This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

Artificial Immune System with Local Feature Selection for Short-Term Load Forecasting

Grzegorz Dudek

***Abstract*—In this work a new forecasting model based on artificial immune system is proposed. The model is used for short-term electrical load forecasting as an example of forecasting time series with multiple seasonal cycles. Artificial immune system learns to recognize antigens representing two fragments of the time series: fragment preceding the forecast (input vector) and forecasted fragment (output vector). Antibodies as recognition units recognize antigens by selected features of input vectors and learn output vectors. In the test procedure new antigen with only input vector is recognized by some antibodies. Its output vector is reconstructed from activated antibodies. The unique feature of the proposed artificial immune system is the embedded property of local feature selection. Each antibody learns in the clonal selection process its optimal subset of features (a paratope) to improve its recognition and prediction abilities. In the simulation studies the proposed model was tested on real power system data and compared with other artificial immune system-based forecasting models as well as neural networks, ARIMA and exponential smoothing. The obtained results confirm good performance of the proposed model.**

***Index Terms*—artificial immune system, clonal selection, local feature selection, short-term load forecasting, time series**

I. Introduction

Time series representing different phenomena and processes from industry, economy, demography, meteorology, etc., express various patterns. In general, there are four components to a time series: a trend, seasonal variations, cyclical variations, and irregular variations. A good example of time series expressing complex behavior is electricity demand in a power system. In Fig. 1 hourly load of the Polish power system is shown. This time series is nonstationary in mean and variance, it contains nonlinear trend, irregular component, and three seasonal periods: daily, weekly and annual. The daily and weekly profiles change during the year. The daily profile depends on the day of the week as well. These all features of load time series have to be captured by a flexible forecasting model.

The load time series forecasting in short horizons from minutes to days (short-term load forecasting; STLF) is essential for power system control and scheduling. The load forecast are necessary for energy companies to make important decisions related to planning of electricity production and transmission. As a basic driver of electricity prices the power system load should be forecasted precisely. The forecast accuracy translates to financial performance of the energy market participants such as electricity producers and suppliers, energy trading companies, transmission and storage system operators, investment firms and also financial institutions.

The complexity of the STLF problem and its importance caused the development of variety of forecasting models in recent decades. They can be roughly classified as conventional and unconventional models. The former include statistical analysis, regression methods and smoothing techniques such as autoregressive integrated moving average (ARIMA) [1] and exponential smoothing (ES) [2]. The latter employ newer computational methods such as artificial intelligence and machine learning ones [3], [4]. Some of the most popular of them are: neural networks, fuzzy inference systems, neuro-fuzzy systems, support vector machines and ensembles of models.

Seasonal ARIMA models can capture stochastic nature of the load time series and their multiple seasonal cycles. Exogenous variables can be included into the model. A limitation of ARIMA is its linear nature. Estimation of ARIMA parameters is usually considered subjective and

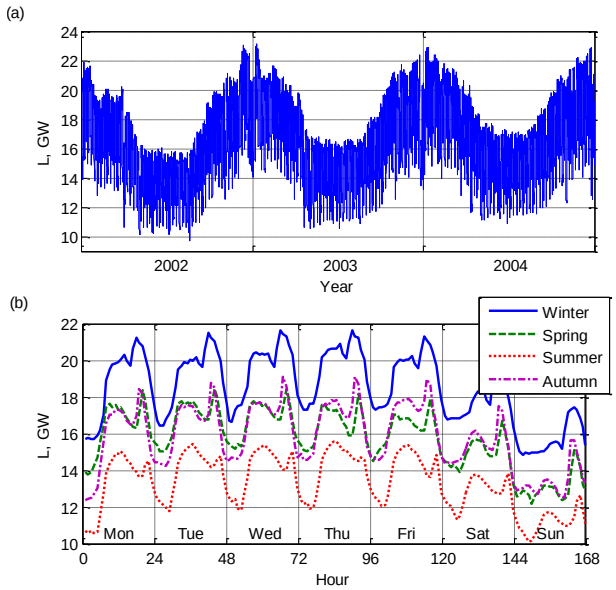

Fig. 1. The hourly electricity demand in Poland in three-year (a) and one-week (b) intervals.

This work was supported in part by the Polish Ministry of Science and Higher Education under Grant N N516 415338.

G. Dudek is with the Department of Electrical Engineering, Czestochowa University of Technology, 42-200 Czestochowa, Al. Armii Krajowej 17, Poland (e-mail: dudek@el.pcz.czest.pl).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

difficult to apply.

ES can express heteroscedastic time series with seasonal variability and nonlinear relationships between their terms but exogenous variables cannot be included into the model. An overparameterization and a large number of starting values to estimate are disadvantages of ES. Recently proposed exponential smoothing formulations applied for STLF are presented in [2].

Neural networks, widely used in STLF, have many attractive features such as: universal approximation property, learning ability, and robustness to noise in data. Disadvantages of neural networks include: overfitting, instability in training (sensitivity to the initial parameter values), difficulty in selection of an optimal architecture, weak extrapolation capacity and many parameters to estimate. Some recent algorithms to improve training of neural STLF models are presented and compared in [5]. The way of designing of STLF neural model to ensure an optimal generalization capacity in [6] is shown.

Fuzzy logic allows imprecise, incomplete and ambiguous information to be introduced into the STLF model. This is effective approach to deal with uncertainty. In neuro-fuzzy systems the knowledge for building if-then rule base is gained directly from data in learning process. But usually a structure of neuro-fuzzy system is complex and the number of parameters is large, so learning is difficult and does not guarantee convergence to the global minimum. Many successful applications have been reported on using fuzzy systems as STLF models, e.g. [7], where fuzzy logic is combined with wavelet transform and neural network or [8], where interval type-2 fuzzy logic systems are used to directly model and handle uncertainties.

Another useful computational intelligence tools for STLF are artificial immune systems (AIS). AIS are biologically inspired computation methods having many attractive features from machine learning and computational intelligence perspectives. These include [9]: self-organization and self-optimization, adaptation ability, learning from examples, distributed and parallel operation, pattern recognition and memorization, anomaly detection, multilayer structure and generalization capability.

The origin of AIS has its roots in the pioneering work of Farmer, Packard and Perelson [10], where theoretical immune network models were proposed to describe immune memory. Early works forming the basis of solid foundation for AIS focused on the immune network theory [11], and computer security (network intrusion detection, computer virus detection) [12], [13]. In the last years a diverse set of immune inspired algorithms have been developed to solve various computational problems. The four major concepts from immune systems underlie AIS algorithms [14]:

- negative selection,
- artificial immune networks,
- clonal selection and
- danger theory and dendritic cells.

Application areas that have been addressed by AIS can be summarized as [15]:

- learning (clustering, classification, recognition, robotic and control applications),
- anomaly detection (fault detection, computer and network security applications) and
- optimization (continuous and combinatorial).

AIS were used as STLF models [16], [17] (see AIS1 and AIS2 described in section V) and as learning methods for STLF models built on neural networks [18].

In this work the AIS with local feature selection (AISLFS) as a supervised learning regression algorithm for STLF is designed. The immune cells, antibodies (ABs), learn the time series patterns included in antigens (AGs). AGs are recognized by AB paratopes representing selected features of the input vector (time series fragment preceding the forecasted fragment). Each AB with its paratope represents a limited region of the input space corresponding to the hyperball defined in some subspace. AB recognition regions are shaped in a clonal selection procedure. This includes optimization of paratopes individually for each AB and corresponds to the local feature selection. Locally irrelevant or redundant features are omitted. ABs also learn output vectors representing forecasted fragments of the time series. The goal of AIS learning is to cover the input space by hyperballs defined in different subspaces in such a way that prediction capability of the system is maximized.

The novelty of AISLFS is an alternative representational abstraction: each AB defines recognition region in a different subspace. Moreover, a final decision is made collectively by many ABs, which are competent only in their own recognition regions (weak learners). New representation way and collective response of immune cells are consistent with theoretical perspectives presented by McEwan and Hart [19]. They noted that an epitope is a discontinuous region on the three-dimensional surface of a molecule (our energetic residues). It is not a predefined object but it becomes an epitope by virtue of binding to a receptor in the context of a particular interaction. This corresponds to an epitope definition in AISLFS (see Section II-B below). According to [19] the immune repertoire is not a population of centroids, prototypes, or support vectors, but an overcomplete dictionary of basis functions. Each cell receptor defines a different subspace of the original n-dimensional input space. The system is composed of many simple learners (immune cells) with weak representational capabilities. The search space for the immune repertoire is enriched to the space of classifiers (or regression models in our case), and the regression function becomes a weighted vote or averaged prediction amongst an ensemble of learners. As in boosting, a set of weak learners can be aggregated into an arbitrarily strong learning algorithm. An increase in representational power is achieved through the diversity of single learners (defined in low-dimensional subspaces) and an increase in stability through their integration. A motivation for dimensionality reduction, which is performed in AISLFS in local version, is a curse of dimensionality. This issue has been discussed by Stibor et al. [20]. In the context of AIS, where hyperballs define recognition regions, the problem is that the volume of a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

hyperball quickly approaches zero with increasing dimensionality of the space and any metric becomes increasingly meaningless as data points tend to become equidistant. Thus, low dimensional intuitions about distance and density are highly inadequate and undermine the very concepts that traditional AIS abstractions build upon.

In AISLFS AB can bind to many distinct epitopes (poly-recognition) and, similarly, AG can be bound by many ABs (poly-clonality). This idea, called degeneracy, derived from the works of experimental and theoretical immunologists [21], gains recent interest in both immunology and its computational abstractions and generates interest in the AIS community [22], [23]. Researchers embrace this degeneracy as an important feature of the immune system [24]. In AISLFS degeneracy allows the system to train immune memory and to generate collective response of the immune cells.

This paper is organized as follows. In Section II the idea of AISLFS for STLF is presented. Time series representation is described, how antibody and antigens are built is shown and AB adaptation process and prediction procedure are outlined. The detailed AISLFS algorithm for STLF is presented in Section III and discussed in Section IV. In Section V the proposed forecasting model is tested on real load data. It is compared with two others AIS-inspired STLF models as well as with neural networks, ARIMA and exponential smoothing. Finally, Section VI concludes the paper. The symbols that appear in the following description of the AISLFS algorithm are listed in Table I.

II. Idea of AISLFS for STLF

This work is a continuation of research on AISLFS. In [25] AISLFS was proposed as a classifier with unique feature: the local feature selection. For each region of the input vector space represented by an AB a separate subset of relevant features is created. This allows the recognition system to improve its recognition capacity when different features are important in different regions of the input space.

