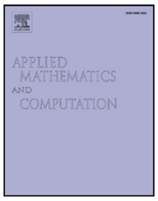
Applied Mathematics and Computation 270 (2015) 13–24

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc



Web page classification based on a simplified swarm optimization

Ji-Hyun Lee[a], Wei-Chang Yeh[b], Mei-Chi Chuang[a,b,*]

[a] Graduate School of Culture Technology, KAIST, Daejeon 305-701, Republic of Korea
[b] Integration and Collaboration Laboratory, Department of Industrial Engineering and Engineering Management, National Tsing Hua University, P.O. Box 24-60, Hsinchu 300, Taiwan, ROC

ARTICLE INFO

Keywords:
Web page classification
Simplified swarm optimization
Taguchi method

ABSTRACT

Owing to the incredible increase in the amount of information on the World Wide Web, there is a strong need for an efficient web page classification to retrieve useful information quickly. In this paper, we propose a novel simplified swarm optimization (SSO) to learn the best weights for every feature in the training dataset and adopt the best weights to classify the new web pages in the testing dataset. Moreover, the parameter settings play an important role in the update mechanism of the SSO so that we utilize a Taguchi method to determine the parameter settings. In order to demonstrate the effectiveness of the algorithm, we compare its performance with that of the well-known genetic algorithm (GA), Bayesian classifier, and K-nearest neighbor (KNN) classifiers according to four datasets. The experimental results indicate that the SSO yields better performance than the other three approaches.

1. Introduction

There has been an exponential increase of information available on the World Wide Web (WWW), and thus finding web pages that present information to satisfy requirements is very difficult. When we want to search for particular information and type 1 to 3 keywords in a search engine, the search engine will typically retrieve tens of thousands of web pages to us. Therefore, if we want to search using the search engine efficiently and quickly, we need an efficient classification method of web pages. Many studies have applied statistical classification or machine learning such as Bayesian classifiers [16], K-nearest neighbor classifiers [21], neural networks [17], and support vector machines [6] to solve the problem of classification. Nevertheless, the complexity of these approaches increases rapidly when there are large numbers of features and web pages. Therefore, we utilize the meta-heuristic approach to overcome the problem.

In web page classification problems, meta-heuristic approaches have been adopted in a number of studies. In recent years, many GA-based web page classification methods have been proposed [9–11,13,14]. In this study, we propose a novel simplified swarm optimization (SSO) method to solve the problem and compare the experimental results with those of GA. Simplified swarm optimization is a new meta-heuristic approach originally designed by Yeh [23], and it can overcome the drawbacks of particle swarm optimization (PSO) [2] in discrete problems and the drawbacks of GA in continuous problems. Simulation results reveal that the presented SSO has a better convergence rate and a higher quality solution than PSO alone.

* Corresponding author at: Integration and Collaboration Laboratory, Department of Industrial Engineering and Engineering Management, National Tsing Hua University, P.O. Box 24-60, Hsinchu 300, Taiwan, ROC. Tel.: +886 35742443.
E-mail address: meichi0525@gmail.com, d9734801@oz.nthu.edu.tw (M.-C. Chuang).

http://dx.doi.org/10.1016/j.amc.2015.07.120
0096-3003/© 2015 Published by Elsevier Inc.

In this paper, we propose a web page classification method by using the SSO to solve the problem, and the objective of our problem is to determine the role of a given web page. In terms of feature extraction, some researchers selected HTML tags [7,14,18] as features in earlier studies. However, Özel [10] pointed out that the use of HTML tags and terms in each tag will improve the accuracy of classification. We therefore consider HTML tags and terms at the same time on a web page as features and assign different weights to these features by using an SSO-based classifier. In experiments, we collect web pages from the Open Directory Project web site, and also compare our system with the well-known Bayesian classifier and K-nearest neighbor classifiers to demonstrate the efficiency and robustness of the SSO-based classifier.

The rest of this paper is organized as follows: In Section 2, we define the web page classification problem, and provide a general overview of PSO, the SSO and GA methods. Section 3 describes the methodology, our SSO-based classification system and GA-based classification system. The data used in the study, the detailed experimental results, and a discussion are presented in Section 4. Finally, Section 5 draws conclusions and notes some directions for future work.

2. Literature review

2.1. Genetic algorithm

Genetic algorithms (GA) have received considerable attention for their potential as an optimization technique for complex problems and have been successfully applied in the area of industrial engineering. Well-known applications include reliability design, scheduling and sequencing, facility layout and location, vehicle routing and scheduling, group technology, transportation, and many others.

The usual form of the genetic algorithm was described by Goldberg [3]. Genetic algorithms start with an initial set of random solutions called a population. Each individual in a population is called a chromosome, representing a solution to the problem at hand. Chromosomes evolve through successive iterations, called generations, and are evaluated by using some measures of fitness. According to the fitness value, fitter chromosomes have higher probabilities of being selected. To create the next generation, genetic algorithms use a crossover and mutation operator. A crossover operator merges two chromosomes from the current generation to create new chromosomes, called offspring. A mutation operator is a background operator that produces spontaneous random changes in various chromosomes.

2.2. Particle swarm optimization algorithm

Particle Swarm Optimization (PSO) is kind of evolutionary computation, advocated by Eberhart and Kennedy [2]. PSO was produced through simulating the behaviors of bird groups searching for food. Bird groups used the simplest and most effective method to search food. They searched the surrounding areas where those birds nearest to food were present. Therefore, in PSO, each bird searched in the areas meant optimum value of matter. The birds were called particles, and the position of each particle was expressed by x_{ij}, with i referring to the ith particle; j: dimension of searched space and v_{ij}: flying velocity of each particle in each dimension. Otherwise, each particle had a fitness value for optimum matter, and each particle all knew its optimum fitness value and optimum position at present called Particle best value; *pbest* at present. This information was just like the experiences owned by each particle. At the same time, each particle also knew its optimum value and optimum position in particle swarm at present called Globe best value; *gbest* at present. Such information was just like the experiences owned by other particles. The steps of PSO are described in the following.

