Ant Colony Optimization An Introduction.

Thomas Stützle

stuetzle@informatik.tu-darmstadt.de

http://www.intellektik.informatik.tu-darmstadt.de/˜tom.

Technische Universität Darmstadt

Fachbereich Informatik

What is Ant Colony Optimization?

Ant Colony Optimization (ACO) is a population-based, general search technique for the solution of difficult combinatorial problems which is inspired by the pheromone trail laying behavior of real ant colonies.

Outline of the talk

- Biological Inspiration

- ACO algorithms

- ACO applications

- ACO metaheuristic

- ACO theory

- Conclusions

Biological Inspiration

Pheromone trail following behavior

- in many ant species the visual perceptive faculty is only very rudimentarily developed

- most communication among individuals is based on chemicals called pheromone

- a particular type in some ant species is *trail pheromone*, used for marking and following paths

⤳ *collective trail laying / trail following behavior is the inspiring source of Ant Colony Optimization*

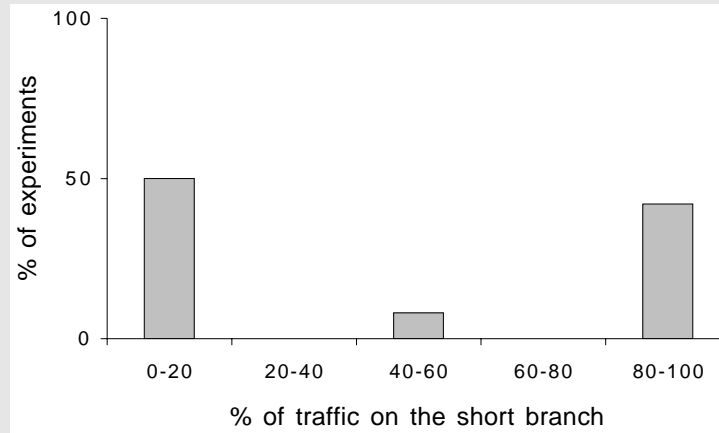# *Double bridge experiments* *Deneubourg et al.*

- laboratory colonies of *Iridomyrmex humilis*

- ants deposit pheromone while walking from food sources to nest and vice versa

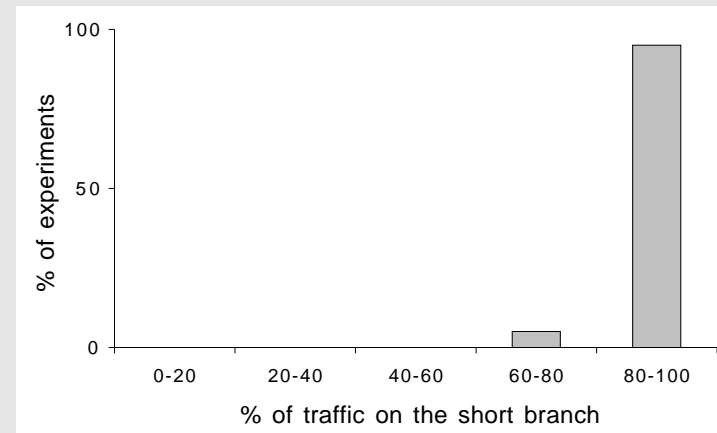- ants tend to choose, in probability, paths marked by strong pheromone concentrations

