Lab 1: Trail Laying/Following & Ant Colony Optimization

This laboratory requires the following equipment:

- C/C++ programming tools
- Matlab

The laboratory duration is three hours. The laboratory is not graded, however, we encourage you to take notes during the course of this laboratory to help preparing for the exams.

A solution sketch will be posted after the lab session.

1.1 Office hours

Additional assistance outside the lab period (office hours) can be requested on appointment using the dis-ta@groupes.epfl.ch mailing list.

1.2 Information

In the following text you will find several exercises and questions.

- The notation \( S_1 \) means that the question can be solved using only additional simulation.
- The notation \( Q_1 \) means that the question can be answered theoretically, without any simulation.
- The notation \( I_1 \) means that the problem has to be solved by implementing a piece of code and performing a simulation.
- The notation \( B_1 \) means that the question is optional and should be answered if you have enough time at your disposal.

To prepare you for the exams and to permit you better time planning during the exercise session, we show an indicative number of points for each exercise in parentheses. The combined total number of points for each laboratory exercise is 100.

1.3 Self-organization

This lab is intended to be a simple introduction to the topic of self-organized systems. Recall that the key ingredients observed in natural self-organization are:

- Positive feedback (e.g., recruitment mechanisms)
- Negative feedback (e.g., exhaustion, saturation, competition)
- Randomness (e.g., initial fluctuations)
- Multiple interactions (e.g., interactions with the environment and other individuals)

In this lab you will conduct a set of simulations to investigate several models of foraging strategies and an ACO algorithm, the AS discussed in class. These simulations are meant to illustrate the use of swarms in creating practical solutions to difficult problems. While we will not examine particularly complex systems, the generalization to these problems is straightforward, and there will be opportunities for inquisitive students to investigate further.
2 Working Environment:

The laboratory sessions will use Ubuntu Linux. If your machine is running Windows, reboot it and select Ubuntu from the boot menu.

Log in with your GASPAR username and password. Your home directory will be mounted at /home/username/myfiles. This is a private storage folder which you can access from anywhere (see http://enacit.epfl.ch/stockage/etudiant.shtml for information on how to use it outside of the lab). **Always store your work in this folder.** Anything stored in just /home/username will be deleted on log-out.

3 Lab Part 1: Antsim

During this lab you will become a virtual ant expert. You will be given the following four species of ants to examine, and you are to explain their performance in different environments.

1) **Justant:** The basic ant. It lays pheromone wherever it travels until its supply runs out. The pheromone evaporates rather quickly. Justant prefers to move toward regions of higher pheromone concentration, but it also has a tendency to wander randomly.

2) **Sneezy:** Similar to the Justant, except it suffers from an inability to perceive any pheromone trails.

3) **Greedy:** Similar to the Justant, except it must always move toward an area of higher pheromone concentration.

4) **Diffrant:** Most different from Justant. It lays pheromone much more slowly, and its pheromone evaporates more slowly. Also, it has a lower tendency to wander and a smaller pheromone supply.

Antsim simulates the behavior of each species on different environment maps. In each map there is a nest, from which the ants emerge slowly throughout the simulation. The goal of each ant is to find a source of food and return a bit of it to the nest. Antsim will return the average amount of food returned over multiple simulation runs, for a given environment-species pair.

3.1 Getting started with Antsim

First, download and expand (tar xzvf lab01.tar.gz) the file lab01.tar.gz provided on Moodle. Use the simulator by changing into the part1 directory and entering:

```
cd lab01/part1
make
./antsim [environment] [species] [runs] > simul.m
```

- `environment = {1,2,3,4}` specifies the map in which the ants will roam.
- `species = {1,2,3,4}` specifies the species to simulate.
- `runs` specifies the number of simulation runs to perform. Use 1 to get a feeling for what is happening qualitatively, and then increase to 10 or 20 to get a better estimate of food return rate.
Go through antsim.c to get a feeling about how the simulator and the parameters defining each species work. This will make it easier to answer the questions below. Also, feel free to change the .map files to test hypothesis about species behavior.

To visualize the result of the previous simulation you have to start Matlab. Once Matlab is started execute the script created by antsim to load the data of the simulation (Do not forget to change your “Working Directory”):

```
>> simul
Environment: 1
Species: 1
Food picked up (avg over 100 runs): 683.3
Food returned to the nest (avg over 100 runs): 292.3
```

It gives you the average performance of the ant species you selected. You can then either display an animation of the simulation to get a feeling on how the ants interact with the environment (for runs = 1):

```
>> antsim_play
```

or a step by step animation with (for runs = 1):

```
>> antsim_stepbystep
```

or display a histogram of the results (for runs > 1):

```
>> antsim_histogram
```

![Histogram of returned food (100 runs)](image)
3.2 Two Equal Paths (Symmetric Bridge)

Environment 1 is as follows. N is the nest and F is the food. The **black** cells are paths open to the ants, and the **white** cells are obstacles. An ant can move to any of the 8 cells surrounding its current location (except for obstacles).

