Exercise 1

1) Rewrite the ACO algorithm for solving TSP seen in the course.
2) Evaluate the complexity of this algorithm.
3) Run the two first iterations for an instance of n=4 cities A, B, C, D such as :
   AB=6 AC=5 AD=8 BC=4 BD=4 CD=5
   With each of the following parameter configuration:
   a) Q=20, α=1, β=1 , τ0=1 ;
   b) Q=20, α=1, β=5 , τ0=1 ;
   c) Q=20, α=5, β=1 , τ0=1 ;
   Take ρ = 0.01 , ρ = 0.01 puis ρ = 0.1. What can we conclude?

4) Modify the algorithm for the versions:
   a) Elitist with e = n/10 ;
   b) MMAS for taumin= 0.1 and taumax=1 ;

Exercise 2

Define a distance function in order to design ACO algorithm for solving each of the following problems:

a) Simple KSP
b) SAT-3 ;
c) 8 Queens ;
d) Graph coloring ;
e) $1 \mid\mid \sum_{j=1}^{n} T_j$;
f) Multidimensionnel KSP ;
g) Multiple KSP ;

Exercise 3.

ACO design for a permutation scheduling problem.

The single machine total weighted tardiness problem (SMTWTP) is a well-known NP-hard scheduling problem. Given n jobs to be scheduled on a single machine. Each job i is characterized by its weight w_i, a processing time p_i, and a due date d_i. For a given schedule, let C_i define the completion time of the job i. The objective of the SMTWTP problem is to minimize the total weighted tardiness:

$\sum_{i=1}^{n} w_i \times T_i$ where $T_i = \max(0, C_i - d_j)$ represents the tardiness for the job i.

The main problem-dependent elements in designing an ACO algorithm are the pheromone information and the solution construction. Propose a pheromone information that is suitable to the problem. How this pheromone information will be used in the solution construction? Notice that the relative position of the job is much more important than its direct predecessor or successor in the schedule such as in the traveling salesman problem. In addition to the pheromone, a heuristic to guide the ants toward good solutions must be proposed. Which priority greedy heuristic may be used to decide the next job to schedule?

Exercise 4.

ACO for the set covering problem.

The set covering problem (SCP) represents an important class of NP-hard combinatorial optimization problems with many applications in different domains: transportation network, integer coefficients linear programming, assignment problems, simplification of Boolean expressions, and so on. The SCP problem is generally defined by its Boolean covering matrix: $(a_{ij})_m \cdot n$ where $i \in I = 1,...,m$ and $j \in J = 1,...,n$. An element of the matrix $a_{ij} = 1$ if the row i is covered by the column j, and $a_{ij} = 0$ otherwise. Each column j of the matrix is characterized by a positive cost c_j and a cardinality ($card_j$), which is the number of rows covered by the column j. The SCP problem consists in choosing a subset of columns at a minimal cost in a way to cover all the rows (at least a 1 on each line). The SCP can be stated as minimizing

$\sum_{j=1}^{n} c_j x_j$ such as: $j = \begin{cases} 1 & \text{if the column j belong to the solution x} \\ 0 & otherwise \end{cases}$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m$$

$$x_j \in \{0, 1\}, j = 1, \ldots, n$$

The constraint maintains the integrity of the system, that is, each row is covered at least by one column. The parameter α_i is defined as the set of columns that cover the line i and β_j and as the set of lines covered by the column j: $\alpha_i = \{j \in J/a_{ij} = 1\}$ $\beta j = \{i \in I|aij = 1\}$

Propose a greedy algorithm to solve the SCP problem. Based on this greedy algorithm, design a solution construction strategy for the ACO algorithm. Once a covering has been established by an ant, an iterative function that eliminates the redundant columns is performed, that is, those that are useless (superfluous) since their elements are covered by other columns of the covering. This function iterates on the columns, beginning from the last one whose cost is more significant, to optimize the cost.

Exercise 5.

ACO for the shortest common super sequence (SCS) problem.

String problems are one of the most important and well-studied problems in computer science. Such an example of string problems is the SCS problem. Given an alphabet and a set L of strings over this alphabet, the SCS problem consists in finding a minimal length of super sequence of each string of L. A string S is a super sequence of a string A if S can be obtained from A by inserting in A zero or more characters. For instance, if the set L is {bbbaaa,bbaaab,cbaab,cbaaa}, the optimal super sequence, the one minimizing its length, will be cbbbaaab. The SCS problem has many applications in various domains:

- **Phylogenetics in bioinformatics:** The strings represent DNA from different species. One of the main questions concerning the evolution of those species is what will be their ancestor. For simplicity, let us assume that only mutations based on deletion of nucleotides can occur during the evolution process. Then, most probably the ancestor is the shortest super sequence of the input DNA strings.
- **Conveyer belt design:** Given a set of work pieces $\{W_1,W_2,...,W_n\}$ to be designed. To each work piece W_i is associated a sequence of operations for designing it, $W_i = s_1s_2 ...s_k$. The problem is to produce a conveyer belt on which the n work pieces can be designed. Such a conveyer belt can be defined as a super sequence. The cost of producing the conveyer belt is mainly related to the size of the string representing its sequence. For instance, let us have three work pieces $\{W_1,W_2,W_3\}$ and their associated sequences to produce them: W_1 = bcbacb, caac, bbacac. A conveyer belt on which the three work pieces are designed can be bbacbaacb. Figure 3.66 presents an example of a super sequence for the conveyer belt.Ant can construct solutions by iteratively removing characters from the front of the input strings of L, and appending them to the super sequence. Then, each ant

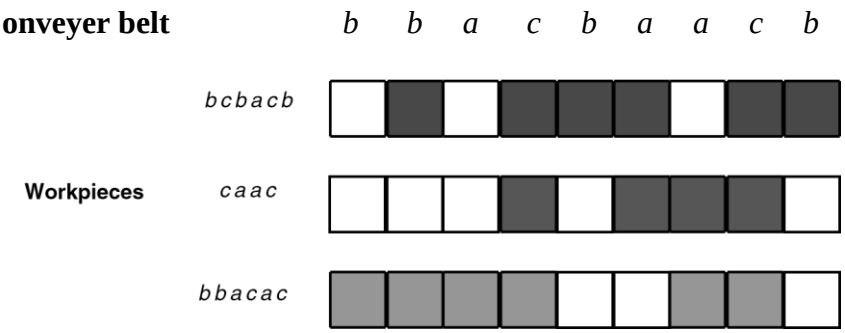

FIGURE 7.

Design of a conveyer belt to produce work pieces. A feasible non optimal solution is shown. The optimal solution has a size of 8 : bcbacacb must maintain a set of pointers to the current front of the strings that represent the selectable components of the constructed solution. The transitions are defined by the rules that select the next character of the super sequence. Only feasible super sequences are generated. Design an ACO algorithm for this problem.

REFERENCES

- E. Bonabeau, M. Dorigo, G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems, 1999
- M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem, 1997.
- M. Dorigo and T. Stützle. Ant Colony Optimization, MIT. Press, 2004.
- J. Kennedy and R. Eberhart. Swarm Intelligence, 2001.
- http://www.cnrs.fr/Cnrspresse/n386/html/n386a09.htm