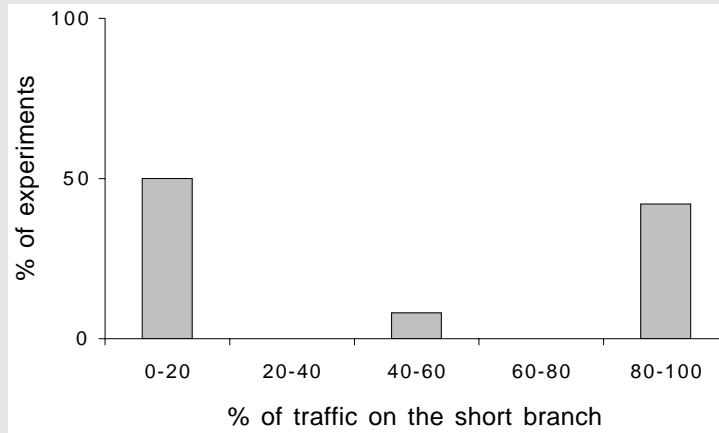15 cm

Nest 60° Food

Nest 1 2 Food
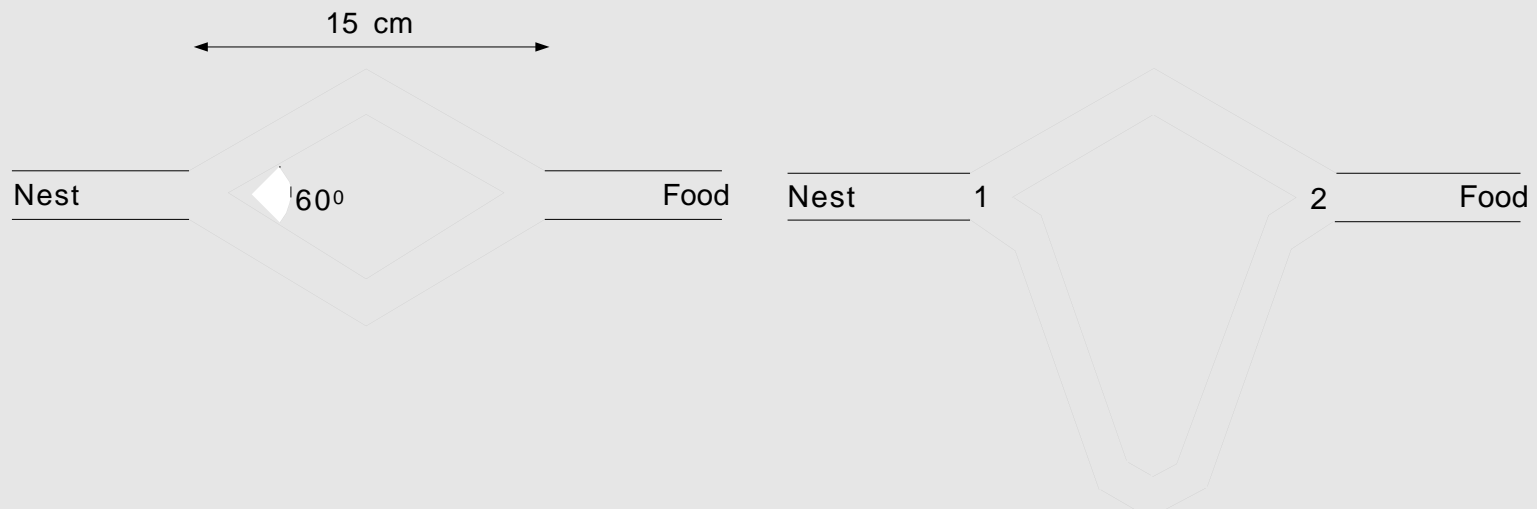
Experimental results

- equal length bridges: convergence to a single path

Experimental results

- equal length bridges: convergence to a single path

- different length paths: convergence to short path

Stochastic model

- a stochastic model was derived from the experiments and verified in simulations

- functional form of transition probability

$$p_{i,a} = \frac{(k + \tau_{i,a})^\alpha}{(k + \tau_{i,a})^\alpha + (k + \tau_{i,a'})^\alpha}$$

- $p_{i,a}$: probability of choosing branch a when being at decision point i

  $\tau_{i,a}$: corresponding pheromone concentration

Towards artificial ants

- real ant colonies are solving *shortest path problems*

- Ant Colony Optimization takes elements from real ant behavior to solve more complex problems than real ants

- In ACO, artificial ants are *stochastic solution construction procedures* that probabilistically build solutions exploiting

  - (artificial) *pheromone trails* which change dynamically at run time to reflect the agents' acquired search experience

