Feature Selection based on Rough Sets and Particle Swarm Optimization

Xiangyang Wang [a,*], Jie Yang [a], Xiaolong Teng [a], Weijun Xia [b], Richard Jensen [c]

[a] Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200030, China
[b] Institute of Automation, Shanghai Jiao Tong University, Shanghai 200030, China
[c] Department of Computer Science, The University of Wales, Aberystwyth

Abstract: We propose a new feature selection strategy based on rough sets and Particle Swarm Optimization (PSO). Rough sets has been used as a feature selection method with much success, but current hill-climbing rough set approaches to feature selection are inadequate at finding optimal reductions as no perfect heuristic can guarantee optimality. On the other hand, complete searches are not feasible for even medium-sized datasets. So, stochastic approaches provide a promising feature selection mechanism. Like Genetic Algorithms, PSO is a new evolutionary computation technique, in which each potential solution is seen as a particle with a certain velocity flying through the problem space. The Particle Swarms find optimal regions of the complex search space through the interaction of individuals in the population. PSO is attractive for feature selection in that particle swarms will discover best feature combinations as they fly within the subset space. Compared with GAs, PSO does not need complex operators such as crossover and mutation, it requires only primitive and simple mathematical operators, and is computationally inexpensive in terms of both memory and runtime. Experimentation is carried out, using UCI data, which compares the proposed algorithm with a GA-based approach and other deterministic rough set reduction algorithms. The results show that PSO is efficient for rough set-based feature selection.

Keywords: Feature selection; Rough Sets; Reduct; Genetic Algorithms; Particle Swarm Optimization

*Tel.: +86 21 62933739
E-mail: wangxiangyang@sjtu.edu.cn (Xiangyang Wang)

1. Introduction

In many fields such as data mining, machine learning, pattern recognition and signal processing, datasets containing huge numbers of features are often involved. In such cases, feature selection will be necessary (Liu and Motoda, 1998; Guyon and Elisseeff, 2003). Feature selection is the process of choosing a subset of features from the original set of features forming patterns in a given dataset. The subset should be necessary and sufficient to describe target concepts, retaining a suitably high accuracy in representing the original features. The importance of feature selection is to reduce the problem size and resulting search space for learning algorithms. In the design of pattern classifiers it can improve the quality and speed of classification. (Kudo and Sklansky, 2000).

Due to the abundance of noisy, irrelevant or misleading features, the ability to handle imprecise and inconsistent information in real world problems has become one of the most important requirements for feature selection. Rough sets (Pawlak, 1982, 1991, 1997) can handle uncertainty and vagueness, discovering patterns in inconsistent data. Rough sets have been a useful feature selection method in pattern recognition (Chouchoulas and Shen, 2001). The rough set approach to feature selection is to select a subset of features (or attributes), which can predict the decision concepts as well as the original feature set. The optimal criterion for rough set feature selection is to find shortest or minimal reducts while obtaining high quality classifiers based on the selected features (Swiniarski and Skowron, 2003).

There are many rough set algorithms for feature selection. The most basic solution to finding minimal reducts is to generate all possible reducts and choose any with minimal cardinality, which can be done by constructing a kind of discernibility function from the

dataset and simplifying it (Bazan et al, 2000; Komorowski et al, 1999). Starzyk uses strong equivalence to simplify discernibility functions (Starzyk et al., 1998; Janusz et al., 2000). Obviously, this is an expensive solution to the problem and is only practical for very simple datasets. It has been shown that finding minimal reducts or all reducts are both NP-hard problems (Skowron, 1992). Therefore, heuristic approaches have to be considered.

In general, there are two kinds of rough set methods for feature selection, hill-climbing (or greedy) methods and stochastic methods (Vafaie and Imam, 1994). The hill-climbing approaches usually employ rough set attribute significance as heuristic knowledge. They start off with an empty set or attribute core and then adopt forward selection or backward elimination. Forward selection adds in turn, one at a time, the most significant attribute from the candidate set, until the selected set is a reduct. Backward elimination is the reverse, starting with the full attribute set and removing attributes incrementally. X. Hu gives a reduction algorithm using the positive region-based attribute significance as the guiding heuristic (X. Hu 1995a, 1995b). Wang develops a conditional information entropy-based reduction algorithm, using conditional entropy-based attribute significance (Wang, 2004, 2002). K. Hu computes the significance of an attribute making use of heuristic ideas from discernibility matrices and proposes a heuristic reduction algorithm (K. Hu et al., 2003). The positive region and conditional entropy-based methods choose a minimal feature subset that fully describes all concepts in a given dataset. The discernibility matrix-based method is to select a feature subset with high discriminatory power, which guarantees the maximal between-class separability for the reduced data sets. These methods consider the best candidate attribute, trying to find a minimal reduct.

However, hill-climbing methods do not guarantee to find an optimal or minimal reduct. As no perfect heuristic exists, there can be no guarantee of optimality. Using attribute significance to discriminate between candidates may lead the search down a non-minimal path. It is impossible to predict which combinations of attributes will lead to an optimal reduct with the addition or deletion of single attributes.

Some researchers use stochastic methods for rough set feature selection (Bazan, 2000). Wróblewski uses genetic algorithms to find minimal reducts (Wróblewski, 1995). He combines a genetic algorithm with a greedy algorithm to generate short reducts. However, it uses highly time-consuming operations and cannot assure that the resulting subset is really a reduct. Bjorvand applies genetic algorithms to compute approximate reducts (Bjorvand, 1997a, 1997b). He takes Wróblewski's work as a foundation, but makes several variations and practical improvements both in speed and the quality of approximation. To obtain a good initial population for the GA, Bjorvand includes the attribute core in all candidates. In addition to this, he uses domain knowledge to get the average size of actual reducts and lets the number of features in the candidates be similar to the number in the reducts. Also, he allows the user to assign a relative weight to each attribute when creating the initial population. To avoid wasting much processing power in a wrong search direction, he adopts a dynamic mutation rate that is proportional to the redundancy in the population, preventing all individuals from becoming equal. Zhai proposes an integrated feature extraction approach based on rough set theory and genetic algorithms (Zhai, 2002). Rough sets are used to perform consistency checks, concept formation and approximation. By calculating the lower and upper approximations, training data is split into certain training data and possible training data. Then, a GA

discovers best rules from the data sets. The fitness function is defined as the classification quality of the extracted rules. Ultimately, the features or attributes within rules with highest indices are selected. Jensen finds minimal rough set reducts using another stochastic strategy, Ant Colony Optimization (ACO) (Jensen and Shen, 2003).

Hill-climbing methods are more efficient when dealing with little noise and a small number of interacting features, but are not assured of optimality. Stochastic methods can provide a more robust solution at the expense of increased computational effort (Vafaie and Imam, 1994). For systems where the optimal or minimal subset is required (perhaps due to the cost of feature measurement), stochastic feature selection must be used.

In this article we propose a new feature selection mechanism, investigating how particle swarm optimization (PSO) can be applied to find optimal feature subsets or rough set reducts. PSO is a new evolutionary computation technique proposed by Kennedy and Eberhart (Kennedy and Eberhart, 1995a, 1995b). The particle swarm concept was motivated from the simulation of social behavior. The original intent was to graphically simulate the graceful but unpredictable movement of bird flocking. The PSO algorithm mimics the behavior of flying birds and their means of information exchange to solve optimization problems. Each potential solution is seen as a particle with a certain velocity, and "flies" through the problem space. Each particle adjusts its flight according to its own flying experience and its companions' flying experience. The particle swarms find optimal regions of complex search spaces through the interaction of individuals in a population of particles. PSO has been successfully applied to a large number of difficult combinatorial optimization problems; studies show that it often outperforms Genetic Algorithms (Kennedy and Spears, 1998). PSO is particularly attractive for feature selection in that

particle swarms will discover the best feature combinations as they fly within the problem space. The performance of the proposed algorithm is evaluated using several UCI datasets. It can be seen that PSO has a strong search capability in the problem space and can discover optimal solutions quickly.

The rest of this paper is structured as follows. Section 2 describes the fundamentals of rough set theory. The principles of PSO and PSO for Rough set-based Feature Selection algorithm (PSORSFS) are presented in Section 3. The effectiveness of the method is demonstrated, compared with other algorithms on UCI datasets and discussed in Section 4. Finally, Section 5 concludes the article.

The algorithms used in the comparison include the positive region-based attribute reduction algorithm (POSAR) (Jensen and Shen, 2003; Hu, 1995b; Wang, 2004), conditional entropy-based attribute reduction (CEAR) (Wang, 2002, 2004), discernibility matrix-based attribute reduction (DISMAR) (Hu, 2003) and GA-based attribute reduction (GAAR) (Wroblewslki, 1995; Bazan, 1998, 2000). Due to paper length restrictions, we do not describe such algorithms here, more details can be found in the related references.

2. Rough set preliminaries

Rough set theory (Pawlak, 1991, 1997) is a new mathematical approach to imprecision, vagueness and uncertainty. In an information system, every object of the universe is associated with some information. Objects characterized by the same information are indiscernible with respect to the available information about them. Any set of indiscernible objects is called an elementary set. Any union of elementary sets is referred to as a crisp set- otherwise a set is rough (imprecise, vague). Vague concepts cannot be characterized in terms of information about their elements. A rough set is the

approximation of a vague concept by a pair of precise concepts, called lower and upper approximations. The lower approximation is a description of the domain objects which are known with certainty to belong to the subset of interest, whereas the upper approximation is a description of the objects which possibly belong to the subset. Relative to a given set of attributes, a set is rough if its lower and upper approximations are not equal.

