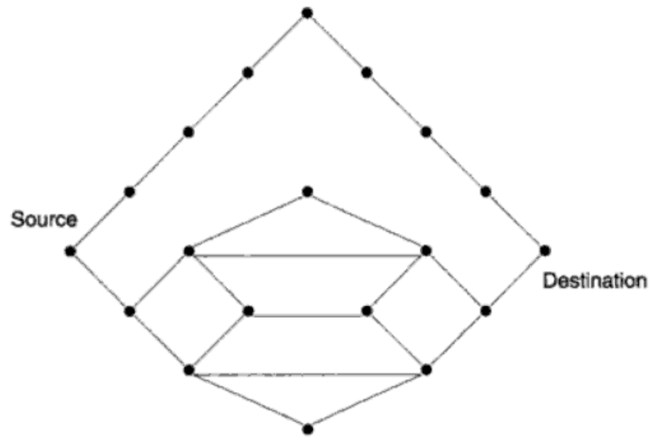
NAT-DL: Self-study questions (and answers) on ACO etc.   Set 4 (week 6)

1. The figure on the right shows an example from the ACO book by Dorigo and Stuetzle. What results do you expect for an ant colony algorithm that does not use taboo lists (except for inhibition of immediate return to the previous node)?

   Answer: Assume the visibility is given by the inverse of the lengths of the links in the picture. The densely connected path in the lower part of the graph will most likely lead to cycles. Cycles can be discouraged by a high evaporation rate, but then the algorithm will not generate a memory of good solutions. So without a taboo list, cycles seem to be unavoidable. Some ants will travel the upper part which leads them straight to the goal such that in most of the runs the algorithm will end up with the suboptimal solution along the upper part (8 vs. 5 steps in the lower part). Because the problem is small, there is still some chance to find the optimal solution. Conclusion: Taboo lists are useful.

2. Discuss the application of ACO to the eight-queens puzzle. This puzzle is the problem of putting eight chess queens on an 8x8 chessboard such that none of them are able to capture any other using the standard chess queen's moves, cf. en.wikipedia.org/wiki/Eight_queens_puzzle.

   Answer: (s. S. Khan et al.“Solution of n-Queen problem using ACO”. In proc. of 13th IEEE International Multi-topic Conference (INMIC 2009), Islamabad, Pakistan.)
   Each ant simply place queen by queen on the chessboard, which can be done in a search space of n times *nxn* nodes. Think of *nxn* rows and n columns: each ant runs from left to right (i.e. n steps including the initial placement) each time selecting one of the nxn fields of the chessboard). Local heuristic is whether a queen can “kill” any one of the queens that are already there for this ant. The paper also studies alpha and beta and finds alpha slightly >1 and beta about 1.5 to be good values.

3. **[Numerical exercise]** Run the standard ACO on the travelling salesperson problem with N cities. You may use code from http://www.aco-metaheuristic.org/aco-code/ or elsewhere or partially reuse your code from the 1st assignment. Start with n_{ants}=N, alpha=1, beta=2, rho=0.75. How can you influence the quality of the stationary solution. Consider the standard deviation of the tour length over the ants during one iteration.

   Answer: Strictly speaking, the quality of the stationary solution can only be influenced by the choice (or changes) of the parameters before the solution becomes stationary. However, theoretical studies often consider quasi-stationary solutions: Run the algorithm for some parameters and wait until the characteristics (effective decree, mean, and variance of the solutions) do not change any more, then change parameters. In physics, this is called adiabatic approximation, it allows us to tell whether the parameters are responsible for a certain change or whether it had happened anyway).
   Smaller values of alpha and beta should help, as well as higher evaporation, i.e. larger rho. Generally a low standard deviation is a sign of (premature) convergence, while high standard deviation is not a problem if you make sure to keep the overall-best ant (and to reinsert it should the mean tend to increase or to use some other form of elitism).

4. Consider the following (very small) TSP:
   $d(A,B) = 2$, $d(A,C) = 3$, $d(A,D) = 5$, $d(B,C) = 3$, $d(B,D) = 3$, $d(C,D) = 4$.
   a. How many different tours are possible? What are the lengths of these tours?
   b. Which tour is most likely to be found by the ants?
   c. Compare the case of a single ant to a population of two or more ants on this problem.

   Answer:
   a) (4 choices for first city) x (3 for 2nd) x (2 for third) = 4! = 24. Circular permutations are the same so divide by 4. In addition, it is to be checked whether tours can be reversed, i.e. whether ABCDA is the same as ADCBA (etc.): The ants may choose the two tours differently based on the local heuristics, so we need to consider 6 tours, i.e. N!/N = (N-1)! (assuming you go back to the starting city).
   Two tours have lengths of 12 (ABDCA, ACDBA), the other four have all length 14. So, avoiding the longest link (AD) specifies already the optimum.
   b) The ants will initially prefer to go from A to B. The next decision (whether to go from B to C or D) is equally likely, but the tour the continues from B to C will receive less pheromone.
   c) We essentially have three unique tours, after removing circular permutations, and reverse orderings, two of which are equivalent. In this problem, we aim to avoid the longer tours.
   A single ant will make its first decision, and it is more likely that it ends up in one ot the longer tours, and in the single shorter tour. It will put some pheromone, and will then be even more likely to continue on the longer tour and may get stuck there (if there is not enough further exploration). For two ants, it is more likely that at least one ant finds the short tour, and will guide by its pheromone also the other ant to the better tour, so more ants are less likely to get trrapped. If there is deception (a short first leg), then it is even more likely to get trapped. We cannot hope to have no deception in problems to which we apply ACO (in the same way one cannot hope to have no local optima in the other algorithms that we have considered), but the collective "intelligence" of the ants is able to mitigate the effects of decptions or multiple paths.