In each step, ants decide to either wander or follow the pheromone. A wandering ant moves randomly to any open cell (including the one it came from). An ant following the pheromone favors moving to cells with a high pheromone concentration and never goes back to the cell it just came from.

*S1*: Simulate the different species on this environment.

**Q2(2)**: Rank, from best to worst, the fitness of each species in this environment (i.e., by highest average food return rate). Ties are possible.

**Q3 (3)**: Explain the strategy of the fittest species.

**Q4 (5)**: Explain the lower performance of each of the other species.
3.3 Two Unequal Paths (Asymmetric Bridge)

Environment 2 is as follows:

![Environment Diagram]

**S5:** Simulate the different species on this environment.

**Q6 (2):** Rank, from best to worst, the fitness of each species in this environment.

**Q7 (3):** Explain the strategy of the fittest species with respect to the particularity of this environment.

**Q8 (5):** Explain the lower performance of each of the other species.
3.4 Multiple Paths

Environment 3 is as follows:

![Environment Diagram]

**S9**: Simulate the different species on this environment.

**Q10 (2)**: Rank, from best to worst, the fitness of each species in this environment.

**Q11 (3)**: Explain the strategy of the fittest species with respect to the particularity of this environment.

**Q12 (5)**: Explain the lower performance of each of the other species.
3.5 Open Terrain with Bridge

Environment 4 is as follows:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>

S13: Simulate the different species on this environment.

Q14(5): You probably observed that all species except Sneezy got a very bad performance. Why are these species not successful? What problems are specific to this type of environment?

Q15(3): How might these difficulties be avoided?

Q16(2): Why does Sneezy still achieve acceptable performance?

Q17(5): In many types of complex search systems there is a delicate balance between exploration and exploitation. Briefly explain the contribution of each of the species parameters (as defined in species.c) toward these conflicting goals.

I18(15): The provided code has facility for the use of more than one type of pheromone (as defined by PHER_KINDS in environment.h). However, antsim.c only makes use of one pheromone. How might you use a second pheromone so that ants other than Sneezy can perform well in this environment? Modify PHER_KINDS in environment.h (one line), the species parameters in species.c (two lines) and the function ant_strategy in antsim.c (three lines) as necessary. Run the modified code and observe the results. Try to understand and explain how the different types of pheromones are deployed and how this improves the performance of ants.

4 Lab Part 2: Ant Colony Optimization and the Traveling Salesman Problem

Ant Colony Optimization (ACO) algorithms are meta-heuristic approaches that allow solving a suite of hard optimization problems by using the ant colony/trail laying metaphor [Dorigo2004]. They
are inspired by the optimization capabilities of foraging ants as it can be observed in the bridge experiments of J.L. Deneubourg.

The Traveling Salesman Problem (TSP) is a classical optimization problem. It deals with finding the shortest path that connect a number of cities, and passes every city once and only once. Due to its immediate connection to the shortest path problem that ants face during foraging, ACO approaches have been first tested on a TSP problem.

In this part of the lab you will familiarize yourself with different ACO algorithms to solve the TSP, and observe and evaluate their performance experimentally.

4.1 Basic Ant System

The first ACO algorithm, Ant System (AS), was presented in Marco Dorigo’s PhD thesis in 1992. In AS, \( m \) ants travel from a random starting point from city to city until all \( n \) cities have been visited. Hereby, paths are chosen randomly with a probability which is a function of the amount of pheromones already deposited and of the distance to the next city. The ants are capable of memorizing the visited nodes. The probability for an ant \( k \) at city \( i \) to go to city \( j \), is then given by

\[
p_{ij}^k = \frac{Q_{ij}}{\sum_{l \in F_i^k} Q_{il}} \quad \text{with } j \in F_i^k
\]  

(1)

where \( F_i^k \) is the feasible neighborhood for ant \( k \) at city \( i \) and \( Q_{ij} \) a combined metric of the quality of the route. In AS, the quality of a route \( ij \) is a function of the pheromones already deposited by other ants given by \( \tau_{ij} \), and a heuristic \( \eta_{ij} = 1/d_{ij} \), with \( d_{ij} \) the distance between city \( i \) and city \( j \). Thus, \( Q_{ij} \) is defined as

\[
Q_{ij} = [\tau_{ij}]^\alpha [\eta_{ij}]^\beta
\]

(2)

with \( \alpha \) and \( \beta \) allowing us to fine tune the impact of pheromones and heuristic information on the metric.

