The Far From Most String Problem (FFMSP). The FFMSP belongs to the family of string selection and comparison problems known as *sequence consensus problems*, where a finite set of sequences is given and one is interested in finding their consensus, that is, a new sequence that represents as much as possible all the given sequences. Among the consensus problems, the FFMSP is computationally one of the hardest ones with applications in several fields, including molecular biology where one is interested in creating diagnostic probes for bacterial infections or in discovering potential drug targets.

In order to be able to formally state the FFMSP, the following notation is introduced.

- An alphabet $\Sigma = \{c_1, \ldots, c_k\}$ is a finite set of k different characters.
- $s^i = (s^i_1, s^i_2, \ldots, s^i_m)$ denotes a sequence of m characters (that is, of length m) over alphabet Σ.
- Given two sequences s^i and s^l over Σ such that $|s^i| = |s^l|$, the (generalized) Hamming distance $d_H(s^i, s^l)$ between s^i and s^l is calculated as follows:

$$d_H(s^i, s^l) := \sum_{j=1}^{|s^i|} \phi(s^i_j, s^l_j) \ , \tag{1}$$

  where s^i_j and s^l_j denote the character at position j in sequence s^i and in sequence s^l, respectively, and $\phi : \Sigma \times \Sigma \mapsto \{0,1\}$ is a predicate function such that $\phi(a,b) = 0$, if $a = b$, and $\phi(a,b) = 1$, otherwise.
- Given a set $\Omega = \{s^1, \ldots, s^n\}$ of n sequences of length m over Σ, d^Ω_H denotes the minimum Hamming distance between all sequences in Ω. Formally,

$$d^\Omega_H := \text{mín}\left\{ d_H(s^i, s^l) \mid i, l \in \{1, \ldots, n\}, i < l \right\} \ . \tag{2}$$

  Note that $0 \leq d^\Omega_H \leq m$.
- Each sequence s of length m over Σ is a valid solution to the FFMSP. Given any valid solution s and a threshold value $0 \leq t \leq m$, set $P^s \subseteq \Omega$ is defined as follows:

$$P^s := \{s^i \in \Omega \mid d_H(s^i, s) \geq t\}. \tag{3}$$

Given a fixed threshold value t (where $0 \leq t \leq m$), a finite alphabet Σ of size k, and a set $\Omega = \{s^1, \ldots, s^n\}$ of n sequences of length m over Σ, the goal of the FFMSP consists in finding a sequence s^* of length m over Σ such that P^{s^*} is of maximal size. In other words, given a solution s, the objective function $f : \Sigma^m \mapsto \{1, \ldots, n\}$ to maximize is defined as follows:

$$f(s) := |P^s| \tag{4}$$

The standard integer linear programming (ILP) model for this problem works on two sets of binary variables. The first set contains for each position i ($i = 1, \ldots, m$) of a possible solution and for each character $c_j \in \Sigma$ ($j = 1, \ldots, k$) a binary variable $x_{i,c_j} \in \{0,1\}$. The second set consists of a binary variable $y_r \in \{0,1\}$ ($r = 1, \ldots, n$) for each of the n input sequences provided in set Ω. The linear integer program itself can then be stated as follows.

$$\mathbf{max} \sum_{r=1}^{n} y_r \tag{5}$$

subject to:

$$\sum_{c_j \in \Sigma} x_{i,c_j} = 1 \quad \text{for } i = 1, \ldots, m \tag{6}$$

$$\sum_{i=1}^{m} x_{i,s_i^r} \leq m - t \cdot y_r \quad \text{for } r = 1, \ldots, n \tag{7}$$

$$x_{i,c_j}, y_r \in \{0, 1\}$$

Hereby, constraints (6) ensure that for each position i of a possible solution exactly one character from Σ is chosen. Moreover, constraints (7) ensure that y_r can only be set to 1 if and only if the number of differences between $s^r \in \Omega$ and the possible solution (as defined by the setting of the variables x_{i,c_j}) is at least t. Remember, in this context, that s_i^r denotes the character at position i in $s^r \in \Omega$.

Software package. In the Google Drive folder of the course you find a software package written in C++ (see sub-folder `software`). The package contains basic code for reading FFMSP problem instances. Moreover, it contains templates in C++ for the implementation of a greedy algorithm, a local search method, a metaheuristic, and a hybrid metaheuristic. It also contains a template for the implementation of the ILP model of the FFMSP problem for solving it with CPLEX. The software is implemented for GCC and Linux, but also compiles in Visual Studio, for example, under Windows. Versions of CPLEX for Linux, Windows and Mac can be downloaded from the respective Google Drive subfolder. Note that, before compiling, CPLEX must be installed on your machine (see the `Readme` file for Linux). After having installed CPLEX, the above-mentioned templates can be compiled in Linux executing the `make` command.

Important: for a positive course evaluation you should choose and complete at least 3 tasks out of the first 5 tasks. Moreover, everyone has to complete tasks 5 and 6.

Task 0: FFMSP ILP model for CPLEX. Using the file `cplex.cpp`, implement the ILP model for the FFMSP in order to be solved by CPLEX. Note that in the Google Drive subfolder CPLEX_example_code you find an exemplary implementation of of an ILP model for another NP-hard problem called *minimum dominating set* (MDS). Consult the `Readme` file of that subfolder for learning how to compile and use the example.

Task 1: Greedy heuristic. Design and implement a greedy algorithm for the FFMSP, using the file `greedy.cpp`. The output of the file and the facilities for time measurement will be explained during class.

Task 2: Local search method. Design and implement a local search method for the FFMSP using the file `local_search.cpp`. Possibly make use of your greedy algorithm for obtaining an initial solution for your local search. Otherwise, start with a random solution.

Task 3: Metaheuristic. Design and implement a local search method for the FFMSP using the file `metaheuristic.cpp`. Your metaheurisitic might use in some way your greedy heuristic and your local search method. This is, however, not required.

Task 4: Hybrid metaheuristic. Design and implement a hybrid metaheuristic for the FFMSP using the file `hybrid_metaheuristic.cpp`. You might adapt one of the hybrid metaheuristics seen in class for this purpose, or you might invent your own one.

Task 5: Experimental evaluation. Apply your implemented techniques to all problem instances from the Goole Drive folder `instances`. For more information consult file `Readme-experimentation` in the same folder.

Tarea 6: Writing a report. Write a report that contains the following. A short description of your implemented techniques and the results of the experimental evaluation. Showing the results in numerical form in a table is sufficient. But, if you want, you might also use graphical means for this purpose.

Tarea 7: Sumitting the report. Your report should be sent in PDF format to the email address `christian.blum@csic.es`. Deadline is April 8, 2022.