IMPROVING NEURAL NETWORK RIVER FORECASTING WITH SWARM-BASED OPTIMIZATION ALGORITHMS

RICCARDO TAORMINA

Ph.D

The Hong Kong Polytechnic University

2016

The Hong Kong Polytechnic University

Department of Civil and Environmental Engineering

Improving Neural Network River Forecasting with Swarm-based Optimization Algorithms

Riccardo Taormina

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

July 2015

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it produces no material previously published or writer, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Riccardo Taormina

Abstract

Neural Network River Forecasting (NNRF) entails the use of Artificial Neural Networks (ANNs) for the prediction of streamflow quantities. Despite the amount of research on the subject, NNRF still struggle to move from the academic context to the operational context due to a number of unresolved issues. Major problems of NNRF methodologies include a) difficulties in quantifying the uncertainty of model predictions, b) the lack of standardized methodologies for identifying optimal predictors and suitable functional forms of the underlying data-driven model, and c) concerns with the black-box nature of NNRF models which drive practitioners to favour physically-based alternatives.

The main contribution of this thesis is to show that these issues, albeit very different in nature, can all be addressed by developing NNRF models using Global Optimization. In particular, this work introduces three new Particle Swarm Optimization (PSO) variants which are employed to devise novel ad-hoc applications aimed at solving each particular issue. These algorithms are the Multi-Objective Fully Informed Particle Swarm (MOFIPS) optimization, the Binary-coded Fully Informed Particle Swarm (BFIPS), and its multi-objective generalization (MBFIPS). Testing these new techniques will also provide insights on the real effectiveness of PSO for data-driven hydrological modelling, a task which has been only partially accomplished by the research community. In addition, this thesis advocates the use of Extreme Learning Machines (ELMs) as alternative NNRF models. Although research in other fields has shown that ELMs provides better accuracy at much faster speed compared to ANNs, at the time of writing, they have never been employed for NNRF modeling.

There are four applications at the core of this thesis. In a first application it is demonstrated that better deterministic PSO-trained NNRF models can be obtained by formulating cross validation as a bi-objective optimization problem using MOFIPS to perform ANN calibration. The benefits of bi-objective optimization are also shown for the construction of NNRF prediction intervals. This is done in a second application where MOFIPS and the Lower Upper Bound Estimation method are employed for fast and straightforward development of interval-based models.

In a third study, a novel approach for model and Input Variable Selection (IVS) that employs BFIPS and MBIFPS along with the ELMs is presented. A comparison with 4 existing techniques, done using the tools of a comprehensive framework, suggests that the developed ELM-based models are more accurate in performing the IVS task for data-driven hydrological modelling.

Lastly, BFIPS, MBFIPS and ELMs are employed to investigate whether more accurate prediction of streamflow discharges can be achieved by including expert knowledge in NNRF model development. In particular, total streamflow predictive accuracy of modular models (MM) trained to perform an implicit baseflow separation is compared against that of global models (GM). The results for 9 different watersheds in northern United States show that MMs underperform GMs in predicting the total flow. In addition, the study demonstrates that greater accuracy in baseflow separation usually corresponds to worse total flow predictions, suggesting that these two objectives are conflicting, rather than compatible.

Publications

Articles in Journals

Taormina, R., Chau, K. W., & Sivakumar B. (2015). Neural network river forecasting through baseflow separation and binary-coded swarm optimization. *Journal of Hydrology*, 529, 1788-1797.doi:10.1016/j.jhydrol.2015.08.008

Taormina, R., & Chau, K. W. (2015). Data-driven input variable selection for rainfall-runoff modeling using binary-coded particle swarm optimization and Extreme Learning Machines. *Journal of Hydrology*, 529, 1617-1632. doi:10.1016/j.jhydrol.2015.08.022

Taormina, R., & Chau, K. W. (2015). ANN-based interval forecasting of streamflow discharges using the LUBE method and MOFIPS. *Engineering Applications of Artificial Intelligence*, 45, 429-440. doi:10.1016/j.engappai.2015.07.019

Taormina, R., & Chau, K. W. (2015). Neural network river forecasting with multi-objective fully informed particle swarm optimization. *Journal of Hydroinformatics*, 17, 99–112. doi:10.2166/hydro.2014.116