Step 1: Initialization
Randomly generate initial solution and velocity of each particle.

Step 2: Evaluate the fitness value
Calculate the fitness value of each particle.

Step 3: Update Particle's local best fitness value and position
For each particle, compare its current fitness value with its best fitness value *pbest*. If its current fitness value is better than *pbest*, then replace *pbest* by the current fitness value.

Step 4: Update the global best fitness value and position of the population
For each particle, compare its best fitness value with *gbest*, the global optimal fitness value of all the particles in the population. If its best fitness value is better than *gbest*, then replace *gbest* by its best fitness value.

Step 5: Update the velocity and the position of each particle
At iteration t, the velocity and the position of particle i in the jth dimension is calculated as follows:

$$v_{ij}^{t} = wv_{ij}^{t-1} + c_1 r_1^{t-1}\left(p_{ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2^{t-1}\left(g_j^{t-1} - x_{ij}^{t-1}\right) \quad (1)$$

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t} \quad (2)$$

where w is the inertia weight, c_1 and c_2 are the cognition learning factors, r_1^{t-1} and r_2^{t-1} are two random uniform numbers between 0 and 1 at iteration $t-1$, v_{ij}^{t-1} is the jth dimensional velocity of particle i at iteration $t-1$, x_{ij}^{t-1} is the jth dimensional position of particle i at iteration $t-1$, $p_{ij}^{t-1}$is the jth dimensional best solution of particle i up to iteration $t-1$, $g_j^{t-1}$is the jth dimensional best solution of all the particles up to iteration $t-1$.

2.3. Simplified swarm optimization

In the past decade, there has been an increasing interest in soft computing motivated approaches, including Neural Networks, GAs, Tabu Searches, Simulated Annealing, Ant Colony Optimization, PSOs, and SSOs, for finding optimal or good quality solutions to larger problems ([22–25]). Among these methods, the simplified swarm optimization (SSO) is the most recently proposed approach. It was originally designed by Yeh [23], to overcome the drawbacks of PSO in discrete problems. Simulation results reveal that the SSO has a better convergence rate and a higher quality solution than PSO alone. The SSO has some appealing features, including easy implementation, having just a few parameters to tune, and a fast convergence rate. The SSO, a population-based stochastic optimization technique, belongs to the category of Swarm Intelligence methods, and is an evolutionary computational method inspired by particle swarm optimization (PSO). The major difference between PSO and the SSO lies in the update mechanism. Analogous to PSO, each solution moves towards the best previous solution and the best solution in the whole swarm in every iteration in the SSO. However, to maintain population diversity, enhance the capacity of escaping from the local optimum, and fully exploit local search capability, random movement is added to the SSO. Thus, in the SSO, each solution is a compromise of the current solution, the best solution in its history thus far, the best solution among all existing solutions thus far, and random movement. It is much simpler than other major soft computing (SC) techniques such as PSO, where it is necessary to calculate both the velocity and position functions, the GA, which requires genetic operations such as crossover and mutation and estimation of distribution algorithm (EDA), with which it is difficult and complicated to build an appropriate probability model [23].

3. Methodology

3.1. Problem definition

There has been an exponential increase of information available on the World Wide Web (WWW), and consequently finding web pages that present information to satisfy requirements is very difficult. If we want to search efficiently and quickly with a search engine, we need an efficient method of classifying web pages. In this paper, we discuss a binary classification approach to categorize web pages into one of two classes (e.g., “art” or “not art”). We adopt HTML tags and terms as classification features and utilize the SSO-based web page classifier to determine weights of each feature in the training dataset. Moreover, we propose a novel SSO to solve the problem of functional classification and overcome the drawbacks of PSO and GA. After the learning step of the SSO-based classifier, the weights are then used to classify new web pages. The architecture of the proposed system is shown in Fig. 1 and each step is explained below.

3.2. Collection of Web pages

We collect web pages from the Open Directory Project web site. The first dataset is collected from the area of Art and contains 1034 web pages. The second dataset is collected from the Computers area and contains 1021 web pages. The third dataset is collected from the Health area and contains 1031 web pages. The fourth dataset is collected from the Science area and contains 1032 web pages.

3.3. Feature extraction

We extract the terms from HTML tags and consider HTML tags and terms in each tag as classification features. In addition, 10 HTML tags, <TITLE>, <B>, <S>, <E>, <I>, <H1>, <H2>, <H3>, <P>, <A>, and <L>, are utilized for experiments. They denote title, bold, strong, emphatic, italic, header at level 1, header at level 2, header at level 3, paragraph, anchor, and list, respectively. The title tag represents a part of the title in a page. The bold, strong, emphatic, and italic tags are used to emphasize words; we therefore assign the terms to the same group. The header tags also represent a part of the header in a page, and we assign terms of three headers to the same group. The paragraph tag denotes a paragraph of text or a group of notions. The HTML code uses the anchor tag to create a hyperlink to other pages, and links similar contents generally. The HTML code can create an ordered or unordered list in a page.

3.4. Text processing

We take all the terms from each of the above mentioned tags as classifier features, and apply stopword removal and Porter’s stemming algorithm [12] to reduce the redundancy of the computation. Stop word processing entails removing non-informative words such as propositions, conjunctions, and certain high frequency words. Stemming is a technique to reduce words to their grammatical roots. For example, the meaning of the words “moves”, “moving”, and “moved” is similar to that of the word “move”, and therefore we only select the word “move” as a root in a web page.

