Ant System (AS)

\begin{verbatim}
Loop
  Place one ant on each city \(i\) (there are \(n\) cities)
  For step = 1 to \(n\) do:
    For \(k = 1\) to \(n\) do:
      Each ant adds a city to its path \(i\)
      Choose the next city to move to applying a
      probabilistic state transition rule
    End-for
  End-for
Update pheromone trails
Until End condition
\end{verbatim}


PSO vs. AS

- Particle + velocity vs. Ant + pheromone
  - Internal memory (pbest) vs. no-memory
- Communication style?
  - Direct vs. indirect
  - P2P vs. broadcasting
- Updating style?
  - Particle position and velocity are updated back-to-back while new ant tours and pheromone are produced separately.
### AS Transition Rules

- **Probabilistic Transition Rule:**
  \[
  p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in J_i \cup J_k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}
  \]

### AS Pheromone update Rule

- **Pheromone Update Rule:**
  \[
  \tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta \tau_{ij}(t)
  \]
  \[
  \Delta \tau_{ij}(t) = \sum_{l \in J_i \cup J_k} \Delta \tau_{il}^l(t)
  \]
  \[
  \Delta \tau_{il}^l(t) = \text{quality}^l
  \]

*Pheromone on all edges decrease with the same percentage*

*Pheromone on more visited edges that are parts of better tours are increased more than that on other edges.*

### Ant Colony System (ACS)

*Gambardella & Dorigo, 1996*

- **New transition rule (pseudo random proportional action rule):**
  \[
  j = \arg \max_{u \in J} \left\{ \frac{[\tau_{ui}(t)]^\alpha [\eta_{ui}]^\beta}{\sum_{l \in J_i} [\tau_{ul}(t)]^\alpha [\eta_{ul}]^\beta} \right\}, \quad r = r_0
  \]

  - \(r_0\) is a parameter that can be tweaked, \(r \sim U(0,1)\).
  - \(J\) is a city selected according to the probability calculated previously, with \(\alpha=1\).

- **New Transition Rule**
  \[
  p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in J_i \cup J_k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}
  \]

  - \(r_0\) is used to balance exploration and exploitation.
    - \(r \leq r_0\): select the edge that has the highest pheromone concentration and has the shortest length (exploitation).
    - \(r > r_0\): an edge is chosen probabilistically according to its distance to node i and the amount of pheromone on the edge (exploration).
    - \(r_0\) is set as 0.9 (more exploitation) in their experiments, with \(\beta=2\) (more greedy as \(\alpha=1\)).
New Pheromone Global Update Rule

- Similar to AS, global update is performed after all ants have completed their tours.
- Similar to AS, pheromone evaporates on all edges with the same percentage.
- However, new pheromone only added to the edges that belong to the best tour. (more exploitation or exploration?)

\[ \tau_{ij}(t) = (1 - \rho_1) \tau_{ij}(t) + \rho_1 \Delta \tau_{ij}(t) \]
\[ \Delta \tau_{ij} = \frac{1}{f(x'(t))} \mathbf{1}(i, j) \in x'(t) \]
\[ 0 \text{, otherwise} \]

Pheromone Global Update Rule

- The same amount of pheromone is added to each edge in the best tour.
- By increasing the pheromone on the edges of the best tour, more ants would select these edges in the next iteration. (more exploitation or exploration?)
- large \( \rho_1 \):
  - previous experience is neglected in favor of more recent experience (best tour in the last iteration). (more exploration or exploitation?)
- \( \rho_1=0.1 \) in their experiments.

New Local Pheromone Update Rule

- During the construction of a tour, after an ant selected an edge, the pheromone on that edge is decreased using the rule:

\[ \tau_{ij}(t) = (1 - \rho_2) \tau_{ij}(t) + \rho_2 \tau_0 \]
\[ \tau_0 = (n_G L)^{-1} \]

- \( n_G \) is the number of nodes (sites) in graph \( G \).
- \( L \) is a very rough approximation of the optimal tour length.
- \( \rho_2=0.1 \) in their experiments
- Number of ants is 10

Local Pheromone Update

- The rule reduces the pheromone of an edge after an ant has visited it, making it less attractive being selected by other ants during the same generation.
- Consequently, other ants will more likely to explore other edges in the remaining steps of an iteration and not to converge to the same path.
- When ants in the same iteration explore different paths, there is a higher probability that one of them will find an improved solution rather than converging to the same path.
- Local pheromone update increase the diversity of ant solutions. (more exploration or exploitation?)
New Candidate List

- There is a preferred candidate list containing \( c_l \) number of closest cities to city \( i \);
- When selecting next city to visit after \( i \), the candidate list is checked first and the closest city is selected.
- When the candidate list is empty, one city is selected from the rest of the un-visited cities using the new transition rule.
- Biased toward the cities that are closest to the current city. (more exploration or exploitation?)