  - *heuristic information* on the problem instance being solved

Travelling Salesman Problem
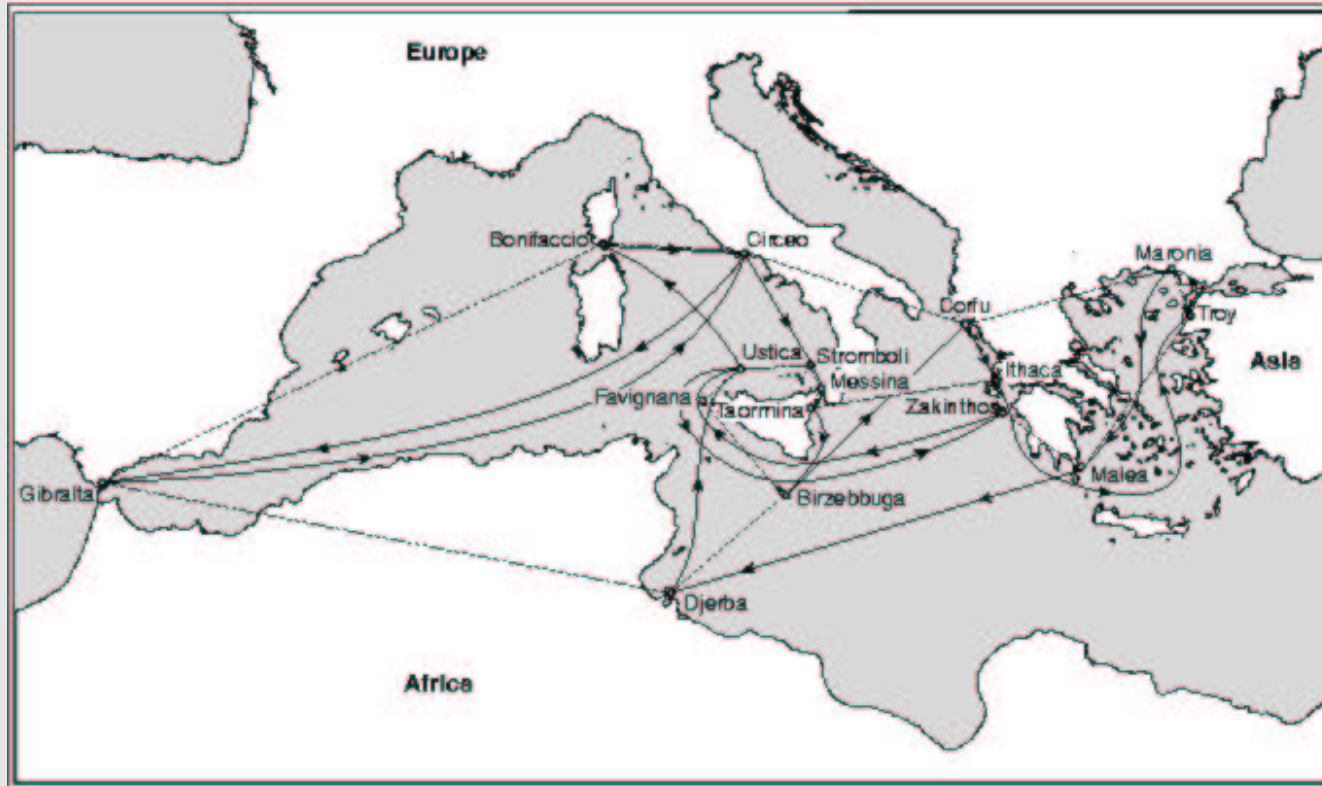
Stochastic solution construction
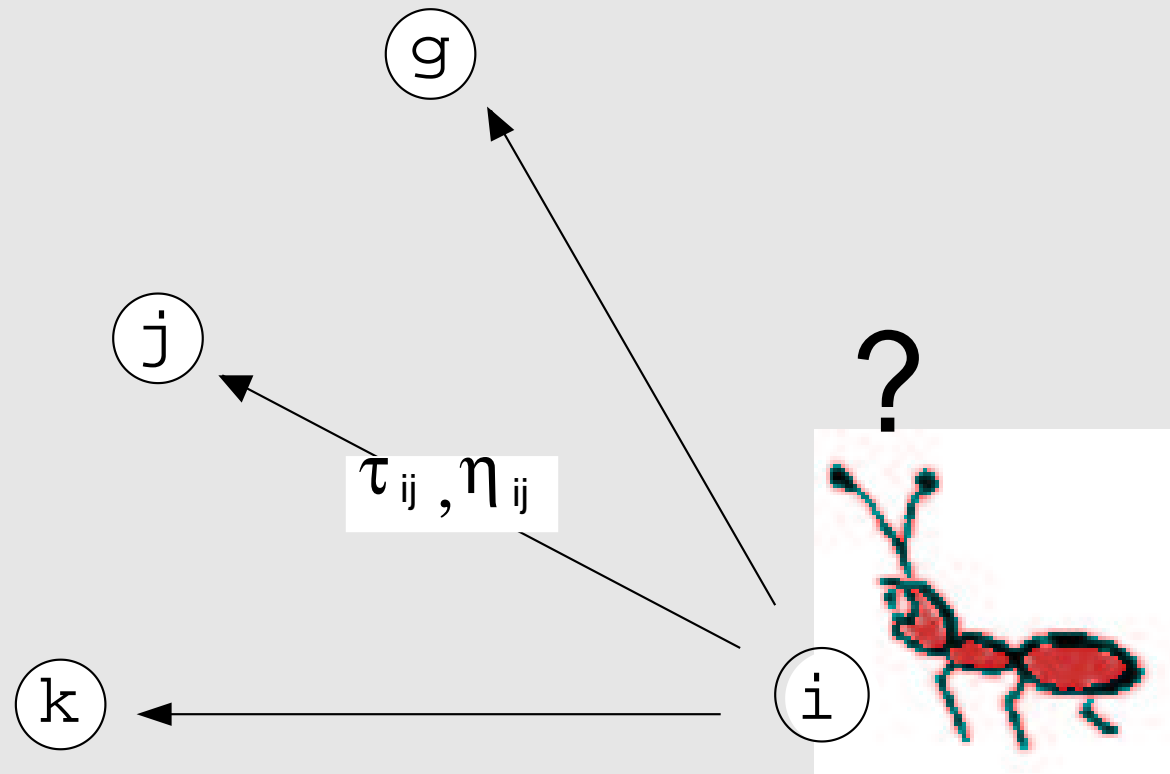
Stochastic solution construction

