# Nature Inspired Multi-Swarm Heuristics for Multi-Knowledge Extraction

Hongbo Liu[1,3], Ajith Abraham[2], and Benxian Yue[3]

[1] School of Information Science and Technology, Dalian Maritime University, 116026 Dalian, China
lhb@dlut.edu.cn
http://hongboliu.torrent.googlepages.com
[2] Centre for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology, Trondheim, Norway
ajith.abraham@ieee.org
http://www.softcomputing.net
[3] School of Electronic and Information Engineering, Dalian University of Technology, Dalian 116023, China

**Abstract.** Multi-knowledge extraction is significant for many real-world applications. The nature inspired population-based reduction approaches are attractive to find multiple reducts in the decision systems, which could be applied to generate multi-knowledge and to improve decision accuracy. In this Chapter, we introduce two nature inspired population-based computational optimization techniques namely Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) for rough set reduction and multi-knowledge extraction. A Multi-Swarm Synergetic Optimization (MSSO) algorithm is presented for rough set reduction and multi-knowledge extraction. In the MSSO approach, different individuals encodes different reducts. The proposed approach discovers the best feature combinations in an efficient way to observe the change of positive region as the particles proceed throughout the search space. We also attempt to theoretically prove that the multi-swarm synergetic optimization algorithm converges with a probability of 1 towards the global optimal. The performance of the proposed approach is evaluated and compared with Standard Particle Swarm Optimization (SPSO) and Genetic Algorithms (GA). Empirical results illustrate that the approach can be applied for multiple reduct problems and multi-knowledge extraction very effectively.

## 1 Introduction

Rough set theory [1,2,3] provides a mathematical tool that can be used for both feature selection and knowledge discovery. It helps us to find out the minimal attribute sets called '*reducts*' to classify objects without deterioration of classification quality and induce minimal length decision rules inherent in a given information system. The idea of reducts has encouraged many researchers in studying the effectiveness of rough set theory in a number of real world domains, including medicine, pharmacology, control systems, fault-diagnosis, text

J. Koronacki et al. (Eds.): Advances in Machine Learning II, SCI 263, pp. 445–466.
springerlink.com 

categorization, social sciences, switching circuits, economic/financial prediction, image processing, and so on [4,5,6,7,8,9,10].

The reduct of an information system is not unique. There may be many subsets of attributes, which preserve the equivalence class structure (i.e., the knowledge) expressed in the information system. Although several variants of reduct algorithms are reported in the literature, at the moment, there is no accredited best heuristic reduct algorithm. So far, it is still an open research area in rough sets theory.

Particle swarm algorithm is inspired by social behavior patterns of organisms that live and interact within large groups. In particular, it incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior, from which the Swarm Intelligence (SI) paradigm has emerged [11]. The swarm intelligent model helps to find optimal regions of complex search spaces through interaction of individuals in a population of particles [12,13,14]. As an algorithm, its main strength is its fast convergence, which compares favorably with many other global optimization algorithms [15,16]. It has exhibited good performance across a wide range of applications [17,18,19,20,21]. The particle swarm algorithm is particularly attractive for feature selection as there seems to be no heuristic that can guide search to the optimal minimal feature subset. Additionally, it can be the case that particles discover the best feature combinations as they proceed throughout the search space.

The main focus of this Chapter is to investigate Multi-Swarm Synergetic Optimization (MSSO) algorithm and its application in finding multiple reducts for difficult problems. The rest of the Chapter is organized as follows. Some related terms and theorems on rough set theory are explained briefly in Section 3. Particle swarm model is presented and the effects on the change of the neighborhoods of particles are analyzed in Section 4. The proposed approach based on particle swarm algorithm is presented in Section 5. In this Section, we describe the MSSO model in detail and theoretically prove the properties related to the convergence of the proposed algorithm. Experiment settings, results and discussions are given in Section 6 and finally conclusions are given in Section 7.

## 2 Related Research Works

Usually real world objects are the corresponding tuple in some decision tables. They store a huge quantity of data, which is hard to manage from a computational point of view. Finding reducts in a large information system is still an NP-hard problem [22]. The high complexity of this problem has motivated investigators to apply various approximation techniques to find near-optimal solutions. Many approaches have been proposed for finding reducts, e.g., discernibility matrices, dynamic reducts, and others [23,24]. The heuristic algorithm is a better choice. Hu et al. [25] proposed a heuristic algorithm using discernibility matrix. The approach provided a weighting mechanism to rank attributes. Zhong and Dong [26] presented a wrapper approach using rough sets theory with greedy

heuristics for feature subset selection. The aim of feature subset selection is to find out a minimum set of relevant attributes that describe the dataset as well as the original all attributes do. So finding reduct is similar to feature selection. Zhong's algorithm employed the number of consistent instances as heuristics. Banerjee et al. [27] presented various attempts of using Genetic Algorithms in order to obtain reducts.

Conventional approaches for knowledge discovery always try to find a good reduct or to select a set of features [28]. In the knowledge discovery applications, only the good reduct can be applied to represent knowledge, which is called a single body of knowledge. In fact, many information systems in the real world have multiple reducts, and each reduct can be applied to generate a single body of knowledge. Therefore, multi-knowledge based on multiple reducts has the potential to improve knowledge representation and decision accuracy [29]. However, it would be exceedingly time-consuming to find multiple reducts in an instance information system with larger numbers of attributes and instances. In most of strategies, different reducts are obtained by changing the order of condition attributes and calculating the significance of different condition attribute combinations against decision attribute(s). It is a complex multi-restart processing about condition attribute increasing or decreasing in quantity. Population-based search approaches are of great benefits in the multiple reduction problems, because different individual trends to be encoded to different reduct. So it is attractive to find multiple reducts in the decision systems.

## 3 Rough Set Reduction

The basic concepts of rough set theory and its philosophy are presented and illustrated with examples in [1,2,3,26,28,30,31]. Here, we illustrate only the relevant basic ideas of rough sets that are relevant to the present work.

In rough set theory, an information system is denoted in 4-tuple by $S = (U, A, V, f)$, where $U$ is the universe of discourse, a non-empty finite set of $N$ objects $\{x_1, x_2, \cdots, x_N\}$. $A$ is a non-empty finite set of attributes such that $a : U \rightarrow V_a$ for every $a \in A$ ($V_a$ is the value set of the attribute $a$).

$$V = \bigcup_{a \in A} V_a$$

$f : U \times A \rightarrow V$ is the total decision function (also called the information function) such that $f(x, a) \in V_a$ for every $a \in A$, $x \in U$. The information system can also be defined as a decision table by $S = (U, C, D, V, f)$. For the decision table, $C$ and $D$ are two subsets of attributes. $A = \{C \cup D\}$, $C \cap D = \emptyset$, where $C$ is the set of input features and $D$ is the set of class indices. They are also called condition and decision attributes, respectively.

Let $a \in C \cup D$, $P \subseteq C \cup D$. A binary relation $IND(P)$, called an equivalence (indiscernibility) relation, is defined as follows:

$$IND(P) = \{(x, y) \in U \times U \quad | \quad \forall a \in P, f(x, a) = f(y, a)\} \tag{1}$$

The equivalence relation $IND(P)$ partitions the set $U$ into disjoint subsets. Let $U/IND(P)$ denote the family of all equivalence classes of the relation $IND(P)$. For simplicity of notation, $U/P$ will be written instead of $U/IND(P)$. Such a partition of the universe is denoted by $U/P = \{P_1, P_2, \cdots, P_i, \cdots\}$, where $P_i$ is an equivalence class of $P$, which is denoted $[x_i]_P$. Equivalence classes $U/C$ and $U/D$ will be called condition and decision classes, respectively.