The main advantage of rough set analysis is that it requires no additional knowledge except for the supplied data. Rough sets perform feature selection using only the granularity structure of the data (Jensen and Shen, 2003).

Let I=(U, A) be an information system, where U is the universe, a non-empty finite set of objects. A is a non-empty finite set of attributes. For $\forall a \in A$ determines a function $f_a : U \to V_a$. If $P \subseteq A$, there is an associated equivalence relation:

$$IND(P) = \{(x, y) \in U \times U \mid \forall a \in P, f_a(x) = f_a(y)\} \qquad (1)$$

The partition of U, generated by IND(P) is denoted U/P. If $(x, y) \in IND(P)$, then x and y are indiscernible by attributes from P. The equivalence classes of the P-indiscernibility relation are denoted $[x]_P$. The indiscernibility relation is the mathematical basis of rough set theory.

Let $X \subseteq U$, the P-lower approximation $\underline{P}X$ and P-upper approximation $\overline{P}X$ of set X can be defined as:

$$\underline{P}X = \{x \in U \mid [x]_P \subseteq X\} \qquad (2)$$

$$\overline{P}X = \{x \in U \mid [x]_P \cap X \neq \phi\} \qquad (3)$$

Let $P, Q \subseteq A$ be equivalence relations over U, then the positive, negative and boundary regions can be defined as:

$$POS_P(Q) = \bigcup_{X \in U/Q} \underline{P}X \qquad (4)$$

$$NEG_P(Q) = U - \bigcup_{X \in U/Q} \overline{P}X \qquad (5)$$

$$BND_P(Q) = \bigcup_{X \in U/Q} \overline{P}X - \bigcup_{X \in U/Q} \underline{P}X \qquad (6)$$

The positive region of the partition U/Q with respect to P, $POS_P(Q)$, is the set of all objects of U that can be certainly classified to blocks of the partition U/Q by means of P. A set is rough (imprecise) if it has a non-empty boundary region.

An important issue in data analysis is discovering dependencies between attributes. Dependency can be defined in the following way. For $P, Q \subseteq A$, P depends totally on Q, if and only if $IND(P) \subseteq IND(Q)$. That means that the partition generated by P is finer than the partition generated by Q. We say that Q depends on P in a degree k $0 \le k \le 1$ denoted $P \Rightarrow_k Q$, if

$$k = \gamma_P(Q) = \frac{|POS_P(Q)|}{|U|} \qquad (7)$$

If k=1, Q depends totally on P, if 0<k<1, Q depends partially on P, and if k=0 then Q does not depend on P. In other words, Q depends totally (partially) on P, if all (some) objects of the universe U can be certainly classified to blocks of the partition U/Q, employing P.

In a decision system the attribute set contains the condition attribute set C and decision attribute set D, i.e. $A = C \cup D$. The degree of dependency between condition and decision attributes, $\gamma_C(D)$, is called the quality of approximation of classification, induced by the set of decision attributes (Pawlak, 1997).

The goal of attribute reduction is to remove redundant attributes so that the reduced set provides the same quality of classification as the original. A reduct is defined as a subset R

of the conditional attribute set C such that $\gamma_R(D) = \gamma_C(D)$. A given decision table may have many attribute reducts, the set of all reducts is defined as:

$$\mathrm{Re}\,d = \{R \subseteq C \mid \gamma_R(D) = \gamma_C(D), \forall B \subset R, \gamma_B(D) \neq \gamma_C(D)\} \quad (8)$$

In rough set attribute reduction, a reduct with minimal cardinality is searched for. An attempt is made to locate a single element of the minimal reduct set $\mathrm{Re}\,d_{\min} \subseteq \mathrm{Re}\,d$:

$$\mathrm{Re}\,d_{\min} = \{R \in \mathrm{Re}\,d \mid \vee R' \in \mathrm{Re}\,d, |R| \leq |R'|\} \quad (9)$$

The intersection of all reducts is called the core, the elements of which are those attributes that cannot be eliminated. The core is defined as:

$$Core(C) = \cap \mathrm{Re}\,d \quad (10)$$

3. PSO for Feature Selection

3.1 The principle of PSO

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart in 1995 (Kennedy and Eberhart, 1995a, 1995b). The original intent was to graphically simulate the graceful but unpredictable movements of a flock of birds. Initial simulations were modified to form the original version of PSO. Later, Shi introduced inertia weight into the particle swarm optimizer to produce the standard PSO (Shi and Eberhart, 1998a, 2001).

PSO is initialized with a population of random solutions, called 'particles'. Each particle is treated as a point in an S-dimensional space. The *i*th particle is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{iS})$. The best previous position (pbest, the position giving the best fitness value) of any particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, ..., p_{iS})$. The index of

the best particle among all the particles in the population is represented by the symbol 'gbest'. The rate of the position change (velocity) for particle i is represented as $V_i = (v_{i1}, v_{i2}, ..., v_{iS})$. The particles are manipulated according to the following equation:

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id}) \qquad (11)$$

$$x_{id} = x_{id} + v_{id} \qquad (12)$$

Where $d = 1,2,...,S$, w is the inertia weight, it is a positive linear function of time changing according to the generation iteration. Suitable selection of the inertia weight provides a balance between global and local exploration, and results in fewer iterations on average to find a sufficiently optimal solution. The acceleration constants c_1 and c_2 in equation (11) represent the weighting of the stochastic acceleration terms that pull each particle toward pbest and gbest positions. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement toward, or past, target regions. rand() and Rand() are two random functions in the range [0,1].

Particles' velocities on each dimension are limited to a maximum velocity, Vmax. It determines how large steps through the solution space each particle is allowed to take. If Vmax is too small, particles may not explore sufficiently beyond locally good regions. They could become trapped in local optima. On the other hand, if Vmax is too high particles might fly past good solutions.

The first part of equation (11) provides the "flying particles" with a degree of memory capability allowing the exploration of new search space areas. The second part is the "cognition" part, which represents the private thinking of the particle itself. The third part is the "social" part, which represents the collaboration among the particles. Equation (11)

is used to calculate the particle's new velocity according to its previous velocity and the distances of its current position from its own best experience (position) and the group's best experience. Then the particle flies toward a new position according to equation (12). The performance of each particle is measured according to a pre-defined fitness function.

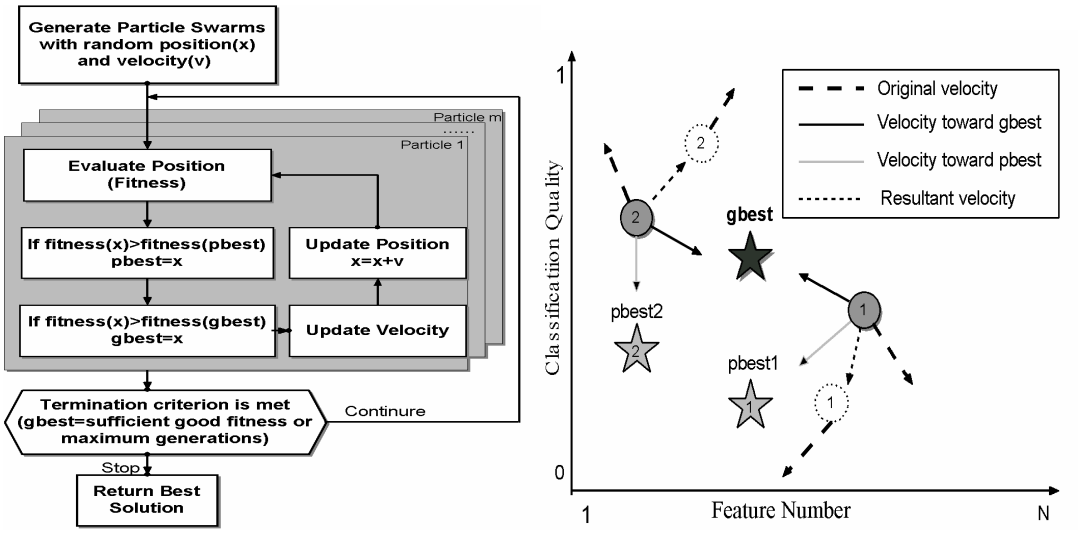

Figure 1: (a) PSO process. (b) PSO-based Feature Selection. The principle of updating velocity. Individual particles (1 and 2) are accelerated toward the location of the best solution, gbest, and the location of their own personal best, pbest, in the two dimension problem space (feature number and classification quality)

We give the pseudo code of the PSO algorithm here, with a graphic demonstration of PSO given in figure 1.

Algorithm PSO

Input:

m: the swarm size; c_1, c_2: positive acceleration constants; w: inertia weight

MaxV: maximum velocity of particles

MaxGen: maximum generation

MaxFit: maximum fitness value

Output:

P_{gbest}: Global best position

Begin

Swarms $\{x_{id}, v_{id}\}$ =Generate(m); /* Initialize a population of particles with random positions and velocities on S dimensions*/

Pbest(i)=0; $i = 1,...,m, d = 1,...,S$

Gbest=0; Iter=0;

While(Iter<MaxGen and Gbest<MaxFit)

{ For(every particle i)

{ Fitness(i)=Evaluate(i);

IF(Fitness(i)>Pbest(i))

{Pbest(i)=Fitness(i); $p_{id} = x_{id}$; $d = 1,...,S$ }

IF(Fitness(i)>Gbest)

{Gbest=Fitness(i); gbest=i;}

}

For(every particle i)

{ For(every d){

$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id})$

IF($v_{id} > MaxV$) { $v_{id} = MaxV$;}

IF($v_{id} < -MaxV$) { $v_{id} = -MaxV$;}

$x_{id} = x_{id} + v_{id}$

}

}

```
    Iter=Iter+1;
  }/*rand() and Rand() are two random functions in the range [0,1]*/
  Return P_{gbest}
End
```

3.2 PSO and Rough set-based Feature Selection (PSORSFS)

We can use the idea of PSO for the optimal feature selection problem. Consider a large feature space full of feature subsets. Each feature subset can be seen as a point or position in such a space. If there are N total features, then there will be 2^N kinds of subset, different from each other in the length and features contained in each subset. The optimal position is the subset with least length and highest classification quality. Now we put a particle swarm into this feature space, each particle takes one position. The particles fly in this space, their goal is to fly to the best position. Over time, they change their position, communicate with each other, and search around the local best and global best position. Eventually, they should converge on good, possibly optimal, positions. It is this exploration ability of particle swarms that should better equip it to perform feature selection and discover optimal subsets.

