Advanced algorithmic : Practical session #3

Meta-Heuristics

2 supervised practical session + 1 unsupervised session + 1 supervised session with Quiz

The exercises can be programmed in any language (either your favorite one or the one that best fits to the subject). The knowledge about how to use this language is considered as a pre-requisite.

⚠There will be 3 supervised sessions with 1 unsupervised session. During the last supervised session you will have to fill in a questionnaire.

Exercise 1 : find a binary chain that solves a UBQP

A UBQP (Unconstrained Binary Quadratic Problem) can be described under the form :

$$\min f(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij}.X[i]X[j]$$

where Q is $n \times n$ constant matrix and X is a n-tuple of binary variables. For instance, we can consider the following matrix Q :

$$Q = \begin{pmatrix} -17 & 10 & 10 & 10 & 0 & 20 \\ 10 & -18 & 10 & 10 & 10 & 20 \\ 10 & 10 & -29 & 10 & 20 & 20 \\ 10 & 10 & 10 & -19 & 10 & 10 \\ 0 & 10 & 20 & 10 & -17 & 10 \\ 20 & 20 & 20 & 10 & 10 & -28 \end{pmatrix}$$

For the solution $X = \boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{0}$, $f(X)$ equals 6.

UBQP is notable for its ability to represent a significant variety of important problems [?]. The applicability of this representation has been reported in diverse settings covering a range from financial analysis to combinatorial optimization problems pertaining to graphs, as well as machine scheduling, cellular radio channel allocation etc.

We would like to find a solution X that minimizes the function $f(X)$ for a given matrix Q described in a file. For this example, the file `partition6.txt` contains first the size n (6) then a number p (2) (we will explain it's meaning later) then 6×6 elements separated with blank spaces. The file`graph12345.txt` is another example.

In order to answer the first questions you can create an array Q containing the values written above. The automatic reading of the data from a text file should be encoded before the Quiz.

1. Write a function that returns an initial solution randomly choosen for this problem, i.e., returns tuple of n bits randomly choosen (the size of the tuple should be a parameter that is instantiated with the value read in the input file).
2. Write a function that evaluates this solution with respect to f where Q is given in the input file.
3. Program a function `best_neighbor` that returns the best neighboring solution of X where a neighbor X' of X is a sequence of bits that differs from X only by one bit.

We recall the `Steepest Hill-Climbing` Algorithm :

```
• start from a solution $s$
• $nb\_moves \leftarrow 0$;  $STOP \leftarrow false$
Repeat
    • $s' \leftarrow$ best_neighbor($s$)
    • if better($f(s'), f(s)$)  then $s \leftarrow s'$
                                else $STOP \leftarrow true$        /* local optimum */
    • $nb\_moves$++
Until $nb\_moves =$ MAX_moves or $STOP$
Return($s$)
```

4. Program the `Steepest Hill-Climbing` Algorithm
5. Program a variant with restart, i.e. do an external loop around the `Steepest Hill-Climbing` that allows you to start from a new random solution, then do an attempt : i.e., `MAX_moves` moves towards best neighbors, then restart again with a new random solution. This external loop should be executed `MAX_attempts` times.
6. Run your programs on the two files `partition6.txt` and `graph12345.txt`.

   We recall the Tabu Method :

```
• start from a solution $s$
• Tabou $\leftarrow$ []
• $nb\_moves \leftarrow 0$;  $bsol \leftarrow s$;  $STOP \leftarrow false$
Repeat
    • if non_Tabu_neighbor($s$) $\neq \emptyset$
   then $s' \leftarrow$ best_non_Tabu_neighbor($s$)
   else $STOP \leftarrow true$                    /* nomore non Tabu neighbor*/
    • Tabu $\leftarrow$ Tabu + $\{s\}$
    • if better($s'$,$bsol$) then $bsol \leftarrow s'$ /*best current solution*/
    • $s \leftarrow s'$
    • $nb\_moves$++
Until $nb\_moves =$ MAX_moves or $STOP$
Return($bsol$)
```

   where the Tabu list is implemented in FIFO of fixed size k.

7. Program this tabu method and try different sizes for the tabu list.
8. We now have some constraints on the solutions, we are searching for a binary sequence such that its sum is greater than p. Modify your best_neighbor function in order to take this constraint into account where p is the second number given in the file describing the problem (for `partition6.txt` this number is 2, for `graph12345.txt` it is 4). Test the Steepest-Hill Climbing with Restart program with this new neighboring function.

Exercise 2 : Travelling salesman

The famous travelling salesman problem (TSP) asks the following question : “Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the starting city ?”

We dispose of a file describing a list of n cities (their number `n` is given in the beginning of the file) with their identifiers and their coordinates under the form :
`Id, x, y` where `Id` is the number of the city, and (`x`,`y`) are its coordinates. the cities are given in the order of their `Id` from 1 to n (hence `Id` is not useful and can be deduced from the number of the line).

A solution of this problem is a sequence of n cities, it represents a route starting from point (0,0) visiting each of the cities of the sequence then returning to point (0,0). For sake of simplicity, the length of the path between two cities is the Euclidian distance between them : $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

For instance, using the file `tsp5.txt` describing the coordinates of 5 cities, the solution

5	3	4	1	2

corresponds to a route of length 263.88 km.

The problem is to find a solution with the shortest total length.

1. Write a function that returns an initial solution randomly chosen, i.e., a tuple of n cities randomly chosen (the size of the tuple should be a parameter which value is given in the input file). For this function you can either use a constructive method (create a chained list of cities and choose randomly the index (in $[1..n]$) of the city to remove from the list, then choose an index in $[1..(n-1)]$ remove the second city, etc.) or an iterative method (from a solution X where the cities are given in increasing order from 1 to n and for $i = 1$ to n replace $X[i]$ by $X[random(n)]$).
2. Write a function that evaluates this solution, i.e., the length of the route of the travelling salesman starting from (0,0) through all the cities $X[1] \dots X[n]$ and coming back to (0,0).
3. Program a `best_neighbor` function that returns the best neighboring solution of X where a neighbor X' of X is the sequence obtained by swapping two cities in the sequence X.
4. Use the Steepest Hill-Climbing algorithm and its variant with restart limited to MAX_attempts times on the files `tsp5.txt` and `tsp101.txt`.
5. Use the tabu method and try different sizes for the tabu list and for `MAX_moves`.

Work to do during unsupervised sessions

Before the last supervised session, you should work on the programs done for excercise 1 and 2. The questionnaire will be distributed at the last supervised session and you will have two hours to finish to encode the exercises in order to answer questions about them.