УДК 004.8

МЕТАЭВРИСТИЧЕСКИЕ МЕТОДЫ ОПТИМИЗАЦИИ ПАРАМЕТРОВ НЕЧЕТКИХ КЛАССИФИКАТОРОВ

Ходашинский Илья Александрович
Д.т.н., профессор, ТУСУР, e-mail: hodashn@rambler.ru
Анфилофьев Александр Евгеньевич
Аспирант, ТУСУР, e-mail: yowwi00@gmail.com
Бардамова Марина Борисовна
Студентка, ТУСУР, e-mail: 722bmb@gmail.com
Ковалев Виталий Сергеевич
Студент, ТУСУР, e-mail: vitaly_979@mail.ru
Мех Максим Анатольевич
Студент, ТУСУР, e-mail: maxcimkj93@gmail.com
Сонич Ольга Константиновна
Студентка, ТУСУР,e-mail: zlasjasok@gmail.com
Томский государственный университет систем управления и радиоэлектроники (ТУСУР), 634050 г. Томск, пр. Ленина 40

Аннотация. Представлены следующие алгоритмы оптимизации параметров нечетких систем: алгоритм сорняков, гравитационный алгоритм, алгоритм прыгающих лягушек, гармонический поиск, дифференциальная эволюция, алгоритм «Всемирный потоп». С помощью указанных алгоритмов построены на реальные данные из репозитария KEEL.
Ключевые слова: нечеткие системы, оптимизация параметров, метаэвристики

Введение. Нечеткие системы находят широко применение в различных областях деятельности: автоматическом управлении, распознавании образов, принятии решений. Важным этапом при построении нечетких систем является задача определения оптимальных параметров таких систем. В качестве формального аппарата решения указанной задачи часто используются метаэвристические методы [5].

Целью статьи является анализ эффективности применения различных метаэвристических методов при оптимизации параметров нечетких классификаторов.

1. Постановка задачи. Нечеткий классификатор задается правилами вида [4]:

R_{ji} : ЕСЛИ $x_1=A_{i1}$ И $x_2=A_{i2}$ И $x_3=A_{i3}$ И … И $x_n=A_{in}$ ТО class=c_j,

где **x** – вектор признаков классифицируемого объекта; c_j – идентификатор j-того класса, $j\in [1, m]$, A_{ik} – нечеткий терм, характеризующий k-ый признак в ji-ом правиле R_{ji} ($i \in [1, |Rj|]$, $j\in [1, m]$), R_j – множество правил, относящих наблюдение к классу с идентификатором c_j.

В процессе нечеткой классификации объект относится к каждому классу с определенной степенью принадлежности, вычисленной следующим образом:

$$\beta_j(\mathbf{x}) = \sum_{R_{ji}} \prod_{k=1}^{n} A_{ik}(x_k), \quad j = 1,2,\ldots,m.$$

Выходом классификатора является метка класса, определяемая следующим образом:

$$\text{class} = c_{j^*},\ j^* = \arg\max_{1 \le j \le m} \beta_j .$$

Нечеткий классификатор может быть представлен функцией $c = f(\mathbf{x}, \boldsymbol{\theta})$, где $\boldsymbol{\theta}$ – вектор, описывающий базу правил.

На множестве обучающих данных (таблице наблюдений) $\{(\mathbf{x}_p;\ c_p),\ p = 1,\ldots,\ z\}$ определим единичную функцию

$$delta(p, \boldsymbol{\theta}) = \begin{cases} 0, & \text{если } c_p = f(c_p, \boldsymbol{\theta}) \\ 1, & \text{иначе} \end{cases}, p = 1,2,\ldots, z,$$

тогда численный критерий ошибки классификации выражается следующим образом:

$$E(\boldsymbol{\theta}) = \frac{\sum_{p=1}^{z} delta(p, \boldsymbol{\theta})}{z}.$$

Целью построения нечётких систем является поиск таких параметров этих систем, которые сводят к минимуму ошибку $E(\boldsymbol{\theta})$. Минимизация выполняется с помощью представленных ниже метаэвристических методов.

