Mathematical and Computational Applications, Vol. 18, No. 3, pp. 176-197, 2013

SOFT COMPUTING METHODS IN BIOINFORMATICS: A COMPREHENSIVE REVIEW

Bekir KARLIK

Selçuk University, Department of Computer Engineering, Konya, Turkey
bkarlik@selcuk.edu.tr

Abstract- Applications of genomic and proteomic, epigenetic, pharmacogenomics, and systems biology have shown increased a lot, resulting in an explosion in the amount of highly dimensional and complicated data being generated. The data of bioinformatics fields are always with high-dimension and small samples. Genome-wide investigations generate in large numbers of data and there is a need for soft computing methods (SCMs) such as artificial neural networks, fuzzy systems, evolutionary algorithms, metaheuristic and swarm intelligence algorithms, statistical model algorithms etc. that can deal with this amount of data. The use of soft computing methods has been increased to a variety of bioinformatics applications. It is used to inquire the underlying mechanisms and interactions between biological molecules in a lot of diseases, and it is a main tool in any biological (or biomarker) discovery process. The aim of this article is to introduce soft computing methods for bioinformatics. These methods present supervised or unsupervised classification, clustering and statistical or stochastic heuristics models for knowledge discovery. In this article, the current problems and the prospects of SCMs in the application of bioinformatics is also discussed.

Key Words- Soft computing, bioinformatics, computational methods, algorithms

1. INTRODUCTION

Bioinformatics research, develop, and apply computational approaches for analyzing, and thus expanding, the use of biological, behavioral, and medical data. There are many biological domains of bioinformatics where SCMs are applied for knowledge extraction of biological and medical problems of from data. These problems can be classified into six different domains: genomics, proteomics, microarrays, systems biology, evolution and text mining [1].

Genomics domain is one of the most important domains in bioinformatics which discipline in genetics applications recombinant DNA, DNA sequencing methods, and to sequence, assemble, and analyze the function and structure of genomes. Genomics data requires pre-processing in order to acquire useful information. As a first step, from genome sequences, it is possible to extract the location and structure of the genes. Sequence information can be used for gene function and RNA secondary structure prediction [2,3].

Proteomic domain is an essential application of SCMs for protein structure prediction. Proteins are very complicated macromolecules with thousands of atoms and bounds. For this reason, the number of possible structures consists of very big data which makes protein structure prediction a very complicated combinatorial problem where complex optimization methods such as SCMs are required [1].

Genomic and proteomic data analysis is essential tools for understanding the underlying factors that are included in human illness problems [4]. Applications of genomic and proteomic technologies have seen a high increase, resulting in a huge amount of multi dimensional and complicated data being created [5,6]. This is due to their ability to get over with multi dimensional complicated datasets such as those developed by protein mass spectrometry and DNA microarray experiments. As such, artificial neural networks have been applied to diagnosis of illness problems and authentication of biomarkers. Feature selection is used along with classifier architecture to avoid over-fitting, to create more efficient classifier and to supply more insights into the underlying causal relationships [7].

Microarray domain is the management of complicated experimental data for application of computational methods in biology. Complicated experimental data causes of two types of problems. The first one is data need to pre-processing, i.e. modified to be suitably used by SCMs. The second is the analysis of the data which depends on what we search for. The most well known applications are on pattern recognition, classification and genetic network in the case of the microarray data [1].

Systems biology domain is another important domain of biology that incorporates with the soft computing methods. Systems biology is the work of systems of biological components, which might be molecules, cells, organisms or entire species. It is very complicated to model the life processes that take place inside the cell. Thus, SCMs are extremely helpful when modeling biological networks especially genetic networks, signal transduction networks and metabolic pathways [8].

Evolution domain, especially phylogenetic tree reconstruction can also take advantage of SCMs. Phylogenetic trees (or evolutionary tree) is a schematic representations of organisms' evolution demonstration the inferred evolutionary relationships a variety biological species (or other entities) based upon similarities and differences in their physical and genetic features. There are many different reasons behind the alignment of biological sequences. Biological sequence alignment helps to discover functional and structural similarity of sequences. Scientists work with these aligned sequences to constitute phylogenetic trees, characterize protein families, and estimate protein structure [9-10]. Generally, they were constituted belonging to different features such as morphological features, metabolic features, etc. but, nowadays, with the great amount of genome sequences available, phylogenetic tree construction algorithms are based on the comparison between different genomes. This comparison is made by means of multiple sequence alignment, where optimization methods are very useful.

Text mining domain is a side effect of the application of SCMs for bioinformatics because of the increasing amount of data. This allows for a new source of valuable information which is required for the knowledge extraction. Thus, text mining is becoming more and more interesting in computational biology, and it is being applied in functional annotation, cellular location estimation and protein interaction analysis [11].

Bioinformatics is a discipline that built upon the fields of computer and information sciences. It relies mainly upon strategies to achieve, store, organize, archive, analysis, and visualize data. Bioinformatics (or computational biology) encompasses the development and application of data-analytical and theoretical methods, mathematical modeling and computational simulation techniques to the study of biological, medical, and behavioral systems. SCMs are well-suited for many bioinformatics problems including gene selection, clustering and classification, signal processing and image analysis in bioinformatics works, supervised or unsupervised classification with multi-dimensional input variables is frequently encountered. Thus, SCMs are able to get over with multi dimensional complicated datasets [12]. SCMs are used to solve other bioinformatics problems. SCMs can be divided into two class as supervised and unsupervised learning rules. Unsupervised or clustering techniques is used to group similar genomic or proteomic profiles and therefore is elucidate relationships within sample groups. These techniques is also assigned biomarkers to sub-groups based on their expression profiles across patient samples. Although clustering is useful for exploratory analysis, it is delimited due to its inability to incorporate expert knowledge. Furthermore, classification and feature ranking are supervised, knowledge-based soft computing methods that estimate the distribution of biological expression data and, in doing so, can extract important information about these experiments. Classification is closely coupled with feature ranking, which is a main data reduction technique that uses to estimate classification error or other statistical tests to score features [13]. Figure 1 shows that Classification of the topics where soft computing methods are applied [1].

