Università della Svizzera italiana

Scuola universitaria professionale della Svizzera italiana

IDSIA
Istituto Dalle Molle di studi sull'intelligenza artificiale

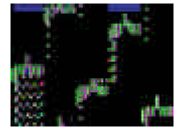
Ant Colony Optimization

Luca Maria Gambardella

IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Manno, Lugano, Switzerland
www.idsia.ch

Insects, Social Insects, and Ants

- 10^{18} living insects (rough estimate)
- ~2% of all insects are social
- Social insects are:
  - All ants
  - All termites
  - Some bees
  - Some wasps



- 50% of all social insects are ants
- Avg weight of one ant between 1 and 5 mg
- Tot weight ants ~ Tot weight humans

How Do Ants Coordinate their Activities?

- Ants do not directly communicate. The basic principle is *stigmergy*, a particular kind of indirect communication based on environmental modification
- Stimulation of workers by the performance they have achieved
  Grassé P. P., 1959
- Foraging behavior: searching for food by parallel exploration of the environment

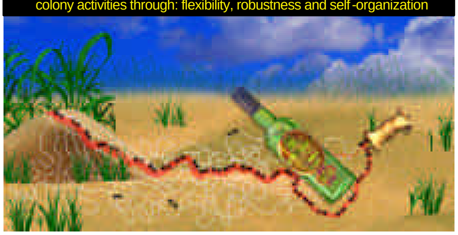
Shortest paths: an emerging behavior from stigmergy

- **Foraging ant colonies can synergistically find *shortest paths* in *distributed / dynamic* environments:**
  - While moving back and forth between nest and food ants mark their path by *pheromone* laying
  - Step-by-step routing decisions are biased by the *local* intensity of pheromone field (*stigmergy*)
  - Pheromone is the colony's collective and distributed *memory*: it encodes the collectively learned quality of local routing choices toward destination target

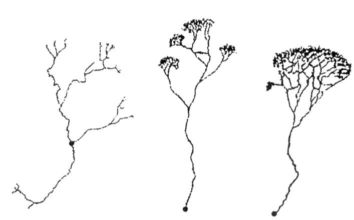
R. Beckers, J. L. Deneubourg and S. Goss, Trails and U-turns in the selection of the shortest path by the ant Lasius Niger, *J. of Theoretical Biology*, 159, 1992

How Ants Find Food

Social insects, following simple, individual rules, accomplish complex colony activities through: flexibility, robustness and self-organization

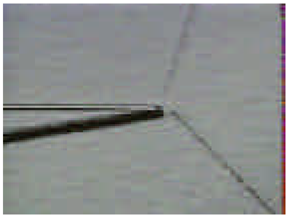


Ants Foraging Behavior



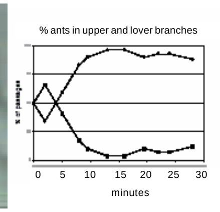
Pheromone Trail Following

Ants and termites follow pheromone trails



Simple Bridge Experiment

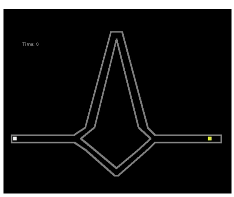
Goss et al., 1989, Deneubourg et al., 1990



Asymmetric Bridge Experiment

Goss et al., 1989

Dorigo & Bertolissi, 1998



From ants to agents

- **Reverse-engineering of ant colony mechanisms: *Ant Colony Optimization* (ACO) metaheuristic:**
  - Combinatorial optimization
  - Adaptive routing
- **Multiple autonomous/concurrent agents (*ants*): solution construction as *sequential decision process*:**
  - *Model: a network of decision points where the quality of the choices is expressed by pheromone variables*
  - *Building Solutions = constructing a path in the network according to a stochastic decision policy*
  - Use of solution outcomes to iteratively update pheromone (*generalized policy iteration* based on Monte Carlo sampling)
  - No explicit solutions representation. The collectively learned knowledge is distributed in the *pheromone*

Dorigo M., Di Caro G., Gambardella L.M., "Ant Algorithms for Distributed Discrete Optimization", *Artificial Life,* Vol. 5, N. 2, 1999.

