A Swarm-based Rough Set Approach for Group Decision Support Systems

Mingyan Zhao[1], Hongbo Liu[2,3], Ajith Abraham[2,3] and Emilio Corchado[3,4]
[1]School of Software, Dalian University of Technology, 116620 Dalian, China.
[2]School of Computer Science, Dalian Maritime University, 116024 Dalian, China.
[3]Machine Intelligence Research Labs (MIR Labs),
Scientific Network for Innovation and Research Excellence, USA.
[4]Department of Civil Engineering, University of Burgos, 09006 Burgos, Spain.
mingy_zhao@163.com, lhb@dlut.edu.cn, ajith.abraham@ieee.org, escorchado@ubu.es

Abstract

This paper present a class of investment problem, in which many items could be chosen in a group decision environment. Usually there is a decision table from the board of directors after discussions. Most of the data come from their experience or estimation. The information is redundant and inaccurate. Swarm-based rough set approach is introduced to make an attempt to solve the problem. Rough set theory provides a mathematical tool that can be used for both feature selection and information reduction. The swarm-based reduction approaches are attractive to find multiple reducts in the decision systems, which could be applied to generate multiple investment planning and to improve the decision. Empirical results illustrate that the approach can be applied for the class of investment problems effectively.

1 Introduction

Investment problem is one of the most popular issues in today's economic life. Usually nobody knows accurately which investment planning is absolutely right and wise before the last result is made a showdown. Rough set theory [1, 2, 3, 4] provides a mathematical tool that can be used for the inaccurate and redundant information. It helps us to find out the minimal item sets called '*reducts*' to make a decision. A good investment planning is from the judgments of multiple directors or experts. So the reduct of the information system is usually not unique. There may be many subsets of items, which preserve the equivalence class structure 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. What's more, conventional rough set-based information reduction usually tries to find a good reduct or to select a set of features [5].

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 [6]. The swarm intelligent model helps to find optimal regions of complex search spaces through interaction of individuals in a population of particles [7, 8, 9, 10]. As an algorithm, its main strength is its fast convergence. It has exhibited good performance across a wide range of applications [11, 12, 13, 14]. Swarm-based approaches are of great use in the multiple reduction search, because different individual trends to be encoded to different reducts. The particle swarm algorithm is particularly attractive for rough set reduction to discover multiple reducts or the best item combinations as they proceed throughout the search space [15].

The main focus of this paper is to investigate swarm-based rough set reduction algorithm and its application in finding multiple reducts for the investment problem. The rest of the paper is organized as follows. Some related terms and theorems on rough set theory are explained briefly in Section 3. Particle swarm approach for reduction is introduced in Section 4. The algorithm performance demonstration are given in Section 5 and finally conclusions are given in Section 6.

2 Problem Description

It would be related to the company's survival to decide how to operate limited investment funds. Usually it depends on the Board of Directors to discuss the solutions. It is difficult to draw a unanimous conclusion. They are unwilling to follow the majority rule simply, since the truth often lies in the hands of a minority. There would also be great risk

978-0-7695-3745-0/09 $25.00 © 2009 IEEE
DOI 10.1109/HIS.2009.287

Table 1. A group decision planning information.

$Planning$	$estate$	$stock$	$fund$	$bond$	$profit$
p_1	1	1	1	1	10
p_2	2	2	2	1	20
p_3	1	1	1	1	10
p_4	2	3	2	3	10
p_5	2	2	2	1	20
p_6	3	1	2	1	10
p_7	1	2	3	2	30
p_8	2	3	1	2	40
p_9	3	1	2	1	20
p_{10}	1	2	3	2	30
p_{11}	3	1	2	1	20
p_{12}	2	3	1	2	40
p_{13}	4	3	4	2	20
p_{14}	1	2	3	2	40
p_{15}	4	3	4	2	30

if they "put all eggs in one basket". They list all the investment items and evaluate the profits. Although the president of the board can make a decision arbitrarily, he/she is reluctant to do so for their own benefits and collective benefits reluctantly. They turn to score the potential investment planning together with some employed experts or consultants. For example, Table 1 shows some investment planning information. In p_4, there would be 2 units of the real estate investment, 3 units of stock, 2 units of funds, and buy 3 units of the national bonds, then 10 units of expected profits. All the investment planning would not be judged simply through the expected profits, which are not both accurate and reliable. It is possible not to isolate completely each investment in real estate and investment in government bonds. In the current economic crisis, U.S. Government sells the national bonds for getting the funds to remedy the real estate market. It is also possible that some of the considered items make no contribution for the profits.