To apply the PSO idea to feature selection, some matters must first be considered.

Representation of position

We represent the particle's position as binary bit strings of length N, where N is the total number of attributes. Every bit represents an attribute, the value `1' means the corresponding attribute is selected while `0' not selected. Each position is an attribute subset.

Representation of Velocity

The velocity of each particle is represented as a positive integer, varying between 1 and Vmax. It implies how many of the particle's bits (features) should be changed, at a particular moment in time, to be the same as that of the global best position, i.e. the velocity of the particle flying toward the best position. The number of different bits between two particles relates to the difference between their positions. See Fig.1(b) for the principle of velocity updating.

For example, P_{gbest}=[1 0 1 1 1 0 1 0 0 1], P_i=[0 1 0 0 1 1 0 1 0 1]. The difference between gbest and the particle's current position is P_{gbest}-P_i=[1 –1 1 1 0 –1 1 –1 0 0]. A value of 1 indicates that compared with the best position, this bit (feature) should be selected but is not, which will decrease classification quality and lead to a lower fitness value. Assume that the number of 1's is *a*. On the other hand, a value of -1 indicates that, compared with the best position, this bit should not be selected, but is selected. Redundant features will make the length of the subset longer and lead to a lower fitness value. The number of -1's is *b*. We use the value of *(a-b)* to express the distance between two positions; *(a-b)* may be positive or negative. Such variation makes particles exhibit an exploration ability within the solution space. In this example, *(a-b)*=4-3=1, so $P_g - P_i$=1.

Position Update Strategies

After updating the velocity, a particle's position will be updated by the new velocity. Assume that the new velocity is V; the number of different bits between the current particle and gbest is x_g. Two cases exist when updating the position:

1) V<=x_g. In this case, the particle's velocity is less than, or equal to, the position

difference between the particle and gbest. V bits of the particle are randomly changed, different from that of gbest. The particle then moves toward the global best while still exploring the search space, instead of simply being same as gbest.

2) V> x_g . In this case, the particle's velocity overruns the position difference between the particle and gbest. In addition to changing all the different bits to be same as that of gbest, we should further randomly ('random' implies 'exploration ability') change (V- x_g) bits outside the different bits between the particle and gbest. So after the particle reaches the global best position, it keeps on moving some distance toward other directions, enabling further search.

Velocity Limitation (Maximum Velocity, Vmax)

The maximum velocity Vmax serves as a constraint to control the global exploration ability of a particle swarm. A larger Vmax facilitates global exploration, while a smaller Vmax encourages local exploitation. When Vmax is too low, particles have more difficulty escaping from locally optimal regions. If Vmax is too high, particles might fly past good solutions (Kennedy, 1997).

In our experimentation, we initially limited the particles' velocity in the region [1, N]. However, it was noticed that after several generations the swarms converged to a good but non-optimal solution, and in the following generations the gbest remained stationary. This indicates that the velocity is too high and particles often 'fly past' the optimal solution.

We set Vmax= (1/3)*N and limit the velocity within the range [1, (1/3)*N], which prevents an overly-large velocity. A particle can be near to an optimal solution, but a high velocity may make it move far away. By limiting the maximum velocity, particles cannot fly too far away from the optimal solution. Once finding a global best position, other

particles will adjust their velocities and positions, searching around the best position. After many tests, we found that an appropriate maximum velocity value is (1/3)*N. If V<1, then V=1. If V>(1/3)*N, V=(1/3)*N. PSO can often find the optimal solution quickly under such a limit.

Fitness Function

We define the fitness function in equation (13):

$$Fitness = \alpha * \gamma_R(D) + \beta * \frac{|C| - |R|}{|C|} \qquad (13)$$

Where $\gamma_R(D)$ is the classification quality of condition attribute set R relative to decision D, |R| is the '1' number of a position or the length of selected feature subset. |C| is the total number of features. α and β are two parameters corresponding to the importance of classification quality and subset length, $\alpha \in [0,1]$ and $\beta = 1 - \alpha$.

This formula means that the classification quality and feature subset length have different significance for feature selection task. In our experiment we assume that classification quality is more important than subset length and set $\alpha = 0.9, \beta = 0.1$. The high α assures that the best position is at least a real rough set reduct. The goodness of each position is evaluated by this fitness function. The criteria are to maximize fitness values.

4. Experimental Results and Discussions

We implement the PSORSFS algorithm and other four feature selection algorithms in MatLab 6.5. The computer is Intel P4, 2.66 GHz CPU; 512MB RAM and the system is Windows XP Professional. The five algorithms are tested and compared on 27 discrete

UCI datasets (Blake et al., 1998). The two algorithms, GAAR and PSORSFS require additional parameter settings for their operation. These are given in Table 1.

Table 1 PSORSFS&GAAR parameter settings

	Population	Generation	Crossover Probability	Mutation Probability	c_1	c_2	weight	Velocity
GA	100	100	0.6	0.4	-	-	-	-
PSO	20	100	-	-	2.0	2.0	1.4~0.4	1~(1/3)*N

In PSORSFS, the inertia weight decreases along with the iterations, varying from 1.4 to 0.4 according to the equation (14).

$$\text{Weight} = (\text{weight}-0.4) * (\text{MAXITER} - \text{Iter}) / \text{MAXITER} + 0.4 \quad (14)$$

Where MAXITER is the maximum iteration (generation) and Iter is the current iteration.

For 14 of 27 datasets, the five algorithms find the same reducts. These are listed in Table 2. The 'Features' column and 'Instances' column give out the total number of features (attributes) and instances in the datasets. The number of decision rules and the classification accuracy with different reducts are also shown. We use the LEM2 algorithm (Stefanowski, 1998) to extract rules from the data and the global strength (Bazan, 1998, 2000) for rule negotiation in classification. We apply ten-fold cross validation to estimate the classification accuracy. We also show the time in seconds for 100 generations necessary for the generation of reducts by PSORSFS.

Table 2 Experimental results on 14 datasets

Dataset	Features	Instances	Reduct size	Number of rules	Classification accuracy (%)	Time of PSORSFS (s)
Balloon1	4	20	2	4	100	10.172
Balloon2	4	20	2	8	100	9.75
Balloon3	4	20	2	4	100	10.984
Balloon4	4	16	4	6	80	14.078
Balance-Scale	4	625	4	207	88.5	3001.1
Lenses	4	24	4	9	87.6	24.844
Hayes-Roth	4	132	3	15	89.5	95.422
Corral	6	64	4	6	100	36.016
Monk1	6	124	3	20	93.5	103.406

Monk2	6	169	6	82	65.4	297.875
Monk3	6	432	3	9	97.2	327.406
PostoperativePatient	8	90	8	37	59.9	138.75
Parity5+2	10	1024	5	128	100	615.844
Parity5+5	10	1024	5	32	100	722.14

Table 3 Reduct sizes found by Feature Selection algorithms

Dataset	Features	Instances	Rough set Reduction Algorithms				
			POSAR	CEAR	DISMAR	GAAR	PSORSFS
Tic-tac-toe	9	958	8	7	8	8	8
Breastcancer	9	699	4	4	5	4	4
M-of-N	13	1000	7	7	6	6	6
Exactly	13	1000	8	8	6	6	6
Exactly2	13	1000	10 **	11	10**	11	10**
Vote	16	300	9	11	8**	9	8**
Zoo	16	101	5	10	5	6	5
Lymphography	18	148	6**	8	7	8	7
Mushroom	22	8124	5	5	6	5	4*
Led	24	2000	6	12	18	8	5**
Soybean-small	35	47	2	2	2	6	2
Lung	56	32	4	5	4	6	4
DNA	57	318	7	6	6	7	6

* Optimal solution ** Exclusive optimal solution

Table 4 Classification results with different reducts
1: Number of rules; 2: Classification accuracy

Dataset	POSAR		CEAR		DISMAR		GAAR		PSORSAR		
	1	2	1	2	1	2	1	2	1	2	Time (s)
Tic-tac-toe	93	94.42	126	77.89	161	86.21	91	93.05	70	96.32	5719
Breastcancer	67	95.94	75	94.20	67	95.94	64	95.65	64	95.80	1686.4
M-of-N	35	100	35	100	35	100	35	100	35	100	1576.6
Exactly	50	100	50	100	50	100	50	100	50	100	1671.6
Exactly2	217	83.7	178	69.6	230	83	200	80.8	217	83.7	5834.0
Vote	25	94.33	25	92.33	28	93.67	25	94.0	25	95.33	424.17
Zoo	13	96.0	13	94.0	13	94	13	92.0	10	96.0	87.5
Lymphography	32	85.71	42	72.14	40	74.29	38	70.0	39	75.71	336.172
Mushroom	19	100	61	90.83	19	100	19	100	23	99.70	14176
Led	10	100	228	83.10	257	78.85	10	100	10	100	1758.3
Soybean-small	5	100	4	100	4	100	4	97.50	4	100	25.719
Lung	11	86.67	13	73.33	14	73.3	12	70.0	8	90.0	26.203
DNA	173	33.23	192	26.45	191	36.45	191	33.87	169	49.68	1667.0

Table 5 Experimental Results by RSES

Dataset	Total Reducts	Minimal size Of reducts	Number of rules	Classification accuracy (%)
Tic-tac-toe	9	8	1085	100
Breastcancer	20	4	455	94.4
M-of-N	1	6	19	92.9
Exactly	1	6	41	85.9
Exactly2	1	10	242	76.4
Vote	2	8	67	92.9
Zoo	33	5	109	96.8
Lymphography	424	6	353	85.9
Mushroom	292	4	480	98.3
Led	140	5	6636	100
Soybean-small	--	2	31	92.5
Lung	--	4	100	75
DNA	--	5	2592	74.3

Note: For the last three datasets (Soybean-small, Lung and DNA), RSES cannot find all reducts by the exhaustive algorithm as it requires too much memory. Instead, we use a fast genetic algorithm to find 10 reducts.

