

Ant Colony Optimization

a bio-inspired metaheuristic



Gianni A. Di Caro

gianni@idsia.ch



Dalle Molle Institute for Artificial Intelligence (IDSIA)
Lugano, Switzerland

```
procedure Construction_metaheuristic()  
  S = {set of all feasible solutions};  
  I = {index set for the decision variables};  
  X = {domain set of the decision variables};  
  t  $\leftarrow$  0;  
   $\xi_t \leftarrow \emptyset$ ; {STARTING SOLUTION SET IS EMPTY};  
  J_t  $\leftarrow$  0; {STARTING OBJECTIVE VALUE IS 0};  
  while ( $\xi_t \notin S \wedge \neg \text{termination\_criterion}$ )  
    i_t  $\leftarrow$  select_variable_index(I |  $\xi_t$ );  
    x_{i_t}  $\leftarrow$  assign_variable_value(X |  $\xi_t, i_t$ );  
     $\xi_{t+1} \leftarrow$  include_in_solution( $\xi_t, x_{i_t}$ );  
    J_{t+1}  $\leftarrow$  update_cost(J_t; x_{i_t} |  $\xi_{t+1}$ );  
    t  $\leftarrow$  t + 1;  
  end while  
  if ( $\xi_t \in S$ )  
    return  $\xi_t, J_t$ ;  
  else  
    return {"No feasible solution found"};  
  end if
```

- ▶ **Backtracking:** A `remove_from_solution()` function can be also used to remove variables (and possibly to assign them a different value in a next step) during the construction process (e.g., to repair infeasibility)
- ▶ **Examples:** Greedy algorithms, Rollout algorithms (Bertsekas et al., 1997), Dynamic Programming (Bellman, 1957), Ant Colony Optimization (Dorigo & Di Caro, 1999), GRASP (Pardalos et al., 1995) ...

Example: 0-1 Knapsack problem

$$\begin{array}{ll} \max & Z = \sum_{j=1}^p f_j y_j \\ \text{s.t.} & \sum_{j=1}^p a_j y_j \leq b \\ & y_j \in \{0, 1\}, \quad j = 1, 2, \dots, p \end{array}$$

- ▶ The coefficients of the objective, f_j , and those of the capacity constraint, a_j , are ≥ 0
- ▶ In a construction approach we can iterate over all items (the variables y_j), assigning $y_j = 0$ to not include item j in the solution, or $y_j = 1$ to include j
- ▶ In practice, at the beginning we can set all variables y_j to 0, and then select the items we want to include in the solution until the capacity constraint is exceeded
- ▶ A smarter construction heuristic is the following:
 - ▶ Since in principle we want to include items that bring high profit and have low weight (in order to include as many as possible of them), let's sort the items according to the pseudo-utility ratio $u_j = \frac{f_j}{a_j}$, such that: $u_j \geq u_i \geq u_k \geq \dots$
 - ▶ Once the p items are sorted according to their pseudo-utility, the items are included in the solution one at-a-time, by selecting at each step the unselected item with the highest utility ratio. The procedure stops when the sum of the selected items exceeds the capacity constraint b

```

procedure Improvement_metaheuristic()
   $S = \{\text{set of all feasible solutions}\};$ 
   $\mathcal{N} = \text{neighborhood structure defined on } S;$ 
   $m = \text{memory of search states and solution values};$ 
   $(\xi, m) \leftarrow \text{get\_initial\_solution}(S);$ 
  while ( $\neg \text{termination\_criterion}$ )
     $(\xi', m) \leftarrow \text{step}(\mathcal{N}(\xi), m);$ 
    if ( $\text{accept\_new\_solution}(\xi', \xi, m)$ )
       $\xi \leftarrow \xi';$ 
       $m \leftarrow \text{store\_solution\_if\_new\_best\_solution}(\xi);$ 
    end if
  end while
  if ( $\text{at\_least\_one\_feasible\_solution\_has\_been\_generated}(m, S)$ )
    return  $\text{best\_solution\_found}(m);$ 
  else
    return "No feasible solution found";
  end if

```

- ▶ **Examples:** All the different flavors of Local Search heuristics, Simulated Annealing (Kirkpatrick et al., 1982), Tabu Search (Glover et al., 1995), Genetic Algorithms (?) (Goldberg, 1989) ...

- ▶ **Neighborhood function:** a mapping \mathcal{N} that associates to each feasible solution s of an optimization problem a subset $\mathcal{N}(s)$ of other feasible (or even unfeasible) solutions. The mapping can be conveniently expressed as a rule M that, given s , defines the set of solutions that identify the neighborhood by applying some *modification* procedure (insertion, removal, exchange) on the components of s :

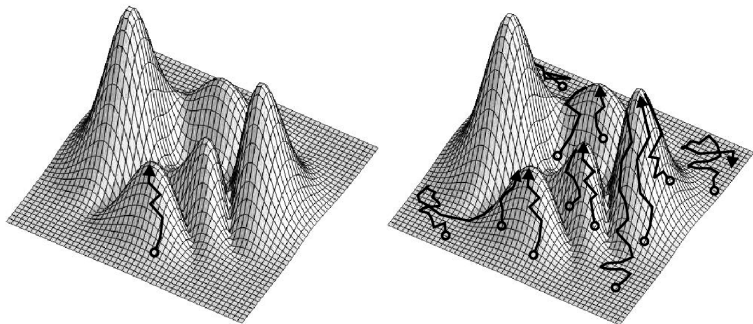
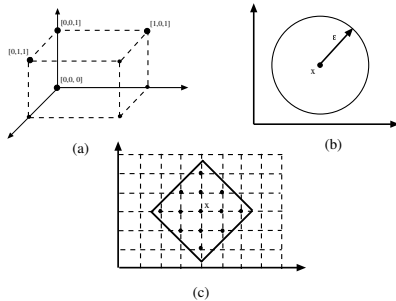
$$\mathcal{N}(s) = \{s' : s' \in S \wedge s' \text{ can be obtained from } s \text{ by applying the rule } M(s)\}$$

- ▶ **Properties of a good neighborhood definition:**
 - ▶ Each solution belongs to its own neighborhood: $s \in \mathcal{N}(s)$
 - ▶ Each solution $s \in S$ can be reached in a finite number of steps by moving between adjacent neighborhoods
 - ▶ The mapping preserves a certain degree of correlation between the value $J(s)$ associated to the point s and the values associated to the points in $\mathcal{N}(s)$
- ▶ A solution $s^* \in S$ is said to be **locally optimal** with respect to the neighborhood function \mathcal{N} (or, also, **\mathcal{N} -optimal**) if (for minimization problems):

$$J(s^*) \leq J(s), \quad \forall s \in \mathcal{N}(s^*)$$

- ▶ A neighborhood is **exact** if the local optima are also global optima
- ▶ A neighborhood is **exponential** if $|\mathcal{N}(S)|$ grows exponentially with problem size
- ▶ The larger the neighborhood, the higher the probability that a locally optimal solution is also globally optimal. On the other hand, the larger the neighborhood, the longer it takes to reach the local optimum → **It's important to find the right balance between size and search**

Examples of neighborhoods and of local search trajectories



▶ Selection of the next solution:

- ▶ Best improvement (complete exploration, 2-Opt, 3-Opt)
- ▶ First improvement (partial exploration, 2-Opt, 3-Opt)
- ▶ Best in the neighborhood (e.g., Tabu Search)
- ▶ Probabilistic selection and acceptance (e.g., Simulated Annealing)
- ▶ ...