The local feature selection is inspired by the binding of an AB to an AG. This binding occurs between amino acid residues forming an epitope and a paratope. Only selected residues, so called energetic residues, take part in the binding. They correspond to the selected features. ABs are the recognition units with paratopes (corresponding to the subsets of selected features) formed in the learning process: immune memory creation process. Each AB has label with target output value (class symbol or function value). The final ABs with their paratopes form the immune memory. They correspond to the set of hyperballs defined in different subspaces. This hyperballs, called AB recognition regions, cover the input space in such a way that in each hyperball there are only training points having the same (in classification problems) or similar (in regression problems) labels. The biological inspirations behind AISLFS are further discussed in [25].

The similarities and differences between the proposed AISLFS for STLF and AISLFS classifier [25] can be summarized as follows:

- In both algorithms samples are represented by AGs and recognition units are represented by ABs. Both AGs and ABs have labels. In AISLFS classifier labels include class symbol, while in AISLFS for STLF they include a forecast fragment of a time series.
- The algorithm structures are similar. They include clonal selection loop which is performed for each AB. In both algorithms inside this loop the same searching process for finding best AB paratopes is implemented (tournament searching algorithm).
- Hypermutation operators which modify clone paratopes are similar for both algorithms. But in the implementation of AISLFS classifier described in [25] only one bit of a paratope is mutated. In AISLFS for STLF proposed in this work more bits can be mutated (depending on the parameter value, by which we can control a mutation intensity).
- Affinity measure is defined in the same way in both algorithms but the cross-reactivity threshold, on which it is based, has different meaning. In AISLFS classifier it depends on the classes of AB and neighboring AGs. In AISLFS for STLF it depends on the forecasting error estimated for each AG and AB.

TABLE I
List of symbols

Symbol	Description
$\Phi = \{(\mathbf{x}_i, \mathbf{y}_i)\}$	training set
Ω	AB paratope: the set of indices of selected features
Ξ	set of indices of ABs which recognition regions cover the AG
Ψ	set of indices of AGs in AB recognition region
δ	threshold error determining r
σ	parameter controlling the hypermutation range
τ	forecast horizon in days
D_k	dispersion of the time series elements in the period k
$L_{k,t}$	t-th load time series element in the period k
$\bar{L}_k$	mean load in the period k
N	number of training samples
P	AB power: $\lvert\Psi\rvert$
S	maximum number of the successive iterations without result improvement
X	input pattern space
Y	output pattern space
Z	number of clones
$a(\mathbf{p}_l, \mathbf{x}_j, \Omega_l)$	affinity of the l-th AB for the j-th AG
c	parameter adjusting r
$d(\mathbf{p}_l, \mathbf{x}_j, \Omega_l)$	distance in X between the l-th AB and j-th AG
m	number of bits swapping in hypermutation of a clone
n	total number of features
$\boldsymbol{p}$	element of AB: vector corresponding to a vector $\mathbf{x}$
$\boldsymbol{q}$	label of AB: vector corresponding to a vector $\mathbf{y}$
r	cross-reactivity threshold
$\mathbf{v}$	n-element binary vector corresponding to the paratope Ω
w_k	weight of activated k-th memory cell
$\mathbf{x}$	n-component input pattern
$\mathbf{y}$	n-component output pattern

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

- AB labels in AISLFS for STLF are not fixed, as in AISLFS classifier, but are calculated from labels of recognized AGs.
- The clones are evaluated using the same criteria in both algorithms.
- The model output is calculated differently in both algorithms. In AISLFS classifier this is an avidity depending on the affinities of activated memory ABs. In AISLFS for STLF it is calculated as a weighted average of the labels of activated memory ABs.

A. *Time Series Representation*

STLF is a regression problem $X \rightarrow Y$. An input vector $\mathbf{x}_i = [x_{i,1}\ x_{i,2}\ \ldots\ x_{i,n}]^T \in X = \mathbb{R}^n$ represents a vector of loads (L) in successive timepoints of the daily period k: $\mathbf{L}_k = [L_{k,1}\ L_{k,2}\ \ldots\ L_{k,n}]^T$. An output vector $\mathbf{y}_k = [y_{k,1}\ y_{k,2}\ \ldots\ y_{k,n}]^T \in Y = \mathbb{R}^n$ represents a vector of loads in successive timepoints of the forecasted daily period $k+\tau$: $\mathbf{L}_{k+\tau} = [L_{k+\tau,1}\ L_{k+\tau,2}\ \ldots\ L_{k+\tau,n}]^T$, where τ is a forecast horizon in days. Vectors $\mathbf{x}$ and $\mathbf{y}$ we call x-patterns and y-patterns, respectively, because they map load vectors $\mathbf{L}$ in specific ways. The components of input pattern $\mathbf{x}_k$ are defined as follows:

$$x_{k,t} = \frac{L_{k,t} - \bar{L}_k}{D_k}, \tag{1}$$

where: $k = 1, 2, \ldots, N$ – the daily period number, $t = 1, 2, \ldots, n$ – the time series element number in the period k, $L_{k,t}$ – the t-th load time series element in the period k, $\bar{L}_k$ – the mean load in the period k, $D_k = \sqrt{\sum_{l=1}^{n}(L_{k,l} - \bar{L}_k)^2}$ – the dispersion of the time series elements in the period k.

The input patterns $\mathbf{x}_k$ are normalized versions of the load vectors $\mathbf{L}_k$. They all have unity length, zero mean and the same variance. Note that the load time series which is nonstationary in mean and variance is represented by x-patterns having the same mean and variance. The trend and additional seasonal variations, i.e. the weekly and annual ones, are filtered. The x-patterns carry information only about the shapes of the daily load curves.

The output pattern $\mathbf{y}_k$ has components defined as follows:

$$y_{k,t} = \frac{L_{k+\tau,t} - \bar{L}_k}{D_k}, \tag{2}$$

where: $k = 1, 2, \ldots, N$, $t = 1, 2, \ldots, n$.
This transformation is similar to (1) but note that instead of coding variables $\bar{L}_{k+\tau}$ and $D_{k+\tau}$ determined for the day $k+\tau$, we use coding variables $\bar{L}_k$ and D_k determined for the day k. This is because the coding values for the day $k+\tau$ are not known in the moment of forecasting. Using the known coding values for the day k enables us to calculate the forecast of vector $\mathbf{L}_{k+\tau}$ when the forecast of pattern $\mathbf{y}_k$ is generated by the model. To do this transformed equation (2) is used:

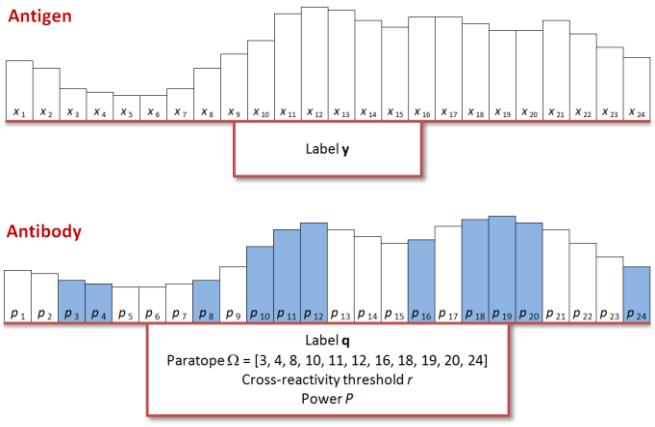

Fig. 2. AG and AB structures (n = 24). x- and p-vectors correspond to the chains of amino acids taking part in binding. Dark bars of p-vectors form a paratope. An epitope of AG consists the same components as a paratope.

$$\hat{L}_{k+\tau,t} = \hat{y}_{k,t} D_k + \bar{L}_k, \tag{3}$$

where $\hat{y}_{k,t}$ is the forecasted t-th component of the pattern $\mathbf{y}_k$.

By transforming the time series into x- and y-patterns we unify the input and output data. Now the relationship between input and output variables are simpler. This results in the simpler and more accurate forecasting model. Time series data representation using patterns is discussed more widely in [26].

B. *Antigens and Antibodies*

In AISLFS each pair of patterns ($\mathbf{x}_k$, $\mathbf{y}_k$) from the training set is represented by individual AG. Components of x-pattern correspond to the chain of amino acid residues of the protein building AG (i.e. x-pattern corresponds to the primary structure of the protein). Some of the x-pattern components form the AG epitope such as in nature some amino acid residues form an epitope in the tertiary structure of the protein (conformational epitope). An epitope is defined as a subset of x-pattern components which correspond to the AB paratope. One AG can have many epitopes as it can be bound by many ABs with different paratopes. The label of AG contains a target y-pattern. The AG population correspond to the training set $\Phi = \{(\mathbf{x}_k, \mathbf{y}_k) : k \in \Delta\}$, where Δ is the set of numbers of y-patterns representing the same day type {Monday, …, Sunday} as the forecasted y-pattern. We prepare training sets for each day type and then we construct the forecasting models separately for each day type because the values of coding variables $\bar{L}_k$ and D_k determining the levels and dispersion of y-patterns are dependent on the day of the week.

The immune system representatives are ABs. They recognize AGs and construct the forecast of y-pattern for new AGs with empty labels (representing query patterns). AB is composed of five elements: ($\mathbf{p}$, $\mathbf{q}$, Ω, r, P). Vectors $\mathbf{p}$ and $\mathbf{q}$ correspond to vectors $\mathbf{x}$ and $\mathbf{y}$, respectively. Vectors $\mathbf{p}$ and $\mathbf{q}$ of the k-th AB are initialized by k-th training example: $\mathbf{p}_k = \mathbf{x}_k$, $\mathbf{q}_k = \mathbf{y}_k$ (in the same way as the k-th AG). A vector $\mathbf{p}$ correspond to the chain of amino acid residues of the protein

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

building AB. A vector **q** is saved in the AB label. Ω is a paratope. This is a set containing numbers of selected components of the p-vector. *r* is a cross-reactivity threshold. *P* is an AB power (representativeness) expressing how many training AGs it recognizes. Vectors **p** are fixed but vectors **q**, paratopes Ω, thresholds *r* and powers *P* are modified during training. Structures of AG and AB in Fig. 2 are shown.

Elements **p**, Ω and *r* define the recognition region of AB. This is a hyperball of radius *r* centered at point $\mathbf{p}'=[p_t]_{t\in\Omega}$. So the hyperball is defined in a subspace of the *n*-dimensional input feature space *X*. This subspace contains dimensions which are saved in the set Ω. We call this subspace the Ω-subspace. (Figs. 3 and 4 in [25] clarify AB recognition regions in Ω-subspaces.)

