available at www.sciencedirect.com

www.elsevier.com/locate/ecolinf

Discovery of predictive rule sets for chlorophyll-*a* dynamics in the Nakdong River (Korea) by means of the hybrid evolutionary algorithm HEA

Hongqing Cao[a,*], *Friedrich Recknagel*[a], *Gea-Jae Joo*[b], *Dong-Kyun Kim*[b]

[a]*School of Earth and Environmental Sciences, University of Adelaide, Adelaide, Australia*
[b]*Department of Biology, Pusan National University, Busan, South Korea*

ARTICLE INFO

Article history:
Received 15 February 2005
Received in revised form 29 July 2005
Accepted 9 August 2005

Keywords:
Hybrid evolutionary algorithm
Rule sets
Chl.*a*
Sensitivity analysis
Nakdong River

ABSTRACT

This paper presents a hybrid evolutionary algorithm (HEA) to discover complex rule sets predicting the concentration of chlorophyll-*a* (Chl.*a*) based on the measured meteorological, hydrological and limnological variables in the hypertrophic Nakdong River. The HEA is designed: (1) to evolve the structure of rule sets by using genetic programming and (2) to optimise the random parameters in the rule sets by means of a genetic algorithm. Time-series of input–output data from 1995 to 1998 without and with time lags up to 7 days were used for training HEA. Independent input–output data for 1994 were used for testing HEA. HEA successfully discovered rule sets for multiple nonlinear relationships between physical, chemical variables and Chl.*a*, which proved to be predictive for unseen data as well as explanatory. The comparison of results by HEA and previously applied recurrent artificial neural networks to the same data with input–output time lags of 3 days revealed similar good performances of both methods. The sensitivity analysis for the best performing predictive rule set unraveled relationships between seasons, specific input variables and Chl.*a* which to some degree correspond with known properties of the Nakdong River. The statistics of numerous random runs of the HEA also allowed determining most relevant input variables without a priori knowledge.

1. Introduction

It has been demonstrated that ecological time series, which are highly complex and nonlinear, can be successfully unraveled and predicted by artificial neural networks (ANN) and genetic algorithms (e.g. Recknagel et al., 1997, 1998; Maier et al., 1998; Stockwell, 1999; Liu and Yao, 1999; Jeong et al., 2001; Whigham and Recknagel, 2001; Recknagel et al., 2002; Jeong et al., 2003a,b; Lee et al., 2004; Recknagel et al., 2005). Even though ANN are very competitive in classifying or predicting noisy data by minimizing the root mean square error of approximations they lack an explicit representation. By contrast, Whigham and Recknagel (2001) proposed grammar based genetic programming to evolve functions and rules, and Bobbin and Recknagel (2003) applied an evolutionary based learning algorithm to discover predictive rules for population dynamics in limnological data. Even though both approaches allowed to discover predictive rules for ecological relationships they had following limitations: (1) the rules were relatively simple with attributes being associated only with constant parameters rather than function to reflect complex relationship between multiple attributes, and (2) the parameters which determine the output values on the rules are generated randomly rather than being simultaneously opti-

* Corresponding author.
E-mail address: hongqing.cao@adelaide.edu.au (H. Cao).

1574-9541/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ecoinf.2005.08.001

mised during the evolution. Whigham and Recknagel (2001) performed the hill climbing mutation for the fine tuning of the random real numbers and Bobbin and Recknagel (2003) adopted a self-adapting evolutionary algorithm to modify these parameters. However both methods fail when the number of parameters increases with the complexity of the rule.

This research aims at rule-based prediction and explanation of chlorophyll-*a* (Chl.*a*) dynamics by means of a hybrid evolutionary algorithm (HEA). HEA evolves the structure of the rule set by using genetic programming, and optimises the random parameters on the rule set by using a general genetic algorithm. Rules discovered by HEA have the IF-THEN-ELSE structure and allow imbedding complex functions synthesised from various predefined arithmetic operators. The maximum tree depth and rule set size control the complexity of rule sets.

The results demonstrate that HEA allows to discover rule sets which predict well unseen data and represent causal relationships between physical and chemical variables and Chl.*a* dynamics. Moreover the statistics of numerous random runs of the HEA also allowed determining most relevant input variables without a priori knowledge.

2. Hybrid evolutionary algorithm

Evolutionary algorithms (EA) are adaptive methods which mimic processes of biological evolution, natural selection and genetic variation. They search for suitable representations of a problem solution by means of genetic operators and the principle of "survival of the fittest". Due to their merits of self-organization, self-learning, intrinsic parallelism and generality, EA have been successfully applied to pattern recognition, economic prediction, optimum control and parallel processing (Goldberg, 1989; Bäck et al., 1997).

The principal framework of the suggested hybrid evolutionary algorithm (HEA) is represented in Fig. 1. HEA uses genetic programming (GP) to generate and optimize the structure of rule sets and a genetic algorithm (GA) to optimize the parameters of a rule set. GP (Koza, 1992, 1994; Banzhaf et al., 1997) is an extension of genetic algorithms (GA) (Holland, 1975; Mitchell, 1996) in which the genetic population consists of computer programs of varying sizes and shapes. In standard GP, computer programs can be represented as parse trees, where a branch node represents an element from a function set (arithmetic operators, logic operators,

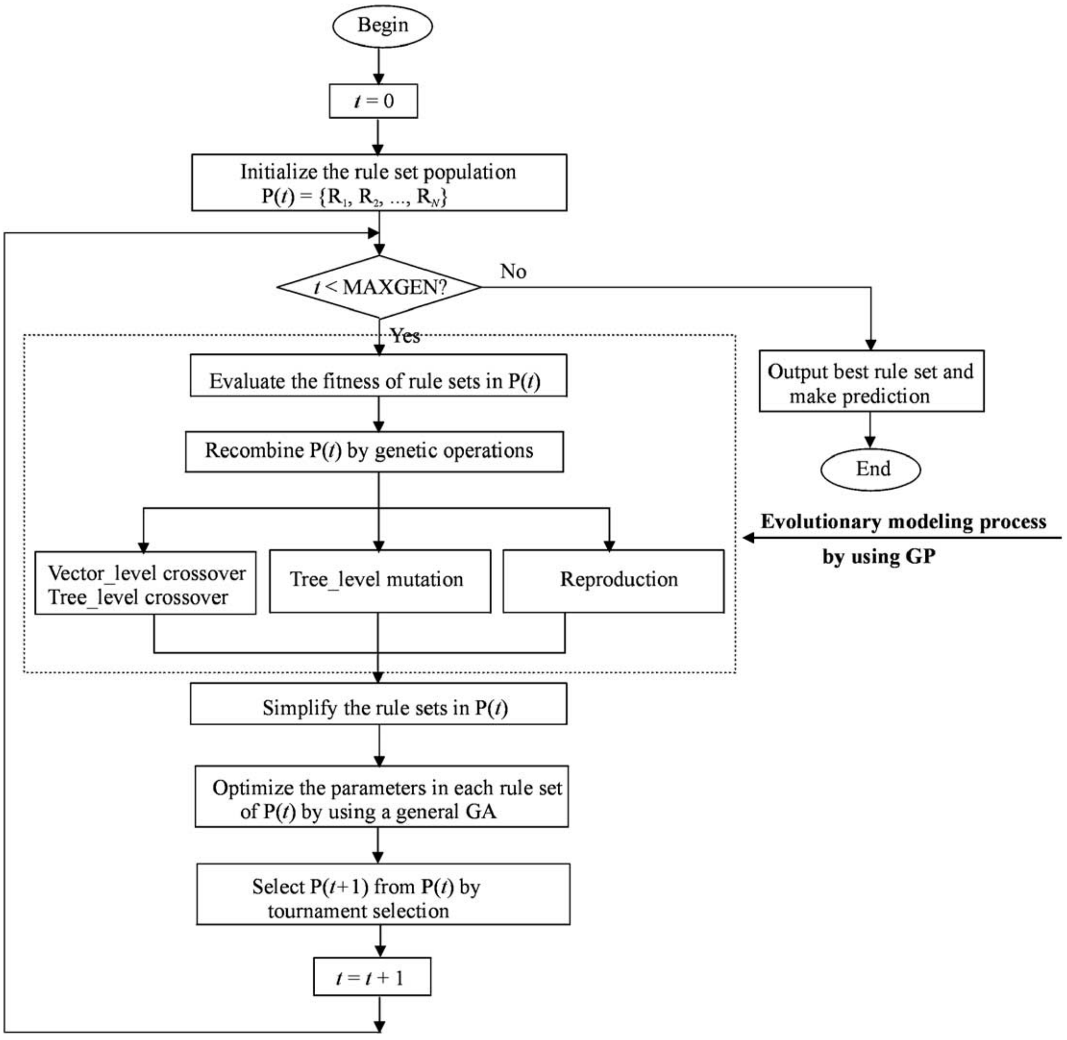
Begin
t = 0
Initialize the rule set population P(t) = {R1, R2, ..., RN}
t < MAXGEN?
No
Yes
Evaluate the fitness of rule sets in P(t)
Recombine P(t) by genetic operations
Vector_level crossover Tree_level crossover
Tree_level mutation
Reproduction
Output best rule set and make prediction
End
Evolutionary modeling process by using GP
Simplify the rule sets in P(t)
Optimize the parameters in each rule set of P(t) by using a general GA
Select P(t+1) from P(t) by tournament selection
t = t + 1

