MRIGANK BADOLA 2K18/IT/073

Multi-objective problems are those where the quality of a solution is defined by its performance in relation to several, possibly conflicting, objectives. In practice it turns out that a great many applications that have traditionally been tackled by defining a single objective function (quality function) have at their heart a multi-objective problem that has been transformed into a single-objective function in order to make optimisation tractable.

Let's take the case study from a research paper:

Designing a Novel Hybrid Swarm Based Multi-Objective Evolutionary Algorithm for Finding DNA Motifs

In this paper they presented a novel local search for improving the ability of multi-objective evolutionary algorithms when finding repeated patterns -motifs- in DNA sequences.
In the metaheuristic design, two competing goals must be taken into account: exploration and exploitation. Exploration is needed to cover most of the optimization problem search space and provide a reliable estimation of the global optimum. In turn, exploitation is also important since normally the solutions refinement allows the achievement of better results.

In this work they took advantage of both concepts by combining the exploration capabilities of a population-based evolutionary algorithm and the power of a local search, especially designed to optimize the Motif Discovery Problem.

To solve the MDP they had to optimize three conflicting objective functions: motif length, support, and similarity; as well as satisfy a set of constraints. Given a set of sequences $S = \{S_i \text{ where } i = 1, 2, \ldots, D\}$ of nucleotides defined on the alphabet $B = \{A, C, G, T\}$. $Si = \{S_i^j \text{ where } j = 1, 2, \ldots, w_i\}$ is a sequence of nucleotides, where w_i is the sequence width. The set of all the sub-sequences contained in S is $\{s_i^{j_i} \text{ where } i = 1, 2, \ldots, D, j_i = 1, 2, \ldots, w_i - l + 1\}$, where j_i is the binding site of a possible motif instance s_i^j on sequence S_i, and l is the motif length, the first objective to be maximized. To obtain the values of the other two objectives they had to buildthe Position Indicator Matrix (PIM) $A = \{A_i \text{ where } i = 1, 2, \ldots, D\}$ of the motif, where

$A_i = \{A_i^j$ where $j = 1, 2, \ldots, w_i\}$ is the indicator row vector with respect to a sequence S_i. A_i^j is 1 if the position j in S_i is a starting position of a binding site, and 0 otherwise. They referred to the number of motif instances as $|A| = \sum_{i=1}^{D} \sum_{j=1}^{w_i} A_i^j$ Only those sequences that achieve a motif instance of certain quality with respect to the consensus motif are taken into account when they built the final motif. This is indicated by the second objective to be maximized, the support.

$S(A) = \{S(A)_1, S(A)_2, \ldots, S(A)_{|A|}\}$ is a set of $|A|$ motif instances, where $S(A)_i = S(A)_i^1 S(A)_i^2 \ldots S(A)_i^l$ is the i^{th} motif instance in $|A|$. S(A) can also be expanded as $(S(A)^1, S(A)^2, \ldots, S(A)^l)$, where $S(A)_j = S(A)_j^1 S(A)_j^2 \ldots S(A)_j^l$ is the list of nucleotides on the j^{th} position in the motif instances. Then, they built the Position Count Matrix (PCM) N(A) with the different nucleotide bases on each position of the candidate motifs (A) which have passed the threshold marked by the support. $N(A) = \{N(A)^1, N(A)^2, \ldots, N(A)^l\}$ and $N(A)^j = \{N(A)_b^j$ where $b \in B\}$ where $N(A)_b^j = |\{S(A)_i^j | S(A)_i^j = b\}|$. The dominant nucleotides of each position are normalized in the Position Frequency Matrix (PFM) $\widehat{N} = \frac{N(A)}{|A|}$. Finally, they calculated the third objective value, the similarity, by averaging all the dominance values of each PFM column, as is indicated in the following expression:

$$Similarity\ (motif) = \frac{\sum_{i=1}^{l} max_b\{f(b,i)\}}{l}$$

Where f(b,i) is the score of nucleotide b in column i in the PFM and $max_b\{f(b, i)\}$ is the value of the dominant nucleotide in column i.

$Complexity = \log_N \frac{l!}{\prod(n_i!)}$ where N=4 or DNA sequences, l is the motif length, and n_i is the number of nucleotides of type $i \in \{$ A, C, G, T $\}$.

Finally, we look at local search operation. The implemented local search defines three important parameters: the window size (WS), the search direction (DIR), and the reference string (REF).

The window size defines the substring length that they would attempt to find in the corresponding DNA sequences.

WS = 1 to 7 since 7 is minimum motif's length

The search direction is the direction that we must take to find the selected substring. DIR = 0 indicates that we begin the search from the beginning of the sequence, i.e., from the nucleotide 0; DIR = 1 that we have to choose a random direction (right or left) from the starting positions of the corresponding candidate motif; DIR = 2 that we have to check the quality of the solutions resulting from searching in both directions (right and left from the starting position) and select the one which produces the best solution.

The REF parameter indicates which motif (among all candidates and the consensus motif) is used as reference. REF = 0 we select the candidate motif of the first sequence, with REF = 1 we choose the candidate of a randomly selected sequence, with REF = 2 we use the consensus motif, and with REF = 3 we consider the candidate closer to the consensus motif at a nucleotide level

Let's look at the 4 steps required.

- We set the value of the WS, DIR, and REF parameters. - Example: We use WS = 5, DIR = 1, and REF = 2.
- We load the reference string considering the REF parameter. - Example: As REF = 2, we use the consensus motif as reference string, in this case: ACGTAACG
- We repeat the following steps on all sequences where we have found any candidate motif. - Example: We repeat the following steps on the 5 sequences that compose the solution
  - We select a random substring among the possible windows. - Example: As WS = 5 and the consensus motif is ACGTAACG, we randomly choose one among the four possible substrings: ACGTA, CGTAA, GTAAC, or TAACG. In this case we select the first substring: ACGTA.
  - We search the substring of size WS in the corresponding sequence following the indications of DIR.- Example: With DIR = 1 we search the substring in the corresponding sequence following a random direction, right or left of the candidate motif starting position.
  - If we find the substring we update the starting location of the processed candidate motif.

- If WS > 1 and we do not find the substring, we return to the step 3.2, reducing the value of WS by one. Example: If we do not find the ACGTA substring in the sequence and WS > 1, we reduce the WS value in one unit(WS = WS −1) and we will search, in this case, the ACGT substring.

- We evaluate the resulting solution and if it is able to dominate the previous one, we exchange them.

And hence by applying this hybrid algorithm, this problem can be solved.