Evolutionary Intelligence (2021) 14:357–367
https://doi.org/10.1007/s12065-019-00218-5

SPECIAL ISSUE

Deluge based Genetic Algorithm for feature selection

Ritam Guha[1] · Manosij Ghosh[1] · Souvik Kapri[1] · Sushant Shaw[1] · Shyok Mutsuddi[1] · Vikrant Bhateja[2] · Ram Sarkar[1]

Received: 8 January 2019 / Revised: 23 February 2019 / Accepted: 27 February 2019 / Published online: 7 March 2019

Abstract
Feature selection methods are used to identify and remove irrelevant and redundant attributes from the original feature vector that do not have much contribution to enhance the performance of a predictive model. Meta-heuristic feature selection algorithms, used as a solution to this problem, need to have a good trade-off between exploitation and exploration of the search space. Genetic Algorithm (GA), a popular meta-heuristic algorithm, lacks exploitation capability, which in turn affects the local search ability of the algorithm. Basically, GA uses mutation operation to take care of exploitation which has certain limitations. As a result, GA gets stuck in local optima. To encounter this problem, in the present work, we have intelligently blended the Great Deluge Algorithm (GDA), a local search algorithm, with GA. Here GDA is used in place of mutation operation of the GA. Application of GDA yields a high degree of exploitation through the use of perturbation of candidate solutions. The proposed method is named as Deluge based Genetic Algorithm (DGA). We have applied the DGA on 15 publicly available standard datasets taken from the UCI dataset repository. To show the classifier independent nature of the proposed feature selection method, we have used 3 different classifiers namely K-Nearest Neighbour (KNN), Multi-layer Perceptron (MLP) and Support Vector Machine (SVM). Comparison of DGA has been performed with other contemporary algorithms like the basic version of GA, Particle Swarm Optimisation (PSO), Simulated Annealing (SA) and Histogram based Multi-Objective GA (HMOGA). From the comparison results, it has been observed that DGA performs much better than others in most of the cases. Thus, our main contributions in this paper are introduction of a new variant of GA for FS which uses GDA to strengthen its exploitational ability and application of the proposed method on 15 well-known UCI datasets using KNN, MLP and SVM classifiers.

Keywords Feature selection · Deluge based Genetic Algorithm · Genetic Algorithm · Great deluge algorithm · Metaheuristic · Local search · UCI dataset

1 Introduction

Recent years have seen an exponential growth of various datasets in terms of size. But, in contrast to that the time requirement to process them has become shorter as people need a quick answer to their query. With the advent of machine learning algorithms, this seemingly impossible task has become plausible to the researchers. But most of the real-world datasets contain many irrelevant or redundant information which impedes the processing ability of

✉ Ritam Guha
ritamguha16@gmail.com

Manosij Ghosh
manosij1996@gmail.com

Souvik Kapri
souvikkapri1998@gmail.com

Sushant Shaw
susvicky393@gmail.com

Shyok Mutsuddi
shyokmutsuddi21@gmail.com

Vikrant Bhateja
bhateja.vikrant@gmail.com

Ram Sarkar
raamsarkar@gmail.com

[1] Computer Science and Engineering Department, Jadavpur University, 188, Raja S.C. Mallick Road, Kolkata, West Bengal 700032, India

[2] Electronics and Communication Engineering Department, Shri Ramswaroop Memorial Group of Professional Colleges, Lucknow, UP 226028, India

the machine learning algorithms. Moreover, these irrelevant pieces of information increase the processing time of the learning model. Due to these reasons, researchers have introduced certain pre-processing techniques to allow models to handle even the noisy data with ease [1]. Before feeding the datasets to any machine learning model, these pre-processing techniques refine the datasets based on some parameters in order to reduce the said redundant and/or irrelevant information present therein.

In this regard, it is to be noted that the interpretation of information may vary in different environments. For example, in pattern classification domain various features, used to represent the different patterns in the feature space, define the information set. One of the most used pre-processing techniques related to pattern classification is the Feature Selection (FS) [2] which is a method used to refine the datasets by keeping only the important and relevant features. In this way, FS helps in overcoming some major drawbacks generally found with the raw datasets. Due to the reduction of irrelevant features, it is observed that the machine learning model becomes more effective in terms of both performance (such as recognition or classification ability) and processing time as the dimension of the feature vector becomes less. There are many ways to perform FS. The most basic method is Blind Search (BS) [3] which considers each and every feature combination to decide the optimal subset of features. But this process has an exponential time complexity and hence is not a feasible solution if the number of features in a dataset is large. As an improvement over BS, researchers have introduced various heuristic algorithms [4, 5]. Such algorithms generate a random subset of features and try to improve the same over time to find an optimal solution. Hence, the time required to process the features decreases to a large extent. But heuristic approaches are more problem-specific [6] and often use greedy methods [7] to reach the optimal feature set. However, the more advanced version of FS includes meta-heuristic algorithms [8–11] which are generally problem independent in nature. They do not take advantage of any problem specific parameter. Population-based meta-heuristics, instead of a single subset, start to solve this FS problem by generating a set of subsets (called population). Each subset in this process is known as a candidate solution to the FS problem.