ACO

- *ACO algorithms* are multi-agent systems that exploit *artificial stigmergy* for the solution of combinatorial optimization problems.
- Artificial ants live in a discrete world. They construct solutions making stochastic transition from state to state.
- They deposit artificial pheromone to modify some aspects of their environment (search space). Pheromone is used to dynamically store past history of the colony.
- Artificial Ants are sometime "augmented" with extra capabilities like local optimization or backtracking

Similarities with Real Ants

- Colony of simple cooperative individuals.
- an artificial pheromone trail is used for local stigmergetic communication
- a sequence of local moves to find shortest path
- a stochastic construction policy (exploration and exploitation) based on local information

Differences with real ants

- Artificial ants use a discrete world
- Artificial ants have internal state and memory
- The deposited pheromone is proportional to the quality of the solution (some real ants have a similar behavior)
- extra capabilities (lookahead, local optimization, backtracking)

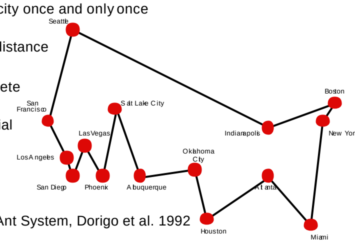
Travelling Salesman Problem (TSP)

Problem: given N cities, and a distance function d between cities, find a tour that:

(1) goes through every city once and only once

(2) minimizes the total distance

- Problem is NP-complete
- Classical combinatorial optimization problem to test algorithms



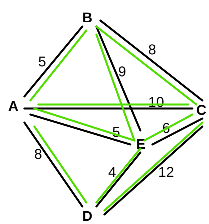
First ACO application, Ant System, Dorigo et al. 1992

Search Space

Discrete Graph

To each edge is associated a *static value* returned by an heuristic function *h*(*r*,*s*) based on the edge-cost

Each edge of the graph is augmented with a pheromone trail *t*(*r*,*s*) deposited by ants. *Pheromone is dynamic* and it is learned at run-time



ACS: Ant Colony System for TSP

Loop
Randomly position m artificial ants on n cities
For city=1 to n
For ant=1 to m
{*Each ant builds a solution by adding one city after the other*}
Select probabilistically the next city according to exploration and exploitation mechanism
Apply the local trail updating rule
End for
calculate the length L_m of the tour generated by ant m
End for
Apply the global trail updating rule using the best ant
Until End_condition

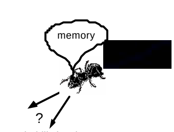
Dorigo M., Gambardella L.M, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem , *IEEE Transactions on Evolutionary Computation* Vol. 1,No. 1,pp. 53-66, 1997

ACS State Transition rule

Next city is chosen between the not visited cities according to a probabilistic rule

Exploitation: the best edge is chosen

Exploration: one of the edge in proportion to its value



ACS state transition rule: formulae

$$s = \begin{cases} \arg\max_{u \in Jk(r)} \left\{ [t(r,u)] \cdot [h(r,u)]^b \right\} & \text{if } q \leq q_0 \quad \text{(Exploitation} \\ S & \text{otherwise} \quad \text{(Explorat} \end{cases}$$

where

- S is a stochastic variable distributed as follows:

$$p_k(r,s) = \begin{cases} \dfrac{[t(r,s)] \cdot [h(r,s)]^b}{\sum_{u \in J_k(r)} [t(r,u)] \cdot [h(r,u)]^b} & \text{if } s \in J_k(r) \\ 0 & \text{othe} \end{cases}$$

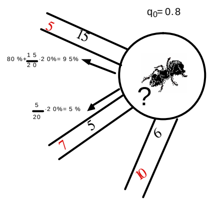
- τ is the trail
- η is the inverse of the distance
- $J_k(r)$ is the set of cities still to be visited by ant k positioned on city r
- β and q_0 are parameters

ACS state transition rule: example

next state:

with probability q_0 exploitation

with probability $(1\text{-}q_0)$ biased exploration



ACS local trail updating … similar to evaporation

If an edge (r,s) is visited by an ant

$$t(r,s) = (1- r)\cdot t(r,s) + r\cdot \Delta t(r,s)$$

with $\Delta t(r,s) = t_0$

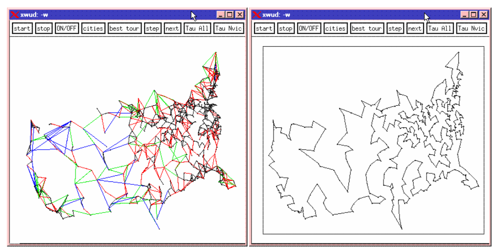
ACS global trail updating

At the end of each iteration, the best ant so far, is allowed to reinforce its tour by depositing additional pheromone proportional to the length of the tour

$$t(r,s) \leftarrow (1- a)\cdot t(r,s) + a \cdot Dt(r,s)_{Global}$$

where

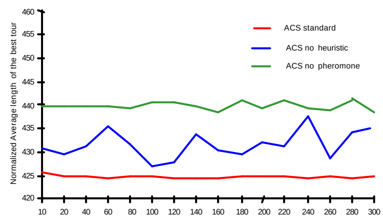
$$Dt(r,s)_{Global} = \frac{1}{L_{best}}$$



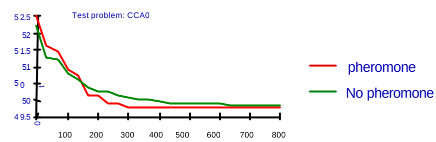
Best solutions structures emerge step by step from the computation

Among the state of the art algorithms for TSP and ATSP problems

Pheromone is useful?



