Chaotic Rough Particle Swarm Optimization Algorithms

Bilal Alatas and Erhan Akin
Firat University
Turkey

1. Introduction

The problem of finding appropriate representations for various is a subject of continued research in the field of artificial intelligence and related fields. In some practical situations, mathematical and computational tools for faithfully modeling or representing systems with uncertainties, inaccuracies or variability in computation should be provided; and it is preferable to develop models that use ranges as values. A need to provide tolerance ranges and inability to record accurate values of the variables are examples of such a situation where ranges of values must be used (Lingras, 1996). Representations with ranges improve data integrity for non-integral numerical attributes in data storage and would be preferable due to no lose of information. Rough patterns proposed by Lingras are based on an upper and a lower bound that form a rough value that can be used to effectively represent a range or set of values for variables such as daily weather, stock price ranges, fault signal, hourly traffic volume, and daily financial indicators (Lingras, 1996; Lingras & Davies, 2001). The problems involving, on input/output or somewhere at the intermediate stages, interval or, more generally, bounded and set-membership uncertainties and ambiguities may be overcome by the use of rough patterns. Further developments in rough set theory have shown that the general concept of upper and lower bounds provide a wider framework that may be useful for different types of applications (Lingras & Davies, 2001).
Generating random sequences with a long period and good uniformity is very important for easily simulating complex phenomena, sampling, numerical analysis, decision making and especially in heuristic optimization. Its quality determines the reduction of storage and computation time to achieve a desired accuracy. Chaos is a deterministic, random-like process found in non-linear, dynamical system, which is non-period, non-converging and bounded. Moreover, it has a very sensitive dependence upon its initial condition and parameter (Schuster, 1998). The nature of chaos is apparently random and unpredictable and it also possesses an element of regularity. Mathematically, chaos is randomness of a simple deterministic dynamical system and chaotic system may be considered as sources of randomness.
Chaotic sequences have been proven easy and fast to generate and store, there is no need for storage of long sequences (Heidari-Bateni & McGillem, 1994). Merely a few functions (chaotic maps) and few parameters (initial conditions) are needed even for very long sequences. In addition, an enormous number of different sequences can be generated simply

Source: Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, ISBN 978-3-902613-09-7, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria

by changing its initial condition. Moreover these sequences are deterministic and reproducible. The choice of chaotic sequences is justified theoretically by their unpredictability, i.e., by their spread-spectrum characteristic, and ergodic properties.
In this chapter, a generalization of particle swarm optimization (PSO) based on rough values has been proposed. Furthermore, sequences generated from chaotic systems substitute random numbers in all phases of PSO where it is necessary to make a random-based choice. By this way it is intended to develop the global convergence and to prevent to stick on a local solution. The proposed chaotic rough particle swarm optimization algorithm (CRPSO) can complement the existing tools developed in rough computing using chaos. Definitions of basic building blocks of CRPSO such as rough decision variable, rough particle, and different chaotic maps have been provided. Application of CRPSO in data mining has also been performed.

2. Rough Particle Swarm Optimization (RPSO)

Objects, instances, or records can be described by a finite set of attributes. The description of an object is an *n*-dimensional vector, where *n* is the number of attributes that characterizes an object. A pattern is a class of objects based on the values of some attributes of objects belonging to the class.

Let *x* be an attribute in the description of an object and $\underline{x}, \overline{x}$ represent lower and upper bounds (endpoints) of *x* such that $\underline{x} \leq \overline{x}$. A rough pattern value of each attribute variable consists of lower and upper bounds and can be presented as Eq. (1). It can be diagrammatically seen in Figure 5. It is as a closed, compact, and bounded subset of the set of real numbers R.

$$x = \left(\underline{x}, \overline{x}\right) \tag{1}$$

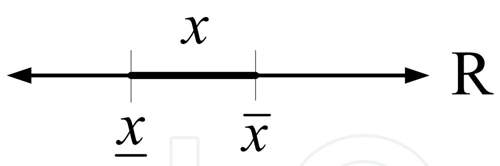

Figure 1. A rough value

If $0 \leq \underline{x}$ the rough value is called a *positive rough value*, and we write $x > 0$. Conversely, if $\overline{x} \leq 0$ we call the rough value a *negative rough value*, and write $x < 0$. Positive or negative rough values are the two types of *sign coherent rough values*. If $\underline{x} = 0$ or $\overline{x} = 0$ we call the rough value a *zero-bound rough value.* A zero-bound positive rough value is called a *zero-positive rough value.* Similarly, a zero-bound negative rough value is called a *zero-negative rough value.* A rough value that has both positive and negative values is called a *zero-straddling rough value*. These definitions are summed up in Table 1.