C. Antibodies Adaptation and Antigen Label Prediction

Recognition regions of ABs are adapted during training (paratopes Ω and cross-reactivity thresholds *r* change) to the population of AGs (training sample). These adaptations are made using hypermutation in the clonal selection loop (Algorithm 1), where each AB generates clones with modified paratopes, cross-reactivity thresholds, labels and powers. In each iteration of the clonal selection loop the best clone is selected. It becomes a parent AB in the next iteration. The best clone is that one which recognizes the highest number of training AGs. AG is considered to be recognized by a clone when its epitope is similar to the clone paratope (this similarity is measured using affinity) and also labels of both, AG and the clone, are similar. These similarities (in both Ω-subspace and *Y* space) between clone and many AGs means that the clone is representative and is able to predict labels of many AGs lying in its recognition region. Such clones are searched in the clonal selection loop and recorded as memory cells.

As a result of training the population of immune memory cells is constructed. An AB from the immune memory recognizes as many training AGs as possible. In the test phase new AG with empty label is recognized by some ABs, i.e. it falls into the recognition regions of these ABs. The AG label is predicted from the labels of activated ABs, their powers and affinities in Ω-subspaces.

III. Algorithm of AISLFS for STLF

In this section the algorithm of AISLFS for STLF is described in detail. The symbols that appear in the following description are listed in Table I. The flowchart of the algorithm is given in Fig. 3.

Step 1. The AG population corresponds to the set of training samples from Φ: $PopAG = \{AG_j=(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{N}$.

Step 2. The AB population is composed of *N* ABs: $PopAB = \{AB_k=(\mathbf{p}_k, \mathbf{q}_k, \Omega_k, r_k, P_k)\}_{k=1}^{N}$. The p- and q-patterns of ABs are created by copying training samples: $\mathbf{p}_k = \mathbf{x}_k$ and $\mathbf{q}_k = \mathbf{y}_k$. This method of initialization prevents putting ABs in empty regions without AGs. The AB paratopes Ω_k can be initialized in three ways:

- using all features (all components of vector **p**),
- using randomly selected features or
- using features selected as important in previous runs of the algorithm for similar forecasting task.

The similar task is that one, in which to train the immune memory the training set is used that contains mostly the same AGs as in the current forecasting task. For example in the task of load forecast for some Wednesday we can use the initial AB population with paratopes Ω_k optimized in the forecasting task for the last Wednesday.

The cross-reactivity thresholds and AB powers do not

```
PopAB = {AB_k=(p_k, q_k, Ω_k, r_k, P_k)}_{k=1}^{N}
PopAG = {AG_j=(x_j, y_j)}_{j=1}^{N}
for k = 1 to N do
  pAB = AB_k  //parent AB
  //clonal selection loop
  while not stop condition do
    for l = 1 to Z do
      Cl_l = pAB  //l-th clone
      Ω_l = hypermutation(Ω_l)
      r_l = CR_treshold(similarity(Cl_l, PopAG))
      a_l = affinity([d(p_l, x_j, Ω_l)]_{j=1}^{N}, r_l)
      Ψ_l = {j ∈ {1,2,...,N} : a(p_l, x_j, Ω_l)>0}
      q_l = label(a_l, {y_j : j ∈ Ψ_l})
      P_l = |Ψ_l|
    end
    bCl = winner({P_l}_{l=1}^{Z},{|Ω_l|}_{l=1}^{Z}) //best clone
    pAB = bCl
  end
end
```

Algorithm 1: Pseudocode of AISLFS clonal selection loop.

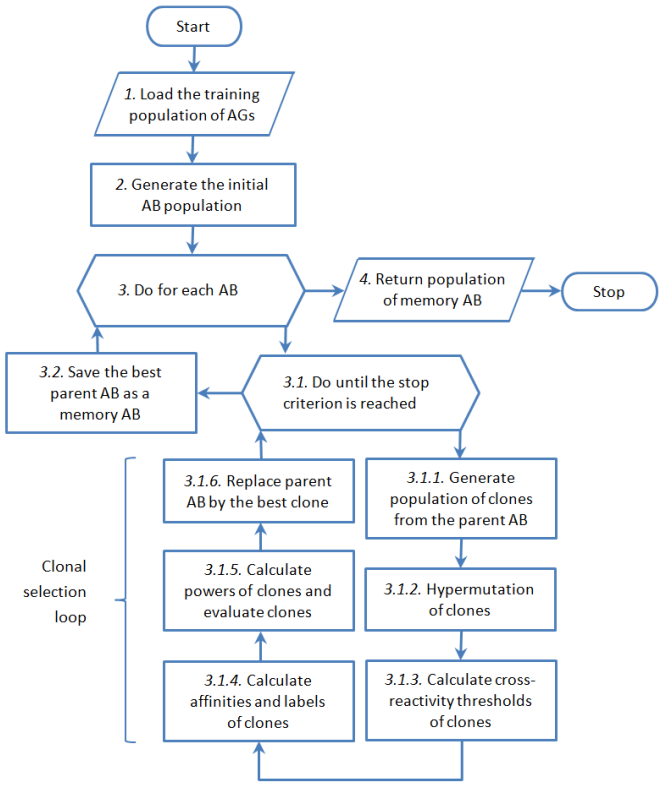

Fig. 3. Flowchart of the AISLFS for STLF.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

require initialization.

Step 3.1. In this loop the clones of *k*-th AB are generated, mutated and evaluated. The best clone becomes the parent AB in the next iteration. The goal is to find the best paratope of the *k*-th AB. This is an combinatorial optimization problem in which the subset of features is searched for which the criterion function described in step 3.1.5 is maximized. We employ the tournament searching algorithm [27], which is a stochastic global search method, to search the paratope space. In simulation study reported in [27] the tournament searching outperformed genetic algorithm and simulated annealing as well as deterministic algorithms in the feature selection problem.

Tournament searching is built into the clonal selection loop. It uses hypermutation to generate new *Z* candidate solutions (clones with different paratopes) from the parent solution (parent AB). The best clone (tournament winner selected according to (11)) replaces the parent AB in the next iteration, even if it is worse than the parent AB. This allows the solution to escape from local extrema of the criterion function (this is discussed further in Section IV).

The clonal selection loop is stopped when during last *S* iterations no result improvement is observed.

Step. 3.1.1. The population of clones is generated from the parent AB. The number of clones, $Z \geq 1$, is dependent on the paratope space size, which is $2^n - 1$. This space contains all binary vectors **v** of size *n*. Ones in these vectors corresponds to elements of Ω-sets. So the *k*-th AB paratope can be represented by vector $\mathbf{v}_k = [v_{k,1}\ v_{k,2}\ \ldots\ v_{k,n}]$, where:

$$v_{k,t} = \begin{cases} 1, & \text{if } t \in \Omega_k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The parameter *Z* controls the sampling accuracy of the neighborhood of the parent AB paratope, i.e. global-local search properties of the algorithm. The neighborhood is defined here as the set of all vectors **v** reachable by hypermutation of the parent AB paratope. *Z* can be constant or increasing during training, which results in a change in the character of searching the AB paratope neighborhood: from random and global for small *Z* towards deterministic and local for greater *Z*. These issues will be more explained in Section IV.

Step 3.1.2. The hypermutation changes the paratope of each clone. It operates on a vector **v** corresponding to the clone paratope. It swaps $m > 0$ bits in **v** randomly selected from a uniform distribution. The number of bits swapping *m* is determined randomly using normal distribution: $m \sim \lceil |N(0,\sigma)| \rceil$. If drawn $m > n$, then we assume the new value of *m* as $m - \lfloor (m-1)/n \rfloor n$. If drawn $m = 0$, we assume $m = 1$. The parameter σ controls the distribution of a discrete variable *m*, i.e. the range of mutation. For a given value of σ the mutation probability of *m* bits depends on *n*. It can be calculated from the cumulative distribution function (cdf) for $N(0,\sigma)$. The probability of mutation of one bit $P(m = 1) = 2\cdot[\text{cdf}(0) - \text{cdf}(-1)] + 2\cdot[\text{cdf}(-n) - \text{cdf}(-n-1)] + 2\cdot[\text{cdf}(-2n) - \text{cdf}(-2n-1)] \ldots = 2\sum_{i=0}^{\infty}[\text{cdf}(-i\cdot n) - \text{cdf}(-i\cdot n - 1)]$. Generally, a probability of mutation of $m \in [1, n]$ bits can be calculated from: $P(m) = 2\sum_{i=0}^{\infty}[\text{cdf}(-i\cdot n - m + 1) - \text{cdf}(-i\cdot n - m)]$. For example, when $\sigma = 1.4826$ and $n = 4$, we get: $P(m = 1) = 0.5 + 6.23\cdot10^{-3} + 6.69\cdot10^{-8} + \ldots = \sim 0.51$, $P(m = 2) = 0.3227 + 6.93\cdot10^{-4} + 1.26\cdot10^{-9} + \ldots = \sim 0.32$, $P(m = 3) = 0.1343 + 4.95\cdot10^{-5} + 1.52\cdot10^{-11} + \ldots = \sim 0.13$, and $P(m = 4) = 0.0360 + 2.27\cdot10^{-6} + 1.17\cdot10^{-13} + \ldots = \sim 0.04$. When $\sigma \to 0$, then $P(m = 1) \to 1$, and $P(m > 1) \to 0$ (in this case for each clone among *Z* generated we chose randomly different bit to swap). When $\sigma \to \infty$, then $P(m = 1) = P(m = 2) = \ldots = P(m = n) \to 1/n$.

After hypermutation the clone paratopes are randomly modified to an extent dependent on σ.

Step 3.1.3. The *l*-th clone cross-reactivity threshold r_l is determined after the clone mutation. The idea is that the clone recognition region should cover as much AGs as possible with **x** and **y** vectors similar to the clone **p** and **q** vectors, respectively. The method of determining the cross-reactivity threshold needs the population of AG to be split into two classes. Let us assume that *l*-th clone is derived from the *k*-th parent AB. This AB was initiated by *k*-th AG having in its label pattern $\mathbf{y}_k$, which encodes load vector $\mathbf{L}_{k+\tau}$. The *j*-th AG is assigned to class 1, if it can forecast vector $\mathbf{L}_{k+\tau}$ with error (MAPE) not higher than δ. The forecast is calculated from the pattern $\mathbf{y}_j$ (included in the *j*-th AG label) decoded using coding variables for *k*-th pattern **y** ($\bar{L}_k$ and D_k). So the class 1 contains such AGs from *PopAG* for which:

$$100\frac{1}{n}\sum_{t=1}^{n}\left|\frac{L_{k+\tau,t} - \hat{L}_{k+\tau,t}}{L_{k+\tau,t}}\right| \leq \delta \tag{5}$$

where: $\hat{L}_{k+\tau,t} = y_{j,t}D_k + \bar{L}_k$, $L_{k+\tau,t}$ is the *t*-th component of the load vector $\mathbf{L}_{k+\tau}$ encoded in the label of *k*-th AB, $y_{j,t}$ is the *t*-th component of y-pattern from the label of *j*-th AG (which is classified), and δ is a threshold error (MAPE).