Taormina, R., Chau, K. W., & Sethi, R. (2012). Artificial neural network simulation of hourly groundwater levels in a coastal aquifer system of the Venice lagoon. *Engineering Applications of Artificial Intelligence*, 25, 1670–1676. doi:10.1016/j.engappai.2012.02.009

Conference Posters

Taormina, R., Chau, K. W., & Galelli. S. (2015). Comparison of ANN and ELM for Data-Driven Streamflow Prediction Applications. *Asia Oceania Geosciences Society (AOGS) 12th Annual Meeting - AOGS 2015*. August 2015. Singapore

Acknowledgments

First of all, I would like to thank my supervisor, Prof. K.W. Chau, and the Research Grants Council of Hong Kong for the opportunity they gave me to pursue my Ph.D. research at the Department of Civil and Environmental Engineering of the Hong Kong Polytechnic University, under the prestigious Hong Kong Ph.D. Fellowship Scheme.

Secondly, I would like to thank my family and friends scattered around the world for the support they gave me through all these years, regardless of the distance that separated us. A special thank goes to the STF group in Hong Kong, for probably the best Tuesdays (and Mondays sometimes) I had in my life.

Then I would like to thank all the Faculty members and researchers with whom I had the honor to work during my Ph.D. studies. In particular, I wish to thank Prof. Rajandrea Sethi at Politecnico di Torino, with whom I started my research career; Prof. Bellie Sivakumar, that gave me the chance to work under his supervision at the University of New South Wales; and Prof. Stefano Galelli, with whom I have the pleasure to continue my research duties at the Singapore University of Technology and Design.

Lastly, I am deeply in debt with the cities of Hong Kong, Sydney and Singapore for being my home during all the time of my life as a Ph.D. Student, and a continuous source of inspiration for becoming a better researcher, and a better man.

Thanks!

Table of Contents

Abbreviations

AIC	Akaike Information Criterion
ANN	Artificial Neural Networks
BF	BaseFlow
BFI	BaseFlow Index
BFIPS	Binary-coded Fully Informed Particle Swarm
BIC	Bayesian Information Criterion
BNN	Bayesian Neural Network
BS	Base-flow Separation
BS-MM	Baseflow Separation-based Modular Model
CE	Coefficient of Efficiency
CFS	Constructive Forward Selection
CGF	Conjugate Gradient with Fletcher-Reeves updates
CGP	Conjugate Gradient with Polak-Ribiére updates
CI	Confidence Interval
CL	Confidence Level
CWC	Coverage Width-based Criterion
EC	Evolutionary Computation
EF	Excess Flow
ELM	Extreme Learning Machines
FIPS	Fully Informed Particle Swarm
GA	Genetic Algorithm

GM	Global Model
GO	Global Optimization
GPS	Generalized Pattern Search
IIS	Iterative Input Selection
IVS	Input Variable Selection
LM	Levenberg-Marquardt
LUBE	Lower Upper Bound Estimation
LUEM	Local Uncertainty Estimation Model
MBFIPS	Multi-objective Binary-coded Fully Informed Particle Swarm
MdAPE	Median Absolute Percentage Error
MI	Mutual Information
MM	Modular Model
MOEA	Multi-Objective Evolutionary Algorithm
MOFIPS	Multi-Objective Fully Informed Particle Swarm
MP	Most Precautionary (referred to MOFIPS-based LUBE solutions)
MSLE	Mean Squared Logarithmic Error
NI	Narrowest Interval (referred to MOFIPS-based LUBE solutions)
NNRF	Neural Network River Forecasting
NSGA-II	Non-dominated Sorting Genetic Algorithm II
PAES	Pareto-Archived Evolution Strategy
PCIS	Partial Correlation Input Selection
PI	Prediction Interval