▶ Neighborhood exploration strategy:

- ▶ Complete exploration of the neighborhood (2-Opt, 3-Opt)
- ▶ Candidate list (define a smaller neighborhood, 2-Opt, 3-Opt)
- ▶ Don't make the same moves done in the recent past (Tabu Search)
- ▶ Random sampling (e.g., Simulated Annealing)
- ▶ Variable exploration (e.g., final intensification)
- ▶ Variable neighborhood
- ▶ ...

▶ Termination criterion:

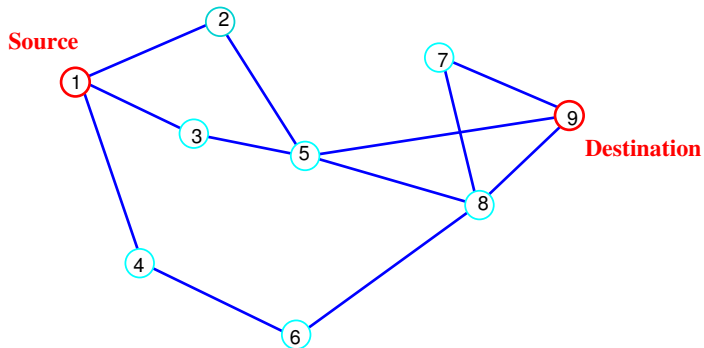
- ▶ Maximum number of checked solutions
- ▶ Maximum CPU time
- ▶ Stagnation / Restarting / Perturbation (e.g., Iterated Local Search)
- ▶ ...

- ▶ A multi-agent metaheuristic with adaptive and learning components whose characteristics result extremely competitive for constrained/distributed/dynamic shortest path problems
- ▶ A swarm intelligence metaheuristic
- ▶ Designed after ant colonies, that precisely display a distributed and adaptive shortest path behavior
- ▶ Any combinatorial problem can be conveniently represented in terms of shortest paths problems . . .

- ▶ **Ant System** [Dorigo et al., 1991], designed for the TSP: the first algorithm reverse engineering the pheromone laying-following ant colony behavior
- ▶ Application of the same ideas to different combinatorial problems (e.g., TSP, QAP, SOP, Routing...)
- ▶ Several **state-of-the-art algorithms** (e.g, ACS [Gambardella & Dorigo, 1996], AntNet [Di Caro & Dorigo, 1998])
- ▶ Definition of the **Ant Colony Optimization (ACO) metaheuristic** [Dorigo, Di Caro & Gambardella, 1999; Di Caro 2004]: abstraction of mechanisms and a posteriori synthesis
- ▶ Convergence proofs, links with other frameworks (e.g., reinforcement learning, Monte Carlo methods, Swarm Intelligence), application to new problems (e.g., scheduling, subset, robotics), workshops, books ... (1999-2010)

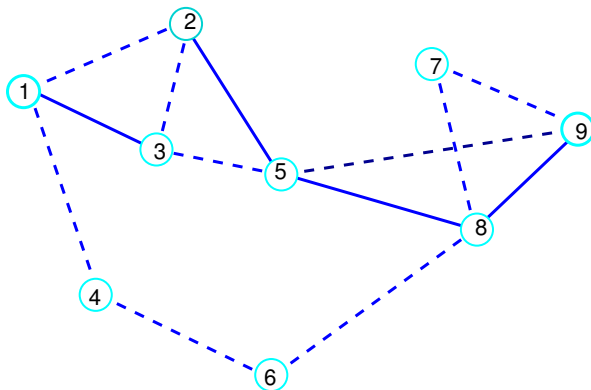
Shortest paths example (1)

- ▶ **Shortest path (SPP):** $C = \{\text{graph nodes}\}$
- ▶ Ex. Sequential decision processes (capacited graph)



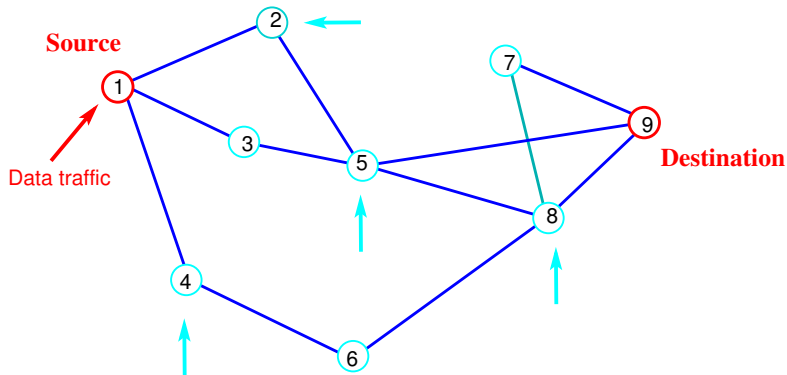
Shortest paths example (2)

- ▶ **Traveling salesman problem (TSP):** $C = \{\text{cities to visit}\}$
- ▶ Ex. Goods delivery
- ▶ Constrained shortest path



Shortest paths example (3)

- ▶ **Data routing:** $C = \{\text{network nodes}\}$
- ▶ Shortest path + Multiple traffic flows to route simultaneously
- ▶ Telecommunication networks



Stigmergy is at the core of most of all the amazing collective behaviors exhibited by the ant/termite colonies (nest building, division of labor, structure formation, cooperative transport)

- ▶ Grassé (1959) introduced this term to explain **nest building in termite societies**
- ▶ Goss, Aron, Deneubourg, and Pasteels (1989) showed how stigmergy allows **ant colonies to find shortest paths** between their nest and sources of food
- ▶ These mechanisms have been **reverse engineered** to give rise to a multitude of ant colony inspired algorithms based on stigmergic communication and control
- ▶ The **Ant Colony Optimization metaheuristic (ACO)** is the most popular, general, and effective swarm intelligence framework based on these principles

Stigmergy means any form of indirect communication among a set of possibly concurrent and distributed agents which happens through acts of local modification of the environment and local sensing of the outcomes of these modifications

- ▶ The local environment's variables whose value determine in turn the characteristics of the agents' response, are called **stigmergic variables**
- ▶ Stigmergic communication and the presence of stigmergic variables is expected (depending on parameter setting) to give rise to a **self-organized global behaviors**
- ▶ **Blackboard/post-it, style of asynchronous communication**

- ▶ **Leading to diverging behavior at the group level:**

- ▶ The height of a pile of dirty dishes floating in the sink
- ▶ Nest energy level in foraging robot activation [Krieger and Billeter, 1998]
- ▶ Level of customer demand in adaptive allocation of pick-up postmen [Bonabeau et al., 1997]

- ▶ **Leading to converging behavior at the group level:**

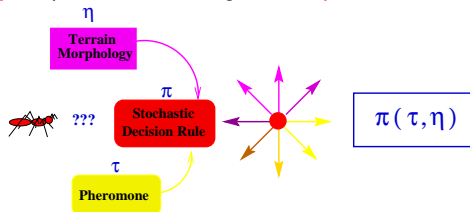
- ▶ Intensity of **pheromone trails** in ant foraging: convergence of the colony on the **shortest path (SP)**