Definition	**Condition**
positive rough value (x>0)	Iff $\underline{x}$ >0
negative rough value (x<0)	Iff $\overline{x}$ <0
zero-positive rough value (x$\geq$0)	Iff $\underline{x}$ =0
zero-negative rough value (x$\leq$0)	Iff $\overline{x}$ =0
zero-straddling rough value (x<>0)	Iff $\overline{x}$ >0 and $\underline{x}$ <0

Table 1. Definitions on rough values

The *midpoint (mid), radius (rad)*, and *width* of a rough value *x* are defined as:

$$mid(x) = (\overline{x} + \underline{x})/2 \tag{2}$$

$$rad(x) = (\overline{x} - \underline{x})/2 \tag{3}$$

$$width(x) = (\overline{x} - \underline{x}) = 2rad(x) \tag{4}$$

Since *x* = (*mid*(*x*)-*rad*(*x*), *mid*(*x*)+*rad*(*x*)) rough values can also be represented in terms of midpoint and radius instead of endpoints.
Rough values are useful in representing an interval or set of values for an attribute, where only lower and upper bounds are considered relevant in a computation. It may be very popular for many areas of computational mathematics. For example, by computing with rough values, it is possible (with some error) to evaluate a function over an entire interval rather than a single value. In other words, if we evaluate a function *f*(*x*) over some interval of *x* (e.g. $x\in$ $(\underline{x},\overline{x})$), we know what the possibly overestimated bounds of the function are within that interval. Since working with rough values always produces exact or overestimated bounds, it cannot miss a critical value in a function. Therefore, it is very useful for robust root finding, global maximum/minimum finding, and other optimization problems.
In fact, a conventional pattern can be easily represented as a rough pattern by using both lower and upper bounds to be equal to the value of the variable. Some operations on rough values can be implemented as:

$$x + y = (\underline{x},\overline{x}) + (\underline{y},\overline{y}) = (\underline{x}+\underline{y}, \overline{x}+\overline{y}) \tag{5}$$

$$x - y = (\underline{x},\overline{x}) + (-(\underline{y},\overline{y})) = (\underline{x}-\overline{y}, \overline{x}-\underline{y}) \tag{6}$$

$$x \times y = (\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})) \tag{7}$$

$$\frac{1}{x} = \left(\frac{1}{\overline{x}}, \frac{1}{\underline{x}}\right), \quad 0 \notin (\underline{x},\overline{x}) \tag{8}$$

$$\frac{x}{y} = \frac{(\underline{x},\overline{x})}{(\underline{y},\overline{y})} = \frac{(\underline{x},\overline{x})}{\left(\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right)}, \quad 0 \notin (\underline{y},\overline{y}) \tag{9}$$

$$c \times x = c \times (\underline{x}, \overline{x}) = (\underline{x}, \overline{x}) \times c = \begin{cases} (c \times \underline{x}, c \times \overline{x}) & if \quad c \geq 0 \\ (c \times \overline{x}, c \times \underline{x}) & if \quad c < 0 \end{cases} \tag{10}$$

In fact, these operations are borrowed from the conventional interval calculus (Lingras & Davies, 1999).
The algebraic properties of addition and multiplication operations on rough values have been described in Table 2.

Algebraic properties	**Description**	**Condition**
Commutativity	$x+y=y+x$	No condition
	$xy=yx$	
Associativity	$(x+y)+z=x+(y+z)$	No condition
	$(xy)z=x(yz)$	
Neutral Element	$0+x=x$	No condition
	$1.x=x$	
Distributivity	$x(y+z)=xy+xz$	If $\underline{x} = \overline{x}$
	$x(y+z)=xy+xz$	If $y \geq 0$ and $z \geq 0$ (non-negative terms)
	$x(y+z)=xy+xz$	If $y \leq 0$ and $z \leq 0$ (non-positive terms)
	$x(y+z)=xy+xz$	If $x \geq 0$, $\underline{y}=0$ and $\overline{z}=0$ (positive factor, zero-straddling terms)
	$x(y+z)=xy+xz$	If $x \leq 0$, $\underline{y}=0$ and $\overline{z}=0$ (negative factor, zero-straddling terms)
	$x(y-z)=xy-xz$	If $y \geq 0$ and $z \leq 0$ (non-negative terms variation)
	$x(y-z)=xy-xz$	If $y \leq 0$ and $z \geq 0$ (non-positive terms variation)

Table 2. Algebraic properties

A rough particle r is string of rough parameters r_i:

$$r = (r_i \mid 1 \leq i \leq n) \tag{11}$$

A rough parameter r_i is a pair of conventional parameters, one for lower bound called lower parameter ($\underline{r_i}$) and the other for upper bound called upper parameter ($\overline{r_i}$):

$$r_i = \left(\underline{r_i}, \overline{r_i}\right) \tag{12}$$

Figure 2 shows examples of rough particles.