*Lower Approximation*: Given a decision table $T = (U, C, D, V, f)$. Let $R \subseteq C \cup D$, $X \subseteq U$ and $U/R = \{R_1, R_2, \cdots, R_i, \cdots\}$. The $R$-lower approximation set of $X$ is the set of all elements of $U$ which can be with certainty classified as elements of $X$, assuming knowledge $R$. It can be presented formally as

$$APR_R^-(X) = \bigcup \{R_i \quad | \quad R_i \in U/R, R_i \subseteq X\} \tag{2}$$

*Positive Region*: Given a decision table $T = (U, C, D, V, f)$. Let $B \subseteq C$, $U/D = \{D_1, D_2, \cdots, D_i, \cdots\}$ and $U/B = \{B_1, B_2, \cdots, B_i, \cdots\}$. The $B$-positive region of $D$ is the set of all objects from the universe $U$ which can be classified with certainty to classes of $U/D$ employing features from $B$, i.e.,

$$POS_B(D) = \bigcup_{D_i \in U/D} APR_B^-(D_i) \tag{3}$$

*Positive Region*: Given a decision table $T = (U, C, D, V, f)$. Let $B \subseteq C$, $U/D = \{D_1, D_2, \cdots, D_i, \cdots\}$ and $U/B = \{B_1, B_2, \cdots, B_i, \cdots\}$. The $B$-positive region of $D$ is the set of all objects from the universe $U$ which can be classified with certainty to classes of $U/D$ employing features from $B$, i.e.,

$$POS_B(D) = \bigcup_{D_i \in U/D} B_-(D_i) \tag{4}$$

*Reduct*: Given a decision table $T = (U, C, D, V, f)$. The attribute $a \in B \subseteq C$ is $D - dispensable$ in $B$, if $POS_B(D) = POS_{(B-\{a\})}(D)$; otherwise the attribute $a$ is $D - indispensable$ in $B$. If all attributes $a \in B$ are $D - indispensable$ in $B$, then $B$ will be called $D - independent$. A subset of attributes $B \subseteq C$ is a $D - reduct$ of $C$, iff $POS_B(D) = POS_C(D)$ and $B$ is $D - independent$. It means that a reduct is the minimal subset of attributes that enables the same classification of elements of the universe as the whole set of attributes. In other words, attributes that do not belong to a reduct are superfluous with regard to classification of elements of the universe. Usually, there are many reducts in an instance information system. Let $2^{|A|}$ represent all possible attribute subsets $\{\{a_1\}, \cdots, \{a_{|A|}\}, \{a_1, a_2\}, \cdots, \{a_1, \cdots, a_{|A|}\}\}$. Let $RED$ represent the set of reducts, i.e.,

$$RED = \{B \quad | \quad POS_B(D) = POS_C(D), POS_{(B-\{a\})}(D) < POS_B(D)\} \tag{5}$$

*Multi-knowledge*: Given a decision table $T = (U, C, D, V, f)$. Let $RED$ represent the set of reducts, Let $\varphi$ is a mapping from the condition space to the decision space. Then multi-knowledge can be defined as follows:

$$\Psi = \{\varphi_B \quad | \quad B \in RED\} \tag{6}$$

*Reduced Positive Universe* and *Reduced Positive Region*: Given a decision table $T = (U, C, D, V, f)$. Let $U/C = \{[u_1^{'}]_C, [u_2^{'}]_C, \cdots, [u_m^{'}]_C\}$, Reduced Positive Universe $U^{'}$ can be written as:

$$U^{'} = \{u_1^{'}, u_2^{'}, \cdots, u_m^{'}\}. \tag{7}$$

and

$$POS_C(D) = [u_{i_1}^{'}]_C \cup [u_{i_2}^{'}]_C \cup \cdots \cup [u_{i_t}^{'}]_C. \tag{8}$$

Where $\forall u_{i_s}^{'} \in U^{'}$ and $|[u_{i_s}^{'}]_C/D| = 1 (s = 1, 2, \cdots, t)$. Reduced positive universe can be written as:

$$U_{pos}^{'} = \{u_{i_1}^{'}, u_{i_2}^{'}, \cdots, u_{i_t}^{'}\}. \tag{9}$$

and $\forall B \subseteq C$, reduced positive region

$$POS_B^{'}(D) = \bigcup_{X \in U^{'}/B \wedge X \subseteq U_{pos}^{'} \wedge |X/D|=1} X \tag{10}$$

where $|X/D|$ represents the cardinality of the set $X/D$. $\forall B \subseteq C$, $POS_B(D) = POS_C(D)$ if $POS_B^{'} = U_{pos}^{'}$ [31]. It is to be noted that $U^{'}$ is the reduced universe, which usually would reduce significantly the scale of datasets. It provides a more efficient method to observe the change of positive region when we search the reducts. We didn't have to calculate $U/C$, $U/D$, $U/B$, $POS_C(D)$, $POS_B(D)$ and then compare $POS_B(D)$ with $POS_C(D)$ to determine whether they are equal to each other or not. We only calculate $U/C$, $U^{'}$, $U_{pos}^{'}$, $POS_B^{'}$ and then compare $POS_B^{'}$ with $U_{pos}^{'}$.

## 4 Particle Swarm Optimization Algorithm

The classical particle swarm model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the $d$-dimension problem space to search the new solutions, where the fitness $f$ can be measured by calculating the number of condition attributes in the potential reduction solution. Each particle has a position represented by a position-vector $\boldsymbol{p}_i$ ($i$ is the index of the particle), and a velocity represented by a velocity-vector $\boldsymbol{v}_i$. Each particle remembers its own best position so far in a vector $\boldsymbol{p}_i^{\#}$, and its $j$-th dimensional value is $p_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\boldsymbol{p}^*$, and its $j$-th dimensional value is $p_j^*$. When the particle moves in a state space restricted to zero and one on each dimension, the change of probability with time steps is defined as follows:

$$P(p_{ij}(t) = 1) = f(p_{ij}(t-1), v_{ij}(t-1), p_{ij}^{\#}(t-1), p_j^*(t-1)). \tag{11}$$

where the probability function is

$$sig(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}}. \tag{12}$$

At each time step, each particle updates its velocity and moves to a new position according to Eqs.(13) and (14):

$$v_{ij}(t) = wv_{ij}(t-1) + c_1 r_1(p_{ij}^{\#}(t-1) - p_{ij}(t-1)) + c_2 r_2(p_j^{*}(t-1) - p_{ij}(t-1)) \tag{13}$$

$$p_{ij}(t) = \begin{cases} 1 & \text{if } \rho < sig(v_{ij}(t)); \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Where $c_1$ is a positive constant, called as coefficient of the self-recognition component, $c_2$ is a positive constant, called as coefficient of the social component. $r_1$ and $r_2$ are the random numbers in the interval [0,1]. The variable $w$ is called as the inertia factor, which value is typically setup to vary linearly from 1 to near 0 during the iterated processing. $\rho$ is random number in the closed interval [0,1]. From Eq. (13), a particle decides where to move next, considering its current state, its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm. Figure 1 illustrates how the position is reacted on by its velocity. The pseudo-code for particle swarm optimization algorithm is illustrated in Algorithm 1.

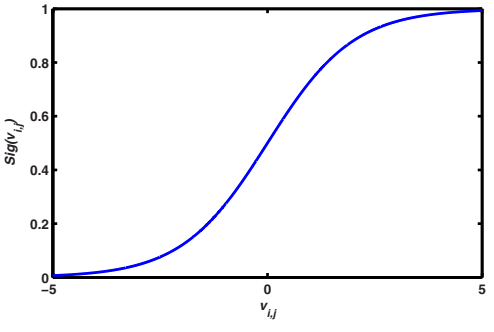


**Fig. 1.** Sigmoid function for PSO

The particle swarm algorithm can be described generally as a population of vectors whose trajectories oscillate around a region which is defined by each individual's previous best success and the success of some other particle. Some previous studies have discussed the trajectory of particles and its convergence [14,12,32,33]. Bergh and Engelbrecht [33] overviewed the theoretical studies, and extended these studies to investigate particle trajectories for general swarms to include the influence of the inertia term. They also provided a formal proof that each particle converges to a stable point. It has been shown that the trajectories of the particles oscillate as different sinusoidal waves and converge quickly, sometimes prematurely. Various methods have been used to identify some other particle to influence the individual.