- ▶ While walking or touching objects, the ants lay a volatile chemical substance, called **pheromone**



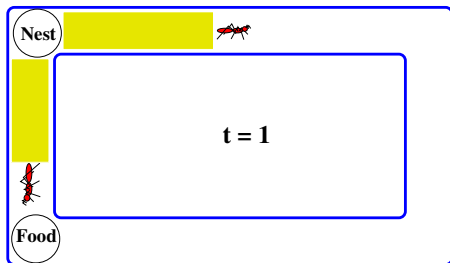
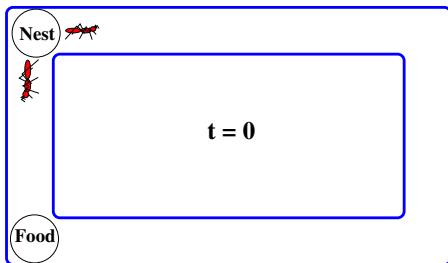
- ▶ Pheromone distribution **modifies the environment** (the way it is perceived by other ants) creating a sort of **attractive potential field** for the ants
- ▶ This is useful for retracing the way back, for mass recruitment, for labor division and coordination, to find shortest paths...

- ▶ While walking, at each step a **routing decision** is issued. Directions locally marked by **higher pheromone intensity** are preferred according to some **probabilistic rule**:

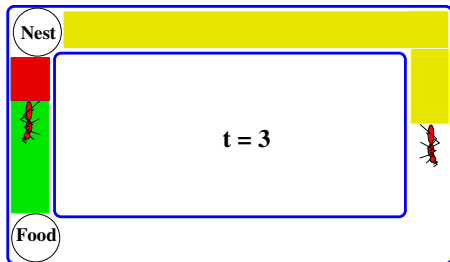
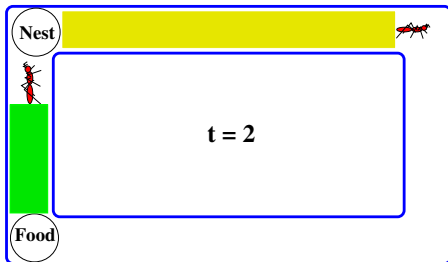


- ▶ This basic pheromone laying-following behavior is the main ingredient to allow the colony converge on the **shortest path** between the nest and a source of food

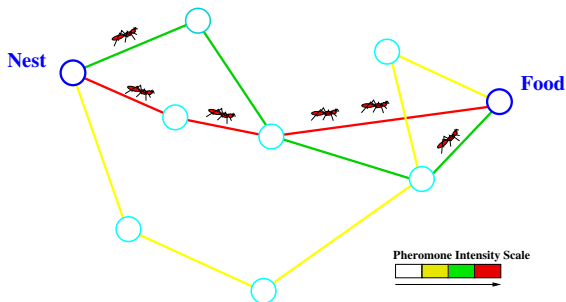
SP behavior: a simple example...



Pheromone Intensity Scale



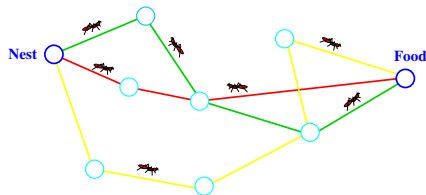
... and a more complex situation



- ▶ Multiple **decision nodes**
- ▶ A path is **constructed** through a **sequence of decisions**
- ▶ Decisions must be taken on the basis of local information only
- ▶ A traveling **cost** is associated to node **transitions**
- ▶ **Pheromone intensity locally encodes decision goodness** as collectively estimated by the **repeated path sampling** (by the ants, agents, robots, ...)

Ant colonies: Ingredients for shortest paths

- ▶ A number of concurrent autonomous (simple?) agents (ants)
- ▶ Forward-backward path following/sampling
- ▶ Local laying and sensing of pheromone
- ▶ Step-by-step stochastic decisions biased by local pheromone intensity and by other local aspects (e.g., terrain)
- ▶ Multiple paths are concurrently tried out and implicitly evaluated
- ▶ Positive feedback effect (local reinforcement of good decisions)
- ▶ Iteration over time of the path sampling actions
- ▶ Persistence (exploitation) and evaporation (exploration) of pheromone

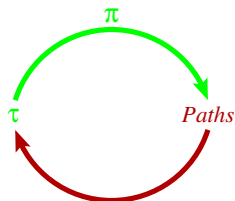


... Convergence onto the shortest path?

What pheromone represents in abstract terms?

- ▶ Distributed, dynamic, and collective memory of the colony
- ▶ Learned goodness of a local move (routing choice)
- ▶ Circular relationship:
pheromone trails modify environment \rightarrow locally bias ants decisions \rightarrow modify environment

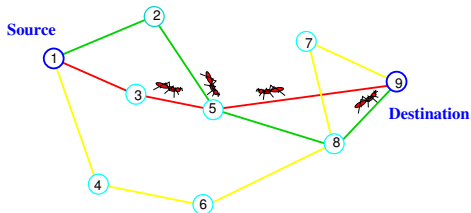
Pheromone distribution biases path construction



Outcomes of path construction are used to modify pheromone distribution

```
procedure ACO_metaheuristic()  
  while ( $\neg$  stopping_criterion)  
    schedule_activities  
      ant_agents_construct_solutions_using_pheromone();  
      pheromone_updating();  
      daemon_actions(); /* optional */  
    end schedule_activities  
  end while  
return best_solution_generated;
```

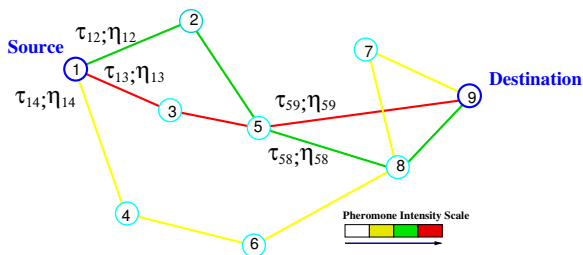
Multiple, iterated solution construction using a stochastic decision policy



- ▶ Each ant is an **autonomous agent that incrementally constructs a path** $\mathcal{P}_{1 \rightarrow 9}$ (i.e., proposes a solution to the problem): **at construction step t ant k represents/holds a partial solution ξ_t^k**
- ▶ There might be one or more ants **concurrently active** at the same time: **ants do not need synchronization or global control and coordination**
- ▶ At each node i (construction step t) the next hop decision is taken according to a **stochastic decision policy π_ϵ** that assigns a selection probability to each feasible next hop $j \in \mathcal{N}(i)$
 - ▶ The decision policy depends on information locally maintained at node i about the estimated quality of each next hop $j \in \mathcal{N}(i)$, this information is maintained in **pheromone** and **heuristic** variable arrays, $\vec{\tau}_i$ and $\vec{\eta}_i$:

$$\pi_\epsilon(i, j; \vec{\tau}_i, \vec{\eta}_i)$$

- ▶ An ant is equivalent to a stochastic construction process locally directed by $\vec{\tau}_i$ and $\vec{\eta}_i$