FS methods are broadly classified into three categories namely—filter methods [12–14], wrapper methods [8, 9, 15] and embedded methods [16–18]. Among these, filter methods do not consult with any learning algorithm to select the relevant features. They rather depend on the intrinsic or statistical properties of features. This reduces the time requirement of the FS process but as this works without the supervision of a learning algorithm, it is generally unable to produce good results. On the other hand, wrapper methods take the help of a learning algorithm to find out the optimal subset of features. Although wrapper methods need more time when compared to filter methods, they can generally produce good results more often. Embedded methods combine the advantages of both filter and wrapper methods. They consult both intrinsic properties of features and a learning algorithm to form the best candidate solutions. Even though it seems like an embedded method is the best approach among all, they have their own drawbacks. If the trade-off between filter and wrapper constituents of an embedded method is not established properly, it may bring adverse effects on the overall FS model. That's why in this paper, we have proposed a novel wrapper-based FS framework based on a popular evolutionary algorithm called Genetic Algorithm (GA) [1, 19, 20] which can be used to solve any FS problem.

GA is a meta-heuristic algorithm that is conceptualized from Charles Darwin's theory of natural evolution. This popular algorithm basically imitates the procedure of natural selection where the fittest individuals get chosen for reproduction of the offspring of the next generation. This implies that GA works exactly in the way child chromosomes are created from the parents' chromosomes. During this reproduction procedure, GA applies two operations on the chromosomes namely - crossover and mutation. Through the formation of child chromosomes, GA passes the fitter genes onto the next generations. When applied to FS, chromosomes become the candidate solutions which undergo a series of crossover and mutation operations to build the optimal solution. The better solutions are passed to the next iteration and the weaker solutions are discarded. In this way, optimal solutions are obtained after some generations. It is to be noted that any FS/optimisation algorithm should take care of two important aspects of the feature space namely exploitation and exploration in order to generate the optimal solution. Here exploration means how thoroughly the whole search space is covered and exploitation implies the extent of refinement achieved in a particular portion of the search space. Though GA can achieve very good exploration through crossover operation, it lacks extensive exploitation. In general, mutation means some random change or perturbation of the chromosomes. Thus, mutation helps chromosomes of GA to achieve some local search (exploitation) in the search space. Mutation, however, is unable to achieve the full extent of exploitation needed in the algorithm. We have used the Great Deluge Algorithm (GDA) [21] to achieve this perturbation of the candidate solutions. As the name suggests, it follows an analogy that a person who is climbing a hill sometimes moves down in anticipation of reaching a new maximum height to avoid getting wet when the water level rises from the bottom. This very concept is used here to get a better optimal solution in FS. We can consider classification accuracies to be hills. The value of the accuracy represents the height of the hills and hill tops are the optimum values. More the height, more is the accuracy. Now when

GA chromosomes reach a certain hill top, we can say that it gets to a local maximum but that maximum may not be the global one. So, to get to the hill with maximum height, the chromosomes must descend from the current hill top and go on searching around the neighbouring area. This job can be performed using GDA.

Deluge based Genetic Algorithm (DGA) focuses on solving FS problem using a modified version of GA which uses GDA as a scheme for modification. Though GA follows a very simple algorithm, it shows almost equivalent performance like other complex meta-heuristic like Ant Colony Optimisation (ACO) [8], Particle Swarm Optimisation (PSO) [9], Simulated Annealing (SA) [5] etc. This motivates us to design our FS model which is simple in terms of computation but improves the performance of basic GA.

2 Related work

After the introduction of FS to the research community, it has become a hugely accepted pre-processing technique for its ability to reduce redundancy and/or irrelevancy in datasets and problem independent nature. Researchers all over the world have come up with several new algorithms or modified some existing algorithms in the domain of FS. In this section, we have discussed how the usage of GA in FS has evolved and also a few applications of GA in real-world pattern classification problems. Applications of GDA to solve various real-world problems have also been explored.

GA is one of the primary optimisation algorithms employed to perform FS in the literature. Due to its simplicity, it has been widely accepted in many fields of research. Leardi et al. introduced GA in FS domain [22]. Since then many modified versions of GA have been employed in FS. In [15], Yang et al. modified the fitness function to make GA multi-objective. The fitness function consisted of classification accuracy of the feature subset and the cost of achieving that accuracy. Proper trade-offs were used based on the importance of the two parameters to form the solution. In [23], Huang et al. proposed a way to hybridise GA with Mutual Information (MI). Their method also used a combination of filter and wrapper methods to perform FS. They implemented a local search heuristic technique to get rid of the insignificant features selected by the initial random population. The chromosomes were ranked using a filter method but the final performance was evaluated in a wrapper fashion. A conditional MI was computed between the candidate feature and the already-selected features as well as the classes. During the local search process, the candidates were ranked according to the MI values. Classic GA is weak in fine-tuning near local optimum positions. In order to improve the fine-tuning capability of GA Oh et al. proposed a hybrid GA in [24]. After mutation stage of GA, it used a local searching procedure which explored the search space until chromosomes had the desired number of selected features. Siedlecki et al. in [25] introduced the use of GA for FS in automatic pattern classifier design. In [20], a histogram based GA to perform FS on Devanagari numeral datasets was proposed. After performing GA for a number of times their results were combined by drawing a histogram and selecting the features which were above the mean histogram value. Application of GA in microarray data can be seen in [19].