Effectiveness of distributed pheromone learning



Best tour length as a function of elapsed CPU time (avg on 100 runs)

Comparison of ACS with other heuristics on random TSPs

Problem name	ACS (average)	SA (average)	EN (average)	SOM (average)
City set 1	**5.88**	**5.88**	5.98	6.06
City set 2	6.05	**6.01**	6.03	6.25
City set 3	**5.58**	5.65	5.70	5.83
City set 4	**5.74**	5.81	5.86	5.87
City set 5	**6.18**	6.33	6.49	6.70

Comparisons on average (25 trials) tour length obtained on five random 50-city symmetric TSP

Comparison of ACS with other natural algorithms on geometric TSPs

Problem name	ACS	GA	EP	SA	Optimum
Eil50 (50-city problem)	**425** (**427.96**) **[1,830]**	428 (N/A) [25,000]	426 (**427.86**) [100,000]	443 (N/A) [68,512]	**425** (N/A)
Eil75 (75-city problem)	**535** (**542.37**) **[3,480]**	545 (N/A) [80,000]	542 (549.18) [325,000]	580 (N/A) [173,250]	**535** (N/A)
KroA100 (100-city problem)	**21,282** (**21,285.44**) **[4,820]**	21,761 (N/A) [103,000]	N/A (N/A) [N/A]	N/A (N/A) [N/A]	**21,282** (N/A)

Best integer tour length, best real tour length (in parentheses) and number of tours required to find the best integer tour length (in square brackets)
Optimal length is available only for integer tour lengths ACS results on 25 trials

ACS on some geometric TSP problems

Problem name	ACS best integer length (1)	ACS number of tours generated to best	ACS average integer length	Standard deviation	Optimum (2)	Relative error $\frac{(1)-(2)}{(2)} * 100$	CPU sec to generate a tour
d198 (198-city problem)	15,888	585,000	16,054	71	15,780	0.68 %	0.02
pcb442 (442-city problem)	51,268	595,000	51,690	188	50,779	0.96 %	0.05
att532 (532-city problem)	28,147	830,658	28,523	275	27,686	1.67 %	0.07
rat783 (783-city problem)	9,015	991,276	9,066	28	8,806	2.37 %	0.13
fl1577 (1577-city problem)	22,977	942,000	23,163	116	[22,204 – 22,249]	3.27÷3.48 %	0.48

Integer length of the shortest tour found, number of tours to find it, avg integer length (over 15 trials), its std dev, optimal solution, and the relative error of ACS

ACS Extension

Current wisdom says that a very good strategy for the approximate solution of combinatorial optimization problems is the coupling of:

- constructive heuristic, and
- local search

The problem is to find good couplings:

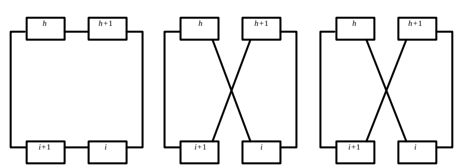
ACO (and other derived algorithms) seems (as shown by experimental evidence) to provide such a good coupling

ACS plus local search

Loop
- Randomly position m agents on n cities
- For step=1 to n
  - For ant=1 to m
    - Apply the state transition rule
    - Apply the local trail updating rule
- Apply local search *each solution is optimized by a problem specific heuristic*
- Apply the global trail updating rule using the best optimized solution

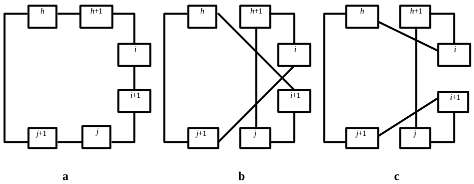
Until End_condition

Local Search



A 2-exchange always inverts a path.

Local Search



A 3-exchange without (b) and with (c) path inversion

ACS-3-opt applied to TSP

Problem name	ACS-3-opt best result (length)	ACS-3-opt best result (sec)	ACS-3-opt average (length) (1)	ACS-3-opt average (sec)	Optimum (2)	% Error (1)-(2) / (2)
d198 (198-city problem)	15,780	16	15,781.7	238	15,780	0.01 %
lin318* (318-city problem)	42,029	101	42,029	537	42,029	0.00 %
att532 (532-city problem)	27,693	133	27,718.2	810	27,686	0.11 %
rat783 (783-city problem)	8,818	1,317	8,837.9	1,280	8,806	0.36 %

Results obtained by ACS-3-opt on TSP problems taken from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20-22, 1996, Nagoya, Japan