- ▶ At each decision node i an **array of pheromone variables**: $\vec{\tau}_i = [\tau_{ij}] \in \mathbb{R}, \forall j \in \mathcal{N}(i)$ is available to the ant agent to issue the routing decision
- ▶ $\tau_{ij} = q(j|i)$: estimate of the quality of moving to next node j conditionally to the fact of being in i . Pheromone values are collectively learned by the ants through path sampling
- ▶ At each decision node i an **array of heuristics variables** $\vec{\eta}_i = [\eta_{ij}] \in \mathbb{R}, \forall j \in \mathcal{N}(i)$ is also available to the ant agent to take the routing decision
- ▶ η_{ij} is also an estimate of $q(j|i)$ but it is derived from a process or a priori knowledge not related to the ant actions (e.g., node-to-node distance)

Combining τ and η in the Ant-routing table

- ▶ The values of τ_i and η_i at each node i must be combined in order to assign a unique **quality value** to each locally available next hop $j \in \mathcal{N}(i)$ (to **bias the construction process**):

$$\mathcal{A}_i(j) = f_\tau(\vec{\tau}_i, j) \circ f_\eta(\vec{\eta}_i, j)$$

- ▶ $\mathcal{A}_i(j)$ is called the **Ant-routing table**: it summarizes all the information locally available in j to an ant agent to make next hop selection
- ▶ **Examples (most used combinations)**:
 - ▶ $\tau_{ij}^\alpha \cdot \eta_{ij}^\beta$ (**multiplicative combination**, α and β weighting parameters)
 - ▶ $\alpha\tau_{ij} + (1 - \alpha)\eta_{ij}$ (**additive combination**, α relative weighting parameter)
- ▶ The functional form of f_τ and f_η and of their composition defines how the ant-learned pheromone information and the heuristic information are **combined** and **weighted** in order to locally take optimized decisions
 - ▶ Which is the right balance between τ and η ?

- ▶ At the t -th step of its solution construction process the ant agent k arrives at decision node i and performs the following actions:
 - 1 Calculates the values $\mathcal{A}_i(j)$ for all $j \in \mathcal{N}^F(i) \subseteq \mathcal{N}(i)$ that are still feasible given the status of the solution construction process, that is, given the ant partial solution ξ_t^k
 - 2 Derives from each $\mathcal{A}_i(j)$ a probability value $\pi_\epsilon(i, j)$, that reflects the estimated quality of adding j to the partial solution in the perspective of building a good complete solution to the problem
 - 3 Draws a uniform random number and selects the next hop according to the selection probabilities $\pi_\epsilon(i, j)$ (the larger $\pi_\epsilon(i, j)$, the larger the probability of j being selected)
 - 4 Expand the partial solution ξ_t^k according to the selection resulting from 3.

- ▶ Common choices to define the selection probabilities for the stochastic decision policy:

- ▶ **Random-proportional:** $\pi_\epsilon(i, j) = \frac{\mathcal{A}_i(j)}{\sum_{k \in \mathcal{N}^F(i)} \mathcal{A}_i(k)}$

- ▶ **ϵ -greedy:**
$$\pi_\epsilon(i, j) = \begin{cases} 1 & \text{if } j = \arg \max\{\mathcal{A}_i(j), j \in \mathcal{N}^F(i)\} \\ 0 & \text{otherwise} \end{cases}$$

if $p_b > p_u$:

else : $\pi_\epsilon(i, j) = 1/|\mathcal{N}^F(i)|, \forall j \in \mathcal{N}^F(i)$

$p_u \in [0, 1)$ is a small threshold value for uniform exploration (e.g., $p_u = 0.05$: if the randomly generated value $p_b \in [0, 1]$ is larger than p_u , the best (greedy) selection is issued, otherwise an unbiased uniform selection is issued)

- ▶ **Soft-max:** $\pi_\epsilon(i, j) = \frac{e^{\mathcal{A}_i(j)/T}}{\sum_{k \in \mathcal{N}^F(i)} e^{\mathcal{A}_i(k)/T}}, \quad T \approx 0$

After a solution has been built: Pheromone updating

- ▶ (Real) Ants update pheromone **online step-by-step** → Implicit path evaluation based on on traveling time and rate of updates
 - ▶ Ant's way is inefficient and risky (maybe the followed path is really bad or is not even feasible)
 - ▶ When possible, the “right” way is **online delayed** + **pheromone manager filter**:
 - ▶ **Complete the path**
 - ▶ **Evaluate** (assign a score) and **Select** (is this path worth to be up/down reinforced?)
 - ▶ *Selection* can be performed by a **pheromone manager** that, according to the score, decides whether the pheromone variables associated to the solution should be up/down reinforced or not (e.g., the pheromone manager can use the policy that only the best solution in the current iteration should determine an update in the pheromone variables)
 - ▶ “Retrace” and assign credit / reinforce the quality value of the decisions (pheromone variables) issued to built the path
 - ▶ Total path cost J can be safely used as reinforcement signal (not always trivial to calculate)
- TSP:**
 $s = (1, 3, 5, 7, 9), J(s) = c_{13} + c_{35} + c_{57} + c_{79} + c_{91}, \tau_{13} \leftarrow \tau_{13} + 1/J(s), \tau_{35} \leftarrow \tau_{35} + 1/J(s)$
- ▶ **Online step-by-step**: locally decrease pheromone for exploration (e.g., ACS)
 - ▶ **Offline**: *daemon*, **evaporation**: $\tau_{ij} \leftarrow \rho \tau_{ij}, \rho \in [0, 1],$

```
procedure ant_construct_solution()  
  initialize_ant_parameters();  
   $t \leftarrow 0$ ;  $\xi_0 \leftarrow \emptyset$ ;  $\mathcal{M} \leftarrow \emptyset$ ;  
   $\xi_t \leftarrow$  get_starting_partial_solution();  
   $\mathcal{M} \leftarrow$  update_ant_memory( $\xi_t, 0$ );  
  while ( $\xi_t \notin$  {set of feasible solutions})  
     $\mathcal{A}^t \leftarrow$  read_local_ant_routing_table( $\xi_t$ );  
     $\mathcal{P}^t \leftarrow$  compute_transition_probabilities( $\mathcal{A}^t, \mathcal{M}, \text{PROBLEM\_CONSTRAINTS}$ );  
     $\xi_{t+1} \leftarrow$  apply_ant_decision_policy( $\pi_e, \mathcal{P}^t, \text{PROBLEM\_CONSTRAINTS}$ );  
    move_to_new_state( $\xi_{t+1}$ );  
    if (online_step_by_step_pheromone_update)  
      update_pheromone_variable_used_to_select_the_state_transition();  
      update_ant_routing_table();  
    end if  
     $\mathcal{M} \leftarrow$  update_ant_memory( $\xi_{t+1}, \text{get\_transition\_cost}(\xi_t, \xi_{t+1})$ );  
     $t \leftarrow t + 1$ ;  
  end while  
   $J \leftarrow$  evaluate_constructed_solution( $\mathcal{M}, \xi_t$ );  
return  $\xi_t, J$ ;    RETURN THE COMPLETE SOLUTION AND ITS VALUE
```



```
procedure ant_online_delayed_pheromone_update()  
  update_pheromone_variables_used_in_solution( $J, \mathcal{M}$ );  
  update_ant_routing_tables( $J, \mathcal{M}$ );  
end procedure
```

```

procedure AntSystem()
  Pheromone model:  $\tau_{ij} \equiv$  goodness of selecting city  $j$  when  $i$  is the last added city;
  Heuristic variables:  $\eta_{ij} \equiv 1 /$  distance from city  $i$  to city  $j$ ;
   $m \leftarrow$  number of ants per iteration;
   $nn_{tour} \leftarrow$  find_initial_solution_with_nearest_neighbor();
  Init pheromone:  $\tau_{ij} = \tau_0 = 1 / (n \cdot nn_{tour})$ ,  $\forall i, j \in \{1, \dots, n\}$ ;
  for  $t := 1, \dots, iterations\_num$  do
    for  $k := 1, \dots, m$  do
      ant_construct_solution(¬online_step_by_step_pheromone_update);
    end for
    for  $k := 1, \dots, m$  do // ALL ANTS UPDATE PHEROMONE VARIABLES
      ant_online_delayed_pheromone_update();
    end for
    foreach edge( $i, j$ ),  $i, j \in \{1, \dots, n\}$  // PHEROMONE EVAPORATION ON ALL EDGES
       $\tau_{ij}(t) \leftarrow \rho \tau_{ij}(t - 1)$ ,  $\rho \in [0, 1]$ ;
    end foreach
  end for
return best_solution_generated;