PICP	Prediction Interval Coverage Probability
PINAW	Prediction Interval Normalized Average Width
PINRW	Prediction Interval Normalized Root-mean-square Width
PMI	Partial Mutual Information
PSO	Particle Swarm Optimization
R4MS4	Fourth Root Mean Quadrupled Error
E	
RBF	Radial Basis Functions
RMSE	Root Mean Squared Error
SA	Selection Accuracy
SCG	Scaled Conjugate Gradient
SI	Swarm Intelligence
SLFN	Single-hidden Layer Feed-forward neural Networks
SOM	Self-Organizing Map
SPEA	Strength Pareto-Evolutionary Algorithm
STDEV	STandard DEViation
SVM	Support Vector Machines
TF	Total Flow

Symbols

Chapter 2

$\boldsymbol{x}$	ANN inputs
y	Observed output
$\hat{y}(\boldsymbol{x})$	ANN output
p	Number of input variables
L	Number of hidden neurons
$\boldsymbol{a}_i$	Hidden layer connection weights for *i*-th hidden neuron
β_i	Output layer connection weights for *i*-th hidden neuron
b_i	ANN bias
$G(\cdot)$	ANN activation function
e	Natural exponential function
s	Steepness of sigmoidal function
t	Threshold of sigmoidal function
$\boldsymbol{\theta}$	Set of ANN parameters
$\widehat{\boldsymbol{\theta}}_N$	Estimated set of ANN parameters
$r_j(\boldsymbol{\theta})$	ANN residuals
$F(\boldsymbol{\theta})$	Sum of squared residuals
N	Total number of observations
$\boldsymbol{J}$	Jacobian matrix
$\boldsymbol{H}$	(Section 2.1) Hessian matrix
ϑ	Scaling coefficient for LM training

k	Number of folds
λ	(Section 2.1) Regularization parameter
$f_L(\boldsymbol{x})$	ELM output
q	Number of ELM output nodes
$\mathbb{R}$	Set of real numbers
$\boldsymbol{x}_j$	j-th input pattern
$\boldsymbol{t}_j$	j-th output pattern
$\boldsymbol{T}$	Array of output patterns
$\boldsymbol{H}$	(Section 2.2) Hidden layer output matrix
$\boldsymbol{H}^+$	Moore-Penrose generalize inverse of $\boldsymbol{H}$
$\widehat{\boldsymbol{\beta}}$	Estimated hidden layer parameters
λ	(Section 2.2) Ridge regression constant
m	ELM model complexity

Chapter 3

d	Problem dimension (particle size)
$\boldsymbol{X}_i$	Position of i-th particle
$\boldsymbol{V}_i$	Velocity of i-th particle
$\boldsymbol{P}_i$	Personal best of i-th particle
$\boldsymbol{G}$	Overall best position in the neighborhood
$\otimes$	Point-wise multiplication
$\boldsymbol{U}(0,\cdot)$	Vector of uniformly distributed random numbers between 0 and a positive real number

χ	Constriction coefficient
$\varphi, \varphi_1, \varphi_2$	Acceleration constants
X_{MIN}, X_{MAX}	Minimum and maximum value for particle positions
V_{MIN}, V_{MAX}	Minimum and maximum value for particle velocity
$\boldsymbol{P}_{nbr_j}$	Particle's *j*-th neighbor
K_i	Number of neighbors of particle *i*
a_j	Weighting coefficient
$S(\cdot)$	Logistic function
m	Number of particles subject to bit-flipping mutation
μ	Mutation rate
$\boldsymbol{L}$	MOFIPS leading particle
M	Maximum size of MOFIPS Pareto-front
u	Uniform random number in [0,1]
η_m	Polynomial probability distribution shape parameter
δ	Polynomial mutation parameter
ρ_m	Percentage of particles subjected to polynomial mutation
Υ	Convergence metric
Δ	Diversity metric

Chapter 5

l, u	Lower and upper bound of prediction interval
n	Number of observations

R	Range of the observed variable
$N(0,\sigma)$	Random number from Gaussian distribution with mean 0 and standard deviation σ

