Interdisciplinary Sciences: Computational Life Sciences (2019) 11:444–459
https://doi.org/10.1007/s12539-017-0276-x

ORIGINAL RESEARCH ARTICLE

# Identification of Disease Critical Genes Using Collective Meta-heuristic Approaches: An Application to Preeclampsia

Surama Biswas[1] · Subarna Dutta[1] · Sriyankar Acharyya[1]

Received: 6 January 2017 / Revised: 6 November 2017 / Accepted: 21 November 2017 / Published online: 1 December 2017

**Abstract**
Identifying a small subset of disease critical genes out of a large size of microarray gene expression data is a challenge in computational life sciences. This paper has applied four meta-heuristic algorithms, namely, honey bee mating optimization (HBMO), harmony search (HS), differential evolution (DE) and genetic algorithm (basic version GA) to find disease critical genes of preeclampsia which affects women during gestation. Two hybrid algorithms, namely, HBMO-kNN and HS-kNN have been newly proposed here where kNN (k nearest neighbor classifier) is used for sample classification. Performances of these new approaches have been compared with other two hybrid algorithms, namely, DE-kNN and SGA-kNN. Three datasets of different sizes have been used. In a dataset, the set of genes found common in the output of each algorithm is considered here as disease critical genes. In different datasets, the percentage of classification or classification accuracy of meta-heuristic algorithms varied between 92.46 and 100%. HBMO-kNN has the best performance (99.64–100%) in almost all data sets. DE-kNN secures the second position (99.42–100%). Disease critical genes obtained here match with clinically revealed preeclampsia genes to a large extent.

**Keywords** Gene selections · Harmony search · Honey bee mating optimization · K nearest neighbor · Meta-heuristics · Preeclampsia

## 1 Introduction

Identification of disease critical genes by analyzing gene expression datasets is important in bioinformatics. The number of genes in gene expression dataset may rise above 40,000. On the other hand, a number of genes which may be called disease critical are considered to be limited. Otherwise, close monitoring of the genes and further study for therapeutic intervention may not be conducted easily. Due to its profound impact on maternal and neonatal health all over the world, a disease, named preeclampsia [1] has been focused in this paper. Pregnant women get affected by this disease. As there are heredity issues behind the disease [2–4], identification of the disease-causing genes has gained importance in recent times.

Gene expression datasets are required to accomplish many important problems in bioinformatics. These datasets are obtained through microarray technology [5, 6]. The data presents protein production information of thousands of genes, either in different time steps or in different conditions (patients, disease type, tissues) in organisms. Gene expression data are obtained from different public repositories (Gene Expression Omnibus, ArrayExpress). These data are undergone pre-processing stages such as normalization [7, 8] dillies. The final data is in the form of a matrix. The rows of the data matrix are denoted by genes and the columns (often termed as samples) are denoted by either time steps or biological conditions. Depending on the types of column, the data set is known either as time series or as steady state gene expression data, respectively. Each matrix element is a floating point number, mapped in a certain range depending on the type of normalization. Analysis of gene expression

✉ Surama Biswas
surama.biswas@gmail.com

Subarna Dutta
dutta.subarna88@gmail.com

Sriyankar Acharyya
srikalpa8@gmail.com

1 Department of Computer Science and Engineering, Maulana Abul Kalam Azad University of Technology, West Bengal (MAKAUT, WB), BF-142, Sector-I, Salt Lake, Kolkata, West Bengal 700064, India

data provides insight into the normal as well as disease-related cell functioning in organisms.

In preeclampsia, some genes are likely to be differentially expressed in the tissues of an organ named placenta. So, for gene expression analysis of preeclampsia, data are collected from the placenta of different patients (normal/diseased). Using this data, meta-analysis had been performed [9–11] to obtain the differentially expressed genes of preeclampsia [12].

In recent times, meta-heuristic algorithms have wide applications in the domain of bioinformatics [13]. Many meta-heuristics have been successfully applied in the areas of reconstruction of gene regulatory network [14], clustering or bi-clustering [15] and genome-wide association studies to detect effective epistatic interactions among genes [16–18].

Gene selection is usually performed using sample classification. On multi-class samples of gene expression data, various classification techniques have been employed to determine disease class [19, 20]. Different paradigms of data science, mathematical modeling, statistical schemes and computational theories have been applied to this field. Widely used data classifiers such as k nearest neighbor (kNN) [21] and support vector machine (SVM) [22] have been used in sample/disease classification [23]. Meta-heuristic algorithms such as genetic algorithms (GA) [24], particle swarm optimization (PSO) [25] have been used to optimize gene selection. In a well-structured algorithm, Li et al. (2001) [26] proposed a combination of GA and kNN (referred the algorithm as GA-kNN) to perform gene selection by sample classification [27]. Li et al. (2001) [26] also showed a direction, how using statistical analysis, genes may be ranked according to their impact in causing a disease. Thereafter, numerous algorithms [28–38] have been proposed to enrich the field of gene selection by sample classification. For prioritization of disease critical genes of preeclampsia, Tejera et al. (2013) performed co-expression network analysis [39] and compared their results with two widely used state-of-the-art classification methods, namely, GANN (combination of GA and a classifier named nearest neighbor) and GADA (combination of GA and a classifier named discriminant analysis). Very recently, the meta-heuristic algorithms such as variable neighborhood search (VNS) [40], differential evolution (DE) [41], simulated annealing (SA) [42] and PSO have been combined with kNN to form VNS-kNN, DE-kNN, SA-kNN and PSO-kNN, respectively. These algorithms have been used to identify preeclampsia genes [43, 44].

Here, in this paper, methodologies employing meta-heuristics, sample classification and statistical analysis have been developed for the selection of the disease critical genes. Four meta-heuristic algorithms, namely, honey bee mating optimization (HBMO) [45], harmony search (HS) [46], DE and GA have been used. Classifier kNN is embedded in each algorithm for sample classification. Two hybrid algorithms, HBMO-kNN, and HS-kNN have been newly proposed here. Other two algorithms, namely, DE-kNN [43] and SGA-kNN (a modified version of GA-kNN [26]) have also been implemented to present a comparative study of four hybrid algorithms. From an algorithm (GA-kNN) proposed by Li et al. (2001) [26], Li_01 we have taken many ideas that have been used here. A simple traditional version of GA [47] has been incorporated in SGA-kNN. In this research, three preeclampsia datasets were collected for experiments. In an experiment, all (four) hybrid algorithms have been executed and performances of the algorithms have been compared. In an experiment, during execution of each algorithm, candidate solutions having classification accuracy greater than 90% have been saved. Later on, from each dataset, the genes commonly returned by all algorithms have been termed as critical genes. In different experiments, the classification accuracies of algorithms varied between 92.46 and 100%. Among all algorithms, HBMO-kNN was found best in classification accuracy (99.64–100%). DE-kNN secures second place in classification accuracy (99.42–100%). Numbers of genes, common in the output of each algorithm from three datasets are 87, 73 and 74, respectively. Widely known preeclampsia genes, STS, EPHX1, LEP, LRP8, ADD1, INSR have been detected here. The algorithms, HBMO-kNN and HS-kNN are newly proposed in this paper and the statistical approach involving different meta-heuristics is also newly planned and found to be reliable.

Rest of the paper has been organized as follows: Sect. 2 discusses the formulation of gene selection problem as an optimization problem. The algorithm kNN-fit (to achieve fitness of a candidate solution, using classifier kNN) and the four meta-heuristic based algorithms for gene selection have been discussed in Sect. 3. Discussion regarding data and experimental results are presented in Sect. 4. Section 5 concludes the paper giving some future scopes of research.

## 2 Gene Selection by Sample Classification

The input taken is a microarray gene expression dataset (say matrix $X$). Generally, the total number of genes present in $X$ is above 40,000. The objective of this work is to identify a very small subset (containing 30 genes) of genes in $X$, so that the subset collectively discriminate between normal and preeclampsia affected samples. The search involves ${}^{n}C_{d}$ possibilities, where, $n$ is the number of genes in the dataset and $d$ is the number of genes in the subset. So, optimization techniques are applied to identify the subset of critical genes in the huge and diversified search space.

## 2.1 Input Dataset X

The structure of gene expression dataset $X$, on which the study is based, is provided in Table 1. Here, as in standard form, gene $G_i$ is $i$th row identifier. The columns contain normal samples ($N_l$ stands for $l$th sample) and diseased samples ($P_o$ stands for $o$th sample). Here, $i \in \{1, 2, ..., n\}$, $l \in \{1, 2, ..., maxN\}$ and $o \in \{1, 2, ..., maxP\}$, where $n$ is the number of genes, $maxN$ is the number of normal samples and $maxP$ is the number of preeclampsia affected samples in data set $X$. Each entry is the normalized expression level $g_{i,j}^{C}$ of gene $G_i$ at sample $j$ (class $C$ of sample $j$ may be either normal or diseased, $j \in \{1, 2, ..., m\}$ where $m = maxN + maxP$).

## 2.2 Expression Sub-matrix $x_S$

A candidate solution $S$ contains $d$ non-repetitive gene indexes chosen randomly from $X_{index}$. Here, $X_{index} = (1, 2, \ldots, n)$ and $n$ is the number of genes in the dataset. A matrix $x_S$, $x_S \subset X$ has been obtained from $X$ in the following way. For each gene index in $S$, the corresponding row of $X$ has been copied and appended in $x_S$. The appearance of candidate solution $S$, gene expression matrix $X$ and expression sub-matrix $x_S$ are shown in Fig. 1. Here, a small example has been shown where number of rows (genes) in gene expression matrix $X$ is 13. Each row is denoted as $G_1, G_2, \ldots, G_{13}$. Number of elements ($d$) in candidate solution $S$ is 4. Each randomly chosen index $i$, ($i \in \{1, 2, 3, \ldots, 13\}$) in $S$, points to the row $G_i$ in $X$. Each such row is appended in $x_S$.

