Information Sciences Letters
An International Journal

http://dx.doi.org/10.18576/isl/121030

Improved Henry gas optimization for predicting high-income Factors

*S. EL-gazzar**, *H. Abdel-kader and A. Haroon*

Department of Information Systems, Faculty of Computer Science and Information Systems, Menoufiya University, Shebin El-Koom, Egypt

Received: 22 Jul. 2023, Revised: 8 Aug. 2023, Accepted: 9 Sep. 2023.
Published online: 1 Oct. 2023.

Abstract: The Henry gas solubility optimization (HGSO) is a meta-heuristic algorithm based on Henry's law. In this paper, β–hill climbing operator is introduced to enhance the ability of the local search, which improves the shortcoming of the original HGSO algorithm. The improved Henry gas solubility optimization algorithm (βHC-HGSO) is based on the β -hill climbing local search, which is used for selecting a subset of relevant features for high income to improve the classification accuracy. The random forest (RF) expert system was employed to explain the performance of the proposed algorithm. According to empirical research, the performance of the improved Henry gas solubility(βHC-HGSO) is better than the original algorithm.

Keywords: Feature selection, β–hill climbing, Henry gas solubility optimization (βHC-HGSO), prediction.

1 Introduction

Machine learning is a rapidly developing field for capturing data from existing datasets. It depends on a learning algorithm for classification, regression, clustering, or time-series prediction to get a result. In the classification task, the target is to predict the class of each sample in the dataset. To complete this task, two major phases are typically used: training and testing. In the training phase, the objective is to build a classifier in the form of a function that matches each record in the training set (samples with known classes) to its corresponding class. While in the testing phase, the aim is to assess the ability of the classifier to perfectly predict the unknown samples of the test set (samples with unknown classes). As the dimensionality of a dataset increases, the classification task becomes more complex and computationally expensive.

To handle this issue, we need to use feature selection (FS). Feature selection is a way in machine learning to find the best set of features for building optimized models. Thus, FS succeeded in obtaining the smallest subset of features while simultaneously maintaining the highest classification accuracy. Real-world applications of FS may include medical diagnosis, bioinformatics, fault detection, text mining, and many others.

There are three types of FS methods: filter, wrapper, and embedded. In the filter methods [1], correlations between the features are considered in the evaluation process, and no external evaluators are involved. The classification model is trained using the available attributes of a dataset in the embedded methods, and the results are used to evaluate the correlation of each attribute.

In terms of classification accuracy, wrapper approaches surpass filter methods. Using the wrapper method, numerous approaches to the FS problem have been presented over time, including greedy search, heuristic search, random search, and the exhaustive search method. Each of these methods has its own set of factors that influence the method's overall performance. Due to various advantages, meta-heuristics have drawn the attention of researchers to solve optimization problems. i) They are adaptive to dynamic changes; ii) They have the ability to self-organize; iii) They do not require any specific mathematical properties; iv) They can evaluate numerous solutions simultaneously; v) They are widely used in practice; and vi) They have frequently proven to be more trustworthy than traditional approaches.

Recently, several metaheuristic-based techniques for solving the FS problem have been proposed. Some of the most well-known works in this field are: particle swarm optimization (PSO) [2], harmony search (HS) [3], artificial bee colony (ABC) [4], and ant colony optimization (ACO) [5], while some of the more recent and promising methods are: grey wolf optimizer (GWO) [6], grasshopper optimization algorithm (GOA) [7], whale optimization algorithm (WOA) [8], salp swarm algorithm (SSA) [9],gravitational search algorithm (GSA) [10]. and Henry gas solubility optimization

*Corresponding author e-mail: shadia_elgazzar@hotmail.com

© 2023 NSP
Natural Sciences Publishing Cor.

algorithm (HGSO) [11].

Meta-heuristic algorithms produce promising results when applied to FS problems; however, a significant question remains as to whether more optimization approaches are required to achieve even better results.

In this regard, we present an improved Henry gas solubility optimization algorithm based on β-Hill climbing local search to compete with the well-known state-of-the-art optimization algorithms in this field. This study's main contribution is:

- An improved Henry gas solubility optimization algorithm is proposed. An optimization technique known as the "hill climbing method" can create a search trajectory in the search space that leads to the local optima, β-operator is utilized in hill climbing to control the balance between exploration and exploitation during the search.
- The improved algorithm adds a new search method. In this way, the disadvantages of the single search method of the original algorithm are overcome, and the accuracy of the optimal solution will be greatly improved after the second search.
- The effect of a classifier-based improved HGSO, can be measured using a random forest classifier (RF) and compared to the same classifier-based original HGSO.

The remainder of this paper is constructed as follows: Section 2 represents the background. Related work In Section 3, the suggested approach is explained in Section 4. Section 5 displays the experimental results. In Section 6, conclusions are presented.

2. Background

Economic data (considered big data) is known for its complexity. The data comes in various structures and sizes. It has outliers and dependence between independent variables, in addition to its high dimensions. The modern algorithms required to work with large datasets add a level of complexity to inference and require different approaches to model fitting. To predict factors that influence extremely high incomes for individuals (based on a large set of personal factors), we need to use feature selection methods to exclude the unimportant factors by finding the relevance of features based on a specified classifier. There are many feature selection methods that give good results, but we need to achieve better results.