**Algorithm 1.** Particle Swarm Optimization Algorithm

01. Initialize the size of the particle swarm $n$, and other parameters.
02. Initialize the positions and the velocities for all the particles randomly.
03. While (the end criterion is not met) do
04. $t = 1$;
05. Calculate the fitness value of each particle;
06. $\boldsymbol{p}^* = argmin_{i=1}^{n}(f(\boldsymbol{p}^*(t-1)), f(\boldsymbol{p}_1(t)), f(\boldsymbol{p}_2(t)), \cdots, f(\boldsymbol{p}_i(t)), \cdots, f(\boldsymbol{p}_n(t)))$;
07. For $i$= 1 to $n$
08. $\boldsymbol{p}_i^{\#}(t) = argmin_{i=1}^{n}(f(\boldsymbol{p}_i^{\#}(t-1)), f(\boldsymbol{p}_i(t))$;
09. For $j = 1$ to $d$
10. Update the $j$-th dimension value of $\boldsymbol{p}_i$ and $\boldsymbol{v}_i$
10. according to Eqs.(13),(14),(12);
12. Next $j$
13. Next $i$
13. $t++$
14. End While.

Eberhart and Kennedy called the two basic methods as "$g$best model" and "$l$best model" [11]. In the $g$best model, the trajectory for each particle's search is influenced by the best point found by any member of the entire population. The best particle acts as an attractor, pulling all the particles towards it. Eventually all particles will converge to this position. In the $l$best model, particles have information only of their own and their nearest array neighbors' best ($l$best), rather than that of the whole swarm. Namely, in Eq. (13), $g$best is replaced by $l$best in the model. The $l$best model allows each individual to be influenced by some smaller number of adjacent members of the population array. The particles selected to be in one subset of the swarm have no direct relationship to the other particles in the other neighborhood. Typically $l$best neighborhoods comprise exactly two neighbors. When the number of neighbors increases to all but itself in the $l$best model, the case is equivalent to the $g$best model. Unfortunately there is a large computational cost to explore the neighborhood relation in each iteration when the number of neighbors is too little. Some previous studies has been shown that $g$best model converges quickly on problem solutions but has a weakness for becoming trapped in local optima, while $l$best model converges slowly on problem solutions but is able to "flow around" local optima, as the individuals explore different regions [36]. Some related research and development during the recent years are also reported in [21,34,35,37].

## 5 Rough Set Reduction Algorithm Based on Swarms

Blackwell and Branke [38] investigated a multi-swarm optimization specifically designed to work well in dynamic environments. The main idea is to split the population of particles into a set of interacting swarms. These swarms interact locally by an exclusion parameter and globally through a new anti-convergence operator. The results illustrated that the multiswarm optimizer significantly outperformed

the other considered approaches. Niu et al. [39] presented a multi-swarm cooperative particle swarm optimizer, inspired by the phenomenon of symbiosis in natural ecosystems. The approach is based on a master - slave model, in which a population consists of one master swarm and several slave swarms. The slave swarms execute a single PSO or its variants independently to maintain the diversity of particles, while the master swarm evolves based on its own knowledge and also the knowledge of the slave swarms. In the simulation studies, several benchmark functions are performed, and the performances of their algorithms are compared with the standard PSO (SPSO) to demonstrate the superiority. The multi swarm approaches let several swarms of particles cooperate to find good solutions. Usually they have to be designed for specific problems. In this Section, we design a multi-swarm synergetic optimization algorithm for rough set reduction and multi-knowledge extraction. The sub-swarms are encoded with different reducts, which is suitable for searching multiple reducts in decision systems.

### 5.1 Coding and Evaluation

Given a decision table $T = (U, C, D, V, f)$, the set of condition attributes, $C$, consists of $m$ attributes. We set up a search space of $m$ dimensions for the reduction problem. Accordingly, each particle's position is represented as a binary bit string of length $m$. Each dimension of the particle's position maps one condition attribute. The domain for each dimension is limited to 0 or 1. The value '1' means the corresponding attribute is selected while '0' not selected. Each position can be "decoded" to a potential reduction solution, a subset of $C$. The particle's position is a series of priority levels of the attributes. The sequence of the attribute will not be changed during the iteration. But after updating the velocity and position of the particles, the particle's position may appear real values such as 0.4, etc. It is meaningless for the reduction. Therefore, we introduce a discrete particle swarm optimization technique for this reduction problem. The particles updates its velocity according to Eq. (13), considering its current state, its own experience, and the experience of its successful particle in its neighborhood swarm. Each dimension of the particles' position would be explored between 0 and 1 through Eqs. (12) and (14).

During the search procedure, each individual is evaluated using the fitness. According to the definition of rough set reduct, the reduction solution must ensure the decision ability is the same as the primary decision table and the number of attributes in the feasible solution is kept as low as possible. In our algorithm, we first evaluate whether the potential reduction solution satisfies $POS_E^{'} = U_{pos}^{'}$ or not ($E$ is the subset of attributes represented by the potential reduction solution). If it is a feasible solution, we calculate the number of '1' in it. The solution with the lowest number of '1' would be selected. For the particle swarm, the lower number of '1' in its position, the better the fitness of the individual is. $POS_E^{'} = U_{pos}^{'}$ is used as the criterion of the solution validity.

In the proposed encoding representations, we consider particle's position encoding as the binary representation of an integer. The step size is equal to 1, that is, the dimension of the search space is then 1. In practice, when the binary string

is too long for a large scale attribute reduction problem, it is difficult to use it as an integer. It is time-consuming for each iteration. So we split it into a small number (say $H$) of shorter binary strings, each one is seen as an integer. Then the dimension of the problem is not anymore 1, but $H$. The swarm algorithm with two strategies is called as Bi-metrics Binary PSO (Figures 2 and 3).

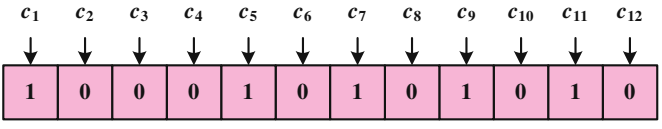


**Fig. 2.** Bi-metrics Binary Representation 1

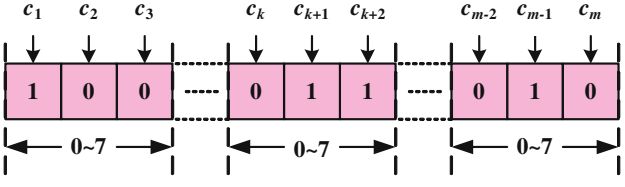


**Fig. 3.** Bi-metrics Binary Representation 2

### 5.2 Multi-Swarm Synergetic Model

To employ a multi-swarm, the solution vector is split amongst the different populations according to some rule in such a way that the simplest of the schemes does not allow any overlap between the spaces covered by different populations. To find a solution to the original problem, representatives from all the populations are combined to form the potential solution vector, which, in turn, is passed on the error function. This adds a new dimension to the survival game: cooperation between different populations [33,40].