The AGs which do not meet condition (5) are assigned to class 2. Note that similarity between the *j*-th AG and *l*-th clone is not measured directly between their **y** and **q** patterns but between load vectors encoded in these patterns. This allows us to control the acceptable real forecast error (δ).

Let us define the distance measure between vector **p** of the *l*-th clone and vector **x** of the *j*-th AG in Ω_l-subspace:

$$d(\mathbf{p}_l, \mathbf{x}_j, \Omega_l) = \left(\sum_{t\in\Omega_l} | p_{l,t} - x_{j,t} |^e\right)^{1/e} \tag{6}$$

where $e = 1$ for the Manhattan metric, and $e = 2$ for the Euclidean metric.

Note that the distance is calculated by taking into account only these components of **p** and **x** vectors which are in the paratope Ω_l.

The *l*-th clone cross-reactivity threshold is calculated as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

$$r_l(\Omega_l) = d(\mathbf{p}_l, \mathbf{x}_A, \Omega_l) + c[d(\mathbf{p}_l, \mathbf{x}_B, \Omega_l) - d(\mathbf{p}_l, \mathbf{x}_A, \Omega_l)] \quad (7)$$

where: B is the index of the nearest AG from class 2 to the l-th clone (in Ω_l-subspace), A is the index of the farthest AG from class 1 to the l-th clone, such that $d(\mathbf{p}_l,\mathbf{x}_A,\Omega_l) < d(\mathbf{p}_l,\mathbf{x}_B,\Omega_l)$, and $c \in [0, 1)$ is a parameter allowing us to adjust the cross-reactivity threshold in the range [$r_{l\min} = d(\mathbf{p}_l,\mathbf{x}_A,\Omega_l)$, $r_{l\max} = d(\mathbf{p}_l,\mathbf{x}_B,\Omega_l)$). This is illustrated in Fig. 4.

The partitioning of the AG population into two classes is performed for each AB independently. The result is the class distribution table (Fig. 5) showing classes of AGs for each AB, where $c_{j,l}$ is the class of j-th AG for the l-th AB. Obviously, $c_{j,j} = 1$. Vector $\mathbf{c}_j = [c_{j,1}\ c_{j,2}\ \ldots\ c_{j,N}]$ expresses the class distribution for the j-th AG.

The cross-reactivity thresholds are determined in different Ω-subspaces for different ABs. Fig. 3 in [25] shows examples of AB recognition regions in 2D space.

AGs, which are contained in the recognition region of the l-th clone, i.e. AGs lying in the hyperball of radius r_l centered at $\mathbf{p}_l' = [p_{l,t}]_{t\in\Omega_l}$, are deemed to be recognized by this clone. Such AGs are similar to the clone not only in the input feature space but also in the output load space (note that the clone recognition region covers only class 1 AGs).

The threshold error δ affects the partitioning of AGs into classes. Smaller values of δ result in fewer AGs in class 1 and smaller recognition regions of ABs. This introduces a high specialization of ABs, which recognize smaller number of AGs with a greater degree of similarity. But in such case new AGs can be unrecognized by AB population being outside of the AB recognition regions. Too large value of δ makes that AG is recognized by many ABs. This leads to an increase in the model bias and translates to greater forecasting errors. Thus the parameter δ allows us to control the bias-variance tradeoff. Its value is adjusted experimentally dependent on the data.

Step 3.1.4. The affinity of the l-th clone for the j-th AG is defined as follows:

$$a(\mathbf{p}_l, \mathbf{x}_j, \Omega_l) = \begin{cases} 0, & \text{if } d(\mathbf{p}_l, \mathbf{x}_j, \Omega_l) \ge r_l(\Omega_l) \text{ or } r_l(\Omega_l) = 0 \\ 1 - \dfrac{d(\mathbf{p}_l, \mathbf{x}_j, \Omega_l)}{r_l(\Omega_l)}, & \text{otherwise} \end{cases} \quad (8)$$

where $a(\mathbf{p}_l, \mathbf{x}_j, \Omega_l) \in [0, 1]$ and $r_l(\Omega_l)$ is the cross-reactivity threshold of the l-th clone in Ω_l-subspace.
Note that the affinity is zero for the AG lying outside of the clone recognition region. For AG lying inside this region it is a linearly decreasing function of distance (6), from 1 for $\mathbf{x}_j = \mathbf{p}_l$, to 0 for AG lying on the border of the recognition region ($d(\mathbf{p}_l, \mathbf{x}_j, \Omega_l) = r_l(\Omega_l)$).

The label of the l-th clone derived from the k-th AB contains the forecast of the pattern $\mathbf{y}_k$ paired with $\mathbf{x}_k = \mathbf{p}_l$. This forecast is calculated as the average value of labels of AGs, which are recognized by this clone, weighted by affinities:

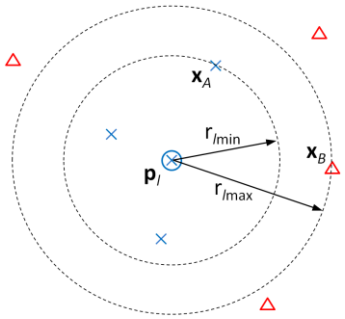

Fig. 4. The cross-reactivity threshold range ($\Omega_l = \{1, 2\}$, AGs from class 1 are marked by crosses, AGs from class 2 are marked by triangles, and the l-th clone is marked by a circle).

AG \ AB	1	2	…	N
1	$c_{1,1}$	$c_{1,2}$	…	$c_{1,N}$
2	$c_{2,1}$	$c_{2,2}$	…	$c_{2,N}$
…	…	…	…	…
N	$c_{N,1}$	$c_{N,2}$	…	$c_{N,N}$

Fig. 5. Table of AG class distribution.

$$\mathbf{q}_l = \frac{\sum_{j\in\Psi_l} a(\mathbf{p}_l, \mathbf{x}_j, \Omega_l)\mathbf{y}_j}{\sum_{j\in\Psi_l} a(\mathbf{p}_l, \mathbf{x}_j, \Omega_l)} \quad (9)$$

where Ψ_l is a set of indices of AGs which lie in the recognition region of the l-th clone.
A clone represents a set of AGs, so its label is created based on the labels of AGs from this set. The greatest weight in the mean (9) has AG which initiated this clone. The AG weights decrease linearly depending on the distance between $\mathbf{p}_l$ a $\mathbf{x}_j$ measured in Ω_l-subspace.

Step 3.1.5. The number of AGs recognized by the clone, P, is a measure of its representativeness. The clones with high values of P represent typical patterns in terms of the typical shape of the daily curve as well as typical relationship between x- and y-patterns. AG encoding atypical patterns or atypical relationship between input and output patterns is represented by a separate AB, which has only this AG in its recognition region.

The number of AGs in the recognition region of the l-th clone, dependent on its paratope Ω_l, is the evaluation measure of the clone:

$$P_l = |\Psi_l(\Omega_l)| \to \max \quad (10)$$

This measure, called a power, is maximized in the clonal selection loop. The algorithm in step 3.1.6 selects the best clone from the current population of clones. If several clones have the same maximal power $P_{\max}$, the winner is one of them with the smallest paratope $\min|\Omega|$. This condition is an additional criterion for assessing clones, leading to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

minimization of the feature number. If several clones have the same maximal power $P_{\max}$ and paratope size $\min|\Omega|$, the winner is chosen randomly among them. Thus, the winner status is given by the formula:

$$z_l = \begin{cases} 1, & \text{if } l \in \Theta \text{ and } |\Theta| = 1 \text{ or } l = \xi, \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

where $z_l = 1$ for the winner, $\Theta = \arg\min_{l \in \Theta'} \{|\Omega_l|\}$, $\Theta' = \arg\max_{l=1,2,\ldots,Z} \{P_l\}$, and ξ is the randomly selected clone index from the set Θ.

Step 3.1.6. The clone winning in the *i*-th iteration of the clonal selection loop replaces the parent AB, even if it is worse than the parent AB ("worse" means that the winner has lower power than the parent AB, or the same power but bigger paratope size). This allows the algorithm to escape from the traps of local extrema.

Step 3.2. In this step the best parent AB generated in the clonal selection loop is selected and saved. This is one of the parent AB which covers the most AGs and has the smallest paratope. This AB becomes a memory cell.

Step 4. A result of training is a population of memory cells, which cover the input space X in different Ω-subspaces. They are used for recognition new AGs and to predict AG labels.

After AISLFS training the immune memory is ready for prediction of new instances. This process corresponds to the secondary immune response when a new AG is presented to the trained immune memory. The test AG with empty label, representing input pattern $\mathbf{x}^*$ (query pattern) is recognized by some of the memory cells having non-zero affinity for it. Let Ξ be the set of indices of these memory ABs. The label of the test AG is calculated as a weighted average of the labels of activated memory ABs. Two components of weights are introduced. The first one expresses the affinity of the memory AB for the test AG, and the second one expresses the AB power. The test AG label, i.e. the forecasted y-pattern corresponding to input pattern $\mathbf{x}^*$, is calculated as follows:

$$\hat{\mathbf{y}} = \sum_{k \in \Xi} w_k(\mathbf{p}_k, \mathbf{x}^*, \Omega_k)\mathbf{q}_k \tag{12}$$

where Ξ is a set of indices of ABs which recognition regions cover the test AG, and

$$w_k(\mathbf{p}_k, \mathbf{x}^*, \Omega_k) = \frac{P_k a(\mathbf{p}_k, \mathbf{x}^*, \Omega_k)}{\sum_{l \in \Xi} P_l a(\mathbf{p}_l, \mathbf{x}^*, \Omega_l)} \tag{13}$$

Fig. 6 illustrates the test procedure. In this figure the highest affinity for the test AG has AB_3 (0.50), the lower affinity has AB_2 (0.25) and the lowest affinity has AB_1 (0.03). The AB powers are: $P_1 = 9$, $P_2 = 6$, and $P_3 = 3$. The AB weights are: $w_1 = 0.082$, $w_2 = 0.459$ and $w_3 = 0.459$. Thus, the biggest impact on the response have the second and third ABs.

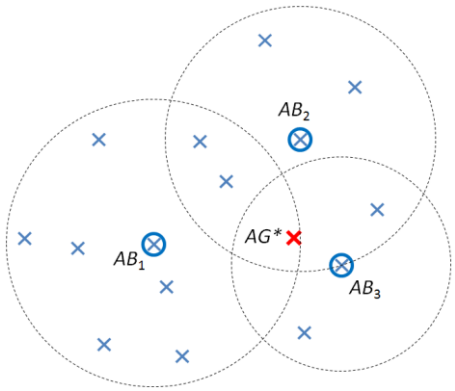

Fig. 6. AG recognition by ABs in the test procedure for $\Omega_k = \{1, 2\}$, $k = 1, 2, 3$. Thin crosses are training AGs, thick cross is the test AG, circles are ABs.