This section provides the background and basic concepts of HGSO, and the random forest classifier (RF).

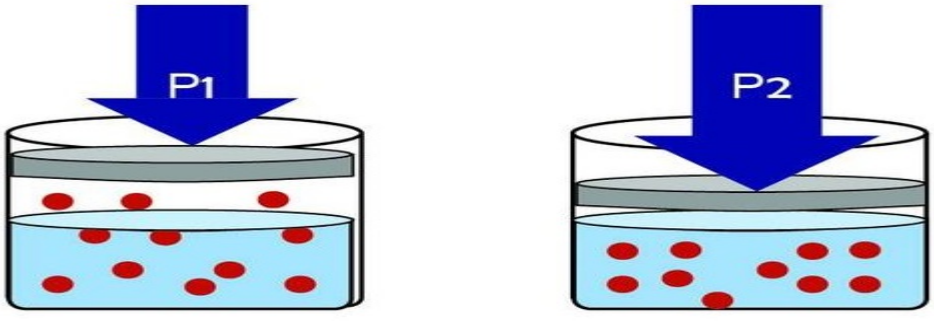
2.1. Henry Gas Solubility Optimization (HGSO) Algorithm

Henry's law of gases serves as the foundation for the Henry Gas Solubility Optimization Algorithm (HGSO), a physical-based algorithm. The law explains the phenomenon of gas solubility in a liquid at a particular pressure. Figure 1 illustrates the solubility of gas particles under two different pressures. Based on the above theory, a mathematical model can be formulated to construct the HGSO algorithm as follows:

In the initialization process, the population of candidate solutions with *N* gas particles is initialized as in Equation (1).

$$x_i^0 = lb_i + \ rand_i \times (ub_i + lb_i) \quad (1)$$

Where x_i^0 is the initial position of the *i*th gas particle, the lower and upper bounds of hyperspace related to the *i*th candidate solution are denoted by lb_{i} and ub_{i}, respectively.



a) A saturated solution of a gas is in equilibrium
b) If the pressure is increased to p2 the volume of the gas at pressure p1 decreases

Fig. 1: Gas particles dissolve into a liquid under partial pressure

© 2023 NSP
Natural Sciences Publishing Cor.

In Equation (1), $rand_i$ is a randomly generated real value in the range [0, 1]. As each gas particle has some properties, these are also initialized using Equation (2).

$$H_j^0 = l_1 + rand_1, p_{i.j}^0 = l_2 \times rand_2, c_j^0 = l_3 \times rand_3 \quad (2)$$

Where H_j^0 is the initial value of Henry's constant for type, $p_{i,j}^0$ represents the initial value of partial pressure of gas *i* in cluster *j*, and C_j^0 represents the initial constant value of type *j*. In Equation (2), constants l_1,l_2, and l_3 with values of 5E-02, 100, and 1E-02, respectively.

Clustering, for each gas type, the population of gas particles is divided into k clusters. Each cluster has a unique Henry's constant value (*Hj*).

Evaluation, the fitness value of the gas particles in each cluster is evaluated to assign the best cluster $X_{j,best}$. All candidate solutions are ranked in order to find the global best solution X_{best} in the entire population.

Update Henry's Coefficient, as the pressure on gas particles changes during each iteration, it is critical to update Henry's coefficient H_j^{t+1} using Equation (3).

$$H_j^{t+1} = H_j^t \times exp\left[-c_j \times \left(\frac{1}{T^t} - \frac{1}{T^\theta}\right)\right]. T^t = \exp\left(\frac{-t}{t_{max}}\right) \quad (3)$$

Where H_j^t Henry's coefficient for cluster *j* in iteration *t*, T^t refers to temperature at iteration *t*, T^θ is a constant with value 298.15, and the maximum number of iterations is t_{max} .

Update Solubility, during the *t*th iteration, the solubility $S_{i,j}^t$ of the *i*th gas particle in the *j*th cluster must be updated using Equation (4):

$$s_{i,j}^t = \mathrm{K} \times H_j^{t+1} \times P_{i,j}^t \quad (4)$$

Where *K* is a constant, and $P_{i,j}^t$ is the partial pressure on *i*th gas particle in *j*th cluster.