As mentioned above, one of the most important applications is to solve multiple reduct problems and multi-knowledge extraction. Those reducts usually share some common classification characteristics in the information systems. They are apt to cluster into different groups. Sometime they are also the members of several groups at the same time [41]. To match the classification characteristics, we introduce a multi-swarm search algorithm for them. In the algorithm, all particles are clustered spontaneously into different sub-swarms of the whole swarm. Every particle can connect to more than one sub-swarm, and a crossover neighborhood topology is constructed between different sub-swarms. The particles in the same sub-swarm would carry some similar functions as possible and search their optimal. Each sub-swarm would approach its appropriate position (solution), which would be helpful for the whole swarm to keep in a good balance state. Figure 4 illustrates a multi-swarm topology. In the swarm system, a swarm

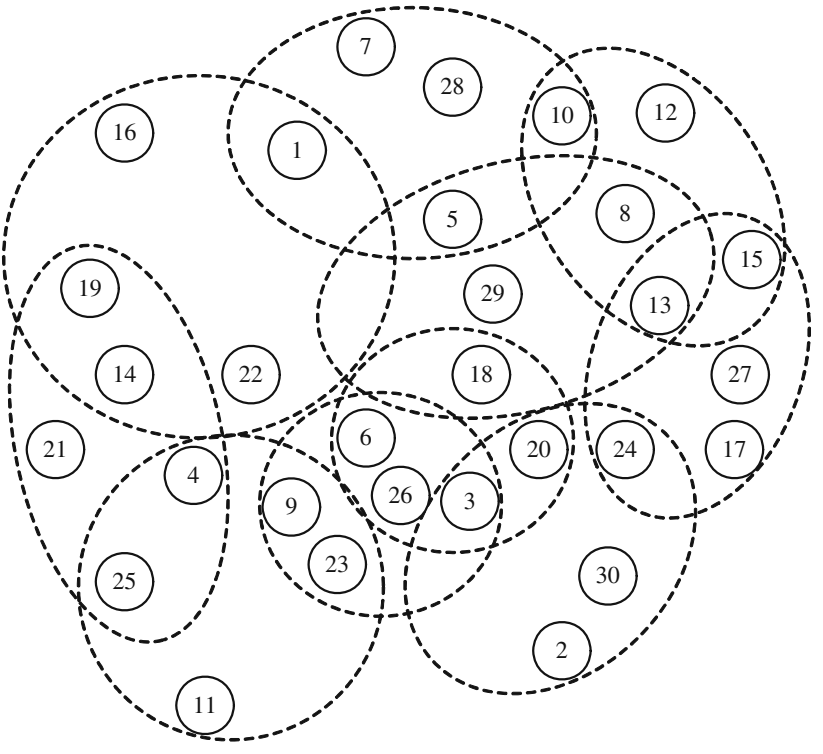


**Fig. 4.** A multi-swarm topology

with 30 particles is organized into 10 sub-swarms, each sub-swarm consisting of 5 particles. Particles 3 and 13 have the maximum membership level, 3. During the iteration process, the particle updates its velocity followed by the location of the best fitness achieved so far by the particle itself and by the location of the best fitness achieved so far across all its neighbors in all sub-swarms it belongs to. The process makes an important influence on the particles' ergodic and synergetic performance. The multi-swarm algorithm for the reduction problem is illustrated as follows:

**Step 1.** Calculate $U^{'}$, $U^{'}_{pos}$ using Eqs. (7) and (9).
**Step 2.** Initialize the size of the particle swarm $n$, and other parameters. Initialize the positions and the velocities for all the particles randomly.
**Step 3.** Multiple sub-swarms $n$ are organized into a crossover neighborhood topology. A particle can join more than one sub-swarm. Each particle has the maximum membership level $l$, and each sub-swarm accommodates default number of particles $m$.
**Step 4.** Decode the positions and evaluate the fitness for each particles, if $POS^{'}_{E} \neq U^{'}_{pos}$, the fitness is punished as the total number of the condition attributes, else the fitness is the number of '1' in the position.

**Step 5.** Find the best particle in the swarm, and find the best one in each sub-swarms. If the "global best" of the swarm is improved, $noimprove = 0$, otherwise, $noimprove = 1$. Update velocity and position for each particle at the iteration $t$.

5.01 For $m = 1$ to $subs$
5.02 $\boldsymbol{p}^* = argmin_{i=1}^{subs_m}(f(\boldsymbol{p}^*(t-1)), f(\boldsymbol{p}_1(t)),$
5.02 $f(\boldsymbol{p}_2(t)), \cdots, f(\boldsymbol{p}_i(t)), \cdots, f(\boldsymbol{p}_{subs_m}(t)));$
5.03 For $ss = 1$ to $subs_m$
5.04 $\boldsymbol{p}_i^{\#}(t) = argmin(f(\boldsymbol{p}_i^{\#}(t-1)), f(\boldsymbol{p}_i(t));$
5.05 For $d = 1$ to $D$
5.06 Update the $d$-th dimension value of $\boldsymbol{p}_i$ and $\boldsymbol{v}_i$
5.06 according to Eqs.(13), (12), and (14);
5.07 Next $d$
5.08 Next $ss$
5.09 Next $m$

**Step 6.** If $noimprove = 1$, goto Step 3, the topology is re-organized. If the end criterion is not met, goto Step 4. Otherwise, provide the best solution (output), the fitness.

### 5.3 Algorithm Analysis

For analyzing the convergence of the multi-swarm algorithm, we first introduce the definitions and lemmas [42,43,44], and then theoretically prove that the algorithm converges with a probability 1 or strongly towards the global optimal.

Xu *et al.* [45] analyzed the search capability of an algebraic crossover through classifying the individual space of genetic algorithms, which is helpful to comprehend the search of genetic algorithms such that premature convergence and deceptive problems [46] could be avoided. In this Subsection, we also attempt to theoretically analyze the performance of the multi-swarm algorithm with crossover neighborhood topology. For the sake of convenience, let crossover operator $|_c$ denote the wheeling-round-the-best-particles process.

Consider the problem $(P)$ as

$$(P) = min\{f(\boldsymbol{x}) : \boldsymbol{x} \in D\} \tag{15}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$, $f(\boldsymbol{x}) : D \to R$ is the objective function and $D$ is a compact Hausdorff space. Applying our algorithm the problem $(P)$ may be transformed to $P'$ as

$$(P') = \begin{cases} minf(\boldsymbol{x}) \\ \boldsymbol{x} \in \Omega = [-s, s]^n \end{cases} \tag{16}$$

where $\Omega$ is the set of feasible solutions of the problem. A swarm is a set, which consists of some feasible solutions of the problem. Assume $S$ as the encoding space of $D$. A neighborhood function is a mapping $\mathcal{N} : \Omega \to 2^{\Omega}$, which defines for each solution $S \in \Omega$ a subset $\mathcal{N}(S)$ of $\Omega$, called a neighborhood. Each solution in $\mathcal{N}(S)$ is a neighbor of $S$. A local search algorithm starts off with an

initial solution and then continually tries to find better solutions by searching neighborhoods [47]. Most generally said, in swarm algorithms the encoding types $S$ of particles in the search space $D$ are often represented as strings of a fixed-length $L$ over an alphabet. Without loss of generality, $S$ can be described as

$$S = \underbrace{z_m \times \cdots \times z_m}_{L} \tag{17}$$

where $z_m$ is a finite field about integer number $\mod m$. Most often, it is the binary alphabet, *i.e.* $m = 2$.

**Proposition 1.** *If $k$ alleles are '0's in the nontrivial ideal $\Omega$,* i.e. *$L - k$ alleles are uncertain, then $\theta_\Omega$ partitions $\Omega$ into $2^k$ disjoint subsets as equivalence classes corresponding to Holland's schema theorem [48,49],* i.e.*, each equivalence class consists of some '1's which $k$ alleles in $\Omega$ with '0' are replaced by '1's. Let $A \in S/\theta_\Omega$, then there is an minimal element $m$ of $A$ under partial order $(S, \vee, \wedge, \neg)$, such that $A = \{m \vee x \mid x \in \Omega\}$.*

**Theorem 1.** *Let $A$, $B$, $C$ are three equivalence classes on $\theta_\Omega$, where $\theta_\Omega$ is the congruence relation about $\Omega$. $\exists\ x \in A$, $y \in B$, and $x \mid_c y \in C$, then $C = \{x \mid_c y \mid x \in A, y \in B\}$.*

*Proof.* Firstly, we verify that for any $d_1, d_2 \in \Omega$, if $x \mid_c y \in C$, then $(x \vee d_1) \mid_c (y \vee d_2) \in C$. In fact,

$$\begin{aligned}(x \vee d_1) \mid_c (y \vee d_2) =& (x \vee d_1)c \vee (y \vee d_2)\bar{c} \\ & (xc \vee y\bar{c}) \vee (d_1 c \vee d_2 \bar{c}) \\ & (x \mid_c y) \vee (d_1 c \vee d_2 \bar{c})\end{aligned} \tag{18}$$

Obviously, $(d_1 c \vee d_2 \bar{c}) \in \Omega$, so $(x \vee d_1) \mid_c (y \vee d_2) \equiv (x \mid_c y) (\mod \theta_\Omega)$, *i.e.* $(x \vee d_1) \mid_c (y \vee d_2) \in \Omega$.