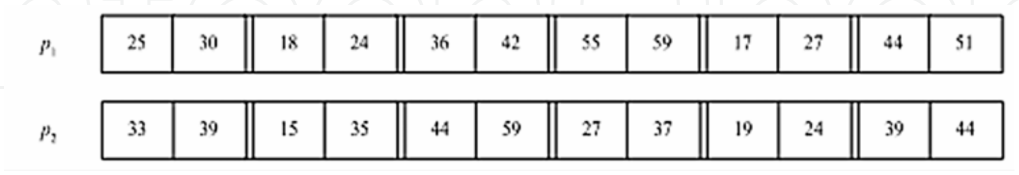

Figure 2. Rough particles

The value of each rough parameter is the range for that variable. The use of range shows that the information represented by a rough particle is not precise. Hence, an information measure called *precision* may be useful when evaluating the fitness levels (Lingras, 1996; Lingras & Davies, 2001).

$$precision(r) = -\sum_{1\le i\le n}\left(\frac{\overline{r_i} - \underline{r_i}}{Range_{\max}(r_i)}\right) \tag{13}$$

Here, $Range_{\max}(r_i)$ is the length of maximum allowable range for the value of rough parameter r_i.
The conventional parameters and particles used in PSO algorithms are special cases of their rough equivalents as shown in Figure 3. For a conventional particle *p*, *precision(p)* has the maximum possible value of zero.

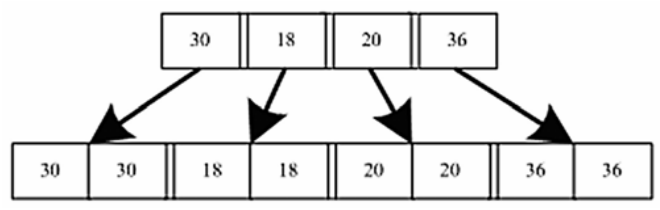

Figure 3. Conventional particle and its rough equivalent

In boundary constraint problems, it is essential to ensure that values of decision variables lie inside their allowed ranges after velocity or position update equations. This technique can also be generalized for RPSO algorithm. Constraint that the lower bounds in rough variables should be less than the upper bounds is already satisfied with RPSO algorithm.

3. Chaotic Particle Swarm Optimization (CPSO)

Generating random sequences with a long period and good uniformity is very important for easily simulating complex phenomena, sampling, numerical analysis, decision making and especially in heuristic optimization. Its quality determines the reduction of storage and computation time to achieve a desired accuracy. Generated such sequences may be "random" enough for one application however may not be random enough for another.

Chaos is a deterministic, random-like process found in non-linear, dynamical system, which is non-period, non-converging and bounded. Moreover, it has a very sensitive dependence upon its initial condition and parameter (Schuster, 1998)]. The nature of chaos is apparently random and unpredictable and it also possesses an element of regularity. Mathematically, chaos is randomness of a simple deterministic dynamical system and chaotic system may be considered as sources of randomness.
A chaotic map is a discrete-time dynamical system