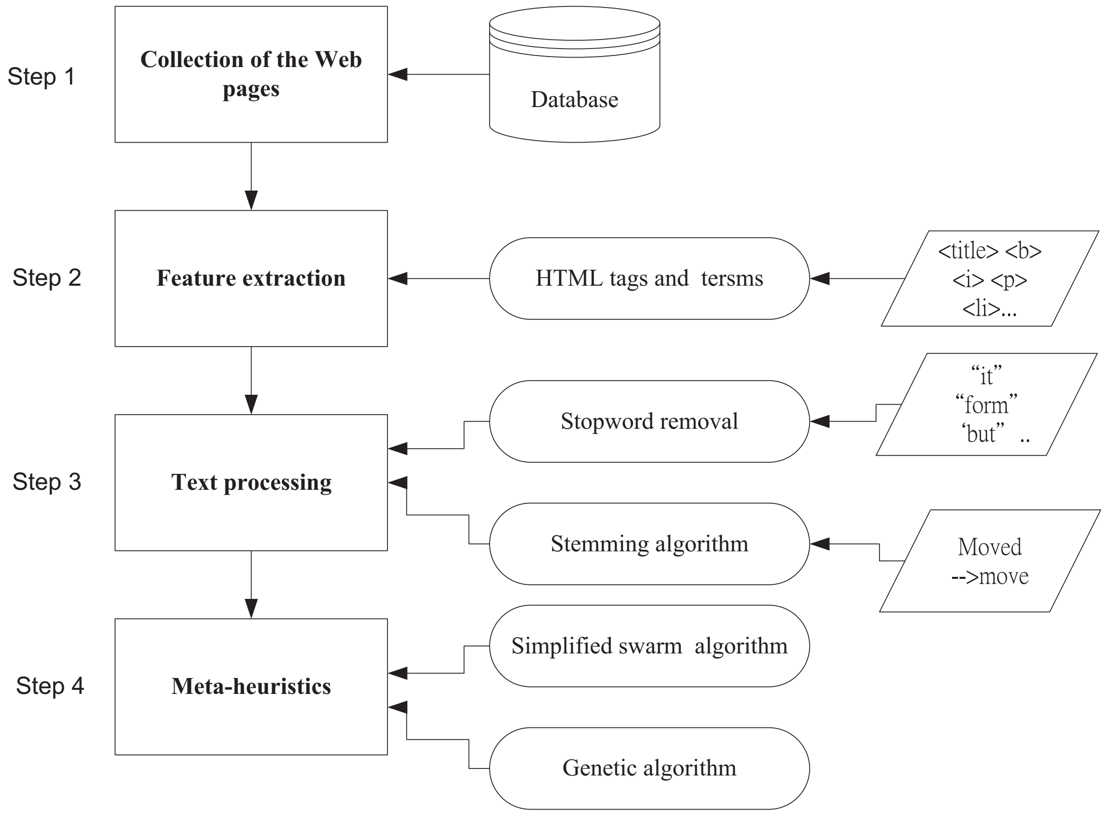


Fig. 1. Flowchart of the proposed methodology.

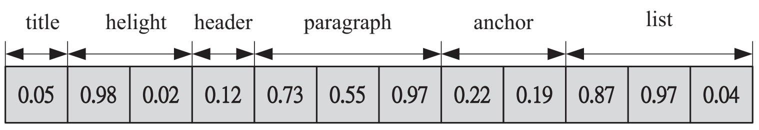


Fig. 2. The chromosome.

3.5. Meta-heuristics

We propose a novel SSO classifier to solve web page classification problems. In order to demonstrate the efficiency of our algorithm, we compare results obtained from the SSO with those yielded by GA. The procedures of GA and the SSO implemented to address the problem are as follows.

3.5.1. Genetic algorithm

3.5.1.1. Initialization. Initial solutions are randomly generated. In this problem, we have m tags and each tag has n_i terms. Each gene represents a weight, and we denote w_{ij} as the weight of tag i and term j, and each weight is a real number between 0 and 1. For example, a chromosome of a problem with 6 groups of tags and 12 terms is shown in Fig. 2. The group of highlights contains <B>, <S>, <E>, and <I> tags; the group of headers contains <H1>, <H2>, and <H3> tags.

3.5.1.2. Calculate the fitness function value. We adopt the F_1 value [19,20] to determine the fitness of each chromosome, where the F_1 value is defined as follows:

$$\text{Fitness function} = F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

where the *Precision* and *Recall* are computed according to the contingency matrix [4,14] shown in Table 1. According to Table 1, *TP* (*TN*) is a positive (negative) document that is labeled correctly by the classifier; *FN* (*FP*) denotes a positive (negative) document that is labeled incorrectly [4]. These terms are used to compute *Precision* and *Recall* as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Table 1
Contingency matrix for a 2-class classification system.

		Actual class	
		Positive class	Negative class
Predicted class	Classed positive	*TP*	*FP*
	Classed negative	*FN*	*TN*

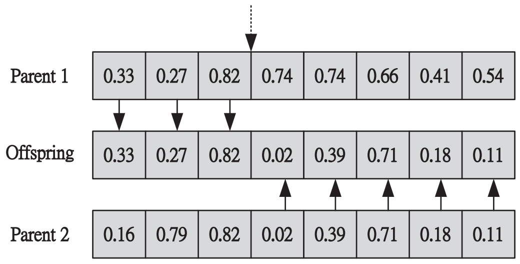


Fig. 3. Crossover operator.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

We adopt the cosine similarity proposed by Salton, Wong, and Yang [15] to determine the class of a web page. If the similarity is larger than the predefined threshold, the web page will be classified as a positive class; otherwise, it will be classified as a negative class. We assume that the predefined threshold is 0.09 in all experiments. We utilize a chromosome as a classifier to classify positive and negative web pages in the training and testing dataset. Let $\vec{d} = (d_{11}, d_{12}, \ldots, d_{1n_1}, d_{21}, \ldots, d_{mn_m})$ denote a web document vector. Let $\vec{w} = (w_{11}, w_{12,} \ldots, w_{1n_1}, w_{21}, \ldots w_{mn_m})$ denote a weight vector. The document vectors can be computed using the frequencies of occurrence of the terms that belong to tag *i* within documents; however, the frequencies of occurrence of the terms might vary widely in each document, and therefore we normalize the document vectors by dividing the maximum value in that document vector [1]. The cosine similarity is calculated according to Eq. (6).