IV. Discussion

The proposed AISLFS searches the paratope space. The goal is to find for each AB the best subspace of X (Ω-subspace), in which this AB represents most AGs (has the highest power). The AGs represented by AB are its nearest neighbors in Ω-subspace and have similar labels to AB initial label. The similarity in AB and AG labels means that the load vector encoded in AG label can forecast load vector initially encoded in AB label with error not higher than the assumed threshold error δ. All AGs represented by AB lie in the AB recognition region, i.e. inside the hyperball of radius r centered at $\mathbf{p}' = [p_{l,t}]_{t \in \Omega}$. The paratope Ω is found individually for each AB in the searching procedure. The radius r is also adjusted individually for each AB, so that the AB recognition region covers in Ω-subspace as much AGs similar to AB in labels as possible. The final label of AB is calculated as the weighted mean of labels of all AGs represented by this AB.

The searching procedure implemented in the clonal selection loop is stochastic to avoid local extremum traps. The population of Z clones is generated from the parent AB. The paratope of each clone is changed using hypermutation operator. The best clone is selected, i.e. that one with the highest power. If more than one clone has the same maximal power, that one is selected with the smallest paratope. The winning clone replaces the parent AB in the next iteration, even if it is worse than the parent AB (this enables the searching process to escape from local extrema). The searching process is illustrated in Fig. 7. The goal of this process is to create the memory AB. The best parent AB generated in the clonal selection loop (parent ABs are marked with crosses in Fig. 7) becomes one of the memory ABs.

In the clonal selection loop the solution space is searched locally in the neighborhood of the solution represented by the parent AB using hypermutation operator. The neighborhood has a probabilistic nature, i.e. the reachability of the points from that neighborhood is not the same. The transition probability to the certain neighboring point is dependent on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

the mutation range controlled by the parameter σ. Reachability of the points, which differ from the solution represented by the parent AB with increasing number of bits, decreases monotonically at a rate dependent on the inverse of σ.

Two criteria are used for evaluation of clones: clone power, which is maximized and clone paratope size, which is minimized. The two-criterion optimization implemented in the clonal selection loop is not Pareto optimization, because here the criteria are not equivalent. The primary criterion is the clone power, and the paratope size is a secondary criterion used when there is more than one clone with power $P_{\max}$. Both criteria lead to the better coverage of the X space by ABs (this is discussed in Section IV.C in [25]) and to improvement in generalization ability of the model.

In the proposed AISLFS the final decision on the forecast is taken collectively by the activated memory ABs (ensemble of ABs). This ensemble of ABs is local: for different AGs different ensembles are formed composed of memory ABs which cover the recognized AG in different subspaces. This is similar to the method of combining component decision models called the random subspace method [28], used in the random forests. Differences between these two approaches are that the subspaces in the proposed model are not random, but optimized using some criteria, and the component models (memory ABs) are locally competent (in random forest approach each decision tree is used as a globally competent model). Due to combining responses of the component models we improve generalization and stability of the final model.

The AISLFS has five parameters (hyperparameters):

- clone population size, Z,
- the number of iterations without improvement of results as a stop criterion for the clonal selection loop, S,
- the threshold error determining the cross-reactivity threshold, $\delta \geq 0$,
- the parameter adjusting the cross-reactivity threshold, $c \in [0, 1)$,
- the width of the normal distribution controlling the hypermutation range, $\sigma > 0$.

The clone population size, Z, determines the accuracy of searching the neighborhood of the current base point (parent AB paratope) and, hence, directs the search. The search process changes the character from the global and random one for small values of Z, to the local and more deterministic one for increasing values of Z. When $Z = 1$ the searching process comes down to a random walk: only one candidate point (clone) is generated from the base point, and it replaces the base point regardless of its evaluation. On the other hand, when the population of clones represents all points reachable (maximal value of Z), we obtain hill climbing algorithm, sensitive to local extrema. The parameter Z is dependent on the size of the solution space. It can be constant or increasing during searching. In the later case the solution space is searched globally in the initial stage, and locally in the final stage.

The larger values of S increase the chances to leave the local extreme, but increase the computation time as well. The value of S depends on the size of the solution space and the clone population size.

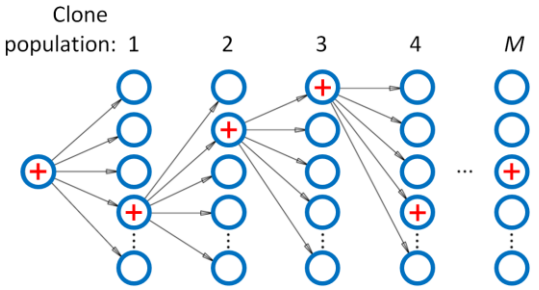

Fig. 7. The solution space searching in the clonal selection loop.

The parameters δ and c determine the cross-reactivity threshold r, and hence the bias-variance tradeoff of the model. For smaller values of these parameters the AB recognition regions decrease, which results in reduction in bias and increase in variance. ABs with too small recognition regions do not cover sufficiently the X space, which results in increase in the number of unrecognized AGs. On the other hand, increase in the values of δ and c provides to better coverage of the X space, but the bias increases. The values of these two parameters should be selected experimentally for a given forecasting task.

The parameter σ controls the hypermutation range. For higher values of σ the paratopes of clones differ more from the paratope of the parent AB. The jumps in the solution space are the smallest for $\sigma \to 0$, and the bigest for $\sigma \to \infty$. Thus, the σ value as well as the Z value decides about the exploration-exploitation tradeoff of the algorithm.

The output of AISLFS is the forecasted y-pattern $\hat{\mathbf{y}}$ corresponding to query pattern $\mathbf{x}^*$. But the model can also forecast only one component of y-pattern. So instead of MIMO model we have MISO model. In such case the model is optimized to get the best accuracy for y_t. The algorithm described above does not change. The only difference is that instead of $\mathbf{q}$ and $\mathbf{y}$ in (9) and (12) we use q_t and y_t, respectively.

V. Simulation Study

In this section training and optimization of AISLFS is presented and its performance is studied on four real STLF problems. AISLFS results are compared with results achieved by two other AIS-based models as well as neural networks, ARIMA and exponential smoothing.

A. Training and Optimization of AISLFS for STLF

The task is to forecast the hourly load of the Polish power system at hour t = 1, 6, 12, 18, 24 for the next day (τ = 1). The data are from the period 2002-2004 (see Fig. 1; these data can be downloaded from the website http://gdudek.el.pcz.pl/varia/stlf-data). The test period covers successive days of January 2004 (without atypical January 1st) and July 2004. Thus we have (30 + 31)·5 = 305 forecasting tasks (the tasks are marked

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

by symbols: hour/day/year, e.g. 12/July 1/2004). For each forecasting task (test sample) the training samples are selected individually from the period from January 1st, 2002 to the day preceding the forecasted day. The AISLFS in MISO version is trained for each forecasting task.

To select the best values of hyperparameters we change them according to the following steps:

1. $\sigma \to 0$, $\sigma = 0.7803, 1.1882, 1.9069, 3.9437$ and $\sigma \to \infty$, at $c = 1$ and $\delta = 2$,
2. $c = 0, 0.25, 0.5, 0.75, 1$, at $\delta = 2$ and the optimal value of σ selected in step 1,
3. $\delta = 1, 1.25, \ldots, 4$, at the optimal values of σ and c selected in steps 1 and 2, respectively.

Ranges and densities of parameters were selected based on previous experiments. Parameters Z and S were constant: Z = round(n/3) and S = 10. The successive values of σ considered in step 1 correspond to the following probabilities of one bit mutation $P(m = 1)$: 1, ~0.8, ~0.6, ~0.4, ~0.2, and 1/n (the probabilities of mutations of 2, 3, ..., n bits are determined from the normal distribution with standard deviation σ; see step 3.1.2 in Section III). The Euclidean metric was used as a distance measure between vectors **p** and **x** (6). The mean absolute percentage error (MAPE) was used as the forecasting error (MAPE is traditionally used as an error measure in STLF). The quality of the model during training was measured on validation samples, i.e. five pairs of patterns ($\mathbf{x}_i$, $y_{i,t}$) from the training set Φ for which $\mathbf{x}_i$ belongs to the set of five nearest neighbors of the query pattern $\mathbf{x}^*$. So the model hyperparameters are estimated on the most similar patterns to the query pattern.

The values of parameters σ and c varying in ranges specified above did not significantly affect the validation error: MAPE ranged from 1.26 to 1.30. Also error dispersion did not show greater sensitivity to the value of σ and c. The number of unrecognized AGs decreased with increasing c from 2.19 to 1.09%.

The most favorable values of the error threshold δ which gave the lowest MAPE (from 1.25 to 1.27), were within the range of 1.5 to 3.0. At lower δ some AGs remained unrecognized. The value of $\delta \geq 2$ provided recognition of more than 99% validation AGs.

It should be noted that changes in the model hyperparameters do not cause drastic changes in the forecast errors, which is very valuable property. For further experiments described in this section it was selected: σ = 1.9069, c = 1 and δ = 2.

Errors for the test samples $MAPE_{tst}$ averaged over 30 training sessions are given in Table II. The average number of features forming AB paratopes was 7, which gives a compression ratio 73%. The frequencies of features in paratopes of immune memory ABs in Fig. 8 are shown.

Fig. 9 shows the query pattern $\mathbf{x}^*$ for the forecasting task 12/July 1/2004 (Thursday) and paratopes of all activated memory ABs. The paratopes contain 7 out of 24 features on average. The smallest paratope (AB78) included only two features, the largest one (AB67) 14 features. The average

TABLE II
FORECAST ERRORS AND THEIR INTERQUARTILE RANGES FOR MISO MODEL

Hour	January		July		Mean	
	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}
1	1.12	1.02	1.11	1.11	1.12	1.06
6	1.52	1.45	0.92	0.86	1.23	1.06
12	1.57	1.03	0.93	0.79	1.24	0.84
18	1.42	1.53	0.74	0.78	1.07	0.95
24	1.62	1.26	1.13	1.00	1.37	1.22
Mean	1.45	1.26	0.97	0.91	1.21	1.03

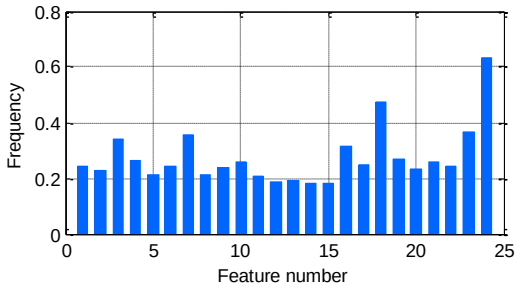

Fig. 8. Frequences of features (components of **x**) in AB paratopes Ω.

number of the clonal selection loop iterations in this forecasting task was 30. The figure also shows the AB weights (13). AB powers, cross-reactivity thresholds, affinities, labels and components of vectors **p** forming the paratopes of memory ABs created for this forecasting task in Fig. 10 are presented. The ABs in this case correspond to the learning samples which represent the daily load curves of the successive Wednesdays (x-patterns) and Tuesdays (y-patterns) from history. As we can see from this figure the activated ABs represent daily load curves from the period immediately preceding the forecasted day and from periods shifted in time one or two years back. This is because the days from these periods have similar load shapes. Daily load curves from winter periods showing greater dispersion than curves from summer periods are represented by ABs with lower powers (see Fig. 10 (a)).