The experimental results of the other 13 datasets are listed in Table 3. The five rough set reduction algorithms are compared. We present the best solution (in terms of feature subset length) each algorithm finds. For these datasets, some may have more than one optimal reduct; some however have only one (exclusive) optimal reduct. From the results, it can be seen that in some situations hill-climbing methods can locate the optimal solution. For example, POSAR finds the exclusive optimal solution for dataset Exactly2 and Lymphography. DISMAR finds the exclusive optimal solution for dataset Exactly2 and Vote. But for other datasets, sub-optimal solutions are found, containing redundant features. CEAR often contains more redundant features than POSAR and DISMAR.

As for the stochastic search algorithms, GAAR and PSORSFS, from the experimental results it can be seen that PSORSFS performs better than GAAR. PSORSFS successfully finds the optimal reducts on most of the datasets. For example, PSORSFS finds an optimal reduct for the Mushroom data, and finds the exclusive optimal reduct for Exactly2, Vote

and Led.

During the experiments, the rough ordering of techniques with respect to time is: POSAR<CEAR<PSORSAR<GAAR<DISMAR. DISMAR takes a significant amount of time for the computation of the discernibility matrix, the time increasing quickly with increasing number of instances in the dataset. CEAR may cost time on computing equivalence classes. The stochastic algorithms need time in generation iterations.

Let N be the number of features (conditional attributes) and M the total objects. The time complexity of POSAR is $O(NM^2)$ (Nguyen, 1996; Hu, 1995b), and that of the reduction based on conditional information entropy (CEAR) is $O(NM^2)+O(M^3)$, which is composed of the computation of core and non-core attribute reduct (Wang, 2002). DISMAR has total time complexity $O((N+\log M)M^2)$ (Hu, 2003). For GAAR and PSORSFS, the complexity of the fitness function is $O(NM^2)$ (Wroblewski, 1995). The other impact on time is the number of generation iterations. For the GA method, the crossover and mutation operations also take much time. As for PSORSFS, time is mainly spent on evaluating the particles' position (fitness function).

To graphically illustrate the progress of the particle swarm as it searches for optimal solutions, we take generation as the horizontal coordinate and the fitness value as the vertical coordinate. This should illustrate the process of improvement of the global best as the number of generations increase. To further highlight the search process, we graph subset length of every particle's current position (horizontal coordinate) against classification quality (vertical coordinate). Each point in the figure is a particle.

The example for Exactly2 is given below and the other examples are listed in appendix A.

Example 1

The process of the particle swarms searching for optimal solutions for dataset Exactly2 is given in Table 6 and Figures 2 and 3. In Table 6, "Best Solution" lists the best feature subset encountered at a particular iteration, in which each number denotes one feature of the dataset.

Table 6 PSO searching process on Exactly2

Iter	Best Solution	Fitness Value	Feature Subset Length
1	1,2, 4, 5, 7, 8, 9, 10, 11, 12, 13	0.8272	11
2	1,2, 4, 5, 6, 7, 8, 9, 10, 11, 13	0.8362	11
3	1,2, 4, 5, 6, 7, 8, 9, 10, 11, 13	0.8362	11
4--11	1,2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	0.8663	12
12	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13	0.9154	11
13	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	0.9231	10