Update Position, the position of the *i*th gas particle in the *j*th cluster for iteration *t* = 1 + 1 is updated using Equation (5):

$$\mathrm{x}_{\mathrm{i,j}}^{\mathrm{t+1}} = \mathrm{x}_{\mathrm{i,j}}^{\mathrm{t}} + \mathrm{F} \times \mathrm{rand}_1 \times \gamma \times \left(\mathrm{x}_{\mathrm{j.best}} - \mathrm{x}_{\mathrm{i,j}}^{\mathrm{t}}\right) + \mathrm{F} \times \mathrm{rand}_2 \times \alpha \times \left(\mathrm{S}_{\mathrm{i,j}}^{\mathrm{t}} \times \mathrm{X}_{\mathrm{best}} - \mathrm{X}_{\mathrm{i,j}}^{\mathrm{t}}\right),$$
$$\text{Where}\gamma = \beta \times \exp\left(-\frac{\mathrm{F}_{\mathrm{best}+\epsilon}^{\mathrm{t}}}{\mathrm{F}_{\mathrm{i,j}+\epsilon}^{\mathrm{t}}}\right). \epsilon = 0.05 \quad (5)$$

Where *F* is used to control search direction by flagging, γ is the ability of a gas particle with respect to its cluster, and α is the influence of other gas particles on *i*th particle. In Equation (5), $rand_1$ and $rand_2$ are two different randomly generated real values between [0, 1], and ε is a small value to avoid the divide by zero error.

Escape from local optimum: HGSO selects the worst solutions, as in Equation (6), for re-initialization in order to implement the strategy of avoiding local optima problems:

$$\mathrm{N_w} = \mathrm{N} \times [\mathrm{rand} \times (\mathrm{C_2} - \mathrm{C_1}) + \mathrm{C_1}], \mathrm{C_1} = 0.1 \text{ and } \mathrm{C_2} = 0.2 \quad (6)$$

Where *N* denotes the population size and *rand* denotes a random number between [0, 1]. Equation (1) is used to re-initialize the position of the worst solutions chosen in this step.

2.2. Classification algorithm

In this study, the machine learning (ML) algorithm Random Forest (RF) was used as a classification algorithm during the FS process to evaluate the precision and quality of solutions. The aforementioned machine learning algorithm was used in all experiments because it's simple and easy to implement, as well as very useful in locating the preferred subset of features when compared to other complex supervised ML methods. The classifier used in this study is described in the following section.

2.2.1 Random Forest (RF)

The Random Forest is an ensemble machine learning algorithm that makes decisions by combining the power of multiple decision trees [12]. The algorithm generates decision trees from randomly selected data samples and gets predictions (voting) from each tree. The random forest algorithm combines the voting from multiple decision trees to generate the final solution by averaging all the votes. This process of combining the output of multiple individual models is called "Ensemble Learning". Random forest is often found to be the most accurate learning algorithm to date.

© 2023 NSP
Natural Sciences Publishing Cor.

Algorithm 1 illustrates the pseudocode of the Random Forest.

Algorithm 1 pseudo-code of RF algorithm

Precondition: A training set S: = (x1, y1)… (x_n +y_n), features F and number of trees in forest B.

```
1. function RANDOMFOREST ( S,F )
2.   H ← ∅
3.   for i ∈ 1……..B do
4.       S^(i) ← A bootstrap sample from S
5.       h_i ← RANDOMIZED TREE LEARN( S^(i) , F)
6.       H ← H ∪ {h_i}
7.   end for
8.   return H
9. end function
10. Function RANDOMIZED TREE LEARN ( S,F )
11.     At each node :
12.       f ← very small subset of F
13.       Split on best feature in f
14.     return the learned tree
15. end function
```

The algorithm steps are as follows: To begin, we take a bootstrap sample from S, where S (i) is the *ith* bootstrap for each tree in the forest. To modify the algorithm, at each node of the tree, we randomly select some subset of the features f ⊆ F. where F is the set of features. The node then splits on the best feature in f instead of F, f is a lot smaller than F. Then we use the modified algorithm to learn the decision-tree. The most computationally expensive aspect of decision tree learning is deciding which features to split. The speed of the tree's learning can be increased by reducing the number of features.

3. Related Work

HGSO is a natural science-inspired algorithm that uses Henry gas solubility law to solve global optimization problems. The main changes of premature convergence and a poor balance of exploration and exploitation remain, which means that solving some complex optimization problems is still difficult. In order to make up for the shortcomings of the HGSO algorithm, some efforts have been made in terms of real-world problems and theoretical research. Li et al.[13] produce the Lévy motion-based Henry gas solubility optimization algorithm (Lévy-HGSO) and the Brown motion-based Henry gas solubility optimization algorithm (Brown-HGSO). Mohammadi et al. [14] propose an algorithm named Quantum HGSO (QHGSO) algorithm. The proposed changes improve HGSO's ability to create a balance between exploitation and exploration for a better investigation of the solution space. Hashim et al.[15] present an improved HGSO algorithm for solving the DNA motif discovery problem. Bi et al. [16] propose improved Henry gas solubility optimization with dynamic opposite learning, the sine cosine factor, conversion probability, and an interval contraction strategy. Abd Elaziz. et.al [17] present a modified Henry gas solubility optimization (HGSO) that is based on the whale optimization algorithm (WOA) and comprehensive opposition-based learning (COBL) for optimum task scheduling (HGSWC). Xie et al. [18] propose an improved Henry gas solubility optimization algorithm (HHOHGSO) based on the Harris Hawk optimization.

4. Improved HGSO algorithm based on β- Hill climbing local search

4.1 β- Hill climbing

In Henry Gas Solubility Optimization (HGSO), when the gas particle's position is updated, the searching strategy for the optimal solution is quite simple, which results in low search precision. This study tries to discover other operators with strong searching abilities to compensate for the initial particle movement strategy of the HGSO algorithm.