In Fig. 10 (e) the paratopes of memory ABs are visualized. Black squares in this diagram indicate features selected to paratopes. As paratope size is minimized in the secondary criterion, there is many white squares corresponding to inactive features. Smaller paratope increases the AB power. So, the AG is recognized by as many memory ABs as possible. Similarly to ensembles of weak learners, increasing number of recognition units (ABs) provide to improvement in generalization ability of the model and higher precision (smaller variance).

The highest bar in the Figs. 10 (b) and (d) relates to AB50 which vector **p** represents atypical day: January 1, 2003. The power of this AB is 1. It was assumed during training that if the AB contains only one AG in its recognition region, which means that this AG is an outlier, the clonal selection loop is skipped and the paratope of this AB contains all the p-vector components (note only black squares for AB50 in Fig. 10 (e)).

Fig. 11 illustrates how the forecast is calculated from the labels and weights of activated ABs. Each activated AB is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

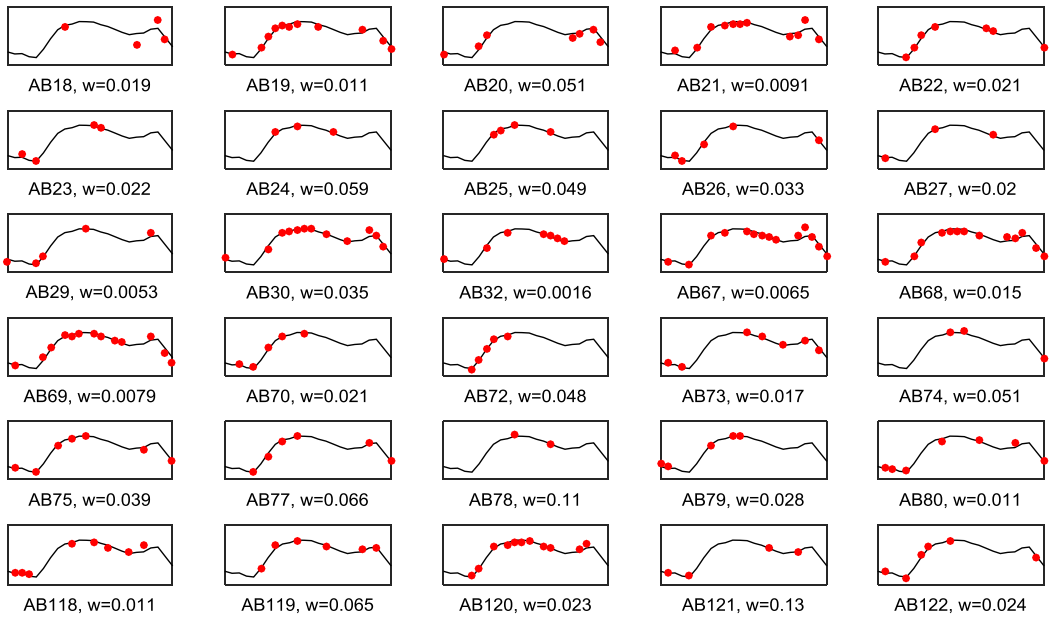

Fig. 9. Query pattern $\mathbf{x}^*$ for the forecasting task 12/July 1/2004 (solid line) and paratopes of all activated ABs (points).

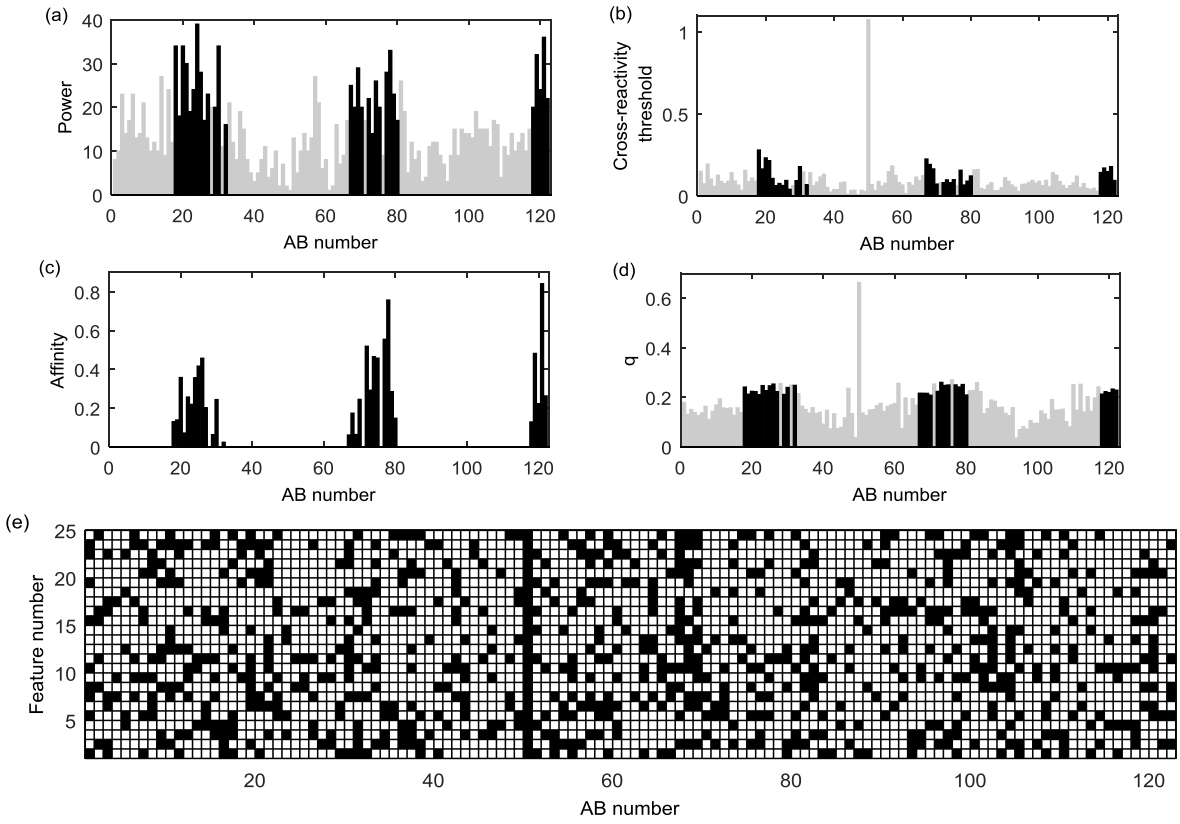

Fig. 10. Immune memory for forecasting task 12/July 1/2004 (black bars indicate activated ABs, black squares in (e) indicate features selected to paratopes).

marked by point (q_k, w_k) in this figure. The x-coordinate of the centroid of points (q_k, w_k), $k \in \Psi$, is the forecasted value for the query pattern $\mathbf{x}^*$.

In Table III errors for MIMO version of the AISLFS are presented. In this case the output of the forecasting model is vector $\mathbf{y}$. As can be seen from this table the errors in MIMO case are not worse than in MISO case.

B. Comparative studies of AISLFS with other models

In this section AISLFS is examined in real STLF problems on four time series:

- PL: time series of the hourly load of the Polish power system from the period of 2002–2004 (this is the same time series as in the previous section). The test sample

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

includes data from 2004 with the exception of 13 atypical days (e.g. public holidays),

- FR: time series of the half-hourly load of the French power system from the period of 2007–2009. The test sample includes data from 2009 except for 21 atypical days,
- GB: time series of the half-hourly load of the British power system from the period of 2007–2009. The test sample includes data from 2009 except for 18 atypical days,
- VC: time series of the half-hourly load of the power system of Victoria, Australia, from the period of 2006–2008. The test sample includes data from 2008 except for 12 atypical days.

On the basis of previous simulations it was assumed: σ = 1.9069 and c = 1. Error threshold δ was searched with the step of 0.25 in ranges . The proposed model in MIMO version is compared with other two AIS-based STLF models as well as with models based on ARIMA, exponential smoothing (ES) and neuron network (NN).

The first AIS-based model (AIS1) was proposed in [16] and analyzed in [29]. In this model patterns **x** and **y** are concatenated and represented by AGs with epitopes $\mathbf{u} = [\mathbf{x}^T \mathbf{y}^T]^T$. AB paratope is constructed analogously to the AG epitope: $\mathbf{v} = [\mathbf{p}^T \mathbf{q}^T]^T$. AGs are recognized by ABs which play a role of clusters: hyperballs of radius r with centers in points **p**. The paratopes **v** are modified during training (immune memory creation in clonal selection loop) to cover the AGs in the best way and minimize the forecast error. In the forecasting phase an incomplete AG is presented having only the x-part of an epitope (query pattern). It is recognized by a set of ABs from the immune memory which cover it in X subspace. We infer about the missing y-part of the epitope on the basis of q-parts of paratopes of the activated ABs.

The second AIS-based model (AIS2) was proposed in [17] and analyzed in [29]. It is composed of two population of ABs. Population of ABs of type x (ABx) recognize AGs representing x-patterns (AGx), whilst the population of ABs of type y (ABy) recognize AGs representing y-patterns (AGy). Patterns **x** are epitopes of AGxs and paratopes of ABxs, and patterns **y** are epitopes of AGys and paratopes of ABys. Epitopes and paratopes are fixed. ABx has the cross-reactivity threshold r defining the recognition region with center in the point **x**. Similarly, ABy has a recognition region of radius s with center in the point **y**. Radii r and s are adjusted individually during training, so that AB covers AGs which epitopes are similar to the AB paratope. AB represents a cluster of similar AGs in the pattern space X or Y. Sizes of the recognition regions of ABs depend on the data distribution in the spaces X and Y. After the two populations of the immune memory have been created, the empirical conditional probabilities $P(ABy_k \mid ABx_j)$, $j, k = 1, 2, \ldots, N$, that the i-th AGy stimulates the k-th ABy, when the corresponding i-th AGx stimulates the j-th ABx, are determined. These probabilities are used to determine the forecast pattern **y** paired with the query pattern $\mathbf{x}^*$.