Comparison of ACS-3-opt and GA+local search on TSPs

Problem name	ACS-3-opt average (length) (1)	ACS-3-opt average (sec)	ACS-3-opt % error (1)-(3) / (3)	STSP-GA average (length) (2)	STSP-GA average (sec)	STSP-GA % error (2)-(3) / (3)	Optimum (3)
d198 (198-city problem)	15,781.7	238	0.01 %	15,780	253	0.00 %	15,780
lin318 (318-city problem)	42,029	537	0.00 %	42,029	2,054	0.00 %	42,029
att532 (532-city problem)	27,718.2	810	0.11 %	27,693.7	11,780	0.03 %	27,686
rat783 (783-city problem)	8,837.9	1,280	0.36 %	8,807.3	21,210	0.01 %	8,806

Results obtained by ACS-3-opt and by STSP-GA on ATSP problems taken from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20-22, 1996, Nagoya, Japan

ACS-3-opt applied to ATSP

Problem name	ACS-3-opt best result (length)	ACS-3-opt best result (sec)	ACS-3-opt average (length) (1)	ACS-3-opt average (sec)	Optimum (2)	% Error (1)-(2) / (2)
p43 (43-city problem)	2,810	1	2,810	2	2,810	0.00 %
ry48p (48-city problem)	14,422	2	14,422	19	14,422	0.00 %
ft70 (70-city problem)	38,673	3	38,679.8	6	38,673	0.02 %
kro124p (100-city problem)	36,230	3	36,230	25	36,230	0.00 %
ftv170* (170-city problem)	2,755	17	2,755	68	2,755	0.00 %

Results obtained by ACS-3-opt on ATSP problems taken from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20-22, 1996, Nagoya, Japan

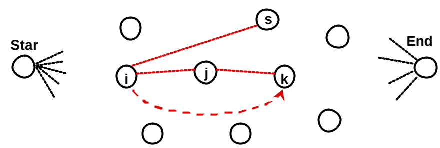
Comparison of ACS-3-opt and GA+local search on ATSPs

Problem name	ACS-3-opt average (length) (1)	ACS-3-opt average (sec)	ACS-3-opt % error (1)-(3) / (3)	ATSP-GA average (length) (2)	ATSP-GA average (sec)	ATSP-GA % error (2)-(3) / (3)
p43 (43-city problem)	2,810	2	0.00 %	2,810	10	0.00 %
ry48p (48-city problem)	14,422	19	0.00 %	14,440	30	0.12 %
ft70 (70-city problem)	38,679.8	6	0.02 %	38,683.8	639	0.03 %
kro124p (100-city problem)	36,230	25	0.00 %	36,235.3	115	0.01 %
ftv170 (170-city problem)	2,755	68	0.00 %	2,766.1	211	0.40 %

Results obtained by ACS-3-opt and by ATSP-GA on ATSP problems taken from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20-22, 1996, Nagoya, Japan

Sequential Ordering Problem

It consists of finding a minimum weight Hamiltonian path on a directed graph subject to multiple precedence constraints among nodes.



SOP models real-world problems like production planning, single-vehicle pick-up and delivery and transportation problems

Gambardella L.M, Dorigo M., An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem, *INFORMS Journal on Computing*, vol.12(3), pp. 237-255, 2000

Sequential Ordering Problem

- Escudero (1988)
- *General ATSP Problem*
  - Precedence Constrained ATSP Polytope (Balas, Fischetti, Pulleyblank, 1995).
  - Branch and Cut (Ascheuer, 1996)
  - Maximum Partial Order/Arbitrary Insertion GA (Chen and Smith, 1996)
- *Pick-Up and Delivery*
  - Lexicographic search with labeling Procedure (Savelsbergh, 1990).

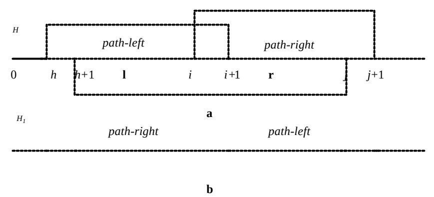
HAS-SOP: Hybrid Ant System for SOP

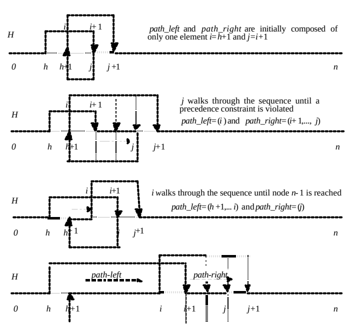
- Costructive phase based on ACS
- Trail updating as ACS
- New local search strategy based on a combination between lexicographic search and a new labeling procedure.
- New data structure to drive the search
- First in literature that uses a local search edge-exhange strategy to directly handle multiple constraints without any increase in computational time.