Fig. 1 – Flowchart of the hybrid evolutionary algorithm.

elementary functions of at least one argument), and a leaf node represents an element from a terminal set (variables, constants and functions of no arguments). These symbolic programs are subsequently evaluated by means of "fitness cases". Fitter programs are selected for recombination to create the next generation by using genetic operators, such as crossover and mutation. This step is iterated for consecutive generations until the termination criterion of the run has been satisfied. A general genetic algorithm (GA) is used to optimize the random parameters in the rule set.

2.1. *Structure optimization of rule sets using GP*

2.1.1. *Encoding*

We suppose each rule set has the form of

$$
\begin{aligned}
&\mathrm{IF}(T_{\mathrm{IF1}})\\
&\mathrm{THEN}\ y=(T_{\mathrm{THEN1}})\\
&\mathrm{ELSE}\\
&\mathrm{IF}\ (T_{\mathrm{IF2}})\\
&\mathrm{THEN}\ y=(T_{\mathrm{THEN2}})\\
&\mathrm{ELSE}\\
&\vdots\\
&\mathrm{IF}\ (T_{\mathrm{IFK}})\\
&\mathrm{THEN}\ y=(T_{\mathrm{THENK}})\\
&\mathrm{ELSE}\ y=(T_{\mathrm{ELSEK+1}})
\end{aligned}
\tag{1}
$$

where K is the size of the rule set, i.e. the number of IF branches, y is the output variable. Then each chromosome in the rule set population can be represented as a vector of binary trees denoted as (T_{IF1}, T_{THEN1}, T_{IF2}, T_{THEN2}, …, T_{IFK}, T_{THENK}, $T_{\mathrm{ELSEK+1}}$).

By defining the following three function sets as

Logic function set: F_L={AND, OR}
Comparison function set: $F_C=\{>, <, \geq, \leq\}$
Arithmetic function set: F_A={+, −, *, /, sin, cos, exp, ln}

the function sets of the IF_Tree (i.e. T_{IF1}, T_{IF2}, …, T_{IFK}} and the THEN/ELSE_Tree (i.e. T_{THEN1}, T_{THEN2}, …, T_{THENK}, $T_{\mathrm{ELSEK+1}}$) can be described as

$$F_{\mathrm{IF}}=F_L\cup F_C\cup F_A \text{ and } F_{\mathrm{THEN/ELSE}}=F_A$$

respectively. The terminal sets of the IF_Tree and the THEN/ELSE_Tree are the same as

$$T=\{x_1,\ldots,x_n,c\}$$

where n is the number of input variables and c is a random constant. For example, a rule set with the form of

$$
\begin{aligned}
&\mathrm{IF}((\ln|x_5|<98)\mathrm{AND}((x_2>30.8)\mathrm{OR}(x_3*x_4\leq 49.4)))\\
&\mathrm{THEN}\,y=x_1+x_3*\sin x_4-x_2*x_6\\
&\mathrm{ELSE}\\
&\mathrm{IF}\left((x_6>20.5)\mathrm{AND}\left(\frac{x_1}{x_2}\leq 4\right)\right)\\
&\mathrm{THEN}\,y=x_3*\exp(x_1)+\frac{x_4}{4}\\
&\mathrm{ELSE}\,y=\frac{x_2-3.5}{x_1*x_3+4.8}
\end{aligned}
\tag{2}
$$

can be represented as a vector of binary trees (T_{IF1}, T_{THEN1}, T_{IF2}, T_{THEN2}, T_{ELSE3}) illustrated in Fig. 2. Besides the function sets, the complexity of a rule set can be controlled by the predefined maximum size of a rule set (MAXK) and the maximum tree depth (D_{IF} and $D_{\mathrm{THEN/ELSE}}$ for the IF_Tree and the THEN/ELSE_Tree, respectively).

2.1.2. *Fitness evaluation*

Suppose that the ith observed data for the input variables and the output variable are (x_{1i}, x_{2i}, …, x_{ni}) and y_i respectively. As for each rule set with the form of (1), we calculate the return values (TRUE/FALSE) of T_{IF1}, T_{IF2}, …, T_{IFK} in sequence based on the observed values of input variables to find out which condition is first satisfied. Say the first IF_Tree to be satisfied is $T_{\mathrm{IF}m}$, we choose the corresponding THEN_Tree $T_{\mathrm{THEN}m}$ to calculate the predicted value of y_i denoted as $\hat{y}_i$. If none of these IF_Trees is satisfied, the only choice is to use the last tree $T_{\mathrm{ELSEK+1}}$ to calculate $\hat{y}_i$. Such procedure is performed on each data point from the training data. We define the RMSE (Root Mean Square Error) as the fitness function:

$$\mathrm{Fitness}=\sqrt{\frac{1}{k}\sum_{i=1}^{k}(\hat{y}_i-y_i)^2}$$

where k is the number of training data points. Obviously, here the lower the fitness value is the better is the rule set.

2.1.3. *Genetic operators*

Since each rule set is represented as a vector of trees, there are two levels of crossover available, the vector-level crossover and the tree-level crossover.

Consider two parents:

parent a: ($T^{(a)}_{\mathrm{IF1}}$, $T^{(a)}_{\mathrm{THEN1}}$, $T^{(a)}_{\mathrm{IF2}}$, $T^{(a)}_{\mathrm{THEN2}}$, …, $T^{(a)}_{\mathrm{IFKA}}$, $T^{(a)}_{\mathrm{THENKA}}$, $T^{(a)}_{\mathrm{ELSEKA+1}}$) and
parent b: ($T^{(b)}_{\mathrm{IF1}}$, $T^{(b)}_{\mathrm{THEN1}}$, $T^{(b)}_{\mathrm{IF2}}$, $T^{(b)}_{\mathrm{THEN2}}$, …, $T^{(b)}_{\mathrm{IFKB}}$, $T^{(b)}_{\mathrm{THENKB}}$, $T^{(b)}_{\mathrm{ELSEKB+1}}$)

where KA and KB are the sizes of the rule set a and b, respectively.