Secondly, from Proposition 1, $\exists m, n, d_3, d_4 \in \Omega$ of $A, B$, such that $x = m \vee d_3$, $y = n \vee d_4$. As a result of analysis in Eq.(18), $x \mid_c y \equiv (m \mid_c n) (\mod \theta_\Omega)$, *i.e.*, $m \mid_c n \in C$.

Finally, we verify that $m \mid_c n$ is a minimal element of $C$ and $(m \mid_c n) \vee d = (m \vee d) \mid_c (n \vee d)$. As a result of analysis in Eq.(18), if $d_1 = d_2 = d$, then $m \mid_c n \vee d = (m \vee d) \mid_c (n \vee d)$. Therefore $m \mid_c n$ is a minimal element of $C$.

To conclude, $C = \{(m \mid_c n) \vee d \mid d \in \Omega\} = \{x \mid_c y \mid x \in A, y \in B\}$. The theorem is proven.

**Proposition 2.** *Let $A$, $B$ are two equivalence classes on $\theta_\Omega$, and there exist $x \in A$, $y \in B$, such that $x \mid_c y \in C$, then, $x \mid_c y$ makes ergodic search $C$ while $x$ and $y$ make ergodic search $A$ and $B$, respectively.*

**Definition 1 (Convergence in terms of probability).** *Let $\xi_n$ a sequence of random variables, and $\xi$ a random variable, and all of them are defined on the same probability space. The sequence $\xi_n$ converges with a probability of $\xi$ if*

$$\lim_{n \to \infty} P(|\xi_n - \xi| < \varepsilon) = 1 \tag{19}$$

*for every $\varepsilon > 0$.*

**Definition 2 (Convergence with a probability of 1).** *Let $\xi_n$ a sequence of random variables, and $\xi$ a random variable, and all of them are defined on the same probability space. The sequence $\xi_n$ converges almost surely or almost everywhere or with probability of 1 or strongly towards $\xi$ if*

$$P\left(\lim_{n\to\infty} \xi_n = \xi\right) = 1; \tag{20}$$

*or*

$$P\left(\bigcap_{n=1}^{\infty} \bigcup_{k\geq n} [|\xi_n - \xi| \geq \varepsilon]\right) = 0 \tag{21}$$

*for every $\varepsilon > 0$.*

**Theorem 2.** *Let $\boldsymbol{x}^*$ is the global optimal solution to the problem (P′), and $f^* = f(\boldsymbol{x}^*)$. Assume that the clubs-based multi-swarm algorithm provides position series $\boldsymbol{x}_i(t)$ $(i = 1, 2, \cdots, n)$ at time $t$ by the iterated procedure. $\boldsymbol{p}^*$ is the best position among all the swarms explored so far,* i.e.

$$\boldsymbol{p}^*(t) = arg \min_{1\leq i\leq n} (f(\boldsymbol{p}^*(t-1)), f(\boldsymbol{p}_i(t))) \tag{22}$$

*Then,*

$$P\left(\lim_{t\to\infty} f(\boldsymbol{p}^*(t)) = f^*\right) = 1 \tag{23}$$

*Proof.* Let

$$\begin{aligned} D_0 &= \{\boldsymbol{x} \in \Omega | f(\boldsymbol{x}) - f^* < \varepsilon\} \\ D_1 &= \Omega \setminus D_0 \end{aligned} \tag{24}$$

for every $\varepsilon > 0$.

While the different swarm searches their feasible solutions by themselves, assume $\Delta p$ is the difference of the particle's position among different club swarms at the iteration time $t$. Therefore $-s \leq \Delta p \leq s$. $Rand(-1, 1)$ is a normal distributed random number within the interval [-1,1]. According to the update of the velocity and position by Eqs.(13)∼(14), $\Delta p$ belongs to the normal distribution, *i.e.* $\Delta p \sim [-s, s]$. During the iterated procedure from the time $t$ to $t + 1$, let $q_{ij}$ denote that $\boldsymbol{x}(t) \in D_i$ and $\boldsymbol{x}(t + 1) \in D_j$. Accordingly the particles' positions in the swarm could be classified into four states: $q_{00}$, $q_{01}$, $q_{10}$ and $q_{01}$. Obviously $q_{00} + q_{01} = 1$, $q_{10} + q_{11} = 1$. According to Borel-Cantelli Lemma and Particle State Transference [21], proving by the same methods, $q_{01} = 0$; $q_{00} = 1$; $q_{11} \leq c \in (0, 1)$ and $q_{10} \geq 1 - c \in (0, 1)$.

For $\forall \varepsilon > 0$, let $p_k = P\{|f(\boldsymbol{p}^*(k)) - f^*| \geq \varepsilon\}$, then

$$p_k = \begin{cases} 0 & \text{if } \exists T \in \{1, 2, \cdots, k\}, \boldsymbol{p}^*(T) \in D_0 \\ \bar{p}_k & \text{if } \boldsymbol{p}^*(t) \notin D_0, t = 1, 2, \cdots, k \end{cases} \tag{25}$$

According to Particle State Transference Lemma,

$$\bar{p}_k = P\{\boldsymbol{p}^*(t) \notin D_0, t = 1, 2, \cdots, k\} = q_{11}^k \leq c^k. \tag{26}$$

Hence,

$$\sum_{k=1}^{\infty} p_k \leq \sum_{k=1}^{\infty} c^k = \frac{c}{1-c} < \infty. \tag{27}$$

According to Borel-Cantelli Lemma,

$$P\left(\bigcap_{t=1}^{\infty} \bigcup_{k \geq t} |f(\boldsymbol{p}^*(k)) - f^*| \geq \varepsilon\right) = 0 \tag{28}$$

As defined in Definition 2, the sequence $f(\boldsymbol{p}^*(t))$ converges almost surely or almost everywhere or with probability 1 or strongly towards $f^*$. The theorem is proven.

## 6 Experiment Results and Discussions

The algorithms used for performance comparison were the Standard Particle Swarm Optimization (SPSO) ([11]) and a Genetic Algorithm (GA) ([50,51]). These algorithms share many similarities. GA is powerful stochastic global search and optimization methods, which are also inspired from the nature like the PSO. Genetic algorithms mimic an evolutionary natural selection process. Generations of solutions are evaluated according to a fitness value and only those candidates with high fitness values are used to create further solutions via crossover and mutation procedures. Both methods are valid and efficient methods in numeric programming and have been employed in various fields due to their strong convergence properties. Specific parameter settings for the algorithms are described in Table 1, where $D$ is the dimension of the position, i.e., the number of condition attributes. Besides the first small scale rough set reduction problem shown in Table 2, the maximum number of iterations is set as

**Table 1.** Parameter settings for the algorithms

| Algorithm | Parameter name | Value |
|---|---|---|
| | Size of the population | $(even)(int)(10 + 2 * sqrt(D))$ |
| GA | Probability of crossover | 0.8 |
| | Probability of mutation | 0.01 |
| | Swarm size | $(even)(int)(10 + 2 * sqrt(D))$ |
| | Self coefficient $c_1$ | $0.5 + log(2)$ |
| PSO(s) | Social coefficient $c_2$ | $0.5 + log(2)$ |
| | Inertia weight $w$ | 0.91 |
| | Clamping Coefficient $\rho$ | 0.5 |

$(int)(0.1 * recnum + 10 * (nfields - 1))$ for each trial, where $recnum$ is the number of records/rows and $nfields - 1$ is the number of condition attributes. Each experiment (for each algorithm) was repeated 10 times with different random seeds. If the standard deviation is larger than 20%, the number of trials were increased to 20.