Ants for SOP

- Each ant iteratively starts from node 0 and adds new nodes until all nodes have been visited and node *n* is reached.
- When in node *i*, an ant chooses probabilistically the next node *j* from the set *F*(*i*) of feasible nodes.
- *F*(*i*) contains all the nodes *j* still to be visited and such that all nodes that have to precede *j*, according to precedence constraints, have already been inserted in the sequence

Local Search





HAS-SOP

PROB	TSPLIB Bounds	MPO/AI Best	MPO/AI Avg	MPO/AI Time (sec)	HAS-SOP Best	HAS-SOP Avg	HAS-SOP Time (sec)
ft70.1.sop	39313	39545	39615	120	39313	39313.0	29.8
ft70.2.sop	[39739,40422]	40422	40435	120	40419	40433.5	114.1
ft70.3.sop	[41305,42535]	42535	42558	120	42535	42535.0	64.4
ft70.4.sop	[52269,53562]	53562	53583	120	53530	53566.5	38.2
kro124p.1.sop	[37722,40186]	40186	40996	240	39420	39420.0	115.2
kro124p.2.sop	[38534,41677]	41667	42576	240	41336	41336.0	119.3
kro124p.3.sop	[40967,50876]	50876	51085	240	49499	49648.8	262.8
kro124p.4.sop	[64858,76103]	76103	76103	240	76103	76103.0	57.4
rbg323a.sop	[3136,3157]	3157	3161	2760	3141	3146.0	1685.5
rbg341a.sop	[2543,2597]	2597	2603	3840	2580	2591.9	2149.6
rbg358a.sop	[2518,2599]	2599	2636	6120	2555	2561.2	2169.3
rbg378a.sop	[2761,2833]	2833	2843	8820	2817	2834.3	2640.3

We tested and compare our algorithms on a set of problems in TSPLIB using a SUN Ultra SPARC 1 (167Mhz)

PROB	TSPLIB Bounds	NEW Lower Bounds	NEW Upper Bounds	HAS-SOP All Best	Avg Result	Std.Dev.	Avg Time (sec)
ESC63.sop	62			62	62.0	0	0.1
ESC78.sop	18230			18230	18230.0	0	6.9
ft53.1.sop	[7438,7570]		7531	7531	7531.0	0	9.9
ft53.2.sop	[7630,8335]		8026	8026	8026.0	0	18.4
ft53.3.sop	[9473,10935]		10262	10262	10262.0	0	2.9
ft53.4.sop	14425			14425	14425.0	0	0.4
ft70.1.sop	39313			39313	39313.0	0	29.8
ft70.2.sop	[39739,40422]	3 9803	40419	40419	40433.5	24.6	114.1
ft70.3.sop	[41305,42535]	41305		42535	42535.0	0	64.4
ft70.4.sop	[52269,53562]	53072	53530	53530	53566.5	7.6	38.2
kro124p.1.sop	[37722,40186]	37761	39420	39420	39420.0	0	115.2
kro124p.2.sop	[38534,41677]	38719	41336	41336	41336.0	0	119.3
kro124p.3.sop	[40967,50876]	41578	49499	49499	49648.8	249.7	262.8
kro124p.4.sop	[64858,76103]			76103	76103.0	0	57.4
prob.100.sop	[1024,1385]	1027	1190	1190	1302.4	39.4	1918.7
rbg109a.sop	1038			1038	1038.0	0	14.6
rbg150a.sop	[1748,1750]			1750	1750.0	0	159.1
rbg174a.sop	2033			2033	2034.7	1.4	99.3
rbg253a.sop	[2928,2987]	2940	2950	2950	2950.0	0	81.5
rbg323a.sop	[3136,3157]	3137	3141	3141	3146.0	1.4	1685.5
rbg341a.sop	[2543,2597]	2543	2574	2574	2591.9	11.8	2149.6
rbg358a.sop	[2518,2599]	2529	2545	2545	2561.2	5.2	2169.3
rbg378a.sop	[2761,2833]	2817	2817	2817	2834.3	10.7	2640.3

The best-known results for many test problems from TSPLIB has been improved by using HAS-SOP

PROB.	RND	MPO /AI	ACS-SOP	RND+LS	MPO/AI+LS	HAS-SOP
prob.100	1440.1%	134.%	40.62%	50.07%	47.58%	**17.46%**
rbg109a	64.57%	0.33%	1.93%	0.08%	0.06%	**0.00%**
rbg150a	37.85%	0.19%	2.54%	0.08%	0.13%	**0.00%**
rbg174a	40.86%	0.01%	2.16%	0.15%	**0.00%**	0.08%
rbg253a	45.85%	0.03%	2.68%	0.21%	**0.00%**	**0.00%**
rbg323a	80.14%	1.08%	9.60%	1.27%	**0.08%**	0.21%
rbg341a	125.46%	3.02%	12.64%	4.41%	**0.96%**	1.54%
rbg358a	151.92%	7.83%	20.20%	4.98%	2.51%	**1.37%**
rbg378a	131.58%	5.95%	22.02%	4.17%	1.40%	**0.88%**
avg	235.38%	17.0%	12.71%	7.27%	5.86%	**2.39%**