In [21], Dueck et al. proposed a new optimisation heuristic named as GDA. The experimental results showed that the algorithm was as good as Threshold Accepting (TA) [26] in optimisation. Baykasoglu et al. used GDA to solve constrained mechanical design problems in [27]. For the first time, they tried to modify the performance of GDA to solve complex nonlinear optimisation problems by embedding eight different chaotic maps in its neighbourhood. GDA was used multiple times to efficiently solve University Time-Tabling problems. In [28], Obit et al. used an extended version of non-linear GDA to perform University Time-Tabling. The algorithm incorporated tournament selection, mutation and a replacement strategy to modify GDA. McCollum et al. in [29] introduced a two-phase approach incorporating Extended-GD (EGD) and applied it on the datasets presented in 2nd International Timetabling Competition (ITC2007). Their technique consists of a construction phase followed by an improvement phase. EGD was used to improve the solutions as a part of the improvement phase. The results obtained by this method was compared to the results obtained by the winner of the competition (ITC2007). In 2011, Mafarja et al. applied modified GDA to perform attribute reduction [30]. The entire search space was divided into three regions. In each region, the water level was updated based on the quality of the current solution. The procedure achieved decent results when applied to 12 benchmark datasets. In the field of FS, GDA is used to improve local search of various state-of-the-art algorithms. This can be seen in [31] where Badawi et al. used a hybridised version of GA and GDA, Memetic Algorithm (MA), to perform fish classification.

Mutation causes little change in chromosomes since mutation probability is generally low. This leads to a poor exploitation capability in GA and the algorithm may get stuck in a local optimum. An attempt is made to overcome the said problem by looking into the neighbouring area of a candidate feature subset. GDA may allow acceptance of some solutions with lower fitness to reach a higher peak—solution with better fitness. From the literature survey done above, it can be concluded that although GA and GDA have been used separately to solve various real-world problems, but there has been very less research attempts to form a hybridisation of the two for application in FS. This motivates

us to introduce a new approach to unite them. GDA replaces mutation which allows us to perform a more thorough search of the neighbourhood thereby improving exploitation capability of GA.

3 Proposed methodology

This section contains a detailed explanation of our proposed method called DGA. It has already been mentioned that GA is a widely used optimisation algorithm having a constrained exploitation ability. This affects accuracy by disallowing proper search near local optimums i.e. local search is lacking. GDA is a method which allows the solutions to degrade to a certain extent which ultimately helps to achieve greater local search. So, the basic idea is instead of restricting the exploitation of the method, we can perform a better local search to reach the global maximum. This is why we've used GDA to increase the local searching capability of GA.

3.1 Genetic Algorithm

GA [15] incorporates the concept of chromosome manipulation while the transfer of genetic information from parent chromosomes to child chromosomes. GA mainly consists of four key steps i.e. population creation, parent selection, crossover and mutation which resemble the biological process of chromosome formation. At first, GA creates a population of chromosomes which represent initial candidates to become optimal solutions. After selection of parent chromosomes from the population, they undergo crossover to form child chromosomes. These child chromosomes are then mutated which allows for the restoration of lost genes. After all these steps, the off-springs which survive pass on their genetic information to the next generation. GA follows the same procedure in FS. Each candidate solution in GA is represented by a chromosome. Chromosomes are basically vectors of '0's and '1's which depict that the corresponding feature is discarded or selected by that chromosome respectively. Each time from a population of chromosomes, two parent chromosomes are selected. They perform crossover to form two child chromosomes which then undergo mutation. These child chromosomes replace the parent chromosomes if they have better fitness values else their parents are carried over to the next generation.

3.2 GDA

In 1993, Dueck proposed a new local search technique known as GDA [21] which followed an approach also applicable in real life. The basic idea of GDA is that a person climbing a hill sometimes needs to descend to reach a new height in order to avoid getting wet in the rising rain water. The pseudo code of GDA is as follows:

Algorithm: Great Deluge Algorithm

Terminology:

ins : initial solution
cs : current solution
wl : water level
rs : rain speed
qc : quality of cs

```
1.  Start
2.  Initialize ins, wl and rs
3.  Repeat
4.          cs =a stochastic small perturbation of ins
5.          Compute qc = quality (cs)
6.          if (qc > wl)
7.                  ins = cs
8.                  wl = wl + rs
9.          else
10.                 continue
11. Until stopping criterion (Some fixed number of iterations with no significant improvement)
12. End
```