$$x_{k+1} = f(x_k), \quad 0 < x_k < 1, \ k = 0, \ 1, \ 2, \cdots \tag{14}$$

running in chaotic state. The chaotic sequence $\{x_k : k = 0, 1, 2, \dots\}$ can be used as spread-spectrum sequence as random number sequence.
Chaotic sequences have been proven easy and fast to generate and store, there is no need for storage of long sequences (Heidari-Bateni & McGillem, 1994). Merely a few functions (chaotic maps) and few parameters (initial conditions) are needed even for very long sequences. In addition, an enormous number of different sequences can be generated simply by changing its initial condition. Moreover these sequences are deterministic and reproducible.
Recently, chaotic sequences have been adopted instead of random sequences and very interesting and somewhat good results have been shown in many applications such as secure transmission (Wong et al., 2005; Suneel, 2006), and nonlinear circuits (Arena et al., 2000), DNA computing (Manganaro & Pineda, 1997), image processing (Gao et al., 2006). The choice of chaotic sequences is justified theoretically by their unpredictability, i.e., by their spread-spectrum characteristic and ergodic properties.
One of the major drawbacks of the PSO is its premature convergence, especially while handling problems with more local optima. In this paper, sequences generated from chaotic systems substitute random numbers for the PSO parameters where it is necessary to make a random-based choice. By this way, it is intended to improve the global convergence and to prevent to stick on a local solution. For example, the value of *inertia weight* is the key factors to affect the convergence of PSO. Furthermore the values of random numbers that affect the stochastic nature are also key factors that affect the convergence of PSO. In fact, however, these parameters can’t ensure the optimization’s ergodicity entirely in phase space, because they are random in traditional PSO.
New approaches introducing chaotic maps with ergodicity, irregularity and the stochastic property in PSO to improve the global convergence by escaping the local solutions have been provided. The use of chaotic sequences in PSO can be helpful to escape more easily from local minima than can be done through the traditional PSO. When a random number is needed by the classical PSO algorithm it is generated by iterating one step of the chosen chaotic map that has been started from a random initial condition at the first iteration of the PSO. New chaos embedded PSO algorithms may be simply classified and described as Table 3. In this table first column represents the name of PSO. The second column represents which values it effect to. And the last column, divided in to three sub columns, represents the bounds of the values they can take from the selected chaotic maps. For example CPSO3 only effects to second acceleration coefficient (c_2) and the values taken from the selected chaotic map is scaled between 0.5 and 2.5. When rough representation is used these names take a “R” for representing the “Rough” after “C” that represents “Chaotic”. Namely when rough representation is used for “CPSO1” it is named as “CRPSO1”.

Name	**Effect**	**Scaled Values of Chaotic Maps**		
CPSO1	Initial velocities and position	Lower bound - upper bound of each decision variable		
CPSO2	c_1	0.5 – 2.5	-	-
CPSO3	c_2	0.5 – 2.5	-	-
CPSO4	c_1 and c_2	0.5 – 2.5	0.5 – 2.5	-
CPSO5	r_1	0.0 – 1.0	-	-
CPSO6	r_2	0.0 – 1.0	-	-
CPSO7	r_1 and r_2	0.0 – 1.0	0.0 – 1.0	-
CPSO8	w, r_1, and r_2	0.0 – 1.0	0.0 – 1.0	0.0 – 1.0
CPSO9	w	0.0 – 1.0	-	-
CPSO10	w and c_1	0.0 – 1.0	0.5 – 2.5	-
CPSO11	w and c_2	0.0 – 1.0	0.5 – 2.5	-
CPSO12	w, c_1, and c_2	0.0 – 1.0	0.5 – 2.5	0.5 – 2.5

Table 3. Characteristics of CPSO algorithms

Note that CPSO1 can be used together with the other CPSO classes. The chaotic maps that generate chaotic sequences in PSO phases used in the experiments are listed below.

Logistic Map: One of the simplest maps which was brought to the attention of scientists by Sir Robert May in 1976 that appears in nonlinear dynamics of biological population evidencing chaotic behavior is logistic map, whose equation is the following (May, 1976):

$$X_{n+1} = aX_n(1 - X_n) \tag{15}$$

In this equation, X_n is the n-th chaotic number where n denotes the iteration number. Obviously, $X_n \in (0, 1)$ under the conditions that the initial $X_0 \in (0, 1)$ and that $X_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$. a=4 have been used in the experiments.

Sinusoidal Iterator: The second chaotic sequence generator used in this paper is the so-called sinusoidal iterator (Peitgen et al., 1992) and it is represented by

$$X_{n+1} = ax_n^2 \sin(\pi x_n) \tag{16}$$

When a=2.3 and X_0=0.7 it has the simplified form represented by

$$X_{n+1} = \sin(\pi x_n) \tag{17}$$

It generates chaotic sequence in (0, 1)

Gauss Map: The Gauss map is used for testing purpose in the literature (Peitgen et al., 1992) and is represented by:

$$X_{n+1} = \begin{cases} 0 & , X_n = 0 \\ 1/X_n \bmod(1) & , X_n \in (0,1) \end{cases} \tag{18}$$

$$1/X_n \bmod(1) = \frac{1}{X_n} - \left\lfloor \frac{1}{X_n} \right\rfloor \tag{19}$$

and $\lfloor z \rfloor$ denotes the largest integer less than z and acts as a shift on the continued fraction representation of numbers. This map also generates chaotic sequences in (0, 1).

Zaslavskii Map: The Zaslavskii Map (Zaslavskii, 1978) is an also an interesting dynamic system and is represented by:

$$X_{n+1} = \left(X_n + v + aY_{n+1}\right)\bmod(1) \tag{20}$$

$$Y_{n+1} = \cos\left(2\pi X_n\right) + e^{-r} Y_n \tag{21}$$

Its unpredictability by its spread-spectrum characteristic and its large Lyapunov exponent are theoretically justified. The Zaslavskii map shows a strange attractor with largest Lyapunov exponent for v=400, r=3, a=12. Note that in this case, $Y_{n+1} \in [-1.0512, 1.0512]$.