2. Гармонический поиск (HS). Алгоритм основан на принципе создания музыкальных фраз. Новая музыкальная фраза может быть случайной или производной от сыгранной ранее [2].

Алгоритм имеет следующие параметры: r_{accept}, $r_{pa} \in [0,1]$ – используются при создании нового вектора, S – размер популяции, N – количество итераций. Первым этапом выполнения является инициализация популяции векторов вещественных значений $\boldsymbol{\theta}$.

Принцип оптимизации алгоритма заключается в генерации на каждой итерации нового вектора $\boldsymbol{\theta}_{new}$ на основе случайно выбранного вектора из популяции $\boldsymbol{\theta}_r$. Сгенерированный новый вектор оценивается оптимизируемой фитнес-функцией $E(\boldsymbol{\theta})$, полученное значение сравнивается со значением фитнес-функции худшего вектора в популяции $\boldsymbol{\theta}_{worst}$ и происходит замена худшего вектора на новый, если значение фитнес-функции нового вектора меньше. По завершению выполнения N итераций, алгоритм возвращает лучший по значению фитнес-функции вектор в популяции $\boldsymbol{\theta}_{best}$. Псевдокод алгоритма приведен ниже.

Вход: S, N, r_{accept}, r_{pa}.
Выход: $\boldsymbol{\theta}_{best}$.
Popul := $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S\}$;
цикл пока ($N > 0$)
    $\boldsymbol{\theta}_r$:= Random_choose(*Popul*)
    цикл по d от 1 до $|\boldsymbol{\theta}_i|$
        если (rand(0,1) < r_{accept}) то
            $\boldsymbol{\theta}_{new}[d] := \boldsymbol{\theta}_r[d]$
            если (rand(0,1) < r_{pa}) то
                $\boldsymbol{\theta}_{new}[d] := \boldsymbol{\theta}_{new}[d] \pm \text{rand}(0,1)\cdot(\text{Max}(\boldsymbol{\theta}_{new}[d]) - \text{Min}(\boldsymbol{\theta}_{new}[d]);$
        иначе $\boldsymbol{\theta}_{new}[d] := \text{rand}(0,1)\cdot(\text{Max}(\boldsymbol{\theta}_{new}[d]) - \text{Min}(\boldsymbol{\theta}_{new}[d]);$
        конец цикла;
        если ($E(\boldsymbol{\theta}_{new}) > E(\boldsymbol{\theta}_{worst})$) то

$\boldsymbol{\theta}_{worst} := \boldsymbol{\theta}_{new}$;

$N := N - 1$;

конец цикла;

вывод $\boldsymbol{\theta}_{best}$:= Search_best(*Popul*).

3. Гравитационный алгоритм (GSA) основан на фундаментальных законах тяготения. Популяция представляет собой систему частиц, между которыми действуют силы притяжения [7]. На вход алгоритма подаются следующие параметры: количество частиц N, количество итераций T, начальное значение гравитационной постоянной G_0, коэффициент точности поиска α, малая константа ε. Значение гравитационной постоянной рассчитывается на каждой итерации на основе монотонно убывающей функции. Для каждой i-ой частицы из популяции $\boldsymbol{\theta}$ на t-ой итерации рассчитываются следующие физические характеристики: $m_i(t)$ – масса, $a_i(t)$ – ускорение, $V_i(t)$ – скорость. На последнем шаге происходит обновление позиции частицы путем изменения текущих координат на величину, пропорциональную скорости. Расчеты проводятся до истечения итераций T, затем на выход подается вектор с наименьшим значением ошибки $\boldsymbol{\theta}_{best}$. Ниже приведен псевдокод алгоритма.

Вход: N, S, G_0, α, ε.

Выход: значение $\boldsymbol{\theta}_{best}$.