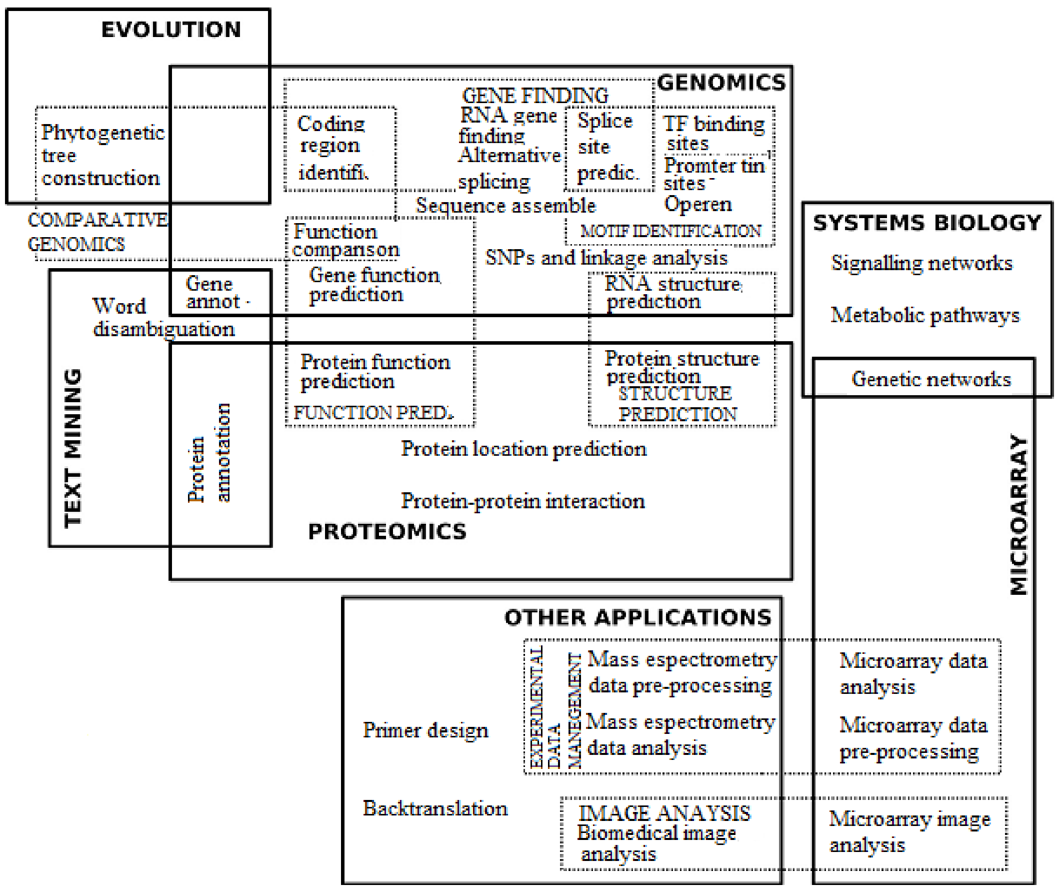

Figure 1. Classification of the topics where soft computing methods

This article is organized as follows: The first section is an introduction to the literature of previous researches. The second section presents SCMs and gives some brief information for different types of well known algorithms of SCMs. The third section discusses the applications of SCMs on some bioinformatics problems. This section is also defined literature studies on bioinformatics. Final section explains the conclusions of this revision on SCMs in bioinformatics.

2. SOFT COMPUTING METHODS

Soft computing methods (SCMs) consist in programming computers to optimize a performance criterion by using example data or past experience. Basically, soft computing is not a homogeneous body of concepts and methods. Rather, it is a partnership of distinct techniques that in one way or another conform to its guiding principle. At this point, the aim of soft computing is to utilize the tolerance for imprecision and uncertainty to achieve tractability, robustness and low solutions cost. SCMs deal with imprecision, uncertainty, partial truth, and approximation to achieve practicability, robustness and low solution cost [14]. It must be noticed that the efficiency of the learning and inference algorithms, as well as their space and time complexity and their transparency and interpretability, can be as important as their learning accuracy [15]. The well known SCMs are: Artificial neural networks, Fuzzy systems, Bayesian network, Evolutionary algorithms, Genetic algorithms, Metaheuristic and Swarm Intelligence (such as Ant colony optimization, Bees algorithms, Bat algorithm, Cuckoo search, Harmony search, Firefly algorithm, Artificial immune systems, Particle swarm optimization etc), and Chaos theory. Generally speaking, SCMs resemble biological processes more closely than traditional methods, which are largely based on formal logical systems, such as sentential logic and predicate logic, or rely heavily on computer-aided numerical analysis (as in finite element analysis). SCMs are intended to complement each other. Unlike hard computing schemes, which strive for exactness and full truth, SCMs exploit the given tolerance of imprecision, partial truth, and uncertainty for a particular problem. Another common contrast comes from the observation that inductive reasoning plays a larger role in soft computing than in hard computing.

2.1. Artificial Neural Networks

Artificial Neural Networks (ANN) is an information processing model, implemented in hardware or software that is modeled after biological process of the brain studied. Artificial neural network has ability to derive meaning from imprecise or complicated data to extract patterns and to detect trends that are not easily to recognize by humans or other computer techniques [16-17]. ANN has been mainly used to examine the complicated relationships between input and output variables in many scientific and technological areas including biomedical and bioinformatics [18-19]. Some well-known ANN algorithms such as Back-Propagation (BP), Radial Based Function (RBF), and Support Vector Machines (SVM) are mostly used to solve bioinformatics problems.

The algorithm of Back-propagation used generalized delta learning rule is an iterative gradient algorithm designed to minimize the root mean square error between the actual output of a multilayered feed-forward ANN and a desired output. Each layer is fully connected to the previous layer, and has no other connection. The algorithm of Back-propagation classifier can be described as [20];

- Initialization: Set all the weights and biases to small real random values.
- Presentation of input and desired outputs: Present the input vector x(1), x(2),…,x(N) and corresponding desired response d(1),d(2),…,d(N), one pair at a time, where N is the number of training patterns.
- Calculation of actual outputs: Use Equation given below to calculate the output signals

  $$y_1, y_2, ..., y_{N_M} \quad y_i = \varphi\left(\sum_{j=1}^{N_{M-1}} w_{ij}^{(M-1)} x_j^{(M-1)} + b_i^{(M-1)}\right),\ i = 1, ..., N_{M-1}$$

- Adaptation of weights (w_{ij}) and biases (b_i):

  $$\Delta w_{ij}^{(l-1)}(n) = \mu . x_j(n) . \delta_i^{(l-1)}(n)$$

  $$\Delta b_i^{(l-1)}(n) = \mu . \delta_i^{(l-1)}(n)$$

  where

  $$\delta_i^{(l-1)}(n) = \begin{cases} \varphi'(net_i^{(l-1)})\left[d_i - y_i(n)\right], & l = M \\ \varphi'(net_i^{(l-1)}) \sum_k w_{ki} . \delta_k^{(l)}(n), & 1 \le l \le M \end{cases}$$

in which $x_j(n)$= output of node *j* at iteration *n*, *l* is layer, *k* is the number of nodes of output of neural network, *M* is output layer, φ is activation function [21]. Sigmoid and hyperbolic tangent activation functions are more effective than the other activation functions [22]. The learning rate is indicated by μ. It may be noted here that a large value of the learning rate may cause to faster convergence but may also result in oscillation.

Radial basis function (RBF) neural network is based on supervised learning. RBF's are embedded in a two layer neural network, where each hidden layer implements a radial activated function. The output layer realize a weighted sum of outputs of hidden layer. All hidden nodes simultaneously receive the n-dimensional real valued input vector X. The output of hidden-layer, Z_j is obtained by closeness of the input X to an n-dimensional parameter vector μ_j associated with the *j*th hidden layer. The response characteristics of the *j*th hidden layer (*j* = 1, 2, …, *J*) is assumed as

$$Z_j = K(\| x - \mu_j \| / \sigma_j^2)$$

where K is a strictly positive radials symmetric function (kernel) with a unique maximum at its 'centre' μ_j and which drops off very fast to zero away from the centre. σ_j is the width of the receptive field in the input space from layer *j* [23]. This means that Z_j has an perceptible value only when the distance $\| x - \mu_j \|$ is smaller than the width σ_j.Given an input vector *X*, the output of the RBF network is the *L*-dimensional activity vector *Y*, whose *l*th component (*l* = 1, 2 … *L*) is given by,

$$Y_l(x) = \sum_{j=1}^{j} w_{lj} Z_j(x)$$

Support Vector Machines (SVM) is specifically used to solve a binary classification problem in a supervised manner and the learning problem is formulated as a quadratic optimization problem where the error surface is free of any local minimum and has global optimum [24]. SVM is to build an optimal separating hyper plane in such a way that the margin of separation between two classes is maximized. SVM accomplish this desirable property on the basis of the principle of structural risk minimization. To realize the SVM based classifiers for linearly separable patterns, let us consider a training set indicated by $\{(x_j,y_j)\}$ (j=1,..., N), where x_j is the n-dimensional input feature vector and y_j indicates the desired (or target) output. The input patterns indicated by the desired output $y_j = 1$ constitute the positive group and the desired output $y_j = -1$ constitute the negative group [25].