In this paper, β–hill climbing operator is introduced into the HGSO algorithm. β–hill climbing [19] is the simplest form of the local search-based methods that use an intelligent stochastic strategy to define systematically different search space regions. In β–hill climbing, a new explorative operator called β has been utilized based on an idea inspired by the

© 2023 NSP
Natural Sciences Publishing Cor.

uniform mutation operator of the Genetic Algorithm (GA).

In this search strategy, at each iteration, the search space of the current solution will be defined based on the function N, and unbounded search space will be defined based on the β operator.

The β -hill climbing starts with a random solution $x = (x_1, x_2 \ldots . x_N)$. It generates a new solution iteratively $x' = (x'_1, x'_2, \ldots x'_N)$ based on two operators: neighborhood navigation (i.e., N-operator) and β operator.

The function improving (N(x)) is used with the 'random walk' acceptance rule in the N-operator stage, where a random neighboring solution of the solution *x* is adopted in each iteration as follows:

$$x'_i = x_i \pm \mathrm{U}(0,1) \times \mathrm{bw} \;\; \exists! \in [1, \mathrm{N}] \tag{7}$$

Note that i is randomly selected from the dimensionality range,$i \in [1,2, \ldots . . N]$.The bandwidth between the current and new values is specified by the parameter *bw*.

In β operator stage, the variables of the new solution are assigned values based on the existing values of the current solution or randomly from the available range with a probability of β where $\beta \in [0,1]$ as follows:

$$\mathrm{X}'_\mathrm{i} \leftarrow \begin{cases} \mathrm{x}_r & rnd \leq \beta \\ \mathrm{x}_i & \text{otherwise} \end{cases} \tag{8}$$

where $x_r \in x_i$ is the possible range for the decision variable x'_i and *rnd* generates a uniform random number between 0 and 1.

Two operators achieve the convergence toward the optimal solution in the β-hill climbing: neighborhood navigation (N operator) and β-operator. The N-operator navigates the neighboring solutions of the current one and randomly selects one with a better objective value. In β-operator, the convergence can be achieved by constructing a controlled portion of the current solution, and therefore, the convergence rate could be accelerated. The β-operator can be the source of exploration, while the N-operator can be considered the source of exploitation. In terms of search space navigation, β-operator climbing is able to jump from one search space region to another using β-operator, which can be thought of as a source of exploration. Figure 2 illustrates the flowchart of β-hill climbing.

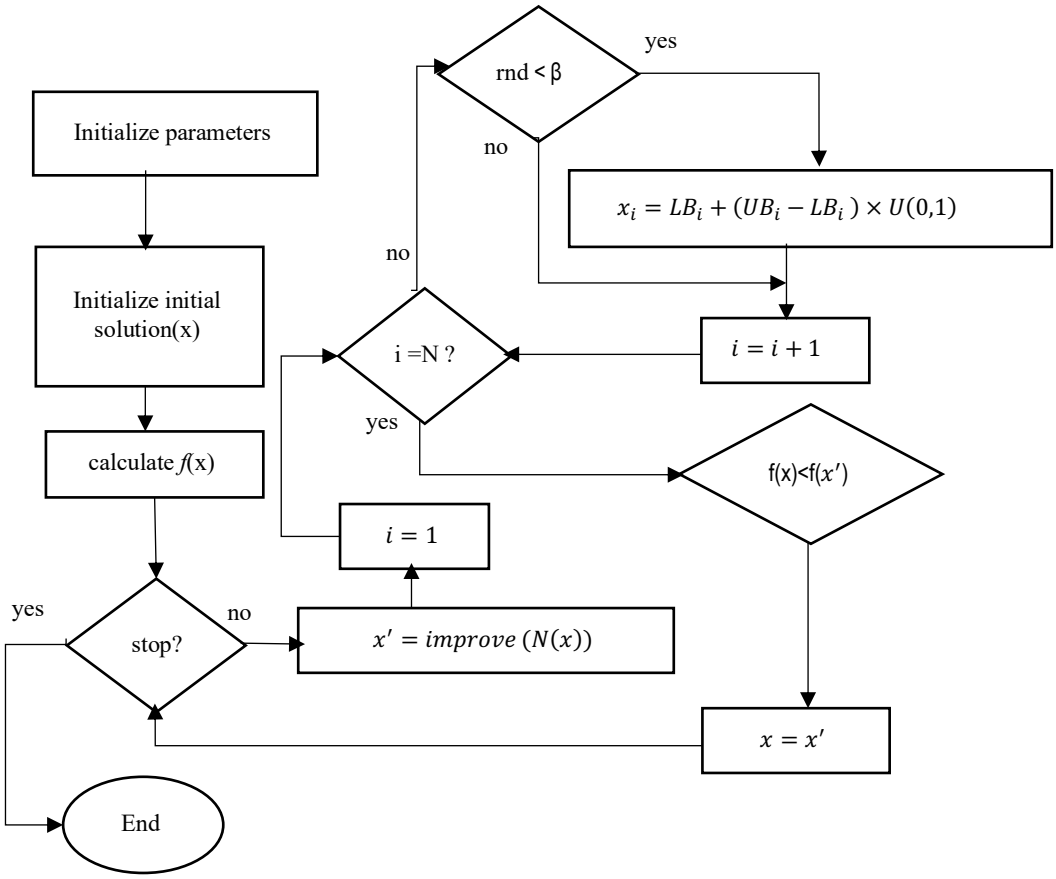


Fig. 2: Flowchart of β-hill climbing

© 2023 NSP
Natural Sciences Publishing Cor.