$$sim(w, d) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} w_{ij} \times d_{ij}}{\sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n_i} w_{ij}^2} \times \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n_i} d_{ij}^2}} \tag{6}$$

3.5.1.3. Selection. In most practical applications, a roulette wheel approach is adopted as the selection procedure. The underlying idea is to determine the selection probability for each chromosome proportional to the fitness value. For each chromosome *k*with fitnessf_k, its selection probabilityp_k is calculated as follows:

$$p_k = f_k / \sum_{j=1}^{N} f_j \tag{7}$$

We then make a wheel according to this probability. The selection process begins by spinning the roulette wheel *N*times. Each time, a chromosome is selected for a new population in this way. In GA, the fitness function is usually the reciprocal of the objective value to reflect the relative superiority or inferiority if there are no further constraints.

3.5.1.4. Crossover. Crossover is an operation to generate new offspring from two parents. It is the main operator in the genetic algorithm. In this study, we use the one-point crossover operator, which means all genes are changed after a single crossover point. As shown in Fig. 3, the resulting offspring would be (0.33, 0.27, 0.82, 0.02, 0.39, 0.71, 0.18, and 0.11) if the first three genes are selected from parent 1, and the other genes are from parent 2.

3.5.1.5. Mutation. Mutation is another main operator of GA. It is used to prevent premature and falling into local optima. This operation can be viewed as a transition from the current solution to a neighborhood solution in a local search algorithm. As shown in Fig. 4, the third gene is randomly selected and altered to 0.07 according to its range.

3.5.2. Simplified swarm optimization

3.5.2.1. Randomly generate initial solutions. A particle is created at random, and consists of a list of feature weights, which are real numbers between 0 and 1. The representation of a particle is the same as that in Section 3.5.1.1.

3.5.2.2. Evaluation function. The fitness value of each particle is calculated according to Eq. (3).

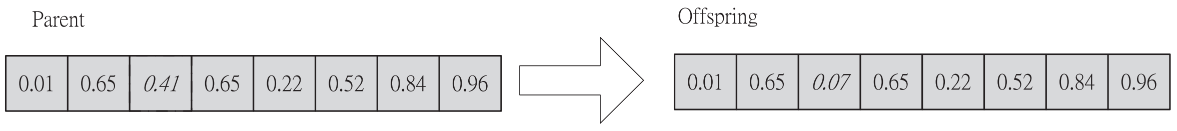


Fig. 4. Mutation procedure.

Table 2
Number of documents for training and testing set in four datasets.

Dataset	Train	Test	Total
Art	821	213	1034
Computers	813	208	1021
Health	832	199	1031
Science	818	214	1032

3.5.2.3. Compare the performance of each particle to its best performance so far. If particle *k* finds a solution that is better than the last iteration, it then replaces the old position with the new position; in addition, particle *k* updates its objective function value.

3.5.2.4. Compare the performance of each particle to the global best performance so far. If particle *k* finds a solution that is better than the *gbest* of the last iteration, *gbest* is replaced with the new particle.

3.5.2.5. Update position of particles. The underlying principle of a traditional PSO is that the next position of each particle is a compromise of its current position, the best position in its history so far, and the best position among all the existing particles. However, we eliminate the role of *pbest* in the SSO. The reason is that it needs to compare *pbest* with the current solution if the current solution improves (see Step 3 in Section 2.2). It also takes another $O(n)$ to replace *pbest* by the current solution if it is better than *pbest*. Now, we propose Eq. (8) to update the positions of particles.

$$x_{kd}^{t} = \begin{cases} x_{kd}^{t-1} \text{ if } \xi \in [0, C_w) \\ g_k \text{ if } \xi \in [C_w, C_g) \\ x \text{ if } \xi \in [C_g, 1) \end{cases} \tag{8}$$

where $x_{kd}^{t}$is the weigh in the *d*th dimension of *k*th particle at iteration t, ξ is a uniform random number in the range [0, 1), and x is a random number between 0 and 1. Thus, in the SSO, each solution is a compromise of the current solution, the best solution among all existing solutions and a random movement that enhances that capacity of escaping from the local optimum. Notice that $C_w = c_w$, and $C_g = C_w + c_g$, c_w and c_g represent the possibilities of the new variable value generated from the current solution, *gbest* and a random number in the SSO, respectively. If $0 \le \xi < C_w$ is true, then the original individual will be kept. If $C_w \le \xi < C_g$ is true, then the original individual will be replaced by *gbest*. If $C_g \le \xi < 1$ is true, then the original individual will be replaced by new individuals generated randomly.

4. Experimental results

In order to evaluate the performance of the proposed algorithms, a computational experiment is conducted and the results are presented in this section. We also compare performance of our SSO with that of GA, Bayesian and KNN, respectively.

4.1. Experiment environment

We run all data on an Intel Core 2 duo 2.93 GHz computer with 2GB RAM. All the algorithms are coded in C on a Windows XP operating system.

4.2. Data set

We use web pages from the Open Directory Project web site (http://www.dmoz.org/) to train the weights and test system performance. Table 2 shows the training and testing set. We randomly selected some data from each dataset as the training set, and left the remainder for the testing set. The ratio of training set to test set was 8:2 approximately. Table 3 lists the number of features in each group from four datasets.

Table 3
Number of features of each tag for each dataset.

Group	Tag	Dataset			
		Art	Computers	Health	Science
Title	<TITLE>	1323	1239	1233	1464
Highlight	<B> <S> <I><E>	6938	4137	5191	6910
Header	<H1> <H2><H3>	4316	4257	3781	3855
Paragraph	<P>	16,278	14,073	15,777	243,95
Anchor	<A>	13,649	11,405	13,074	15,158
List	<L>	10,660	11,395	11,039	15,579
Total		53,164	46,506	50,095	67,361

Table 4
Factors and levels of the Taguchi method for GA and SSO.