Population = $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S\}$;

цикл пока ($N > 0$)

цикл по i от 0 до S

$m[i] := (E[\boldsymbol{\theta}_i] - E[\boldsymbol{\theta}_{worst}]) / (E[\boldsymbol{\theta}_{best}] - E[\boldsymbol{\theta}_{worst}])$;

цикл по j от 0 до S

$R[i, j] := \left\| \boldsymbol{\theta}_j - \boldsymbol{\theta}_i \right\|$;

цикл по d от 1 до $|\boldsymbol{\theta}_i|$

$a_i^d := a_i^d + rand * M[j] * (\theta_j^d - \theta_i^d) / (R[i, j] + \varepsilon)$;

$V_i^d[t+1] := rand * V_i^d[t] + a_i^d[t]$;

$\theta_i^d[t+1] := \theta_i^d[t] + V_i^d[t+1]$;

конец цикла

конец цикла

конец цикла

$N := N - 1$;

конец цикла

вывод $\boldsymbol{\theta}_{best}$:= Search_best(*Population).*

4. Алгоритм прыгающих лягушек (FLA) имитирует поведение группы лягушек в процессе поиска пищи. Основой алгоритма является комбинирование локального поиска в пределах каждого из мемплексов (группы) и глобального поиска путем обмена информацией о положениие лучших агентов этих мемплексов и определения на этой основе глобально лучшего агента [1]. Ниже приведены параметры алгоритма и его пошаговое представление.

Population – популяция; *LB, UB* – верхняя и нижняя граница области определения признака соответственно; S – размер популяции; q – количество мемплексов; p – количество агентов в каждом мемплексе; T – количество глобальных итераций; *Iter* – количество локальных

итераций; $\boldsymbol{\theta}_{best}$, $\boldsymbol{\theta}_{worst}$ – лучшее и худшее положение агента; $\boldsymbol{\theta}_{global}$ – глобальное лучшее положение агента; rand_i – равномерно распределенное случайное число в интервале [0,1].

Вход: S, *q*, *p*, *T*, *Iter*, *LB*, *UB*.

Выход: $\boldsymbol{\theta}_{best}$.

Population := { $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S$};

цикл по *d* от 1 до *T*

Find ($\boldsymbol{\theta}_{worst} := Max(E(\boldsymbol{\theta}))$, $\boldsymbol{\theta}_{best} := Min(E(\boldsymbol{\theta}))$);

Sort ($F = \{\boldsymbol{\theta}_1 < \boldsymbol{\theta}_2 < \ldots < \boldsymbol{\theta}_S\}$);

PartitionMemeplex (*Population* = $\{m_1, m_2, \ldots, m_q\}$);

цикл по *k* от 1 до *Iter*

цикл по *j* от 1 до *q*

цикл по *i* от 1 до *p*

$$\boldsymbol{\theta}_{i,new} := \text{rand}_i * (\boldsymbol{\theta}^{best} - \boldsymbol{\theta}^{worst}) + \boldsymbol{\theta}_i;$$

если ($E(\boldsymbol{\theta}_{i,new}) < E(\boldsymbol{\theta}_i)$) то

$$\boldsymbol{\theta}_i := \boldsymbol{\theta}_{i,new};$$

иначе

$$\boldsymbol{\theta}_{i,new} := \text{rand}_i * (\boldsymbol{\theta}^{global} - \boldsymbol{\theta}^{worst}) + \boldsymbol{\theta}_i;$$

если ($E(\boldsymbol{\theta}_{i,new}) < E(\boldsymbol{\theta}_i)$) то

$$\boldsymbol{\theta}_i := \boldsymbol{\theta}_{i,new};$$

иначе

$$\boldsymbol{\theta}_{i,new} := LB + \text{rand} * (UB - LB);$$

конец цикла;

конец цикла;

конец цикла;

CombineMemeplex (*Population* = $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S\}$);

конец цикла;

вывод $\boldsymbol{\theta}_{best}$:= Search_best(*Population*).

5. Дифференциальная эволюция (DE) – алгоритм, принадлежащий к группе эволюционных алгоритмов [9]. Параметрами алгоритма являются: $F \in [0,1]$ – параметр для создания нового вектора, S – размер популяции, N – количество итераций. На первом этапе выполнения происходит генерация популяции векторов $\boldsymbol{\theta}$. Далее на основе каждого вектора в популяции $\boldsymbol{\theta}_{cur}$ генерируется новый вектор $\boldsymbol{\theta}_{new}$. Если $E(\boldsymbol{\theta}_{new}) < E(\boldsymbol{\theta}_{cur})$, то текущий вектор заменяется в популяции новым. В завершение алгоритм возвращает лучший вектор в популяции $\boldsymbol{\theta}_{best}$.