To analyze the effectiveness and performance of the considered algorithms, first we tested a small scale rough set reduction problem shown in Table 2. In the experiments, the maximum number of iterations was fixed as 10. Each experiment were repeated 10 times with different random seeds. The results (the number of reduced attributes) for 10 GA runs were all 2. The results of 10 PSO runs were also all 2. The optimal result is supposed to be 2. But the reduction result for 10 GA runs is $\{2, 3\}$ while the reduction result for 10 PSO runs are $\{1, 4\}$ and $\{2, 3\}$. Table 3 depicts the reducts for Table 2 (Please also see Figure 5). For the small scale rough set reduction problem, GA has a same result than PSO. GA only provides one reduct, while PSOs provide one more reduct. There seems a conflict between the instances 13 and 15. It depends on conflict analysis and how to explain the knowledge, which will be tackled in future publications.

**Table 2.** A decision table

| *Instance* | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $d$ |
|---|---|---|---|---|---|
| $x_1$ | 1 | 1 | 1 | 1 | 0 |
| $x_2$ | 2 | 2 | 2 | 1 | 1 |
| $x_3$ | 1 | 1 | 1 | 1 | 0 |
| $x_4$ | 2 | 3 | 2 | 3 | 0 |
| $x_5$ | 2 | 2 | 2 | 1 | 1 |
| $x_6$ | 3 | 1 | 2 | 1 | 0 |
| $x_7$ | 1 | 2 | 3 | 2 | 2 |
| $x_8$ | 2 | 3 | 1 | 2 | 3 |
| $x_9$ | 3 | 1 | 2 | 1 | 1 |
| $x_{10}$ | 1 | 2 | 3 | 2 | 2 |
| $x_{11}$ | 3 | 1 | 2 | 1 | 1 |
| $x_{12}$ | 2 | 3 | 1 | 2 | 3 |
| $x_{13}$ | 4 | 3 | 4 | 2 | 1 |
| $x_{14}$ | 1 | 2 | 3 | 2 | 3 |
| $x_{15}$ | 4 | 3 | 4 | 2 | 2 |

Further we considered the datasets in Table 4 from AFS[1], AiLab[2] and UCI[3]. Figures 6, 7 and 8 illustrate the performance of the algorithms for lung-cancer, lymphography and mofn-3-7-10 datasets, respectively. For lung-cancer dataset, the results (the number of reduced attributes) for 10 GA runs were 10: { 1, 4, 8, 13, 18, 34, 38, 40, 50, 55 } (The number before the colon is the number of condition attributes, the numbers in brackets are attribute index, which represents

[1] http://sra.itc.it/research/afs/
[2] http://www.ailab.si/orange/datasets.asp
[3] http://www.datalab.uci.edu/data/mldb-sgi/data/

**Table 3.** A reduction of the data in Table 2

| Reduct | *Instance* | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $d$ |
|---|---|---|---|---|---|---|
| $\{1,4\}$ | | | | | | |
| | $x_1$ | 1 | | | 1 | 0 |
| | $x_2$ | 2 | | | 1 | 1 |
| | $x_4$ | 2 | | | 3 | 0 |
| | $x_6$ | 3 | | | 1 | 0 |
| | $x_7$ | 1 | | | 2 | 2 |
| | $x_8$ | 2 | | | 2 | 3 |
| | $x_9$ | 3 | | | 1 | 1 |
| | $x_{13}$ | 4 | | | 2 | 1 |
| | $x_{14}$ | 1 | | | 2 | 3 |
| | $x_{15}$ | 4 | | | 2 | 2 |
| $\{2,3\}$ | | | | | | |
| | $x_1$ | | 1 | 1 | | 0 |
| | $x_2$ | | 2 | 2 | | 1 |
| | $x_4$ | | 3 | 2 | | 0 |
| | $x_6$ | | 1 | 2 | | 0 |
| | $x_7$ | | 2 | 3 | | 2 |
| | $x_8$ | | 3 | 1 | | 3 |
| | $x_9$ | | 1 | 2 | | 1 |
| | $x_{13}$ | | 3 | 4 | | 1 |
| | $x_{14}$ | | 2 | 3 | | 3 |
| | $x_{15}$ | | 3 | 4 | | 2 |

**Table 4.** Datasets used in the experiments

| Dataset | Size | $\|C\|$ | Class | GA | | PSO | | MSSO | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $L$ | $R$ | $L$ | $R$ | $L$ | $R$ |
| lung-cancer | 27 | 56 | 3 | 10 | 1 | 6 | 3 | 6 | 3 |
| zoo | 101 | 16 | 7 | 5 | 1 | 5 | 2 | 5 | 3 |
| corral | 128 | 6 | 2 | 4 | 1 | 4 | 1 | 4 | 1 |
| lymphography | 148 | 18 | 4 | 7 | 1 | 6 | 2 | 6 | 1 |
| hayes-roth | 160 | 4 | 3 | 3 | 1 | 3 | 1 | 3 | 1 |
| shuttle-landing-control | 253 | 6 | 2 | 6 | - | 6 | - | 6 | - |
| soybean-large-test | 296 | 35 | 15 | 12 | 1 | 10 | 3 | 8 | 3 |
| monks | 432 | 6 | 2 | 3 | 1 | 3 | 1 | 3 | 1 |
| xd6-test | 512 | 9 | 2 | 9 | - | 9 | - | 9 | - |
| balance-scale | 625 | 4 | 3 | 4 | - | 4 | - | 4 | - |
| breast-cancer-wisconsin | 683 | 9 | 2 | 4 | 1 | 4 | 2 | 4 | 3 |
| mofn-3-7-10 | 1024 | 10 | 2 | 7 | 1 | 7 | 1 | 7 | 1 |
| parity5+5 | 1024 | 10 | 2 | 5 | 1 | 5 | 1 | 5 | 1 |

a reduction solution); the results of 10 PSO runs were PSO 7: { 1, 6, 12, 27, 29, 35, 41 }, 6: { 2, 3, 12, 22, 25, 56 }, 7: { 2, 3, 8, 12, 22, 31, 49 }; the results of 10 MSSO runs were 6: { 4, 6, 14, 31, 49, 53 }, 6: { 4, 6, 9, 23, 27, 54 }, 6: { 3, 10,

20, 32, 34, 56 }. For lymphography dataset, the results of 10 GA runs all were 7: { 2, 11, 12, 13, 14, 16, 18 }; the results of 10 PSO runs were PSO 7: { 3, 8, 11, 12, 13, 14, 15 }, 6: { 2, 13, 14, 15, 16, 18 }, 6: { 2, 13, 14, 15, 16, 18 }; the results of 10 MSSO runs were 6:{ 2, 13, 14, 15, 16, 18 }. For soybean-large-test dataset, the results of 10 GA runs all were 12: { 1, 3, 4, 5, 6, 7, 13, 15, 16, 22, 32, 35 }; the results of 10 PSO runs were 10:{ 1, 3, 5, 6, 7, 12, 15, 18, 22, 31 }, 10: { 1, 3, 5, 6, 7, 15, 23, 26, 28, 30 }, 10: { 1, 2, 3, 6, 7, 9, 15, 21, 22, 30 }; the results of 10 MSSO runs were 9: { 1, 3, 5, 6, 7, 15, 22, 30, 34 }, 8: { 1, 3, 4, 6, 7, 10, 15, 22 }, 9: { 1, 3, 5, 6, 7, 13, 22, 25, 31 }. Other results are shown in Table 4, in which $L$ is the minimum length and $R$ is the number of the obtained reducts. "-" means that all features *cannot* be reduced. MSSO usually obtained better results than GA and PSO, specially for the large scale problems. Although the three algorithms achieved the same-length results for some datasets, MSSO usually can provide more reducts for multi-knowledge extraction. It indicates that MSSO has a better performance than other two algorithms for the larger scale rough set reduction problem. It is to be noted that PSO usually can obtain more candidate solutions for the reduction problems.