Algorithm 2 illustrates the pseudocode of β hill climbing-HGSO

Algorithm 2 pseudo-code of β hill climbing-HGSO

1. **Initialization**: Number of gas particles N and types i, H_j^0 . $P_{i,j}^0$. $C_j^0 . l_1 . l_2$ (i=1,....N,j=1...n) Using Eq. (1).Eq. (2).
2. Divide gas particles into number of gas types (cluster) with the Henry's constant value (H_j).
3. for t =1: the maximum number of iterations
4. Calculate the fitness value of each particle.
5. Evaluate the best particleX_{ibest} of each cluster and the best particle X_{best} of all particles according to the fitness value.
6. Update henry's coefficient of each gas type Using Eq. (3).
7. Update solubility of each gas particle using Eq. (4).
8. Update the positions of gas particles using Eq. (5).
9. if t ≤ maximum iterations
10. Update the positions of all particles using Eq(8).
11. else
12. Update the positions of all particles using Eq(7).
13. if end
14. Rank and select the number for worst gas particles using Eq. (6).
15. Update the positions of worst gas particles using Eq.(1)
16. t = t+1.
17. for end
18. return best gas particles X_{best}. and its Fitness value

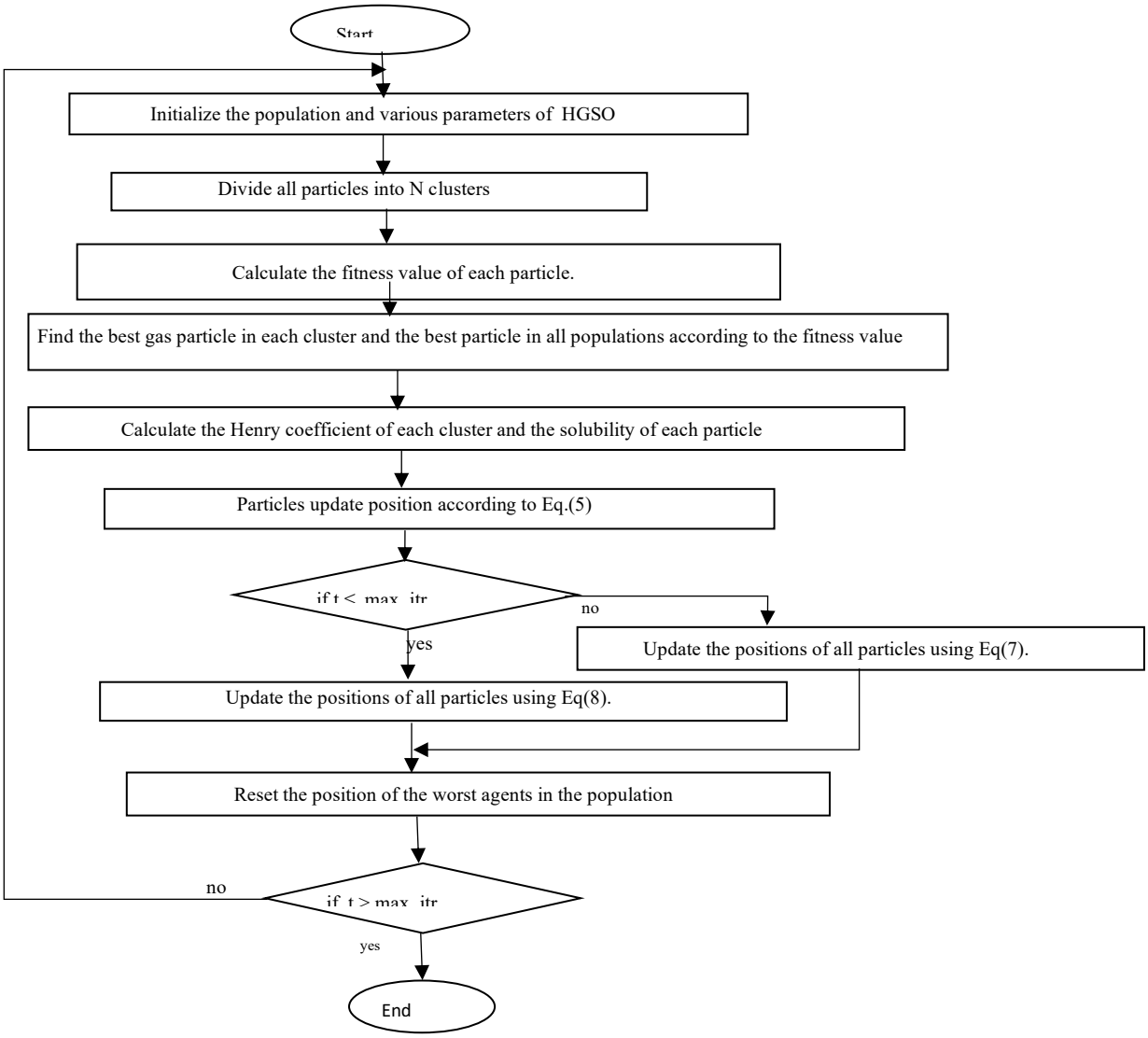


Fig. 3: Flowchart of improved HGSO algorithm

© 2023 NSP
Natural Sciences Publishing Cor.