**Table 1** Structure of microarray gene expression dataset

| Gene | Normal samples | | | | Preeclampsia samples | | | |
|---|---|---|---|---|---|---|---|---|
| | $N_1$ | $N_2$ | … | $N_{maxN}$ | $P_1$ | $P_2$ | … | $P_{maxP}$ |
| $G_1$ | $g_{1,1}^{N}$ | $g_{1,2}^{N}$ | … | $g_{1,maxN}^{N}$ | $g_{1,1}^{P}$ | $g_{1,2}^{P}$ | … | $g_{1,maxP}^{P}$ |
| $G_2$ | $g_{2,1}^{N}$ | $g_{2,2}^{N}$ | … | $g_{2,maxN}^{N}$ | $g_{2,1}^{P}$ | $g_{2,2}^{P}$ | … | $g_{2,maxP}^{P}$ |
| $G_3$ | $g_{3,1}^{N}$ | $g_{3,2}^{N}$ | … | $g_{3,maxN}^{N}$ | $g_{3,1}^{P}$ | $g_{3,2}^{P}$ | … | $g_{3,maxP}^{P}$ |
| • | | | … | | | | … | |
| • | | | … | | | | … | |
| $G_n$ | $g_{n,1}^{N}$ | $g_{n,2}^{N}$ | | $g_{n,maxN}^{N}$ | $g_{n,1}^{P}$ | $g_{n,2}^{P}$ | … | $g_{n,maxP}^{P}$ |

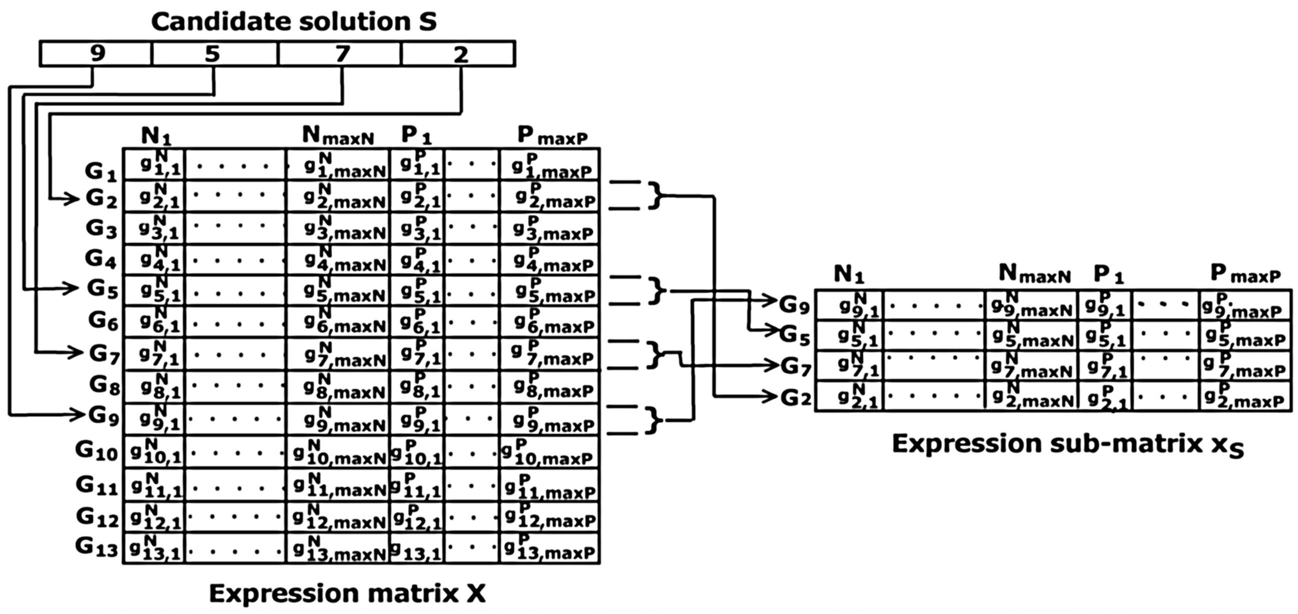

**Fig. 1** An example of deriving $x_S$ from $S$ and $X$

### 2.3 Sample Classification of $x_S$

Each sample in an expression sub-matrix $x_S$ is classified using kNN. If the computed class of a sample in $x_S$ conforms to its original class in $X$, the sample is termed as properly classified sample. Gene selection incorporates those genes in $S$, which maximizes the number of properly classified samples in $x_S$. The number of samples in $x_S$ is same as the number of samples in $X$. The number of rows equals the number of elements in candidate solution $S$. Here, classification accuracy ($CA$) is used to specify the quality of candidate solution $S$ and it has been calculated by Eq. 1,

$$\mathrm{CA} = \frac{\text{Number of properly classified samples in } x_S}{\text{Total number of samples in } X} \times 100\%. \tag{1}$$

## 3 Methods

Meta-heuristic algorithms are used here for gene selection. A candidate solution of length $d$ represents the collection of differentially expressed disease critical genes. Each element is a gene index, selected from gene index vector $X_{index}$. To calculate the fitness of candidate solutions, the classifier kNN [48, 49] is used. Use of kNN in sample classification is explained in Sect. 3.1. Sect. 3.2 contains meta-heuristic-based algorithms for gene selection. Uses of meta-heuristic algorithms, HBMO and HS have been discussed in details. Brief descriptions of the other two algorithms, namely DE and GA are also provided.

### 3.1 k Nearest Neighbor (kNN)

Cross-validation [50] is a kind of model validation, used to assess how far the set of samples in a data set are classifiable. Leave one out (LOO), a type of cross-validation is applied in this research to classify the samples in $x_S$. Here each sample has two classes, normal and diseased. Generally, in cross-validation, the dataset is partitioned into two disjoint sets, namely test-set and train-set. In LOO scheme, $m$ validation tests are performed in a dataset having $m$ samples. In other words, to test the class (normal/disease) of each sample, a validation test is made. In each validation test, the sample to be classified is kept in test-set and remaining $(m-1)$ samples are kept in train-set.

For each of $m$ validation tests, the classifier kNN is used. kNN determines the class of the sample in test-set. In kNN, the distances (here Euclidian Distance) between the sample in test-set and each of $(m-1)$ samples in train-set are computed. The samples in train set are ranked in ascending order by their distance from the test sample. Top $k$ (where $k=3$) samples are taken and named as nearest neighbors of the test sample. Among the nearest neighbors, if the majority (two out of three) of samples have the same class as the test sample, the test sample is considered as properly classified.

If the test sample is properly classified, a score of 1 is assigned to it. A score of 0 is assigned otherwise. The summation of scores in these $m$ validation attempts, each on one sample, gives the total number of properly classified samples in $x_S$. As the objective function for the search is to maximize the number of properly classified samples in $x_S$, the summation of scores gives the fitness value of candidate solution. Description of kNN used for fitness calculation of candidate solution $S$ in gene selection [*kNN-fit* $(x_S)$] is given in Algorithm 1.

The expression sub-matrix $x_S$ is the input to *kNN-fit* $(x_S)$. Initially, all samples in $x_S$ are kept in an empty set *All*. Each sample contains the expression levels of all genes, present in a candidate solution $S$. For each sample in *All*, a cross-validation test is performed. In a cross-validation test, the current sample $i$ is placed in the test set (*Test*), and the training set (*Train*) contains all other samples (in *All*) excluding $i$. For each sample $j \in$ *Train*, the distance of $j$ from $i$ is calculated. Here sample *i, j* and distance *Dist_j_i* is calculated by Eq. (2), Eq. (3) and Eq. (4), respectively,

$$i = \left[g^{C_i}_{1,i\_ind},\ g^{C_i}_{2,i\_ind},\ g^{C_i}_{3,i\_ind}, \dots\ g^{C_i}_{d,i\_ind}\right], \tag{2}$$

$$j = \left[g^{C_j}_{1,j\_ind},\ g^{C_j}_{2,j\_ind},\ g^{C_j}_{3,j\_ind},\ \dots,\ g^{C_j}_{d,j\_ind}\right], \tag{3}$$

$$Dist\_j\_i = \sqrt{\left(g^{C_j}_{1,j\_ind} - g^{C_i}_{1,i\_ind}\right)^2 + \left(g^{C_j}_{2,j\_ind} - g^{C_i}_{2,i\_ind}\right)^2 + \dots + \left(g^{C_j}_{d,j\_ind} - g^{C_i}_{d,i\_ind}\right)^2}. \tag{4}$$

**Algorithm 1.** *kNN-fit( $x_S$ )*

```
Fit_S = 0;
All = {$N_1$, $N_2$, …, $N_{maxN}$, $P_1$, $P_2$, . . . $P_{maxP}$};
for i = 1 to m do  /*for each sample i in All*/
    Test = {i}; Train = All - Test;
    for j = 1 to (m-1) do
        /*for each sample j in  Train*/
        Calculate Euclidian distance of j form i;
    end for
    Sort samples of Train in ascending order
        depending on distance from i;
    Copy k (=3) top ranked samples from sorted
        Train to Neighbor_i;
    Get class $C_i$ ($C_i$ ∈{normal, diseased}) of i;
    for j = 1 to k do
        /*for each sample j in Neighbor_i*/
        Get class $C_j$ ($C_j$ ∈ {normal, diseased}) of
            j;
    end for
    if class of majority of $C_j$  matches with $C_i$
        then  score_i = 1;
    else score_i = 0;
    Fit_S = Fit_S + score_i;
end for
return Fit_S;
```

Here, $C_i$ and $C_j$ are the classes belonging to sample *i* and *j*, respectively. *i_ind* and *j_ind* are the indexes of samples *i* and *j* in $x_S$, respectively. On the basis of their calculated distance using Eq. (4), all *j* ∈ *Train* are sorted in ascending order. *k* (=3) samples with least distance from *i* are kept in a set, *Neighbor_i*. The class of sample *i* ($C_i$) is obtained as follows: if *i_ind* (index of *i* in $x_S$) is less than or equal to *maxN*, $C_i$ is called 'normal'. Otherwise, $C_i$ is 'diseased'. If the class $C_j$ of the most of the samples (*j's*) in *Neighbor_i* matches with $C_i$, *i* is said to be properly classified. If a sample *i* is properly classified, a variable (*score_i*) is set to 1. Otherwise, 0 is assigned to it. *Fit_S* is obtained by taking summation of the individual scores (*score_i*). *Fit_S* is returned as the fitness of current solution *S*.

