Bioinformatics, as the name suggests, is an interdisciplinary area covering biology and informatics. Its two major goals are to: (i) develop computational methods and tools to analyze various kinds of biological data, and (ii) discover new biological knowledge through the use of these tools. To achieve the former goal, various computational techniques have been applied and developed. Combinatorial optimization plays an important role here since many problems in bioinformatics can be formalized as combinatorial optimization problems [1, 2]. In this article, I briefly review how operations research optimization techniques have been applied to bioinformatics and discuss future developments.

It is well-known that genetic information is stored in DNA sequences in almost all organisms, where a DNA sequence is represented by a string of four characters (A, C, G, T) and a gene corresponds to a part of a DNA sequence. Genetic information stored in DNA is translated into amino acid sequences, with each sequence represented by a string of 20 characters or more. Analysis of DNA and amino acid sequences is a key area of bioinformatics. Measuring the similarity between two DNA sequences or two amino acid sequences is fundamental because genes or proteins having similar sequences very often show similar biological and chemical properties.

The similarity of two sequences is usually measured through sequence alignment [2]. A sequence alignment is a superposition between two sequences obtained by appropriately inserting gap symbols denoted by ‘-’ to both sequences so that the resulting sequences have the same length. Suppose that we have two DNA sequences, ACGTTGCAACGT and ACGGGTTGAACCT. Then, the following is an example of an alignment:

\[
\begin{align*}
   \text{ACG} & \quad \text{TTG} & \quad \text{CAA} & \quad \text{ACGT} \\
   \text{ACG} & \quad \text{GTT} & \quad \text{GAA} & \quad \text{ACT}
\end{align*}
\]

where the letters in the same column correspond to each other. There is an exponential number of possible alignments (i.e., the number of ways of inserting gap symbols is exponential). In order to measure the similarity between two sequences, we need to compute an alignment which maximizes the score. Although various scoring schemes have been proposed, we consider here the simplest one in which +1 is assigned to each column consisting of identical letters and ‑1 to other columns and the score is given by the sum of scores of all columns. Then, the score of the above alignment is \(1 + 1 + 1 - 1 + 1 + 1 - 1 + 1 + 1 - 1 + 1 + 1 - 1 = 7\), which is optimal. A naive algorithm that examines all possible alignments will take exponential time. However, this problem can be solved in \(O(n^2)\) time by a dynamic programming algorithm using a two-dimensional table, where it is assumed that both input sequences have length \(O(n)\). Dynamic programming techniques have been applied to many combinatorial optimization problems in bioinformatics, which include prediction of RNA (secondary) structures, comparison of phylogenetic trees, and estimation of the parameters in some probabilistic model (e.g., hidden Markov model).

Although an \(O(n^2)\) time algorithm is enough for comparing short sequences, it is not efficient if used to compare very long sequences (e.g., comparison of genome sequences) and to search for a similar sequence in a database. In such a case, it is impossible to obtain an optimal solution and thus various heuristic techniques, which include a kind of hash functions, have been developed and used. In some cases, it is required to compare multiple sequences simultaneously. Sequence alignment is therefore generalized to \(k\) sequences, which is called multiple alignment. Multiple alignment can be solved in \(O(nk)\) time by dynamic programming using a \(k\)-dimensional table, but is known to be \(NP\)-hard (i.e., computationally intractable) if \(k\) is not fixed. Although there has been no significant improvement on the worst-case time complexity in over 30 years, many heuristic and practical algorithms have been developed for multiple alignment. Due to recent rapid progress of the DNA sequencing technology, a huge number of DNA sequences are being produced. Therefore, how to manage and analyze these huge data is an urgent research topic, which needs very efficient optimization techniques.

Analysis of other types of data, which include three-dimensional protein structures, metabolic networks, and gene expression data, is also important in bioinformatics. Analysis of biological networks and gene expression data is also an important topic in the field of systems biology [1]. Many problems on these data are \(NP\)-hard. In my experience, most problems in bioinformatics can either be: efficiently solved by dynamic programming or \(NP\)-hard. There are several ways to cope with \(NP\)-hardness. One practical way is to use meta-heuristics such as evolutionary computing and simulated annealing. Although such an approach is often very useful, the optimality of a solution is not guaranteed. In order to guarantee the optimality, use of the branch-and-bound technique is a good choice because the size of protein sequences and the size of relevant subnetworks are usually not too large (e.g., less than one thousand). Furthermore, in many cases, problems can be formalized as integer linear programs [1]. Once a problem is formalized as an integer linear program, we can employ efficient and practical solvers, thanks to developments in operations research. This approach has the following merits: (i) it is often easy to write a program to translate an original instance to an integer program, and (ii) improvement of a solver directly leads to improvements in the solution of many problems.

One disadvantage of this approach is the great variation in the time to reach an optimal solution. A solution may be reached rather quickly for a problem while a problem of similar size may not be obtained at all. Another approach is to develop exact algorithms. For example, suppose that we can develop an \(O(1.05n)\) time algorithm for some problem, where a constant factor hidden in \(O\) notation should not be too large. Then, the algorithm would work for instances of \(n \leq 500\). Although this approach is not generally applicable, it is worth trying for important problems.

Apart from optimization type problems, there are many prediction type problems in bioinformatics. Machine learning techniques have been extensively applied to these problems. Since learning tasks are often formalized as continuous or combinatorial optimization problems, optimization techniques again play an important role.

As shown, various kinds of optimization techniques have been extensively applied to bioinformatics. However, rapid increases in biological data call for improvements of optimization techniques. The bioinformatics community looks to OR in its quest for more efficient optimization techniques. In turn, bioinformatics continues to be a source of novel optimization problems to the operations research community.