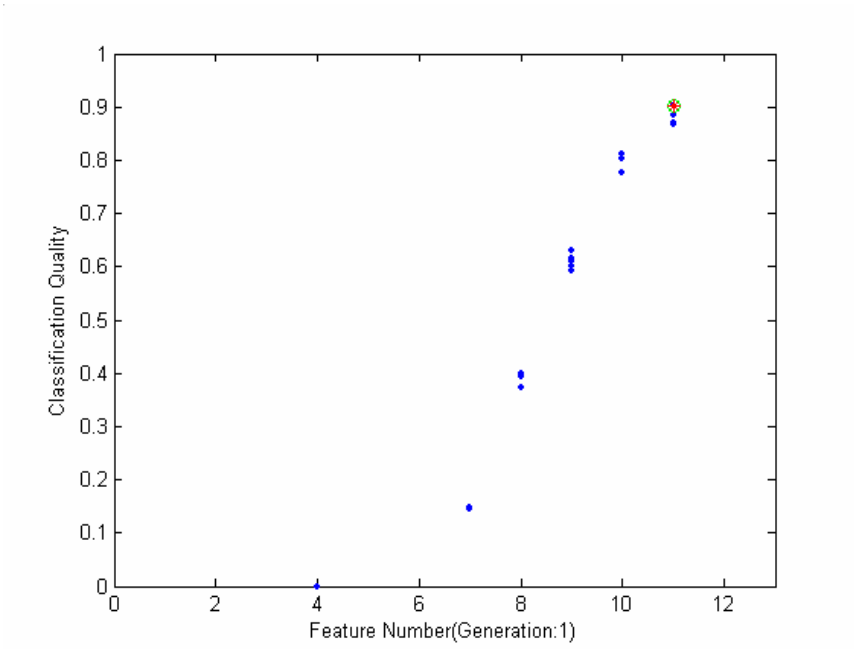

Fig.2 (a) Generation 1 of PSO on dataset Exactly2

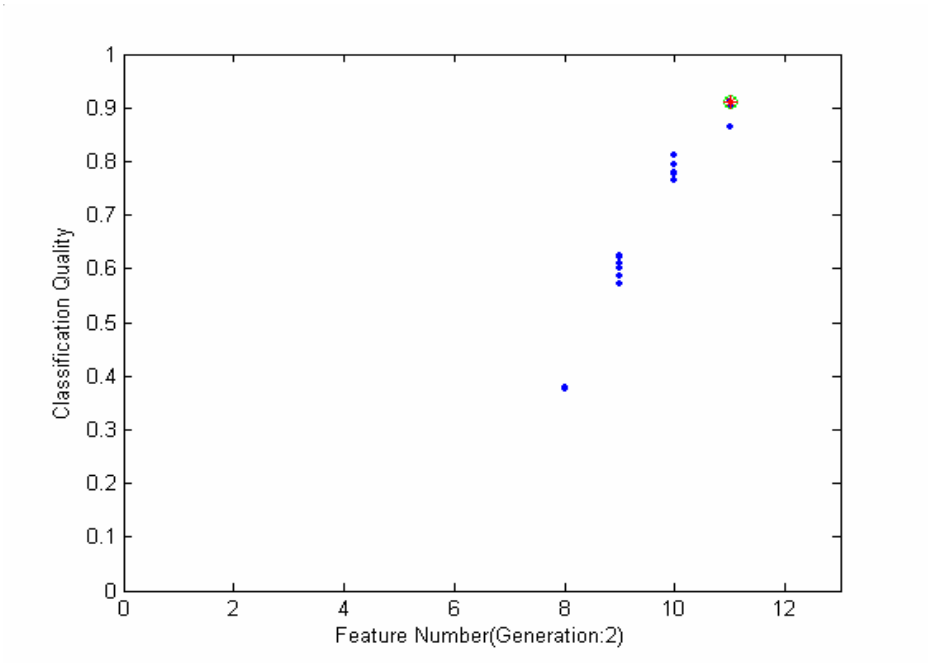

Fig.2 (b) Generation 2 of PSO on dataset Exactly2

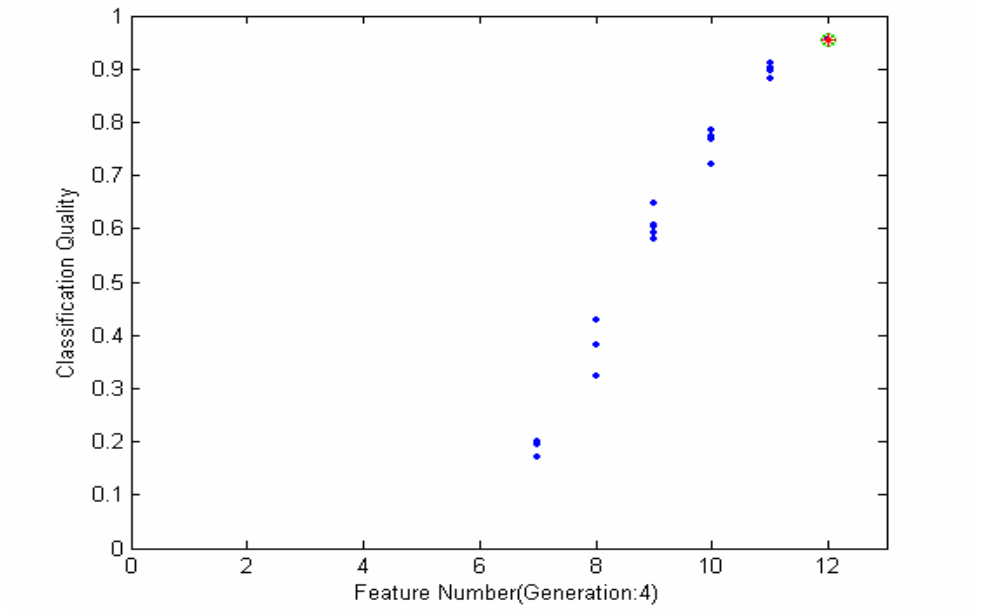

Fig.2 (c) Generation 4 of PSO on dataset Exactly2

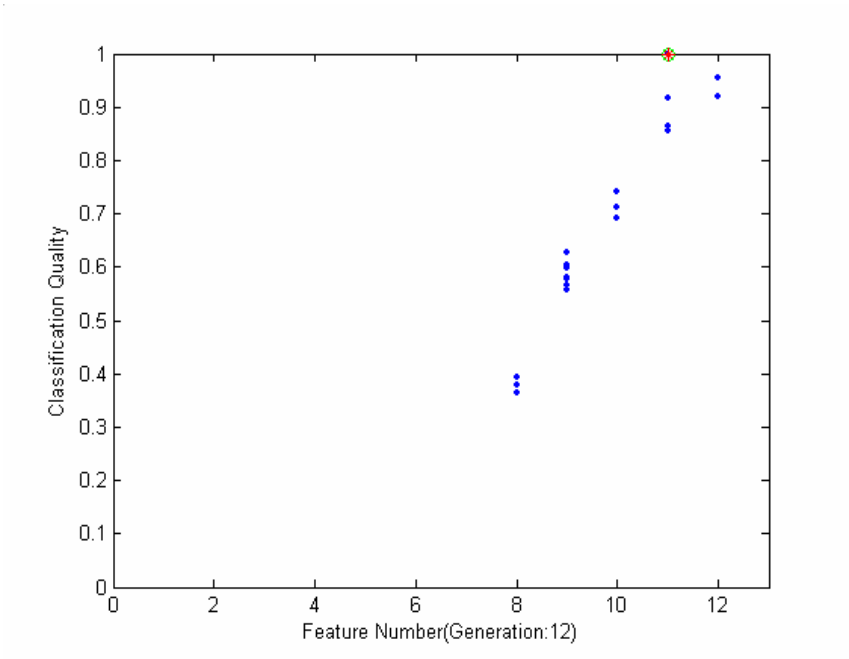

Fig.2 (d) Generation 12 of PSO on dataset Exactly2

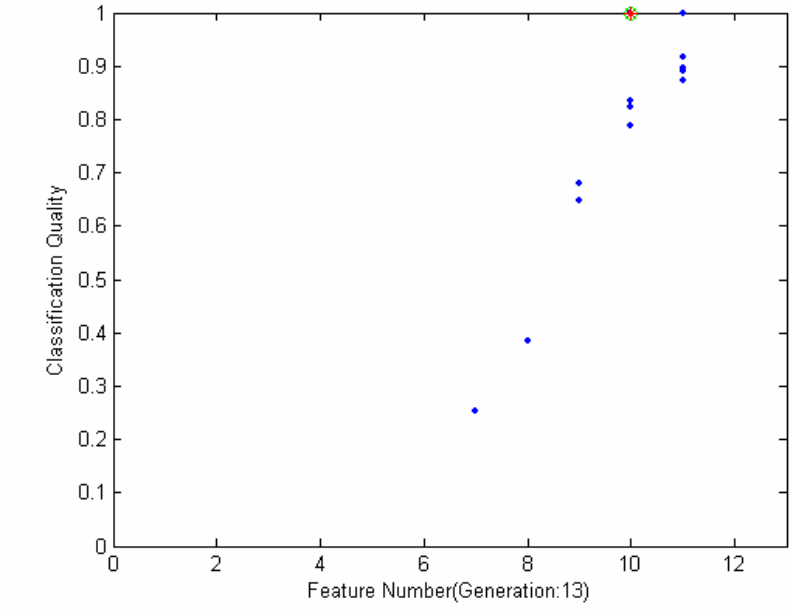

Fig.2 (e) Generation 13 of PSO on dataset Exactly2

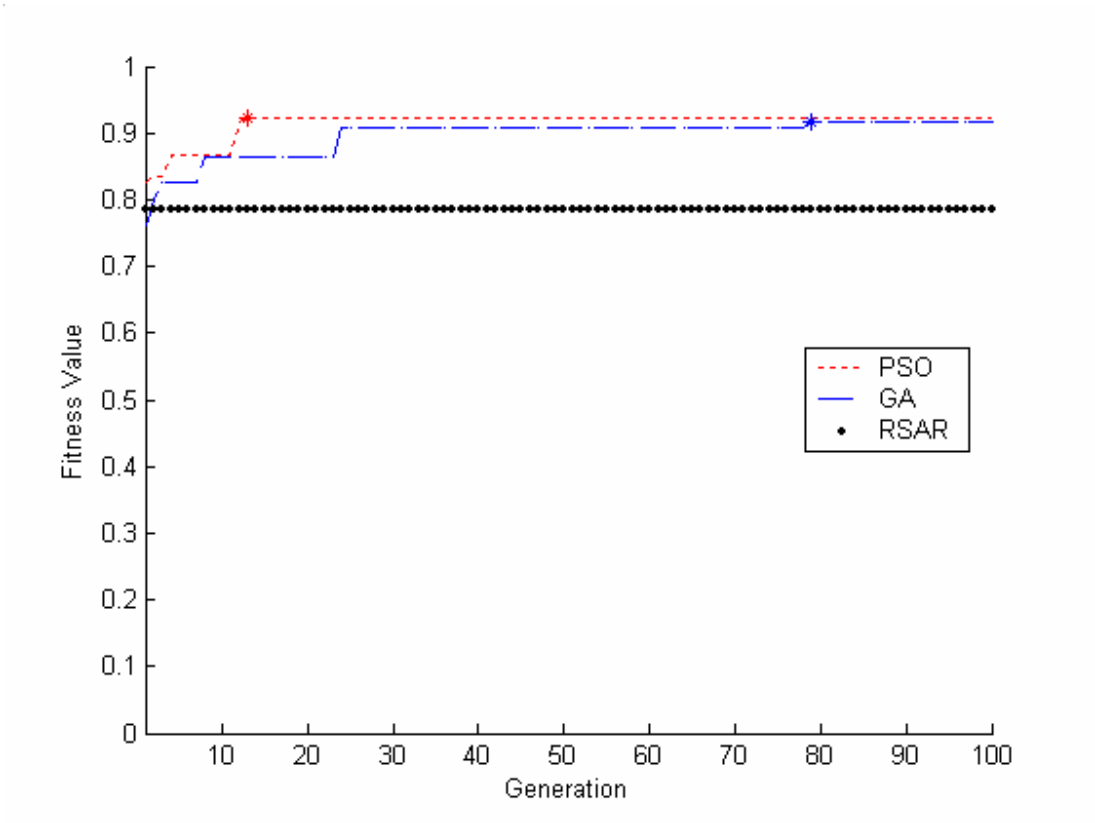

Fig.3 Evolution process of the global best on dataset Exactly2.
(PSORSFS, GAAR, CEAR)

The evaluation criteria or fitness function is a critical issue in the implementation of stochastic algorithms. In the fitness function we use, because the classification quality parameter outclasses that of subset length ($\alpha = 0.9$, $\beta = 0.1$), the optimal solution is assured to be a real rough set reduct or super-reduct. So, the fitness function in PSO is efficient.

From the results and figures, we can see that, compared with GA, PSO is quicker in locating the optimal solution. In general, it can find the optimal solution within tens of generations. If exhaustive search is used to find the optimal reduct in the dataset DNA, there will be tens of thousands of candidate subsets, which is impossible to execute. But with PSO, at the 32nd generation the optimal solution is found.

PSO has a powerful exploration ability; it is a gradual searching process that approaches optimal solutions. The running time of PSO is affected less by the problem dimension (feature numbers), but more by the size of data (see Tables 2 and 4). Mushroom is the largest dataset and requires the largest time for PSORSFS (14176 seconds). Most of the time is spent on the basic computations of rough sets (computing equivalence classes). For some datasets with more features, for example Lung and DNA, after finding a sub-optimal solution, the GA cannot find a better one: the fitness/generation graph is a line. However, PSO can search in the feature space until the optimal solution is found. The GA is affected greatly by the number of features.

PSO comprises a very simple concept, and the ideas can be implemented in a few lines of computer code. It requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory requirements and speed. This optimization technique does not suffer, however, from some of the difficulties of GAs; interaction in the group enhances rather than detracts from progress toward the solution. Further, a particle swarm system has memory, which the genetic algorithm does not have. Changes in genetic populations result in the destruction of previous knowledge of the problem. In particle swarm optimization, individuals who fly past optima are tugged to return towards them; knowledge of good solutions is retained by all particles.

As for PSO, every particle flies in the candidate problem space, adjusts their velocity and position according to the local best and global best. So, all the particles have a powerful search capability, which can help the swarm avoid dead ends.

The comparison of the number of decision rules and the classification accuracy with different reducts are shown in Table 4. We show the time in seconds for 100 generations

necessary for generation of reducts by PSORSFS.