The vector-level crossover is performed as follows. Randomly select a position between the pairs of IF_THEN

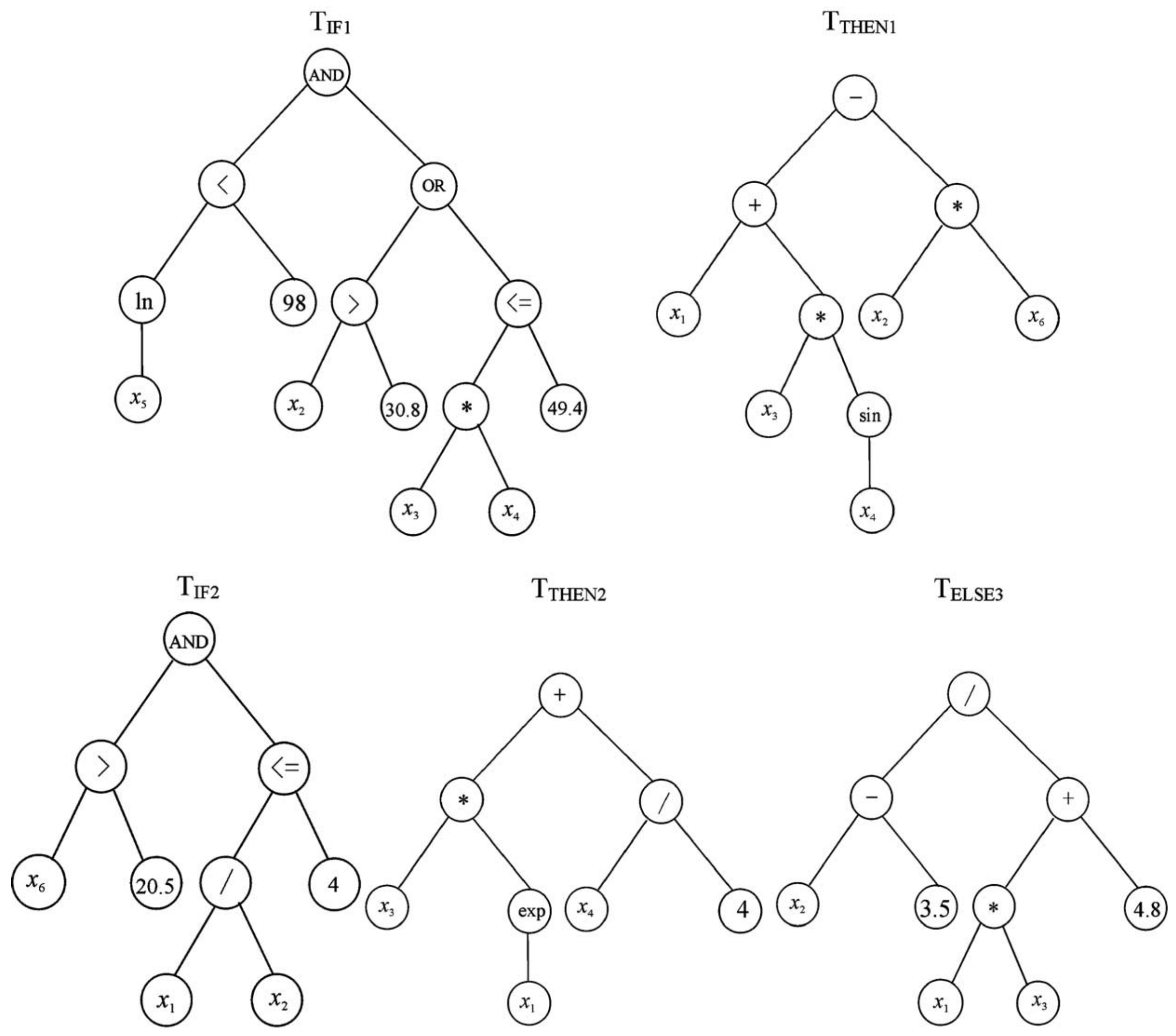

Fig. 2 – An example of the representation of a rule set in GP.

statement within parents *a* and *b* as the crossover point, say *j* and *k* for *a* and *b*, respectively. That is,

parent *a*: ($T^{(a)}_{\mathrm{IF1}}$, $T^{(a)}_{\mathrm{THEN1}}$, $T^{(a)}_{\mathrm{IF2}}$, $T^{(a)}_{\mathrm{THEN2}}$, …, $T^{(a)}_{\mathrm{IF}j-1}$, $T^{(a)}_{\mathrm{THEN}j-1}$, $T^{(a)}_{\mathrm{IF}j}$, $T^{(a)}_{\mathrm{THEN}j}$, $T^{(a)}_{\mathrm{IF}j+1}$, $T^{(a)}_{\mathrm{THEN}j+1}$, …, $T^{(a)}_{\mathrm{IFKA}}$, $T^{(a)}_{\mathrm{THENKA}}$, $T^{(a)}_{\mathrm{ELSEKA+1}}$)
parent *b*: ($T^{(b)}_{\mathrm{IF1}}$, $T^{(b)}_{\mathrm{THEN1}}$, $T^{(b)}_{\mathrm{IF2}}$, $T^{(b)}_{\mathrm{THEN2}}$, …, $T^{(b)}_{\mathrm{IF}k-1}$, $T^{(b)}_{\mathrm{THEN}k-1}$, $T^{(b)}_{\mathrm{IF}k}$, $T^{(b)}_{\mathrm{THEN}k}$, $T^{(b)}_{\mathrm{IF}k+1}$, $T^{(b)}_{\mathrm{THEN}k+1}$, …,$T^{(b)}_{\mathrm{IFKB}}$, $T^{(b)}_{\mathrm{THENKB}}$, $T^{(b)}_{\mathrm{ELSEKB+1}}$)

Then swap the corresponding IF-THEN-ELSE statements below the crossover points and produce two new rule sets:

offspring 1: ($T^{(a)}_{\mathrm{IF1}}$, $T^{(a)}_{\mathrm{THEN1}}$, $T^{(a)}_{\mathrm{IF2}}$, $T^{(a)}_{\mathrm{THEN2}}$, …, $T^{(a)}_{\mathrm{IF}j-1}$, $T^{(a)}_{\mathrm{THEN}j-1}$, $T^{(b)}_{\mathrm{IF}k}$, $T^{(b)}_{\mathrm{THEN}k}$, $T^{(b)}_{\mathrm{IF}k+1}$, $T^{(b)}_{\mathrm{THEN}k+1}$, …,$T^{(b)}_{\mathrm{IFKB}}$, $T^{(b)}_{\mathrm{THENKB}}$, $T^{(b)}_{\mathrm{ELSEKB+1}}$)
offspring 2: ($T^{(b)}_{\mathrm{IF1}}$, $T^{(b)}_{\mathrm{THEN1}}$, $T^{(b)}_{\mathrm{IF2}}$, $T^{(b)}_{\mathrm{THEN2}}$, …, $T^{(b)}_{\mathrm{IF}k-1}$, $T^{(b)}_{\mathrm{THEN}k-1}$, $T^{(a)}_{\mathrm{IF}j}$, $T^{(a)}_{\mathrm{THEN}j}$, $T^{(a)}_{\mathrm{IF}j+1}$, $T^{(a)}_{\mathrm{THEN}j+1}$, …, $T^{(a)}_{\mathrm{IFKA}}$, $T^{(a)}_{\mathrm{THENKA}}$, $T^{(a)}_{\mathrm{ELSEKA+1}}$)

We use either of them as the crossover offspring on condition that its size does not exceed MAXK.