Now suppose we have a machine whose task it is to learn the mapping $x_j \rightarrow y_j$. The machine is defined by a set of possible mappings $x \rightarrow f(x;\alpha)$, where the functions $f(x;\alpha)$ themselves are sorted by the adjustable parameters α. The machine is assumed to be deterministic: for a given input x, and selection of α, it will always give the same output $f(x;\alpha)$. A particular selection of α creates what we will call “trained machine.” Thus, for example, an ANN with fixed structure, with α corresponding to the weights and biases, is a learning machine in this sense. The expectation of the test error for a trained machine is therefore:

$$R(\alpha) = \int \frac{1}{2} \left| y - f(x,\alpha) \right| dP(x,y)$$

Note that, when a density $p(x; y)$ exists, dP(x;y) can be written $p(x;y)$dxdy. This is a nice way of writing the true mean error, but unless we have an estimate of what P(x; y) is, it is not very useful. The quantity R(α) is named the expected risk, or just the risk. Here we will call it as the actual risk, to emphasize that it is the quantity that we are interested in. The “empirical risk” $R_{emp}(\alpha)$ is defined to be just the measured mean error rate on the training set (for a fixed, finite number of observations):

$$\mathrm{Re}\, mp(\alpha) = \frac{1}{2l} \sum_{j=1}^{l} \left| y_j - f(x_j,\alpha) \right|$$

Note that no probability distribution appears here. $R_{emp}(\alpha)$ is a fixed number for a particular choice of α and for a particular training set fxi; yig. The quantity $1/2|y_i$ - $f(x_i$, $\alpha|$ is called the loss. For the case described here, it can only take the values 0 and 1. Now choose some η such that $0 \leq \eta \leq 1$. Then for losses taking these values, with probability 1- η, the following bound holds)[26]:

$$R(\alpha) \leq \mathrm{Re}\,mp(\alpha) + \sqrt{\left(\frac{h(\log(2l/h)+1) - \log(\eta/4)}{l}\right)}$$

where h is a non-negative integer called the Vapnik Chervonenkis dimension, and is a measure of the notion of capacity mentioned above. In the following we will call the right hand side of the last equation as risk bound [27].

2.2. Fuzzy Systems

The Fuzzy system model is the knowledge-based model with linguistic rules. Fuzzy sets are described for all input and output variables and the set of rules. Fuzzy logic ensures the means to process this knowledge and compute output values for given input data. The main problem of this approach is to find a suitable set of linguistic rules that define the system to be modeled [28]. Fuzzy systems is represented in the form of if-then rules or fuzzy conditional statements as in the expression of the form IF A THEN B, where A and B are labels of the fuzzy sets. The set of rules should be complete and provide an answer for every input value.

Fuzzy systems consist of three steps as the fuzzification, fuzzy inference and the defuzzification. The fuzzification module pre-processes the input values submitted to the fuzzy expert system. The inference engine uses the results of the fuzzification module and accesses the fuzzy rules in the fuzzy rule base to infer what intermediate and output values to produce. Fuzzification is the transformation of numerical variables into linguistic variables and the corresponding allocation of the grade of membership (changing between 0 and 1) to the different membership functions [29]. The linguistic combination of the traits is achieved in the fuzzy inference system (FIS). There are two types of FIS models; Mamdani FIS model and Sugeno FIS model. Here we have only described Mamdani FIS model. The rules used are resulted from human knowledge and have the form: if condition, then conclusion. The degree to which each part of the condition has been fulfilled for each rule is known by the belonging grades of membership. The final output of the fuzzy system is provided by the defuzzification module. Through the calculation of the centre of gravity of these areas, the fuzzy values are converted back in order to resolve a single output value from the set. The centroid technique is used for defuzzification. The centroid of composed shape is computed by,

$$Z = [\mu_c(z)z\delta z]/[\mu_c(z)\delta z]$$

where z is the consequent variable and $\mu_c(z)$ is the function of the composed form [30].

Fuzzy c-means (FCM) clustering algorithm is often used as an initial step for the fuzzy systems to find membership values of each training data vector in each cluster. These membership values are assumed to represent best partitions of the given dataset [31]. Formally, clustering an unlabeled data $X = \{x_1, x_2, \ldots, x_N\} \subset Rh$, where N represents the number of data vectors and h the dimension of each data vector, is the assignment of c partition labels to the vectors in X. c-partition of X constitutes sets of (cN){uik} membership values that can be arranged as a (c × N) matrix U = [uik]. The problem of fuzzy clustering is to find the optimum membership matrix U. The most often used function for fuzzy clustering is the weighted within-groups sum of squared errors Jm, which is used to describe the following constrained optimization problem [32]:

$$\min\{j_m(U,V,X) = \sum_{k=1}^{N}\sum_{i=1}^{c}(u_{ik})^m \|x_k - v_i\|_A^2$$

where

$$U \in M_{fcn} = \left\{ U \in \Re^{cN} \,\middle|\, \begin{array}{l} 0 \leqslant u_{ik} \leqslant 1 \ \forall ik \ \& \ \forall k, u_{ik} > 0 \ \exists i \\ 0 < \sum_{k=1}^{N} u_{ik} > \eta \ \forall i \ \& \ \sum_{i=1}^{c} u_{ik} = 1 \ \forall k \end{array} \right\}$$

$V = \{v1, v2, \ldots, vc\}$ is the vector of (unknown) cluster centers, and $\|x\|_A = (x^T Ax)^{1/2}$ an inner product norm. A is an $h \times h$ positive definite matrix, which specifies the form of the clusters. The matrix A is generally selected as the identity matrix, leading to Euclidean distance and, consequently, to spherical clusters. Fuzzy partitions are implement using the FCM algorithm through an iterative optimization of considering the following steps [33]:

- Choose the number of clusters *(c)*, weighting exponent *(m)*, iteration limit (iter), termination criterion *(*_>0*)*, and norm for error $\|V_t - V_{t-1}\|$.
- Guess initial position of cluster centers: $V_0 = \{v_{1,0}, v_{2,0}, \ldots, v_{c,0}\} \subset R^{ch}$.
- Iterate for $t = 1$ iter, calculate

$$u_{ik,t} = \left[\sum_{j=1}^{c}\left(\frac{\|X_k - V_{i,t-1}\|_A}{\|X_k - V_{j,t-1}\|_A}\right)^{2/m-1}\right]^{-1} \quad \text{and} \quad V_{i,t} = \frac{\sum_{k=1}^{N}(u_{ik,t})^m x_k}{\sum_{k=1}^{N}(u_{ik,t})^m}$$

- IF error=$\|V_t - V_{t-1}\| \leq \varepsilon$, THEN stop, and put $(U_f, V_f) = (U_t, V_t)$ for NEXT t.

There is some special model to find the optimum number of clusters model such as the fuzzy function cluster validity index [34].

2.3. Statistical Model Algorithms

Different statistical classification algorithms can also use to solve bioinformatics problems such as K- Nearest Neighbors and Naïve Bayes.

K Nearest Neighbor (K-NN) is an simple non parametric algorithm which is a method for classifying cases based on their similarity to other cases. Similar cases are near each other and dissimilar cases are distant from each other. Thus, the distance between two cases is a measure of their dissimilarity. Training a nearest neighbor model involves computing the distances between cases based upon their values in the feature set. The nearest neighbors to a given case have the smallest distances from that case. The distance is calculated using one of the following measures[35]:

- Euclidean Distance
- Minkowski Distance
- Mahalanobis Distance

Simple K-NN algorithm consists of following steps:

- For each training example <x,f(x)>, add the example to the list of training examples,
- Given a query instance xq ¨ Given a query instance x to be classified, q to be classified, Let x1, x2….xk denote the k instances from training examples that are nearest to xq. Then, return the class that represents the maximum of the k instances.

A Naïve Bayes classifier is a simple but effective probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be independent feature model. According to the precise nature of the probability model, Naïve Bayes classifiers is trained efficiently in a supervised learning setting. In numerous applications, parameter estimation for Naïve Bayes uses the technique of maximum likelihood. In spite of their naive design and apparently over-simplified assumptions, Naïve Bayes classifiers often work much better in many complex real-world situations than one might expect[36]. Note that the naive Bayes classifier assumes the conditional independence of features. This assumption however does not hold in most cases. Despite this apparent violation of the assumption, the naive Bayes classifier exhibits good performance for various natural language processing tasks. An advantage of Naïve Bayes classifier is that it needs to less training data to estimate the parameters (means and variances of the variables) necessary for classification [37]. Naïve Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* decision rule. The corresponding classifier is the function classify defined as follows[36]:

$$classify(f_1,...,f_n) = \arg\max_C \, p(C=c)\prod_{i=1}^{n} p(F_i = f_i \mid C=c)$$

2.3. Metaheuristic and swarm intelligence algorithms

Recently, well-known modern heuristic algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO) are used on bioinformatics problems.