Chapter 6

l_{inputs}	Number of bits needed to encode the selected subset of inputs
l_{nodes}	Number of bits needed to encode the number of hidden neurons
l_{act}	Number of bits needed to encode the type activation function
l_{λ}	Number of bits needed to encode the ridge regression constant
NH_{min}, NH_{max}	Minimum and maximum number of hidden neurons
g	Exponent for AIC and BIC complexity penalty
ρ	Cutoff percentage for selecting solution on the MBFIPS Pareto-front
$RMSE^*$	RMSE value of the selected solution on the MBFIPS Pareto-front
$RMSE_{MIN}$	Minimum value of RMSE on the MBFIPS Pareto-front
K	Number of relevant inputs
P	Total number of candidate inputs
k	Number of relevant inputs selected
p	Number of extraneous inputs retained
SA, SA_c, SA_e	Selection accuracy scores
γ	Weighting factor for SA computation
Q	Flow data
ER	Effective rainfall

Chapter 7

$Q(t)$, $\tilde{Q}(t)$	Observed and estimated total flow at time *t*
$Q_{BF}(t)$, $\tilde{Q}_{BF}(t)$	Observed and estimated baseflow at time *t*
$Q_{EF}(t)$, $\tilde{Q}_{EF}(t)$	Observed and estimated excess flow at time *t*
Q_{BF0}	Initial baseflow value
a	Recession constant
BFI_{max}	Maximum value of BFI
NH_1, NH_2	Number of hidden neurons of the BS-MM
$RMSE_{TF}$	RMSE for total flow predictions
$RMSE_{BF}$	RMSE for baseflow predictions
$RMSE_{EF}$	RMSE for excess flow predictions
E_T	Total error function to minimize for BS-MM training
FLOW	Flow input variables
RAIN	Rainfall input variables
SNOW	Snowfall input variables
SNWD	Snow depth input variables
TMIN, TMAX	Minimum and maximum temperature input variables

List of Figures

List of Tables

PART I. INTRODUCTION

1. Introduction

1.1. Neural Network River Forecasting (NNRF)

In the past twenty years, Artificial Neural Networks (ANNs) have been successfully employed as data-driven modelling tools in many hydrological and water resources contexts. The main reasons behind the popularity of these heuristics lie in their ability to cope with the non-linear, non-stationary and non-Gaussian behaviors typical of hydrological processes (Govindaraju, 2000a, 2000b; Maier and Dandy, 2000; Maier et al., 2010). Common examples of ANN applications range from the estimation of precipitation (Tomassetti et al., 2009; Toth et al., 2000; Wu and Chau, 2013; Wu et al., 2010), and groundwater modelling (Adamowski and Chan, 2011; Coulibaly et al., 2001; Trichakis et al., 2009), to water quality modelling (Chang et al., 2010; Muttil and Chau, 2007, 2006; Wu et al., 2014) and reservoir operations (Chaves and Chang, 2008; Labadie, 2004; Raman and Chandramouli, 1996). However, most applications deal with the rainfall-runoff process and the prediction of streamflow quantities (Dawson and Wilby, 1998, 2001; Kişi, 2004; Minns and Hall, 1996; Shamseldin, 1997; Thirumalaiah and Deo, 1998; Tokar and Johnson, 1999; Tokar and Markus, 2000; Toth, 2009; Wu and Chau, 2011; Wu et al., 2009). Such applications have been collectively termed Neural Network River Forecasting (NNRF) in a recent paper by several prominent authors of the field (Abrahart et al., 2012), and this work positions itself within this area of research. While most NNRF applications regard ANNs and its variants, the term also applies to a wider range of models including hybrid-ANN techniques such as neuro-fuzzy (Chau et al., 2005; Pramanik and Panda, 2009; Sanikhani and Kisi, 2012) and

neuro-wavelet solutions (Adamowski and Sun, 2010; Danandeh Mehr et al., 2013; Kişi, 2009), as well as Radial Basis Functions (RBF) (Fernando and Shamseldin, 2009; Jayawardena and Fernando, 1998; Senthil Kumar et al., 2005) and Support Vector Machines (SVM) (Lin et al., 2006; Nourani et al., 2014, 2009).