Algorithm	Factors	Levels			
		Level 1	Level 2	Level 3	Level 4
GA	Crossover rate (A)	0.60	0.70	0.80	0.90
	Mutation rate (B)	0.01	0.02	0.03	0.04
SSO	C_w (C)	0.15	0.25	0.35	0.45
	C_g (D)	0.55	0.65	0.75	0.85

Table 5
L_{16} orthogonal array and results for GA and SSO.

No.	GA			SSO		
	Control factors			Control factors		
	A	B	S/N ratio	C	D	S/N ratio
1	1	1	−3.47	1	1	−3.11
2	1	2	−3.40	1	2	−3.15
3	1	3	−3.40	1	3	−2.98
4	1	4	−3.52	1	4	−2.78
5	2	1	−3.50	2	1	−3.13
6	2	2	−3.50	2	2	−3.19
7	2	3	−3.51	2	3	−3.02
8	2	4	−3.41	2	4	−2.83
9	3	1	−3.47	3	1	−3.31
10	3	2	−3.38	3	2	−3.2
11	3	3	−3.48	3	3	−3.07
12	3	4	−3.41	3	4	−2.82
13	4	1	−3.48	4	1	−3.43
14	4	2	−3.50	4	2	−3.33
15	4	3	−3.38	4	3	−3.17
16	4	4	−3.51	4	4	−2.86

4.3. Genetic algorithm and simplified swarm optimization parameters

Because the parameter settings for algorithms affect the output quality, we utilize a Taguchi method proposed by Genichi Taguchi to determine suitable levels of the design factors [8]. A Taguchi method adopts orthogonal arrays (OAs) to analyze all factors of experiments with minimal cost and time, and finds the optimal parameter combination by using the signal-to-noise (S/N) ratio.

In all computational experiments, the iteration number is set at 300, and the particle or chromosome number is set at 80, and we use a Taguchi method to determine the crossover rate and mutation rate for GA and C_w and C_g for the SSO. Table 4 shows the parameters and factor levels for the SSO and GA. A L_{16} orthogonal array is used to conduct the experiment described in this paper, and can lay out 16 trials, 2 factors in a column, and 4 levels. The layout of the L_{16} orthogonal array is shown in Table 5. Table 5 also shows the signal-to-noise (S/N) ratio in each experiment for GA and the SSO. Based on a main effect plot, shown in Fig. 5, the optimal level of the crossover rate and mutation rate is set at A3B3. From Fig. 6, the optimal level of C_w and C_g is set at C1D4. Different parameter settings of GA and the SSO are summarized in Table 6.

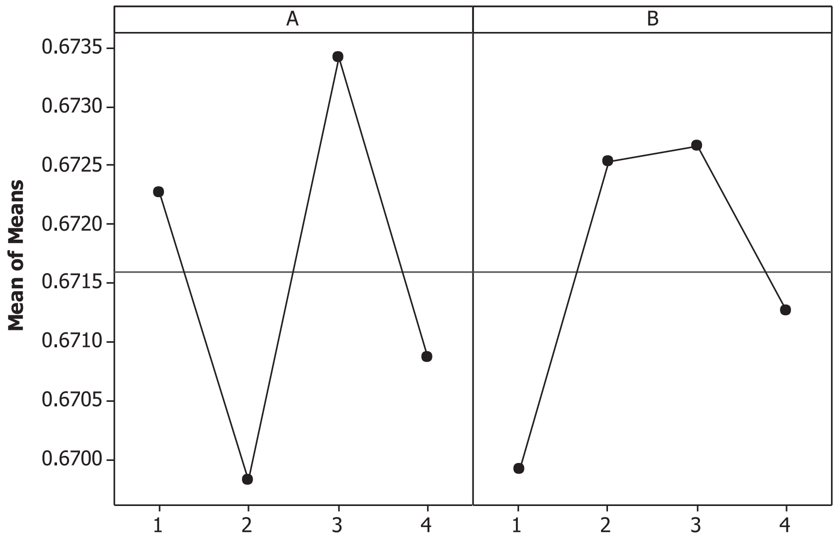


Fig. 5. Main effect plot of response for GA.

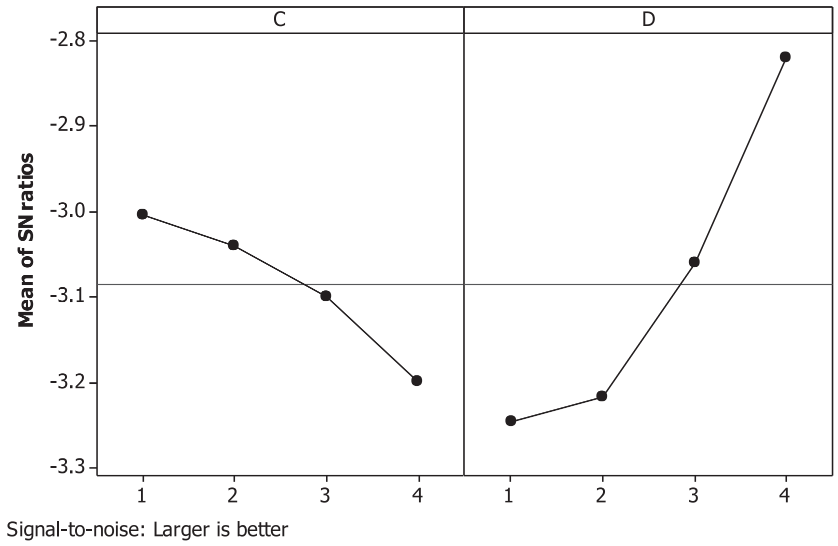


Fig. 6. Main effect plot of response for SSO.

Table 6
Parameter settings of GA and PSO.