The ACS Algorithm

```
Initialize
Loop /\: at this level each loop is called an iteration /\:
  Each ant is positioned on a starting node
Loop /\: at this level each loop is called a step /\:
  Each ant applies a state transition rule to incrementally build a solution
  and a local pheromone updating rule
  Until all ants have a complete solution
A global pheromone updating rule is applied
until end condition
```

Fig. 3. The ACS algorithm.

50 – 100 Cities Results

- ACS-TSP has been shown to be superior over other methods like GA, SA, EP.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>ACS</th>
<th>SA</th>
<th>EP</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 cities</td>
<td>429</td>
<td>429</td>
<td>429</td>
<td>429</td>
</tr>
<tr>
<td>100 cities</td>
<td>(427.99)</td>
<td>(427.99)</td>
<td>(427.99)</td>
<td>(427.99)</td>
</tr>
<tr>
<td>150 cities</td>
<td>535</td>
<td>545</td>
<td>592</td>
<td>592</td>
</tr>
<tr>
<td>200 cities</td>
<td>(600.57)</td>
<td>(600.57)</td>
<td>(600.57)</td>
<td>(600.57)</td>
</tr>
<tr>
<td>250 cities</td>
<td>21,282</td>
<td>21,781</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>300 cities</td>
<td>(21,282.44)</td>
<td>(21,282.44)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>350 cities</td>
<td>(21,282.44)</td>
<td>(21,282.44)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

For some bigger problems

- Candidate list: a preferred candidate list containing \( c_l \) number of closest cities to city \( i \);
- When selecting next city to visit after \( i \), the candidate list is checked first and the closest city is selected.
- When the candidate list is empty, one city is selected from the rest of the un-visited cities using the new transition rule.
- \( c_l \) is 15 in their experiments.
ACO + Candidate List Results

ACO performance for some common genetic problems (over 14 nodes). We report the integer length of the shortest tour found, the number of tours required to find it, the average integer length, the standard deviation, the optimal values (the 100% guarantee), in square brackets, the average integer and average results, given that the optimal solution is not known, and the relative error of ACO.

<table>
<thead>
<tr>
<th>Problem name</th>
<th>ACO total integer length</th>
<th>ACO number of tours generated</th>
<th>ACO average integer length</th>
<th>Standard deviation</th>
<th>Optimum (1)</th>
<th>Optimum (2)</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0198</td>
<td>15,968</td>
<td>585,000</td>
<td>15,284</td>
<td>71</td>
<td>15,780</td>
<td>0.88 %</td>
<td></td>
</tr>
<tr>
<td>(100-city problem)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0198</td>
<td>51,260</td>
<td>595,000</td>
<td>51,090</td>
<td>109</td>
<td>50,779</td>
<td>0.90 %</td>
<td></td>
</tr>
<tr>
<td>(150-city problem)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0052</td>
<td>29,147</td>
<td>630,000</td>
<td>28,023</td>
<td>275</td>
<td>27,688</td>
<td>1.67 %</td>
<td></td>
</tr>
<tr>
<td>(500-city problem)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0470</td>
<td>9,015</td>
<td>891,276</td>
<td>9,088</td>
<td>28</td>
<td>8,806</td>
<td>2.57 %</td>
<td></td>
</tr>
<tr>
<td>(700-city problem)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1057</td>
<td>29,977</td>
<td>843,500</td>
<td>29,763</td>
<td>116</td>
<td>22,914</td>
<td>0.99 %</td>
<td></td>
</tr>
<tr>
<td>(1500-city problem)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TSP Discussion

- Greedy approaches (more exploitation) seem to work better.
  - Select closest city first
  - Select the edge with the most pheromone and the shortest length 90% of the time
  - Reward pheromone to edges that belong to the shortest tour.
- Would this approach work for other problems?