Although NNRF models are known to perform favourably against conceptual models (Carcano et al., 2008; Nayak et al., 2013; Tokar and Markus, 2000), these solutions are seldom applied for operational purposes in real-world contexts. The limited appeal of NNRF to practitioners has to be attributed to some unresolved issues which have been identified in previous research but are still far away from being solved (Abrahart et al., 2012). For instance, the great majority of examples in the literature are concerned only with the development of NNRF models producing deterministic point predictions. This strongly contradicts with the fact that hydrological forecasts can be employed only if a measure of their reliability is attached to each predicted value (Krzysztofowicz, 2001). Indeed, there only a few studies that try to quantify the uncertainty of NNRF outputs, and the proposed solutions are usually complicated (Alvisi and Franchini, 2011; Khan and Coulibaly, 2006; Kingston et al., 2005; Sharma and Tiwari, 2009; Shrestha and Solomatine, 2006; Tiwari and Chatterjee, 2010; Zhang et al., 2009).

Furthermore, while researchers have focused on ad-hoc modifications and incremental refinement of existing techniques, which contributed little to the advancement of the field, NNRF still lacks standardized methodologies for identifying optimal predictors among available inputs, or for selecting the functional form of the underlying data-driven model automatically. The Input Variable Selection (IVS)

framework recently introduced by Galelli et al. (2014) certainly represents a major step forward in this direction. The framework provides a comprehensive set of evaluation criteria and datasets that allow for a thorough assessment of the effectiveness of IVS techniques for data-driven hydrological modelling. While there exist fast and accurate model-free IVS techniques (Fernando et al., 2009; Galelli and Castelletti, 2013; May et al., 2008), model-based alternatives are usually very slow as they entail time-consuming iterative processes requiring the training of a vast number of potential models (May et al., 2011). This is unfortunate since model-based IVS approaches account for the actual gain in model performances given by each selected variable, as well as for the contribution of individually irrelevant candidates with high combined explanatory power (Guyon and Elisseeff, 2003; Kohavi and John, 1997). In addition, these methods could be generalised so that they can also return the optimal model structure and the set of parameters maximizing NNRF model performances (Abrahart et al., 1999; Chen and Chang, 2009; Dawson et al., 2006).

Regardless of NNRF model accuracy in reproducing the hydrograph, there are still widespread criticisms on their black-box nature and cautions against their use in real-world problems in favor of physically plausible alternatives. To overcome this issue, recent efforts have been made to explain the internal workings of ANN, and link the processes taking place within the network to the processes in the watershed (Fernando and Shamseldin, 2009; Jain and Kumar, 2009; Jain et al., 2004; Wilby et al., 2003). Others have focused on the incorporation of expert knowledge into NNRF models in order to improve their hydrological plausibility and overall performances (Corzo and Solomatine, 2007a, 2007b; Jain and Srinivasulu, 2006; Parasuraman et al., 2006;

Srinivasulu and Jain, 2009; Toth, 2009; Zhang and Govindaraju, 2000). While these studies certainly provided insightful guidance and paved the way for future explorations, it is understood that much work is needed to fully address this issue (Abrahart et al., 2012)

1.2. NNRF model development with swarm optimization

At the core of this thesis is the idea that, despite their intrinsic differences, the problems highlighted in the previous paragraphs can be addressed by combining ANN with real- and binary-coded Global Optimization (GO) techniques. GO techniques, and nature-inspired heuristics such as Genetic Algorithms in particular, have been used in several NNRF and sister applications usually in order to 1) perform ANN training via single- (Chau et al., 2005; Jain and Srinivasulu, 2004; Sedki et al., 2009; Wu and Chau, 2006) and multi-objective optimization (de Vos and Rientjes, 2008, 2007) ; 2) optimize NNRF model structures (Abrahart et al., 1999; Chen and Chang, 2009; Corzo and Solomatine, 2007a, 2007b); 3) determine the optimal set of model parameters and ANN architecture (Abrahart et al., 2007; Dawson et al., 2006; Leahy et al., 2008); and 4) perform the IVS task (Bowden et al., 2005a, 2005b).