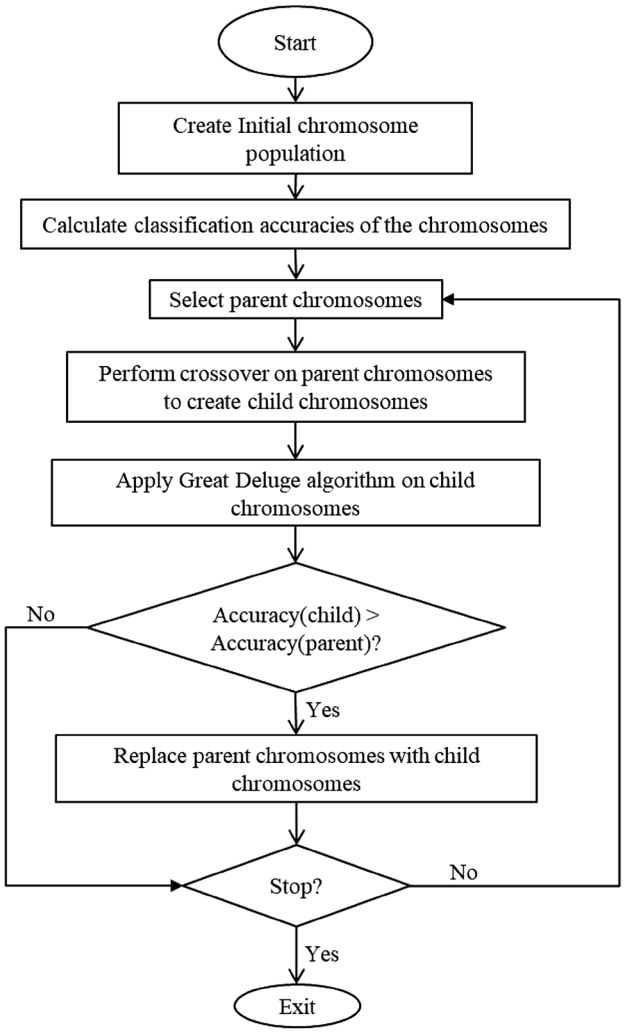


Fig. 1 Flowchart describing the step by step progress of Deluge based Genetic Algorithm

GDA allows degradation of the quality of the candidate solutions to a certain extent in order to achieve a new maximum which was not attainable otherwise. In our proposed method, we have utilised this property of GDA to perform local search efficiently.

3.3 DGA

Our proposed method, DGA, combines GDA with GA in order to address the problems related to basic GA. As mentioned earlier, GA consists of four sub-process—population creation, parent selection, crossover and mutation. Mutation in GA is responsible to achieve local search in the system of solutions but in certain cases, solutions generated by GA suffers from the disadvantage of getting stuck in a local optimum. To avoid this problem, GDA is used to achieve stochastic perturbation of the child chromosomes produced by GA which helps them to circumvent the local maximum to reach the global maximum in the search space. The flowchart of the proposed model is shown in Fig. 1. The algorithm stops when the number of iterations exceed the maximum number of iterations allowed or no change occurs in successive iterations. The maximum number of iterations is taken as 20.

Thus, DGA consists of five steps.

1. Population creation.
2. Parent selection.
3. Crossover.
4. GDA.
5. Child replacement.

The population creation is described in Sect. 3.3.1. The parent selection using roulette wheel is discussed in Sect. 3.3.2. The operations of crossover and GDA are described in Sects. 3.3.4 and 3.3.5 respectively. The replacement of parent chromosomes in the population by the children is described in Sect. 3.3.5.

3.3.1 Population creation

At first population of random chromosomes are created. Chromosomes are vectors of ‘0’s and ‘1’s where ‘1’ represents a feature which gets selected by it and ‘0’ means a feature that is not considered. Each such chromosome represents a feature subset. Thus, the chromosomes are integer vectors with elements in {0,1}. The main objective of the algorithm is to find the most optimal chromosome i.e. high accuracy using a low number of features. The size of population (number of chromosomes) is taken as 20.

3.3.2 Parent selection

After creating a population of chromosomes, GA searches for some pairs of parent chromosomes in order to perform crossover operation. We have used *Roulette Wheel selection* [32] method for selection of parent chromosomes. Each chromosome gets an amount of space on the roulette wheel proportional to its accuracy. A chromosome with higher accuracy gets larger space on the wheel. After placing the chromosomes on the wheel, it is rotated. The wheel has a random pointer which points at the selected chromosome when the wheel stops rotating. As better chromosomes get larger spaces, their chances of selection are more. This is how the roulette wheel selection method works. Using this procedure, two parent chromosomes are selected which then proceed for crossover operation.

3.3.3 Crossover

The selected parent chromosomes then undergo crossover. Uniform crossover [19] is done because of its relative superiority over single-point crossover [33]. With a probability of 0.5, we switch each bit between the two parents. In the end, the child chromosomes are a uniform mixture of the genetic information of the two parents.

3.3.4 GDA

The child chromosomes formed after crossover are then passed through GDA to achieve local search. Each child chromosome undergoes a series of perturbations until they become too weak to be discarded, else it keeps on searching locally around the space to obtain a better solution than its current solution. This significantly enhances the exploitation of search space.

3.3.5 Child replacement

Finally, if the child chromosomes surpass their parents in term of classification accuracy then they are placed in the population (replacing the parents) and carried over to the next generation, otherwise, the parents remain in the population. Note that, we have used the classification model as our fitness function.