Figure 1: A simple instance of a TSP. Edges are marked with pheromones deposited by ants in previous iterations of the AS algorithm.
After completion of a tour, i.e. arriving at the starting city, ants assess the tour, and deposit an amount of pheromones that is inversely proportional to the tour length on every link \( ij \) they visited, i.e.  \( \Delta \tau_{ij}^k = \frac{Q}{L^k} \) with \( L^k \) the total length of the tour, and \( Q \) a parameter adjusted by heuristic. Here we simply set \( Q=1 \).

**Q10(3):** Consider the TSP in Figure 1. Edges are marked with pheromones (the numbers next to the edges) that have been deposited during previous tours of an ant, which is now sitting at the dark grey vertex. What will be the route that is most likely taken by the ant (assume heuristic information not to be available/constant)? Your solution should be a sequence of integers.

**Q20(3):** Again consider the TSP in Figure 1. Ignore now the pheromone values, and let the ant decide solely based on heuristic information. What will be the route that is most likely taken by the ant in this case? Keep in mind that the heuristic information associated with an edge is inversely proportional to the Euclidean distance \( d_{ij} \) between two vertices \( i \) and \( j \), given by the following matrix \( D = (d_{ij})_{i=0,...,4; j=0,...,4} \):

\[
\begin{pmatrix}
0 & 4 & 7 & 10 & 5 \\
4 & 0 & 3 & 7 & 8 \\
7 & 3 & 0 & 5 & 9 \\
10 & 7 & 5 & 0 & 6 \\
5 & 8 & 9 & 6 & 0
\end{pmatrix}
\]

Your solution should be a sequence of integers.

### 4.2 Elitist Ant System (EAS)

EAS is an updated version of AS that give the best route extra pheromone reinforcement.

**S21:** Change the working directory in Matlab to part 2 and type in the command window:

```matlab
cities=randommap(6)
tsp('random',cities)
```

Which creates a matrix with 6 cities at random locations (2D coordinates), and solves the TSP by creating a random tour. You can also type `load eil51`, which will load a `cities` structure with 51 cities whose tour length minimal value is known to be 426 [TSPLIB].

**Q22(10):** Look in the function `solvetsp_guess`, which is found in `tsp.m`. Explain which ant strategy lies in this code. Test your solution by running

```matlab
tsp('guess',cities)
```

for different scenarios created using `randommap()` for a small number of cities, and qualitatively verify your idea using the `eil51` map.

**I23(10):** The function `solvetsp_ant` provides a skeleton of an Elitist Ant System (EAS). In EAS the best ant of a tour is allowed to deploy pheromone. Implement the missing two lines that
allow an ant to calculate the probability for choosing the next city according to the Equation 1. Test your solution by running

\[
tsp('ant', cities)
\]

\textbf{S24:} You can now compare the different approaches (random, guess, and EAS) using the command \texttt{testall(cities)}.

\textbf{Q25(4):} Observe the number of tours that EAS needs for finding the best solution (you can increase the number of tours using the parameter \texttt{ntours}). What happens? Which feature of the algorithm would improve its global performance?

\textbf{Q26(4):} The TSPs considered so far were fully connected, i.e. every city could be reached from every other city, leading to a quadratic distance matrix. How would you need to modify this matrix in order for taking into account cities that are not directly connected?

\subsection*{4.3 Local Search}

Whereas constructive algorithms like the EAS and its derivatives are constructing solutions in an incremental way using some heuristic rules, Local Search algorithms (LS) optimize a given solution by iteratively exploring neighborhoods of the solutions that are generated by local changes.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{tsp_greedy_local_search.png}
\caption{A simple TSP problem solved by a greedy algorithm (left) and optimized by a local search (right).}
\end{figure}

A neighborhood of a solution consists hereby of all possible solutions that can be achieved by \(k\) permutations, for instance by exchanging two edges as depicted in Figure 2.

\textbf{Q27(4):} Write both tours (starting point is ‘a’ for both and the initial direction the same) of Figure 2 as a string/vector of characters and compare them. What happens to the order of a substring when you swap two edges?

\textbf{S28:} A 2-opt local search is implemented in the function \texttt{local_search_2opt}. It systematically evaluates all permutations of a solution that can be achieved by exchanging two edges. Test the function with \texttt{testall(cities,1)}, the second parameter enables local search in all of the above algorithms.

\section{References}

[TSPLIB] \url{http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/}. 