GA Parameter	Value	SSO Parameter	Value
Iteration number	300	Iteration number	300
Population size	80	Population size	80
Crossover rate	0.8	C_w	0.15
Mutation rate	0.03	C_g	0.85

4.4. Simulation results and discussion

We adopt the SSO-based web page classification approach to obtain the weights of each feature and test other web pages according to optimal weights and different document frequency (DF) cut-offs. Document frequency of a term is the number of documents in which it occurs. The experiments are repeated according to different document frequencies for each dataset and the *F*1 value is used to evaluate the classification performance. We also compare our results with those of a standard Bayesian

Table 7
*F*1 values and CPU time of four approaches for the art dataset.

DF	GA		SSO		Bayesian		KNN	
	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)
20	69.44	489.95	70.79	500.74	**76.25**	0.00	**72.00**	4.59
30	71.71	244.93	73.79	250.27	73.23	0.00	71.25	3.30
40	71.84	134.87	73.75	135.53	71.65	0.00	69.21	3.20
50	**73.14**	94.01	74.07	94.66	72.10	0.00	69.65	3.30
60	72.61	65.56	76.01	65.48	71.75	0.00	68.83	3.47
70	71.33	42.06	79.31	42.80	70.97	0.00	67.33	3.77
85	70.99	26.34	70.75	26.56	71.10	0.00	68.65	3.23
95	72.92	26.81	**80.18**	27.51	71.95	0.00	66.44	3.97

Table 8
*F*1 values and CPU time of four approaches for the computers dataset.

DF	GA		SSO		Bayesian		KNN	
	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)
20	69.44	489.95	70.79	500.74	**76.25**	0.00	**72.00**	4.59
30	71.71	244.93	73.79	250.27	73.23	0.00	71.25	3.30
40	71.84	134.87	73.75	135.53	71.65	0.00	69.21	3.20
50	**73.14**	94.01	74.07	94.66	72.10	0.00	69.65	3.30
60	72.61	65.56	76.01	65.48	71.75	0.00	68.83	3.47
70	71.33	42.06	79.31	42.80	70.97	0.00	67.33	3.77
85	70.99	26.34	70.75	26.56	71.10	0.00	68.65	3.23
95	72.92	26.81	**80.18**	27.51	71.95	0.00	66.44	3.97

Table 9
*F*1 values and CPU time of four approaches for the health dataset.

DF	GA		SSO		Bayesian		KNN	
	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)
20	76.98	806.38	77.42	806.31	76.10	0.00	67.34	5.58
35	77.08	610.76	78.05	602.92	75.95	0.00	67.65	4.80
50	76.29	366.17	76.98	372.04	76.43	0.00	64.81	3.53
65	78.93	237.33	80.00	237.12	76.68	0.00	64.79	3.09
80	78.26	160.52	80.00	160.79	76.62	0.00	63.08	2.86
100	78.79	99.20	80.70	99.40	77.12	0.00	72.54	2.88
120	79.59	71.56	83.82	71.86	78.15	0.00	73.97	3.19
140	**80.69**	51.37	**88.03**	52.02	**78.67**	0.00	**74.05**	3.33

classifier [5] and KNN [21]. Tables 7–10 show *F*1 values and CPU time of four approaches in every dataset, and the maximum *F*1 value of every approach of every dataset is marked in boldface in the table.

Table 7 shows *F*1 values of four approaches for the art dataset. We test the art dataset according to 8 different document frequencies, varied from 20 to 95. The highest *F*1 value is 73.14% when using GA, obtained when the document frequency is 50; the lowest performance is 69.44%, obtained when the document frequency is 20. The highest *F*1 value is 76.25% when using Bayesian, obtained when the document frequency is 20; the lowest performance is 70.97%, obtained when the document frequency is 70. The highest *F*1 value is 72.00% when using KNN, obtained when the document frequency is 20; the lowest performance is 66.44%, when the document frequency is 95. However, when we use the SSO approach, the *F*1 value is better than the other four approaches and the highest *F*1 value is 80.18%, when the document frequency is 95.

Table 8 shows *F*1 values and CPU time of four approaches for the computers dataset. We set the document frequency between 20 and 130 for this dataset. KNN is the worst approach among the four approaches, yielding an overall *F*1 value below 70%. When the document frequency is 20 and 35, Bayesian outperforms the other three approaches with values of 79.05% and 78.38%, respectively. GA and the SSO perform best when the document frequency is 130. The highest *F*1 value is 77.74% and the lowest is 72.22% in the case of using GA. With the SSO, the highest *F*1 value is 83.00%, when the document frequency is 130, and the lowest is 73.72%, when the document frequency is 20.

Table 9 shows *F*1 values and CPU time of four approaches for the health dataset and the document frequency is set between 20 and 140. All four approaches perform best when the document frequency is 140, and the overall *F*1 value is above 75% for different document frequencies except for in the case of KNN. KNN is still the worst approach, and the highest *F*1 value is only 74.05%. The highest and lowest *F*1 values using the GA are 80.69% and 76.29%, respectively, and the performance is almost better than that of Bayesian. With the Bayesian approach, the highest *F*1 value is 78.67% and the lowest is 75.95%. However, when we

Table 10
*F*1 values and CPU time of four approaches for science dataset.

DF	GA		SSO		Bayesian		KNN	
	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)	*F*1 (%)	CPU time (s)
30	75.17	497.34	76.15	497.69	74.09	0.00	**70.19**	4.48
40	75.48	384.34	76.57	384.38	74.09	0.00	69.38	3.92
50	74.68	288.40	77.08	287.82	74.77	0.00	70.03	3.63
60	76.53	172.53	78.15	235.50	75.00	0.00	69.43	3.56
70	77.02	152.55	78.55	152.54	74.85	0.00	68.81	3.27
80	76.22	111.19	79.73	111.23	75.08	0.00	69.03	3.20
90	**78.03**	85.73	80.69	86.01	74.92	0.00	69.26	3.20
100	77.17	60.70	**84.29**	61.13	**75.39**	0.00	70.13	3.20

Table 11
Optimal classification results of GA, SSO, Bayesian, and KNN.