3 Rough Set Reduction

The basic concepts of rough set theory and its philosophy are presented and illustrated with examples in [1, 2, 3, 4, 5, 16, 17, 18]. 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 $T = (U, C, D, V, f)$. 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:

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

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.

Positive Region: Given a decision table $T = (U, C, D, V, f)$. Let $B \subseteq C$. 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.,

$$POS_B(D) = \bigcup_{X \in U/B \wedge \forall x, y \in X \Rightarrow f(x,D) = f(y,D)} X. \tag{2}$$

Dependency degree: Given a decision table $T = (U, C, D, V, f)$. For given $U/C = \{x_1,\ x_2,\ \cdots,\ x_n\}$, $U/D = \{Y_1,\ Y_2,\ \cdots,\ Y_m\}$, then dependency degree of $\boldsymbol{D}$ with respect to $\boldsymbol{C}$ is defined as follow:

$$k_C(\boldsymbol{D}) = \frac{1}{|U|} \sum_{i=1}^{m} |POS_C(Y_i)|. \tag{3}$$

where $|U|$ is the cardinality of U, $POS_C(Y_i)$ denotes the positive region of Y_i with respect to $\boldsymbol{C}$. Obviously, $0 \leq k_C(D) \leq 1$. If $k_C(D)$=1, $\boldsymbol{D}$ depends totally on $\boldsymbol{C}$. This means that the partition generated by $\boldsymbol{C}$ is finer than the partition generated by $\boldsymbol{D}$. If $k_C(D)$= 0, $\boldsymbol{D}$ is independent totally of $\boldsymbol{C}$. It means that $\boldsymbol{C}$ has no effect on decision result for $\boldsymbol{D}$. If $0 < k_C(D) < 1$, we say that $\boldsymbol{D}$ depends partially on $\boldsymbol{C}$ in degree $k_C(D)$.

Significance of attributes: Given a decision table $T = (U, C, D, V, f)$. The significance of an attribute c ($c \in C$)

with respect to D is defined as follow:

$$sig_D(c) = k_C(D) - k_{C-\{c\}}(D). \tag{4}$$

Obviously, $0 \leq sig_D(c) \leq 1$. If $C = \{c\}$, then $sig_D(c) = k_C(D) - k_{\emptyset}(D) = k_C(D)$, where $k_{\emptyset}(D) = 0$. The significance of an attribute can be evaluated by measuring effect of removing the attribute from an information table on decision defined by the Table, which generalizes the idea of attribute reduction. The two concepts enable us the evaluation of attributes not only by two-valued scale, $indispensable - dispensable$, but by assigning to an attribute a real number from the interval [0, 1] to express its significance in the decision environment.

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.

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:

$$U^{'} = \{u_1^{'}, u_2^{'}, \cdots, u_m^{'}\}. \tag{5}$$

and

$$POS_C(D) = [u_{i_1}^{'}]_C \cup [u_{i_2}^{'}]_C \cup \cdots \cup [u_{i_t}^{'}]_C. \tag{6}$$

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:

$$U_{pos}^{'} = \{u_{i_1}^{'}, u_{i_2}^{'}, \cdots, u_{i_t}^{'}\}. \tag{7}$$

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

$$POS_B^{'}(D) = \bigcup_{X \in U'/B \wedge X \subseteq U'_{pos} \wedge |X/D|=1} X \tag{8}$$

where $|X/D|$ represents the cardinality of the set X/D. $\forall B \subseteq C$, $POS_B(D) = POS_C(D)$ if $POS_B^{'} = U_{pos}^{'}$ [18]. 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 do not 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 Planning Reduction and Selection

Given a decision table $T = (U, C, D, V, f)$, the set of condition attributes, C, consist of m attributes. We set up a search space of m dimension for the rough set reduction. 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 for this combinatorial problem.

During the search procedure, each individual is evaluated using the fitness. According to the definition of rough set reduct, the reduction solution must ensure that 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 the proposed 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.