The tree-level crossover is performed between the IF_Trees and the THEN/ELSE_Trees of two parents in sequence. First we do the IF_Tree crossover as follows. Randomly choose an IF_Tree from each parent and a node within the tree as a crossover point as well, swap the subtrees rooted at the crossover points and produce two new trees, then use either of them as the corresponding IF_Tree of the offspring on condition that its maximum depth does not exceed D_{IF}. It needs to be pointed out that in the IF_Tree, there are three different types of function nodes which come from F_{L}, F_{C}, F_{A}, respectively, to ensure that the crossover always produces legal rule sets, only the same type of nodes are selected as the crossover points. Afterwards the THEN/ELSE Tree crossover is done by following the similar procedure as above. The only difference is that we can choose any node as the crossover point due to their identical arithmetic node type. Finally we select either of the parents and replace the IF_Tree and the THEN/ELSE_Tree chosen previously with the newly generated ones by the above two-step crossover. Thus we get a complete crossover offspring of the two parents.

Similarly the tree-level mutation is performed on the IF_Tree and the THEN/ELSE_Tree of one parent in sequence. Given parent i:

$(T^{(i)}_{\mathrm{IF1}}, T^{(i)}_{\mathrm{THEN1}}, T^{(i)}_{\mathrm{IF2}}, T^{(i)}_{\mathrm{THEN2}}, \ldots, T^{(i)}_{\mathrm{IFK}}, T^{(i)}_{\mathrm{THENK}}, T^{(i)}_{\mathrm{ELSEK+1}})$

The tree-level mutation of i begins by randomly selecting an IF_Tree $T_{\mathrm{IF}j}$ ($1 \le j \le K$), and also a node within the tree as the mutation point, replacing the subtree rooted at the mutation point with a randomly generated IF_Tree, thus producing a new tree $T^{*(i)}_{\mathrm{IF}j}$. Afterwards the mutation of THEN/ELSE Tree is done by following the similar procedure as above. Suppose that the new tree produced is $T^{*(i)}_{\mathrm{THEN}k}$ or $T^{*(i)}_{\mathrm{ELSEK+1}}$ ($1 \le k \le K$). Thus we get a complete mutation offspring with the form of:

$(T^{(i)}_{\mathrm{IF1}}, T^{(i)}_{\mathrm{THEN1}}, T^{(i)}_{\mathrm{IF2}}, T^{(i)}_{\mathrm{THEN2}}, \ldots, T^{*(i)}_{\mathrm{IF}j}, T^{(i)}_{\mathrm{THEN}j}, \ldots, T_{\mathrm{IF}k}, T^{*(i)}_{\mathrm{THEN}k}, \ldots, T^{(i)}_{\mathrm{IFK}}, T^{(i)}_{\mathrm{THENK}}, T^{(i)}_{\mathrm{ELSEK+1}})$ or
$(T^{(i)}_{\mathrm{IF1}}, T^{(i)}_{\mathrm{THEN1}}, T^{(i)}_{\mathrm{IF2}}, T^{(i)}_{\mathrm{THEN2}}, \ldots, T^{*(i)}_{\mathrm{IF}j}, T^{(i)}_{\mathrm{THEN}j}, \ldots, T^{(i)}_{\mathrm{IFK}}, T^{(i)}_{\mathrm{THENK}}, T^{*(i)}_{\mathrm{ELSEK+1}})$.

2.2. *Simplification of rule sets*

The simplification of rule sets includes the simplification of the IF_Tree and the THEN/ELSE Tree. We use the following consecutive steps to simplify the IF_Tree in each rule set:

(1) Simplification of the arithmetic subtrees: It is done by replacing subtrees which consist of arithmetic operations in F_A between constants by their calculated values.
(2) Simplification of the comparison subtrees: It is done by replacing the subtrees which consist of comparison operations in F_C between constants by their comparison outcome, i.e. 0 or 1 for TRUE and FALSE, respectively.
(3) Simplification of the logic subtrees: We use Table 1 to simplify the AND subtrees and OR subtrees which consist of 0 or 1 in their branch nodes.We only use the above step (1) to simplify the THEN/ELSE_Tree in each rule set.

In addition we delete the redundant pairs of IF_THEN statements from the original rule set by checking the number of the input data points which can satisfy the condition of the IF_Tree. If the number is zero, then the corresponding IF-THEN pair is regarded as making no sense and should be deleted from the original rule set. The size of the rule set can thus be significantly reduced in this way.

The simplification of rule set is performed on all individuals in every generation. This procedure should be done prior to the parameter optimization because it is helpful to reduce the total number of parameters to optimize while maintaining the fitness of the rule set.

Table 1 – The simplification of the logic subtrees

AND	OR
0 AND 0=0	0 OR 0=0
0 AND 1=0	0 OR 1=1
1 AND 0=0	1 OR 0=1
1 AND 1=1	1 OR 1=1
0 AND subtree=0	0 OR subtree=subtree
1 AND subtree=subtree	1 OR subtree=1

2.3. *Parameter optimization of rule sets using a general genetic algorithm*

As the parameters in the rule set, especially those contained in the IF-Trees, play an important role in calculating the accuracy of the rule set, they need to be optimised in each generation. Here we design a general genetic algorithm (GA) to approach this task.

GAs can have various forms due to different representations, fitness evaluations and genetic operators which may vary with specific problems. Among all these components, genetic operators, including crossover and mutation, are usually considered as the most important parts. Here we used a novel crossover operator based on the nonconvex linear combination of multiple parents during the recombination of the population, which proved to work stably and effectively in solving the problem of multiple parameters optimization (Yu et al., 1999).

2.3.1. *Encoding*

At the beginning, we first check all the constants contained in the IF_Trees and the THEN/ELSE_Trees of the rule set, including counting the number of constants l and recording their positions. Each individual in the parameter population can then be represented as an l-dimensional row vector (c_1, c_2, …, c_l) where each component c_i for $i=1, 2, \ldots, l$ is encoded as a floating number and generated randomly ranging from 0 to 20 during the initialization of the parameter population.

2.3.2. *Fitness evaluation*

Before the fitness evaluation of an individual in the parameter population, we first return to the original rule set and replace all constants with the corresponding components of the row vector (i.e. the individual) and then follow the same procedure as in Section 2.1.2 to calculate the fitness.

2.3.3. *Genetic operators*

We use a multiple-parent crossover operator to create a new individual in the parameter population in the following way. Randomly select M different individuals from the old population ($M>2$) denoted as $X_1, X_2, \ldots, X_M$ where $X_k=(c_{1k}, c_{2k}, \ldots, c_{lk})$ (k: $1 \sim M$). Produce M coefficients α_k, where α_k ranges from a to b ($a<0$, $b>1$), which satisfy $\sum_{k=1}^{M} \alpha_k = 1$. Generate a new individual, X, by the nonconvex linear combination of these M individuals as follows:

$$X = \sum_{k=1}^{M} \alpha_k X_k$$

If the fitness value of X is lower than that of the worst individual in the current population, then replace it with X. This step is iterated a predetermined maximum number (MAX) of times. There are three adjustable control parameters M, a, b in this procedure. Setting their optimal values depends upon the properties of the specific problem.