In ARIMA and ES the load time series were decomposed into n series, i.e. for each t a separate series was created. This eliminates the daily seasonality. The ARIMA and ES

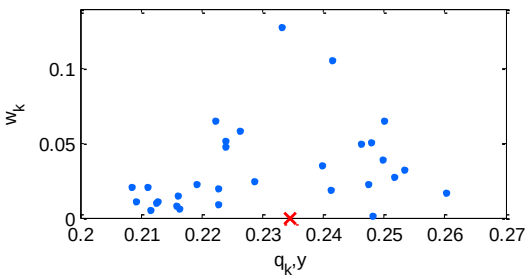

Fig. 11. The forecast (cross) determined from the centroid of points (q_k, w_k) (dots) representing activated ABs.

TABLE III
FORECAST ERRORS AND THEIR INTERQUARTILE RANGES FOR MIMO MODEL

Hour	January		July		Mean	
	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}
1	1.24	1.07	1.18	1.04	1.21	1.06
6	1.35	1.25	0.96	0.94	1.15	1.06
12	1.35	1.19	0.90	0.73	1.12	0.89
18	1.44	1.39	0.70	0.88	1.06	1.04
24	1.57	1.31	1.13	0.91	1.35	1.25
Mean	1.39	1.24	0.97	0.90	1.18	1.06

parameters were estimated for each forecasting task (forecast of system load at time t of day i) using 12-week time series fragments immediately preceding the forecasted day. Atypical days in these fragments were replaced with the days from the previous weeks. Due to using short time series fragments for parameter estimation (much shorter than the annual period) and due to time series decomposition into n series we do not have to take into account the annual and daily seasonality in the models. In such case the number of the parameters is much smaller and they are easier to estimate compared to models with triple seasonality. For each forecasting task the seasonal ARIMA$(p, d, q)\times(P, D, Q)_v$ model was created (where v = 7, i.e. one week period) as well as the ES state space model [30]. To estimate parameters of ARIMA and ES stepwise procedures for traversing the model spaces implemented in the **forecast** package for the R environment for statistical computing [31] were used. These automatic procedures return the optimal models with the lowest Akaike information criterion value.

The NN model is learned locally [32] using training patterns selected from the neighborhood of the query pattern. Patterns are defined in the same way as in this work. For each forecasting task a separate NN is learned using Levenberg-Marquardt algorithm with Bayesian regularization to prevent overfitting. Local fitting implies small number of neurons. Based on the research reported in [32] the network composed of only one neuron with bipolar sigmoid activation function was chosen as an optimal architecture.

In Fig. 12 sample fragments of the time series and their forecasts using different methods are shown. Errors for one day ahead STLF in Table IV are presented. Errors generated by the naïve model are also shown in this table. The naïve forecasts are created as follows: the forecasted daily curve is the same as seven days ago. The best results are marked with an asterisk and the second best ones are marked with a double

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

asterisk (best results were confirmed by Wilcoxon rank sum test with 5% significance level). As we can see from this table neural model outperforms the other ones. The AISLFS takes the second place for PL, FR and GB data. The conventional forecasting models: ARIMA and ES work significantly worse than other models for all time series.

Distributions of percentage errors PE in Table V are characterized by median (Q2), first quartile (Q1) and third quartile (Q3). It can be inferred from this statistics the degree of dispersion and skewness in errors (when Q1 and Q3 are not symmetrical around Q2). Zero median of PE indicates unbiased forecasts. Deviations of median from zero inform that the forecasts may have a general tendency to be too high or too low. Immune systems seem to generate the least biased forecast on average. The highest bias is observed for ARIMA and naïve model. Naïve model generates asymmetrically distributed errors. For other models PE distribution is more or less symmetrical ((Q3–Q2) ≈ (Q2–Q1)).

In Fig. 13 errors for forecast horizons up to 7 days are shown. The rankings of the models for each horizon are presented in Fig. 14. The first ranking is based on the average difference between model error (APE) and the smallest error for the test sample. In this ranking AISLFS occupies the second position for each horizon except $\tau = 5$. The best model for short horizons is NN and for longer horizons is ES. The second ranking is based on the average rank in accuracy

TABLE IV
FORECAST ERRORS AND THEIR INTERQUARTILE RANGES FOR THE EXAMINED MODELS

Model	PL		FR		GB		VC	
	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}
AISLFS	**1.51****	1.49	**1.79****	1.81	**1.67****	1.73	3.13	2.75
AIS1	**1.50****	1.50	1.93	1.95	1.77	1.84	**3.04****	2.75
AIS2	**1.50****	1.51	1.93	1.96	1.78	1.87	3.33	2.93
ARIMA	1.82	1.71	2.32	2.53	2.02	2.07	3.67	3.42
ES	1.66	1.57	2.10	2.29	1.85	1.84	3.52	3.35
NN	**1.44***	1.41	**1.64***	1.70	**1.65***	1.70	**2.92***	2.69
Naïve	3.43	3.42	5.05	5.96	3.52	3.82	4.88	4.55

TABLE V
MEDIAN (Q2), FIRST(Q1) AND THIRD (Q3) QUARTILES OF PE FOR THE EXAMINED MODELS

Model	PL	FR	GB	VC
	Q1 / Q2 / Q3	Q1 / Q2 / Q3	Q1 / Q2 / Q3	Q1 / Q2 / Q3
AISLFS	-1.05/0.02/1.12	-1.31/-0.06/1.23	-1.10/0.12/1.34	-1.86/-0.07/1.74
AIS1	-1.03/0.06/1.17	-1.37/-0.02/1.34	-1.32/0.07/1.40	-1.84/-0.02/1.93
AIS2	-1.05/0.01/1.12	-1.34/-0.02/1.37	-1.26/0.06/1.36	-2.00/-0.06/1.84
ARIMA	-1.05/0.18/1.39	-1.59/0.12/1.88	-1.40/0.15/1.69	-2.59/-0.28/2.02
ES	-1.05/0.10/1.20	-1.48/0.08/1.63	-1.29/0.14/1.51	-2.40/-0.06/2.07
NN	-0.91/0.11/1.17	-1.14/0.00/1.20	-0.99/0.17/1.36	-1.81/-0.13/1.63
Naïve	-1.76/0.42/2.39	-3.86/0.02/3.40	-2.41/0.24/2.67	-2.68/0.14/3.29

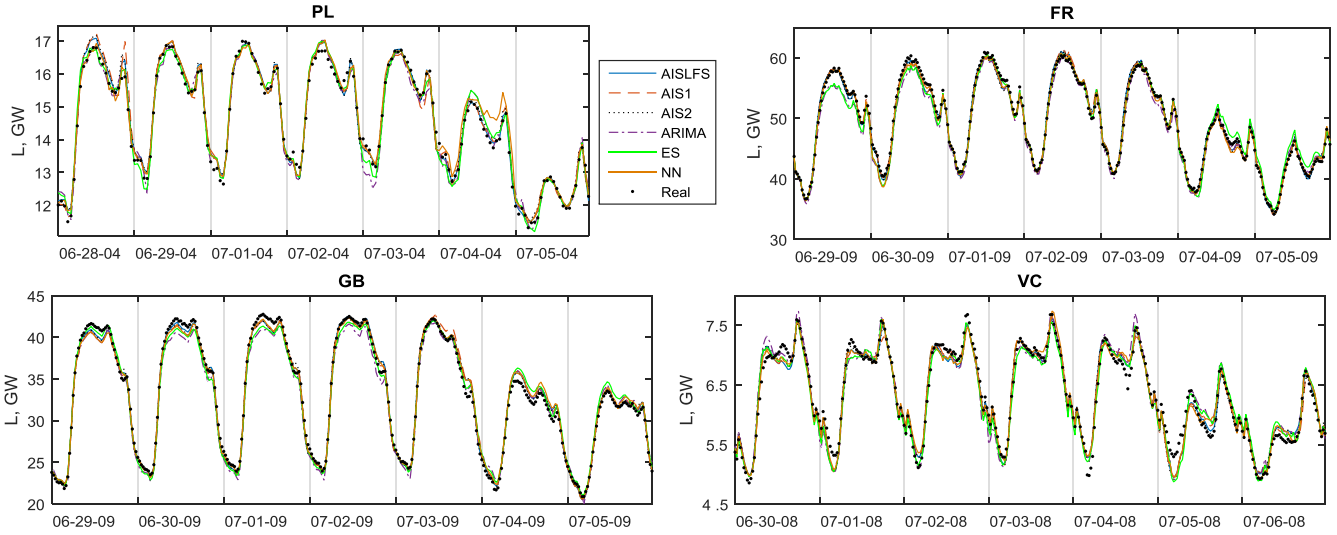

Fig. 12. Sample weekly fragments of the time series and their forecasts for one day ahead horizon.

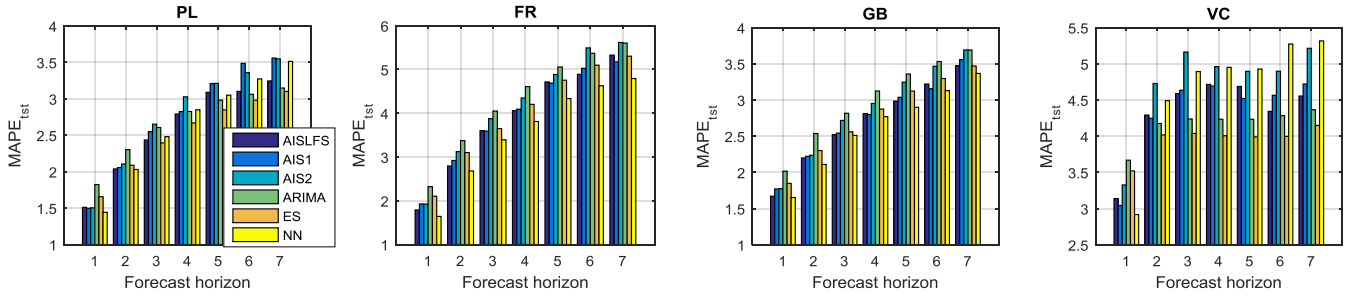

Fig. 13. Errors for different forecast horizons.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2016.2586049, IEEE Transactions on Evolutionary Computation

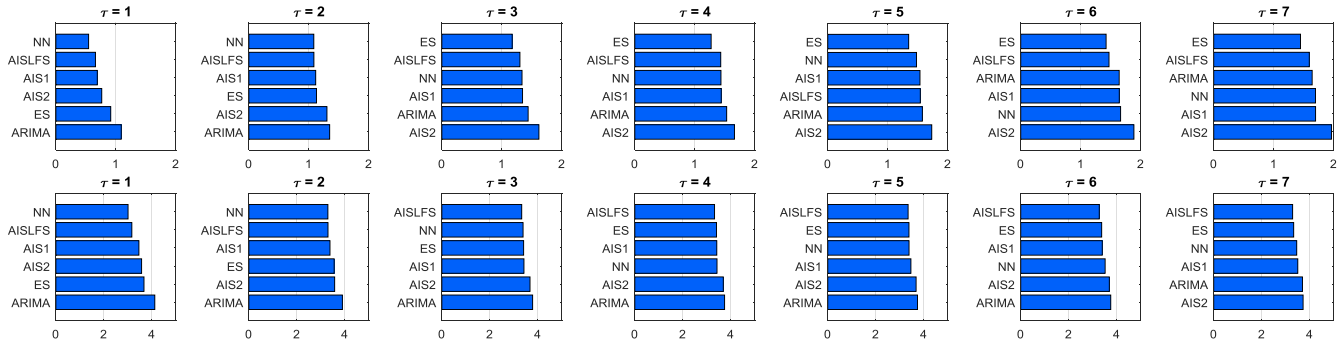

Fig. 14. Rankings of the forecasting models: based on the average difference between model error (APE) and the smallest error for the test sample (top row) and based on the average rank in accuracy ranking for each test sample (bottom row).