4. CRPSO in Data Mining

CRPSO has been used for mining numeric association rules (ARs) from databases in which records concerned are categorical or numeric. In a numeric AR, attributes are not limited to being Boolean but can be numeric (e.g. age, salary, and heat) or categorical (e.g. sex, brand). Thus, numeric ARs are more expressive and informative than Boolean ARs (Ke et al., 2006). An example of a numeric AR in an employee database is:

"Age $\in$ [25, 36] $\wedge$ Sex=Male $\Rightarrow$ Salary $\in$ [2000-2400] $\wedge$ Have_Car=Yes"
(Support = 4%, Confidence = 80%).

In this numeric AR, "Age $\in$ [25, 36] $\wedge$ Sex=Male" is antecedent and "Salary $\in$ [2000-2400] $\wedge$ Have_Car=Yes" is consequent part. This numeric AR states that "4% (support) of the employees are males aged between 25 and 36 and earning a salary of between \$2.000 and \$2.400 and have a car", while "80 % (confidence) of males aged between 25 and 36 are earning a salary of between \$2.000 and \$2.400 and have a car".
Following subsections are description of CRPSO for mining numeric ARs.

4.1 Particle representation

In this work, the particles which are being produced and modified along the search process represent rules. Each particle consists of decision variables which represent the items and intervals. A positional encoding, where the *i*-th item is encoded in the *i*-th decision variable has been used. Each decision variable has three parts. The first part of each decision variable represents the antecedent or consequent of the rule and can take three values: '0', '1' or '2'. If the first part of the decision variable is '0', it means that this item will be in the antecedent of the rule and if it is '1', this item will be in the consequent of the rule. If it is '2', it means that this item will not be involved in the rule. All decision variables which have '0' on their first parts will form the antecedent of the rule while decision variables which have '1' on their first part will form the consequent of the rule. While the second part represents the lower bound, the third part represents the upper bound of the item interval. The structure of a particle has been illustrated in Figure 4, where *m* is the number of attributes of data being mined (Alatas et al., 2007).

Variable$_1$			***Variable$_2$***			**...**			***Variable$_m$***		
AC$_1$	*LB$_1$*	*UB$_1$*	*AC$_2$*	*LB$_2$*	*UB$_2$*				*AC$_m$*	*LB$_m$*	*UB$_m$*

Figure 4. Particle representation

Rounding operator that converts a continuous value to an integer value for the first parts of this representation by truncation is performed when evaluating. Rounded variables are not elsewhere assigned in order to let CRPSO algorithm work with a swarm of continuous variables regardless of the object variable type for maintaining the diversity of the swarm and the robustness of the algorithm.
In the implementation of this particle representation, the second and third part of decision variables will be considered as one value, namely rough value. At first glance, this representation seems to appropriate for only numeric attributes. However it is very straightforward to extend it for discrete, nominal, and numeric attributes. The numeric attributes locates at the beginning of the representation and discrete ones at the end. For discrete attributes only AC_i and V_i where V_i is the value of the attribute are used. Namely, instead of LB_i and UB_i, only Vi is used for values of discrete or nominal attributes.

4.2 Fitness Function

The mined rules have to acquire large support and confidence. CRPSO has been designed to find the intervals in each of the attributes that conform an interesting rule, in such a way that the fitness function itself is the one that decides the amplitude of the intervals. That is why, the fitness value has to appropriately shelter these and it has been shown in Eq. (22).

$$\textbf{\textit{Fitness}} = \alpha_1 \times cover\ (Ant{+}Cons) + \alpha_2 \times \frac{cover(Ant + Cons)}{cover(Ant)} + \alpha_3 \times (NA) \text{-}\ \alpha_4 \times Int\text{-}\ \alpha_5 \times marked \tag{22}$$

This fitness function has four parts. Here, *Ant* and *Cons* are distinct itemsets that are involved in the antecedent and consequent part of the rule respectively. *cover (Ant+Cons)* is ratio of the records that contain *Ant+Cons* to the total number of records in the database. The first part can be considered as support of the rule that is statistical significance of an AR. In fact, the second part can be considered as confidence value. The third part is used for number of attributes in the particle. *NA* is number of attributes in the database that has not '2' in first parts of decision variable of particles. The motivation behind this term is to bias the system to give more quality information to the final user. The last part of the fitness is used to penalize the amplitude of the intervals that conform the itemset and rule. In this way, between two particles that cover the same number of records and have the same number of attributes, the one whose intervals are smaller gives the best information. *Int* has been computed as shown in Eq. (23) where amp_m is the amplitude factor determined for each attribute for balancing the effect of *Int* to the fitness.