Genetic Algorithm (GA) is good candidates for this task since GA is most useful in multiclass, high-dimensionality problems where heuristic knowledge is sparse or incomplete. Holland [38] defined a methodology for studying natural adaptive systems and designing artificial adaptive systems. It is now often used as an optimization technique, based on an analogy to the process of natural selection in biology. A GA approach needs to a population of chromosomes representing a combination of features from the solution set, and needs to a cost function (called an valuation or fitness function). This function computes the fitness of each chromosome. The algorithm manipulates a finite set of chromosomes (the population), based loosely on the mechanism of evolution. In each generation, chromosomes are subjected to certain operators, such as crossover, inversion and mutation, which are analogous to processes which consists of in natural reproduction. Crossover of two chromosomes produces a pair of offspring chromosomes which are synthesis of the traits of their parents[39]. The Basic Genetic Algorithm consists of following steps:

1. Generate random population of n chromosomes ,
2. Evaluate the fitness f(x) of each chromosome x in the population,
3. Generate a new population by repeating following steps till the new population is complete;
    - Select two parent chromosomes from a population depending on their fitness
    - With a crossover probability cross over the parents to form a new offspring. If no crossover was performed, offspring is an exact copy of parents
    - With a mutation probability mutate new offspring at each locus
    - Place new offspring in a new population,
4. Use new generated population for a further run of algorithm,
5. If the end condition is satisfied, stop, and return the best solution in current population,
6. Go to step 2

Differential Evolution (DE) is a population-based search strategy very similar to standard evolutionary algorithms. The major difference is in the reproduction step where offspring is created from three parents using an arithmetic cross-over operator. DE is described for floating-point representations of individuals. DE does not use of a mutation operator that is related some probability distribution function, but introduces a new arithmetic operator which depends on the differences between randomly selected pairs of individuals [40]. For each parent, $x_i(t)$, of generation t, an offspring, $x'_i(t)$ is generated in the following way: Randomly select three individuals from the current population, namely $x_{i1}(t)$, $x_{i2}(t)$, and $x_{i3}(t)$, with $i_1 \neq i_2 \neq i_3 \neq i\psi$ and i_1, i_2, i_3..., U(1,…, s), where s is the population size. Select a random number $r\psi\tilde{}$ □ U(1,…, N_d), where $N_{d\psi}$ is the number of genes of a single chromosome. Then, for all genes $j\psi$= 1, ψN_d, if U(0,ψ_1) < ψP_r, or if $j\psi = r$, let;

$$x'_{i,j}(t)= x_{i3,j}(t)+\gamma[x_{i1}(t) – x_{i2}(t)] \text{ otherwise, let: } x'_{i,j}(t)= x_{i,j}(t).$$

Here, $P_r\psi$ is the probability of reproduction (with $P_r\psi\in[0\psi1]$), $\gamma\psi$ is a scaling factor with $\gamma \in (0\psi\infty)$, and $x'_{i,j}(t)$ and $x_{i,j}(t)$ indicate respectively the *j*th genes (or parameter) of the offspring and the parent. Thus, each offspring consists of a linear combination of three randomly chosen individuals when U(0,ψ1) < ψ P_r; otherwise the offspring inherits directly from the parent. Even when $P_r\psi$= 0, at least one of the parameters of the offspring will differ from the parent [41].

Particle swarm optimization (PSO) is an optimization technique which has been developed being inspired by the social behaviors of swarms like bird flocking or fish schooling by Kennedy and Eberhart [42]. In PSO method, each potential solution is referred as a particle and each particle has positions ($x_{i;j}$) and velocities ($v_{i;j}$) in a *j*-dimensional feature space [43]. The solution set which consists of the particles is called as swarm. At the beginning of the algorithm, each particle is generated by taking random values from the solution space. The success of each particle is determined employing a fitness function. Through the iteration process, the best instance of each particle and the swarm is kept as local bests ($P_{besti;j}$) and global best ($G_{besti;j}$) respectively. The velocity and position of each particle is updated utilizing these equations[44];

$$v_{i;j}(t+1) = wv_{i;j}(t) + c_1R_1(p_{besti;j} - x_{i;j}(t)) + c_2R_2(g_{besti;j} - x_{i;j}(t))$$

$$x_{i;j}(t+1) = x_{i;j}(t) + v_{i;j}(t+1)$$

where *i* is the index of the particle, *j* is the index of the position in particle, *t* shows the iteration number, $v_{i;j}(t)$ is the velocity of the *i*th particle in the swarm on *j*th index of the position in the particle and $x_{i;j}(t)$ is the position. R_1 and R_2 are the random numbers uniformly distributed between 0 and 1. c_1 and c_2 are the acceleration numbers and default values are 2 and w is the inertial weight. The original procedure for implementing PSO is as follows [45]:

1. Generate each particle randomly within the j-dimensional feature space.
2. Evaluate the success of each particle using the tness function.
3. If the success of the current particle is better than the success of Pbesti;j then determine Pbesti;j as the current particle.
4. If the success of the current particle is better than the success of Gbesti;j then determine Gbesti;j as the current particle.
5. Update the velocity and position of the particle using equations given above.
6. Repeat the steps from 2 to 5 until the stopping criteria or maximum iteration is reached.

Artificial Bee Colony (ABC) algorithm has been presented by Karaboga for optimizing numerical problems. The algorithm simulates the intelligent foraging behavior of honey bee swarms. It is a very simple but efficient, robust and population based stochastic optimization algorithm. In ABC algorithm, the colony of artificial bees includes three groups of bees: employed bees, onlookers and scouts. A bee waiting on the dance area for making a decision to select a food source is named onlooker and one going to the food source visited by it before is called employed bee. The other kind of bee is scout bee that carries out random search for discovering new sources. Pseudo-code of the ABC algorithm is [46]:

- Load training samples,
- Generate the initial population z_i, (i=1...SN),
- Evaluate the fitness (f_i) of the population,
- Set cycle to 1,
- repeat
- For each employed bee{
  Produce new solution v_i by using ($v_{ij} = z_{ij} + \phi_{ij}(z_{ij} - z_{kj})$
  Compute the value f_i

Apply greedy selection process}

- Compute the probability values p_i for the solutions (z_i) by (p_i)

$$p_i = \frac{\text{fit}_i}{\sum_{n=1}^{SN} \text{fit}_n}$$

- For each onlooker bee{
  Select a solution z_i depending on p_i
  Produce new solution v_i
  Calculate the value f_i
  Apply greedy selection process}
- If there is an abandoned solution for the scout
  then replace it with a new solution which will be randomly produced by

$$z_i^j = z_{\min}^j + \text{rand}(0, 1)(z_{\max}^j - z_{\min}^j)$$

- Memorize the best solution so far
- cycle=cycle+1
- until cycle=MCN

Ant Colony Optimization (ACO) agents mimic the foraging behavior of their biological counterparts in finding the shortest-path to the food source. The first algorithm following the principles of the ACO metaheuristic is the Ant System, where ants iteratively construct solutions and add pheromone to the paths corresponding to these solutions[47-48]. Path selection is a stochastic procedure based on two parameters, the pheromone and heuristic values. The pheromone value gives an pointing of the number of ants that select the trail recently, while the heuristic value is a problem dependent quality measure. When an ant arrives a decision point, it is more likely to select the trail with the higher pheromone and heuristic values. Once the ant reaches at its destination, the solution corresponding to the ant's followed path is evaluated and the pheromone value of the path is increased accordingly. In addition to, evaporation admits of the pheromone level of all trails to diminish gradually. Therefore, trails that are not reinforced gradually lose pheromone and will in turn have a lower probability of being chosen by subsequent ants. ACO algorithm consists of the specification of the following aspects [49];

- An environment that indicates the problem domain in such a way that it lends itself to incrementally building a solution to the problem.
- A problem dependent heuristic evaluation function, which ensures a quality measurement for the different solution components.
- A pheromone updating rule, which bring in account the evaporation and reinforcement of the trails.
- A probabilistic transition rule based on the value of the heuristic function and on the strength of the pheromone trail that determines the path taken by the ants.
- A clear specification of when the algorithm converges to a solution.