```

▶ Ant-routing table and probabilistic selection policy:

- ▶ ξ_r^k is the partial solution for ant k at the r -th construction step, i is the last city added to ξ_r^k , and \mathcal{N}_i^k indicates the feasible neighbor of ξ_r^k , that is the set of cities that can be feasibly added in i to the solution being constructed by ant k (e.g., $n = 5$, $\xi_3^k = (1, 3, 4)$, $\mathcal{N}(\xi_3^k) = \mathcal{N}_4^k = \{2, 5\}$)

The Ant-routing table at city i at time t : $\mathcal{A}_{ij}(t) = \tau_{ij}(t)\eta_{ij}^\beta$

$$\text{City selection probability in } i, \text{ for ant } k: \pi_{ij}^k(t) = \begin{cases} \frac{\mathcal{A}_{ij}(t)}{\sum_{l \in \mathcal{N}_i^k} \mathcal{A}_{il}(t)} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{otherwise} \end{cases}$$

▶ Pheromone updating (+ evaporation):

$$\tau_{ij}(t) \leftarrow \rho \tau_{ij}(t) + \Delta \tau_{ij}(t)$$

$$\Delta \tau_{ij}(t) = \sum_{k=1}^m \Delta \tau_{ij}^k(t), \quad m = \text{ants per iteration}$$

$$\Delta \tau_{ij}^k(t) = \begin{cases} 1/L^k(t) & \text{if } (i, j) \in T^k(t) \\ 0 & \text{if } (i, j) \notin T^k(t) \end{cases} \quad T^k = \text{tour built by ant } k \text{ at iteration } t, L^k \text{ is its length}$$

```

procedure AntColonySystem()
  Pheromone model:  $\tau_{ij} \equiv$  goodness of selecting city  $j$  when  $i$  is the last added city;
  Heuristic variables:  $\eta_{ij} \equiv 1 /$  distance from city  $i$  to city  $j$ ;
   $m \leftarrow$  number of ants per iteration;
   $nn_{tour} \leftarrow$  find_initial_solution_with_nearest_neighbor();
  Init pheromone:  $\tau_{ij} = \tau_0 = 1 / (n \cdot nn_{tour}), \forall i, j \in \{1, \dots, n\}$ ;
  for  $t := 1, \dots, iterations\_num$  do
    in_parallel  $k := 1, \dots, m$  do // ANTS CONSTRUCT SOLUTIONS IN PARALLEL
       $T^k(t), L^k(t) \leftarrow$  ant_construct_solution(online_step_by_step_pheromone_update);
    end in_parallel
    // ONLY THE BEST ANT TOUR GENERATED SO FAR IS SELECTED TO UPDATE PHEROMONE
    // THE SELECTION CAN BE SEEN AS ISSUED BY A PHEROMONE MANAGER
    best_so_far_ant_update_pheromone( $\{T^k(l), L^k(l)\}, l = 1, \dots, t, k = 1, \dots, m$ );
  end for
return best_solution_generated;

```

► **Ant-routing table and probabilistic selection policy:**

Ant-routing table at city i at time t (for all cities j connected to i):

$$\mathcal{A}_{ij}(t) = \tau_{ij}(t)\eta_{ij}^{\beta} \quad (\beta = 2)$$

State transition rule in i , for ant k (modification of an ϵ -greedy strategy):

$$\text{if } q \leq q_0, \text{ (exploitation) : } \pi_{ij}^k(t) = \begin{cases} 1 & \text{if } j == \arg \max_{s \in \mathcal{N}_i^k} \mathcal{A}_{is}(t) \quad \text{(Greedy choice)} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{if } q > q_0, \text{ (exploration) : } \pi_{ij}^k(t) = \begin{cases} \frac{\mathcal{A}_{ij}(t)}{\sum_{l \in \mathcal{N}_i^k} \mathcal{A}_{il}(t)} & \text{if } j \in \mathcal{N}_i^k \quad \text{(AntSystem rule)} \\ 0 & \text{otherwise} \end{cases}$$

where q is a random variable uniformly distributed in $[0, 1]$, and $q_0 \in [0, 1]$ is a parameter that sets the balance between exploitation and exploration (e.g., $q_0 = 0.9$)

- **Step-by-step pheromone updating** (increase exploration): if edge (i, j) has been just included in the tour ant k is constructing, then decrease the pheromone value τ_{ij} to reduce the probability that the same edge will be included also in the tours being constructed by the other ants

$$\tau_{ij}(t) \leftarrow (1 - \rho)\tau_{ij}(t) + \rho\Delta\tau, \quad \Delta\tau = \tau_0, \quad \rho = 0.1$$

- ▶ Delayed pheromone updating (+ evaporation):

$$\tau_{ij}(t) \leftarrow (1 - \alpha)\tau_{ij}(t) + \alpha\Delta\tau_{ij}^{best}(t), \quad \forall(i, j)$$

$$\Delta\tau_{ij}^{best}(t) = \begin{cases} 1/L^{best}(t) & \text{if } (i, j) \in T^{best}(t) \\ 0 & \text{if } (i, j) \notin T^{best}(t) \end{cases}$$

T^{best} = global-best, best tour built so far by all ants (since the beginning of the iterations)

L^{best} = length of the best so far tour

- ▶ The pheromone on the arcs (i, j) of the best so far tour is increased by $\Delta\tau_{ij}^{best}(t)$, while the pheromone on all the other arcs is decrease because of evaporation $((1 - \alpha)\tau_{ij}(t))$
- ▶ The computational complexity of each iteration is therefore $O(n^2)$ as in the case of AS
- ▶ In the experiments reported in the original paper: $\alpha = \rho = 0.1$
- ▶ The authors have also investigated the use of the iteration-best instead of the global-best

Problem name	ACS	GA	EP	SA	Optimum
Eil50	425 (427.96) [1,830]	428 (N/A) [25,000]	426 (427.86) [100,000]	443 (N/A) [68,512]	425 (N/A)
Eil75	535 (542.37) [3,480]	545 (N/A) [80,000]	542 (549.18) [325,000]	580 (N/A) [173,250]	535 (N/A)
KroA100	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)