TABLE VI
TRAINING TIMES OF THE EXAMINED MODELS (IN SECONDS)

AISLFS	AIS1	AIS2	ARIMA	ES	NN
2.94	2.63	0.76	0.21	0.20	15.19

ranking for each test sample. In this case AISLFS outperforms all other models for horizons longer than two and occupy second position for the shortest horizons.

Training times of the algorithms in Table VI are shown. These are total times of model learning for forecasting 24 hourly loads for the next day at fixed values of hyperparameters. The simulations were made in Matlab R2015a and R 3.2.3 (ARIMA and ES) on the desktop computer with Intel Core2 Quad CPU Q9550@2.83 GHz, 4 GB RAM, Windows 7 64 bit. Classical algorithms, ARIMA and ES, are the fastest but in their cases any learning is not needed, because the final model is computed using maximum likelihood estimation. AIS2 is the fastest among immune-based models. In this model only one parameter is learned: the cross-reactivity threshold. AIS1 and AISLFS having more complex structures are learned longer. The longest training time is for NN. This is because NN is in MISO version and for forecasting 24 hourly loads we need 24 NN models learned individually.

VI. CONCLUSION

The immune system has many attractive features which can be implemented in machine learning algorithms for classification, clustering and regression. In the proposed forecasting model, which is a regression model, the clonal selection mechanism is applied to create specialized ABs (immune memory cells with suitably shaped paratopes) for recognition AGs representing fragments of load time series. In response to the AG representing the time series fragment preceding the forecasted fragment the activated ABs construct the forecasted fragment. Recognition abilities of ABs depends on their paratopes. The paratope corresponding to the selected features of the input vector (pattern representing time series fragment expressed by AG) is learned in the clonal selection loop individually for each ABs. As a result, each AB recognizes AGs by individually selected features. This local feature selection is a unique mechanism used in the proposed AIS-based forecasting model.

Time series representation by patterns expressed in ABs and AGs simplifies the problem of forecasting multiple seasonal nonstationary time series. This is due to filtering out the trend, annual and weekly cycles and unifying the daily cycles. This leads to the simplification of the relationship between input and output variables and consequently to the improvement in accuracy. The simulation studies have shown high accuracy of AISLFS, which is a strong competitor for other popular STLF models such as ARIMA, ES and NNs.

The novelties of the work can be summarized as follows:

- AISLFS uses an alternative representational abstraction: each AB defines recognition region in a different subspace (local feature selection),
- final decision is made collectively by many ABs, which are competent only in their own recognition regions (ensemble of weak learners),
- typically AIS are used for classification, clustering and optimization problems. This implementation is for regression problem,
- ABs and AGs represent fragments of time series. AISLFS in this implementation is used for prediction of time series.

This work is a continuation of research on AISLFS. First application of AISLFS was data classification [25]. It is planned in the future to apply AISLFS to unsupervised learning, where data clusters are formed in subspaces of the feature space based on locally selected features. In this case one point can belong to many different clusters represented by ABs.

REFERENCES

[1] R. Weron, *Modeling and Forecasting Electricity Loads and Prices*. Wiley, 2006.

[2] J.W. Taylor, "Short-term load forecasting with exponentially weighted methods," *IEEE Trans. Power Systems*, vol. 27, no. 1, pp. 458-464, 2012.

[3] S. Tzafestas, E. Tzafestas, "Computational intelligence techniques for short-term electric load forecasting," *Journal of Intelligent and Robotic Systems*, vol. 31, pp. 7–68, 2001.

[4] K. Metaxiotis, A. Kagiannas, D. Askounis, J. Psarras, "Artificial intelligence in short term electric load forecasting: A state-of-the-art survey for the researcher," *Energy Conversion and Management*, vol. 44, pp. 1525–1534, 2003.

[5] C. Cecati, J. Kolbusz, P. Rozycki, P. Siano and B. M. Wilamowski, "A novel RBF training algorithm for short-term electric load forecasting and comparative studies," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6519-6529, 2015.

[6] Ni Ding, C. Benoit, G. Foggia, Y. Besanger and F. Wurtz, "Neural network-based model design for short-term load forecast in distribution systems," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 72-81, Jan. 2016.

[7] D.K. Chaturvedi, A.P. Sinha, O.P. Malik, "Short term load forecast using fuzzy logic and wavelet transform integrated generalized neural network," *International Journal of Electrical Power & Energy Systems*, vol. 67, pp. 230-237, 2015.

[8] A. Khosravi, S. Nahavandi, D. Creighton and D. Srinivasan, "Interval type-2 fuzzy logic systems for load forecasting: A comparative study," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1274-1282, Aug. 2012.

[9] M. Read, P. Andrews, J. Timmis, "An introduction to artificial immune systems," in *The Handbook of Natural Computing*, Springer, 2011.

[10] J. Farmer, N. Packard, and A. Perelson, "The Immune System, Adaptation and Machine Learning," *Physica D*, vol. 22, pp. 187–204, 1986.

[11] H. Bersini and F. Varela, "The immune learning mechanisms: Recruitment reinforcement and their applications," in *Computing with Biological Metaphors*, Chapman and Hall, 1993.

[12] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proc. IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1994, pp. 202-212.

[13] J.O. Kephart, "A biologically inspired immune system for computers," in *Proc. Artificial Life IV: The Fourth International Workshop on the Synthesis and Simulation of Living Systems*, MIT Press., 1994, pp. 130-139.

[14] D. Dasgupta, S. Yu, F. Nino, "Recent advances in artificial immune systems: models and applications," *Applied Soft Computing*, vol. 11, issue 2, pp. 1574-1587, 2011.

[15] E. Hart, J. Timmis, "Application areas of AIS: the past, the present and the future," *Applied Soft Computing*, vol. 8, issue 1, pp. 191-201, 2008.

[16] G. Dudek, "Artificial immune system for short-term electric load forecasting," in *Proc. 9th ICAISC*, LNAI 5097, 2008, pp. 1007–1017.

[17] G. Dudek, "Artificial immune clustering algorithm to forecasting seasonal time series," in *Proc. 3rd ICCCI*, LNAI 6922, 2011, pp. 468-477.

[18] Huang Yue, Li Dan and Gao Liqun, "Power system short-term load forecasting based on neural network with artificial immune algorithm," in *Proc. Control and Decision (CCDC'12)*, 2012, pp. 844-848.

[19] C. McEwan and E. Hart, "Representation in the (artificial) immune system," *J. Math. Model. Algorithms*, vol. 8(2), pp. 125-149, 2009.

[20] T. Stibor, J. Timmis, and C. Eckert, "On the use of hyperspheres in artificial immune systems as antibody recognition regions," in *Proc. ICARIS 2006*, LNCS 4163, 2006, pp. 215-228.

[21] I.R. Cohen, U. Hershberg, and S. Solomon, "Antigen receptor degeneracy and immunological paradigms," *Molecular Immunology*, vol. 40, pp. 993-996, 2004.

[22] K.W. Wucherpfennig, et al., "Polyspecificity of T cell and B cell receptor recognition," *Semin. Immunol*, vol. 19, pp. 216–224, 2007.

[23] M. Mendao, J. Timmis, P.S. Andrews, and M. Davies, "The immune system in pieces: Computational lessons from degeneracy in the immune system," in *Proc. Foundations of Computational Intelligence* (FOCI 2007), 2007, pp. 394-400.

[24] U. Hershberg, S. Solomon, I.R. Cohen, "What is the basis of the immune system's specificity?" in V. Capasso, (ed.) *Mathematical Modelling & Computing in Biology and Medicine*, pp. 377–384, 2003.

[25] G. Dudek, "Artificial immune system for classification with local feature selection," *IEEE Trans. on Evolutionary Computation*, vol. 16, issue 6, pp. 847-860, 2012.

[26] G. Dudek, "Pattern similarity-based methods for short-term load forecasting – Part 1: Principles," *Applied Soft Computing*, vol. 37, pp. 277-287, 2015.

[27] G. Dudek, "Tournament searching method to feature selection problem," in *Proc. 10th ICAISC*, LNAI 6114. 2010, pp. 437–444.

[28] T.K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, 8, pp. 832-844, 1998.

[29] G. Dudek, "Pattern similarity-based methods for short-term load forecasting – Part 2: Models," *Applied Soft Computing*, vol. 36, pp. 422-441, 2015.

[30] R.J. Hyndman, A.B. Koehler, J.K. Ord, R.D. Snyder, *Forecasting with Exponential Smoothing: The State Space Approach*, Springer, 2008.

[31] R.J. Hyndman, Y. Khandakar, "Automatic time series forecasting: The forecast package for R," *Journal of Statistical Software*, vol. 27, no. 3, pp. 1–22, 2008.

[32] G. Dudek, "Forecasting time series with multiple seasonal cycles using neural networks with local learning," in *Proc. 12th ICAISC*, LNAI 7894, 2013, pp. 52-63.

Grzegorz Dudek received his PhD degree in electrical engineering from the Czestochowa University of Technology, Poland, in 2003 and habilitation degree in computer science from Lodz University of Technology, Poland, in 2013. Currently, he is an associate professor at the Department of Electrical Engineering, Czestochowa University of Technology. He is the author of two books concerning machine learning methods for load forecasting and evolutionary algorithms for unit commitment and over 70 scientific papers. He was awarded with 3-rd place in price probabilistic forecasting track of Global Energy Forecasting Competition (GEFCOM 2014) sponsored by IEEE Power & Energy Society. His research interests include pattern recognition, machine learning, artificial intelligence, and their application to classification, regression, forecasting and optimization problems.