Local Search Contribution (+LS): Average Percentages of Deviation from the Best-Known Solution. Results are Obtained over Five Runs of 600 Seconds. Best Results are in Boldface. **RND=Random Restart, MPO/AI=**

Local Search Contribution

No local search			With SOP-3-exchange local search		
Random	**MPO/AI**	**ACS-SOP**	**Random**	**MPO/AI**	**HAS-SOP**
169.26%	**7.59%**	13.44%	3.55%	2.51%	**1.01%**

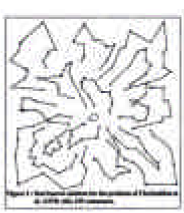
Local Search Contribution (+LS): Average Percentages of Deviation from the Best-Known Solution. Results are Obtained over Five Runs of 600 Seconds on 23 problems . Best Results are in RED. **RND=Random Restart, MPO/AI=**Maximum Partial Order/Arbitrary Insertion, a GA based algorithm by Chen and Smith (1996)

MACO: Multiple Ant Colony Optimization

- In ACO each colony is dedicated to single function optimization.
- In ACO the colony is composed by a set of simple agents which collaborate by communicating.
- We generalize this concept to solve multiple objective function minimization.
- MACO is defined by a colony of ant colonies each one dedicated to minimize a different objective function.
- Colonies (like ants) communicate by exchanging pheromone information.

Vehicle Routing with time Windows

- Problem : to serve a set of customers (with time window constraints) with a fleet of vehicles (with capacity constraints)
- Goal (multiple objective function): minimize the number of vehicles and minimize the travelling distance



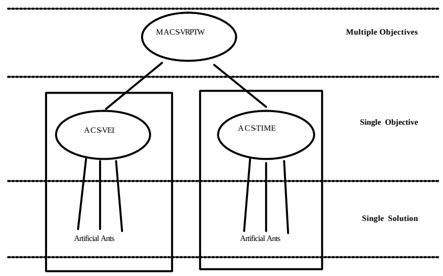
Vehicle Routing with time Windows

- Goal: minimize the number of vehicles and minimize the travelling distance

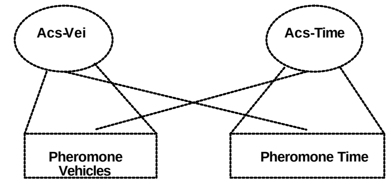
MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows

- One colony is dedicated to vehicles minimization
- The other colony is dedicated to distance minimization.
- The MACO colony is dedicated to synchronize the two colonies.

Gambardella L.M, Taillard E., Agazzi G., MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows , In D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, London, UK, pp. 63-76, 1999



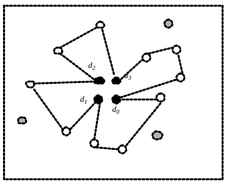
MACS-VRPTW Pheromone Updating

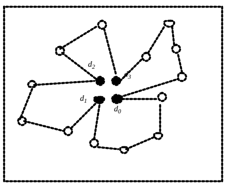


Cross Pheromone Updating Increases Performances

MACS-VRPTW: some details

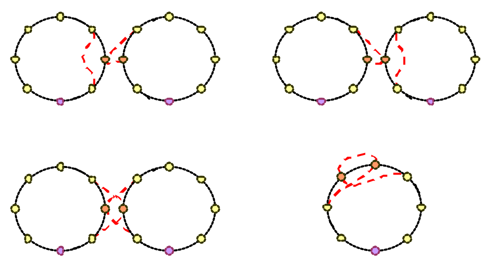
Unfeasible solutions are repaired by insertion procedures





Feasible solutions are improved with local search procedures

Local search exchanges

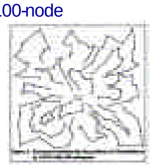


Benchmark problems

With Time Windows (TSPLIB)

56 problems (Solomon, 1987) of six different types (C1,C2,R1,R2,RC1,RC2).

Each data set contains between eight to twelve 100-node problems.



- •C = clustered customers with easy TW.
- •R = customers location generated uniformly randomly over a square.
- •RC = a combination of randomly placed and clustered customers.
- •Sets of type 1 have narrow time windows and small vehicle capacity.
- •Sets of type 2 have large time windows and large vehicle capacity.