4.2 Classification

After completing the improved HGSO process, only the features with values corresponding to one's in x_{best} are retained in the original data. A holdout strategy was used for classification, as the dataset was randomly divided into a training set (80%) and a testing set (20%). The RF (estimators = 50) algorithm evaluates accuracy using a testing set. It is worth noting that the experiment was repeated 30 times in order to obtain meaningful results. The following figure depicts the prediction model's framework.

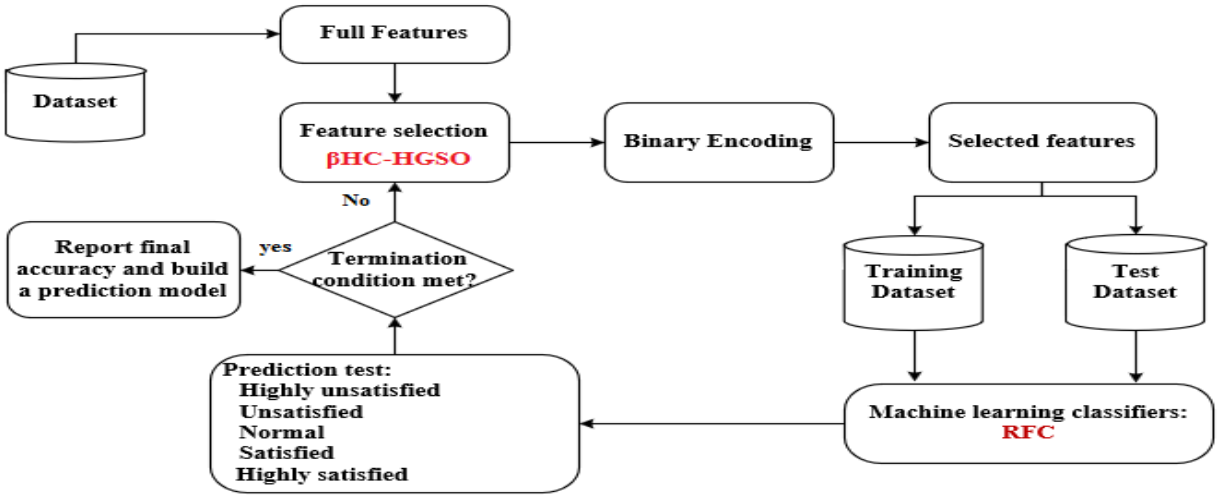


Fig. 4: the prediction model's framework

4.3. Computational complexity

Note that the time needed to use β-hill climbing to solve any optimization problem depends on the complexity of the objective function and the number of iterations needed to converge. Therefore, unlike other algorithmic solutions built for classic problems like sorting or searching, the time complexity of this type of problem cannot be easily measured in advance. Therefore, the time complexity of the proposed algorithm cannot be generalized.

5. Experimental results and analysis

This section highlights the experiments carried out on the proposed HGSO algorithm to validate its efficiency. Parameter settings, datasets used, and performance measures to validate the proposed algorithm are illustrated in this section.

5.1. Dataset description

In this study, the 5-year American Community Survey data is used to verify the performance of the proposed method. The ACS is a continuing, obligatory survey by the United States Census Bureau. The available data on ACS is called 'PUMS', or 'Public Use Microsample Data'.

In this paper, the biggest 5-year PUMS set (concerning years 2015-2019) is used for modelling. The dataset is freely accessible through the US Census portal 'www.census.gov'.

Table 1: description of the dataset

Number of features	Number of instances	Number of classes
22	100000	2

5.2 Performance metrics

In this study, to evaluate the proposed (βHC-HGSO) performance, the algorithm was executed 30 times to increase the statistical significance of the empirical results. For this purpose, some main performance metrics in the FS problem are used as follows:

- Average accuracy (AVG_{ACC}): The accuracy metric is the correct rate of data classification. Equation (9) calculates the average classification accuracy (AVG_{ACC}) obtained by running the algorithm independently 30 times.

$$AVG_{ACC} = \frac{1}{M}\frac{1}{N_s}\sum_{k+1}^{M}\sum_{r=1}^{N_s}(C_{r} == L_{r}) \quad (9)$$

© 2023 NSP
Natural Sciences Publishing Cor.

Where M equals the number of runs, Ns **represents** the size of samples in the test dataset; C_r and L_r denote the classifier output label and the reference class label for sample r, respectively.