## 3.2 Meta-heuristics in Gene Selection

In gene selection, a state space search is conducted to obtain the solution. In a microarray gene expression dataset, the number of genes may be of the order of 40,000. In gene selection problem, only 30 genes are to be chosen. So, the search space contains ${}^{40000}C_{30}$ possibilities. When an optimization problem pertains a large state space or it has high complexities (NP-Hard or NP-complete), an advanced class of algorithm, called meta-heuristic algorithm is applied to get a near-optimal solution in reasonable time. To optimize gene selection, four meta-heuristic algorithms, namely, HBMO, HS, DE and GA (traditional version) have been used. Two new hybrid algorithms, namely, HBMO-kNN and HS-kNN have been proposed here. The other two algorithms, namely, DE-kNN and SGA-kNN, (SGA—traditional version of GA [47]) have been implemented to match the current framework. Short description of different meta-heuristic algorithms and how they have been used in gene selection are described below:

### 3.2.1 Honey Bee Mating Optimization (HBMO)

Like in other population-based meta-heuristics, HBMO [51] uses a population of candidate solutions to explore the search space. In HBMO, the behavior of honey bees during mating is imitated to perform the state space search. A population of queens (candidate solutions), each named $Q_i$ has been initialized randomly. $Q_i$ is associated with a repository $Sp_i$ where she can store *D* drones (matrices, similar in structure as $Q_i$). *W* worker bees are initialized. Each worker bee $w_i$ is a single element variable, initialized randomly. Workers are set to improve queens using local heuristics as follows: A queen $Q_i$ and a worker $w_i$ are chosen randomly. The Queen is copied into a variable named *temp*. A random element in *temp* is selected. The element is replaced by the value stored in $w_i$. If the fitness of *temp* is greater than $Q_i$, $Q_i$ is replaced by *temp*. The generation-based loop starts and continues until its termination condition is met. This termination criterion in HBMO is either iteration count reaches *maxIter* or highest fitness of a queen reaches to a certain value. In a generation, the improved queens are evaluated and ranked first. Each ranked queen is undergone three main steps: (1) choosing its mates by a process called maiden flight (2) crossover and (3) mutation.

In a maiden flight, each queen $Q_i$ is undergone the following process: at energy state $t = 0$, $Q_i$ is assigned with initial energy $e_i(0)$ and initial speed $sd_i(0)$. Energy $e_i(0)$ of $Q_i$ is reduced in steps. At each energy state $e_i(t)$, queen $Q_i$ aims to fill its repository $Sp_i$ with a drone $dr_i^t$ [$dr_i^t$: drone of queen $Q_i$ produced at her energy state $e_i(t)$]. $dr_i^t$ is a vector having the same dimension as $Q_i$ and takes values in the same range specified for $Q_i$. The fitness of $dr_i^t$ is calculated. If the fitness is greater than that of $Q_i$ and number of drones in $Sp_i$ is less than $D$, $dr_i^t$ is added to $Sp_i$. If $dr_i^t$ is less fit than $Q_i$, it is given a chance to be added to $Sp_i$ depending on a small probability ($e^{-del/sd_i(t)}$). Here $del$ is the difference between the fitness of $Q_i$ and$dr_i^t$. $sd_i(t)$ is the speed of $i$th queen at $t$th energy state. Speed and energy of $Q_i$ are reduced by Eq. (5) and Eq. (6), respectively,

$$e_i(t+1) = e_i(t) - \gamma, \tag{5}$$

$$sd_i(t+1) = sd_i(t) * \alpha. \tag{6}$$

Here, $e_i(t)$ is the energy of $Q_i$ at step $t$. $\alpha$ and $\gamma$ are parameters of HBMO.

After all the queens fill their repositories, recombination (crossover) starts. Each queen is recombined with each of the drones present in its repository $Sp_i$, producing 2*$D$ offspring. Here, the crossover is not probability dependent. After crossover, each offspring is mutated with a low mutation probability Pm.

The offspring produced from all queens are evaluated and ranked. $B$ offspring from the top are accepted as broods. These broods are improved by workers using local heuristics, almost in the same way as were done in the case of queens. The only difference is that in queen improvement, a queen was chosen at random but in brood improvement, broods are sequentially chosen for improvement. While there is a queen worse than any brood in brood list, the worst among the queens is replaced by the best brood. When the replacement of a queen by a brood occurs, the brood is deleted from brood list.

**Algorithm 2. HBMO-kNN**

```
Initialize population size Num, number of workers
   W, number of drones D, number of broods B,
   energy reduction parameter $\gamma$ and speed
   reduction parameter $\alpha$;
for i = 1 to W do
   Initialize randomly a worker $w_i$, $w_i \in \bar{X}_{index}$;
   /*$\bar{X}_{index} = (1, 2, \ldots, n,)$,*/
end for
for i = 1 to Num do
   Initialize a queen $Q_i$ as a d dimensional vector
      with each element randomly chosen from $\bar{X}_{index}$;
   Create expression sub-matrix $x_S$ based on $Q_i$;
   Evaluate $Q_i$ using kNN-fit($x_S$);
end for
Rank queens and save the best queen as q*;
while termination criteria not met do
   Improve each queen by workers using local
      heuristics and rank;
   for i = 1 to Num do /*Apply on each queen $Q_i$*/
      Create expression sub-matrix $x_S$ based on $Q_i$;
      Evaluate each queen using kNN-fit($x_S$);
      Create $Sp_i$;
      Initialize two random real values r1 and
         r2,(0.5 ≤ r1 ≤ 1, 14 ≤ r2 ≤ 15);
      $e_i(0)$ = r1; $sd_i(0)$ = r2; t = 1;
      while ($e_i(t)$ > 0) do
         Produce a drone $dr_i^t$in the same way as a
            queen was produced;
            /*$dr_i^t$: A drone produced to mate queen
               $Q_i$ at her energy step $e_i(t)$*/
         Create expression sub-matrix $x_S$ based on
            $dr_i^t$;
         Evaluate $dr_i^t$ using kNN-fit($x_S$);
            /*|$Sp_i$|: cardinality of $Sp_i$*/
         if ((fitness($dr_i^t$) > fitness($Q_i$))
            AND (|$Sp_i$|≤ D))then $Sp_i = Sp_i \cup dr_i^t$;
         else
            Generate a random variable r3,
               0≤r3≤1;
            if ($r3 < e^{-del/sd_i(t)}$) AND (|$Sp_i$| ≤ D))
               then $Sp_i = Sp_i \cup dr_i^t$;
         end if
         $e_i(t+1) = e_i(t) - \gamma$; $sd_i(t+1) = sd_i(t) * \alpha$;
            /*Reduce speed and energy*/
         t = t+1;
      end while
      Crossover: Performed between queen i and
         each drone $dr_i^t \in Sp_i$, resulting 2*D
         offspring;
      In each crossover it is checked no duplicate
         gene occurs in offspring;
      With low probability Pm, mutate some broods;
   end for
   for  i = 1 to (2*D) do /*for each offspring i
      produced by a queen of current population*/
      Create expression sub-matrix $x_S$ based on i;
      Evaluate i using kNN-fit($x_S$);
   end for
   From offspring produced by all queens, select
      top B broods;
   Improve the broods by workers using local
      heuristics;
   for i = 1 to B do /*for each brood i*/
      Create expression sub-matrix $x_S$ based on i;
      Evaluate i using kNN-fit($x_S$);
   end for
   while best brood is more fit than worst queen do
      Replace worst queen by best brood;
      Remove best brood from brood list;
   end while
   Create expression sub-matrix $x_S$ based on $Q_i$;
   Evaluate each queen using kNN-fit($x_S$);
   Save best queen in q*;
end while
return q*;
```

Based on HBMO, a hybrid algorithm called HBMO-kNN has been proposed here for gene selection. Description of this algorithm is provided in Algorithm 2. Each of queen $Q_i$ and drone $dr_i^t$ is a vector having $d$ elements. Each element is a gene index chosen randomly from gene index vector $X_{index}$. Each worker is a single element variable. The value of it is a gene index, randomly chosen from $X_{index}$. Energy $e_i(0)$ and speed $sd_i(0)$ of queen $Q_i$ at step $t = 0$ is initialized randomly within range (0.5–1) and (14–15), respectively. Evaluation of a solution $S$ (queen/drone/offspring/brood), is a two-step process. In the first step, a sub-matrix of $X$, named $x_S$ is prepared, as described using an example in Fig. 1. In the second step, *kNN-fit* ($x_S$) is called with $x_S$ as an argument. Here an attempt is made to classify (which class it belongs to—normal or diseased) each sample $i$ in $x_S$. Number of properly classified samples gives the fitness of the solution. During iterations, global best solution $q^*$ is updated which is returned as output at the end.

The efficiency of a meta-heuristic algorithm mainly depends on the judicious mixture of two strategies—exploration (diversification) and exploitation (intensification). Though HBMO is based on the same theory of genetic evolution (using crossover, mutation and selection) such as genetic algorithm, it incorporates some additional measures of exploration and exploitation. For more exploration, an annealing function $\left(e^{-del/sd_i(t)}\right)$ allows to choose a worse drone by a queen for mating. Improvements of queens and broods by workers and replacement of worse queens by better broods are some of the measures to support exploitation. Also, some problem-specific changes to HBMO, especially, in initialization of speed (originally it was in the range 0.5–1, here 14–15) of the queen has been implemented. Aggregation of all these efforts in HBMO helps it find disease-causing genes more efficiently than the state-of-the-art algorithm GA-kNN.

### 3.2.2 Harmony Search (HS)

Harmony search [52] is one of the current nature-inspired meta-heuristic algorithms. The natural phenomenon behind this algorithm is the harmony improvisation process of a musician. Initially, some harmony memory is there in the musician's mind to play a certain piece of music, which he wants to make more accurate by enormous practices. In every practice, he improvises a new harmony and estimates its quality and according to the quality accepts or discards that harmony. If it gets accepted, the musician's memory is updated. Search for the state where esthetic estimation is highest leads the musician to improve. In practice, instead of one musician, there may be a number of musicians with different musical instruments who are going to perform together. So they have to come to a state at which combination of the harmonies improvised from these instruments provide good esthetic estimation to the listeners.