В статье исследуются две модификации алгоритма дифференциальной эволюции. Данные модификации отличаются механизмом реализации кроссовера. В первой модификации (De_1) применяется бинарный кроссовер, вторая (De_2) использует экспоненциальный. Ниже приведен псевдокод первой модификации алгоритма, реализующей бинарный кроссовер.

Вход: *N*,*S*,*F*

Выход: $\boldsymbol{\theta}_{best}$.

Popul := $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S\}$;;

цикл пока (N>0)

цикл по p от 1 до S

$\boldsymbol{\theta}_{cur}$:= $Popul[p]$;

$\boldsymbol{\theta}_a$,$\boldsymbol{\theta}_b$:= Random_choose($Popul$);

θ_{best} := Search_best($Popul$);

CR :=rand(0,1);

цикл по d от 1 до $|\boldsymbol{\theta}_i|$

если (rand(0,1)< CR то

$\boldsymbol{\theta}_{new}[d]$:=$\boldsymbol{\theta}_{best}[d]$+F*($\boldsymbol{\theta}_a[d]$-$\boldsymbol{\theta}_b[d]$);

иначе $\boldsymbol{\theta}_{new}[d]$:=$\boldsymbol{\theta}_{cur}[d]$;

конец цикла

если ($E(\boldsymbol{\theta}_{new}) < E(\boldsymbol{\theta}_{cur})$) то

$\boldsymbol{\theta}_{cur}$:= $\boldsymbol{\theta}_{new}$;

N:=N-1;

конец цикла

конец цикла

вывод $\boldsymbol{\theta}_{best}$:= Search_best($Popul$).

6. Алгоритм «Всемирный потоп» (GDA) – метаэвристика, построенная на основе имитации процесса затопления участка земли. Задачей является проход по участку, без попадания в воду, с достижением максимальной высоты [8].

Основные обозначения алгоритма: x_i^o – текущее решение; x_i^c – новое решение; $E(\boldsymbol{\theta}_{\text{cur}})$ – текущее значение целевой функции; $E(\boldsymbol{\theta}_{\text{new}})$ – новое значение целевой функции; UP – параметр, увеличивающий уровень воды; u – равномерно распределенное случайное число в интервале [0,1], p – нечетное целое, $LEVEL$ –критерий приемлемости, N – количество итераций, n – количество внутренних итераций, minValueX, maxValueX – минимальное и максимальное значение входного вектора вещественных значений $\boldsymbol{\theta}$.

Принцип работы алгоритма: в поэлементном изменении входного вектора вещественных значений $\boldsymbol{\theta}$, в поисках решения $\boldsymbol{\theta}_{best}$ с минимальным значением оптимизируемой фитнес - функции $E(\boldsymbol{\theta})$. Псевдокод алгоритма представлен ниже.

Вход: $\boldsymbol{\theta}$, UP, p, N, n.

Выход: $\boldsymbol{\theta}_{best}$

x_i^o := Random(minValueX, maxValueX);

u := Random(0,1);

$LEVEL$:= $E(\boldsymbol{\theta}_{\text{cur}})$;

цикл пока (N > 0 или $(x_i^o - x_i^c)/ x_i^o$ < 0.000001)

цикл по i от 0 до n

$x_i^c := x_i^o + (10*u - 5)^p$;

$\boldsymbol{\theta}_{\text{new}} := \boldsymbol{\theta}[x_i^c]$;

если ($(E(\boldsymbol{\theta}_{\text{new}}) < E(\boldsymbol{\theta}_{\text{cur}})$ и $(E(\boldsymbol{\theta}_{\text{new}}) < LEVEL)$) то

$\boldsymbol{\theta}_{\text{cur}} := \boldsymbol{\theta}_{\text{new}}$;

$LEVEL := LEVEL - UP*(LEVEL - E(\boldsymbol{\theta}_{\text{new}}))$;

конец цикла;

конец цикла;

вывод $\boldsymbol{\theta}_{best} := \boldsymbol{\theta}_{\text{cur}}$.