Benchmark problems

	R1		C1		RC1		R2		C2		RC2	
	VEI	DIST	VEI	DIST	VEI	DIST	VEI	DIST	VEI	DIST	VEI	DIST
MACS-VRPTW	**12.00**	1217.73	**10.00**	**828.38**	11.63	1382.42	**2.73**	967.75	3.00	**589.86**	**3.25**	1129.19
RT	12.25	1208.50	**10.00**	**828.38**	11.88	1377.39	2.91	961.72	3.00	**589.86**	3.38	1119.59
TB	12.17	1209.35	**10.00**	**828.38**	**11.50**	1389.22	2.82	980.27	3.00	**589.86**	3.38	1117.44
CR	12.42	1289.95	10.00	885.86	12.38	1455.82	2.91	1135.14	3.00	658.88	3.38	1361.14
PB	12.58	1296.80	10.00	838.01	12.13	1446.20	3.00	1117.70	3.00	589.93	3.38	1360.57
TH	12.33	1238.00	10.00	832.00	12.00	1284.00	3.00	1005.00	3.00	650.00	3.38	1229.00

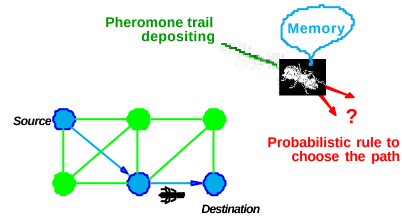
Average of the best solutions computed by different VRPTW algorithms. Best results are in boldface. RT=Rochat and Taillard (1995), TB= Taillard et al. (1997), CR=Chiang and Russel (1993), PB=Potvin and Bengio (1996), TH= Thangiah et al. (1994)

	Old Best			New Best	
Problem	**source**	**vehicles**	**length**	**vehicles**	**length**
r112.dat	RT	10	953.63	**9**	982.140
r201.dat	S	4	1254.09	4	**1253.234**
r202.dat	TB	3	1214.28	3	**1202.529**
r204.dat	S	2	867.33	2	**856.364**
r207.dat	RT	3	814.78	**2**	894.889
r208.dat	RT	2	738.6	2	**726.823**
r209.dat	S	3	923.96	3	**921.659**
r210.dat	S	3	963.37	3	**958.241**
rc202.dat	S	4	1162.8	**3**	1377.089
rc203.dat	S	3	1068.07	3	**1062.301**
rc204.dat	S	3	803.9	3	**798.464**
rc207.dat	S	3	1075.25	3	**1068.855**
rc208.dat	RT	3	833.97	3	**833.401**
tai100a.dat	RT	11	2047.90	11	**2041.336**
tai100c.dat	RT	11	1406.86	11	**1406.202**
tai100d.dat	RT	11	1581.25	11	**1581.244**
tai150b.dat	RT	14	2727.77	14	**2656.474**

New best solution values computed by MACS-VRPTW.
RT=Rochat and Taillard (1995), S = Shaw (1998) TB= Taillard et al. (1997)

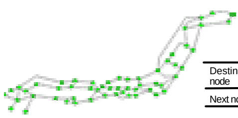
AntNet Applied to Routing in Internet-like Networks

Di Caro and Dorigo, 1997



The Routing Problem

- The practical goal of routing algorithms is to build routing tables



Routing table of node k

Destination node	1	...	j	...	k-1	k+1	...	N
Next node	i_1	...	i_j	...	i_{k-1}	i_{k+1}	...	i_N

- Routing is difficult because costs are dynamic
- Adaptive routing is difficult because changes in the control policy determine changes in the costs and vice versa

AntNet: The Algorithm

- Ants are launched at regular instants from each node to randomly chosen destinations
- Ants are routed probabilistically with a probability function of:
  - (i) some artificial pheromone values, and
  - (ii) some heuristic values, maintained on the nodes
- Ants memorize visited nodes and elapsed times
- Once reached their destination nodes, ants retrace their paths backwards, and update the routing tables

AntNet **is distributed and not synchronized**

Ants' Pheromone Trail Depositing

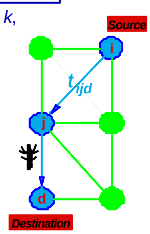
$$\boxed{t^k_{ijd}(t+1) \leftarrow (1-r)\cdot t^k_{ijd}(t) + \Delta t^k_{ijd}(t)}$$

where the (i,j)'s are the links visited by ant k,

and

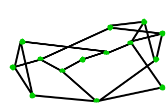
$$\boxed{\Delta t^k_{ijd}(t) = quality^k}$$

where $quality^k$ is set proportional to the inverse of the time it took ant k to build the path from i to d via j

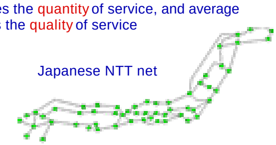


AntNet: Experimental setup

- Realistic simulator (though not industrial)
- Many topologies
- Many traffic patterns
- Comparison with many state-of-the-art algorithms
  (Open Shortest Path First, SPF, Adaptive Bellman-Ford, Q-routing, Predictive Q-routing)
- Performance measures:
  throughput (bit/sec) measures the quantity of service, and average packet delay (sec) measures the quality of service