Table 2 – Parameters used as input and output variables in the evolutionary modelling of rule sets

Division	Categories	Variables	Unit	Mean ± STDEV	Min	Max
Input variables	Meteorological	Irradiance (Irra.)	MJ $m^{-2}day^{-1}$	12.86 ± 6.47	0.18	28.89
	Hydrological	Average precipitation (Prec.)	mm day^{-1}	3.09 ± 10.23	0	185
		Discharge in Samlangjin (Disc.)	CMS day^{-1}	567.03 ± 714.09	101	9087
		Evaporation (Evap.)	mm day^{-1}	3.25 ± 1.61	0	8.8
	Physical	Water temperature (Temp)	°C	17.38 ± 9.1	1.1	34.4
		Secchi depth (SD)	cm	74.22 ± 24.74	5	145
		Turbidity (Turb.)	NTU	17.51 ± 54.4	1.8	648
	Chemical	pH (pH)		8.36 ± 0.8	6.45	10.2
		Dissolved Oxygen (DO)	mg L^{-1}	10.77 ± 3.94	3.4	20
		Nitrate-N (NO_3)	mg L^{-1}	2.69 ± 0.98	0.19	5.61
		Ammonia-N (NH_4)	mg L^{-1}	0.56 ± 0.68	0.0028	4
		Phosphate-P (PO_4)	µg L^{-1}	34.68 ± 25.18	1	117.04
		Dissolved silica (SiO_2)	mg L^{-1}	4.29 ± 3.77	0.01	16.25
Output variable	Biological	Chlorophyll-a (Chl.*a*)	µg L^{-1}	53.53 ± 106.83	0	1035

2.4. *Selection strategy*

We use tournament selection with sample size of 4 to recombine the new rule set population. That is, each time we randomly choose 4 different individuals from the current rule set population and compare their fitness values. The best one among them is added to the new population. This procedure is repeated until the predefined population size *N* is reached. In the meantime an elitism strategy is adopted which means we always keep the best rule set in the current generation to the next generation.

2.5. *Model prediction*

Once the best rule set is obtained in one run, we then test its validity and generality by calculating the predicted values on the testing data points and the RMSE for the testing data. A lower RMSE for the unseen data usually implies that the rule set has better generalised the patterns found in the training data.

3. The Nakdong River dataset

The Nakdong River basin is situated in the southeastern part of South Korea. South Korea experiences four distinct seasons, and is characterized by heavy rainfall during the monsoon season and several typhoon events. The annual mean precipitation across the river basin is about 1200 mm, but more than 50% of the annual rainfall is concentrated during summer (June–August). The annual mean water temperature at the study site was 13.7 °C. The mean water temperature was 2.2 °C during the coldest month (January), and 25.9 °C in August, the warmest month.

A number of climatic and limnological variables have been collected over a 5-year period (1994–1998) for the Nakdong River as shown in Table 2. Precipitation data were obtained from five representative meteorological stations (Andong, Daegu, Hapchon, Jinju, and Miryang) within the Nakdong River basin. River flow data was gained from the Flood Control Center. Irradiance and evaporation data were collected from the Busan Local Meteorological Station, which is the closest to the study site. Weekly water samples were collected at 0.5 m depth and the following water quality parameters were measured: water temperature, Secchi depth, turbidity, pH, concentrations of dissolved oxygen, nitrate-N, ammonia-N, phosphate-P, dissolved silica and Chl.*a*.

A simple linear interpolation has been used to fill missing values to produce a complete daily time series for this period. For this study the data from 1995 to 1998 were used for training and the data of 1994 were used for testing the generalisation behaviour of the resulting rule sets.

4. Modeling experiments

4.1. *Parameter settings and measures*

To examine the effectiveness of the HEA, we applied it to data sets with no-delay and time-lagged inputs by 1–7 days. 100 runs were conducted independently for each data set. All the experiments were performed on a Hydra supercomputer (IBM eServer 1350 Linux) with a peak speed of 1.2 TFlops by using the programming language C. The parameter settings of the HEA are listed in Table 3.

In addition, in order to validate the results of different rule sets not only the training error (fitness) but also the testing error (RMSE) is calculated as follows:

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}$$

where m is the number of testing data points, y_i and $\hat{y}_i$ are the ith observed value and the ith predicted value of Chl.*a*, respectively.

Table 3 – Parameter settings of the hybrid evolutionary algorithm for rule set discovery

Structure Optimization (GP)	$N = 200$
	$F_L = \{AND, OR\}$
	$F_C = \{>, <, \geq, \leq\}$ $F_A = \{+, -, *, /, \exp, \ln\}$
	MAXK = 5 $D_{IF} = D_{THEN/ELSE} = 4$
	MAXGEN = 100
Parameter Optimization (GA)	popsize = 50 $a = -0.5$ $b = 1.5$ $M = 8$
	MAX = 500

Table 4 – Statistical results for the training error and the testing error with various time-lagged input data in 100 runs

Time-lagged input data	Training error				Testing error				Mean runtime (min)
	Max	Min	Mean	STDEV	Max	Min	Mean	STDEV	
No delay	42.04	30.82	36.19	2.30	145.98	78.22	126.12	8.14	78
1-day delay	42.38	31.08	37.28	2.51	143.24	97.26	126.11	6.81	82
2-day delay	44.07	33.68	38.31	1.90	141.19	87.79	126.32	7.42	75
3-day delay	44.51	33.75	39.28	1.98	143.96	96.08	126.60	7.20	86
4-day delay	44.41	34.93	40.29	1.84	168.0	120.47	129.31	5.13	87
5-day delay	45.54	34.98	40.55	1.78	139.76	120.71	130.51	3.15	82
6-day delay	45.67	37.51	40.83	1.61	158.21	127.0	132.66	3.45	79
7-day delay	46.67	36.80	40.39	1.46	137.81	126.76	133.54	2.17	80

4.2. *Results and discussion*

Table 4 represents the statistics for the results of 100 runs of HEA with and without time-lagged input data. It shows that the mean training and testing errors increase only slightly with increasing lag time while the standard deviation (STDEV) of the two errors decreases slightly with increasing lag time. As far as the minimal testing error is concerned, the models trained with less than 4 days lag time appear to be more predictive for unseen data. As Jeong et al. (2001) identified 3 days as optimum time lag to predict Chl.*a* of the Nakdong River in 1994 by recurrent ANN we compare subsequently the 3 days time lag results between the recurrent ANN and HEA.

Fig. 3 illustrates the frequencies of the specific input variable selection in 100 runs of HEA considering a 3 days input time lag. It clearly indicates Secchi depth, dissolved oxygen, turbidity and silica as key input variables for Chl.*a* whilst evaporation, precipitation and irradiance appear to be less important. The information in Fig. 3 allows to select the relevant and important inputs so as to shorten the modelling time and improve the accuracy of the model by HEA, especially when the number of input variables increases significantly.