4 Results and analysis

This section contains the results obtained by DGA over some well-known datasets. We have also compared these results with some state-of-the-art algorithms which further confirm the applicability of our proposed method. It should be noted that though computation cost of GDA is greater than GA, but it outperforms GA in terms of both accuracy and selection of lower number of features. This justifies the use of GDA over GA in real life applications. The description about the datasets used here is given in Sect. 4.1. The classifier details are provided in Sect. 4.2 and results analysis is reported in Sect. 4.3.

4.1 Dataset description

To test DGA, we have selected 15 datasets from the UCI repository [34]. These datasets can be classified into three categories based on the number of attributes or features present—small, medium and large. For our experimentation, we have taken 4 small, 3 large and 8 medium datasets. The descriptions of the datasets are provided in Table 1.

4.2 Classifier description

As learning algorithms, we have used three popular classifiers namely K-Nearest Neighbour (KNN) [35], Multi-layer Perceptron (MLP) [36] and Support Vector Machine (SVM) [37] for finding classification accuracy of the candidate solutions generated by DGA. The accuracy of the chromosomes calculated using these classifiers are their fitness values.

These three classifiers are selected to show the performance of our proposed method in different classification environments of varying complexity. The results clearly show that our proposed method is independent of the classification model which makes DGA more platform independent.

Table 1 Category-wise (small, medium, large) information on 15 UCI datasets used in the experimentation

Type	Dataset	Alias	Number of attributes	Number of instances	Number of classes
Small	Breast cancer	BC	9	699	2
	Monk1	MK1	6	124	2
	Monk2	MK2	6	124	2
	Monk3	MK3	6	124	2
Medium	Horse	HR	27	368	2
	Ionosphere	IO	34	351	2
	Glass	GL	10	214	7
	Vowel	VO	10	528	11
	Wine	WI	13	178	3
	Zoo	ZO	16	101	7
	Sonar	SO	60	208	2
	Soybean-small	SS	35	47	4
Large	Arrhythmia	ARR	279	452	16
	Hill-Valley	HV	101	606	2
	Madelon	MA	500	4400	2

Table 2 The optimal values obtained for the important parameters (K for KNN, number of neurons for MLP, Kernel for SVM) of the classifier models used in the experimentation over different datasets

Dataset	Classifier		
	KNN	MLP	SVM
	Value of k	Number of Neurons in the hidden layer	Kernel
ARR	3	80	Polynomial
BC	5	110	
GL	10	80	
HV	9	150	
HR	3	150	
IO	3	150	
MA	9	80	
MK1	3	80	
MK2	3	80	
MK3	3	50	
SO	11	80	
SS	3	50	
VO	3	110	
WI	3	50	
ZO	3	80	

We have varied parameters of the classifiers to get the most suitable evaluation environment for the candidate solutions. The best-achieved parameter combinations of the classifiers are provided in Table 2.

4.3 Analysis of the outcomes

This section contains the results obtained by DGA and its comparison with other well-known optimisation algorithms. Here, 4 algorithms selected for the comparison are basic GA [15], PSO [9], SA [5] and Histogram Oriented Multi-Objective Genetic Algorithm (HMOGA) [20]. The detailed experimentation results of the proposed work are depicted in Table 3. The best accuracies obtained are marked bold.

KNN due to its low computation time has been used as our classifier for comparison. To keep uniformity in the process of comparison, we have also evaluated other algorithms (present in the comparison) using KNN classifier. Table 4 contains the comparative results. It can be observed from the table that in comparison to other methods DGA performs better in 9 cases out of 15. The results are considered best if the accuracy is the highest. If the accuracies are equal, the number of features acts as the tie-breaker (lower value is considered as better). From Table 4 it can be observed that for small datasets DGA performs better for only 1 dataset, while for medium DGA outperforms contemporary algorithms in 6 cases out of 8 datasets. For large datasets, DGA achieves the best accuracy for 2 out of 3 datasets. Therefore, it can be concluded that DGA performs better as the number of features (feature dimension) increases.

From the results and the corresponding comparison, we can see that in 9 out of 15 datasets, DGA has outperformed other optimisation algorithms. This clearly proves the applicability of DGA in FS.

Table 3 Classification accuracy obtained after FS by DGA using 3 classifiers (KNN, MLP, SVM) over 15 datasets

Dataset	Classifier					
	KNN		MLP		SVM	
	Accuracy (%)	Number of features	Accuracy (%)	Number of features	Accuracy (%)	Number of features
ARR	63.16	142	**70.39**	131	65.42	145
BC	99.00	5	**99.33**	8	98.33	3
GL	87.14	8	**92.86**	6	88.57	6
HV	55.86	50	58.61	49	**77.11**	88
HR	100.00	15	**100.00**	23	100.00	22
IO	96.69	15	**99.34**	18	95.36	17
MA	61.00	242	60.83	234	**61.17**	238
MK1	94.44	3	**100.00**	3	100.00	3
MK2	78.94	6	**86.11**	6	86.19	6
MK3	97.22	2	**100.00**	3	97.22	2
SO	70.15	24	**89.55**	29	68.66	29
SS	100.00	20	**100.00**	19	100.00	25
VO	93.94	7	**95.02**	8	94.37	8
WI	100.00	5	**100.00**	10	100.00	11
ZO	85.37	8	**87.80**	5	85.37	9