American NSF net



Japanese NTT net

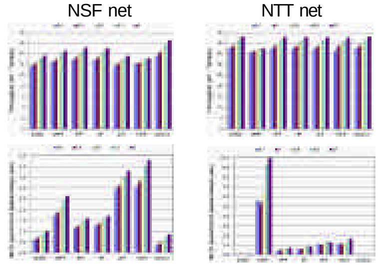
AntNet: Some Results (1)

From Di Caro and Dorigo, 1998, *Journal of Artificial Intelligence Research*

NSF net NTT net

Throughput (b/s)

Avg packet delay (s)

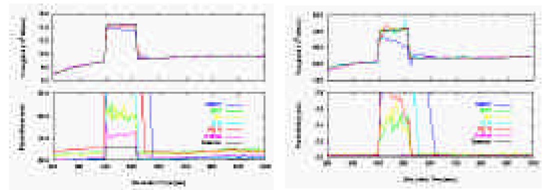


Increasing UP traffic
UP traffic increased by reducing the mean session inter arrival time

AntNet: Adaptiveness

From Di Caro and Dorigo, 1998, *Journal of Artificial Intelligence Research*

NSF net NTT net



Data averaged over a 5 seconds sliding window

The ACO Metaheuristic

Dorigo, Di Caro & Gambardella, *Artificial Life*, 1999

- Ant Colony System and AntNet have been extended so that they can be applied to any shortest path problem on graphs
- The resulting extension is called *Ant Colony Optimization metaheuristic*
- Currently two major application classes:
  - Routing in telecommunications networks
  - *NP*-hard combinatorial optimization problems

The ACO-metaheuristic

procedure *ACO-metaheuristic()*
 while **(not-termination-criterion)**
  schedule subprocedures
   generate-&-manage-ants()
   evaporate-pheromone()
   execute-daemon-actions() ***{Optional}***
  end schedule subprocedures
 end while
end procedure

These are problem specific actions, like local search

Dorigo M., G. Di Caro and L. M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5,2, pp. 137-172, 1999.

From research to Applications

IDSIA

Dyvoil: Dynamic fleet optimization for fuel distribution, Pina Petroli SA, Grancia, CH

Customers

Ask for fuel delivery at home (house heating)

Multiple time windows

Combined delivery (e.g. 2 families)

Stochastic quantity

Accessibility restrictions

AntRoute

Running at MIGROS, the largest supermarket chain in Switzerland (600 shops)

Tours optimisation for non-food palettes distribution with 150-200 vehicles per day

Non-homogeneous fleet

Shop Time Window restriction

Shops accessibility restriction

Tour Minimization

Cost Minimization

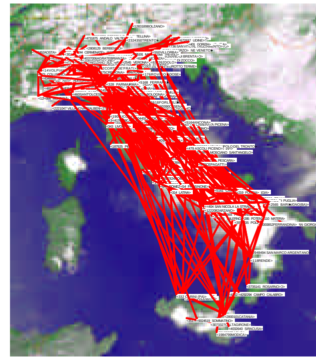
Integration with CADIS and SAP

NUMBER 1 LOGISTICS GROUP

Barilla Group

•Around 700 routes x day

•The company has no own trucks



Number1: the distribution problem

- Pick-up & Delivery: there is not a central depot
- Every order has a source point and a destination point
- Every point of the distribution network has a time window
- Every point of the network has a constant service time
- Heterogeneous point typology: providers, depots, clients
- Homogeneous fleet of vehicles

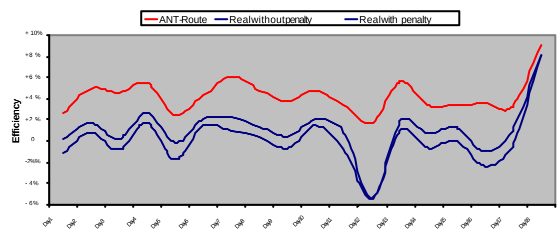
Objective:

Maximization of the average tours' efficiency.

This should implicitly have as a side effect the minimization of the number of tours and of the total km.

Numerical experiments

Efficiency comparison

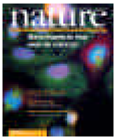


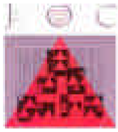
AntRoute as a strategic tool

Conclusion

- ACO is a new meta-heuristic to solve combinatorial optimization inspired by the behavior of real colony of ants.
- The main idea is to let a colony of simple agents collaborate in the search of better and better problem solutions.
- Search space is augmented by artificial pheromone information, that is modified in real time.
- ACO has been able to competitively solve both academic and industrial problems.

Ant Colony Optimization Major Publications





The New York Times
ON THE WEB