5. Consider **one** of the following problems (or any other one that seems to be interesting) and explain how you would use ant colony optimisation to find an acceptable solution: Sequential ordering, classification (e.g. of images), graph colouring, the knapsack problem (or the cutting stock problem), protein folding, the shortest common supersequence problem (for details cf. wikipedia). For this purpose, Dorigo has suggested answering the following questions:
   a) Define a set of candidate solutions and the set of feasible solutions.
   b) Define a greedy construction heuristic:
      i) What are the solution components?
      ii) How do you measure the objective function contribution of addition a solution components
      iii) Is it always possible to construct feasible solutions?
      iv) How many different solutions can be generated with the constructive heuristic?
   c) Define a local search algorithm:
      i) How can local changes be defined?
      ii) How many solution components are involved in each local search step?
      iii) How do you choose which neighbouring solution to move to?
      iv) Does the local search always maintain feasibility of solutions?

   Answer: Analogous to the AntTSP and AntBin in the lecture. The goal of this exercise is twofold: Get an idea to what type of problems ACO is applicable and recall the main steps of the algorithm.

6. Recall the main algorithms that we have studied (i.e. GA, GP, ES, ACO, PSO, DE) and classify them according to Dorigo's criteria for the classification of metaheuristic optimisation algorithms. You may like to represent your answer to this question as a table.
   a. Is the solution obtained by direct construction or by the use of local search?
   b. Are population of solutions used or not?
   c. Is a memory used within the search process or not?
   d. Is the evaluation function fixed or is it modified during search?
   e. Several neighbourhoods or only a single one (i.e. what topology)?
   f. Inspired by biology, physics, or otherwise?
   g. For what type of problems the algorithm can be expected to work well?

   Can you think of new variants of the algorithms, by modification of any criteria or by recombination?

   Answer: See the table on the next page. Obviously, not all variants covered here, and in many of the entries further discussion will be useful.

   For variants, we can consider for example

   - DE as a mixture of GA and PSO.
   - Using a population in SA.
   - In ACO, we could use different pheromone trails for different groups of ants (it could be seen as a population of trains, or as well as an island variant of ACO).
   - Velocity vectors (from PSO) in ES (similar to CMA)
   - Preferred mutations (based on pheromones as in ACO) in GA or GP.
   - PSO with mutations, i.e. some vector components are exchanged (similar to DE)

   Many other combinations and variants are possible, so which one to choose? The following answer are possible (to be discussed later): Choose based on (i) practical experiences with a given algorithm on a certain problem, (ii) background knowledge on the problem, or (iii) let a hyperheuristic algorithm make the choice.

Criterion/ algorithm	GA	ES	GP	ACO	PSO	DE
Solution	Assumed to be constructed from building blocks	Direct construction	Direct construction or by using autonomously defined subroutines	Probabilistic composition from partial solutions	Direct construction	Similar to PSO
Local Search	In order to ensure admissibility; hill-climbing	Possible (hill-climbing)	Editing for syntactical correctness or based on a grammar; hill-climbing to fix numerical constants	In order to ensure admissibility	Often not used	Similar to PSO
Population	Large population	Often population of subpopulations	Large population	Not really, as there is a single pheromone matrix	Small number of particles in the swarm	Similar to PSO
Memory beyond current solutions	Only when elitism is used	Mutabilities	Only when elitism is used	Taboo lists, pheromone matrix	Personal best and generation best	Similar to GA(!) (Here the diversity of the population is explicitly used)
Evaluation	Fitness function	Fitness function (objective function)	Fitness cases (with cross-validation) +bloat control	Tour length used for pheromone update	Fitness function (objective function)	Similar to PSO
Neighbourhood	Restricted in the island variant; many neighbourhoods in the multi-objective variant	Neighbourhood structure is learned by correlations of mutabilities	Usually all-to-all competition; islands possible	Ants exchange information via the common environment (stigmergy)	Often on graphs, i.e. neighbourhood chosen in dependence on problem	Usually all-to-all, but similar to PSO variants are possible
Inspired by nature	Natural evolution	Directed search, not too similar to natural evolution	Epigenetic processes (or perhaps the social group of programmers?)	Possibly by real ants, or by Bayesian theory	Swarm intelligence in bird flocks or 3-body problem	Similar to PSO
Applicability	General discrete search problems of medium dimensions	Problem where neighbourhood-based search seems promising, i.e. few local minima	Problems where short programs are expected as solutions, or where reasonable solutions exist that can be further developed	Sequential, nearly separable problems for which already a good local heuristics exists	Continuous search problems of medium dimensions	Similar to PSO, also often used in hybrid algorithms