$$\frac{UB_m - LB_m}{amp_m} \tag{23}$$

marked is used to indicate that an attribute of a records has previously been covered by a rule. Algorithm is forced to mine different rules in later searches by this way.
α_1, α_2, $\alpha_3, \alpha_4, \alpha_5$ are user specified parameters and one might increase or decrease the effects of parts of fitness function by means of these parameters. *Int* part of the fitness calculation concerns particles parts representing numeric attributes.

4.3 Mutation

Mutation has also been performed in this study. Mutation operator is introduced which mutates a single decision vector of a particle with probability p_{mut}. Four possible mutations (Mata et al., 2002) have been used for CRPSO algorithms:

- **Shifting the whole interval towards the right**: The values in lower and upper bounds are increased.
- **Shifting the whole interval towards the left**: The values in lower and upper bounds are decreased.
- **Incrementing the interval size**: The value of lower bound is decreased and the value of upper bound is increased.
- **Reducing the interval size**: The value of lower bound is increased and the value of upper bound is decreased.

When a particle is chosen to be mutated each decision value is then mutated by one of this four mutation types or not with probability $1/m$, where m is the number of decision value in the particle. Particle positions are updated only if the mutated particles have better fitness.

4.4 Refinement of bound intervals

At the end of the CRPSO search, a refinement in the attributes bounds that belong to the covered rule is performed. This refinement process consists reducing the interval size until the support value is smaller than the support of the original rule encoded in the related particle.

4.5. Parameter Control

The used parameter values for the experiments have been shown in Table 4. Minimum and maximum values for velocity and position depend on the bounds of the decision values. α_1, α_2, α_3, α_4, and α_5 that have been used in fitness values were selected as 0.8, 0.8, 0.05, 0.1, and 0.2 respectively.

Parameters	**Swarm size**	**No. of generations**	**Mutation Probability**
Values	20	1000	0.5

Table 4. Used parameters for PSO algorithms

5. Experimental Results

Synthetic database is created using the function 2 (Agrawal et al., 1993) to distribute the values in records in such a way that they are grouped in pre determined sets. The function is shown in Figure 5. The goal is to most accurately find the intervals of each of the created regions. Namely, a group is assigned to each record, depending on the values that the attributes *age* and *salary* take. Figure 6 shows a graphic representation of the distribution of 5000 records according to the function where only records belong to Group A are presented.

$$\begin{gathered}\text{If } ((age < 40) \wedge (50\text{K} \leq salary \leq 100\text{K})) \vee \\ ((40 \leq age < 60) \wedge (75\text{K} \leq salary \leq 125\text{K})) \vee \\ ((age \geq 60) \wedge (25\text{K} \leq salary \leq 75\text{K})) \Rightarrow \text{Group } A \\ \text{else} \Rightarrow \text{Group } B\end{gathered}$$