7. Алгоритм сорняков (IWO) отражает поведение сорняков на ограниченной территории в борьбе за выживание в течение ограниченного времени [6]: 1) распределение начального/конечного числа семян; 2) производство выросшими растениями семян в зависимости от приспособленности растений; 3) размещение произведённых семян по области поиска; 4) отбор растений с более высокой приспособленностью; 5) повторение пунктов 2-4 до условий окончания процесса.

В качестве вектора для оптимизации выступают антецеденты и консеквенты нечеткой системы. Вектор представлен в виде массива θ_i, i принимает значения от 1 до $\sum_{j=1}^{L} kO_j + R$, где L – количество входных переменных нечеткой системы, k – количество переменных, описывающих терм, O – количество термов для j-ой переменной, R – количество правил в нечеткой системе. Псевдокод алгоритма и параметры приведены ниже. N – количество итераций; S – максимальное количество векторов; n_{min}, n_{max} – минимальное и максимальное количество дочерних векторов; σ – параметр нормального распределения; *Popul* – текущая популяция.

Вход: N, S, n_{min}, n_{max}, σ.

Выход: θ_{best}.

θ^0:= Generate_random();

Add_to_population(*Popul*, θ^0) ;

цикл по *iter* от 1 до N

цикл по s от 1 до текущего количества векторов в *Popul*

$$n^s := \frac{n_{\max} - n_{\min}}{E^{best} - E^{worst}} E(\bar{\theta}^s) + \frac{E^{best} n_{\max} - E^{worst} n_{\min}}{E^{best} - E^{worst}};$$

цикл по j от 1 до n^s

$$\sigma_N := \sigma\left(\frac{N - iter}{N}\right);$$

$$u \to N(0, \sigma_N) := \sigma_N \sqrt{-2\ln(a)} \cos(b);$$

$$\theta_i^{*s,j} := \theta_i^s + u;$$

конец цикла;

Add_to_population(*Popul*, $\theta^{*s,j}$) ;

конец цикла;

Sort(*Popul*)

Save_best(*Popul*, S) ;

конец цикла;

вывод θ_{best} := Search_best(*Popul*).

8. Эксперимент был выполнен на 12 наборах реальных данных из репозитория KEEL (http://www.keel.es). Описания указанных наборов приведены в таблице 1. Эксперимент проводился по принципу кросс-валидации. Для формирования обучающих и тестовых выборок исходный набор данных разделялся на десять равных частей, из девяти частей формировалась обучающая выборка, оставшаяся часть использовалась в качестве тестовой. По каждой обучающей выборке алгоритмом формирования структуры [3] была выполнена инициализация нечеткого классификатора, далее были запущены метаэвристики для оптимизации параметров классификатора.

В таблице 2 приведены усредненные значения процента правильной классификации на 12 наборах данных, классифицированных нечеткими классификаторами, параметры которых оптимизированы рассмотренными метаэвристиками. Жирным шрифтом выделены лучшие результаты классификации на тестовых выборках каждого набора данных. Курсивом выделены лучшие результаты классификации на обучающих выборках. Процент правильной классификации вычислен как разность $100\cdot(1-E(\mathbf{\theta}))$.

Таблица 1. Описание наборов данных

Наименование	Условное обозначение	Количество записей	Количество признаков	Количество классов
iris	irs	150	4	3
wine	wn	178	13	3
glass	gl	214	9	7
newthyroid	nth	215	5	3
cleveland	cld	297	13	5
haberman	hbm	306	3	2
bupa	bp	345	6	2
balance	bl	625	4	3
wisconsin	wsn	699	9	2
pima	pm	768	8	2
spambase	spb	4597	57	2
ring	rng	7400	20	2