As a summary, the 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 $\vec{p_i}$ (i is the index of the particle), and a velocity represented by a velocity-vector $\vec{v_i}$. Each particle remembers its own best position so far in a vector $\vec{p_i}^{\#}$, and its j-th dimensional value is $p_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\vec{p}^{*}$, 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:

$$P(p_{ij}(t) = 1) = f(p_{ij}(t-1), v_{ij}(t-1), p_{ij}^{\#}(t-1), p_j^{*}(t-1)). \tag{9}$$

where the probability function is

$$\Gamma(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}}. \tag{10}$$

At each time step, each particle updates its velocity and moves to a new position according to Eqs.(11) and (12):

$$v_{ij}(t) = wv_{ij}(t-1) + c_1r_1(p^{\#}_{ij}(t-1) - p_{ij}(t-1)) + c_2r_2(p^*_j(t-1) - p_{ij}(t-1)). \tag{11}$$

$$p_{ij}(t) = \begin{cases} 1 & \text{if } \rho < \Gamma(v_{ij}(t)); \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

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. ρ is a random number within the closed interval [0, 1]. From Eq.(11), 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. The pseudo-code for the particle swarm search method is illustrated in Algorithm 1. Since usually the maximum investment is preferable within the fund limits, the larger planning would be chosen.

Algorithm 1 A Rough Set Reduct Algorithm Based on Particle Swarm Optimization Algorithm

```
01.Calculate $U^{'}$, $U^{'}_{pos}$ using Eqs.(5) and (7).
02. Initialize the size of the particle swarm $n$,
02.    and other parameters.
03. Initialize the positions and the velocities
03.    for all the particles randomly.
04.While (the end criterion is not met) do
05.    $t = t + 1$;
06.    Calculate the fitness value of each particle,
06.        if $POS^{'}_{E} \neq U^{'}_{pos}$, the fitness is punished
06.            as the total number of the condition attributes,
06.        else the fitness is the number of '1' in the position.
07.    $\vec{p}^* = argmin^n_{i=1}(f(\vec{p}^*(t-1))$,
07.        $f(\vec{p}_1(t)), f(\vec{p}_2(t)), \cdots, f(\vec{p}_i(t)), \cdots, f(\vec{p}_n(t)))$;
08.    For $i$= 1 to $n$
09.        $\vec{p}^{\#}_i(t) = argmin^n_{i=1}(f(\vec{p}^{\#}_i(t-1)), f(\vec{p}_i(t))$;
10.        For $j$ = 1 to $d$
11.            Update the $j$-th dimension value of $\vec{p}_i$ and $\vec{v}_i$
11.              according to Eqs.(11) and (12);
12.        Next $j$
13.    Next $i$
14.End While.
```

5. Algorithm Performance Demonstration

To analyze the effectiveness and performance of the considered algorithm, we tested the investment problem shown in Table 1. We first transform the gross information to a decision table as shown in Table 3. We reduce and discretize the expected profits of the planning, since only the relative values are considered for our algorithm. In our experiments, the maximum number of iterations was fixed as 10. Each experiment were repeated 10 times using different random seeds. Other parameter settings for the algorithm are described in Table 2, where D is the dimension of the position and each dimension maps one condition attribute.

Table 2. Parameter settings for the algorithm.

Parameter name	Value
Swarm size	$(even)(int)(10 + 2 * sqrt(D))$
Self coefficient c_1	$0.5 + log(2)$
Social coefficient c_2	$0.5 + log(2)$
Inertia weight w	0.91
Clamping Coefficient ρ	0.5

Table 3. A decision table.

$Planning$	c_1	c_2	c_3	c_4	d
p_1	1	1	1	1	1
p_2	2	2	2	1	2
p_3	1	1	1	1	1
p_4	2	3	2	3	1
p_5	2	2	2	1	2
p_6	3	1	2	1	1
p_7	1	2	3	2	3
p_8	2	3	1	2	4
p_9	3	1	2	1	2
p_{10}	1	2	3	2	3
p_{11}	3	1	2	1	2
p_{12}	2	3	1	2	4
p_{13}	4	3	4	2	2
p_{14}	1	2	3	2	4
p_{15}	4	3	4	2	3

The results (the number of reduced attributes) for 10 PSO runs were all 2. The optimal result is supposed to be 2. The reduction result for 10 PSO runs are $\{1,4\}$ and $\{2,3\}$. Table 4 depicts the reducts for Table 3. So the planning p_{15} would be chosen.

6 Conclusions and Future Work

In this paper, we investigated multi-item investment problem using rough set theory and particle swarm optimization techniques. The considered approaches discov-

Table 4. A reduction of the data in Table 3.