Table 4 Comparison of accuracy obtained by proposed FS model DGA with that of GA, PSO, SA and HMOGA using KNN classifier

Dataset	DGA		GA		PSO		SA		HMOGA	
	Number of features	Accuracy (in %)	Number of features	Accuracy (in %)	Number of features	Accuracy (in %)	Number of features	Accuracy (in %)	Number of features	Accuracy (in %)
ARR	142	63.16	**215**	**67.76**	166	65.78	144	58.55	122	56.58
BC	**5**	**99.00**	6	98.79	6	98.32	5	96.99	3	96.32
GL	**8**	**87.14**	7	85.71	7	82.86	9	80.00	3	54.29
HV	**50**	**55.86**	73	54.76	73	55.31	42	51.65	54	51.10
HR	**15**	**100.00**	21	100.00	18	100.00	18	61.76	14	97.05
IO	**15**	**96.69**	25	93.37	20	96.12	18	92.05	17	93.38
MA	**242**	**61.00**	345	56.33	282	57.50	260	51.83	240	60.33
MK1	3	94.44	4	92.59	**3**	**97.22**	3	72.33	3	83.33
MK2	6	78.94	**6**	**81.94**	6	74.31	3	56.02	2	55.09
MK3	2	97.22	**3**	**100.00**	3	97.22	4	90.28	2	97.22
SO	24	70.15	48	76.11	**34**	**80.59**	31	49.25	27	67.16
SS	20	100.00	24	100.00	**14**	**100.00**	13	92.86	19	85.71
VO	**7**	**93.94**	9	91.77	8	89.83	6	64.50	3	66.88
WI	**5**	**100.00**	11	100.00	8	100.00	7	97.87	5	70.21
ZO	**8**	**85.37**	12	85.37	**8**	**85.37**	6	68.29	8	82.93

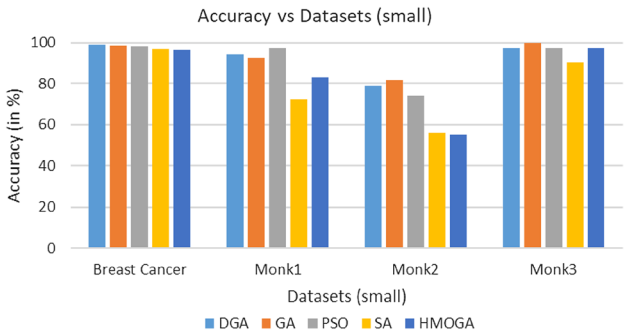


Fig. 2 Comparison of the classification accuracy obtained by DGA with that of GA, PSO and HMOGA over small category datasets

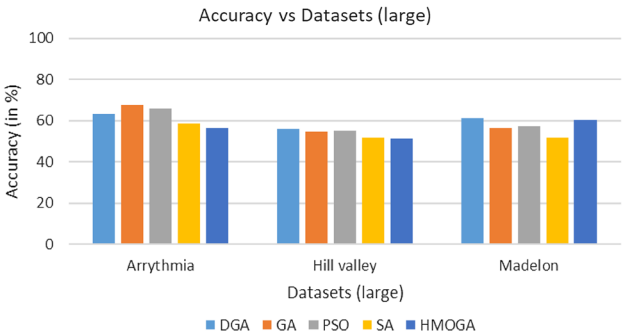


Fig. 4 Comparison of the classification accuracy obtained by DGA with that of GA, PSO and HMOGA over large category datasets

Figures 2, 3, 4, 5, 6 and 7 represent graphical comparisons of DGA with other methods with respect to a number of selected features as well as accuracy over small, medium and large datasets. Figures 2, 3, and 4 show the graphs representing the best accuracy obtained by different FS algorithms over small, medium and large datasets. Similarly, Figs. 5, 6, and 7 show the graphs representing the percentage of features selected for the best solution by the model over small, medium and large datasets.