- Average fitness value (AVG_{Fit}): This metric evaluates algorithm performance by measuring the average fitness value obtained by executing the proposed algorithm independently 30 times, which defines the relationship between minimizing the classification error rate and reducing the number of selected features, as shown in Equation (8). The best value is represented by the lower value, as shown in Equation (10).

$$AVG_{Fit} = \frac{1}{M}\sum_{k=1}^{M} Fit_*^k \quad (10)$$

Where M equals the number of runs and Fit_*^k represents the optimal fitness value obtained from K^{th} run.

- Average size of selected features (AVG_{size}): This represents the average size (or the feature selection ratio) of the selected features as in Equation (11).

$$AVG_{size} = \frac{1}{M}\sum_{k=1}^{M} d_*^k \quad (11)$$

Using Equation (12), we can calculate the overall selection ratio, which corresponds to the ratio between the size of the selected features d and the total size of features D in the original dataset.

$$Overall_{Select_Ratio} = \frac{1}{M}\sum_{k=1}^{M} \frac{d_*^k}{D} \quad (12)$$

Where d_*^k is the number of selected features in the best solution for K^{th} run, and D the total number of features in the original dataset.

- Standard deviation (STD): As for the aforementioned results, the final average results obtained in the 30 independent runs for each algorithm are evaluated in terms of stability as shown in Equation (13).

$$STD_\gamma = \sqrt{\frac{1}{M}\sum_{k+1}^{M}(\gamma_{*-}^k AVG_\gamma)} \quad (13)$$

Where γ denotes the metric to be measured, γ_*^k the value of the metric γ in the k^{th} run, and AVG_γ represents the average of the metric over the 30 independent runs. Note that the STD_γ is calculated for all measures: Accuracy, fitness and number of selected features.

5.3 The performance of βHC-HGSO for RF

The following experiments were carried out to validate the efficiency of the improved HGSO algorithm. These experiments focus on realizing a comparison between βHC-HGSO and HGSO with random forest classifier.

Table 2: accuracy comparison of βHC-HGSO for RF and HGSO-based RF Classifier

model	accuracy			
	best	worst	mean	SD
βHC-HGSO with RF	0.9176	0.9137	0.9156	0.0019
HGSO with RF	0.8934	0.8832	0.8883	0.0061

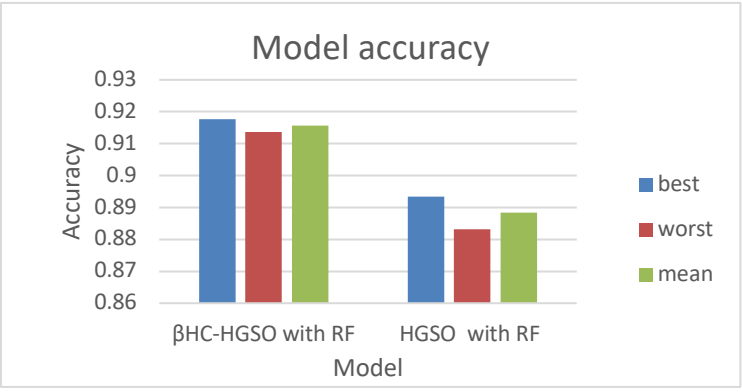


Fig. 5: accuracy comparison of βHC-HGSO for RF and HGSO with RF Classifier

© 2023 NSP
Natural Sciences Publishing Cor.

Table 3: fitness comparison of βHC-HGSO for RF and HGSO-based RF Classifier

model	fitness			
	best	worst	mean	SD
βHC-HGSO with RF	0.0874	0.0940	0.0907	0.0047
HGSO with RF	0.1540	0.1911	0.1750	0.0141

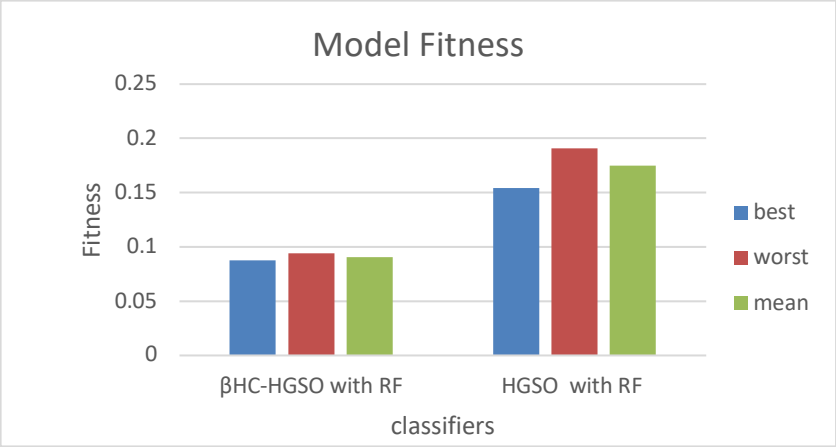


Fig. 6: fitness comparison of βHC-HGSO for RF and HGSO with RF Classifier

Table 4: feature size comparison of βHC-HGSO for RF and HGSO with RF Classifier

model	Feature size			
	best	worst	mean	SD
βHC-HGSO with RF	13	19	14.2	3.969
HGSO with RF	12	15	12.6	1.356

6. Conclusion

In this paper, I propose an improved henry gas optimization algorithm with β-Hill climbing (βHC-HGSO). When compared to other methods, βHC-HGSO performed the search more consistently. HGSO's exploration and exploitation strategies are well implemented because they perform equally well on datasets of varying dimensions, making FS problems versatile. βHC-HGSO with RF efficiently achieved an overall accuracy of 91.76%, while HGSO with RF achieved an overall accuracy of 89.34%. For significantly large datasets, the βHC-HGSO demonstrated a significant advantage.