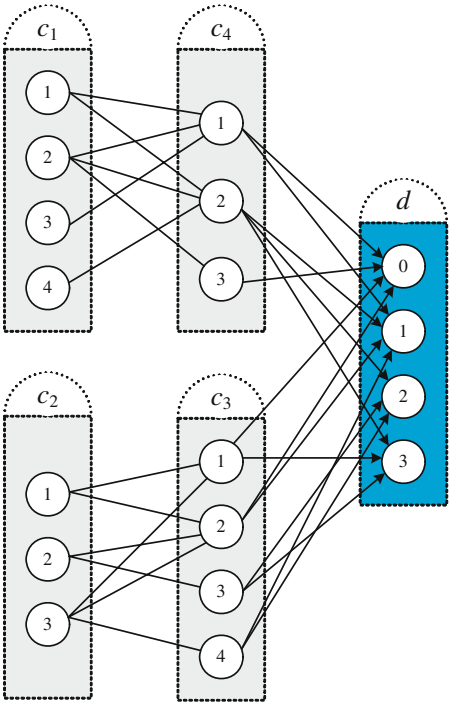


**Fig. 5.** Rule networks based on Table 3

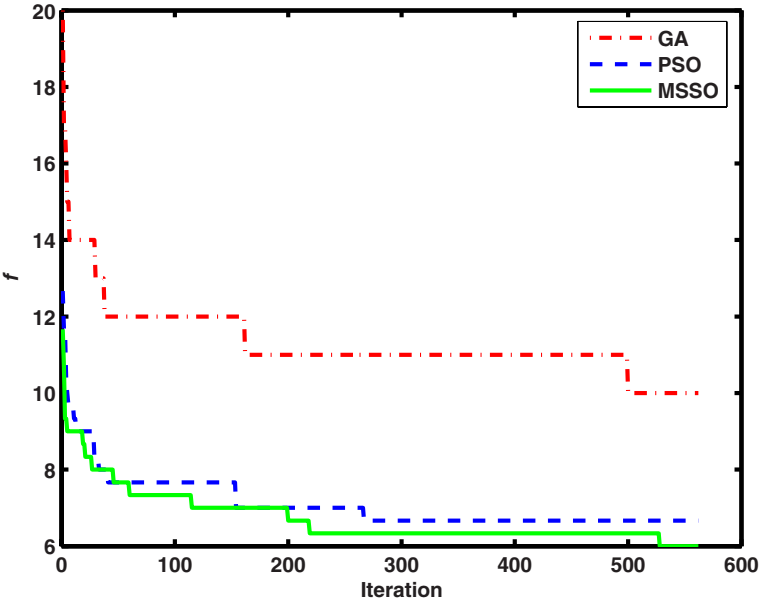


**Fig. 6.** Performance of rough set reduction for lung-cancer dataset

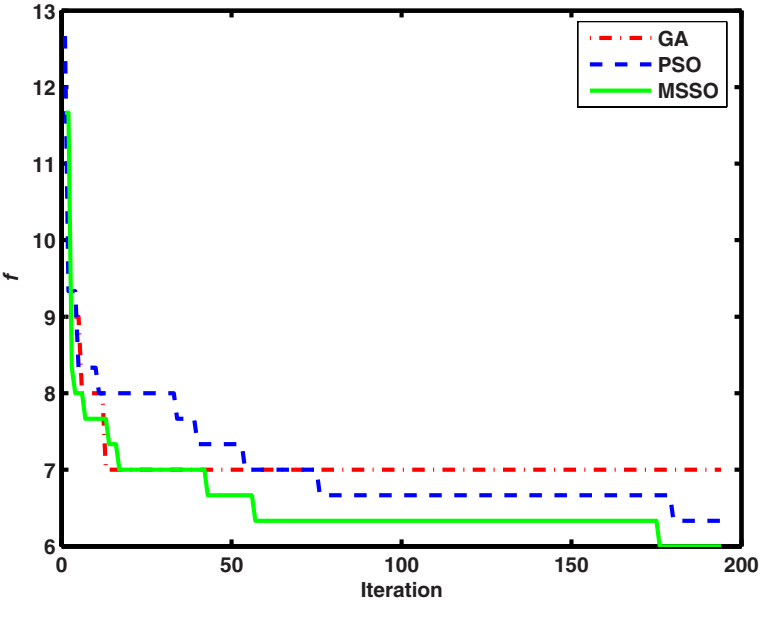


**Fig. 7.** Performance of rough set reduction for lymphography dataset

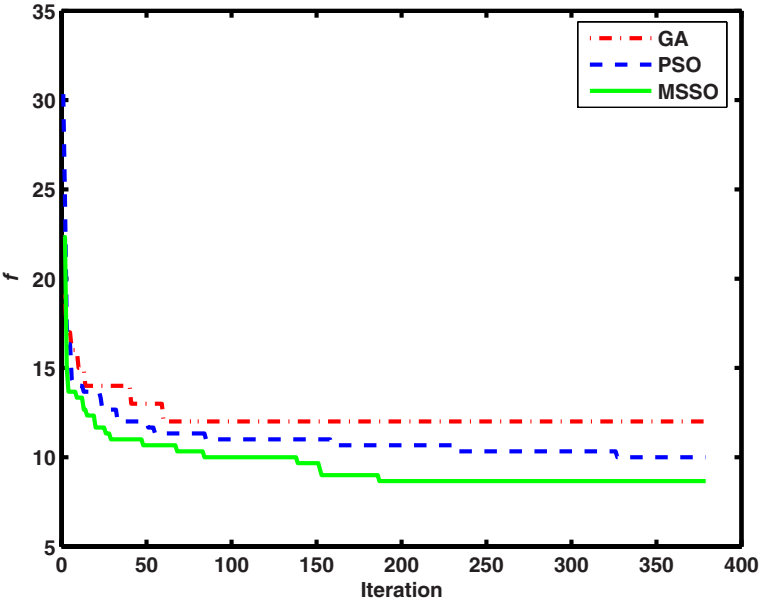


**Fig. 8.** Performance of rough set reduction for soybean-large-test dataset

## 7 Conclusions

In this Chapter, we investigated multi-knowledge extraction using particle swarm optimization and genetic algorithm techniques. The considered approaches discovered the good feature combinations in an efficient way to observe the change of positive region as the particles explored the search space. The multi-swarm search approach offer great benefits for multiple reduction problems, because different individuals encode different reducts. Empirical results indicate that the proposed approach usually obtained better results than GA and standard PSO, specially for large scale problems, although its stability need to be improved in further research. MSSO has better convergence than GA for the larger scale rough set reduction problem, although MSSO is worst for some small scale rough set reduction problems. MSSO also can obtain more candidate solutions for the reduction problems. Empirical results illustrated that the multi-swarm search approach was an effective approach to solve multi-knowledge extraction.

## Acknowledgements

This work was partly supported by NSFC (60873054) and DLMU (DLMU-ZL-200709).