Таблица 2. Эффективность нечетких классификаторов

Интерпретация		Наборы данных											
		irs	wn	gl	nth	cld	hbm	bp	bl	wsn	pm	spb	rng
		Количество правил											
		3	3	7	3	5	2	2	3	2	2	2	2
De_1	Обуч.	98.4	99.8	71.6	98.3	62.5	77.0	75.6	91.4	98.2	79.9	*90.6*	96.3
	Тест	92.7	96.9	66.3	96.3	58.0	73.5	69.0	90.4	**97.4**	**77.5**	89.3	95.1
De_2	Обуч.	98.6	99.6	70.5	99.5	63.4	77.2	74.6	91.5	*98.3*	*80.5*	90.2	*97.5*
	Тест	94.0	91.5	**67.0**	97.2	58.9	74.8	69.7	90.7	96.4	75.8	**89.6**	96.4
HS	Обуч.	*99.0*	*100*	70.0	*99.8*	*66.8*	77.3	*79.1*	*91.9*	98.0	80.3	87.0	95.2
	Тест	96.7	**97.2**	64.9	97.7	57.9	74.8	**74.0**	**91.0**	96.5	75.4	86.6	94.6
GSA	Обуч.	98.3	99.3	62.0	98.3	63.4	77.9	68.9	83.7	96.0	76.9	70.5	82.1
	Тест	**97.3**	97.1	50.8	**98.1**	**62.6**	76.1	69.0	81.8	96.3	74.0	69.7	82.5
FLA	Обуч.	92.6	*100*	*73.1*	93.3	61.1	*81.3*	80.4	82.4	97.3	78.8	87.4	97.4
	Тест	95.2	95.5	61.2	90.2	55.3	73.7	73.0	79.6	96.1	73.1	74.5	**96.6**
GDA	Обуч.	98.1	99.7	65.5	93.1	55.8	78.7	69.4	83.7	96.9	75.9	84.1	93.2
	Тест	**97.3**	92.1	61.0	93.6	56.7	75.9	67.3	83.4	94.9	74.6	83.1	93.9
IWO	Обуч.	96.9	94.5	63.6	93.6	54.4	78.5	68.3	90.7	96.0	69.2	60.6	49.5
	Тест	96.7	94.4	61.7	91.6	53.9	**77.5**	66.8	90.3	97.1	69.1	60.4	49.6

Заключение. В работе представлены метаэвристические методы оптимизации параметров нечетких классификаторов. Работоспособность нечетких классификаторов,

настроенных приведенным алгоритмом, проверена на двенадцати наборах данных из репозитория KEEL. Полученные классификаторы имеют хорошие способности к обучению (высокий процент правильной классификации на обучающих выборках) и не менее хорошие прогностические способности (высокий процент правильной классификации на тестовых выборках). Таким образом, рассмотренные алгоритмы могут быть рекомендованы для практического применения при решении задач оптимизации параметров нечетких классификаторов.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-07-00034а

СПИСОК ЛИТЕРАТУРЫ

1. Eusuff M., Lansey K. Optimization of water distribution network design using the shuffled frog leaping algorithm // Journal of Water Resources Planning and Management. 2003. Vol. 129. P. 210–225. DOI:10.1061/(ASCE)0733-9496(2003)129:3(210)
2. Geem Z.W., Kim J.H., Loganathan G.V. A new heuristic optimization algorithm: harmony search // Simulation. 2001. Vol. 76. P. 60–68. DOI: 10.1177/003754970107600201.
3. Hodashinsky I. A., Gorbunov I. V. Algorithms of the tradeoff between accuracy and complexity in the design of fuzzy approximators // Optoelectronics, Instrumentation and Data Processing. 2013. Vol. 49. P. 569–577. DOI: 10.3103/S875669901306006X.
4. Hodashinsky I.A., Minina D.Y., Sarin K.S. Identification of the parameters of fuzzy approximators and classifiers based on the cuckoo search algorithm // Optoelectronics. Instrumentation and Data Processing. 2015. Vol. 51. P. 234–240. DOI 10.3103/S8756699015030048.
5. Metaheuristics for Production Systems / Editors E.-G. Talbi, F. Yalaoui, L. Amodeo. Springer. London. 2016. 350 p. DOI 10.1007/978-3-319-23350-5.
6. Rad H. S., Lucas C. A Recommender System based in Invasive Weed Optimization Algorithm // IEEE Congress on Evolutionary Computation (CEC 2007). 2007. P. 4297–4304. DOI: 10.1109/CEC.2007.4425032.
7. Rashedi E., Nezamabadi-pour H., Saryazdi S. GSA: A Gravitational Search Algorithm // Information Sciences. 2009. Vol.179. P. 2232–2248. DOI:10.1016/j.ins.2009.03.004.
8. Ravi V. Modified Great Deluge Algorithm versus Other Metaheuristics in Reliability Optimization // Studies in Computational Intelligence. 2007. Vol. 40. P. 21–36. DOI: 10.1007/978-3-540-37372-8_2.
9. Storn R., Price K.V. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces (1995) Technical Report TR-95-012. ICSI (March 1995). ftp://ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.pdf