- ▶ Small Euclidean instances from TSPLIB
- ▶ GA = Genetic algorithm, EP = Evolutionary programming, SA = Simulated annealing
- ▶ Table shows the best integer tour length, the best real tour length (in parentheses), and the number of tours required to find the best integer tour length (in square brackets)
- ▶ Results out of 25 trials

ACS performance over Euclidean instances from TSPLIB

Problem name	ACS best integer length (1)	ACS number of tours generated to best	ACS average integer length	Standard deviation	Optimum (2)	Relative error $\frac{(1)-(2)}{(2)} \cdot 100$	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
- ▶ Integer length of the shortest tour found and number of tours to find it
- ▶ Average integer length (over 15 trials) and its standard deviation
- ▶ Value of the optimal solution, and relative approximation error
- ▶ Cpu time to generate a single tour

procedure AntColonySystem-3-Opt()

Pheromone model: $\tau_{ij} \equiv$ goodness of selecting city j when i is the last added city;
 Heuristic variables: $\eta_{ij} \equiv$ distance from city i to city j ;
 $m \leftarrow$ number of ants per iteration;
 $nn_{tour} \leftarrow$ find_initial_solution_with_nearest_neighbor();
 Init pheromone: $\tau_{ij} = \tau_0 = 1/(n \cdot nn_{tour}), \forall i, j \in \{1, \dots, n\}$;
for $t := 1, \dots, iterations_num$ **do**
 in_parallel $k := 1, \dots, m$ **do** // ANTS CONSTRUCT SOLUTIONS IN PARALLEL
 $T^k(t), L^k(t) \leftarrow$ ant_construct_solution(online_step_by_step_pheromone_update);
end in_parallel
foreach $T^k(t), k := 1, \dots, m$ **do**
 // EACH ANT SOLUTION IS TAKEN AS A STARTING POINT FOR A 3-OPT LOCAL SEARCH THAT IMPROVES IT
 $T^k(t), L^k(t) \leftarrow$ run_3-opt_local_search_starting_from_ant_tour($T^k(t)$);
end foreach
 best_so_far_tour_update_pheromone($\{T^k(l), L^k(l)\}, l = 1, \dots, t, k = 1, \dots, m$);
end for
return best_solution_generated;

ACS-3-Opt performance on TSP

Problemname	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 from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20–22, 1996, Nagoya, Japan

Problemname	ACS-3-opt average (length) (1)	ACS-3-opt average (sec) (2)	ACS-3-opt % error (1)-(3) ----- (3)	STSP-GA average (length) (2)	STSP-GA average (sec) (3)	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 from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20–22, 1996, Nagoya, Japan
- ▶ Symmetric instances
- ▶ STSP_GA is a GA with a Lin-Kernighan local search (called right after the crossover operator in order to produced a population of locally optimized individuals)

ACS-3-Opt performance on Asymmetric 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)
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 %

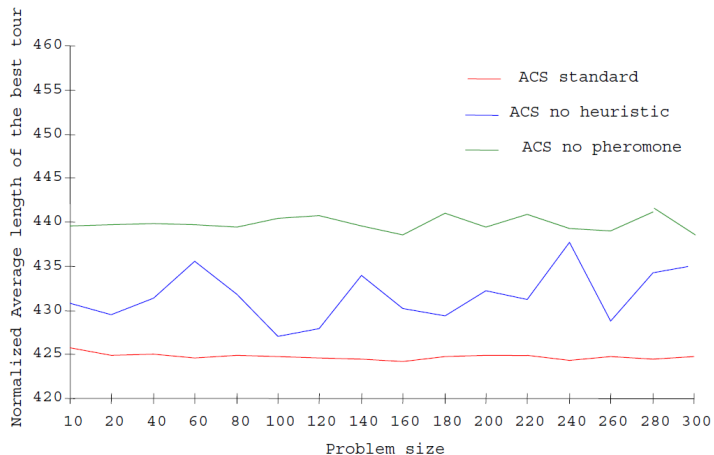
- ▶ Results from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20–22, 1996, Nagoya, Japan
- ▶ Asymmetric instances

ACS-3-Opt performance on ATSP vs. GA+LS

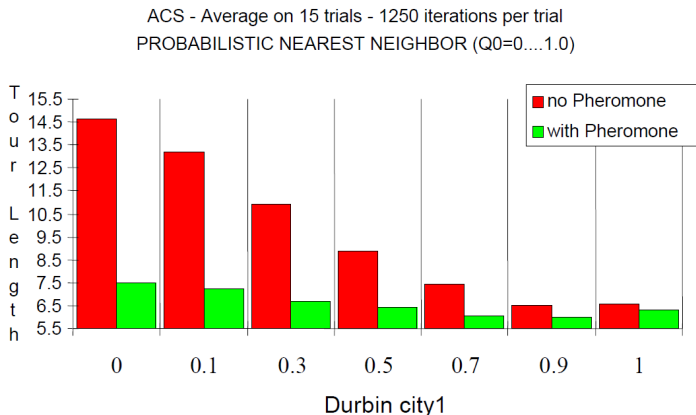
Problem name	ACS-3-opt average (length) (1)	ACS-3-opt average (sec) (2)	ACS-3-opt % error (1)-(3) ----- (3)	ATSP-GA average (length) (2)	ATSP-GA average (sec) (3)	ATSP-GA % error (2)-(3) ----- (3)
p43 (43-city problem)	2,810	2	0.00 %	2,810	10	0.00 %
ry48p (48-city problem)	14,422	19	0.00 %	14,440	30	0.12 %
ft70 (70-city problem)	38,679.8	6	0.02 %	38,683.8	639	0.03 %
kr o124p (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 from the First International Contest on Evolutionary Optimization, IEEE-EC 96, May 20–22, 1996, Nagoya, Japan
- ▶ Small asymmetric instances

Impact of pheromone and of heuristic information



AS vs. Probabilistic Nearest Neighbor



- ▶ ACS without pheromone variables becomes equivalent to a probabilistic nearest neighbor
- ▶ On the x-axis different values for q_0 , the parameter that control the degree of randomness in the ACS selection rule

- ▶ *Max–Min* AS (Stuetzle, 1998) is similar to ACS but adds new ideas of general applicability:

- ▶ Pheromone values are bounded to an explicit value range:

$$\tau_{ij} \in [\tau_{min}, \tau_{max}] \subseteq \mathbb{R}, \quad \forall i, j.$$

This serves to maintain exploration by avoiding that the pheromone level of a choice goes to 0 or that a choice gets a too high level of pheromone. On the other hand, online pheromone updating, which was used in ACS to increase exploration, is not used in *MMAS*

- ▶ Pheromones variables are initialized to their maximum value τ_{max}
- ▶ At each iteration, pheromone variables are updated either for the best solution from the iteration (**local best**) or for the best solution found so far (**global best**):

$$\tau_{ij} \leftarrow \rho \tau_{ij} + \Delta \tau_{ij}^{best}, \quad \Delta \tau_{ij}^{best} = 1/L^{best}$$

- ▶ If the algorithm does not generate a new best solution over a fixed number of iterations and most of the pheromone values are close to τ_{min} , a daemon component operates a **restart** by reinitializing pheromone values and the value of the best so far solution to τ_{max}