- use memory to remember partial tours

- being at a city i choose next city j probabilistically among feasible neighbors

- probabilistic choice depends on pheromone trails τ_{ij} and heuristic information $\eta_{ij} = 1/d_{ij}$

- "usual" action choice rule at node i

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad \text{if } j \in \mathcal{N}_i^k$$

Pheromone update

- use positive feedback to reinforce components of good solutions

  - better solutions give more feedback

- use pheromone evaporation to avoid unlimited increase of pheromone trails and allow forgetting of bad choices

  - pheromone evaporation $0 < \rho \leq 1$

Pheromone update (2)

- Example of pheromone update

$$\tau_{ij}(t) = (1 - \rho) \cdot \tau_{ij}(t-1) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

- $\Delta\tau_{ij}^{k} = 1/L_k$ if arc (i,j) is used by ant k on its tour

  L_k: Tour length of ant k

  m: Number of ants

  The resulting algorithm is called Ant System

Ant System

- Ant System is the first ACO algorithm proposed in 1991 by Dorigo et al.

- encouraging initial results on TSP but inferior to state-of-the-art

Role of Ant System

- "proof of concept"

- stimulation of further research on algorithmic variants and new applications

Ant System: The algorithm

procedure *Ant System for TSP*

    Initialize pheromones

    while (termination condition not met) **do**

        for $i = 1$ **to** $n - 1$ **do**

            for $k = 1$ **to** m **do**

                ApplyProbabilisticActionChoiceRule$(\mathcal{M}^k, \tau, \eta)$

            end

        end

        GlobalPheromoneTrailUpdate

    end

end *Ant System for TSP*

ACO Algorithms: Overview

ACO algorithm	Authors	Year	TSP
Ant System	Dorigo, Maniezzo & Colorni	1991	yes
Elitist AS	Dorigo	1992	yes
Ant-Q	Gambardella & Dorigo	1995	yes
Ant Colony System	Dorigo & Gambardella	1996	yes
$\mathcal{MM}$AS	Stützle & Hoos	1996	yes
Rank-based AS	Bullnheimer, Hartl & Strauss	1997	yes
ANTS	Maniezzo	1998	no
Best-Worst AS	Cordón, et al.	2000	yes
Hyper-cube ACO	Blum, Roli, Dorigo	2001	no

Elitist Ant System   *Dorigo, 1992*

- strong additional reinforcement of best found solution T^{gb} (best-so-far solution)

- pheromone update rule becomes

$$\tau_{ij}(t) = (1 - \rho) \cdot \tau_{ij}(t-1) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e \cdot \Delta\tau_{ij}^{gb}$$

- τ_{ij}^{gb} is defined analogous to $\Delta\tau_{ij}^{k}$

Ant Colony System (ACS)

- *pseudo-random proportional action choice rule*:

  - with probability q_0 an ant k located at city i chooses successor city j with maximal $\tau_{ij}(t) \cdot [\eta_{ij}]^{\beta}$ (exploitation)

  - with probability $1 - q_0$ an ant k chooses the successor city j according to action choice rule used in Ant System (biased exploration)

Ant Colony System (2)

- global pheromone update rule

  - only the best-so-far solution T^{gb} deposits pheromone after each iteration

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

  where $\Delta\tau_{ij}^{gb}(t) = 1/L^{gb}$.

  - pheromone update only affects edges contained in T^{gb}!

Ant Colony System (3)

- local pheromone update rule

  - ants "eat away" pheromone on the edges just crossed

  $$\tau_{ij} = (1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_0$$

  - τ_0 is some small constant value

  - increases exploration by reducing the desirability of frequently used edges

$\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*

- extension of Ant System with stronger exploitation of best solutions and additional mechanism to avoid search stagnation

- only the iteration-best or best-so-far ant deposit pheromone

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

# $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System (2)*

- additional limits on the feasible pheromone trails
  - for all $\tau_{ij}(t)$ we have: $\tau_{min} \leq \tau_{ij}(t) \leq \tau_{max}$
  - counteracts stagnation of search through aggressive pheromone update
  - heuristics for determining τ_{min} and τ_{max}
- pheromone values are initialized to τ_{max}
  - $\rightsquigarrow$ stronger exploration at the start of the algorithm
- pheromone trail re-initialization to increase exploration

Comparison of ACO algorithms

- all algorithms construct $10.000 \cdot n$ tours for symmetric TSPs, $20.000 \cdot n$ tours for asymmetric TSPs

- only solution construction

Instance	*opt*	$\mathcal{MMAS}$	ACS	ASe	AS
eil51.tsp	426	427.6	428.1	428.3	437.3
kroA100.tsp	21282	21320.3	21420.0	21522.83	22471.4
d198.tsp	15780	15972.5	16054.0	16205.0	16705.6
lin318.tsp	42029	42220.2	42570.0	43422.8	45535.2
ry48p.atsp	14422	14553.2	14565.4	14685.2	15296.4
ft70.atsp	38673	39040.2	39099.0	39261.8	39596.3
kro124p.atsp	36230	36773.5	36857.0	37510.2	38733.1
ftv170.atsp	2755	2828.8	2826.5	2952.4	3154.5