The results are also compared with the rough set system RSES (Skowron et al, 2005) (see Table 5). RSES is an excellent toolset for analyzing data with the use of methods based on Rough Set Theory, and is developed by researchers in Warsaw University. We use the Exhaustive algorithm to find reducts. It presents two deterministic algorithms for the computation of the whole reduct set, both algorithms compute the discernibility matrix for this purpose (Bazan et al, 2000). The complexity is exponential. The time complexity to find all reducts is $O(2^N T)$, where T is the computational cost of finding one reduct, and N is the number of attributes (Bell, 1998). For the last three datasets (Soybean-small, Lung and DNA), RSES cannot find all reducts by the exhaustive algorithm, due to memory limitations. As a result of this, we use a fast genetic algorithm to find 10 reducts. We use the decision rules classifier, LEM2 algorithm for global rules generation and 10-fold cross-validation estimation method. We set "Shortening ratio" to 0.9, in order to get a short and minimal rule set. Conflicts are resolved by Standard Voting. Most of the reducts found by PSORSFS result in smaller rules and exhibit higher classification accuracy.

5. Conclusion

This paper discusses the shortcomings of conventional hill-climbing rough set approaches to feature selection. These techniques often fail to find optimal reductions, as no perfect heuristic can guarantee optimality. On the other hand, complete searches are not feasible for even medium-sized datasets. So, stochastic approaches provide a promising feature selection mechanism.

We propose a new optimal feature selection technique based on rough sets and Particle Swarm Optimization (PSO). PSO has the ability to quickly converge (Shi and Eberhart,

1999), it has a strong search capability in the problem space and can efficiently find minimal reducts. Experimental results demonstrate competitive performance. PSO is a promising method for rough set reduction.

More experimentation and further investigation into this technique may be required. The inertia weight (w) and maximum velocity (Vmax) have an important impact on the performance of PSO. The selection of the parameters may be problem-dependent. Vmax serves as a constraint that controls the maximum global exploration ability PSO can have. In many practical problems, it's difficult to select the best Vmax without trial-and-error (Shi and Eberhart, 1998b). In our feature selection problem, we first limit the particles' velocity to N, since [1, N] is the dynamic range of the feature space of each particle. But we find that under such a limitation, particles have poor local exploration ability. So after many tests, we set Vmax to (1/3)*N, which is suitable for our problem (see Section 3.2). The inertia weight balances the global and local exploration abilities. In our experiments, we let it decrease from 1.4 to 0.4 along with the iterations. The performance of the PSO algorithm with linearly decreasing inertia weight can be improved greatly and have better results. The larger inertia weights at the beginning help to find good seeds and the later small inertia weights facilitate fine search (Shi and Eberhart, 1998b, 1999).

In this paper, we apply PSO to find reducts of minimal cardinality and, like classical genetic algorithms, the particle's position is a binary representation for attribute subsets. An extension to the approach would be to select reducts due to the number of decision rules it generates rather than its length alone. If a reduct generates fewer rules, it means that the rules are more general and they should better recognize new objects (Bazan et al, 2000). We should also extend to hybrid algorithms (Wroblewski, 1995, 1996),

order-based PSO for searching approximate entropy reducts (Slezak and Wroblewski, 2003), where the particle's position is a permutation of attributes and PSO is used to find the proper order. Such reducts are much more applicable in practice. The fitness function and position-updating strategy are also key factors in PSO for feature selection, which need to be improved further.

References

Bazan, J., Nguyen, H.S., Nguyen, S.H., Synak, P., Wroblewski, J., 2000. Rough set algorithms in classification problem. Polkowski, L., Tsumoto, S. and Lin, T.Y. (eds.): Rough Set Methods and Applications. Physica-Verlag, Heidelberg, New York, pp. 49-88.

Bazan, J., 1998. A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Table. In: Polkowski, L., Skowron, A. (eds.): Rough Sets in Knowledge Discovery. Heidelberg: Physica-Verlag, pp. 321-365.

Bell, D., Guan, J., 1998. Computational methods for rough classification and discovery. Journal of ASIS. 49 (5), 403–414.

Bjorvand, A.T., Komorowski, J., 1997a. Practical Applications of Genetic Algorithms for Efficient Reduct Computation, Wissenschaft & Technik Verlag, 4, 601-606.

Bjorvand, A.T., 1997b. 'Rough Enough'-a system supporting the rough sets approach. Sixth Scandinavian Conference on Artificial Intelligence SCAI'97.

Blake, C., Keogh, E., Merz, C. J., 1998. UCI repository of machine learning databases. Tech. rep., Department of Information and Computer Science, University of California, Irvine, CA, http://www.ics.uci.edu/mlearn/MLRepository.htm.

Chouchoulas, A., Shen, Q., 2001. Rough set-aided keyword reduction for text Categorization. Applied Artificial Intelligence. 15 (9), 843-873.

Eberhart R.C., Shi, Y., 2001. Particle swarm optimization: Developments, applications and

resources. Proc. IEEE Int. Conf. On Evolutionary Computation. Seoul, pp. 81-86.

Guyon, I., Elisseeff, A., 2003. An Introduction to Variable and Feature Selection. Journal of Machine Learning Research. 3, 1157-1182.

Hu, K., Lu, Y.C., Shi, C.Y., 2003. Feature ranking in rough sets. AI Communications 16(1), 41-50.

Hu, X., Cereone, N., 1995a. Learning in relational databases: A rough set approach. Computational Intelligence. 11 (2), 323~337.

Hu, X., 1995b. Knowledge discovery in databases: An attribute-oriented rough set approach, PhD thesis, Regina university.

Janusz, A., Starzyk, J., Nelson, D.E., Sturtz, K., 2000. A Mathematical Foundation for Improved Reduct Generation in Information Systems. Knowledge and Information Systems. 2, 131-146.

Jensen, R., Shen, Q., 2003. Finding Rough Set Reducts with Ant Colony Optimization. Proceedings of the 2003 UK Workshop on Computational Intelligence, pp 15-22.

Kennedy, J., Eberhart, R.C., 1995a. Particle Swarm Optimization. Proc IEEE Int. Conf. On Neural Networks, Perth, pp. 1942-1948.

Kennedy, J., Eberhart, R.C., 1995b. A new optimizer using particle swarm theory. Sixth International Symposium on Micro Machine and Human Science. Nagoya, pp. 39-43.

Kennedy, J., 1997. The particle swarm: social adaptation of knowledge. IEEE International Conference on Evolutionary Computation, April 13-16, pp. 303 – 308.

Kennedy, J., Spears, W.M., 1998. Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator. Proceedings of the IEEE Int'l Conference on Evolutionary Computation. pp. 39-43.

Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A., 1999. Rough Sets: A Tutorial. In: S.K. Pal, A. Skowron (eds.), Rough Fuzzy Hybridization. A New Trend in Decision-Making. Springer-Verlag Singapore Pte., Ltd., Singapore, pp. 3-98.

Kudo, M., Sklansky, J., 2000. Comparison of algorithms that select features for pattern classifiers. Pattern Recognition, 33 (1), 25–41.

Liu, H., Motoda, H. 1998. Feature selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers.

Nguyen, H.S., 1996. Some efficient algorithms for rough set methods. In: Proceedings of the Sixth International Conference, Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96) **2**, July 1-5, 1996, Granada, Spain pp. 1451-1456.

Pawlak, Z., 1982. Rough Sets. Int. Journal of Computer and Information Sciences 11(5), 341-356.

Pawlak, Z., 1991. Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishing, Dordrecht.

Pawlak, Z., 1997. Rough set approach to knowledge-based decision support. European Journal of Operational Research 99, 48-57

Swiniarski, R.W., Skowron, A., 2003. Rough set methods in feature selection and recognition. Pattern Recognition Letters. 24(6), 833–849.

Shi, Y., Eberhart, R.C., 1998a. A modified particle swarm optimizer. Proc. IEEE Int. Conf. On Evolutionary Computation. Anchorage, AK, USA, pp. 69-73.

Shi, Y., Eberhart, R. C. 1998b. Parameter selection in particle swarm optimization. In Evolutionary Programming VII: Proc. EP98, New York: Springer-Verlag, pp. 591-600.

Shi, Y., Eberhart, R. C., 1999. Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ: IEEE Service Center, pp.1945–1950.

Skowron, A., Rauszer, C., 1992. The discernibility matrices and functions in information systems. In: Slowinski, R. (Eds.): Intelligent Decision Support--Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic Publishers, Dordrecht, pp. 311-362.

Skowron, A., Bazan, J., Son, N.H., Wroblewski, J., et al. RSES 2.2 User's Guide. http://logic.mimuw.edu.pl/~rses. Institute of Mathematics, Warsaw University, Warsaw, Poland. January 19, 2005.

Slezak, D., Wroblewski, J., 2003. Order based genetic algorithms for the search of approximate entropy reducts. In: Wang, G.Y. et al. (eds.): RSFDGrC. LNAI, Vol. 2693. Chongqing, China (2003), pp. 08-311.

Starzyk, J. Nelson, D.E., Sturtz, K., 1998. Reduct generation in information systems. Bulletin of international rough set society. 3, 19-22

Stefanowski, J., 1998. On rough set based approaches to induction of decision rules. In: Skowron, A., Polkowski, L. (eds.): Rough Sets in Knowledge Discovery, Vol. 1. Physica Verlag, Heidelberg, pp. 500-529.

Vafaie, H., Imam, I.F., 1994. Feature selection methods: genetic algorithms vs. greedy-like search. In: Proceedings of International Conference on Fuzzy and Intelligent Control Systems.