The algorithm for *HS* [53] needs to initialize a harmony memory (*HM*), the initial population, with some random values $hm_{i,j}$, $1 \leqslant i \leqslant HMS$ and $1 \leqslant j \leqslant d$; where, $hm_{i,j}$ is an element of *HM*, *HMS* is the size of the population and $d$ is the number of decision variables. Now we have to improvise a new harmony, a new candidate solution $S$. An element of $S$, $s_j$, is chosen in the following way: if a value rand(0,1) is less than harmony memory consideration rate (*HCMR*), the element $s_j$ under each decision variable is chosen randomly from $\overline{hm_j}$, the domain of $j$th decision variable $\left(hm_{1,j}, hm_{HMS,j}\right)$, i.e., the set of consecutive columns of *HM*. Otherwise, the value of $s_j$ is chosen randomly from $X_{index}$. The parameter *HMCR* lies between 0 and 1. If $s_j$ is selected from $\overline{hm_j}$, it implies the intensification of this meta-heuristic. Further $s_j$ is modified as follows: if a value rand(0,1) is less than pitch adjustment rate (*PAR*), a parameter of HS, then $s_j$ is modified by Eq. (7). Though Eq. (7) returns a float value depending on the variance of previous population, that float value has been converted here to an integer value to reflect proper characteristics of a decision variable (gene index) of gene selection problem. Moreover, the equation makes a small change to current gene index,

$$s_j = \text{int}\left(s_j \pm \text{rand}(0,1) * \text{BW}\right), \tag{7}$$

where BW is called band width. This step implies the diversification of this meta-heuristic. Value of BW is calculated by Eq. (8),

$$\text{BW} = k1 \times \sqrt{\text{Var}\left(\overline{hm_j}\right)}, \tag{8}$$

where $k1$ is the proportionality constant. BW is proportional to the standard deviation of the current population, which increases the exploratory power of HS [53]. If the fitness value of that new candidate solution $S$ is better than the fitness value of the worst candidate solution in the *HM*, then the new replaces the worst. We have to continue this up to the number of iterations (*NI*). We can describe this *NI* parameter as the number of generation or the terminating criteria of the outer loop.

**Algorithm 3. HS-kNN**

```
Initialize the parameters - Harmony Memory Size
    HMS, Harmony Memory Consideration Rate HMCR,
    Pitch Adjustment Rate PAR, bandwidth BW,
    proportionality constant k1;
for i = 1 to HMS do
    Initialize a harmony $HM_i$ of dimension d with
        randomly selected gene indexes $hm_{i,j} \in \bar{X}_{index}$,
        where $1 \le j \le d$;
    Create expression sub-matrix $x_s$ for $HM_i$;
    Calculate the fitness value of $x_S$, based on $HM_i$
        using kNN-fit($x_S$);
end for
Calculate the average fitness value $avg_{HM}$, and max
    fitness value $max_{HM}$;
Save global best harmony $hm^* = HM_{max}$;
/*$HM_{max}$ is a harmony having fitness value $max_{HM}$*/
while termination criteria not met do
    /*Iterate NI times*/
    for i = 1 to HMS do /*for each harmony $HM_i \in HM$
        create a new candidate solution S*/
        for j = 1 to d do /* j = decision variable*/
            if(rand(0,1) < HMCR ) then
                $s_j = hm_{new}$, $hm_{new} \in (hm_{1,j}, hm_{HMS,j})$, chosen
                    randomly; /*$s_j$, an element of S*/
                if(rand(0,1) < PAR) then
                    $BW = k1 * \sqrt{Var(\overline{hm_j})}$,
                    $\overline{hm_j} = \{hm_{1,j}, hm_{2,j}, .., hm_{HMS,j}\}$;
                    $s_j = int(s_j \pm rand(0,1) * BW)$;
                end if
            else $s_j = hm_{new}$, $hm_{new} \in (hm_{lower,j}, hm_{upper,j})$;
                /*$\{hm_{lower,j}, ...., hm_{upper,j}\} = \bar{X}_{index}$*/
            end if
        end for
        Create an expression sub-matrix $x_S$ based on
            new S;
        Evaluate the fitness S using kNN-fit($x_S$);
        $newHM_i = S$;
    end for
    Calculate the average fitness value $avg_{newHM}$, and
        maximum fitness value $max_{newHM}$ of newHM;
    if($avg_{newHM} \ge avg_{HM}$ or $max_{newHM} \ge max_{HM}$) then
        Replace HM by newHM;
        $hm^* = HM_{max}$;
    end if
end while
return $hm^*$;
```

Based on HS, an algorithm for gene selection has been proposed here, namely, HS-kNN (Algorithm 3). One harmony in harmony memory *HM* represents one candidate solution *S*, containing *d* genes, randomly selected from gene index vector $X_{index}$. To evaluate this candidate solution *S*, an expression sub-matrix $x_s$ is created. Then $x_s$ is evaluated by *kNN-fit* ($x_S$) and the number of properly classified samples of $x_s$ is given as fitness value of *S*. Replacing only the worst candidate solution by the new one delays the rate of convergence. So we have modified this as follows: instead of replacing the worst candidate solution only, whole population is replaced by a new *HM*, say *newHM*. Each candidate solution *S*, of *newHM* is formed by the process of creating a new harmony, $newHM_i$. The population is taken, if the either the average fitness value or the best fitness value of the new population is better than or equal to either the average fitness value or the best fitness value of the current population, respectively. Best fitness value of the population, i.e., the global best candidate solution is stored in *hm** in every iteration.

The effectiveness of HS as an evolutionary meta-heuristic search is based on the natural phenomenon of how musicians use their short-term memory (here as harmony memory) to create notes resulting in a pleasant harmony. The process of selecting a new harmony from the previous harmony memory (*HM*) corresponds to the exploitation of the search process as the candidates of *HM* are better in fitness. On the other hand, changing any note of the new harmony randomly, or depending upon some function of the past memory is the exploration part. In HS-kNN, the exploration capability of the search is further enhanced by the incorporation of the standard deviation in the expression of bandwidth $\left(BW = k1 * \sqrt{Var\left(\overline{hm_j}\right)}\right)$ of the previous harmony memory which is subsequently used in modification of candidate solution. Moreover, here, a new harmony memory (a set of new harmonies) has been taken instead of taking single new harmony for replacement. Here, the average and the maximum fitness of the new *HM* have been compared with the average and the maximum fitness of current *HM*, respectively. If the new *HM* is better than the current *HM* in any of the two measures (average and maximum), it will replace the current *HM*. This modification enhances the exploitation of the search process resulting in faster convergence without degrading the exploration capability of the search. It can be concluded that, here, the balance between exploration and exploitation is done more effectively. This way of finding the new set of disease critical genes is found more accurate than the state-of-the-art algorithm GA/kNN.

### 3.2.3 Differential Evolution (DE)

DE [54] is a population-based meta-heuristics, used for optimizing real-valued stochastic function. Recently, in a hybrid algorithm named DE-kNN [43], DE and a classifier called kNN have been combined and used for gene selection. Here, DE-kNN is implemented changing parameter values and used in a comparative study for gene selection. In this algorithm, each candidate solution, an agent $A_i$ is a vector having *d* elements. Each element $a_{ij} (j \in \{1, 2, \dots, d\})$ of $A_i$ is a gene index, initially chosen randomly from gene index vector $X_{index}$. In each generation, for each candidate solution $A_i$, a mutation vector $M_i$ is obtained using a formula involving three other candidate solutions of current population and a parameter called differential weight, *F*. A trial vector $T_i$ is obtained using crossover (with probability Cr) between

$M_i$ and $A_i$. Each element of $T_i$ is a non-repeating gene index from $X_{index}$. To calculate the fitness of a candidate solution $S$ ($A_i$ or $T_i$), expression sub-matrix $x_S$ is created based on $S$ and $X$. The function *kNN-fit* ($x_S$) is used to calculate the fitness of a candidate solution $S$. The best solution (depending on fitness) in final population is returned.

### 3.2.4 Genetic Algorithm (GA)

GA [47, 55] is a population-based meta-heuristics which imitate genetic evolution in organisms. In SGA-kNN, simple/traditional version of GA is combined with kNN. GA is a population-based meta-heuristics. A number (*Num*) of candidate solutions, each called chromosome $C_i$ is initialized with a gene index randomly chosen from $X_{index}$, such that each gene index in $C_i$ is unique. Based on each $C_i$ the gene expression sub-matrix $x_S$ is created. Fitness of $C_i$ is calculated calling *kNN-fit* ($x_S$) with $x_S$ as an input argument. In each generation, containing *Num* chromosomes, *Num*/2 selection operations are performed. In each selection operation, two parents are chosen depending on Rowlett wheel. A one-point crossover is performed on selected parents with crossover probability *Pc*, producing two offspring. Mutations are performed with a low probability *Pm* on each offspring. Among the pairs—parents and offspring, the fitter chromosome pair is passed to next generation. During crossover and mutation, measures are taken, such that gene indices in $C_i$ remain unique. The best solution in final population is returned.

The final solution, produced by an algorithm has not been considered here as the output because the input gene expression data is noisy and number of samples is not sufficient. Instead, a collective framework has been proposed here which takes a normalized gene expression dataset as input and encompasses all four meta-heuristic algorithms (HBMO-kNN, HS-kNN, DE-kNN and SGA-kNN). Figure 2 shows how it works in a realistic environment. Multiple executions of each algorithm (HBMO-kNN, HS-kNN, DE-kNN and SGA-kNN) have been separately performed on a dataset. During each execution of an algorithm, candidate solutions (termed here as good solutions) having classification accuracy (fitness) higher or equal to 90% has been stored in an algorithm specific repository. After executions of all algorithms, each algorithm-specific repository is processed to contain the unique genes. Now overlap among the genes in different repositories has been obtained and returned as a disease-causing set of genes.