This thesis work will be focused on the use of Particle Swarm Optimization (PSO), a population-based GO technique devised to mimic natural phenomena such as bird flocking or fish schooling (Kennedy and Eberhart, 1995). Like other GO methods, PSO can be used to find the global optima of non-differentiable objective functions and can be easily adapted to work in discrete search spaces (Kennedy and Eberhart, 1997). Although the intuition behind the algorithm is simple, PSO compares favorably against

other GO methods both in terms of speed and accuracy (Poli et al., 2007). In addition, the implementation of the PSO is straightforward, and the algorithm lends itself extremely well to parallelization and multi-objective generalization. Despite such advantages, there are very few applications of PSO for NNRF model development. These initial studies were only limited to the use of PSO as an alternative to gradient-based techniques for model calibration, and they report contradictory results on PSO efficacy (Chau, 2007, 2006; Piotrowski and Napiorkowski, 2011). In addition, due to the lack of research in this area, many of the improvements made over the years to the original algorithm have been overlooked.

Part of this research has been thus dedicated to devise new variants of the PSO algorithm to be employed for NNRF model development. These new methods are all based on the Fully Informed Particle Swarm (FIPS) variant of swarm optimization (Mendes et al., 2004) ,which is known to outperform canonical PSO. The algorithms have been named the Multi-Objective Fully Informed Particle Swarm (MOFIPS) optimization algorithm, the discrete Binary-coded Fully Informed Particle Swarm (BFIPS) optimization algorithm, and its multi-objective generalization (MBFIPS).

1.3. Thesis objectives

The main goal of this thesis is to employ these original techniques in novel applications that can contribute to the field of data-driven hydrological modeling by directly addressing some of the major issues concerning NNRF. Although a secondary objective of this work is to provide insights on the real effectiveness of swarm optimization, it is important to note that the findings of the presented thesis work could

be extended to other GO algorithms. Some of the applications reported here employ Extreme Learning Machines (ELM) (Huang et al., 2011, 2006, 2004) as the underlying NNRF models. ELMs are three-layered ANNs constructed by randomly assigning the input weights and hidden biases. This operation reduces the ANN to a linear system, and allows for the analytical determination of the output weights using common least-squares. These simplifications drastically increase the speed of the learning process, which was also found to grant better generalization compared with traditional ANNs and other techniques such as SVMs. It was also shown that ELMs are universal approximators that can work with a broad type of activation functions, as long as they are bounded non-constant piecewise continuous. Recently, it was demonstrated that ELMs actually represent a simple unified learning framework for ANNs, polynomial networks, RBFs and SVMs which can be applied to both regression and multiclass classification problems (Huang et al., 2012). Despite these advantages, at the time of this writing ELMs have never been used as NNRF models, thus demonstrating their suitability in modeling streamflow quantities represents another objective of this thesis work.

Featured applications

In a first application for the Shenandoah River watershed, Virginia (US), it is argued that superior PSO-trained NNRF models can be obtained by treating cross-validation as a multi-objective optimization problem. The study shows that common single-objective cross-validation hinders the search performed by the PSO. Accordingly, employing the MOFIPS paradigm results in better performing models with respect to

those obtained with both single-objective PSO as well as some of the most advanced gradient-based optimization algorithms.

MOFIPS optimization can also benefit the generation of interval-based forecasts, as demonstrated in a second application concerning streamflow modeling in the Susquehanna and Nehalem rivers, US. In this innovative application, prediction intervals (PIs) of future streamflow are estimated using the Lower Upper Bound Estimation (LUBE) method (Khosravi et al., 2011). The LUBE method constructs an ANN with two output neurons that directly approximate the lower and upper bounds of the PIs. The training is carried out by minimizing a Coverage Width-based Criterion (CWC), which is a highly nonlinear and non-differentiable function accounting for both coverage probability and interval width. Even for this case, substantial improvements are obtained by using MOFIPS instead of single-objective PSO with cross-validation (Quan et al., 2014). Most importantly, the proposed MOFIPS-based LUBE ANN represents a fast and straightforward model for interval-based NNRF.

In another study, a novel approach for model structure and Input Variable Selection that employs BFIPS and MBIFPS along with the ELMs is presented. The algorithms are utilized to develop fast and accurate ELM-based IVS techniques by encoding the subset of selected inputs and ELM structural characteristics in a binary string. The performances of these methods are assessed using the criteria and the datasets provided by the IVS evaluation framework for environmental modeling (Galelli et al., 2014). From a comparison with 4 major IVS techniques, it emerges that, on average, the proposed methods substantially outperform the other methods in terms of selection accuracy. In particular, the MBFIPS-ELM wrapper was found to be the best

performer overall, reaching an almost perfect specification of the optimal input subset for a partially synthetic rainfall-runoff experiment devised for the Kentucky River basin (US).