The best rule set in terms of the minimal testing error obtained with 3-day-delay input data in 100 runs is:

$$\text{IF } ((\exp(\text{pH})/\text{pH}) < 436.346)$$

$$\text{THEN Chl.}a = \ln(|(\exp(\text{SD}) * \text{NH4})|)$$

$$\text{ELSE Chl.}a = \text{DO} - \text{SD} + 2 * \text{Turb.} - \text{SiO2} - \text{Prec.} - \text{Irra.} + 123.593 \quad (3)$$

The testing and the training error were 96.08 and 41.66, respectively. Fig. 4 shows the training results (top) and the testing results (bottom) of the HEA rule set. The timing and magnitudes of the predicted Chl.*a* compare well with the observed data for most seasons while the summer peak of algal biomass is slightly under-estimated. The predictive validity of the rule set is similarly good as that of the recurrent ANN even though additional zooplankton input data were not used by HEA but used by the recurrent ANN. The *R*-squared value achieved by the recurrent ANN is 0.82 compared to 0.53 achieved by HEA.

In order to better understand the behaviour of the rule set in response to changed relevant input variables, a sensitivity analysis was conducted with a disturbance of the inputs by ±2 STDEV. Due to the nature of rule sets, the sensitivity analysis was applied to different branches of different IF conditions separately. In order to do this, the input data had to be divided into several disjoint data sets satisfying IF-conditions. Then the sensitivity analysis was applied to the corresponding function models defined by THEN/ELSE branches for the specific IF condition one by one. An example for the data division for the rule set (Eq. (3)) is represented in Fig. 5. As this rule set only consists of one IF branch, the input data are divided into two parts depending on if the value of exp(pH)/pH is greater or smaller than 436.346. The corresponding pH value for this threshold is about 8.18. The sensitivity analyses for the THEN_Tree and the ELSE_Tree are plotted in Fig. 6. It shows that the sensitivity of Chl.*a* is always high to Secchi depth, but the changing trends are opposite for the THEN_Tree and the

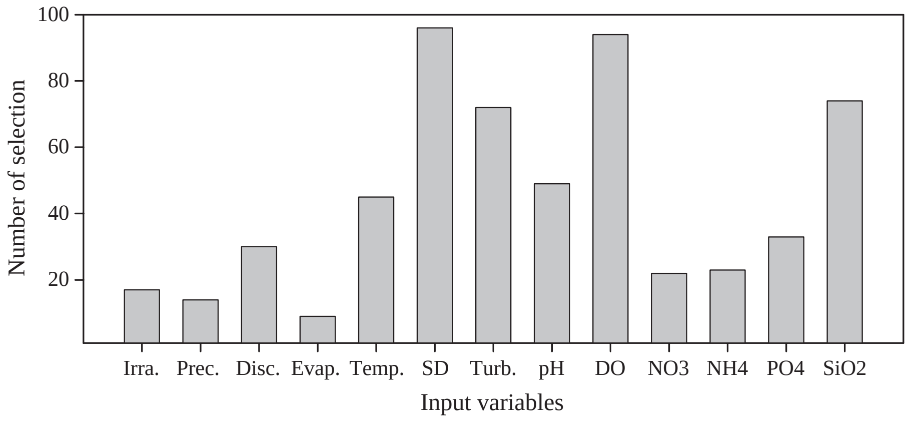

Fig. 3 – The input variable selection with 3-day-delay input data in 100 runs.

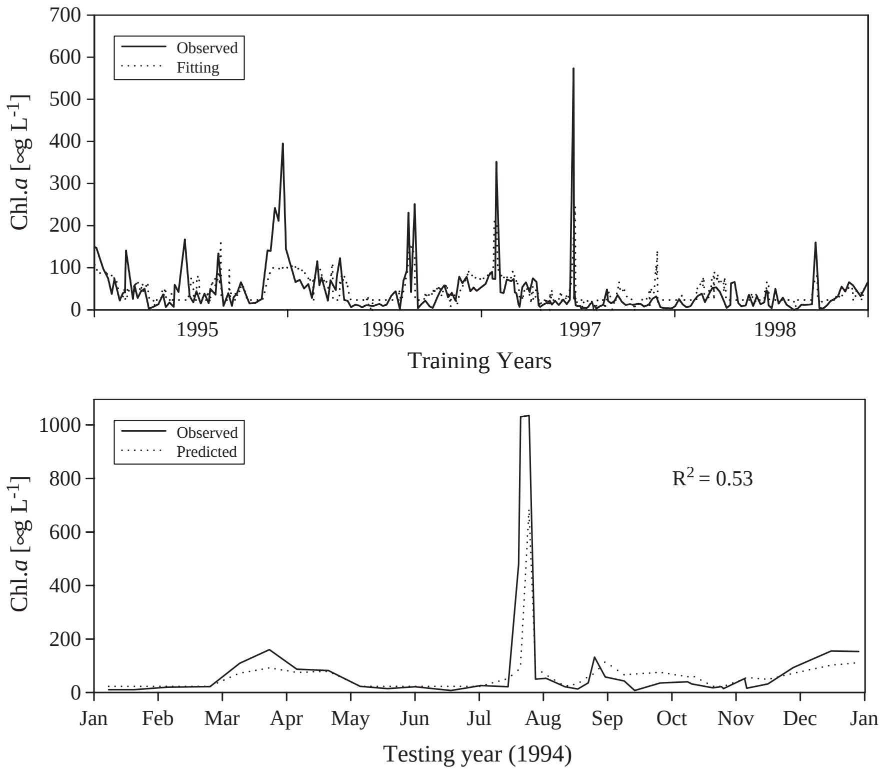

Fig. 4 – The training and testing results of the best rule set in terms of minimal testing error with 3-day-delay input data in 100 runs.

ELSE_Tree. When pH value is less than 8.18, a larger Secchi depth indicates a higher algal biomass. Reversely when pH value is more than 8.18, the Chl.*a* decreases linearly with the increases of Secchi depth. This is explainable if we look into the seasonal distribution of these two data sets. From Fig. 5, we observed that for the THEN_Tree, most data points (below the solid horizontal line of 436) are distributed in June, July while for the ELSE_Tree in March, April and October. Usually there is a lot of rain during summer in the lower Nakdong River. During the rainy season, both the alga biomass and Secchi depth decrease due to the flushing impact of the rain. By contrast during spring and autumn (the clear water phase), significant grazing impact of zooplankton on algae can be observed in the river. Therefore the larger Secchi depth by improved water transparency usually indicates lower algal biomass and turbidity. The sensitivity of Chl.*a* is also high to changes in turbidity, which indicates an opposite trend compared to Secchi depth as high turbidity limits underwater light for photosynthesis. In addition from Fig. 6, we can see that for the THEN_Tree, Chl.*a* experiences

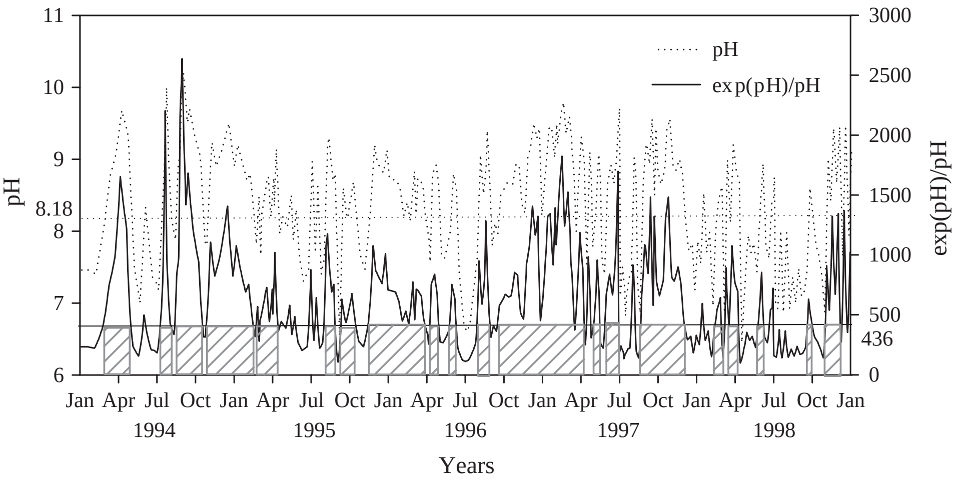

Fig. 5 – Data division and seasonal distribution by the best rule set with 3-day-delay input data.

