Ant Colony Optimization and Aggressive Local Search applied to Bin Packing and Cutting Stock Problems

John Levine and Frederick Ducatelle
Division of Informatics
University of Edinburgh

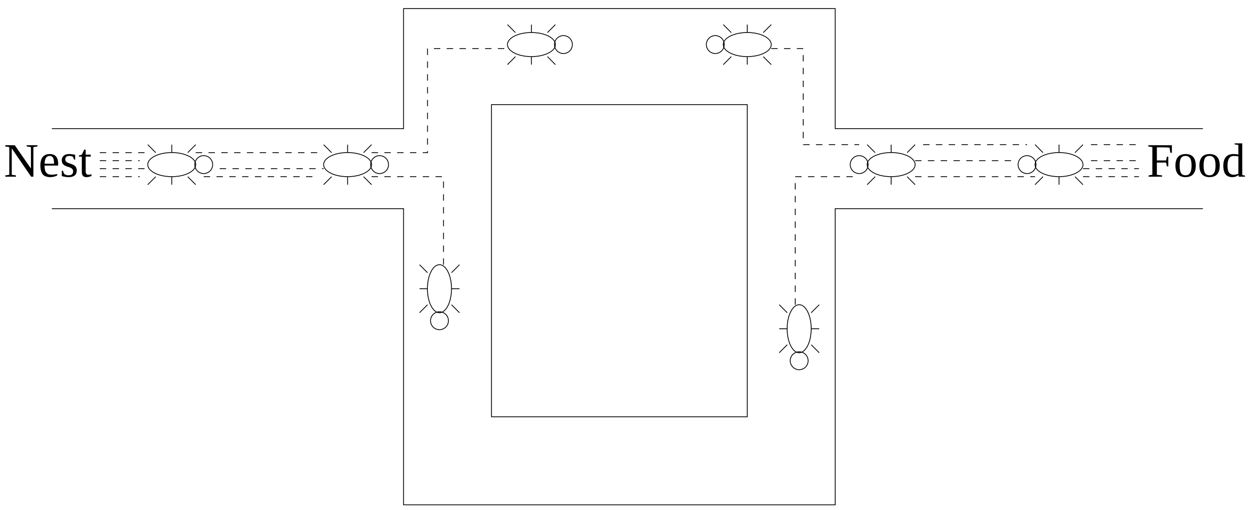
Informatics Jamboree
23rd May 2002

Outline of the Talk

- Introducing ant colony optimization (ACO)
- Introducing bin packing and cutting stock problems
- Applying ACO to bin packing and cutting stock problems
- Comparing to other approaches
- Adding a local search procedure
- Memoryless experiments
- System demonstration
- Conclusions and current directions

Ant Colony Optimization

Biological inspiration: ants find the shortest path between their nest and a food source using *pheromone trails*.



Ant Colony Optimization is a population-based search technique for the solution of combinatorial optimization problems which is inspired by this behaviour.