Reduct	$Planning$	c_1	c_2	c_3	c_4	d
$\{1,4\}$						
	p_1	1			1	1
	p_2	2			1	2
	p_4	2			3	3
	p_6	3			1	1
	p_7	1			2	3
	p_8	2			2	4
	p_9	3			1	2
	p_{13}	4			2	2
	p_{14}	1			2	4
	p_{15}	4			2	3
$\{2,3\}$						
	p_1		1	1		1
	p_2		2	2		2
	p_4		3	2		1
	p_6		1	2		1
	p_7		2	3		3
	p_8		3	1		4
	p_9		1	2		2
	p_{13}		3	4		2
	p_{14}		2	3		4
	p_{15}		3	4		3

ered the good feature combinations in an efficient way to observe the change of positive region as the particles explored the search space. The swarm-based search approach offer great benefits for multiple reduction, because different individuals encode different reducts. The proposed approach also can obtain multiple candidate solutions for the reduction. Empirical results illustrated that the swarm-based search approach was effective to solve the investment problem.

Our future work is targeted to make an attempt for more instances and involve more heuristics approaches.

7 Acknowledgments

The authors would like to thank Dr. Benxian Yue for his scientific collaboration in this research work. This work was partly supported by NSFC (60873054).

References

[1] Z. Pawlak. "Rough Sets". *International Journal of Computer and Information Sciences*, 1982, 11, pp. 341–356.

[2] Z. Pawlak. "Rough Sets: Present State and The Future". *Foundations of Computing and Decision Sciences*, 1993, 18, pp. 157–166.

[3] J. Komorowski, L. Polkowski and A. Skowron. "Rough sets: A tutorial". In: Rough Fuzzy Hybridization. A New Trend in Decision Making. Pal S K, Skowron A (eds.), Springer, 1999, pp. 3–98.

[4] Z. Pawlak. "Rough Sets and Intelligent Data Analysis". *Information Sciences*, 2002, 147, pp. 1–12.

[5] Q. Wu and D. Bell. "Multi-knowledge Extraction and Application". *Lecture Notes in Computer Science*, 2003, 2639, pp. 574–575.

[6] J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, CA, 2001.

[7] M. Clerc and J. Kennedy. "The Particle Swarm-explosion, Stability, and Convergence in A Multidimensional Complex Space". *IEEE Transactions on Evolutionary Computation*, 2002, 6, pp. 58–73.

[8] A.P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, NY, 2005

[9] M. Clerc. *Particle Swarm Optimization*. ISTE Publishing Company, London, 2006.

[10] H. Liu, A. Abraham and M. Clerc. "Chaotic Dynamic Characteristics in Swarm Intelligence". *Applied Soft Computing Journal*, 2007, 7, pp. 1019–1026.

[11] T. Sousa, A. Silva and A. Neves. "Particle Swarm Based Data Mining Algorithms for Classification Tasks". *Parallel Computing*, 2004, 30, pp. 767–783.

[12] J.F. Schute and A.A. Groenwold. "A Study of Global Optimization Using Particle Swarms". *Journal of Global Optimization*, 2005, 3, pp. 193–108.

[13] H. Liu and A. Abraham. "An Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems". *Journal of Universal Computer Science*, 2007, 13(7), pp. 1032–1054.

[14] M. Zhao, A. Abraham, C. Grosan and H. Liu. "A Fuzzy Particle Swarm Approach for Multiobjective Quadratic Assignment Problems". In: Proceedings of the IEEE Second Asia International Conference on Modeling and Simulation, IEEE Computer Society Press, 2008, pp. 516–521.

[15] B. Yue, W. Yao, A. Abraham and H. Liu. "A New Rough Set Reduct Algorithm Based on Particle Swarm Optimization". *Lecture Notes in Computer Science*, 2007, 4527, pp. 397–406.

[16] N. Zhong and J. Dong. "Using Rough Sets with Heuristics for Feature Selection". *Journal of Intelligent Information Systems*, 2001, 16, pp. 199–214.

[17] G. Wang. "Rough Reduction in Algebra View and Information View". *International Journal of Intelligent Systems*, 2003, 18, pp. 679–688.

[18] Z. Xu, Z. Liu, B. Yang and W. Song. "A Quick Attibute Reduction Algorithm with Complexity of Max$(O(|C||U|), O(|C|^2|U/C|))$". *Chinese Journal of Computers*, 2006, 29, pp. 391–399.