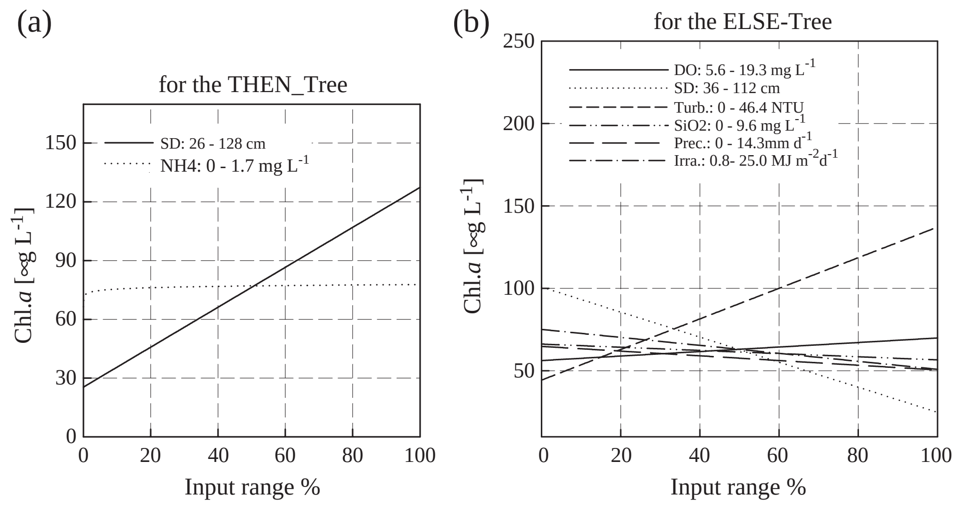

Fig. 6 – The results of sensitivity analysis with wide-ranged disturbance (±2 STDEV) for the best rule set with 3-day-delay input data.

little sensitivity to changes in ammonia-N and for the ELSE_Tree low sensitivity to changes in dissolved oxygen, silica, precipitation and irradiance. These results are very similar to those of ANN.

As pH plays an important role on the data division and Secchi depth indicates distinct relationships of Chl.*a* with different seasons, Fig. 7 illustrates the sensitivity analyses for pH and Secchi depth by changes of the testing data by ±1 STDEV. Whilst the sensitivity of Chl.*a* to pH is relatively high in January, February, June and October, it is moderate to Secchi depth at any time of the year. This is consistent with the previous result shown in Fig. 6.

Table 5 summarises the best rule sets obtained from data with and without time lags. It can be seen that the structures

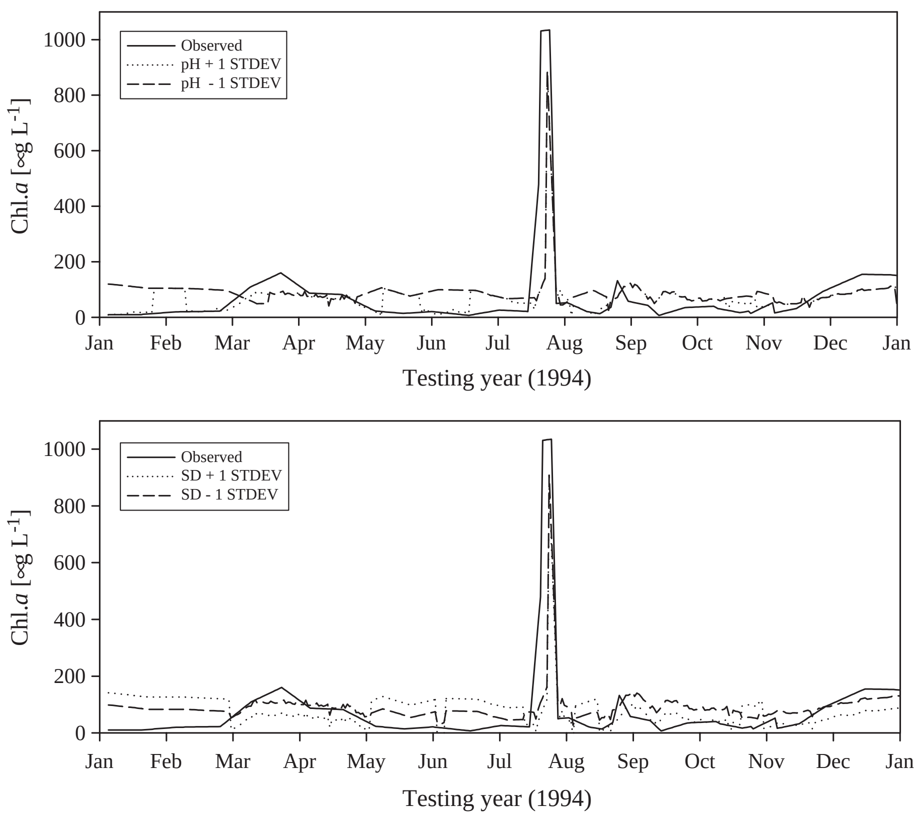

Fig. 7 – The results of sensitivity analysis of the best rule set with 3-day-delay input data on testing year for pH (top) and Secchi depth (bottom) with disturbance (±1 STDEV).

Table 5 – The best rule set in terms of minimal testing error with various time-lagged input data in 100 runs

Time-lagged input data	Best rule set	Testing error	Training error

No delay	IF ((SD>90.603) OR (Prec.>59.823))
	THEN Chl.*a*=17.853
	ELSE
	IF ((SD≤8.531) OR (Prec.>59.823))
	THEN Chl.*a*=17.613
	ELSE
	IF (Prec.>3.990)
	THEN Chl.*a*=DO*3.552
	ELSE
	IF ((SD≥59.823) OR (pH≤8.531))
	THEN Chl.*a*=DO*3.552
	ELSE
	Chl.*a*=Turb.*3+100.826
		78.22	37.46

1-day delay	IF ((SiO2+2*(NH4–pH))≤(–11.696))
	THEN Chl.*a*=Turb.+DO–2*SD+190.977
	ELSE
	Chl.*a*=DO*ln(|(Irra.+22.076)|)–5.641
		97.26	40.51

2-day delay	IF ((((SD>91.608) AND (DO≥4.836)) OR (SD≤13.451)) AND (DO≤13.451))
	THEN Chl.*a*=19.319
	ELSE
	IF ((pH<8.403)) AND (DO≤69.426))
	THEN Chl.*a*=DO+23.026
	ELSE
	IF ((((PO4>10.211) AND (DO≥4.836)) OR (SD≤13.451)) AND (DO≤13.451))
	THEN Chl.*a*=3*Turb.+18.179
	ELSE
	IF (SD≤49.357)
	THEN Chl.*a*=212.253
	ELSE
	Chl.*a*=Turb.*2.675+47.651
		87.79	36.79

4-day delay	IF ((SD≤38.426) OR (exp(SiO2)≥97.812))
	THEN Chl.*a*=3*DO-NH4*SiO2
	ELSE
	Chl.*a*=2*(Turb.+DO-NH4) - SD+96.989
		120.47	42.44