Proposed framework inherits some features from the state-of-the-art algorithms such as GA-kNN [26], GANN [39] and GADA [39]. First, it contains methods (HBMO-kNN, HS-kNN, DE-kNN and SGA-kNN) based on a meta-heuristic algorithm and a classifier (kNN) combination. Second, the proposed framework does not consider the final solution of a meta-heuristic algorithm as a final

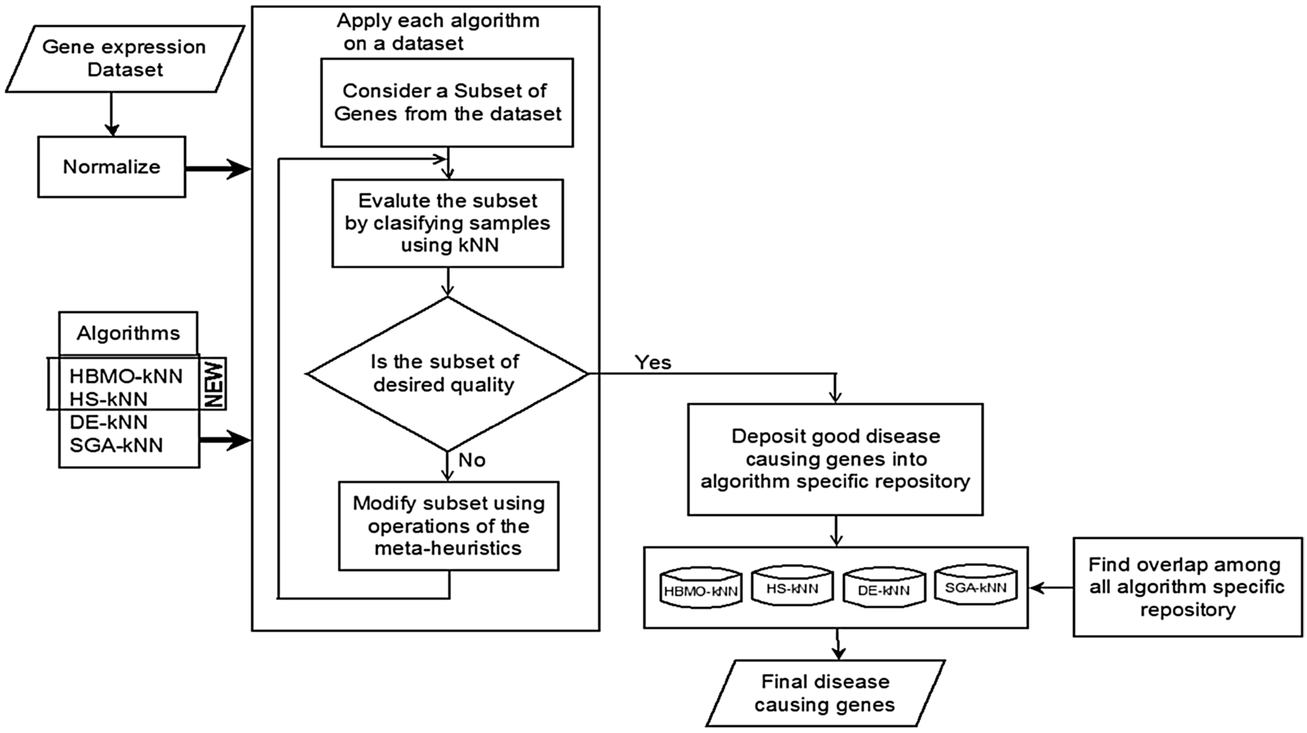
Gene expression Dataset
Normalize
Algorithms
HBMO-kNN
HS-kNN
NEW
DE-kNN
SGA-kNN
Apply each algorithm on a dataset
Consider a Subset of Genes from the dataset
Evalute the subset by clasifying samples using kNN
Is the subset of desired quality
Yes
No
Modify subset using operations of the meta-heuristics
Deposit good disease causing genes into algorithm specific repository
HBMO-kNN
HS-kNN
DE-kNN
SGA-kNN
Find overlap among all algorithm specific repository
Final disease causing genes

**Fig. 2** Proposed framework

disease-causing subset. Finally, during the execution of a meta-heuristic algorithm, good candidate solutions are stored for further statistical processing.

The state-of-the-art algorithms have following disadvantages. First, the stored candidate solutions obtained from the executions of a single meta-heuristic algorithm is inappropriate for statistical processing. This is because of the fact that decision variables change less frequently in successive candidate solutions. As a result, redundant information may be gathered. Second, the process of prioritization of genes contained in intermediate candidate solutions (even good) is questionable as they are obtained from meta-heuristic algorithms which employ inherent randomization. The proposed framework eliminates the first difficulty, because it collects candidate solutions from four meta-heuristic algorithms instead of one. The second disadvantage of the state-of-the-art algorithm has been overcome using overlap, instead of prioritization.

## 4 Data and Experimental Results

Three experiments, each based on a preeclampsia dataset have been conducted. Hardware and software used in the experiments, data of preeclampsia, and parameter setting have been described in Sect. 4.1. The Section 4.2 contains results and discussion.

### 4.1 Data and Experimental Setting

All the experiments have been done on the same computer having Intel Pentium dual-core CPU with a processing speed of 3.0 GHz and inbuilt RAM of 2 GB. All programs are written in Matlab 2011b Windows version 32 bit. Three gene expression datasets with accession number E-MEXP-1050, GSE60438 and GSE10588 have been employed in this research. The first dataset has been downloaded from Array Express, whereas the other two have been collected from Gene Expression Omnibus. Each of these datasets was produced through an underlying microarray experiment. In each such experiment, total RNA was extracted from normal as well as preeclampsia affected decidual tissues of placentas. The expression profile obtained through a microarray experiment is like a matrix (see Table 1). Here each sample (column) contains expression levels of all relevant genes of a decidual tissue. In case of E-MEXP-1050, the expression profile contains 18 diseased and 17 normal samples. Each diseased samples corresponds to a decidual tissue of critical pregnancy (suffered by preeclampsia and/or fetal growth restriction). Each normal sample corresponds to a normal placental tissue. In either case, a sample contains expression levels of 8,937 identical genes. In GSE60438, the expression profile contains 35 preeclamptic samples and 42 normotensive samples. Each sample contains expression levels of 47,323 identical genes. In GSE10588, 17 severely preeclamptic samples and 26 normal samples are present. Each sample has expression levels of 32,878 identical genes. Different aspects of the datasets have been summarized in Table 2.

Each matrix element of a gene expression dataset is a floating point number. A dataset is undergone a pre-processing stage, called normalization. Here, normalization aims to map each element ($g^N_{i,j}/g^P_{i,j}$, see Table 1) of a data matrix in [0, 1]. To apply normalization, the maximum and minimum values ($G_i^{MAX}$and $G_i^{MIN}$, respectively) of each gene ($G_i$) over all samples has been calculated first. Then each element $g_{i,j}$ ($g^N_{i,j}/g^P_{i,j}$) of the matrix has been normalized using Eq. (9)

$$\text{normalize}\left(g_{i,j}\right) = \frac{g_{i,j} - G_i^{MIN}}{G_i^{MAX} - G_i^{MIN}}. \tag{9}$$

With the normalized datasets, three experiments have been conducted. Each experiment is based on a dataset specified in Table 2. In an experiment, all (four) hybrid algorithms, namely, HBMO-kNN, HS-kNN, DE-kNN and SGA-kNN have been employed. In all population-based meta-heuristics except HBMO-kNN, the outer loop, controlling terminating condition is iterated *maxIter* (= 200) times. In these algorithms, population size, *Num* (= 30) is also identical. So, the total number of steps in each population-based algorithm is 6000 (200*30). In HBMO-kNN, whenever the fitness value changes, the point of time is recorded. Later on, each recorded time instance (corresponding to a fitness value) is scaled in the range (1–6000). In all (four) algorithms, the number of elements $d$ in candidate solution $S$ is kept 30. To set values to the important parameters in an algorithm, different values of each parameter are kept in a vector.

**Table 2** Microarray gene expression datasets collected for experiments

| S. no. | Dataset accession no. | Collected from (site) | Number of normal samples | Number of preeclampsia samples | Total number of samples ($m$) | Number of genes ($n$) | Last updated |
|---|---|---|---|---|---|---|---|
| 1 | E-MEXP-1050 | Array Express | 17 | 18 | 35 | 8,793 | 02 May, 2014 |
| 2 | GSE60438 | GEO, NCBI | 42 | 35 | 77 | 47,323 | Aug 05, 2015 |
| 3 | GSE10588 | GEO, NCBI | 26 | 17 | 43 | 32,878 | Jan 27, 2016 |

Then for all possible combinations of values of the different parameters, the algorithm is executed. During these executions, the combination of parameters for which an algorithm achieves the highest fitness is saved. These saved values of different parameters have been assigned to the corresponding parameters during experiments with the algorithm. The dataset taken for parameter setting of each algorithm is E-MEXP-1050. Table 3 shows the parameter values for different algorithms, used during experiments. An example of parameter setting is given in Fig. 3 where the parameters of HS-kNN are estimated. The parameters of HS have been taken like HMCR ∊ {0.90, 0.85, 0.80, 0.75, 0.70} and PAR ∊ {0.70, 0.65, 0.60, 0.55, 0.50}. Though fitness reaches to the highest value at HMCR = 0.70 and PAR = 0.65, but around HMCR = 0.80 and PAR = 0.60, the stability of high fitness value is being observed. So we have set the value of HMCR and PAR to 0.80 and 0.60, respectively.

## 4.2 Results and Discussion

With the algorithm parameters being set, each algorithm has been executed on 25 randomly generated problem instances. During the execution of an algorithm, average fitness, classification accuracy (%), best fitness and standard deviation in fitness have been recorded. The average has been taken over 25 solved instances. For example (see Table 4), in the experiment based on dataset E-MEXP-1050, after execution of HS-kNN, among 25 fitness values, average fitness is obtained as 33.2800. Classification accuracy is calculated by Eq. (1) and the value is 95.0857. The best fitness obtained is 34. The standard deviation in fitness values is 0.5416.

As fitness (number of properly classified samples) depends on the number of samples in a dataset, classification accuracy is a better attribute for comparison. Considering this attribute, the performance of HBMO-kNN is observed as the best in all experiments. In the first and third experiments, based on E-MEXP-1050 and GSE10588 datasets, respectively (Table 4), HBMO-kNN obtained 100% classification accuracy. In the second experiment, based on the GSE60438 dataset (Table 4), HBMO-kNN obtained classification accuracy of 99.6364%.

In all experiments, DE-kNN secured second place with respect to classification accuracy. Moreover, the research observes that (Table 4) DE-kNN is a stable performer as it has the lowest value for standard deviations of fitness as compared to other algorithms in all the experiments. In the second experiment, where HBMO-kNN has classification accuracy less than 100%, standard deviations of DE-kNN are lower than that of HBMO-kNN.