Figure 5. Function used for the experiment

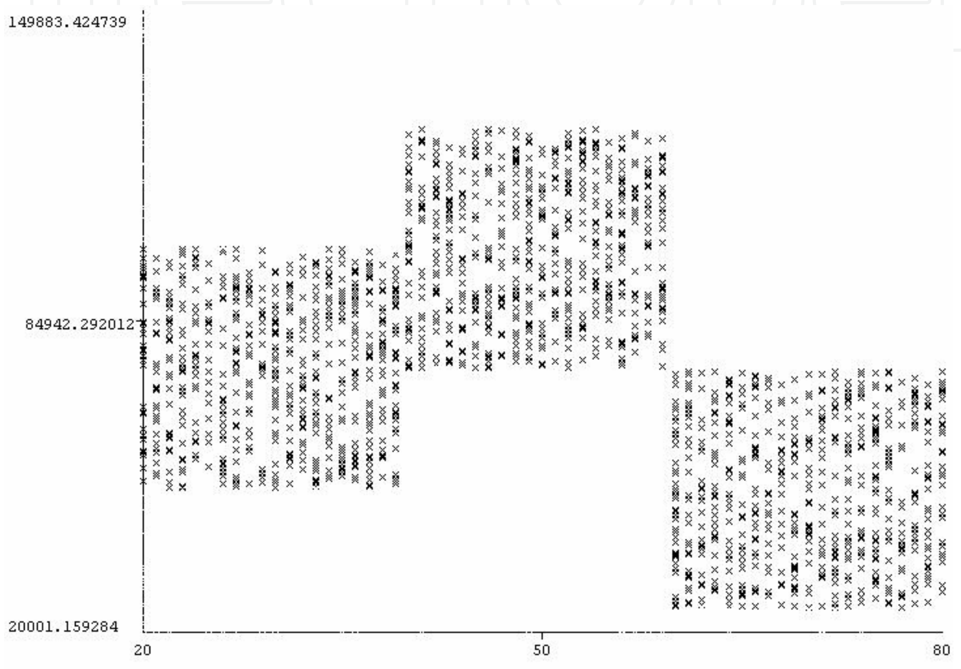

Figure 6. Graphic representation of the function for the experiment

The database has been generated by uniformly distributing the records between the lower and upper values of its domains. For attribute *salary* the extreme values are from 20000 to 150000 and for attribute *age* salary the extreme values are from 20 to 80. The third attribute for the group has also been added. According to the function almost 37.9% of the records belong to Group *A*. the ARs are those that have the attrşbutes salary and age in the antecedent and the attribute Group in the consequent. That is why, representation of the particle respects to this case.

For a fair comparison of the results initial swarm is initialized in a different way. A same record of the database is chosen and the rule is generated departing from it, defining for each value of v_i of the selected attribute a_i, lower limit $v_i - \theta$ and upper limit $v_i + \theta$. θ is a percentage of the value v_i. In this way the swarm is conditioned to cover at least one record. This has not been performed for CRPSO1.

An intuitive measure to verify the efficacy of the CRPSO algorithms, verifying that the mined ARs have larger quality, consists of checking that the intervals of the rules accord to the ones synthetically generated. Mean support and confidence values of mined rules from rough PSO algorithm are 12.21 and 93.33. Acceleration coefficients have been selected as 2 and inertia weight has been gradually decreased from 0.9 to 0.4 for CRPSO1 algorithm.

The obtained results from the proposed CRPSO algorithms are shown in Tables 5. All of the algorithms have mined three rules and mean support and confidence values of these rules have been depicted in this table. If the mean support values are multiplied by 3, values close to 37.9% may be found, which means that the rules have practically covered all the records. Confidence values are also close to 100%, since in the regions there are no records of other groups. Interesting result is that, CRPSO7, CRPSO8, and CRPSO12 have the best performances. CRPSO7 using Zaslavskii map seems the best PSO among others. The mined rules from CRPSO7 using Zaslavskii map is shown in Table 6.

Logistic Map												
	CRPSO1	CRPSO2	CRPSO3	CRPSO4	CRPSO5	CRPSO6	CRPSO7	CRPSO8	CRPSO9	CRPSO 10	CRPSO 11	CRPSO 12
Sup (%)	11.24	10.64	10.96	11.98	10.78	10.80	12.48	12.28	10.82	10.66	11.02	11.54
Conf (%)	98.14	98.12	97.01	97.64	97.42	97.96	99.01	98.98	98.42	98.24	98.56	99.08
Sinusoidal Iterator												
	CRPSO1	CRPSO2	CRPSO3	CRPSO4	CRPSO5	CRPSO6	CRPSO7	CRPSO8	CRPSO9	CRPSO 10	CRPSO 11	CRPSO 12
Sup (%)	11.20	11.24	10.86	11.62	10.86	10.96	12.38	12.28	10.58	10.90	11.06	11.24
Conf (%)	98.48	98.46	98.08	98.06	97.56	97.02	99.08	98.96	98.54	97.98	99.02	98.96
Gauss Map												
	CRPSO1	CRPSO2	CRPSO3	CRPSO4	CRPSO5	CRPSO6	CRPSO7	CRPSO8	CRPSO9	CRPSO 10	CRPSO 11	CRPSO 12
Sup (%)	11.91	11.05	10.86	10.24	10.84	10.97	12.28	12.23	10.81	10.23	9.99	11.98
Conf (%)	97.22	97.62	98.68	98.16	98.26	97.64	99.04	99.06	98.48	97.86	97.56	98.62
Zaslavskii Map												
	CRPSO1	CRPSO2	CRPSO3	CRPSO4	CRPSO5	CRPSO6	CRPSO7	CRPSO8	CRPSO9	CRPSO 10	CRPSO 11	CRPSO 12
Sup (%)	11.05	11.26	10.24	10.22	10.97	10.96	12.62	12.41	10.96	10.82	10.88	11.94
Conf (%)	97.56	97.55	97.22	98.08	98.48	98.69	99.65	99.12	98.62	98.86	97.16	98.84

Table 5. Mean support and confidence values of the rules mined by different CRPSO algorithms