There are some more algorithms which are used to solve bioinformatics problems such as Hidden Markov Model, Decision Trees, Stochastic Optimization, Dynamic Programming, Stochastic Context Free Grammars, Multiple Kernel Learning, Max-Margin Structured Output Learning, Needleman-Wunsch algorithm, Instance-based learning, Case Based Reasoning, and Self Organizing Feature Maps (SOM), Principal Component Analysis, Independent Component Analysis etc. Some type of inductive learning, Evolutionary programming and combinational (or hybrid) models can be used to solve the same problems.

3. SCMs in BIOINFORMATICS

3.1 SCMs applications on Sequence alignment

Sequence alignment is a common task in bioinformatics. It plays an essential role in detecting regions of significant similarity among a collection of primary sequences of nucleic acids or proteins. If they are highly similar, then they have similar 3D structures or share similar functions. Given a family $S = (S_1,..., S_N)$ of N sequences, the problem can formally be represented as a set of sequences, and each sequence has its own length. The characters of sequences are defined over an alphabet Σ including a gap symbol denoted by '–', which is a molecular biology term, indel (insertion or deletion). The indels indicate that some parts of a sequence are inserted or deleted. The sequence is either a DNA, ribonucleic acid (RNA), or amino acid (protein) sequence. The nucleotide bases are adenine (A), cytosine (C), guanine (G), thymine (T), and uracil (U). The alphabet is *{*A, C, G, T*}* and *{*A, C, G, U*}* for DNA and RNA, respectively. The sequence alignment problem has two computational approaches: local alignment and global alignment. Global alignment is used Needleman-Wunch algorithms. Local alignment is used Smith-Waterman algorithms. In global alignment, sequences are aligned as a whole, whereas in local sequence alignment, similarities detected locally between sequences are aligned [50]. Assume that 2 DNA sequences are given as S1 = *{GCTG*AACG*}* and S2 = *{CTATAATC}* with lengths |S1| and |S2|, respectively. This pair of sequences can be aligned as shown in Figure 2.

```
An alignment without gaps: GCTGAACG
                           CTATAATC

An alignment with gaps: GCTGA--A--CG
                        --CT--ATAATC
```

Figure 2. Sequence alignment of 2 DNA sequences

Gap is sequence of g missing characters inserted in a string to achieve alignment. Gaps are assigned with two kinds of negative scores: Gap-open penalty: negative score associated with the initiation of a gap (i.e., with the first missing character), and Gap-extension penalty: negative score associated with each additional missing character.

For reasons of computational complexity, sequence alignment is divided into two categories:

- Pairwise alignment (i.e., the alignment of two sequences).
- Multiple-sequence alignment (i.e., the alignment of three or more sequences).

Pairwise alignment problems have exact solutions by using dynamic programming. Multiple-sequence alignment problems have approximate (heuristic) solutions. The function of sum-of-pairs is the most popular scoring method for evaluation of the quality of the alignment. The goal of general multiple sequence alignment algorithms is to find out the alignment with the highest sum-of-pairs [50]. There are numerous existing methods for sequence alignment. The efficiency of an alignment is assessed by the application of SCMs.

Table 1 summarizes some applications of sequence alignment with their used SCMs chronologically (from 1993 to 2011). In this table, the first column describes the authors and the second column describes the soft computing methods that were used. According to Table 1, we can say that metaheuristic and swarm intelligence algorithms are more useful than the other soft computing algorithms for Sequence alignment.

Table 1. Soft Computing methods applied for sequence alignment problem

Authors	**Used Method**	**Year**	**Ref.**
Ishikawa et al.	Simulated Annealing	1993	51
Kim et al.	Simulated Aannealing	1994	52
Wayama et al.	Genetic Algorithm	1995	53
Notredame and Higgins	Genetic Algorithm	1996	54
Zhang and Wong	Genetic Algorithm	1997	55
Krogh	Hidden Markov Model	1998	56
Chellapilla and Fogel	Evolutionary Programming	1999	57
Maniezzo and Carbonaro	Ant Colony Algorithm	2000	58
Keith et al.	Simulated Annealing	2002	59
Moss and Johnson	Ant Colony Algorithm	2003	60
Shyu et al	Genetic Algorithm	2004	61
Hernandez-Guia	Simulated Annealing	2005	62
Karpenko et al.	Ant Colony Algorithm	2005	63
Horng et al.	Genetic Algorithm	2005	64
Ge and Liang	Hidden Markov Model and Particle Swarm Optimization	2005	65
Omar et al.	Optimization algorithm	2005	66
Chen et al.	Ant Colony Algorithm	2006	67
Rodriguez et al.	Particle Swarm Optimization	2007	68
Lee et al.	Genetic Algorithm and Ant Colony Algorithm	2008	69
Juang and Su	Dynamic Programming and Particle Swarm Optimization	2008	70
Chen et al.	Ant Colony Algorithm	2008	71
Xu and Y. Chen	Particle Swarm Optimization	2009	72
Mikami and J. Shi	Ant Colony Algorithm	2009	73
Chen et al.	Dispersion Graph and Ant Colony Algorithm	2009	84
Lei et al.	Particle Swarm Optimization	2009	75
Bucak and Uslan	Stochastic Optimization	2011	76

3.2. SCMs application on single nucleotide polymorphism problem

Single Nucleotide Polymorphism (SNPs) are simply sequence variations between individuals at a particular point in the genome. Genetic variants mostly consist of SNPs, and human genome is estimated to include around 10 million SNPs [77-78]. Most of these genome-wide association (GWA) studies are aimed to determine genetic variants possibly related to complex diseases. Since SNPs are single-base pair changes, the smallest unit of genetic variation, they are present in very small segments of DNA and are more likely to survive severe environmental degradation than any other form of genetic variation. In this regard, it is generally preferred to use SNPs in GWA studies which are used soft computing methods [79]. The number of individuals and SNPs are quite effective on the statistical significance of a GWA study. However, it is still very expensive and time-consuming to genotype all the SNPs in a large population found in the candidate area for large-scale GWA studies [80].

SNPs found on chromosome set is called a haplotype. High-given methods, each allele are not capable of distinguishing the source chromosome. Position of the two alleles of a SNP is usually only take care of such methods. This is the source of alleles chromosomes identifiable. This combined with the target locus allele genotype is called knowledge. In a healthy state of being of an individual or the individual's phenotype of the patient is called. Figure 3 describes all haplotype, genotype, and phenotype [81].