Wang, G.Y., Yu, H., 2002. Decision Table Reduction based on Conditional Information Entropy. Chinese Journal of Computer, 25 (7), 759-766.

Wang, G.Y. Zhao, J., 2004. Theoretical Study on Attribute Reduction of Rough Set Theory: Comparison of Algebra and Information Views. Proceedings of the Third IEEE International Conference on Cognitive Informatics.

Wroblewski, J., 1995. Finding minimal reducts using genetic algorithms. Proc. of the Second Annual Join Conference on Information Sciences, Wrightsville Beach, NC. September 28-October 1, pp.186-189.

Wroblewski, J., 1996. Theoretical Foundations of Order-Based Genetic Algorithms. Fundamenta Informaticae. IOS Press, 28(3-4), 423-430.

Zhai, L.Y., et al., 2002. Feature extraction using rough set theory and genetic algorithms—an application for the simplification of product quality evaluation. Computers & Industrial Engineering. 43, 661-676.

Appendix A (Experimental Examples)

Example 2

The process of the particle swarms searching for optimal solutions for dataset Vote is given in Table 7 and Figure 4.

Table 7 PSO searching process on Vote

Iter	Best Solution	Fitness Value	Feature Subset Length
1-45	1,2,3,4,7,9,11,13,16	0.9437	9
46-52	1,2,3,4,7,11,13,16	0.9440	8
53	1,2,3,4,11,13,16	0.9443	7
54	1,2,3,4,9,11,13,16	0.9500	8

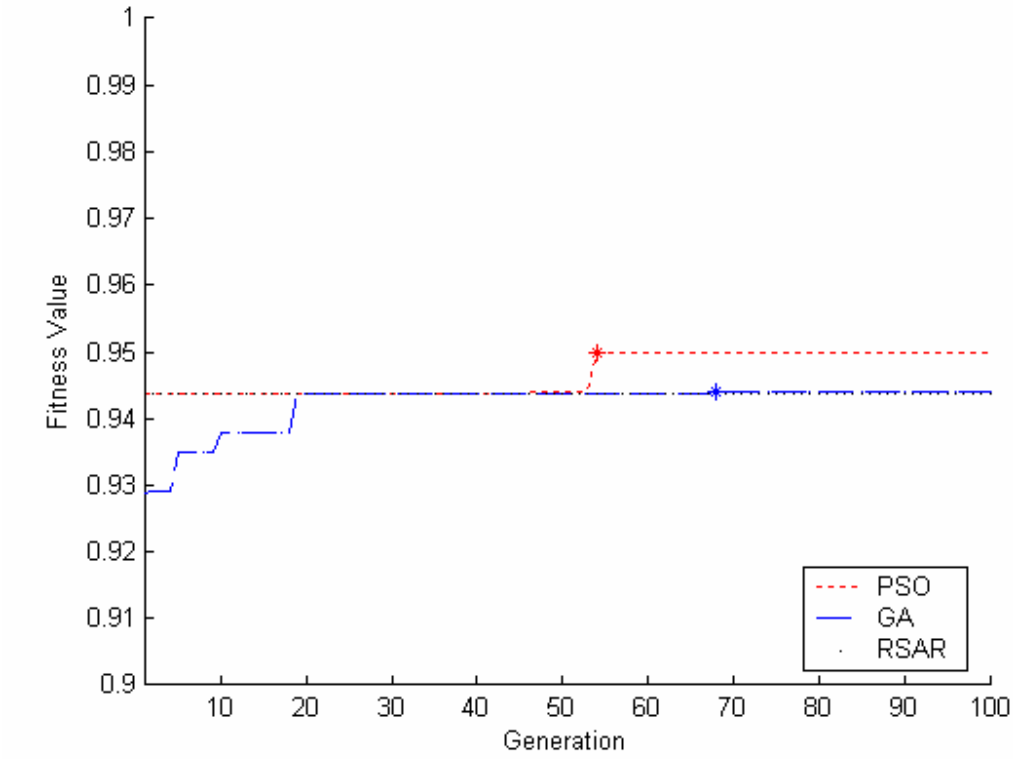

Fig.4 Evolution process of the global best on dataset Vote.
(PSORSFS, GAAR, POSAR)

Example 3

The process of the particle swarms searching for optimal solutions for dataset Mushroom is given in Table 8 and Figure 5.

Table 8 PSO searching process on Mushroom

Iter	Best Solution	Fitness Value	Feature Subset Length
1	3,5,6,9,11,12,14,18,22	0.9591	9
2	3,5,6,9,11,18,21,22	0.9636	8
3	3,5,6,11,18,21,22	0.9682	7
4--15	3,5,13,21,22	0.9773	5
16	3,5,11,22	0.9818	4
…	…	…	…

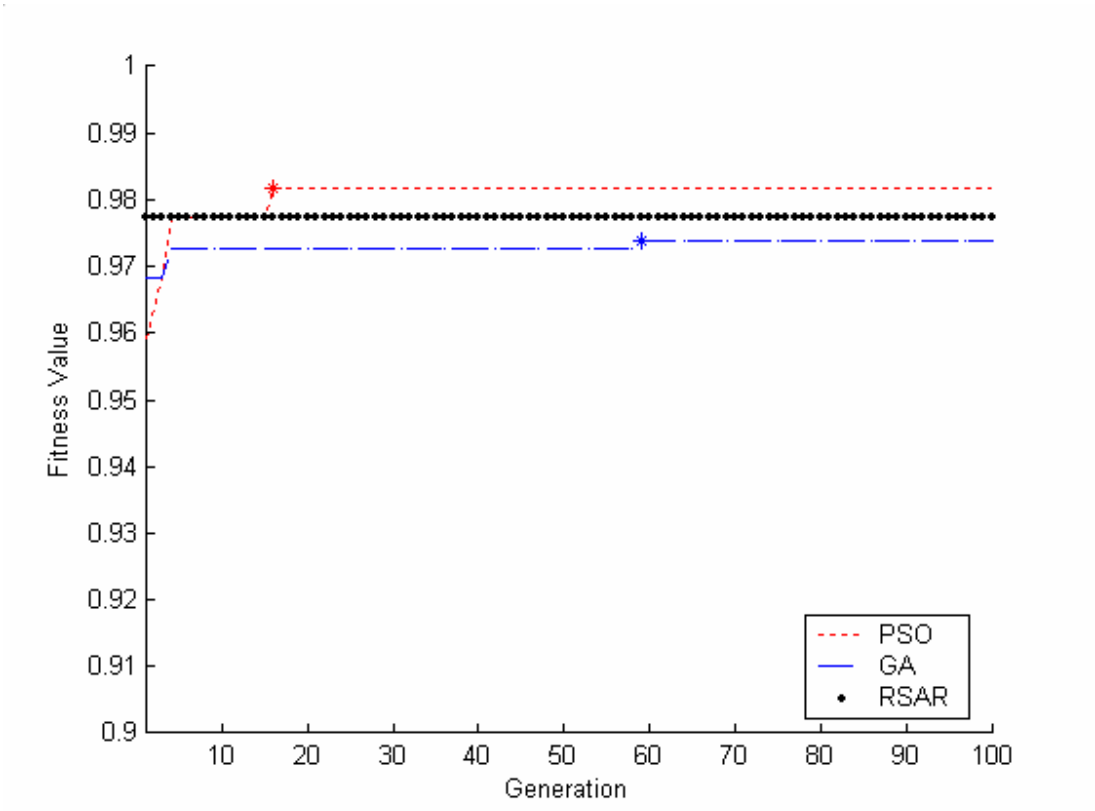

Fig.5 Evolution process of the global best on dataset Mushroom. (PSORSFS, GAAR, POSAR)

Example 4

The process of the particle swarms searching for optimal solutions for dataset Soybean-small is given in Table 9 and Figure 6.

Table 9 PSO searching process on Soybean-small

Iter	Best Solution	Fitness Value	Feature Subset Length
1	5,11,13,18,21,22,23,28,29,32,33,35	0.9657	12
2	5,11,13,18,21,22,23,29,32,33,35	0.9686	11
3--5	1,5,9,11,13,19,22,23,33,35	0.9714	10
6	1,9,13,18,19,22,23,33,35	0.9743	9
7	1,9,13,19,22,23,33,35	0.9771	8
8--15	3,9,13,19,22,23,33	0.9800	7
16--19	3,13,19,22,23,33	0.9829	6
20--21	13,15,22,23,33	0.9857	5
22--27	13, 22,23,33	0.9886	4
28--32	22,23,33	0.9914	3
33	22,23	0.9943	2
…	…	…	…