Comparison of ACO algorithms
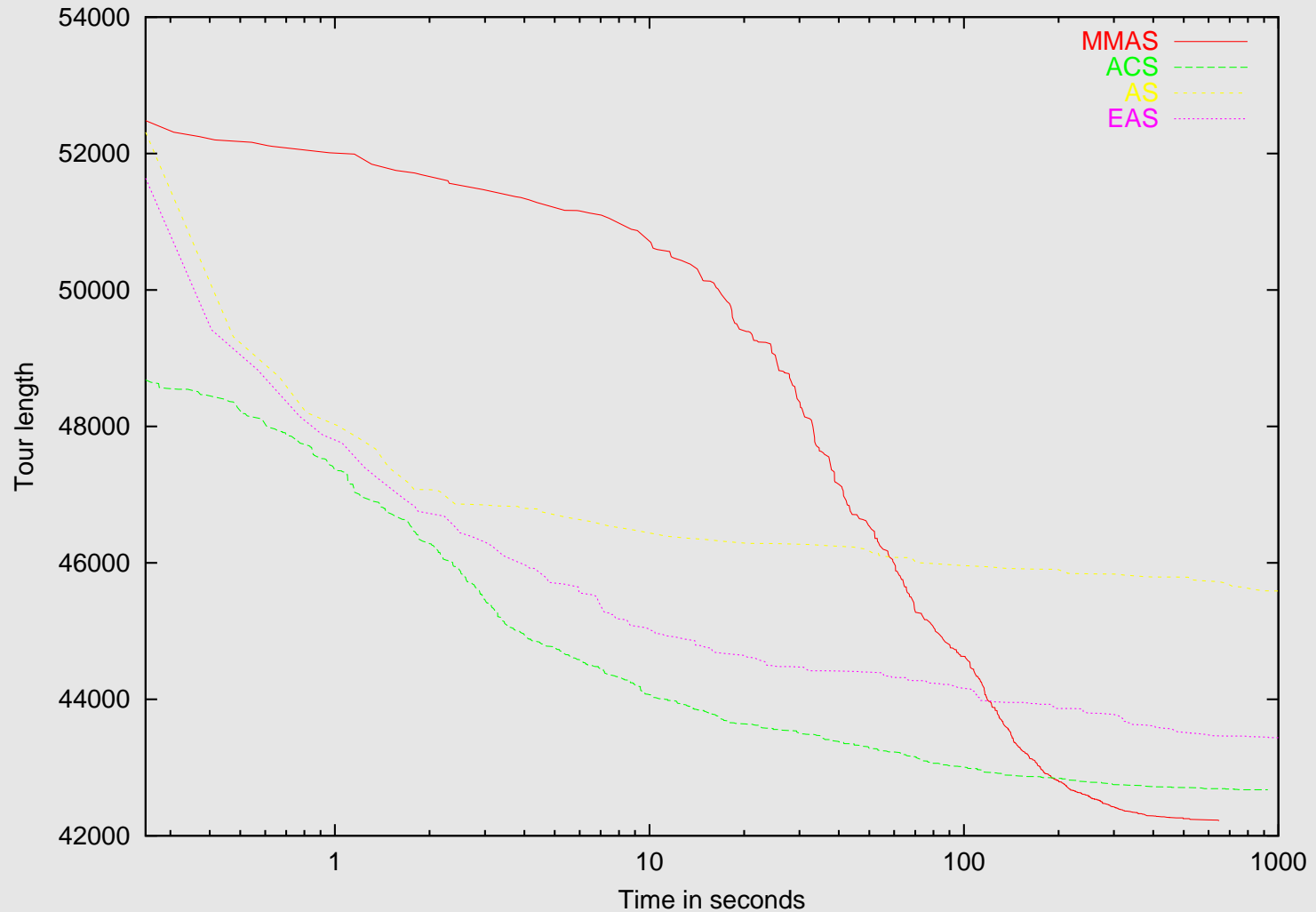
Combining ACO with local search

- current wisdom suggests that good strategy for attacking combinatorial problems is a coupling of a constructive heuristic and a local search

- problem: finding a good coupling

- ACO plus local search seems to be such a good coupling as experimental results suggest

Advantages

- local search fine-tunes constructed solutions

- ACO algorithm alleviates "initialization problem" of local search

Experimental results

- much improved performance; e.g. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System+3-opt:

  - instances with up to 500 cities:
    optimal found within few seconds/minutes

  - for large instances (tested up to 3038 cities) solutions within 0.25% of optimal can be found in reasonable time (< 1 hour)