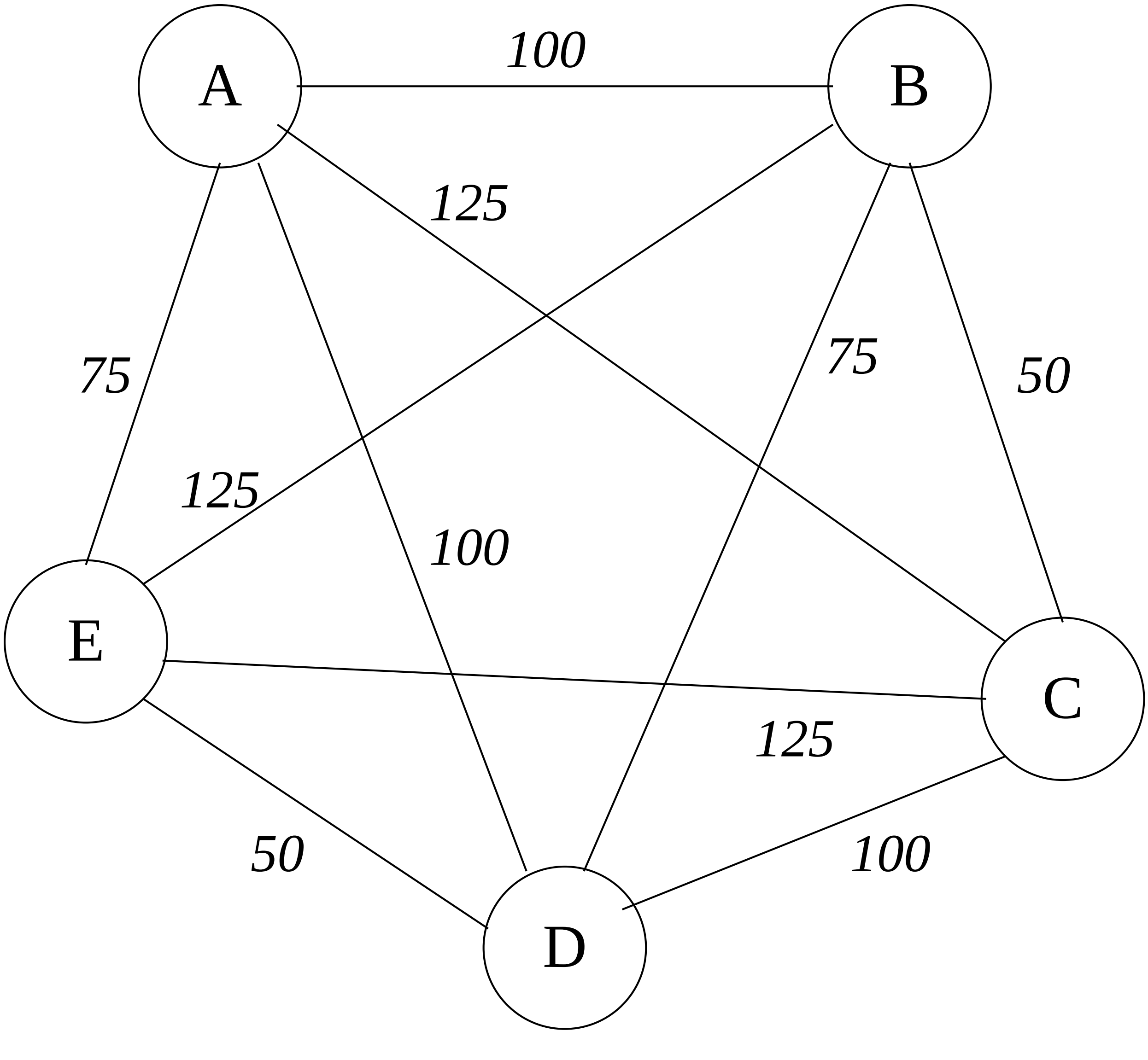
Ant System for the TSP

- Each ant builds a tour from a starting city
- The next city j after city i is chosen stochastically:

$$p(i,j) = \frac{[\tau(i,j)].[\eta(i,j)]^{\beta}}{\sum_{g \in J(i)} [\tau(i,g)].[\eta(i,g)]^{\beta}}$$

- The pheromone trail $\tau(i,j)$ indicates the favorability of city j following city i
- $\eta(i,j)$ is a simple heuristic guiding the construction: $\eta(i,j) = 1/d(i,j)$
- The pheromone trail evaporates a little after every iteration, and is reinforced by good solutions.

Ant System for the TSP: An Example



Ant Colony Optimization: Further Developments

- Improvements to the original algorithm: ACS, MMAS, ASrank, Ant-Q, ANTS, ...