UDK 004.8

METAHEURISTICS FOR PARAMETERS OPTIMIZATION OF FUZZY CLASSIFIERS

Ilya A. Hodashinsky
Dr., Professor, e-mail: hodashn@rambler.ru
Alexander E. Anfilofiev
Graduate student, e-mail: yowwi00@gmail.com
Marina B. Bardamova
Student, e-mail: 722bmb@gmail.com
Vitaly S. Kovalev
Student, e-mail: vitaly_979@mail.ru
Maksim A. Mekh
Student, e-mail: maxcimkj93@gmail.com
Olga K. Sonich
Student, e-mail: zlasjasok@gmail.com
Tomsk State University of Control Systems and Radioelectronics,
40 Lenina Prospect, Tomsk, Russia 634050, Russia,

Annotation. Metaheuristics are widely recognized as efficient approaches for hard optimization problems. This paper addresses the application of metaheuristics for optimizing parameters of fuzzy classifiers. Several numerical experiments on well-known benchmark data sets are carried out to illustrate the effectiveness of the proposed metaheuristics.
Keywords: fuzzy classifiers, parameters optimization, metaheuristics

References

1. Eusuff M., Lansey K. Optimization of water distribution network design using the shuffled frog leaping algorithm // Journal of Water Resources Planning and Management. 2003. Vol. 129. P. 210–225. DOI:10.1061/(ASCE)0733-9496(2003)129:3(210)
2. Geem Z.W., Kim J.H., Loganathan G.V. A new heuristic optimization algorithm: harmony search // Simulation. 2001. Vol. 76. P. 60–68. DOI: 10.1177/003754970107600201.
3. Hodashinsky I. A., Gorbunov I. V. Algorithms of the tradeoff between accuracy and complexity in the design of fuzzy approximators // Optoelectronics, Instrumentation and Data Processing. 2013. Vol. 49. P. 569–577. DOI: 10.3103/S875669901306006X.
4. Hodashinsky I.A., Minina D.Y., Sarin K.S. Identification of the parameters of fuzzy approximators and classifiers based on the cuckoo search algorithm // Optoelectronics, Instrumentation and Data Processing. 2015. Vol. 51. P. 234–240. DOI 10.3103/S8756699015030048.
5. Metaheuristics for Production Systems / Editors E.-G. Talbi, F. Yalaoui, L. Amodeo. Springer. London. 2016. 350 p. DOI 10.1007/978-3-319-23350-5.
6. Rad H. S., Lucas C. A Recommender System based in Invasive Weed Optimization Algorithm // IEEE Congress on Evolutionary Computation (CEC 2007). 2007. P. 4297–4304. DOI: 10.1109/CEC.2007.4425032.
7. Rashedi E., Nezamabadi-pour H., Saryazdi S. GSA: A Gravitational Search Algorithm // Information Sciences. 2009. Vol.179. P. 2232–2248. DOI:10.1016/j.ins.2009.03.004.
8. Ravi V. Modified Great Deluge Algorithm versus Other Metaheuristics in Reliability Optimization // Studies in Computational Intelligence. 2007. Vol. 40. P. 21–36. DOI: 10.1007/978-3-540-37372-8_2.
9. Storn R., Price K.V. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces (1995) Technical Report TR-95-012. ICSI (March 1995). ftp://ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.pdf