- performance is close to state-of-the-art

Experimental results
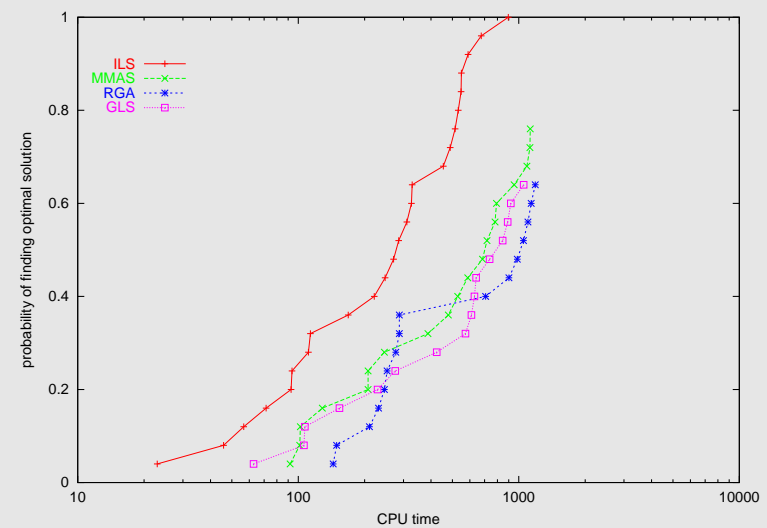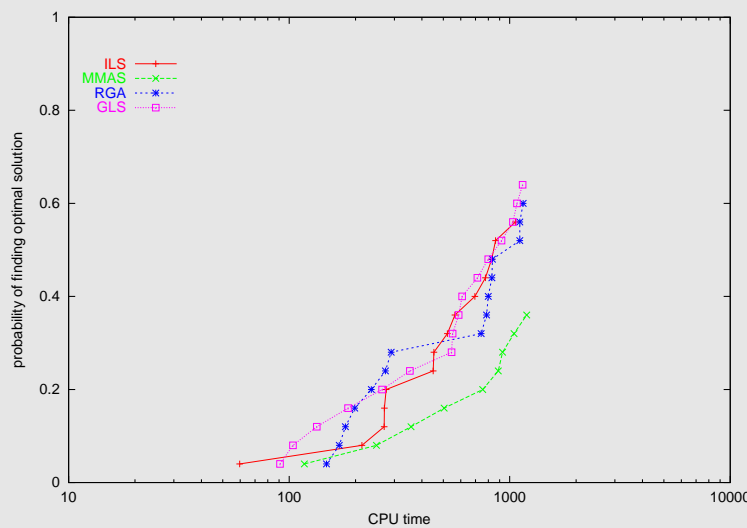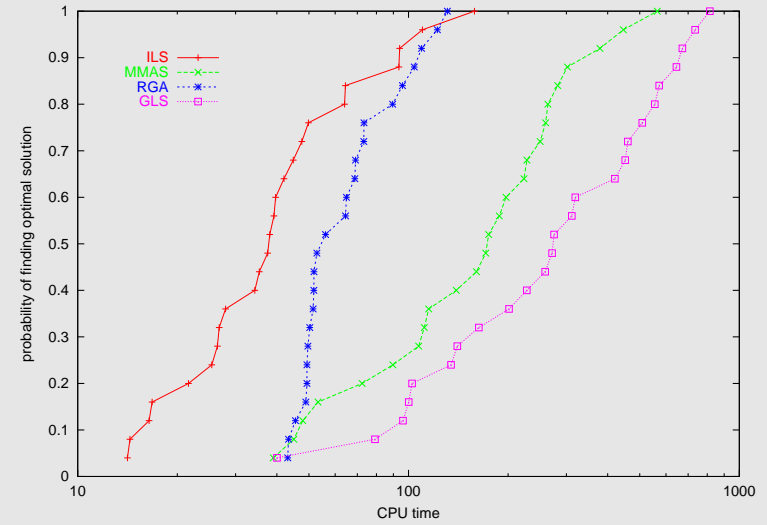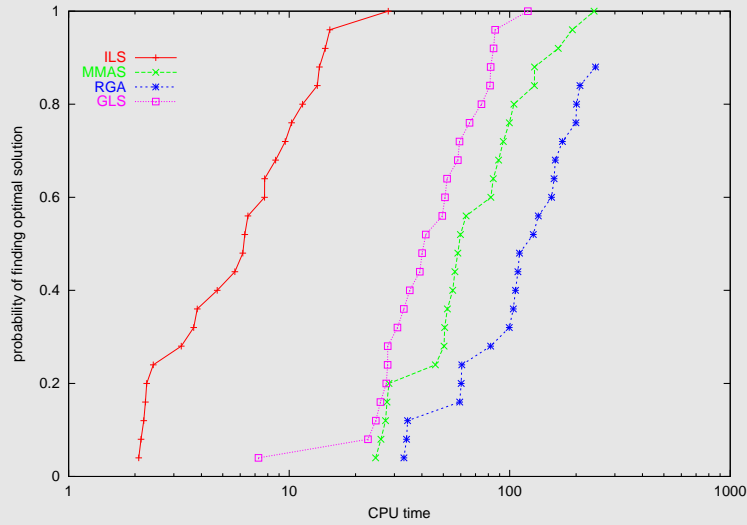
Observations