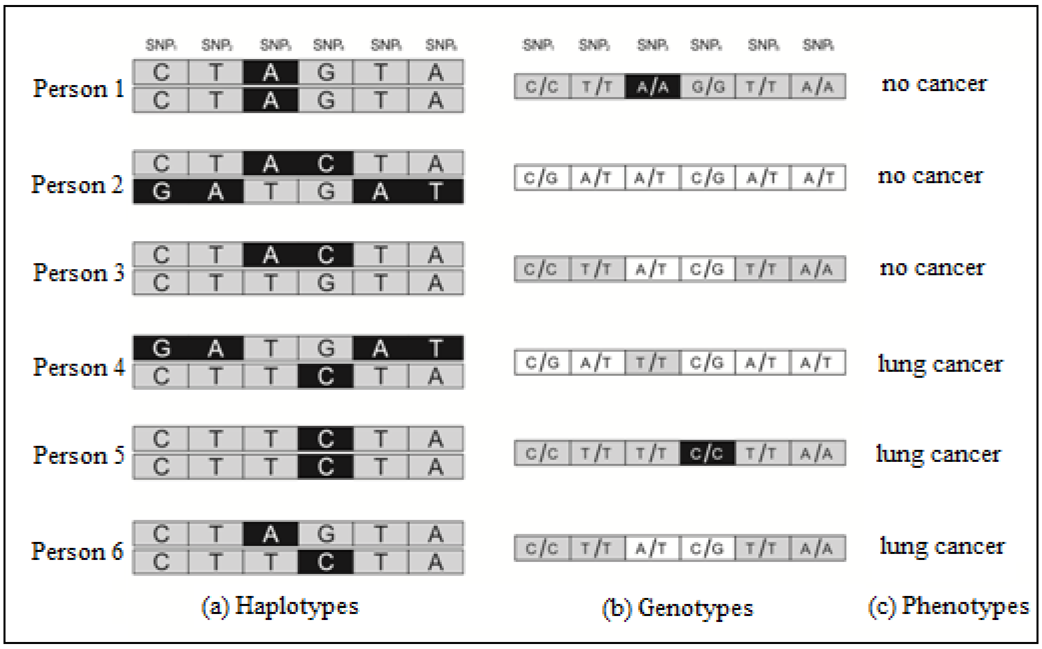

Figure 3. Description of haplotypes, genotypes, and phenotypes

According to the approach used to measure the haplotype tag SNPs knowledge of the methods used for the selection are divided into four groups:

- Methods based on differences in haplotype
- Methods based on correlation coefficient between SNPs
- Methods based on the relationship between phenotype
- Tagged with estimation methods based on SNP

There are different SCMs applied on SNPs in recent years. These methods are based on the fact that human genome can be divided into discrete blocks, and small sets of common haplotypes in each block are shared by a specific population. Table 2 summarizes some applications on SNPs with their used SCMs chronologically.

Table 2. Soft Computing methods applied for single nucleotide polymorphism

Authors	**Used Method**	**Year**	**Ref.**
Tomida et al.	Back-Propagation	2002	82
Ritchie et al.	Back-Propagation and Genetic Algorithm	2003	83
Ao et al.	Hierarchical Clustering and Graph Methods	2004	84
Tomita et al.	Back-Propagation	2004	85
Lin and R. Altman	Principle Component Analysis	2004	86
Lin et al.	Back-Propagation	2006	87
Lee et al.	Bayesian Networks	2006	88
Curtis	Back-Propagation	2007	89
Yang et al.	Particle Swarm Optimization	2008	90
Sun and Kardia	Back-Propagation	2008	91
Yang and Zhang	Back-Propagation and Genetic Algorithm	2008	92
Petrovski et al.	K-Nearest Neighbour	2009	93
Mahdevar et al.	Genetic Algorithm	2010	94
Chuang et al.	Genetic Algorithm and K-Nearest Neighbour	2010	95
Lin and Leu	Particle Swarm Optimization and Support Vector Machines	2010	96
Nahlawi and P. Mousavi	Independent Component Analysis	2010	97
Ilhan and Tezel	Genetic Algorithm, Support Vector Machines	2011	98
Shi et al.	Back-Propagation	2012	99
Karlik and Oztoprak	Back-Propagation	2012	100
Karlik and Oztoprak	*Naïve Bayes*	2012	101
Ilhan and Tezel	Genetic Algorithm, Particle Swarm Optimization, Support Vector Machines	2013	102

According to Table 2, we can say that Artificial Neural Networks (ANN) algorithms are more useful than the other soft computing techniques for single nucleotide polymorphism.

4. CONCLUSIONS

According to this review article we can say that there is a wide area to develop new methods which might share the advantages of SCMs analysis in terms of implementing a parsimonious approach to detect the patterns of multi-marker genotypes which can be observed when an associated susceptibility locus is present. Bioinformatics data needs to be invested in collecting samples of cases and controls and obtaining genotypes it seems sensible to argue that considerable effort should be expended on ensuring that methods of analysis applied to the data obtained are as effective as possible.

I can advice the close collaboration between biologists and bioinformaticians are to make available user-friendly software packages that can be used jointly by researchers with expertise in experimental biology and researchers with expertise in computer science. Because, soft computing methods may not be interpretable by a biologist. For example, a question that requires multiple processors to answer might need the assistance of someone with expertise in parallel computing. To be intuitive to a biologist, the software needs to be easy to use and needs to provide output that is visual and easy to navigate. As a result, we can say that if biologists want to solve bioinformatics problems more easily, they have to collaborate with computer scientist who have experienced on soft computing methods. This collaboration provides to solve complex bioinformatics problems. Future bioinformatics databases and analysis tools that successfully integrate with their collaborations which will prove to be the most useful for biological and biomedical discovery.

5. REFERENCES

1. P. Larranaga, B. Calvo, R. Santana, et al., Machine learning in bioinformatics, *Briefings in Bioinformatics* **7(1)**, 86–112, 2006.
2. R. J. Carter, I. Dubchak, S. R. Holbrook, A computational approach to identify genes for functional RNAs in genomic sequence, *Nucleic Acids Research* **29(19)**, 3928–3938, 2001.
3. C. Mathe, M. F. Sagot, T. Schlex, et al., Current methods of gene prediction, their strengths and weaknesses, *Nucleic Acids Research* **30(19)**, 4103–4117, 2002.
4. S. Aerts, P. Van Loo, Y. Moreau, et al., A genetic algorithm for the detection of new cis-regulatory modules in sets of coregulated genes, *Bioinformatics* **20(12)**, 1974–1976, 2004.
5. J. H. Phan, C. F. Quo, M. D. Wang, Functional genomics and proteomics in the clinical neurosciences: data mining and bioinformatics, *Prog. in Brain Research* **158**, 83–108, 2006.
6. L. J. Lancashire, C. Lemetre, G. R. Ball, An introduction to artificial neural networks in bioinformatics- application to complex microarray and mass spectrometry datasets in cancer studies, *Briefings in Bioinformatics* **10(3),** 315–329, 2009.
7. S. Ma and J. Huang, Penalized feature selection and classification in bioinformatics, *Briefings in Bioinformatics* **9(5),** 392-403, 2008.
8. Bower JM, Bolouri H (eds). *Computational Modeling of Genetic and Biochemical Networks*, MIT Press, March 2004.
9. I. Ö. Bucak and V. Uslan, An analysis of sequence alignment: heuristic algorithms, *Conf. Proc. IEEE Eng Med Biol Soc*., 1824–1827, 2010.
10. V. Uslan and İ. Ö. Bucak, Microarray Image Segmentation Using Clustering Methods, *Mathematical and Computational Applications* **15(2)**, 240–247, 2010.
11. M. Krallinger, R. A. Erhardt, A. Valencia, Text-mining approaches in molecular biology and biomedicine, *Drug Discovery Today* **10(6)**, 439–45, 2005.