- Combining ACO with local search techniques gives very good results

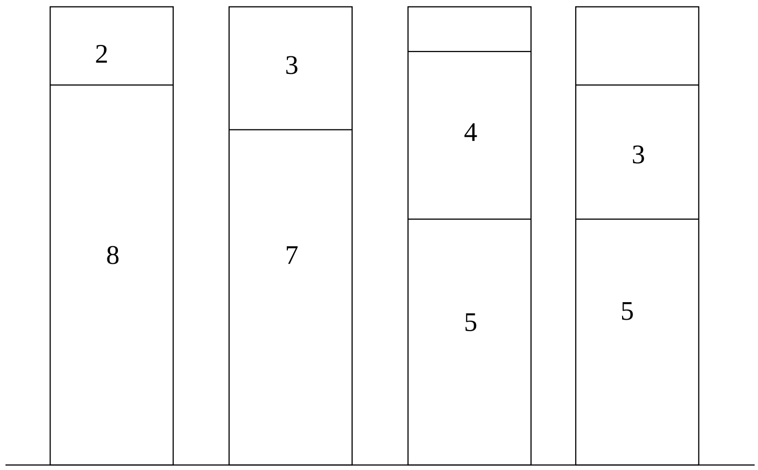
- Other applications: quadratic assignment, flow shop and job shop scheduling, graph coloring, network routing, ...

- References:

1. *Swarm Intelligence: from natural to artificial intelligence* by E. Bonabeau, M. Dorigo and G. Theraulaz
2. *New Ideas in Optimization* by D. Corne, M. Dorigo and F. Glover (eds.)
3. *Ant Colony Optimization* by M. Dorigo and T. Stützle, MIT Press, 2003.

Bin Packing and Cutting Stock Problems

- Packing a number of items in bins of a fixed capacity or cutting items from stocks of a fixed length



- The difference lies in the assortment of small items

- Variations: multiple stock lengths, contiguity, multiple dimensions, ...

Applying ACO to Bin Packing and Cutting Stock Problems

1. How can good packings be reinforced via a pheromone matrix?

2. How can the solutions be constructed stochastically, with influence from the pheromone matrix and a simple heuristic?

3. How should the pheromone matrix be updated after each iteration?

4. What fitness function should be used to recognise good solutions?

AntBin 1: Pheromone Matrix

- BPP and CSP as ordering problems: many permutations are possible

$$\begin{aligned} &| \; 8 \; 2 \; | \; 7 \; 3 \; | \; 5 \; 4 \; | \; 5 \; 3 \; | \\ = &| \; 5 \; 3 \; | \; 7 \; 3 \; | \; 8 \; 2 \; | \; 5 \; 4 \; | \\ = &| \; 3 \; 5 \; | \; 7 \; 3 \; | \; 2 \; 8 \; | \; 5 \; 4 \; | \end{aligned}$$

- BPP and CSP as grouping problems: $\tau(i,j)$ expresses the favorability of having items of size i and j in the same bin/stock

- Pheromone matrix works on item sizes, not items themselves

AntBin 2: Building Solutions

- Every ant starts with an empty bin

- New items are added stochastically:

$$p(s,b,j) = \frac{[\tau_b(j)].[\eta(j)]^\beta}{\sum\limits_{g \in J(s,b)} [\tau_b(g)].[\eta(g)]^\beta}$$

- $\eta(j)$ is the item size j

- $\tau_b(j)$ is the sum of pheromone between item size j and the item sizes already present in bin b

- β has to be defined empirically

AntBin 3: Pheromone Updating

$$\tau(i,j) = \rho.\tau(i,j) + m.f(s_{best})$$

- The pheromone evaporates after every iteration

- There is an update for every time item sizes i and j are combined in a bin/stock of the best solution

- Only the iteration best ant increases the pheromone trail

- Occasionally update with the global best ant instead

AntBin 4: Fitness Function

- Total number of bins in solution: extremely unfriendly fitness landscape – no guidance from $N+1$ bins to N bins

- Need large reward for full or nearly full bins

$$f(s_i) = \frac{\sum_{b=1}^{N} (F_b/C)^2}{N}$$

where N is the number of bins in s_i
F_b is the sum of items in bin b
and C is the bin capacity

- Promotes full bins with the spare capacity in one "big lump"

Pure ACO Results 1: Cutting Stock Problems

Comparing pure ACO to Liang et Al.'s EP solution for the CSP

10 problems, size up to 600 items, 50 runs

Prob	EP			ACO		
	avg	best	time	avg	best	time
6a	80.8	80	347	79.0	79	166
7a	69.0	68	351	69.0	68	351
8a	148.1	147	713	146.0	145	714
9a	152.4	152	1679	151.0	151	1652
10a	220.3	219	4921	218.9	218	4925