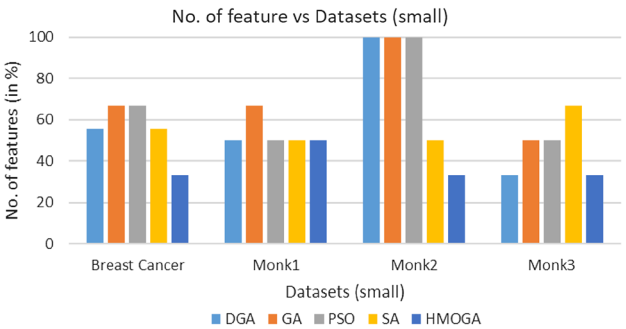
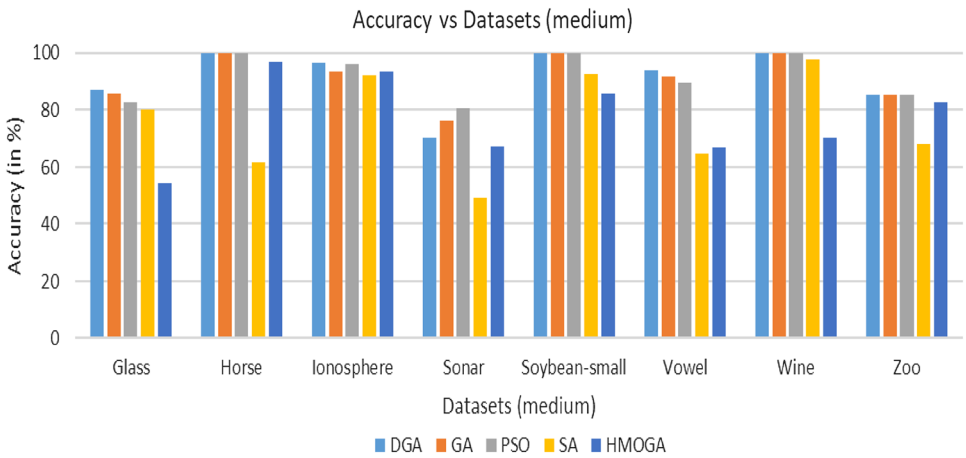


Fig. 5 Comparison of the percentage of features selected by DGA with that of GA, PSO and HMOGA over small category datasets

5 Conclusion

GA is one of the most fundamental optimisation algorithms which has been applied to solve FS problem over the years. Its simple yet robust nature has made it one of the most popular FS techniques. However, the lack of exploitation for GA affects its population. That is why our motive is to build a hybridised model which uses GA but keeping it simple. So, we have chosen GDA to get rid of the disadvantages GA has and created a hybridised model called DGA. GDA's local searching strategy helps GA to move around the local optimum and gradually reach global optimum. Thus, the overall model becomes more efficient in finding a global optimum than basic GA. As GDA is also a very simple algorithm, the complexity of the overall model does not increase too much. The comparison of results of DGA over the selected UCI datasets with that of other heuristic, meta-heuristic and hybridised models speaks for the applicability of our model. In the future, we plan to extend GDA's local

Fig. 3 Comparison of the classification accuracy obtained by DGA with that of GA, PSO and HMOGA over medium category datasets



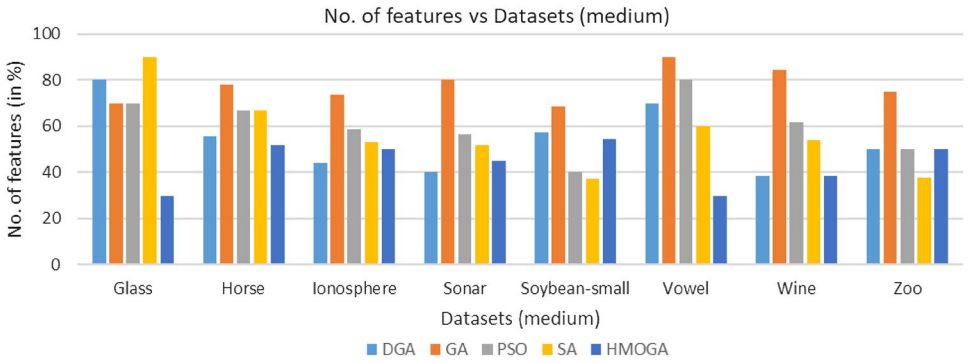


Fig. 6 Comparison of the percentage of features selected by DGA with that of GA, PSO and HMOGA over medium category datasets

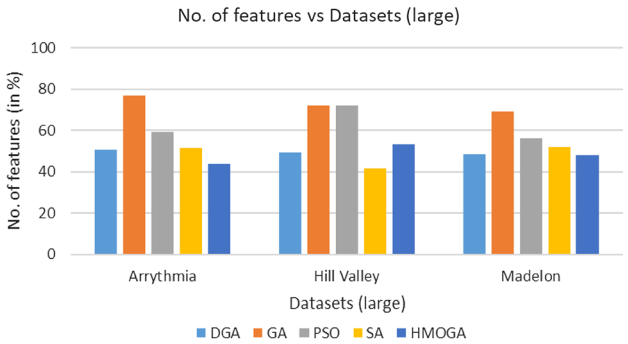


Fig. 7 Comparison of the percentage of features selected by DGA with that of GA, PSO and HMOGA over large category datasets

searching technique and use it for other methods which also suffer from the same exploitation problem as GA. The use of DGA for more application-based FS can also be explored.

Compliance with ethical standards

Conflict of interest The authors declare that they have no competing interests.