Figures 4, 5 and 6 show fitness convergence graphs of all algorithms based on datasets E-MEXP-1050, GSE60438

**Table 3** Parameters of different algorithm

| Algorithms | Parameters |
|---|---|
| HBMO-kNN | $maxIter = 2$, $Num = 5$, $d = 30$, $W = 1000$, $D = 10$, $B = 20$, $\alpha = 0.95$, $\gamma = 0.008$ |
| HS-kNN | $NI = 200$, $HMS = 30$, $d = 30$, HMCR = 0.8, PAR = 0.6, $K = 1.6$ |
| DE-kNN | $maxIter = 200$, $Num = 30$, $d = 30$, $F = 0.25$, $Cr = 0.1$ |
| SGA-kNN | $maxIter = 200$, $Num = 30$, $d = 30$, Pc = 0.7, Pm = 0.02 |

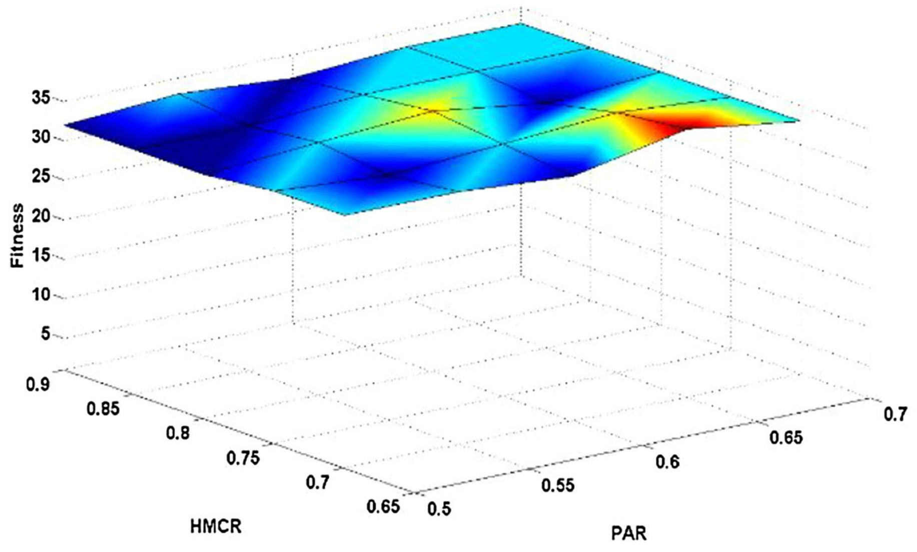

**Fig. 3** Parameter setting of HS-kNN

**Table 4** Performance of algorithms in different experiments; each experiment is based on a dataset

| Dataset | Algorithm | Average fitness | Classification accuracy (%) | Best fitness | Std. dev. of fitness |
|---|---|---|---|---|---|
| E-MEXP-1050 (number of samples = 35) | HBMO-kNN | 35.0000 | 100.000 | 35 | 0.0000 |
| | HS-kNN | 33.2800 | 95.0857 | 34 | 0.5416 |
| | DE-kNN | 34.8000 | 99.4286 | 35 | 0.4082 |
| | SGA-kNN | 32.3600 | 92.4571 | 34 | 0.9074 |
| GSE60438 (number of samples = 77) | HBMO-kNN | 76.7200 | 99.6364 | 77 | 0.4583 |
| | HS-kNN | 75.3600 | 97.8701 | 77 | 0.7572 |
| | DE-kNN | 76.8800 | 99.8442 | 77 | 0.3317 |
| | SGA-kNN | 73.9200 | 96.0000 | 76 | 1.2220 |
| GSE10588 (number of samples = 43) | HBMO-kNN | 43.0000 | 100.000 | 43 | 0.0000 |
| | HS-kNN | 42.2000 | 98.1395 | 43 | 0.5000 |
| | DE-kNN | 43.0000 | 100.000 | 43 | 0.0000 |
| | SGA-kNN | 41.6800 | 96.9302 | 43 | 0.9452 |

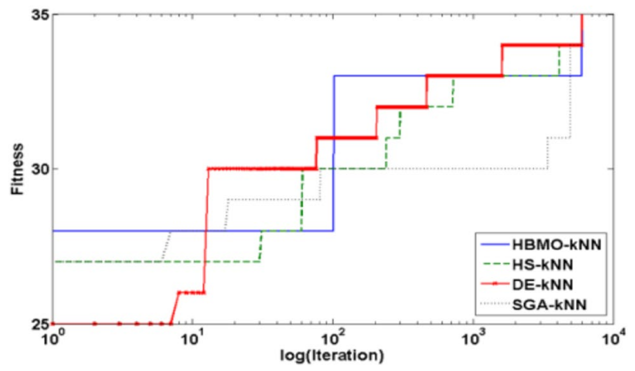

**Fig. 4** Fitness convergence of the experiment (Table 4) based on E-MEXP-1050

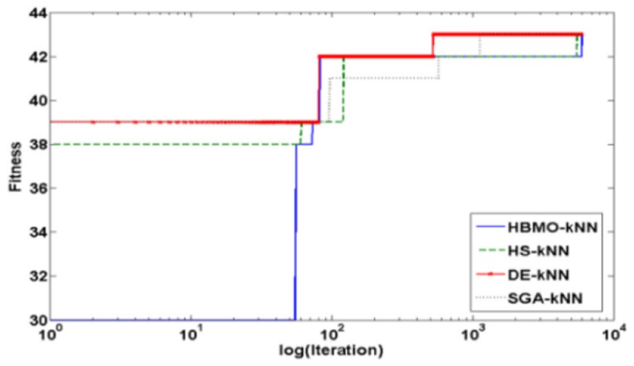

**Fig. 6** Fitness convergence of the experiment (Table 4) based on GSE10588

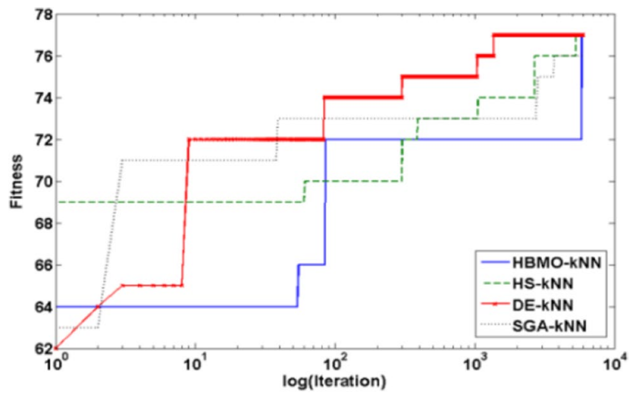

**Fig. 5** Fitness convergence of the experiment (Table 4) based on GSE60438

and GSE10588, respectively. Here, $x$-axis is the logarithm of iteration (= 6000; 200 of *maxIter* *30 of *Num*) and the $y$-axis is fitness. In Fig. 4 (based on E-MEXP-1050), maximum fitness (35, classification accuracy is 100%) has been obtained by HBMO-kNN and DE-kNN. Other two algorithms obtained a fitness of 34. In Fig. 5 (based on GSE60438), HBMO-kNN, HS-kNN and DE-kNN have obtained the highest fitness of 77 (100% classification accuracy). SGA-kNN has obtained fitness of 76. In Fig. 6 (based on GSE10588), HBMO-kNN and DE-kNN have obtained the highest fitness of 43 (100% classification accuracy). HS-kNN and SGA-kNN have obtained the fitness of 42 and 41, respectively.

In Fig. 7, comparison of relative performance of algorithms in different experiments has been shown. Together with four meta-heuristics based algorithms, the performance of simple kNN is also shown here separately. Simple kNN is applied to random initial candidate solutions and it attains classification accuracies in the range of 40–80% in different experiments. This explains the reason why in Figs. 4, 5 and 6 the graphs start with high initial fitness. In the graphs, the global best solutions of different algorithms are plotted. As

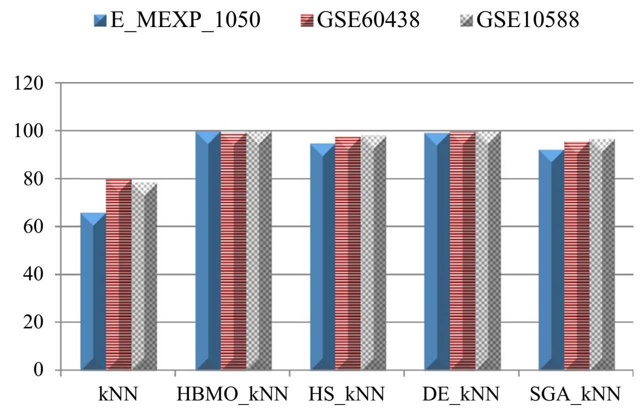

**Fig. 7** Relative performance of kNN, HBMO-kNN, HS-kNN, DE-kNN and SGA-kNN in the different datasets

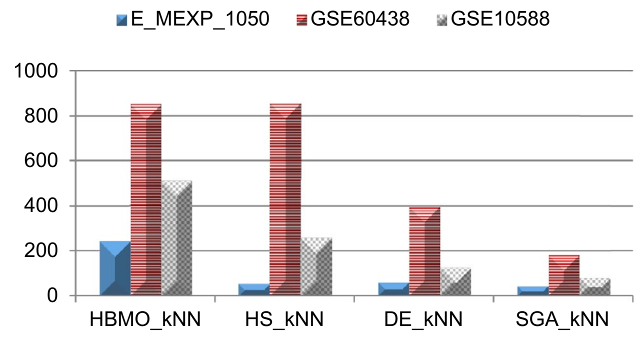

**Fig. 8** Time analysis of HBMO-kNN, HS-kNN, DE-kNN and SGA-kNN in different datasets

the best candidate solution (global best) of initial population is classified through kNN, the initial classification accuracy may be as high as 80%. Due to the application of meta-heuristics rest part of accuracy has been achieved.

Figure 8 shows relative timing property of individual algorithms in different experiments (based on datasets E-MEXP-1050, GSE60438 and GSE10588). SGA-kNN took the least time in all experiments. DE-kNN took the second lowest time. Time in an experiment generally depends on the number of genes in the corresponding gene expression dataset.

In each experiment, during 25 runs of each algorithm, the candidate solutions that reach classification accuracy of 90% or above are saved. Among the saved solutions of different algorithms, the set of genes commonly obtained by all algorithms are stored. These genes have been termed as disease critical genes. Number of disease critical genes obtained from the datasets, E-MEXP-1050, GSE60438 and GSE10588 are 87, 73 and 74, respectively. Many widely known preeclampsia genes [56, 57] including STS, EPHX1, LEP, LRP8, ADD1, YWHAQ, INSR, CCL8 have been found here.