	GA			SSO			Bayesian			KNN		
	Precision (%)	Recall (%)	*F*1 (%)	Precision (%)	Recall (%)	*F*1 (%)	Precision (%)	Recall (%)	*F*1 (%)	Precision (%)	Recall (%)	*F*1 (%)
Art	61.75	89.68	73.14	**92.71**	70.63	**80.18**	62.89	**96.83**	76.25	58.79	92.86	72.00
Computers	67.90	90.91	77.74	**79.55**	85.78	**83.00**	66.86	**96.69**	79.05	57.53	88.43	69.71
Health	70.71	93.60	80.69	**85.08**	91.20	**88.03**	67.43	**94.40**	78.67	65.24	85.60	74.05
Science	68.00	91.54	78.03	**78.67**	90.77	**84.29**	63.35	**93.08**	75.39	58.85	86.92	70.19
Average	65.23	90.44	75.76	**81.81**	83.46	**82.21**	65.13	**95.25**	77.34	60.10	88.45	71.49

Table 12
Friedman test rankings.

	Average rankings				*p*-Value
	GA	SSO	Bayesian	KNN	
Precision	2.75	4.00	2.25	1.00	0.011[a]
Recall	2.75	1.50	4.00	1.75	0.026[a]
F1	2.50	4.00	2.25	1.00	0.013[a]

[a] :alpha=0.05.

use the SSO, the *F*1 value is above 80% over half, and the highest *F*1 value is 88.03%, reflecting the best performance among the four approaches.

Table 10 shows *F*1 values and CPU time of four approaches for the science dataset and we set the document frequency between 30 and 100.The performance using KNN is poor and the *F*1 value is below 71% overall. The performance of GA and the SSO is better than that of Bayesian approach, with the highest and lowest *F*1 values being 74.09% and 75.39% with application of Bayesian. With GA, the highest *F*1 value is 78.03% and the lowest *F*1 value is 74.68%. However, the SSO gives an overall value of more than 76% according to different document frequencies and the highest *F*1 value is 84.29%, reflecting the best performance among the four approaches.

Tables 7–10 also show the CPU time of the four approaches, which includes training and testing times. GA and SSO require the greatest amount of time for training data; however, KNN uses the largest amount of time for testing. Bayesian spent the least time for training and testing data among the four approaches. Even though the execution time of the SSO is the longest among the four algorithms, the SSO provides excellent performance.

Table 11 shows the optimal results of precision, recall, and *F*1 measurement for GA and the SSO, Bayesian, and KNN. Because each algorithm was executed 10 times over each data set Table 11 also shows average results for all algorithms. Though Bayesian shows better results on recall, the SSO reports better results on both precision and *F*1 value than other three algorithms.

The Friedman test results and 95% confidence interval of Turkey's multiple comparisons are shown in Tables 12 and 13, respectively.

Average rankings of *precision* using the Friedman test show that there is a difference among four algorithms, with a Friedman statistic equal to 11.1 and *p*-Value equal to 0.011. Moreover, the Turkey's multiple comparisons show us that the difference is due to the difference in precision between GA and the SSO and with the difference between the SSO and Bayesian and also with the difference between the SSO and KNN.

Average rankings of *recall* using the Friedman test show that there is a difference among four algorithms, with a Friedman statistic equal to 9.3 and *p*-Value equal to 0.026. However, there is no significant difference in recall among four algorithms.

Average rankings of *F*1 using the Friedman test show that there is a difference among four algorithms, with a Friedman statistic equal to 10.8 and *p*-Value equal to 0.013. In addition, the post-hoc analysis shows us that the difference is due to the difference in *F*1 between GA and the SSO and with the difference between GA and KNN and with the difference between the SSO and Bayesian and with the difference between the SSO and KNN and also with the difference between Bayesian and KNN.

Table 13
Turkey's multiple comparisons.

	Algorithm 1	Algorithm 2	*p*-Value	95% Confidence interval	
				Lower bound	Upper bound
Precision					
	GA	SSO	0.001*	−25.9241	−7.9000
	GA	Bayesian	0.915	−7.0541	10.9691
	GA	KNN	0.152	−2.0241	15.9991
	SSO	Bayesian	0.000*	9.8584	27.8816
	SSO	KNN	0.000*	14.8884	32.9116
	Bayesian	KNN	0.386	−3.9816	14.0416
Recall					
	GA	SSO	0.297	−4.1071	17.7821
	GA	Bayesian	0.733	−14.7621	7.1271
	GA	KNN	0.849	−7.9646	13.9246
	SSO	Bayesian	0.057	−21.5996	0.2896
	SSO	KNN	0.727	−14.8021	7.0871
	Bayesian	KNN	0.301	−4.1471	17.7421
*F*1					
	GA	SSO	0.020*	−11.9860	−0.9640
	GA	Bayesian	1.000	−5.4510	5.5710
	GA	KNN	0.034*	0.4015	11.4235
	SSO	Bayesian	0.019*	1.0240	12.0460
	SSO	KNN	0.000*	6.8765	17.8985
	Bayesian	KNN	0.036*	0.3415	11.3635

* p-Value < 0.05.

Table 14
The best features and weights for the computers data set.