5-day delay	IF (((Turb.-SD)≤(–42.831)) OR (SD<10.450))
	THEN Chl.*a*=DO*3.636
	ELSE
	IF (((Turb.–SD)≤(–42.831)) OR ((Temp.*SiO2)>187.346))
	THEN Chl.*a*=DO*3.171
	ELSE
	IF (((Turb.–DO–SD)≤(–58.955)) OR ((Turb.*SiO2)>170.013))
	THEN Chl.*a*=DO*4.051
	ELSE
	IF (((Turb.–DO–SD)≤(–58.955)) OR ((Temp.*SiO2)>170.013))
	THEN Chl.*a*=pH
	ELSE
	IF ((Turb.–Temp.*SiO2–SD)<(–42.842))
	THEN Chl.*a*=Evap.+Turb.*Evap.+26.084
	ELSE
	Chl.*a*=DO*(DO*ln(|ln(|pH|)|))
		120.71	37.27

6-day delay	IF (SiO2>5.158)
	THEN Chl.*a*=49.321/NO3+DO
	ELSE
	IF (Temp.<11.246)
	THEN Chl.*a*=DO-SD+129.784
	ELSE
	IF (((Disc.+ln(|Disc.|))/DO)>56.610)
	THEN Chl.*a*=Disc./(47.853-Irra.)
	ELSE
	Chl.*a*=127.764-SD
		127.0	43.61

7-day delay	IF (Temp.≥30.341)
	THEN Chl.*a*=(DO*53.155)/ln(|(Disc.*SiO2)|)
	ELSE
	IF (SD<29.303)
	THEN Chl.*a*=ln(|(Disc.*ln(|(Disc.*16.103)|))|)
	ELSE
	IF (Temp.≥27.722)
	THEN Chl.*a*=49.614
	ELSE
	IF (Temp.≥11.384)
	THEN Chl.*a*=DO/0.311
	ELSE
	IF (SD<49.89)
	THEN Chl.*a*=138.243-ln(|SiO2|)*58.722
	ELSE
	Chl.*a*=(DO*28.602)/ln(|(Disc.*SiO2)|)
		126.76	39.71

of those rule sets discovered by HEA are diverse, and their sizes change from 1 to 5.

5. Conclusion

A hybrid evolutionary algorithm (HEA) has been developed to discover predictive rule sets in complex ecological data. It has been designed to evolve the structure of rule sets by using genetic programming and to optimise the random parameters in the rule sets by means of a genetic algorithm.

HEA was successfully applied to meteorological, hydrological and limnological time series data of the hypertrophic Nakdong River in order to predict Chl.*a*. The results have demonstrated that HEA is able to discover rule sets, which are predictive for unseen data but also explanatory for relationships between physical, chemical variables and Chl.*a*. The sensitivity analysis for the best performing rule set with 3-day lagged input data indicated distinct relationships between Chl.*a*, Secchi depth and pH which correspond with known seasonal patterns of the Nakdong River. The frequency of input selections by numerous random runs of HEA also allowed to identify most relevant input variables without a priori knowledge.

Future work will include to consecutively swap years of data for training and testing in order to determine whether one generic rule performs best for all testing years.

REFERENCES

Bäck, T., Hammel, U., Schwefel, H.-P., 1997. Evolutionary computation: comments on the history and current state. IEEE Transaction on Evolutionary Computation 1 (1), 5–16.

Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D., 1997. Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann.

Bobbin, J., Recknagel, F., 2003. Evolving rules for the prediction and explanation of blue-green algal succession in lakes by evolutionary computation. In: Recknagel, F. (Ed.), Ecological Informatics. Understanding Ecology by Biologically-Inspired Computation. Springer-Verlag, Berlin, pp. 291–310.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Welsey, Reading, MA.

Holland, J.H., 1975. Adaptation in Natural and Artificial System. University of Michigan Press, Ann Arbor, MI.

Jeong, K.-S., Joo, G.-J., Kim, H.-W., Ha, K., Recknagel, F., 2001. Prediction and elucidation of phytoplankton dynamics in the Nakdong River (Korea) by means of a recurrent artificial neural network. Ecological Modelling 146 (1–3), 115–130.

Jeong, K.-S., Recknagel, F., Joo, G.-J., 2003a. Prediction and elucidation of population dynamics of the blue-green algae *Microcystis aeruginosa*and the diatom *Stephanodiscus hantzschii* in the Nakdong river-reservoir system (South Korea) by a recurrent artificial neural network. In: Recknagel, F. (Ed.), Ecological Informatics. Understanding Ecology by Biologically-Inspired Computation. Springer Verlag, Berlin, pp. 195–210.

Jeong, K.S., Kim, D.K., Whigham, P., Joo, G.J., 2003b. Modeling *Microcystis aeruginosa* bloom dynamics in the Nakdong River by means of evolutionary computation and statistical approach. Ecological Modelling 161, 67–78.

Koza, J.R., 1992. Genetic programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA.

Koza, J.R., 1994. Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge, MA.

Liu, Y., Yao, X., 1999. Time series prediction by using negatively correlated neural networks. Lecture Notes in Artificial Intelligence, vol. 1585. Springer-Verlag, Berlin, pp. 325–332.

Lee, J.H.W., Fernando, T.M.K.G., Wong, K.T.M., 2004. Real time prediction of coastal algal blooms using artificial neural networks. In: Liong, P., Babovic (Eds.), Proceedings of the 6th International Conference on Hydroinformatics. World Scientific Publishing Company.

Maier, H.R., Dandy, G.C., Burch, M.D., 1998. Use of artificial neural networks for modeling cyanobacteria *Anabaena* spp. in the River Murray, South Australia. Ecological Modelling 105, 257–272.

Mitchell, M., 1996. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA.

Recknagel, F., French, M., Harkonen, P., Yabunaka, K., 1997. Artificial neural network approach for modeling and prediction of algal blooms. Ecological Modelling 96 (1–3), 11–28.

Recknagel, F., Fukushima, T., Hanazato, T., Takamura, N., Wilson, H., 1998. Modelling and prediction of phyto- and zooplankton dynamics in Lake Kasumiguara by artificial neural networks. Lakes and Reservoirs: Research and Management 3, 123–133.

Recknagel, F., Bobbin, J., Whigham, P., Wilson, H., 2002. Comparative application of artificial neural networks and genetic algorithms for multivariate time-series modelling of algal blooms in freshwater lakes. Journal of Hydroinformatics 4 (2), 125–134.

Recknagel, F., Welk, A., Kim, B., Takamura, N., 2005. Artificial Neural Network Approach to Unravel and Forecast Algal Population Dynamics in Two Lakes Different in Morphometry and Eutrophication, In: Recknagel, F. (Ed.), Ecological Informatics, 2nd edition. Springer-Verlag, Berlin, pp. 310–329.

Stockwell, D.R.B., 1999. Machine learning methods for ecological modelling. Chapter Genetic Algorithms II: Species Distribution Modeling. Kluwer Academic Publishers, pp. 123–144.

Whigham, P.A., Recknagel, F., 2001. An inductive approach to ecological time series modelling by evolutionary computation. Ecological Modelling 146, 275–287.

Yu, J.X., Cao, H.Q., Chen, Y.Y., Kang, L.S, Yang, H.X., 1999. A new approach to estimation of the electrocrystallization parameters. Journal of Electroanalytical Chemistry 474 (1), 69–73.