- common features among extensions
  - strong exploitation of best found solutions
  - the most efficient extensions use local search
- differences concern essential aspects of the search control
- best performing variants add explicit diversification features

ACO applications: QAP

Design a keyboard layout!
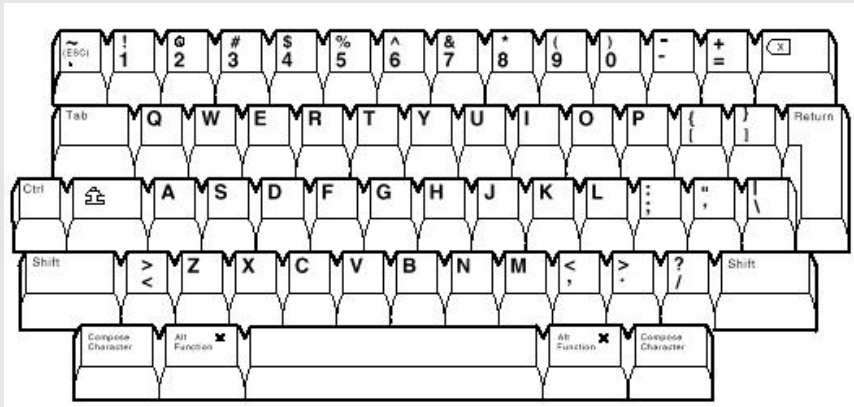
Goal:
find a keyboard layout that
minimizes the average writing
time!

- Distance: time for pressing two keys consecutively

- Flow: frequency of a letter given another letter

- Objective function: average writing time

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} f_{\phi_i \phi_j}$$

ϕ_k is the letter assigned to the key k.

$\mathcal{MMAS}$ for the QAP

Two decisions have to be made:

- which letter (item) is to be chosen next?

- to which key (location) is this letter assigned?

In $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System:

- items are chosen randomly

- τ_{ij} gives desirability of assigning item i to location j

$$p_{ij}^k(t) = \frac{\tau_{ij}(t)}{\sum_{l \in \mathcal{N}_i^k} \tau_{il}(t)}$$

Experimental results – 1

Problem instance	Ro-TS	GH	HAS-QAP	$\mathcal{MMAS}$-QAP$_{TS}$	$\mathcal{MMAS}$-QAP$_{2-\text{opt}}$
random problems with uniformly distributed matrix entries (i)					
tai40a	0.990	0.884	1.989	0.794	1.509
tai50a	1.125	1.049	2.800	1.060	1.795
tai60a	1.203	1.159	3.070	1.137	1.882
tai80a	0.900	0.796	2.689	0.836	1.402
random flows on grids (ii)					
nug30	0.013	0.007	0.098	0.013	0.039
sko49	0.076	0.040	0.141	0.068	0.115
sko56	0.088	0.060	0.101	0.075	0.098
sko64	0.071	0.092	0.129	0.071	0.099
sko72	0.146	0.143	0.277	0.090	0.172
sko81	0.136	0.136	0.144	0.062	0.124
sko90	0.128	0.196	0.231	0.114	0.140

Results – 2

Problem instance	Ro-TS	GH	HAS-QAP	$\mathcal{MMAS}$-QAP$_{TS}$	$\mathcal{MMAS}$-QAP$_{2-opt}$
real-life instances (iii)					
`bur26a-h`	0.002	0.043	0.0	0.006	0.0
`kra30a`	0.268	0.134	0.630	0.134	0.157
`kra30b`	0.023	0.054	0.071	0.044	0.066
`ste36a`	0.155	n.a.	n.a.	0.061	0.126
`ste36b`	0.081	n.a.	n.a.	0.0	0.0
randomly generated real-life like instances (iv)					
`tai40b`	0.531	0.211	0.0	0.402	0.0
`tai50b`	0.342	0.214	0.192	0.172	0.009
`tai60b`	0.417	0.291	0.048	0.005	0.005
`tai80b`	1.031	0.829	0.667	0.591	0.266
`tai100b`	0.512	n.a.	n.a.	0.230	0.114

ACO for static problems

procedure *Ant Colony Optimization*

    Initialize parameters, pheromone trails

    while (termination condition not met) **do**

```
    ConstructSolutions

    ApplyLocalSearch  %optional

    GlobalUpdateTrails
```

    end

end *Ant Colony Optimization*

ACO applications: routing

- goal: build routing tables to direct data traffic optimizing some measure of network performance