12. J. T. L. Wang, M. J. Zaki, H.T.T. Toivonen HTT, et al., *Data Mining in Bioinformatics*, Springer-Verlag, 2004.
13. P. Baldi & S. Brunak, *Bioinformatics. The Machine Learning Approach*, MIT Press, 2001.
14. A. Tettamanzi and M. Tomassini, *Soft Computing:Integrating Evolutionary, Neural, and Fuzzy Systems***,** Springer-Verlag, 2011.
15. P. Baldi P, S. Brunak, Y. Chauvin Y, et al. Assessing the accuracy of prediction algorithms for classification: an overview, *Bioinformatics* **16**, 412–424, 2000.
16. H. White, Learning in Artificial Neural Networks: A statistical perspective, *Neural Computation* **1**, 425-464, 1989.
17. R. P. Lippmann, An Introduction to Computing with Neural Networks, *IEEE Acoustics* **4**, 4-22, 1987.
18- H.R. Öz, B. Karlık, C.A. Evrensel, Application of Artificial Neural Networks Method in Mucus Clearance in Pulmonary Airways, *International Journal of Natural and Engineering Sciences* **3(2),** 28-31, May, 2009.
19. U. Seiffert, L.C. Jain, P. Schweizer, *Bioinformatics Using Computational Intelligence Paradigms*, Springer-Verlag, 2005.
20. O. Karan, C. Bayraktar, H. Gümüşkaya, B. Karlık, Diagnosing Diabetes Using Neural Networks on Small Mobile Devices, *Expert Systems with Applications* **39**, 54-60, 2012.
21. Y. Özbay, R. Pektatlı, B. B. Karlık, A Fuzzy Clustering Neural Network Architecture for Classification of ECG Arrhythmias, *Computers in Biology and Medicine* **36**, 376–388, 2006.
22- B. Karlık and A.V. Olgaç, Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks, International Journal of Artificial Intelligence and Expert Systems **1(4),** 111-122, 2011.
23. C. Bayraktar, B. Karlık, F. Demirezen, Automatic Diagnosis of Otitis Media Diseases Using Wavelet Based Artificial Neural Networks, *1st Inter. Symposium on Computing in Science & Engineering Bioengineering Congress*, 3-5 June, 2010, Kuşadası, Turkey.
24. R. K. Begg, M. Palaniswami, B. Owen, Support vector machines for automated gait classification, *IEEE Transactions on Biomedical Engineering* **52(5),** 828–838, 2005.
25. S. Chandaka, A. Chatterjee, S. Munshi, Cross-correlation aided support vector machine classifier for classification of EEG signals, *Expert Systems with Applications* **36,** 1329–1336, 2009.
26. C. J. C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery **2**, 121–167, 1998.
27. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
28. S. Tasdemir, A. Urkmez, S. Inal. Determination of body measurements on the holstein cows using digital image analysis and estimation of live weight with regression analysis, *Computers and Electronics in Agriculture* **76**, 189–197, 2011.
29. S. Tasdemir, A. Urkmez, S. Inal, A fuzzy rule-based system for predicting the live weight of Holstein cows whose body dimensions were determined by image analysis, *Turkish Journal of Electrical Engineering & Computer Sciences* **19(4),** 689–703, 2011.

30. J.-S. R Jang, C. T. Sun, E. Mizutani, *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, NJ, 1997.
31. J. C. Bezdek, R. Ehrlich, W. Full, FCM: fuzzy c-means algorithm. Computers and Geosciences **10(2–3)**, 191–203, 1984.
32. R.K. De, J. Basak, S.K. Pal, Unsupervised feature extraction using neuro-fuzzy approach, *Fuzzy Sets and Systems* **126**, 277-291, 2002.
33. J. S. R. Jang, C. T. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1997.
34. A. Celikyilmaz and I. B. Turksen, Validation criteria for enhanced fuzzy clustering, *Pattern Recognition Letters* **29**, 97–108, 2007.
35. R.O. Duda, P.H. Hart, D.G. Stork, *Pattern classification and scene analysis*, John Wiley & Son, N.Y, 2012.
36. B. Karlık, Hepatitis Disease Diagnosis Using Backpropagation and the Naïve Bayes Classifiers, *BURCH Journal of Science and Technology* **1(1),** 49–62, 2011.
37. B. Karlık and E. Öztoprak, Personalized Cancer Treatment by Using Naïve Bayes Classifier, *International Journal of Machine Learning and Computing* **2(3),** 339–344, 2012.
38. J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.
39. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, NY, Addison Wesley, 1989.
40. M. Omran, A. Engelbrecht and A. Salman, Differential evolution methods for unsupervised image classification, In Proceedings of the IEEE Congress on Evolutionary Computation (CEC2005) **2**, 966-973, September, 2005.
41. M. G. H. Omran, A. P. Engelbrecht, A. Salman, Particle swarm optimization method for image clustering, *Inter. Journal of Pattern Recognition and Artificial Intelligence* **19(3),** 297–322, 2005.
42. J. Kennedy and R. C. Eberhart, Particle swarm optimization, *Proc. of IEEE International Conference on Neural Networks*, Piscataway, NJ, **4**, 1942–1948, 1995.
43. R. C. Eberhart, A discrete binary version of the particle swarm algorithm, *Proc. of Conference Systems Man Cybernetics*, Piscataway, NJ, 4104–4108, 1997.
44. I. Babaoglu, O. Fındık, et al. A novel hybrid classification method with particle swarm optimization and k-nearest neighbor algorithm for diagnosis of coronary artery disease using exercise stress test data, *International Journal of Innovative Computing, Information and Control* **8(5B),** 1349-4198, May 2012.
45. Y. Shi, *Particle swarm optimization*, IEEE Neural Networks Society, 8-13, 2007.
46. D. Karaboga and C. Ozturk, A novel clustering approach: Artificial Bee Colony (ABC) algorithm, *Applied Soft Computing* **11**, 652–657, 2011.
47. M. Dorigo and L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation* **1(1),** 53–66, Apr. 1997.
48. R. Montemanni, L. M. Gambardella, et al. Ant colony system for a dynamic vehicle routing problem, *Journal of Combinatorial Optimization* **10(4),** 327–343, 2005.
49. D. Martens, M. De Backer, et al. Classification With Ant Colony Optimization, *IEEE Transactions on Evolutionary Computation* **11(5),** 651–665, October 2007.

50. M. S. Rosenberg, *Sequence alignment: methods, models, concepts, and strategies*, University of California Press, 2009.
51. M. Ishikawa, T. Toya, et al. Multiple sequence alignment by parallel simulated annealing, *Computer Applications in the Biosciences* **9**, 267–273, 1993.
52. J. Kim, S. Pramanik, M.J. Chung, Multiple sequence alignment using simulated annealing, *Computer Applications in the Biosciences* **10**, 419–426, 1994.
53. M. Wayama, K. Takahashi, T. Shimizu, An approach to amino acid sequence alignment using a genetic algorithm, *Genome Informatics* **6**, 122–123, 1995.
54. C. Notredame and D.G. Higgins, SAGA: sequence alignment by genetic algorithm, *Nucleic Acids Research* **24**, 1515–1524, 1996.
55. C. Zhang and A.K. Wong, A genetic algorithm for multiple molecular sequence alignment, *Computer Applications in the Biosciences* **13**, 565–581, 1997.
56. A. Krogh, An introduction to hidden markov models for biological sequences, In: S. L. Salzberg, D. B. Searls and S. Kasif. *Computational Methods in Molecular Biology*, Elsevier, 45–63, 1998.
57. K. Chellapilla and G.B. Fogel, Multiple sequence alignment using evolutionary programming, in: *Proc. of the IEEE Congress on Evolutionary Comp*. **1**, 445–452, 1999.
58. V. Maniezzo and A. Carbonaro, An ANTS heuristic for the frequency assignment problem, *Future Generation Computer Systems* **16(8),** 927–935, 2000.
59. J. M. Keith, P. Adams, et al. A simulated annealing algorithm for finding consensus sequences, *Bioinformatics* **18**, 1494–499, 2002.
60. J. D. Moss and C.G. Johnson, An ant colony algorithm for multiple sequence alignment in bioinformatics, *Artificial Neural Networks and Genetic Algorithms* 182–186, Springer, 2003.
61. C. Shyu, L. Sheneman, J. A. Foster, Multiple sequence alignment with evolutionary computation, *Genetic Programming and Evolvable Machines* **5**, 121–144, 2004.
62. M. Hernandez-Guia, R. Mulet, S. Rodriguez-Perez, Simulated annealing algorithm for the multiple sequence alignment problem: the approach of polymers in a random medium, *Physical review. E, Statistical, nonlinear, and soft matter physics* **72,** 031915 (epub), 2005.
63. O. Karpenko, J. Shi, Y. Dai, Prediction of MHC class II binders using the ant colony search strategy, *Artificial Intelligence in Medicine* **35**, 147–156, 2005.
64. J. T. Horng, L.C. Wu, C.M. Lin, B.H. Yang, A genetic algorithm for multiple sequence alignment, *Soft Computing* **9**, 407–420, 2005.
65. H. W. Ge, Y.C. Liang, A hidden Markov model and immune particle swarm optimization-based algorithm for multiple sequence alignment, *Proc. Advances in Artificial Intelligence*, 756–765, 2005.
66. M. F. Omar, R.A. Salam, R. Abdullah, N.A. Rashid, Multiple sequence alignment using optimization algorithms, *International Journal of Computational Intelligence* **1**, 81–89, 2005.
67. Y. Chen, Y. Pan, J. Chen, et al. Multiple Sequence Alignment by Ant Colony Optimization and Divide-and-Conquer, V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, 646 – 653, Springer-Verlag Berlin Heidelberg 2006.
68. P.F. Rodriguez, L.F. Nino, O.M. Alonso, Multiple sequence alignment using swarm intelligence, *International Journal of Computational Intelligence Research*, 2007.