References

1. Malakar S, Ghosh M, Bhowmik S, Sarkar R, Nasipuri M (2019) A GA based hierarchical feature selection approach for handwritten word recognition. Neural Comput Appl 1–20
2. Ghosh M, Malakar S, Bhowmik S, Sarkar R, Nasipuri M (2019) Feature selection for handwritten word recognition using memetic algorithm. In: Advances in intelligent computing. Springer, pp 103–124
3. Culberson JC (1996) On the futility of blind search. Technical Report TR 96-18, University of Alberta, Department of Computing Science, Edmonton, Alberta, Canada,
4. Glover F (1989) Tabu search—part I. ORSA J Comput 1(3):190–206
5. Van Laarhoven AEH (1987) Simulated annealing. in simulated annealing: theory and applications. Springer, Dordrecht, pp 7–15
6. Problem-specific knowledge in heuristics. [Online]. http://antor.uantwerpen.be/problem-specific-knowledge-in-heuristics/. Accessed 07 01 2019
7. Kazakovtsev AL, Antamoshkin AN, Fedosov VV (2016) Greedy heuristic algorithm for solving series of eee components classification problem. In: IOP conference series: materials science and engineering, vol 122(1)
8. Dorigo M, Birattari M (2011) Ant colony optimization. In: Encyclopedia of machine learning, Springer, pp 36–39
9. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on micro machine and human science. MHS'95. pp 39–43
10. Yang J, Honavar V (1998) Feature subset selection using a genetic algorithm. IEEE Intell Syst 13:44–49
11. Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci (Ny) 179(13):2232–2248
12. Liu H, Motoda H (2007) Computational methods of feature selection, vol. 20071386. CRC Press, London
13. Mitra P, Murthy CA, Pal SK (2002) Unsupervised feature selection using feature similarity. IEEE Trans Pattern Anal Mach Intell 24(3):301–312
14. Belli S, López C, Romano J (2007) La excepcionalidad del otro. Athenea Digit. 11:104–113
15. Yang J, Honavar V (1998) Feature subset selection using a genetic algorithm. IEEE Intell Syst their Appl 13(2):44–49
16. Duval B, Hao J-K, Hernandez Hernandez JC (2009) A memetic algorithm for gene selection and molecular classification of cancer. In: Proceedings of the 11th annual conference on genetic and evolutionary computation—GECCO'09, p 201
17. Zhu Z, Ong YS, Dash M (2007) Markov blanket-embedded genetic algorithm for gene selection. Pattern Recognit 40(11):3236–3248
18. Ghosh M, Begum S, Sarkar R, Chakraborty D, Maulik U (2019) Recursive memetic algorithm for gene selection in microarray data. Expert Syst Appl 116:172–185
19. Ghosh M, Adhikary S, Ghosh KK, Sardar A, Begum S, Sarkar R (2019) Genetic algorithm based cancerous gene identification from microarray data using ensemble of filter methods. Med Biol Eng Comput 57(1):159–176
20. Ghosh M, Guha R, Mondal R, Singh PK, Sarkar R (2017) Feature selection using histogram based multi-objective GA for Handwritten Devanagari numeral recognition.
21. Dueck G (1993) New optimization heuristics. J Comput Phys 104(1):86–92
22. Leard R, Farmaceutiche T, Salern B (1996) 3 genetic algorithms in feature selection. pp. 67–86
23. Huang J (2007) A hybrid genetic algorithm for feature selection wrapper based on mutual information. vol 28, pp 1825–1844,

24. Oh I-S, Lee J-S, Moon B-R (2004) Hybrid genetic algorithms for feature selection. IEEE Trans Pattern Anal Mach Intell 26(11):1424–1437
25. Siedlecki JW, Sklansky (1993) A note on genetic algorithms for large-scale feature selection. vol 10, pp 88–107
26. Dueck TG, Scheuer (1990) Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 90(1):161–175
27. Baykasoglu A (2012) Design optimization with chaos embedded great deluge algorithm. Appl Soft Comput J 12(3):1055–1067
28. Landa-Silva D, Obit JH (2009) Evolutionary non-linear great deluge for university course timetabling. In: International conference on hybrid artificial intelligence systems, pp 269–276
29. Mccollum B, Mcmullan PJ, Parkes AJ, Burke EK, Abdullah S (2009) An extended great deluge approach to the examination timetabling problem. pp 10–12
30. Mafarja M, Abdullah S (2011) Modified great deluge for attribute reduction in rough set theory. In: Proceedings—2011 8th international conference on fuzzy systems and knowledge discovery, FSKD 2011, vol 3, pp 1464–1469
31. Badawi UA, Khalil M, Alsmadi S (2013) A hybrid memetic algorithm (genetic algorithm and great deluge local search) with backpropagation classifier for fish recognition. 10(2):348–356
32. Lipowski A, Lipowska D (2012) Roulette-wheel selection via stochastic acceptance. Phys A Stat Mech its Appl 391(6):2193–2196
33. De Jong KA, Spears WM (1992) A formal analysis of the role of multi-point crossover in genetic algorithms. Ann Math Artif Intell 5(1):1–26
34. UCI repository. [Online]. https://archive.ics.uci.edu/ml/datasets.html. Accessed 07 Jan 2019
35. Ablavsky V, Stevens MR (2003) Automatic feature selection with applications to script identification of degraded documents. null, p 750
36. Basu S, Das N, Sarkar R, Kundu M, Nasipuri M, Basu DK (2005) Handwritten 'Bangla' alphabet recognition using an MLP based classfier. In: 2nd National Conf. on computer processing of Bangla-2005, pp 285–291
37. Chaudhari S, Gulati M (2016) Script identification using Gabor feature and SVM classifier. Proc Comput Sci 79:85–92