7. Recommendations

The research recommends using improved henry gas optimization algorithm because of its benefits as The dimensionality reduction algorithm's processing capabilities are enhanced, and data redundancy is better reduced, thanks to the optimization of its parameters. New evolutionary algorithms that have been developed contribute to the emergence of promising new technologies.

Conflicts of Interest Statement

There are no conflicts of interest declared by the authors for the publication of this paper.

Acknowledgment:

The authors are grateful to the anonymous referee for careful checking of the details and for helpful comments that improved this paper.

References

[1] Hancer, E., Xue, B. and Zhang, M. Differential evolution for filter feature selection based on information theory and feature ranking. *Knowledge-Based Systems*, 140, 103-119, (2018)

[2] Xue, B., Zhang, M. and Browne, W. N. Particle swarm optimisation for feature selection in classification: Novel

© 2023 NSP
Natural Sciences Publishing Cor.

initialisation and updating mechanisms. *Applied soft computing*, 18, 261-276, (2014).

[3] Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H. and Mirjalili, S. M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in engineering software*, 114, 163-191, (2017).

[4] Hussain, K., Salleh, M. N. M., Cheng, S., Shi, Y. and Naseem, R. Artificial bee colony algorithm: A component-wise analysis using diversity measurement. *Journal of King Saud University-Computer and Information Sciences*, 32, 794-808, (2020).

[5] Ghimatgar, H., Kazemi, K., Helfroush, M. S. and Aarabi, A. An improved feature selection algorithm based on graph clustering and ant colony optimization. *Knowledge-Based Systems*, 159, 270-285, (2018).

[6] Abdel-Basset, M., El-Shahat, D., El-Henawy, I., De Albuquerque, V. H. C. and Mirjalili, S. A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. *Expert Systems with Applications*, 139, 112824, (2020).

[7] Zakeri, A. and Hokmabadi, A. Efficient feature selection method using real-valued grasshopper optimization algorithm. *Expert Systems with Applications*, 119, 61-72, (2019).

[8] Mafarja, M., Jaber, I., Ahmed, S. and Thaher, T. Whale optimisation algorithm for high-dimensional small-instance feature selection. *International Journal of Parallel, Emergent and Distributed Systems*, 36, 80-96, (2021).

[9] Moayedikia, A., Ong, K.-L., Boo, Y. L., Yeoh, W. G. and Jensen, R. Feature selection for high dimensional imbalanced class data using harmony search. *Engineering Applications of Artificial Intelligence*, 57, 38-49, (2017).

[10] Taradeh, M., Mafarja, M., Heidari, A. A., Faris, H., Aljarah, I., Mirjalili, S. and Fujita, H. An evolutionary gravitational search-based feature selection. *Information Sciences*, 497, 219-239, (2019).

[11] Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W. and Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Future Generation Computer Systems*, 101, 646-667, (2019).

[12] Nguyen, C., Wang, Y. and Nguyen, H. N. Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. *Biomedical Science and Engineering*, 6, 551-560, (2013)

[13] Li, S., Wang, J.-S., Xie, W. and Li, X.-L. An improved Henry gas solubility optimization algorithm based on Lévy flight and Brown motion. *Applied Intelligence*, 52, 12584-12608, (2022).

[14] Mohammadi, D., Abd Elaziz, M., Moghdani, R., Demir, E. and Mirjalili, S. Quantum Henry gas solubility optimization algorithm for global optimization. Engineering with Computers,38, 2329–2348, (2022).

[15] Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S. and Al-Atabany, W. A modified Henry gas solubility optimization for solving motif discovery problem. *Neural Computing and Applications*, 32, 10759-10771, (2020).

[16] Bi, J. and Zhang, Y. An improved Henry gas solubility optimization for optimization tasks. *Applied Intelligence*, 52 (2022), 5966-6006.

[17] Abd Elaziz, M. and Attiya, I. An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing. *Artificial Intelligence Review*, 54 , 3599-3637, (2021).

[18] Xie, W., Xing, C., Wang, J., Guo, S., Guo, M.-W. and Zhu, L.-F. Hybrid henry gas solubility optimization algorithm based on the harris hawk optimization. *Ieee Access*, 8, 144665-144692, (2020) .

[19] Al-Betar, M. A. β-hill climbing: an exploratory local search. *Neural Computing and Applications*, 28, 153-168. (2017),

© 2023 NSP
Natural Sciences Publishing Cor.