69. Z. Y. Lee, S. F. Su, C. C. Chuang, K. H. Liu, Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment, *Applied Soft Computing* **8**, 55–78, 2008.
70. W. S. Juang and S. F. Su, Multiple sequence alignment using modified dynamic programming and particle swarm optimization, *Journal of the Chinese Institute of Engineers* **31**, 659–673, 2008.
71. W. Chen, B. Liao, W. Zhu, H. Liu, Q. Zeng, An ant colony pairwise alignment based on the dot plots, *Journal of Computational Chemistry* **30**, 93–97, 2008.
72. F. Xu and Y. Chen, A method for multiple sequence alignment based on particle swarm optimization, *5th Int. Conf. on Intelligent Computing* 5755, 965–973, Springer, 2009.
73. A. Mikami and J. Shi, A modified algorithm for sequence alignment using ant colony system, *IPSJ Transactions on Bioinformatics* **2**, 63–73, 2009.
74. W. Chen, B. Liao, W. Zhu, X. Xiang, Multiple sequence alignment algorithm based on a dispersion graph and ant colony algorithm, *Journal of Computational Chemistry* **30**, 2031–2038, 2009.
75. X. J. Lei, J. J. Sun, Q. Z. Ma, Multiple sequence alignment based on chaotic PSO, in: Proc. of the Comp. Intelligence and Intelligent Systems **51**, 351–360, Springer, 2009.
76. I. Ö. Bucak and V. Uslan, Sequence alignment from the perspective of stochastic optimization: a survey, *Turkish Journal of Electrical Engineering & Computer Sciences* **19(1),** 157–173, 2011.
77. L. Kruglyak and D.A. Nickerson, Variation is the spice of life, *Nature Genetics* **27**(3), 234–236, 2001.
78. R. Sachidanandam, D. Weissman, et al. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms, *Nature* **409**, 928–933, 2001.
79. B.V. Halldorsson, V. Bafna, et al. A survey of computational methods for determining haplotyes, *Lecture Notes in Computer Science* **2983**, 26–47, 2004.
80. K. Irizarry, V. Kustanovich, et al., Genome-wide analysis of single-nucleotide polymorphisms in human expressed sequences, *Nature Genetics* **26**, 233–236, 2000.
81. D. Crawford and D.A. Nickerson, Definition and clinical importance of haplotypes, *Annual Review of Medicine* **56(1),** 303–320, 2005.
82. S. Tomida, T. Hanai, et al. Artificial neural network predictive model for allergic disease using single nucleotide polymorphisms data, *Journal of Bioscience and Bioengineering* **93(5),** 470–478, 2002.
83. M.D. Ritchie, B.C. White, et al. Optimization of neural network architecture using genetic programming improves detection and modeling of gene-gene interactions in studies of human diseases, *BMC Bioinformatics* **4**, 28, 2003.
84. S.I. Ao, K. Yip, et al. CLUSTAG: hierarchical clustering and graph methods for selecting tag SNPs, *Bioinformatics* **21(8),** 1735–1736, 2004.
85. Y. Tomita, S. Tomida, et al. Artificial neural network approach for selection of susceptible single nucleotide polymorphisms and construction of prediction model on childhood allergic asthma, *BMC Bioinformatics* **1(5),** 120, 2004
86. Z. Lin and R. Altman, Finding haplotype tagging SNP by use of principle component analysis (PCA), *Am. J. Hum. Genetics* **75(5),** 850–861, 2004.

87. E. Lin, Y. Hwang, et al. An artificial neural network approach to the drug efficacy of interferon treatments, *Pharmacogenomics* **7(7),** 1017–1024, 2006.
88. P.H. Lee and H. Shatkay, BNTagger: improved tagging SNP selection using Bayesian networks, *Bioinformatics* **22(14),** 211–219, 2006.
89. D. Curtis, Comparison of artificial neural network analysis with other multimarker methods for detecting genetic association, *BMC Genetics* **8**, 49, 2007.
90. C. Y. Yang, C. H. Hou, L. Y. Chuang, Improved tag SNP selection using binary particle swarm optimization, *In. Proc. of IEEE congress on evolutionary computation (CEC 2008),* 854–60, 2008.
91. Y. V. Sun and S. L. Kardia, Imputing missing genotypic data of single-nucleotide polymorphisms using neural networks, *European Journal of Human Genetics* **16(4),** 487–495, 2008.
92. P. Yang and Z. Zhang, A Hybrid Approach to Selecting Susceptible Single Nucleotide Polymorphisms for Complex Disease Analysis, in Proc. of Inter. Conf. on BioMedical Engineering and Informatics, 214–218, 2008.
93. S. Petrovski, C.E. Szoeke, et al., Multi-SNP pharmacogenomic classifier is superior to single-SNP models for predicting drug outcome in complex diseases, *Pharmacogenet Genomics* **19(2),** 147–152 2009.
94. G. Mahdevar, J. Zahiri, et al., Tag SNP selection via a genetic algorithm, *Journal of Biomedical Informatics* **43(5),** 800–804, 2010.
95. L.Y. Chuang, C.H. Hou, C.Y. Yang, A novel prediction method for tag SNP selection using genetic algorithm based on KNN, *International Journal of Chemical and Biological Engineering* **3(1),** 12–17, 2010.
96. M.H. Lin and C.L. Leu, A hybrid PSO–SVM approach for haplotype tagging SNP selection problem, *International Journal of Computer Science and Information Security* **8(6),** 60–65, 2010.
97. L.I. Nahlawi and P. Mousavi, Single nucleotide polymorphism selection using independent component analysis, Conf. Proc. IEEE Eng. Med. Biol. Soc., 2010:6186-9, 2010.
98. I. Ilhan, Y.E. Göktepe, S. Kahramanlı, Tag SNP selection using GA–SVM approach, *In. Proc. of the IADIS European conference on data mining* 27–34, 2011.
99. H. Shi, Y. Lu, J. Du, et al. Application of back propagation artificial neural network on genetic variants in adiponectin ADIPOQ, peroxisome proliferator-activated receptor-γ, and retinoid X receptor-α genes and type 2 diabetes risk in a Chinese Han population, *Diabetes Technology & Therapeutics* **14(3),** 293-300, 2012.
100. B. Karlık and E. Öztoprak, ANN Based Application of Pharmacogenetics to Personalized Cancer Treatment, in *Proc. ICMLC 2012,* March, **25**, 2012.
101. B. Karlık and E. Öztoprak, Personalized Cancer Treatment by Using Naïve Bayes Classifier, *International Journal of Machine Learning and Computing* **2(3),** 339-344, 2012.
102. İ. İlhan and G. Tezel, A genetic algorithm–support vector machine method with parameter optimization for selecting the tag SNPs, *Journal of Biomedical Informatics* **46**, 328–340, 2013.