Tag	Term	DF	Weight	Tag	Term	DF	Weight
Title	comput	305	0.9253	paragraph	softwar	161	0.0026
Header	comput	173	0.2703		research	147	0.9847
Anchor	comput	292	0.9502		inform	183	0.2794
	system	143	0.0894		program	135	0.1116
	scienc	132	0.2794		scienc	131	0.4404
	softwar	144	0.2571		work	156	0.0384
	home	270	0.0281		network	136	0.4993
	contact	249	0.0266		service	154	0.2008
	service	140	0.2245		technolog	142	0.9274
	new	143	0.0137		site	161	0.0295
	site	133	0.0052		page	147	0.1303
	privacy	133	0.0476		support	152	0.6298
Highlight	comput	149	0.7225		include	140	0.0609
List	comput	230	0.9724		design	135	0.0245
	home	187	0.0202		home	157	0.1410
	contact	184	0.0628		contact	135	0.0285
Paragraph	comput	373	0.3685				
	system	212	0.0214				

The reasons that the SSO outperforms other three algorithms are summarized in the following. The SSO has a good random movement according to simulation results, so it can escape local optimum easily. In addition, the parameter settings are an important part in the algorithm, so we adopted the Taguchi method to find the parameter settings. Thence, the SSO has a higher quality solution than other algorithms.

Table 14 presented the best features and weighs learned by the SSO for the computers data set. The best classifier for the computers data set has 34 features which contain 18 terms from <paragraph> tag, 10 terms from <anchor> tag, 3 terms from <list> tag, 1 term from <title> tag, 1 term from <header> tag and 1 term from <bold> tag. Even though different tags have the same terms, their weights obtained from algorithms are different from each other. The weight of the term "comput" is 0.9253 in <title> tag but it is 0.2703 or 0.9502 in <header> and <anchor> tags, respectively. Therefore, we should consider both tags and terms in our features simultaneously.

5. Conclusions

In this paper, we considered a binary problem and proposed a novel web page classification technique using the SSO. We utilized the document frequency to select features and assigned weights to the features. We adopted the SSO to determine the weights of each feature and tested new web pages according to these weights. Since the parameter setting is an important factor that affects the results of algorithms, we used the Taguchi method to determine this. In order to demonstrate the efficiency of the SSO, we compared it with the well-known GA, Bayesian, and KNN methods. Four datasets that contain art, computers, health, and science are used in this experiment. The experimental results showed that the SSO outperforms the other three approaches and the SSO provided acceptable performance with four datasets. In future research, we may use other feature selection approaches such as principal component analysis (PCA) or latent semantic indexing (LSI) and select the most commonly occurring words in each dataset. Determining the threshold is also important as a good threshold will significantly improve the performance.

Acknowledgments

This research was supported by a Grant from the National Science Council (No. 101-2917-I-007-001).

References

[1] R. Baeza-Yates, B. Ribeiro, Modern Information Retrieval, Addison-Wesley, New York, 1999.
[2] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the 6th IEEE Symposium on MicroMachine and Human Science, Los Alamitos, IEEE, 1995, pp. 39–43.
[3] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[4] J. Han, M. Kanber, Data Mining: Concepts and Techniques, second ed., Morgan Kaufman, San Francisco, 2006.
[5] T. Joachims, Probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, in: Proceedings of International Conference on Machine Learning, Nashville, TN, USA, 1997, pp. 143–151.
[6] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the 10th European Conference on Machine Learning, Springer, Berlin, 1998, pp. 137–142.
[7] S. Kim, B.T. Zhang, Genetic mining of HTML structures for effective Web document retrieval, Artif. Intell. 18 (2003) 243–256.
[8] R.J. Kuo, Y.J. Syu, Z.Y. Chen, F.C. Tien, Integration of particle swarm optimization and genetic algorithm for dynamic clustering, Inf. Sci. 195 (2012) 124–140.
[9] H. Liu, S. Huang, A genetic semi-supervised fuzzy clustering approach to text classification, Lecture Notes in Computer Science 2762 (2003) 173–180.
[10] S.A. Özel, A web page classification system based on a genetic algorithm using tagged-terms as features, Expert Syst. Appl. 38 (2011) 3407–3415.
[11] A. Pietramala, V.L. Policicchio, P. Rullo, I. Sidhu, A genetic algorithm for text classification rule induction, Lecture Notes in Artificial Intelligence 5212 (2008) 188–203.
[12] M.F. Porter, An algorithm for suffix stripping, Program 14 (1980) 130–137.
[13] D. Qi, B. Sun, A genetic k-means approach for automated Web page classification, in: Proceedings of IEEE International Conference on Information Reuse and Integration, IRI-2004, Las Vegas, NV, 2004, pp. 241–246.
[14] A. Ribeiro, V. Fresno, M.C. Garcia-Alegre, D. Guinea, Web page classification: a soft computing approach, Lecture Notes in Computer Science 2663 (2003) 103–112.
[15] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, Commun. ACM 18 (1975) 613–662.
[16] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (2002) 1–47.
[17] A. Selamat, S. Omatu, Web page feature selection and classification using neural networks, Inf. Sci. 158 (2004) 69–88.
[18] Trotman, Choosing document structure weights, Inf. Process. Manag. 41 (2005) 243–264.
[19] Z. Wang, Q. Zhang, D. Zhang, A PSO-based web document classification algorithm, in: Proceedings of IEEE eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Network, and Parallel/Distributed Computing, Qingdao, China, 2007, pp. 659–664.
[20] Y. Yang, An evaluation of statistical approach to text categorization, Inf. Retr. 1 (1999) 69–90.
[21] Y. Yang, S. Slattery, R. Ghani, A study of approaches to hypertext categorization, J. Intell. Inf. Syst. 18 (2002) 219–241.
[22] W.C. Yeh, W.W. Chang, Y.Y. Chung, A new hybrid approach for mining breast cancer pattern using discrete particle swarm optimization and statistical method, Expert Syst. Appl. 36 (2009) 8204–8211.
[23] W.C. Yeh, A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems, Expert Syst. Appl. 36 (2009) 9192–9200.
[24] W.C. Yeh, Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization, IEEE Trans. Syst. Man Cybern. Syst. 42 (2012) 250–261.
[25] W.C. Yeh, Simplified swarm optimization in disassembly sequencing problems with learning effects, Comput. Oper. Res. 39 (2012) 2168–2177.