## References


1. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
2. Pawlak, Z.: Rough Sets: Present State and The Future. Foundations of Computing and Decision Sciences 18, 157–166 (1993)
3. Pawlak, Z.: Rough Sets and Intelligent Data Analysis. Information Sciences 147, 1–12 (2002)
4. Kusiak, A.: Rough Set Theory: A Data Mining Tool for Semiconductor Manufacturing. IEEE Transactions on Electronics Packaging Manufacturing 24, 44–50 (2001)
5. Shang, C., Shen, Q.: Rough Feature Selection for Neural Network Based Image Classification. International Journal of Image and Graphics 2, 541–555 (2002)
6. Tay, F.E.H.: Economic And Financial Prediction Using Rough Sets Model. European Journal of Operational Research 141, 641–659 (2002)
7. Świniarski, R.W., Skowron, A.: Rough Set Methods in Feature Selection and Recognition. Pattern Recognition Letters 24, 833–849 (2003)
8. Beaubouef, T., Ladner, R., Petry, F.: Rough Set Spatial Data Modeling for Data Mining. International Journal of Intelligent Systems 19, 567–584 (2004)
9. Shen, L., Tay, F.E.H.: Tay Applying Rough Sets to Market Timing Decisions. Decision Support Systems 37, 583–597 (2004)
10. Gupta, K.M., Moore, P.G., Aha, D.W., Pal, S.K.: Rough Set Feature Selection Methods for Case-Based Categorization of Text Documents. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 792–798. Springer, Heidelberg (2005)
11. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
12. Clerc, M., Kennedy, J.: The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
13. Clerc, M.: Particle Swarm Optimization. ISTE Publishing Company, London (2006)
14. Liu, H., Abraham, A., Clerc, M.: Chaotic Dynamic Characteristics in Swarm Intelligence. Applied Soft Computing Journal 7, 1019–1026 (2007)
15. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. Natural Computing 1, 235–306 (2002)
16. Abraham, A., Guo, H., Liu, H.: Swarm Intelligence: Foundations, Perspectives and Applications. In: Nedjah, N., Mourelle, L. (eds.) Swarm Intelligent Systems. Studies in Computational Intelligence, pp. 3–25. Springer, Germany (2006)
17. Salman, A., Ahmad, I., Al-Madani, S.: Particle Swarm Optimization for Task Assignment Problem. Microprocessors and Microsystems 26, 363–371 (2002)
18. Sousa, T., Silva, A., Neves, A.: Particle Swarm Based Data Mining Algorithms for Classification Tasks. Parallel Computing 30, 767–783 (2004)
19. Liu, B., Wang, L., Jin, Y., Tang, F., Huang, D.: Improved Particle Swarm Optimization Combined With Chaos. Chaos, Solitons and Fractals 25, 1261–1271 (2005)
20. Schute, J.F., Groenwold, A.A.: A Study of Global Optimization Using Particle Swarms. Journal of Global Optimization 3, 103–108 (2005)
21. Liu, H., Abraham, A.: An Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems. Journal of Universal Computer Science 13(7), 1032–1054 (2007)

22. Boussouf, M.: A Hybrid Approach to Feature Selection. In: Żytkow, J.M. (ed.) PKDD 1998. LNCS, vol. 1510, pp. 231–238. Springer, Heidelberg (1998)
23. Skowron, A., Rauszer, C.: The Discernibility Matrices and Functions in Information Systems. In: Świniarski, R.W. (ed.) Handbook of Applications and Advances of the Rough Set Theory, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
24. Zhang, J., Wang, J., Li, D., He, H., Sun, J.: A New Heuristic Reduct Algorithm Base on Rough Sets Theory. In: Dong, G., Tang, C., Wang, W. (eds.) WAIM 2003. LNCS, vol. 2762, pp. 247–253. Springer, Heidelberg (2003)
25. Hu, K., Diao, L., Lu, Y.-c., Shi, C.-Y.: A Heuristic Optimal Reduct Algorithm. In: Leung, K.-S., Chan, L., Meng, H. (eds.) IDEAL 2000. LNCS, vol. 1983, pp. 139–144. Springer, Heidelberg (2000)
26. Zhong, N., Dong, J.: Using Rough Sets with Heuristics for Feature Selection. Journal of Intelligent Information Systems 16, 199–214 (2001)
27. Banerjee, M., Mitra, S., Anand, A.: Feature Selection Using Rough Sets. Studies in Computational Intelligence, vol. 16, pp. 3–20. Springer, Heidelberg (2006)
28. Wu, Q., Bell, D.: Multi-knowledge Extraction and Application. In: Wang, G., Liu, Q., Yao, Y., Skowron, A. (eds.) RSFDGrC 2003. LNCS (LNAI), vol. 2639, pp. 574–575. Springer, Heidelberg (2003)
29. Wu, Q.: Multiknowledge and Computing Network Model for Decision Making and Localisation of Robots. Thesis, University of Ulster (2005)
30. Wang, G.: Rough Reduction in Algebra View and Information View. International Journal of Intelligent Systems 18, 679–688 (2003)
31. Xu, Z., Liu, Z., Yang, B., Song, W.: A Quick Attibute Reduction Algorithm with Complexity of $\mathrm{Max}(O(|C||U|), O(|C|^2|U/C|))$. Chinese Journal of Computers 29, 391–399 (2006)
32. Cristian, T.I.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317–325 (2003)
33. van den Bergh, F., Engelbrecht, A.P.: A Study of Particle Swarm Optimization Particle Trajectories. Information Sciences 176(8), 937–971 (2006)
34. Grosan, C., Abraham, A., Nicoara, M.: Search Optimization Using Hybrid Particle Sub-swarms and Evolutionary Algorithms. International Journal of Simulation Systems, Science & Technology 6(10), 60–79 (2005)
35. Jiang, C., Etorre, B.: A hybrid Method of Chaotic Particle Swarm Optimization and Linear Interior for Reactive Power Optimisation. Mathematics and Computers in Simulation 68, 57–65 (2005)
36. Liu, H., Li, B., Ji, Y., Tong, S.: Particle Swarm Optimisation from lbest to gbest. In: Abraham, A., Baets, B.D., Koppen, M. (eds.) Applied Soft Computing Technologies: The Challenge of Complexity, pp. 537–545. Springer, Heidelberg (2006)
37. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. IEEE Transactions on Evolutionary Computation 10(3), 281–295 (2006)
38. Blackwell, T., Branke, J.: Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation 10(4), 459–472 (2006)
39. Niu, B., Zhu, Y., He, X., Wu, H.: MCPSO: A Multi-Swarm Cooperative Particle Swarm Optimizer. Applied Mathematics and Computation 185(2), 1050–1062 (2007)
40. Settles, M., Soule, T.: Breeding swarms: a GA/PSO hybrid. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 161–168 (2005)

41. Elshamy, W., Emara, H.M., Bahgat, A.: Clubs-based Particle Swarm Optimization. In: Proceedings of the IEEE International Conference on Swarm Intelligence Symposium, vol. 1, pp. 289–296 (2007)
42. Guo, C., Tang, H.: Global Convergence Properties of Evolution Stragtegies. Mathematica Numerica Sinica 23(1), 105–110 (2001)
43. He, R., Wang, Y., Wang, Q., Zhou, J., Hu, C.: An Improved Particle Swarm Optimization Based on Self-adaptive Escape Velocity. Journal of Software 16(12), 2036–2044 (2005)
44. Weisstein, E.W.: Borel-Cantelli Lemma, From MathWorld – A Wolfram Web Resource (2007), http://mathworld.wolfram.com/Borel-CantelliLemma.html
45. Xu, Z., Cheng, G., Liang, Y.: Search Capability for An Algebraic Crossover. Journal of Xi'an Jiaotong University 33(10), 88–99 (1999)
46. Whitley, L.D.: Fundamental Principles of Deception in Genetic Search. Foundation of Genetic Algorithms, pp. 221–241. Morgan Kaufmann Publishers, California (1991)
47. Mastrolilli, M., Gambardella, L.M.: Effective Neighborhood Functions for the Flexible Job Shop Problem. Journal of Scheduling 3(1), 3–20 (2002)
48. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
49. Goldberg, D.E.: Genetic Algorithms in search, optimization, and machine learning. Addison-Wesley Publishing Corporation, Inc., Reading (1989)
50. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Netherland (2000)
51. Abraham, A.: Evolutionary Computation. In: Sydenham, P., Thorn, R. (eds.) Handbook for Measurement Systems Design, pp. 920–931. John Wiley and Sons Ltd., London (2005)