Each of the preeclampsia genes obtained in this research has been found to be differentially expressed (among preeclamptic and normal samples) in separate previous researches. The appearance of so many important genes together in this collective algorithmic study suggests that they may be a part of some biological pathway. This needs further clinical investigations. Among the important genes, a few such as INSR and CCL8 were previously proved to be associated with adult vascular and metabolic diseases. Later on, an association of these genes with preeclampsia have been established. The appearance of these genes in this research is a further step to establish the association. Some new genes such as MSX2, LOC440737 and UROS have also appeared in this result. No direct association of these genes with preeclampsia has been established so far. MSX2 had been observed to induce Trophoblast invasion in the placenta [58] whereas LOC440737 is considered to be involved in the endometrial–trophoblast interaction. Both of these processes have a complex relationship with preeclampsia [59]. On the other hand, UROS is found responsible for hypertension. Gestational hypertension (hypertension in second or third trimester) may lead to preeclampsia [3]. So, these new genes may have an indirect complex relationship with the disease (preeclampsia) subject to further biological investigation.

Here, one point is to be noted that though the rows in gene expression datasets are termed as genes so far, they are actually named as 'probe ids'. A probe id is an entity associated with a gene, in context to microarray technology. We have used the indexes of probe ids during processing. At last, from an index, the probe id is obtained from the dataset and from probe id the gene symbol is obtained from a special type of data file (named platform file) supplied by NCBI (from where dataset was collected).

The performance of proposed algorithms of this research, HBMO-kNN and HS-kNN has been compared with three state-of-the-art methods. They are GANN and GADA, both implemented by Tejera et al. (2013) [39], and GA-kNN was proposed by Li et al. (2001) [26]. Tejera et al. (2013) have executed the algorithms (GANN and GADA) on two (out of five used in [39]) datasets (E-MEXP-1050 and GSE10588) which have also been used in this research. The average and best classification accuracies of the algorithms published by Tejera et al. (2013) [39], have been used in the comparative study (see Fig. 9) of this research. The other state-of-the-art algorithm, GA-kNN has been downloaded (executable version) from the website [60] of its originator Li et al. (2001) and executed 25 times on E-MEXP-1050 obeying the conditions provided. The parameters, such as number of niches, number of generations, population size, chromosome length, ermination fitness cut-off and number of solutions specified have been set as arguments during executions of the

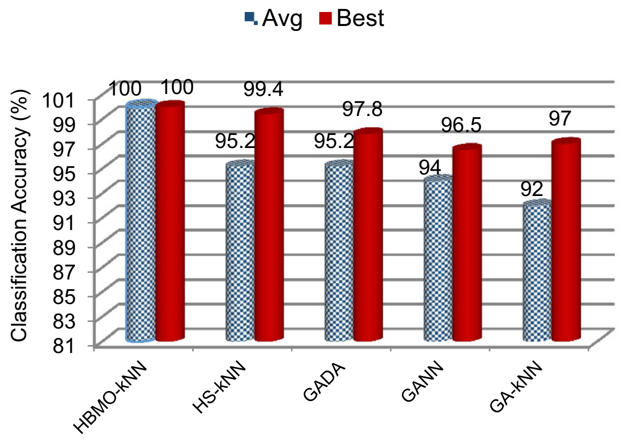

**Fig. 9** Comparison of proposed algorithms (HBMO-kNN and HS-kNN) with three state-of-the-art methods (GADA and GANN [39], and GA-kNN [26]) in terms of average and best classification accuracy (%)

algorithm are 3, 50, 50, 30, 34 and 200, respectively. Though in a run of GA-kNN, the maximum number of generations is 50, if the given highest cut-off fitness (34) out of 35 samples in E-MEXP-1050 has been achieved in fewer generations, the program terminates. The output of an execution of GA-kNN is the number of properly classified samples. The output of each run is stored and classification accuracy (%) has been calculated using Eq. (1).

Observation shows (see Fig. 9), HBMO-kNN outperforms all other algorithms in both average (100) and best (100) classification accuracies. HS-kNN is similar to GADA in average (95.2) classification accuracy but wins in best classification rate (99.4).

Change of the fitness value of each algorithm (HBMO-kNN, HS-kNN and GA-kNN) in successive generations (iterations) has been plotted in Fig. 10. The run (among 25 runs of the algorithm) in which the highest fitness has been achieved by an algorithm has been considered. It has been observed that HBMO-kNN is better than GA-kNN and HS-kNN is as good as GA-kNN in achieving highest fitness.

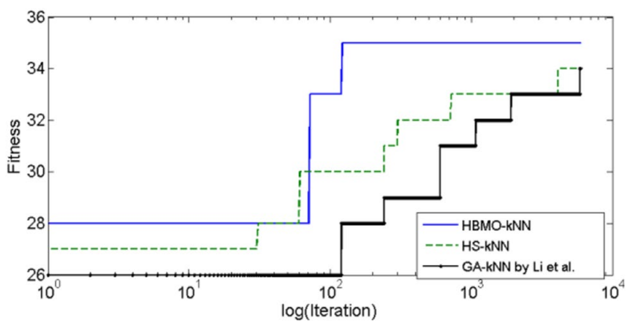

**Fig. 10** Comparison of fitness convergence among HBMO-kNN, HS-kNN and GA-kNN [26]

In addition, both HBMO-kNN and HS-kNN show a higher rate of improvement than GA-kNN. These two important observations signify that the newly proposed algorithms have outperformed the existing state-of-the-art algorithm (GA-kNN) to obtain the subset of disease-causing genes (i.e., maximizing number of properly classified samples).

Some recent gene selection algorithms, not executed on the same dataset as the proposed algorithms, have also been compared with the proposed algorithms on the basis of their overall average classification accuracies in different datasets (see Table 5). Average classification accuracy of two recent gene selection algorithms such as $\chi^2$ DC and IVPSO have been obtained from the results reported by Zang et al. (2014) [36] and Ramyachitra et al. (2015) [37], respectively. Average classification accuracy (CA) of a proposed algorithm, e.g., HBMO-kNN has been calculated by taking the arithmetic mean of the classification accuracies of the algorithm over the three datasets (E-MEXP-1050–100%, GSE60438–99.6364% and GSE10588–100%, see Tables 4, 5). Here also HBMO-kNN is the best with average classification accuracy 99.87% followed by HS-kNN (97.03%).

## 5 Conclusion

Identification of a small subset of genes responsible for a disease by comparing characteristics of normal and diseased gene expression dataset is an important task in bioinformatics. In this paper, three preeclampsia datasets have been collected and identification of disease critical genes out of these datasets has been formulated as an optimization problem. Meta-heuristic algorisms such as HBMO, HS, DE and GA have been used for this purpose. To obtain the fitness of a candidate solution, classifier kNN has been used. Two algorithms, namely, HBMO-kNN and HS-kNN have been proposed here and implemented on three different datasets of preeclampsia. In each experiment, apart from the proposed algorithms HBMO-kNN, HS-kNN, other two algorithms, namely, DE-kNN and SGA-kNN have also been implemented. The performance of these four algorithms has been compared with respect to some standard metrics, such as average fitness, classification accuracy, best fitness and standard deviation of fitness and execution time. The experimental study shows that HBMO-kNN outperformed all other algorithms with a classification accuracy

**Table 5** Comparison among proposed and current gene selection algorithms depending upon overall classification accuracy in different datasets

| Method | HBMO-kNN | HS-kNN | $\chi^2$ DC | IVPSO |
|---|---|---|---|---|
| CA (%) | 99.87 | 97.03 | 85.51 | 96.88 |

of 99.64–100% followed by DE-kNN (99.42–100%). The proposed algorithms (HBMO-kNN and HS-kNN) also have been compared with the state-of-the-art algorithms, giving better results. During execution of each of the algorithms, candidate solutions having classification accuracy above 90% have been saved separately. From the good solutions obtained by an experiment, the genes commonly found by all algorithms are considered to be disease critical. The numbers of disease critical in the three datasets E-MEXP-1050, GSE60438 and GSE10588 are 87, 73 and 74, respectively. Genes such as STS, LEP, LRP8 which are well known as preeclampsia genes have appeared in our findings.

Many algorithms have been proposed so far for gene selection using sample classification. But this research work is important because of implementing a large number of meta-heuristics and compiling their results together. This experimental approach is also important in a sense that it provides a comparative as well as a collective analysis of results. The comparative study shows that one of our proposed algorithms, HBMO-kNN, is the best in performance among all algorithms implemented here. On the other hand, the collective study visualizes that the grouping of solutions of all algorithms and search for common genes achieved significant overlap (87, 73 and 74 in E-MEXP-1050, GSE60438 and GSE10588, respectively).

There are some future scopes. The kNN-based fitness calculation algorithm, kNN_fit, can be easily used in any other meta-heuristic to serve similar purposes. Use of collective strategy using a number of meta-heuristics may be helpful to solve similar problems. New meta-heuristic algorithms may be found to be more appropriate in this context. Identification of disease critical genes of any other disease can be performed using aforesaid methods. Incorporation of other classifiers such as SVM and multilayer perceptron, as a substitute of kNN, may further improve the accuracy of gene selection. As the dataset that can be used for positive control involves more than two classes, kNN_fit may be modified accordingly to facilitate multi-class (more than two) classification. Application of this modified version to such a dataset will be useful to reveal which other diseases play role in increasing the risk of preeclampsia or how preeclampsia may trigger some other disease.

**Acknowledgements** We are thankful to TEQIP-WBUT for funding our research.

## References

1. Lyall F, Belfort M (2007) Pre-eclampsia etiology and clinical practice, 1st edn. Cambridge University Press, Cambridge
2. Laivuori H (2007) Genetic aspects of preeclampsia. Front Biosci 12:2372–2382
3. Williams PJ, Pipkin FB (2011) The genetics of pre-eclampsia and other hypertensive disorders of pregnancy. Best Pract Res Clin Obstet 25:405–417. https://doi.org/10.1016/j.bpobgyn.2011.02.007
4. Haram K, Mortensen JH, Nagy B (2014) Genetic aspects of preeclampsia and the HELLP syndrome. J Pregnancy. https://doi.org/10.1155/2014/910751
5. Eisen MB, Brown PO (1999) DNA arrays for analysis of gene expression. Method Enzymol 303:179–205
6. Lockhart DJ, Winzeler EA (2000) Genomics, gene expression and DNA arrays. Nature 405(6788):827–836
7. Quackenbush J (2002) Microarray data normalization and transformation. Nat Genet Suppl 32:496–501
8. Dillies MA, Rau A, Aubert J et al (2012) A comprehensive evaluation of normalization methods for illumina high-throughput RNA sequencing data analysis. Brief Bioinform 14(6):671–683. https://doi.org/10.1093/bib/bbs046
9. Reimer T, Koczan D, Gerber B, Richter D, Thiesen HJ, Friese K (2002) Microarray analysis of differentially expressed genes in placental tissue of pre-eclampsia: up-regulation of obesity-related genes. Mol Hum Reprod 8(7):674–680. https://doi.org/10.1093/molehr/8.7.674
10. Enquobahrie DA, Meller M, Rice K, Psaty BM, Siscovick DS, Williams MA (2008) Differential placental gene expression in preeclampsia. Am J Obstet Gynecol 199(5):566e1–e11. https://doi.org/10.1016/j.ajog.2008.04.020, doi
11. Hoegh AM, Borup R, Nielsen FC, Sørensen S, Hviid TVF (2009) Gene expression profiling of placentas affected by pre-eclampsia. J Biomed Biotechnol. https://doi.org/10.1155/2010/787545
12. Kleinrouweler CE, Uitert MV, Moerland PD, Ris-Stalpers C, Van der Post JAM, Afink GB (2013) Differentially expressed genes in the pre-eclamptic placenta: a systematic review and meta-analysis. PLoS One 8(7):E68991. https://doi.org/10.1371/journal.pone.0068991
13. Ali AF, Hassanien AE (2016) A Survey of metaheuristics methods for bioinformatics applications. In: Hassanien AE, Grosan C, Tolba MF (eds) Applications of intelligent optimization in biology and medicine. intelligent systems reference library, vol 96. Springer, Cham
14. Biswas S, Acharyya S (2016) Neural model of gene regulatory network: a survey on supportive meta-heuristics. Theor Biosci 135(1–2):1–19. https://doi.org/10.1007/s12064-016-0224-z
15. Madeira SC, Oliveira AL (2004) Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Trans Comput Biol Bioinform 1(1):24–45
16. Jing PJ, Shen HB (2004) MACOED: a multi-objective ant colony optimization algorithm for SNP epistasis detection in genome-wide association studies. Bioinformatics 22(5):31 634–641
17. Tuo S, Zhang J, Yuan X, Zhang Y, Liu Z (2016) FHSA-SED: two-locus model detection for genome-wide association study with harmony search algorithm. PloS One 11(3):e0150669
18. Aflakparast M, Salimi H, Gerami A, Dubé MP, Visweswaran S, Masoudi-Nejad A (2014) Cuckoo search epistasis: a new method for exploring significant genetic interactions. Heredity 112(6):666
19. Golub TR, Slonim TDK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439):531–537
20. Cho JH, Lee D, Park JH, Lee IB (2003) New gene selection method for classification of cancer subtypes considering within-class variation. FEBS Lett 551(1–3):3–7. https://doi.org/10.1016/S0014-5793(03)00819-6
21. Andoni A (1993) K nearest neighbor search: the old, the new, and the impossible (Ph.D. thesis). Department of Electrical Engineering and Computer Science, Cambridge, MA

22. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
23. Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene selection for cancer classification using support vector machines. Mach Learn 46(1):389–422. https://doi.org/10.1023/A:1012487302797
24. Goldberg DE, Deb K (1989) Genetic algorithms in search, optimization and machine learning, 1st edn. Addison-Wesley Publishing Company Inc., Boston
25. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference neural networks held in Perth, WA, (December)
26. Li L, Weinberg CR, Darden TA, Pedersen LG (2001) Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/kNN method. Bioinformatics 17(12):1131–1142. https://doi.org/10.1093/bioinformatics/17.12.1131
27. Li L, Weinberg CR (2003) Gene selection and sample classification using a genetic algorithm and k-nearest neighbor method, In: A practical approach to microarray data analysis, 1st edn, pp 216–229, Springer, New York
28. Tsai CA, Chen CH, Lee TC, Ho IC, Yang UC, Chen JJ (2004) Gene selection for sample classifications in microarray experiments. DNA Cell Biol 23(10):607–614
29. Ye J, Li T, Xiong T, Janardan R (2004) Using uncorrelated discriminant analysis for tissue classification with gene expression data. IEEE/ACM Trans Comput Biol Bioinform 1(4):181–190. https://doi.org/10.1109/TCBB.2004.45
30. Xu X, Zhang A (2005) Virtual gene: a gene selection algorithm for sample classification on microarray datasets. Comput sci—ICCS 3515:1038–1045
31. Zhang X, Lu X, Shi Q, Xu X, Leung HE, Harris LN, Iglehart JD, Miron A, Liu JS, Wong WH (2006) Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data. BMC Bioinform. https://doi.org/10.1186/1471-2105-7-197
32. Díaz-Uriarte R, Andrés SA (2006) Gene selection and classification of microarray data using random forest. BMC Bioinform. https://doi.org/10.1186/1471-2105-7-3
33. Chien-Pang L, Wen-Shin L, Yuh-Min C, Bo-Jein K (2011) Gene selection and sample classification on microarray data based on adaptive genetic algorithm/k-nearest neighbor method. Expert Syst Appl 38(5):4661–4667
34. Yu L, Han Y, Berens ME (2012) Stable gene selection from microarray data via sample weighting. IEEE/ACM Trans Comput Biol Bioinform 9(1):262–272
35. Glaab E, Bacardit J, Garibaldi JM, Krasnogor N (2012) Using rule-based machine learning for candidate disease gene prioritization and sample classification of cancer gene expression data. PLoS One 7(7):e39932. https://doi.org/10.1371/journal.pone.0039932
36. Zhang H, Li L, Luo C, Sun C, Chen Y, Dai Z, Yuan Z (2014) Informative gene selection and direct classification of tumor based on chi-square test of pairwise gene interactions. BioMed research international. https://www.hindawi.com/journals/bmri/2014/589290/. Accessed 20 July 2017
37. Ramyachitra D, Sofia M, Manikandan P (2015) Interval-value based particle swarm optimization algorithm for cancer-type specific gene selection and sample classification. Genome Data 5:46–50. https://doi.org/10.1016/j.gdata.2015.04.027
38. Alshamlan H, Badr G, Alohali Y (2015) mRMR-ABC: a hybrid gene selection algorithm for cancer classification using microarray gene expression profiling. Biomed Res Int. https://doi.org/10.1155/2015/604910
39. Tejera E, Bernardes J, Rebelo I (2013) Co-expression network analysis and genetic algorithms for gene prioritization in preeclampsia. BMC Med Genom. https://doi.org/10.1186/1755-8794-6-51
40. Hansen P, Mladenovi´c N (2001) Variable neighbourhood search: principles and applications. Eur J Oper Res 130:449–467. https://doi.org/10.1016/S0377-2217(00)00100-4
41. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359. https://doi.org/10.1023/A:1008202821328
42. Kirkpatrick S, Gelatt CD, Vecchi MP Jr (1983) Optimization by simulated annealing. Science 220(4598):672–681
43. Dutta J, Biswas S, Saha S, Acharyya S (2015) Identification of disease critical genes causing preeclampsia: meta-heuristic approaches. In: Proceedings of fourth IEEE UP section conference on electrical computer and electronics (UPCON) held in Allahabad, India
44. Saha S, Biswas S, Acharyya S (2016) Gene selection by sample classification using k nearest neighbor and meta-heuristic algorithms. In: Proceedings of sixth IEEE international advance computing conference held in Eluru, India
45. Haddad OB, Afshara A, Marin MA (2006) Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization. Water Resour Manag 20(5):661–680. https://doi.org/10.1007/s11269-005-9001-3
46. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
47. Goldberg DE (1983) Computer-aided pipeline operation using genetic algorithms and rule learning (PhD thesis). The University of Michigan, MI, US
48. Cunningham P, Delany SJ (2007) k-nearest neighbour classifiers (Tech. Rep. UCD-CSI-2007–4). University of Dublin, Dublin
49. Wu X, Kumar V, Quinlan JR, Ghosh J et al (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14:1–37. https://doi.org/10.1007/s10115-007-0114-2
50. Moore AW (2001) Cross-validation for detecting and preventing over-fitting. http://www.cs.cmu.edu/~awm/tutorials. Accessed 7 July 2017
51. Afshara A, Haddad OB, Mariñob MA, Adams BJ (2007) Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. J Franklin Inst 344(5):452–462. https://doi.org/10.1007/s11269-005-9001-3
52. Geem ZW (2010) State-of-the-art in the structure of harmony search algorithm. In: Kacprzyk J (ed) Recent advances in harmony search algorithm. Springer, Berlin, pp 1–10
53. Das S, Mukhopadhyay A, Roy A, Abraham A, Panigrahi BK (2011) Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization. IEEE Trans Syst Man Cybern B Cybern 41(1):89–106
54. Neri F, Tirronen V (2010) Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 33(1):61–106. https://doi.org/10.1007/s10462-009-9137-2
55. Sastry K, Goldberg DE, Kendall G (2005) Genetic algorithms. In: Burke EK, Graham K (eds) Search methodologies: introductory tutorials in optimization and decision support system, 2nd edn. Springer, New York, pp 97–125
56. Gratton AM, Ye L, Brownfoot FC, Hannan NJ et al (2016) Steroid sulfatase is increased in the placentas and whole blood of women with early-onset preeclampsia. Placenta 48:72–79. https://doi.org/10.1016/j.placenta.2016.10.008
57. Hogg K, Blair JD, Dadelszen PV et al (2013) Hypomethylation of the *LEP* gene in placenta and elevated maternal leptin concentration in early onset pre-eclampsia. Mol Cell Endocrinol 367(1–2):64–73
58. Liang H, Zhang Q, Lu J et al. (2016) MSX2 induces trophoblast invasion in human placenta. Plos One. https://doi.org/10.1371/journal.pone.0153656
59. Lyall F, Bulmer JN, Duffie E (2001) Human trophoblast invasion and spiral artery transformation. Am J Patho 158(5):1713–1721
60. https://www.niehs.nih.gov/research/atniehs/labs/bb/staff/li/index.cfm. Accessed 10 July 2017