- a routing table entry r_{ijd} gives for node i and destination node d of a data packet the next node j to move to

- routing is difficult because routing costs are dynamic

- adaptive routing is difficult because changes in control policy result in changes in the performance measures

ACO algorithm: AntNet *Di Caro, Dorigo, 1997-1999*

- ants are launched at regular intervals from each node to randomly chosen destinations

- ants are routed probabilistically biased by artificial pheromone values and heuristic information

- ants memorize path and trip time

- when reaching destination node, ants retrace path and update pheromone trails

- data packets are routed deterministically

AntNet is distributed and asynchronous

Experimental results

- comparisons were done to a number of state-of-the-art algorithms

- experiments were run under varying traffic conditions

- AntNet compared very favorably to other algorithms and was shown to be very robust

- reason: ACO matches well the problem characteristics
  - stochastic state transitions
  - only distributed information is available

Applications of ACO

- Main application fields
  - $\mathcal{NP}$-hard problems
  - dynamic shortest path problems (network routing)
- World class performance on
  - sequential ordering, vehicle routing, quadratic assignment, open-shop scheduling, shortest common super-sequence problem, 2D-HP protein folding, packet-switched network routing, mobile ad-hoc network routing ..
- Very good performance on many other problems
- Industrial applications
  - AntOptima

The ACO Metaheuristic   *Dorigo, Di Caro, Gambardella 1999*

- ACO metaheuristic was proposed *a posteriori* as a common framework for ACO applications

- artificial ants in ACO are seen as stochastic solution construction procedures

- solution construction is biased by
  - pheromone trails which change at run-time
  - heuristic information on the problem instance
  - the ants' memory

- additional capabilities via daemon actions

The ACO metaheuristic

procedure *ACO metaheuristic*
    ScheduleActivities

```
ManageAntsActivity()
EvaporatePheromone()
DaemonActions() {Optional}
```

    end ScheduleActivities
end *ACO metaheuristic*

ACO as tree search

- ACO metaheuristic is centrally based on notion of construction graph

- alternative point of view: ants as stochastic tree search procedures

- highlights ants as stochastic construction procedures = stochastic search tree exploration

- this latter view highlights natural relationships of ACO to tree search procedures such as Branch-and-Bound, backtracking, beam-search, etc.

ACO Theory

- convergence proofs *Gutjahr 1999–2005; Stützle, Dorigo 2001-2004*

- formal relationships to other methods
  - model-based search framework *Zlochin et al. 2005*
  - relationship to methods such as PBIL, stochastic gradient descent, cross-entropy method or EDAs
    Meuleau, Dorigo, 2002

- models of ACO behavior
  - modelling the dynamics of ACO *Merkle, Middendorf 2002-04*
  - examination of search bias *Blum et al. 2002-05*

ACO Theory—Convergence proofs

- Gutjahr (1999) has proved **convergence to the optimal solution** of a Graph-based Ant System:

  - Let P_t be the probability with which an agent traverses the optimal tour at iteration t.

  - For each $0 < \epsilon < 1$ and for a fixed evaporation rate, if the number of agents is large enough, then

  $$P_t \leq 1 - \epsilon, \quad \forall t \leq t_0, t_0 = t_0(\epsilon)$$

  - For each $0 < \epsilon < 1$ and for a fixed number of agents, if the evaporation rate is small enough, then

  $$P_t \leq 1 - \epsilon, \quad \forall t \leq t_0, t_0 = t_0(\epsilon)$$

ACO Theory—Convergence proofs

- Stützle and Dorigo (2001/02) proved **convergence properties** for a class of ACO algorithms called $\text{ACO}_{\tau_{min}}$:
  - for any $\epsilon > 0$ they prove that the probability of finding at least once an optimal solution is $P^*(t) \geq 1 - \epsilon$
  - $P^*(t)$ tends to one for $t \to \infty$

- these results are directly extendable to two of the experimentally most successful ACO algorithms: $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Ant Colony System

- Gutjahr (2002) extends these proofs by to convergence with probability one

Recent Trends in ACO

- continued interest in applications

- new applications

  - multi-objective optimization

  - dynamic optimization problems

  - stochastic optimization problems

- increasing interest in theoretical issues

- methodology for applying ACO

Conclusions

- Ant Colony Optimization is becoming a mature field

  - variety of algorithms available

  - many successful applications

  - theoretical results

  - dedicated workshops (ANTS'1998 — 2006) and journal special issues

Conclusions

- Ant Colony Optimization is becoming a mature field
  - variety of algorithms available
  - many successful applications
  - theoretical results
  - dedicated workshops (ANTS'1998 — 2006) and journal special issues

ACO is the most successful technique of the wider field of Swarm Intelligence

Interested in more?