Rule	Sup(%)	Conf(%)
If $age \in [20, 40] \wedge salary \in [50136, 99869] \Rightarrow$ Group A	12.63	98.94
If $age \in [41, 59] \wedge salary \in [76779, 12469] \Rightarrow$ Group A	12.62	100
If $age \in [61, 80] \wedge salary \in [25440, 73998] \Rightarrow$ Group A	12.61	100

Table 6. ARs mined by CRPSO7 using Zaslavskii map

6. Conclusions

In this chapter chaotic rough PSO, CRPSO, algorithms that use rough decision variables and rough particles that are based on notion of rough patterns have been proposed. Different chaotic maps have been embedded to adapt the parameters of PSO algorithm. This has been done by using of chaotic number generators each time a random number is needed by the classical PSO algorithm. Twelve PSO methods have been proposed and four chaotic maps have been analyzed in the data mining application. It has been detected that coupling emergent results in different areas, like those of PSO and complex dynamics, can improve the quality of results in some optimization problems and also that chaos may be a desired process. It has been also shown that, these methods have somewhat increased the solution quality, that is in some cases they improved the global searching capability by escaping the local solutions. The proposed CRPSO algorithms can complement the existing tools developed in rough computing using chaos. These proposed methods seem to provide useful extensions for practical applications. More elaborated experiments by using optimized parameters may be performed with parallel or distributed implementation of these methods.

7. Acknowledgements

This work is supported by Firat University Scientific Research and Projects Unit (Grant. No. 1251) and is supported in part by a grant from Scientific and Technological Research Council of Turkey, National Scholarship Program for PhD Students.

8. References

Agrawal, R., Imielinski, T., Swami, A. (1993). Database Mining: A Performance Perspective, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, p.914-925

Alatas, B., Akin E., Karci, A. (2007). MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules, *Applied Soft Computing*, Elsevier, http://dx.doi.org/10.1016/j.asoc.2007.05.003

Arena P., Caponetto R., Fortuna L., Rizzo A., La Rosa M. (2000). Self Organization in non Recurrent Complex System, *Int. J. Bifurcation and Chaos*, Vol. 10, No. 5, 1115–1125

Gao H., Zhang Y., Liang S., Li D. (2006). A New Chaotic Algorithm for Image Encryption, *Chaos, Solitons and Fractals*, 29, 393-399

Heidari-Bateni G. & McGillem C. D. (1994). A Chaotic Direct-Sequence Spread Spectrum Communication System, *IEEE Trans. on Communications*, Vol. 42 No. (2/3/4), 1524-1527

Ke, K., Cheng, J. Ng, W. (2006). MIC Framework: An Information-Theoretic Approach to Quantitative Association Rule Mining, *ICDE* '06, 112-114.

Lingras, P. (1996). Rough Neural Networks, *International Conference on Information Processing and Management of Uncertainty*, Granada, Spain, pp. 1445-1450

Lingras, P. & Davies, C. (1999). Rough Genetic Algorithms, 7th International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing RSFDGrC 1999, *Lecture Notes In Computer Science;* Vol. 1711, 38-46

Lingras, P. & Davies, C. (2001). Applications of Rough Genetic Algorithms, *Computational Intelligence: An International Journal*, Vol . 3, No. 17, 435-445

Manganaro G. & Pineda de Gyvez J. (1997). DNA Computing Based on Chaos, *IEEE International Conference on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 255–260

Mata, J., Alvarez, J., Riquelme, J. (2002). Discovering Numeric Association Rules via Evolutionary Algorithm, *Lecture Notes in Computer Science*, Volume 2336, Springer Verlag, 40-51.

May R. M. (1976). Simple Mathematical Models with very Complicated Dynamics. *Nature* 261: 459

Peitgen H., Jurgens H., Saupe D. (2004). *Chaos and Fractals*, Springer-Verlag, 0387202293, Berlin, Germany

Schuster H. G. (1995). *Deterministic Chaos: An Introduction,* Physick- Verlag GmnH, John Wiley & Sons Ltd, 3527293159, Federal Republic of Germany

Suneel M. (2006). Chaotic Sequences for Secure CDMA, *Ramanujan Institute for Advanced Study in Mathematics*, 1-4

Wong K., Man K. P., Li S., & Liao X. (2005). More Secure Chaotic Cryptographic Scheme based on Dynamic Look-up Table, *Circuits, Systems & Signal Processing*, Vol. 24, No. 5, 571-584

Zaslavskii G. M. (1978). The Simplest Case of a Strange Attractor, *Physic Letters A*, Vol. 69, 145-147