- ▶ In AS_{rank} (Bullnheimer et al., 1999) the ants that at each iteration are allowed to update pheromone variables are chosen according to the following strategy:
 - ▶ If $J_i(t)$ is the cost of the solution generated by the i -th ant at iteration t , the m ants are ranked according to J 's values and only the best $\sigma - 1$ ants in the ranking are allowed to update pheromone on their solutions, for an amount of pheromone proportional to the ant rank
 - ▶ Also the edges included in the best so far solution receive pheromone, scaled by σ
- ▶ The overall dynamics is therefore:

$$\tau_{ij}(t) \leftarrow (1 - \rho)\tau_{ij}(t) + \sigma\Delta\tau_{ij}^+(t) + \Delta\tau_{ij}^r(t)$$

where $\Delta\tau_{ij}^+(t) = 1/J^+(t)$ and $\Delta\tau_{ij}^r(t) = \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu}(t)$, with $\Delta\tau_{ij}^{\mu}(t) = (\sigma - \mu)/J^{\mu}(t)$ if the ant with rank μ has included pair (c_i, c_j) in its solution and $\Delta\tau_{ij}^{\mu}(t) = 0$ otherwise. $J^{\mu}(t)$ is the cost of the solution constructed by the ant with rank μ at iteration t .

- ▶ As a matter of fact, the best instances of ACO algorithms for (static/centralized) combinatorial problems are those making use of a problem-specific local search daemon procedure
- ▶ It is conjectured that ACO's ants can provide **good starting points for local search**. More in general, a construction heuristic can be used to quickly build up a complete solution of good quality, and then a modification procedure can take this solution as a starting point, trying to further improve it by modifying some of its parts
- ▶ This **hybrid two-phases search** can be iterated and can be very effective if each phase can produce a solution which is locally optimal within a different class of feasible solutions, with the intersection between the two classes being minimal

Problem representation for taking decisions: Pheromone model (1)

- ▶ A proper formal description of what a pheromone model is it would require a long discussion (see Di Caro, 2004), so let's try to get a practical understanding through the use of examples ...
- ▶ **Construction approach:** it starts at time step $t = 0$ with an empty partial solution $\xi_0 \leftarrow \emptyset$ (that represents the **state of the construction process**) and then, at the subsequent steps $t = 1, \dots, n$, iteratively adds solution components c_t to ξ_t until a complete, feasible, solution has been built:

$$\xi_{t+1} \leftarrow \xi_t \oplus c_t$$

where the \oplus operator indicates that the selected solution component c_t is added to the partial solution ξ_t according to the selected modality (e.g., insertion, extension, inclusion):

- ▶ **TSP (sequence):** if $\xi_t = (c_0, c_1, \dots, c_{t-1})$ is a sequence of t distinct cities, c_t will be a new city ($\notin \xi_t$) that can be added at one of the two ends of the sequence, or inserted at a some point:
 - ▶ Sequence extension: $\xi_{t+1} \leftarrow (\xi_t, c_t)$, or $\xi_{t+1} \leftarrow (c_t, \xi_t)$
 - ▶ Insertion in sequence: $\xi_{t+1} \leftarrow (c_0, c_1, \dots, c_j, c_t, c_{j+1}, \dots, c_{t-1})$
- ▶ **Set covering:** if ξ_t is a set of columns, c_t is a new column added to the set: $\xi_{t+1} \leftarrow \xi_t \cup c_t$
- ▶ **Knapsack:** if ξ_t is a set of items in the knapsack, c_t is a new item added to the set:
 $\xi_{t+1} \leftarrow \xi_t \cup c_t$
- ▶ **TSP (set):** if ξ_t is a set of edges ($city_i, city_j$) partially disjoint (until a complete tour is built), c_t is a new edge (feasible given ξ_t) which is added to the set: $\xi_{t+1} \leftarrow \xi_t \cup c_t$

- ▶ At each step of the construction process, the decision about the next component to add to the partial solution involves two aspects:
 - ▶ Feasibility
 - ▶ Optimization
- ▶ Both can be approached *locally*, considering the characteristics current partial solution, and/or *globally*, in the perspective of the complete solution that will be built
 - ▶ Example: the partial solution can be made infeasible and be *repaired* in future steps
 - ▶ Example: a *greedy* optimization policy implements a myopic optimization strategy
- ▶ **ACO: the pheromone model deals with the optimization part of the decision, while feasibility is dealt by some ad hoc mechanism, which is usually simple and not based on future repair**
 - ▶ **Feasibility check:** If ξ_t is the current ant state, that is the current partial solution of an ant construction process, $\mathcal{N}(\xi_t)$ is the feasible neighbor of ξ_t , that is the set of feasible components that can added to ξ_t
 - ▶ 5-cities TSP: $\xi_3 = (1, 3, 4)$, $\mathcal{N}(\xi_3) = \{1, 2, 3, 4, 5\} \setminus \{\xi_3\} = \{2, 5\}$
 - ▶ 6-columns Set Covering: $\xi_4 = \{1, 2, 5, 6\}$, $\mathcal{N}(\xi_4) = \{1, 2, 3, 4, 5, 6\} \setminus \{\xi_4\} = \{3, 4\}$
 - ▶ 5-items Knapsack: $\xi_3 = \{1, 3, 5\}$,
 $\mathcal{N}(\xi_3) = \{i \in \{1, 2, 3, 4, 5\} \setminus \{\xi_3\} \mid w_i + (w_1 + w_3 + w_5) \leq W\}$

- ▶ **Pheromone / optimization:** $f(\xi_t)$ = function that implements some feature extraction of the state ξ_t , a pheromone variable is a variable that quantifies $q(c_k|f(\xi_t)), \forall c_k \in \mathcal{N}(\xi_t)$, that is, the estimated quality (estimated by solution sampling) of adding component c_k into the current partial solution ξ_t
- ▶ A pheromone variable expresses how good is to make the **state transition** $\xi_t \rightarrow \xi_{t+1} = \xi_t \oplus c_k$
- ▶ f is a state feature extraction function that defines which features from the current state should be taken into account to take optimized decisions (f is set by the algorithm designer!)
 - ▶ **TSP:** if $\xi_t = (c_1, \dots, c_{t-1}, c_t)$ and $f(\xi_t) = c_t$, all state information before the last included city is discarded and to guide decisions we define $(n-1)^2$ pheromone variables of type:

τ_{ij} = how good is to add city j when the ant is at city i in tour construction

- ▶ **TSP:** other choices are possible, for instance, $f(\xi_t) = \{c_t, c_{t-1}\}$, in this case we need to define $\binom{n}{2}(n-2)$ pheromone variables of type:

$\tau_{\{i,k\}j}$ = how good is to add city j when the last two cities correspond to the set $\{i, k\}$

- ▶ **Set cover:** $f(\xi_t) = \emptyset$, in this case we have n = number of columns, pheromone variables of type:

τ_j = how good is to add column j in the solution

- ▶ **Set cover:** $f(\xi_t) = |\{\xi_t\}|$, or $f(\xi_t)$ = number of rows covered by ξ_t , or ...