Parameters:

$n_{ants} = n_{items}$

$\beta = \{2, 5, 10\}$

n_{sols} = set to match time for EP

Pure ACO Results 2: Bin Packing Problems

Comparing pure ACO to Martello and Toth's Reduction Algorithm and Falkenauer's HGGA

Uniform problems: bin capacity is 150, items are randomly chosen in the range [20,100]

Four sizes: 120, 250, 500 and 1000 items, 20 random instances in each size, 1 run

Prob	HGGA		MTP		ACO	
	bins	time	bins	time	bins	time
u120	+2	381	+2	370	+2	376
u250	+3	1337	+12	1516	+12	1414
u500	0	1015	+44	1535	+42	1487
u1000	0	7059	+78	9393	+70	9272

Parameters:

$n_{ants} = n_{items}$
$\beta = 2$ (u120), 10 (u250, u500, u1000)
$n_{sols} =$ set to match time for HGGA/MTP

ACO Algorithms plus Local Search

- HGGA is a *hybrid* genetic algorithm, consisting of a GA *plus* a local search technique

- Current wisdom suggests that ACO plus local search is also a good hybrid coupling

- Each ant's solution is improved by a local search procedure before the best solutions are reinforced

- ACO algorithm alleviates the initialization problem of local search

Local Search Procedure for the BPP and CSP

- In every ant's solution, the n least full bins are opened and their contents are made free

- Items in the remaining bins are replaced by larger free items

- This gives fuller bins with larger items and smaller free items to reinsert

- The free items are reinserted via FFD

- The procedure is repeated until no further improvement is possible

- Only the global best ant increases the pheromone trail

Local Search: An Example

The solution before local search (the bin capacity is 10):

The bins: | 3 3 3 | 6 2 1 | 5 2 | 4 3 | 7 2 | 5 4 |

Open the two smallest bins:

Remaining: | 3 3 3 | 6 2 1 | 7 2 | 5 4 |
Free items: $5, 4, 3, 2$

Try to replace 2 current items by 2 free items, 2 current by 1 free or 1 current by 1 free:

First bin:	3 3 3 $\rightarrow$ 3 5 2	new free:	$4, 3, 3, 3$
Second bin:	6 2 1 $\rightarrow$ 6 4	new free:	$3, 3, 3, 2, 1$
Third bin:	7 2 $\rightarrow$ 7 3	new free:	$3, 3, 2, 2, 1$
Fourth bin:	5 4 stays the same		

Reinsert the free items using FFD:

Fourth bin: 5 4 $\rightarrow$ 5 4 1
Make new bin: 3 3 2 2
Final solution: | 3 5 2 | 6 4 | 7 3 | 5 4 1 | 3 3 2 2 |

Repeat the procedure: no further improvement possible

Hybrid ACO Results 1: Cutting Stock Problems

Comparing hybrid ACO to Liang et Al.'s EP solution for the CSP

Prob	EP			HACO		
	avg	best	time	avg	best	time
6a	80.8	80	347	79.0	79	1
7a	69.0	68	351	68.0	68	1
8a	148.1	147	713	143.0	143	5
9a	152.4	152	1679	149.0	149	10
10a	220.3	219	4921	215.0	215	249

All 5 problems reliably solved to the theoretical lower bound

Parameters:

$n_{ants} = 10$

$\beta = 2$

$n_{bins} = 4$

$n_{sols} = 20000$

Hybrid ACO Results 2: Bin Packing Problems

Comparing hybrid ACO to Martello and Toth's Reduction Algorithm and Falkenauer's HGGA

Prob	HGGA		MTP		HACO	
	bins	time	bins	time	bins	time
u120	+2	381	+2	370	0	1
u250	+3	1337	+12	1516	+2	52
u500	0	1015	+44	1535	0	50
u1000	0	7059	+78	9393	0	147
u2000	–	–	–	–	0	531
u4000	–	–	–	–	0	7190

Parameters:

$n_{ants} = 10$

$\beta = 2$ (u120-u1000), 1 (u2000, u4000)

$n_{bins} = 4$

$n_{sols} = 20000$

Memoryless Experiments

- Is hybrid ACO really just doing random restart hill-climbing?