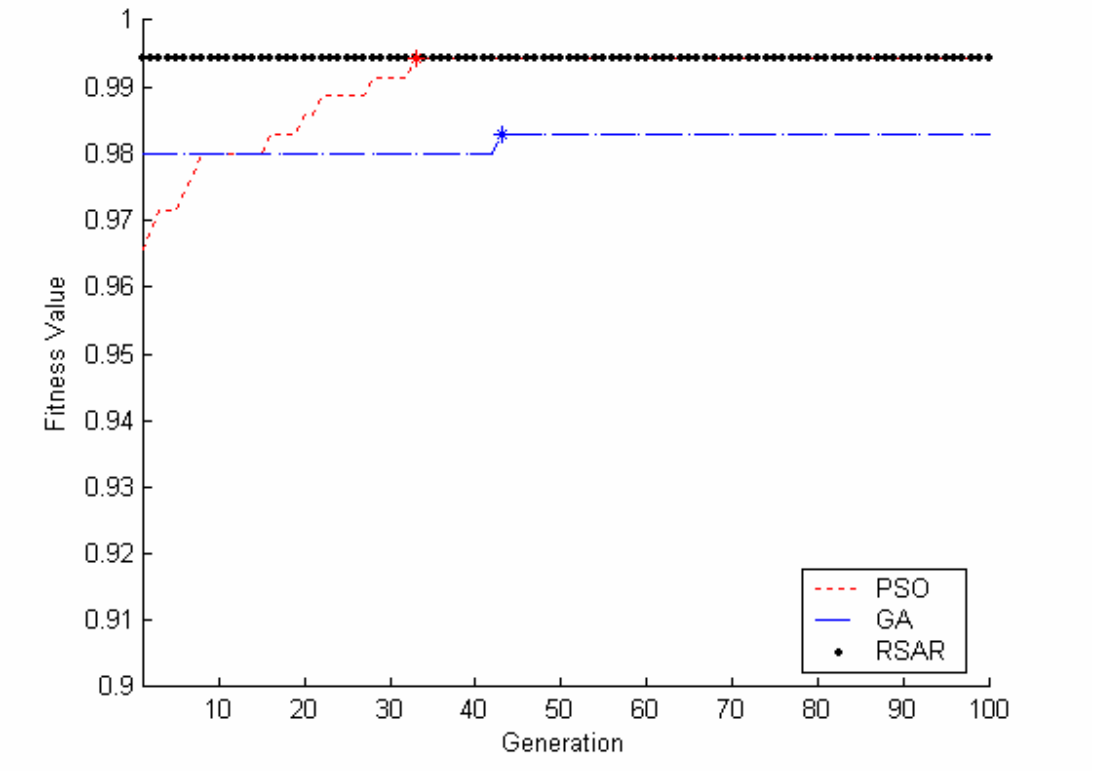

Fig.6 Evolution process of the global best on dataset Soybean-small (PSORSFS, GAAR, POSAR)

Example 5

The process of the particle swarms searching for optimal solutions for dataset Lung is given in Table 10 and Figure 7.

Table 10 PSO searching process on Lung

Iter	Best Solution	Fitness Value	Feature Subset Length
1	2,9,14,15,24,25,30,31,32,36,40,42,43,44,51,54,55	0.9696	17
2--5	2,9,14,15,25,30,31,32,36,40,42,43,44,51,54,55	0.9714	16
6	2,9,15,25,30,31,32,40,42,43,44,48,51,54,55	0.9732	15
7--8	2,9,15,25,30,32,40,42,43,44,48,51,54,55	0.9750	14
9	2,9,15,25,29,30,40,42,43,44,48,51,55	0.9768	13
10--21	9,15,25,30,33,35, 40,42,43,44, 51,55	0.9786	12
22--23	9,15,25,30, 33,35, 40,42,43,44,55	0.9804	11
24--25	9,15,25,30,33, 40,42,43,44,55	0.9821	10
26	9,15,25,30,33, 40,42, 44,55	0.9839	9
27--31	9,15,25,30,33, 42, 44,55	0.9857	8
32	9,25,30,33, 42, 55	0.9893	6
33--39	9,25,30,33,55	0.9911	5
40	9,30,33,55	0.9929	4

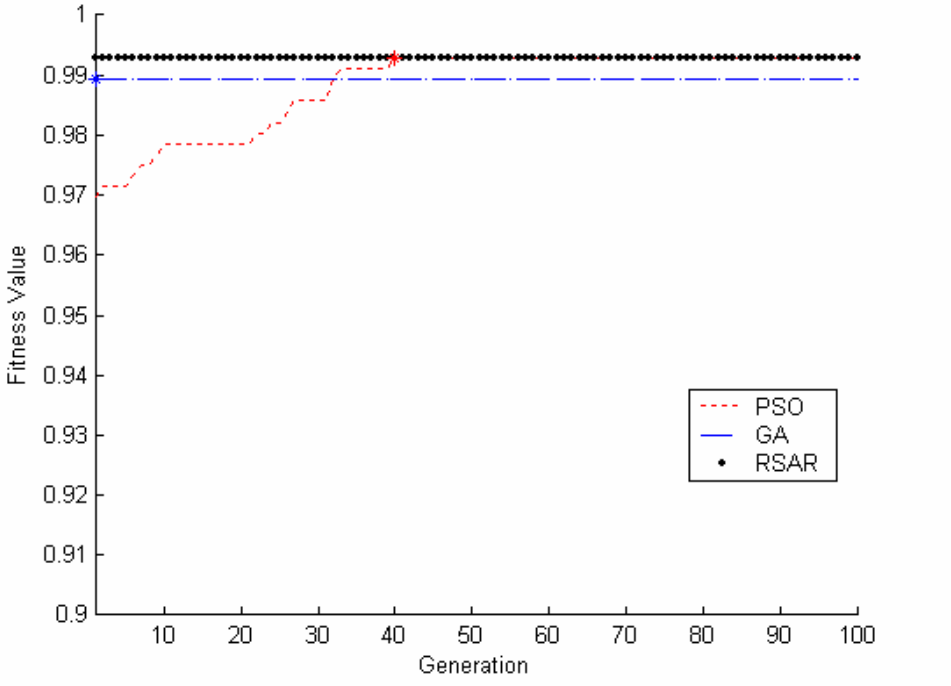

Fig.7 Evolution process of the global best on dataset Lung (PSORSFS, GAAR, POSAR)

Example 6

The process of the particle swarms searching for optimal solutions for dataset DNA is given in Table 11 and Figure 8.

Table 11 PSO searching process on DNA

Iter	Best Solution	Fitness Value	Feature Subset Length
1	2,3,9,10,12,16,21,25,27,30,31,36,40,4247,50,52,56	0.9684	18
2	2,3,9,10,12,25,27,29,30,31,36,40,4247,50,52,56	0.9702	17
3	2,3,9,10,12,25,29,31,36,40,42,52,54,56	0.9737	15
4--7	2,9,10,12, 29,31,36, 40,42, 52, 54,56,57	0.9772	13
8--11	2,9,12, 29,31,34,36, 42, 52, 54,56,57	0.9789	12
12--15	2,9,12, 29,31,34,36, 42, 52,56,57	0.9807	11
16--19	9,12, 29,31,34,36, 42, 52,56,57	0.9825	10
20--23	9,12, 29,31,36, 42, 52,56,57	0.9842	9
24--27	9,12, 29,31, 36, 42, 52,57	0.9860	8
28--31	9,12, 29,31, 36, 42,57	0.9877	7
32	9,12, 29,31, 36, 42	0.9895	6

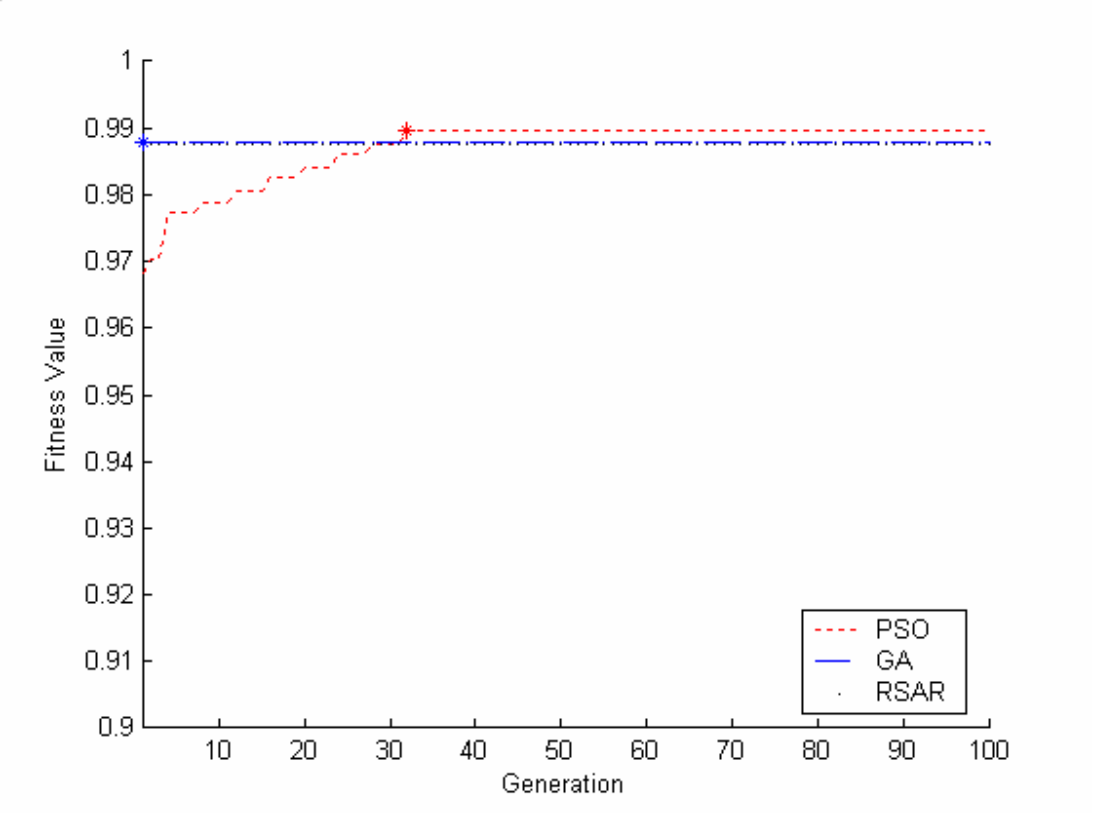

Fig.8 Evolution process of the global best on dataset DNA (PSORSFS, GAAR, POSAR)