- ▶ The pheromone model implements the way the original problem is represented for taking decisions
 - ▶ “Representation” always plays a major role for the success of an algorithm (e.g., chromosome in genetic algorithms, neighborhood in improvement-based local search)
- ▶ Pheromone variables are the parameters of the decision policy of the ants, and are the learning target of the collective ant sampling actions
- ▶ The definition of a “good” pheromone model is essential for the success of an ACO algorithm. Pheromone variables have to be learned, therefore:
 - ▶ It is necessary to define the right balance between retained state information and computational aspects (number of pheromone variables)
 - ▶ Part of the state information must be necessarily discarded for practical computational reasons. If the number of solutions of a combinatorial optimization problem is huge, the number of partial solutions is more than huge!
 - ▶ *Dynamic programming* is an exact construction approach that uses full state information

- ▶ **Representation of the problem (pheromone model $\bar{\tau}$):** what are the most effective state features to consider to learn good decisions?
- ▶ **Heuristic variables $\bar{\eta}$:** what are they and what's their relative weight with respect to the pheromone in the decisions?
- ▶ **Ant-routing table $\mathcal{A}(\tau, \eta)$:** how pheromone and heuristic variables are combined together to define the goodness of a decision?
- ▶ **Stochastic decision policy π_ϵ :** how to make good exploration without sacrificing exploitation of promising/good actions?
- ▶ **Policies for pheromone updating:** online, offline?
- ▶ **Scheduling of the ants:** how and/or how many per iteration?
- ▶ **What about restarting after stagnation?**
- ▶ **Is it useful to put limits in pheromone ranges?**
- ▶ **Pheromone initialization and evaporation, other constants, ... ?**
- ▶ **Daemon components:** what's the effect of local search?

- ▶ State feature (pheromone model): **Last component** (for assignment problems)
- ▶ Heuristic variables: **Problem's costs or lower bounds**
- ▶ Ant-routing functions: **Multiplicative or additive**
- ▶ Decision policy: **ϵ -greedy and random proportional**
- ▶ Pheromone updating: **Elitist strategies**
- ▶ Number of ants: **a few ants per a large number of iterations**
- ▶ Pheromone values: **Bounded ranges**
- ▶ Daemon procedures: **Problem-specific local search**

- ▶ Few proofs of **asymptotic probabilistic convergence**:
 - ▶ in **value**: ants will find the optimal solution
 - ▶ in **solution**: all the ants will converge on the optimal solution
- ▶ Only mild mathematical assumptions
- ▶ Convergence results are mainly valid for TSP-like problems
- ▶ It's important to put finite bounds on the pheromone values

- ▶ M. Dorigo and G. Di Caro, The Ant Colony Optimization Meta-Heuristic, in *New Ideas in Optimization*, D. Corne, M. Dorigo and F. Glover editors, McGraw-Hill, pages 11–32, 1999
- ▶ M. Dorigo, G. Di Caro and L. M. Gambardella, Ant Algorithms for Discrete Optimization, *Artificial Life*, Vol. 5(2), pages 137-172, 1999
- ▶ G. Di Caro, *Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks*, Ph.D. thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, 2004
- ▶ M. Dorigo and T. Stuetzle, *Ant Colony Optimization*, MIT Press, Cambridge, MA, 2004
- ▶ M. Dorigo and L. M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, Vol. 1(1), pages 53-66, 1997

Example: ANTS: Approximate Non-deterministic Tree-Search (Maniezzo, 1999)

→ ACO + Relaxations

- ▶ *Idea*: the heuristic attractiveness η_{ij} of a move can be estimated by means of lower bounds on the cost of the completion of a partial solution → combining mathematical programming techniques with metaheuristics

For each feasible component j to select for a state transition, $\xi_t \oplus j \rightarrow \xi_{t+1}$, the heuristic function η_{ij} is defined as the lower bound obtained by fixing ξ_{t+1} in the solution and then solving with a relaxation (e.g., an AP relaxation for the TSP or QAP)

$$\eta_{ij}(t) = LB(\xi_{t+1})$$

- ▶ Apart from this, the ANTS matheuristic follows the same general approach as other ACO algorithms with the following additional differences . . .

- ▶ At the beginning a lower bound LB for the problem is computed and its value is used to initialize pheromone variables
- ▶ (Usually) an additive formula is used as ant-routing function, weighting τ_{ij} and η_{ij} in more or less equally ($\alpha \approx 0.5$):

$$A_{ij}(t) = \alpha\tau_{ij}(t) + (1 - \alpha)\eta_{ij}(t)$$

- ▶ Full partial states could be used, that is, pheromone variables could be associated to pairs: $(\xi_t, j \in \mathcal{N}(\xi_t))$
- ▶ Pheromone updating uses **dynamic scaling** to better discriminate small improvements in the latest stage of the search:

$$\Delta\tau_{ij}^k = \tau_0 \left(1 - \frac{J^k - LB}{\langle J \rangle - LB} \right),$$

where J^k = Value of the solution generated by the k -ant, and $\langle J \rangle$ = exponential average of the last N (5-10) solutions

Matheuristics is a hot topic in combinatorial: optimization

- ▶ V. Maniezzo, T. Stuetzle, S. Voß, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, Annals of Information Systems, Springer, 2008

Example of using the ANTS approach for a 10-cities TSP

- ▶ A convenient lower bound for the TSP can be obtained by considering the relaxation resulting from the associated Assignment Problem (AP) \equiv TSP formulation minus the Hamiltonian tour constraints
- ▶ During each iteration of the algorithm, each ant constructs a solution as usual. Let ξ_4^a be the state of ant a after 4 steps, with $\xi_4^a = (1, 5, 4, 2)$ The last included city is $i = 2$, and the feasible neighbor is $\mathcal{N}_i^a \equiv \mathcal{N}(\xi_4^a) = \{3, 6, 7, 8, 9, 10\}$
- ▶ The heuristic attractiveness η_{ik} for a city $k \in \mathcal{N}_i^a$ is calculated as:

- ▶ For each $k \in \{3, 6, 7, 8, 9, 10\}$ do

1 Consider the partial solution $\xi_5^a(k)$ that would result by adding city k to the partial solution ξ_4^a :
 $\xi_5^a(k) = (1, 5, 4, 2, k)$

2 Calculate a lower bound $LB(\xi_5^a(k))$ based on this partial solution $\xi_5^a(k)$. That is, in the formulation of the AP relaxation fix all the edges associated to $\xi_5^a(k)$:

$$\begin{aligned} \min \quad & LB(\xi_5^a(k)) = \sum_{i=1}^{10} \sum_{j=1}^{10} d_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{10} x_{ij} = 1 \quad j = 1, \dots, 10 \\ & \sum_{j=1}^{10} x_{ij} = 1 \quad i = 1, \dots, 10 \\ & x_{15} = x_{54} = x_{42} = 1 \\ & x_{2k} = 1 \\ & x_{ij} \in \{0, 1\} \quad i, j \in \{1, \dots, 10\} \end{aligned}$$

These are from ξ_4^a

This results from the considered extension of ξ_4^a with k

d_{ij} is the distance between cities i and j , and the binary variable x_{ij} sets the inclusion of edge (i, j)

3 Assign the solution $LB(\xi_5^a(k))$ of the AP relaxation to the heuristic attractiveness:

$$\eta_{ik} \leftarrow LB(\xi_5^a(k))$$