- Example: Costa and Hertz graph coloring application

- Method: run AntBin again on both sets of problems, but with the pheromone update "switched off" – gives memoryless random restart hill-climbing

```
trail.decay();
trail.increase(globalBest);
```

- Use exactly the same parameters as previous runs

Memoryless Results 1: Cutting Stock Problems

Comparing hybrid ACO with random restart hill-climbing for the cutting stock problems:

Prob	HACO			No memory		
	avg	best	time	avg	best	time
6a	79.0	79	1	79.0	79	24
7a	68.0	68	1	68.0	68	1
8a	143.0	143	5	144.0	144	1064
9a	149.0	149	10	150.0	150	997
10a	215.0	215	249	216.8	216	1707

Memoryless Results 2: Bin Packing Problems

Comparing hybrid ACO with random restart hill-climbing for the bin packing problems:

Prob	HACO		No memory	
	bins	time	bins	time
u120	0	1	0	1
u250	+2	52	+6	166
u500	0	50	+5	432
u1000	0	147	+10	1850
u2000	0	531	+43	19286
u4000	0	7190	+118	131137

Demonstration

```
snake[antbin] java AntBin problem10a.txt 2 10 4
FFD solution:
  Fitness: 0.9476300904977368  Bins: 221  Waste: 730
Iteration 0: 0.93279210264075 +++ 223 --> 0.97405198776758 +++ 218
Iteration 1: 0.92470982142857 +++ 224 --> 0.98336277521761 +++ 217
Iteration 2: 0.93142688091679 +++ 223 --> 0.99018004115226 +++ 216
Iteration 3: 0.93912537537537 +++ 222 --> 0.99053690843621 +++ 216
Iteration 4: 0.93947384884884 +++ 222 --> 0.99057227366255 +++ 216
Iteration 5: 0.93910097597597 +++ 222 --> 0.99112654320987 +++ 216
Iteration 9: 0.93973473473473 +++ 222 --> 0.99113297325102 +++ 216
Iteration 12: 0.9391072322322 +++ 222 --> 0.99193158436214 +++ 216
Iteration 13: 0.9330026158445 +++ 223 --> 0.99200745884773 +++ 216
Iteration 17: 0.9401076076076 +++ 222 --> 0.99202096193415 +++ 216
Iteration 29: 0.9478381096028 +++ 221 --> 0.99205439814814 +++ 216
Iteration 30: 0.9476150075414 +++ 221 --> 0.99309092078189 +++ 216
Iteration 36: 0.9558345959595 +++ 220 --> 0.99313400205761 +++ 216
Iteration 37: 0.9398354604604 +++ 222 --> 0.99315393518518 +++ 216
Iteration 43: 0.9479901960784 +++ 221 --> 0.99317001028806 +++ 216
Iteration 72: 0.9491855203619 +++ 221 --> 0.99318029835391 +++ 216
Iteration 75: 0.9478425087983 +++ 221 --> 0.99318544238683 +++ 216
Iteration 76: 0.9405311561561 +++ 222 --> 0.99319958847736 +++ 216
Iteration 111: 0.949826546003 +++ 221 --> 0.99324074074074 +++ 216
Iteration 125: 0.933454783258 +++ 223 --> 0.99922803617571 +++ 215
Fitness: 0.9992280361757102 Bins: 215 Waste: 10 Iteration: 125
 | 64:56 | 58:62 | 67:23:30 | 30:44:46 | 64:56 | 22:33:65 | 64:35:
21 | 30:23:67 | 54:66 | 56:64 | 27:66:27 | 41:36:43 | ...
```

Current Directions

- Try on a wider variety of problems

- Other local search methods

- Open random bins in the local search, with bias towards the least full bins

- Adaptive ants for parameter setting:

  - Each ant has a different value of β

  - Make more of the good ants and kill off the bad ones (GA style)