate at each node in which way it is most favorable to proceed.
- Ants are started at each node continuously.
- Ants choose the neighbour according to pheromone intensity
- Pheromones are modified in two ways:
  1) Each ant diminishes the intensity on the way it proceeds.
  2) In regular intervals, the best path found so far is determined, and the pheromones are made stronger on that path.

Advantages

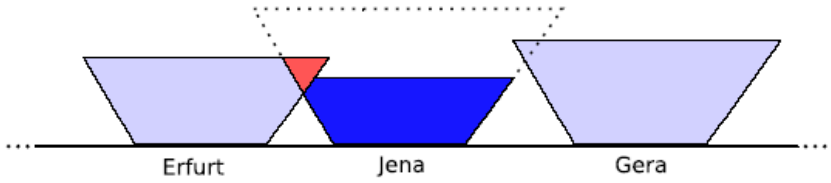
- Solution will be improved goal-oriented towards the optimum.
- The deminishing of traces will avoid getting stuck in local optima (this can be proved mathematically).
- The procedure will answer with a solution at any time (anytime property).
- Algorithm adopts automatically to changed traffic conditions or requirements..

TSP ant algorithm in another practical application

Transfer from TSP to navigation display problem (master thesis 2007)



It depends on the order of cities considered how long you can display a city‘s name:



Problem:

Find an order in which the maximum number of cities may be displayed at all zoom levels.

TSP ant algorithm in another practical application

Transfer from TSP to navigation display problem (master thesis 2007)

Analogy of display problem to TSP:

- Display feasibility depending on city order corresponds to path lengths in TSP
- Pheromones indicate how useful it is to consider city j directly after city i.

Results of ant procedure:

- Continuously, new solutions were found.
- The quality of the results could be evaluated and reconsidered after inspecting the output (learning property).

It depends on the order of cities considered how long you can display a city‘s name: