Scientific Journal of Impact Factor(SJIF): 3.134
e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2,Issue 3, March -2015

Bio-Inspired Bayesian Network Learning Algorithm for Bayesian Network Classifiers

Shefali K Singhal[1]

[1]*Professor, Computer Science, MBICT Engineering College, shefali.singhal@gmail.com*

Abstract *—Learning an optimal Bayesian belief network for a Bayesian network classifier is a NP-hard problem. A number of heuristic-based algorithms have been proposed for supervised learning of Bayesian belief network such as Tree-Augmented Naïve Bayes (TAN). In the past decade, bio-inspired swarm intelligence (**SI**) based algorithms have be proposed for many optimization problems. Swarm Intelligence based algorithms are characterized by the collective decentralized decision-making of several independent agents to search for the optimal solution in the solution search space.*

*In this paper, we propose a novel swarm intelligence based Hunting Group Search Bayesian Network (**HuGS**) algorithm to learn the Bayesian network for a Bayesian network classifier. HuGS has been inspired by the behavior of a pack of hunting animals such as wolves. The classification accuracy of the proposed HuGS algorithm is tested against other Bayesian network classifiers such as Naïve Bayes and TAN. The performance of HuGS is evaluated using 20 benchmark datasets from the UCI classification datasets. Our experiments show that HuGS outperforms the other state of the art Bayesian network learning algorithms.*

Keywords- *Swarm Intelligence, Evolutionary Computation, Machine Learning, Hunting Search Algorithm, Classifier, Naive Bayes, Bayesian network*

I. INTRODUCTION

A classification} algorithm takes a set of training data and creates a classifier. Once the classifier is learned from the training dataset, new data samples can be classified using the classifier. Let X_1, ... , X_n be the attributes of the data. A data sample *d* can be described as x_1, ..., x_n where x_i represents the values of attribute X_i. Learning a classifier from the training data can be viewed as learning the mapping $c = f(d)$, that can predict the class *c* of the data sample *d*.

A bayesian belief network classifier comprises of two parts: (1) a bayesian belief network graph (*directed acyclic graph*) that captures the dependencies between the attributes and (2) conditional probability tables.

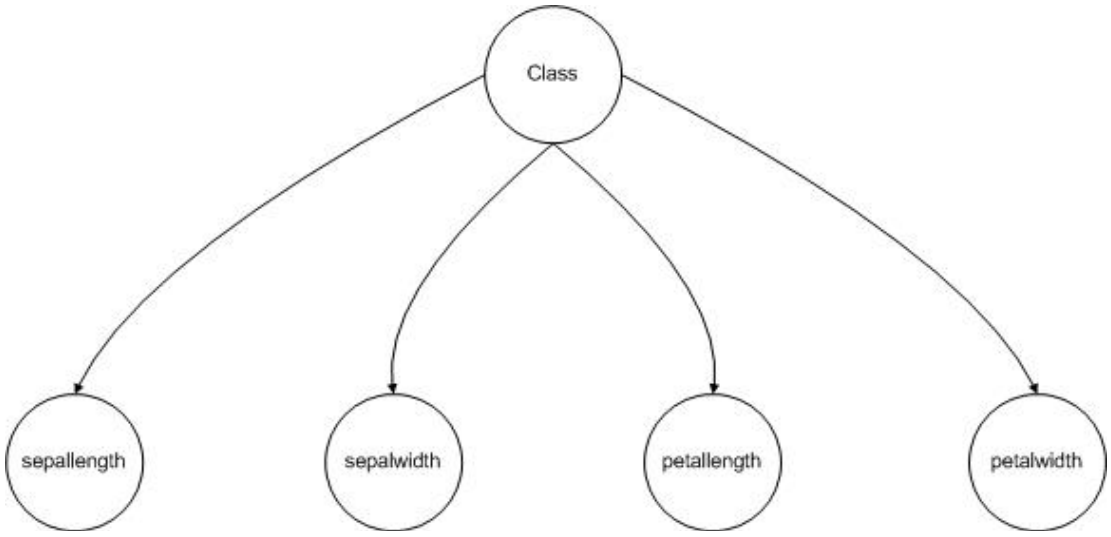

Figure 1. A naive bayes Bayesian network graph for the iris dataset

Each node in the bayesian belief network graph represents an attribute A_i of the dataset. An arc between 2 nodes indicates the attribute dependency. Figure 1 depicts a naive bayes based bayesian belief network graph for the iris dataset from the UCI machine learning datasets downloaded from the WEKA website [1]. In a naive bayes network, every attribute is conditionally dependent only on the class attribute, as depicted in the above figure. Consequently, the class attribute is the parent of all the attributes of the iris dataset (i.e. sepallength, sepalwidth, petallength, petalwidth). Since

all the attributes in a naive bayes bayesian network graph is conditionally independent of each other, there are no arcs between the attributes.

Each attribute in a bayesian belief network classifier has an associated conditional probability table (CPT). The CPT for an attribute *X* gives the conditional probability distribution *P(X|Parents(X))*. The bayesian belief network graph along with the CPT describe a joint probability over attributes *X*, given as:

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | \mathbf{Pa}(x_i)) \quad \ldots\ldots \quad (1)$$

where $Pa(x_i)$ are the parents of variable x_i in the bayesian belief network graph.

A naive bayes classifier makes the class conditional independence assumption. This implies that if the class of the data sample is known then the values of the attributes of the data sample are conditionally independent of each other. However, in reality, this assumption is often false. Bayesian belief network classifier enable the encoding of the dependency between the attributes. However learning an optimal bayesian belief network classifier has been shown to an NP-hard problem [2] [3].

In the past decade, many metaheuristic algorithms based on Swarm Intelligence [4][5] have been used to solve constraint optimization problems. SI algorithms incorporate a large number of simple homogeneous agents with simple behaviors who collaborate and share information to find the global optima. SI-inspired algorithms are characterized by their simplicity and the lack of a central management. Consequently, these algorithms are relatively fast, robust and effective in finding near optimal if not optimum solutions. For example, ants can locate food over long distances without the help of advanced communications and find the shortest part to the food. Even though each ant individually does not have a clue, collectively as a swarm, they are intelligent.

Hunting search algorithm [6][7] is a bio-inspired metaheuristic algorithm for optimization problems and has been inspired by the behavior of animals such as wolves who hunt in packs and collaborate to catch a prey. Each member of the group positions itself based on the position of the other members in the group. In case the prey escapes, the hunting group repositions itself. Hunting Search Algorithm has been sucessfully applied in many domains such as clustering [8], feature selection [9][10] and high-dimensional function optimization problems [11]. However, to the best of our knowledge, Hunting Search Algorithm has not be used to learn the bayesian network structure for a bayesian network classifier.

In this paper, we propose a hunting group search based bayesian network classifier named HuGS-Classifier. The HuGS-Classifier learns the structure of the Bayesian network using the hunting search optimization technique. This paper is structured as follows: Section II gives a brief overview on the related work in Bayesian network classifiers, swarm intelligence and hunting search algorithm. In section III, we introduce the HuGS-Classifier and explain the steps to learn the bayesian network using the hunting search meta-heuritic algorithm. Section IV presents the experimental methodology and discusses the results. Finally our conclusions are stated in section V along with some remarks for future work.

II. RELATED WORK

Bayesian belief network classifier can be learned in two steps: (a) Learning the bayesian belief network graph and (b) learning the CPTs for the graph learned in step a. Learning the optimal Bayesian belief network for a Bayesian network classifier is an NP-hard problem [2][3]. A number of heuristic-based algorithms have been proposed for learning of bayesian belief network such as Naive-Bayes [12], K2 [13], TAN [14]. An overview of these methods is given by Cheng et al. in [15]. Learning the CPT is straight-forward and can be done by estimating the likelihood of the value of the attribute given value of its parent attributes.

Hunting Search Algorithm described by Oftadeh et al. [6] has been inspired by hunting animals such as wolves that collaborate in order to catch prey. This meta-heuristic-based algorithm has been used for constrained optimization problems such as transmission-constrained unit commitment [16], no-wait flowshop scheduling [17] and other optimization problems [18][19].

In the last decade, there has been active research in the area of Swarm Intelligence inspired data ming algorithms. A comprehensive summary of the data mining/machine learning algorithms based on swarm intelligence has been compiled by Martens et al. [20]. As noted in this paper, most of the research in SI-inspired data mining algorithms has been concentrated on rule based classifiers, and clustering. However, to the best of our knowledge, hunting search meta-heuristic has not been used for learning Bayesian network classifier.

III. HuGS CLASSIFIER

A Naive Bayes classifier ignores the dependencies amongst the attributes. The Bayesian network classifiers overcomes this short-coming by modeling the dependencies between the attributes in the form of a Bayesian network. The dependencies in the attributes are modeled by adding directed arcs between the attributes. The proposed network learning algorithm, **Hu**nting **G**roup **S**earch (HuGS) Bayesian Network Classifier, is inspired from hunting search

algorithm. The number of arcs (*e*) to be added and the maximum number of parents (*k*) of a node are the design parameters of this Bayesian network learning algorithm.

HuGS learning algorithm is outlined in Algorithm 1 given below. In general, the HuGS algorithm works as follows: First, it initializes a set of hunters. Once the hunters have been initialized, each hunter performs local search by moving towards the leader and position correction mechanisms. After a few iterations, all the hunters are re-initialized to new positions. This ensures that the algorithm does not get stuck in local optima. The algorithm terminates when the maximum number of iterations have been completed.

Algorithm 1: HuGS Algorithm

```
1.  Input: A dataset of training examples.
2.  Return: A bayesian network for D.
3.  BEGIN
4.  BestNetwork_global = Ø
5.  Initialize the parameters
6.  for i=0 to HG_Size
7.      Initialize hunter[i]   // Create solution as per algo.
8.  end for
9.  for  i=0 to HG_Size
10. Identify the leader hunter[i]
11. BestNetwork_global = hunter[i]
12. end for
13. repeat
14.   for i=0 to HG_Size
15.     Update hunter[i] to  hunter[i]' to move towards the leader
16.     if Q(hunter[i]') > Q(hunter[i]) then
17.         hunter[i] = hunter[i]'
18.     end if
19.   end for
20.   for  i=0 to HG_Size
21.     Update hunter[i] to  hunter[i]'  for position correction
22.     if Q(hunter[i]') > Q(hunter[i]) then
23.         hunter[i] = hunter[i]^{'}$}
24.     endif
25.   end for
26.   Identify the leader hunter[i]
27.   BestNetwork_global = hunter[i]
28.   Re-organize the hunters
29. until max_iterations
30. return BestNetwork_global
31. END
```

The algorithm learns the structure of the Bayesian network as following:

Step 1: Algorithm Initialization: In the first step, the algorithm parameters are initialized. These include the hunting group size (*HG_Size*), the maximum movement towards leader (*MML*), and the hunting group consideration rate (*HGCR*). The hunting group size (*HG_Size*) denotes the number of hunters in the hunting group (*HG*). We initialize hunting group (*HG_Size = 10*) to 10 hunters in our experimental setup. Each hunter represents a candidate solution.

Step 2: Hunting Group Initialization: Once the global algorithm parameters are set in step 1, the algorithm initializes each hunter in the hunting group specified by *HG_size*. Algorithm 2 shows the creation of the hunter (candidate solution). The model parameters to be initialized in this step are the *k*, the maximum number of parents of a node, and *e*, the maximum number of edges to be added to the model. The solution Bayesian network classifier graph (*BNC*) is constructed by adding edges of the form $X_1 \rightarrow X_2$, where $X_1 \neq X_2$ and X_1 and X_2 are the predictor attributes. Subject to the constraints of the maximum number of parents *k*, *e* directed edges are added to network, ensuring that no edge creates a cycle in the network graph. Each hunter chooses the model parameters *k* and *e* randomly from the feasible design space of our problem.

Once the Bayesian network classifier has been learnt, HuGS algorithm identifies the best solution, which is marked as the leader. Since HuGS algorithm is designed for solving classification problems, the leader is the solution which has the best predictive performance. In this paper, we use *accuracy* as the quality measure of our solution. The accuracy of a classifier on a given dataset is the percentage of tuples that are correctly classified, and is given as follows:

Algorithm 2: Create Hunters Procedure

```
1.  BEGIN CreateHunter()
2.  BNC ← Ø
3.  k = hunter.selectMaxParents()
4.  e = hunter.selectNumEdges()
5.  i = 0
6.  while i < e
7.      {i ← j }= hunter.addEdge()
8.      BNC ← BNC ∪ {i ← j}
9.      if BNC.findCycle == true or
            BNC.exceedParentLimit == true then
10.         BNC ← BNC – {i ←j}
11.     endif
12. end while
13. BNC.LearnCPTs()
14. return BNC
```

$$Accuracy = \frac{TP + TN}{TC} \quad \ldots\ldots \quad (2)$$

where,

True Positives (TP): Positive tuples correctly labeled by the classifier.

True Negatives (TN): Negative tuples correctly labeled by the classifier.

Total Count (TC): Total tuple count in the dataset.

Step 3: Movement towards the leader: Once the leader has been identified using equation 2, a local search is carried out to improve the quality of the solution. The local search is carried out as follows: All the hunters in the HuGS update their positions by moving towards the leader as per equation 3:

$$x_i^{'} = x_i + rand \times x_i^l - x_i \quad \ldots\ldots \quad (3)$$

where maximum movement toward the leader (**MML**) depends on the problem under consideration. We use a value of 0.5 in our implementation. x_i^l is the position of the leader. After each hunter moves towards the leader, it's quality is evaluated as per equation 2. If quality of the solution is better, the hunter keeps the new position otherwise it moves back to its original position.

Step 4: Position Correction: The hunters perform *position correction* based on the hunters current positions and the hunting group consideration rate (**HGCR**). Based on the HGCR, the hunter either keeps the position from the HG or moves a new location based on the distance radius $*Ra*$. The new hunter position is set as shown in equation 4.

$$x_i^j \rightarrow \begin{cases} x_i^j \in HG & with\ p(HGCR) \\ x_i^j = x_i^j \pm Ra & with\ p(1 - HGCR) \end{cases} \quad \ldots\ldots \quad (4)$$

where, x_i^j is the value of the i^{th} decision variable of the j^{th} hunter in the hunting group.

The parameter HGCR denotes the probability of picking the existing position from the HG. HGCR is set at 0.6 in our implementation. *Ra* denoted the fixed radius distance, that the hunter moves and is problem dependent parameter. In this step, the hunter searches in its own local neighborhood to look for a better solution. If a better solution is found, the hunter moves to the new location; otherwise it keeps its own position. In our implementation, Ra takes the value 2.

Step 5: Reorganization: In this step, the hunters are reorganized to a new starting position. This is carried out to prevent the hunting group from being trapped in a local optimum. In our implementation, if there is no improvement in the quality of the solution after 10 iterations of movement towards the leader and position correction, we re-initialize the hunting group as described in step 2.

Step 6: Termination: Steps 3 to 5 are repeated until maximum number of iterations are completed. In this implementation, steps 3 to 5 are repeated for 100 iterations. Once maximum iterations are completed, the HuGS algorithm returns the best Bayesian network learned by the algorithm.

Table 1. Details of the data set used for evaluation

Sr. No.	Dataset	Number of data samples	Number of attributes	Number of classes
1	**balance-scale**	625	5	3
2	**breast-cancer**	286	10	2
3	**car**	1728	7	4
4	**cmc**	1473	10	3
5	**colic**	368	23	3
6	**credit-a**	690	16	2
7	**diabetes**	768	9	2
8	**glass**	214	10	7
9	**heart-c**	303	14	5
10	**heart-h**	294	14	5
11	**heart-statlog**	270	14	2
12	**hepatitis**	155	20	2
13	**ionosphere**	351	35	2
14	**iris**	150	5	2
15	**labor**	57	17	2
16	**lymph**	148	19	4
17	**primary-tumor**	339	18	22
18	**vehicle**	846	19	4
19	**vote**	435	17	2
20	**zoo**	101	18	7

IV. EXPERIMENTS AND RESULTS

We used the Weka machine learning library to implement the HuGS classifier [1]. The HuGS classifier was evaluated using 20 benchmark classification datasets selected from UCI (University of California, Irvine) repository. These datasets were downloaded from the Weka website. The characteristics of these datasets have been summarized in table 1. These datasets represent a wide array of domains and varied characteristics. As the classification algorithm requires a discretized dataset having no missing values, the following preprocessing steps were applied:

- Datasets having continuous attributes were discretized using the *Discretize* filter implemented in Weka. It used a 10-bin unsupervised discretization.
- Datasets with missing values were preprocessed using the *ReplaceMissingValues* unsupervised filter in Weka. This filter replaces missing values with the modes and means from the training data.

The predictive accuracy of the HuGS classifier was compared against other widely used classifiers such as ZeroR (baseline), Naive Bayes, K2, TAN, and Tabu search. ZeroR classifier chooses the most common class in the training set as the predicted class. ZeroR is used to provide a baseline to test the effectiveness of the classifiers.

Table 2 shows the predictive accuracy obtained by the classifiers. The performance of the classifier was evaluated by running a 10-fold cross-validation (CV) on the datasets listed in the table. A 10-fold cross validation divides the dataset into 10 partitions. 9 partitions are used to train the classifier whereas the remaining 1 partition is used to test the performance of the classifier learned. The 10-fold CV is run 10 times wherein each iteration a different partition is used as the test set. Each classifier is evaluated via 10 runs of 10-fold cross validation procedure.

As noted in table 2, overall the classification accuracy of HuGS classifier is best amongst the classifiers compared in this paper. HuGS classifier outperforms all the other classifiers in the experiment for 15 out of the 20 datasets. For the remaining five datasets, its classification accuracy is close to the accuracy of the best classifier. The results can be summarized as below:

- HuGS outperforms ZeroR in classification accuracy for all the datasets.
- HuGS outperforms Naive Bayes in classification accuracy for 19 datasets and tie for 1 dataset.
- HuGS outperforms K2 in classification accuracy for 17 datasets and tie for 1 dataset.
- HuGS outperforms TAN in classification accuracy for 18 datasets.
- HuGS outperforms Tabu in classification accuracy for 18 datasets and tie for 1 dataset.

Table 2. Experimental Results: Predictive Accuracy and Standard Deviation

No.	Dataset	HuGS	ZeroR	NB	K2	TAN	Tabu
1	**balance-scale**	**91.48 ± 2.15**	45.76 ± 0.0	**91.48 ± 2.15**	**91.48 ± 2.15**	85.21 ± 4.22	**91.48 ± 2.15**
2	**breast-cancer**	**75.03 ± 2.45**	70.27 ± 0.0	72.93 ± 1.50	72.76 ± 2.28	69.86 ± 2.61	72.09 ± 1.22
3	**car**	94.00 ± 2.06	70.02 ± 0.0	85.50 ± 3.65	85.63 ± 4.18	**94.30 ± 3.63**	85.76 ± 4.12
4	**cmc**	**52.89 ± 5.99**	42.70 ± 0.0	50.72 ± 2.97	50.75 ± 3.13	50.95 ± 10.38	50.75 ± 3.13
5	**colic**	**71.83 ± 2.68**	61.47 ± 0.0	70.02 ± 1.76	69.37 ± 2.76	69.18 ± 3.96	69.64 ± 1.79
6	**credit-a**	**86.05 ± 4.28**	55.50 ± 0.0	84.68 ± 2.35	84.71 ± 2.12	84.53 ± 3.88	84.91 ± 3.57
7	**diabetes**	**76.64 ± 2.83**	65.10 ± 0.0	75.67 ± 2.394	75.59 ± 3.33	75.06 ± 6.02	76.36 ± 4.50
8	**glass**	**63.41 ± 3.19**	35.51 ± 0.0	57.99 ± 2.88	57.38 ± 2.74	56.72 ± 3.71	56.35 ± 4.47
9	**heart-c**	83.46 ± 2.72	54.45 +0.0	83.30 ± 1.89	**83.76 ± 2.97**	80.79 ± 4.13	82.37 ± 2.91
10	**heart-h**	**83.91 ± 2.11**	63.94 ± 0.0	83.80 ± 1.83	83.33 ± 1.33	83.19 ± 2.95	83.06 ± 1.55
11	**heart-statlog**	**84.15 ± 2.39**	55.55 ± 0.0	83.74 ± 0.99	82.81 ± 2.95	81.51 ± 2.42	82.56 ± 2.73
12	**hepatitis**	**86.45 ± 1.24**	79.35 ± 0.0	84.06 ± 0.48	85.03±1.31	83.81±1.96	83.87±1.53
13	**ionosphere**	**93.16 ± 1.33**	64.10 ± 0.0	90.82 ± 0.78	92.05 ± 2.46	91.11 ± 3.79	91.79 ± 1.87
14	**iris**	95.87 ± 1.31	33.33 ± 0.0	94.4 ± 1.26	94.4 ± 1.17	91.45 ± 1.47	**96.86 ± 0.94**
15	**labor**	**97.54 ± 0.69**	64.91 ± 0.0	96.67 ± 0.99	94.21 ± 1.41	93.86 ±1.17	95.08 ± 1.31
16	**lymph**	**87.29 ± 1.47**	54.73 ± 0.0	86.08 ± 1.17	85.27 ± 1.13	85.27 ± 2.44	85.07 ± 0.73
17	**primary-tumor**	**48.58 ± 2.54**	24.77 ± 0.0	47.17 ± 2.68	47.17 ± 2.65	46.19 ± 4.62	46.63 ± 2.88
18	**vehicle**	72.34 ± 4.78	25.52 ± 0.0	61.09 ± 3.98	61.52 ± 3.50	**72.68 ± 4.86**	62.09 ± 4.52
19	**vote**	93.12 ± 1.28	61.38 ± 0.0	90.23 ± 0.85	**94.71 ± 0.0**	94.67 ± 2.74	94.27 ± 2.99
20	**zoo**	**98.01 ± 0.0**	40.59 ± 0.0	93.96 ± 1.10	96.73 ± 1.06	97.03 ± 1.05	94.75 ± 1.25

V. Conclusion

In this paper, we proposed a novel bio-inspired meta-heuristic algorithm HuGS to learn the network structure for a Bayesian belief network classification algorithm. HuGS has been inspired by the behavior of animals such as wolves who hunt in a group and collaborate to catch a prey. Our experimental results show that HuGS outperforms the other state of the art Bayesian network learning algorithm for Bayesian network classifiers. Also, due to the simplicity of the algorithm, HuGS can be applied for real-world problems.

The HuGS algorithm that has been implemented in this paper, has been learned using static relational data. However, learning a Bayesian network for a high dimensional or streaming data remains an area of active research.

REFERENCES

[1] I. Witten, E. Frank, M. Hall, "Data Mining: Practical Machine Learning Tools and Techniques." [Online]. url: http://www.cs.waikato.ac.nz/ml/weka/book.html.

[2] David Maxwell Chickering, "Learning Bayesian Networks is NP-Complete", Learning from Data: Artificial Intelligence and Statistics V, pp. 121-130, Springer-Verlag, 1996.

[3] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks", Artificial intelligence, vol. 42, pp. 393-405, Elsevier, 1990.

[4] X. S. Yang, "Nature-Inspired Optimization Algorithms", Elsevier, 2014.

[5] X. S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, "Swarm Intelligence and Bio-Inspired Computation", Elsevier, 2013.

[6] R. Oftadeh, M.J. Mahjoob, M. Shariatpanahi, "A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search", vol. 60, pp. 2087-98, Computers Mathematics with Applications, 2010.

[7] R. Tang, S. Fong, Xin-She Yang, S. Deb, "Wolf search algorithm with ephemeral memory", Seventh International Conference on Digital Information Management, 2012.

[8] T. Rui, S. Fong, Xin-She Yang, S. Deb, "Nature-Inspired Clustering Algorithms for Web Intelligence Data", Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on, 2012.

[9] S. Fong, K. Lan, R. Wong, "Classifying human voices by using hybrid SFX time-series preprocessing and ensemble feature selection", BioMed research international, 2013.

[10] S. Fong, Xin-She Yang, S. Deb, "Swarm Search for Feature Selection in Classification",Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on, 2013.

[11] W. Husheng, Z. Fengming, "A uncultivated wolf pack algorithm for high-dimensional functions and its application in parameters optimization of PID controller", Evolutionary Computation (CEC), 2014 IEEE Congress on, 2014.
[12] N. Friedman, D. Geiger, M. Goldszmidt, "Bayesian network classifiers", Machine learning, 1997.
[13] G. F. Cooper, E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data", Machine learning, 1992.
[14] N. Friedman, D. Geiger, M. Goldszmidt, "Bayesian Network Classifiers", Machine learning, 1997.
[15] J. Cheng, R. Greiner, "Comparing Bayesian network classifiers", Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, 1999.
[16] K. Zare, S.M. Hashemi, "A solution to transmission-constrained unit commitment using hunting search algorithm", Environment and Electrical Engineering (EEEIC), 2012 11th International Conference on, 2012.
[17] B. Naderi, M. Khalili, A. A. Khamseh, "Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines, International Journal of Production Research, 2014.
[18] E. Dougan, F. Erdal, "Hunting search algorithm based design optimization of steel cellular beams", Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion, 2013.
[19] K. Waiyakan, P. Luangpaiboon, "Heat Treatment Process Optimization Using Hunting Search and Ant Sense on Path of Steepest Ascent", Applied Mechanics and Materials, 2012.
[20] D. Martens, B. Baesens, T. Fawcett, "Editorial survey: swarm intelligence for data mining", Machine Learning, 2011.