Lastly, BFIPS, MBFIPS and ELMs are employed to investigate whether more accurate prediction of total streamflow could be achieved by including expert knowledge in the development of data-driven rainfall-runoff models. In particular, the effectiveness of modular models (MM) trained to perform an implicit baseflow separation is put to the test and compared against that of global models (GM) for 9 different gaging stations in northern United States. The ELM modules fit separately the base flow (BF) and excess flow (EF) components as obtained by a digital filter, and the MM reconstructs the total flow (TF) by adding these two signals at the output. BFIPS is employed for the identification of filter parameters and model structure by minimizing a weighted function of the errors of the TF, BF, and EF. On the other hand, the selection of the most relevant inputs is done using MBIPS-ELM. The results show that there is no evidence that MMs outperform GMs for predicting the TF. In addition, the baseflow produced by the MM largely underestimates the actual baseflow component expected for most of the considered gages. This occurs because the values of the filter parameters maximizing overall accuracy do not reflect the geological characteristics of the river basins. Indeed, setting the filter parameters according to expert knowledge results in accurate baseflow separation but lower accuracy of TF predictions, suggesting that these two objectives are intrinsically conflicting rather than compatible.

1.4. Structure of the thesis

This thesis is organized as follows. Part II provides the necessary background on the methods employed in this work. In particular, Chapter 2 presents the models used, i.e. ANNs and ELMs, while Chapter 3 briefly reviews GO, PSO and FIPS before introducing the BFIPS, MOFIPS and MBFIPS algorithms. These are presented in Section 3.6, 3.7.2, and 3.7.4 respectively. The four applications summarized in the previous section are described in details in Chapters 4 to 7 which constitute Part III of this manuscript. The last part of this work will contain a summary of the thesis, along with additional conclusions and a brief outline of possible future developments.

PART II. METHODS

2. Data-driven models

2.1. Artificial Neural Networks

2.1.1. Introduction

Artificial Neural Networks (ANNs) are biologically inspired mathematical models that are able to map an unknown input-output relationship relying on data samples gathered from the system under examination. Such models have been proven to approximate any differential function to a chosen degree of accuracy (Hornik et al., 1989), provided the number of parameters incorporated in the model is large enough. The simplest, yet more popular ANN, is the multi-layered Feed-forward Neural Network (FNN), which has been widely employed by hydrologists end engineers due to its ability to model nonlinear, non-stationary and non-Gaussian processes like those encountered in water resources contexts (Govindaraju, 2000; Haykin, 2008; Maier and Dandy, 2000; Maier et al., 2010). FNNs can be represented as a graph where a number of processing units, or *neurons*, are arranged in *layers* and linked by *synaptic connections* (Fig. 2.1). In FFNs, the information is allowed to flow from one layer to the next one only in a single direction, which goes from the input layer to the output layer. The nonlinear processing takes place in all the neurons between the input and the output layer, which are called *hidden neurons* and are grouped in one or more *hidden layers*. No processing is done in the input neurons, while the output neurons usually perform a linear rescaling to match the range of the output variable that has to be estimated. The hidden units in the network receive their inputs through the synaptic connections, whose

strength is identified by scalar *synaptic weights*. The inputs are multiplied by the weights associated to each synapse to form the exciting field entering the receiving neuron. In the neuron, a nonlinear activation function, usually of sigmoidal shape, transforms the received field in the output activation value for the neuron. This procedure is carried out in all the hidden layers in the network, until the output units compute the final response for the input pattern that has been presented to the network. The mapping of the desired input-output relationship through a FNN is then obtained by tuning the synaptic weights of the network, usually by means of deterministic iterative algorithms that minimize an error function of the residuals between the observed output of the system and the response produced by the model. After the training is finished, the neural network model can be validated to check for its generalization ability, i.e. its performances on data which has not seen during the training process, and then employed on new data for the task it has been devised to accomplish.

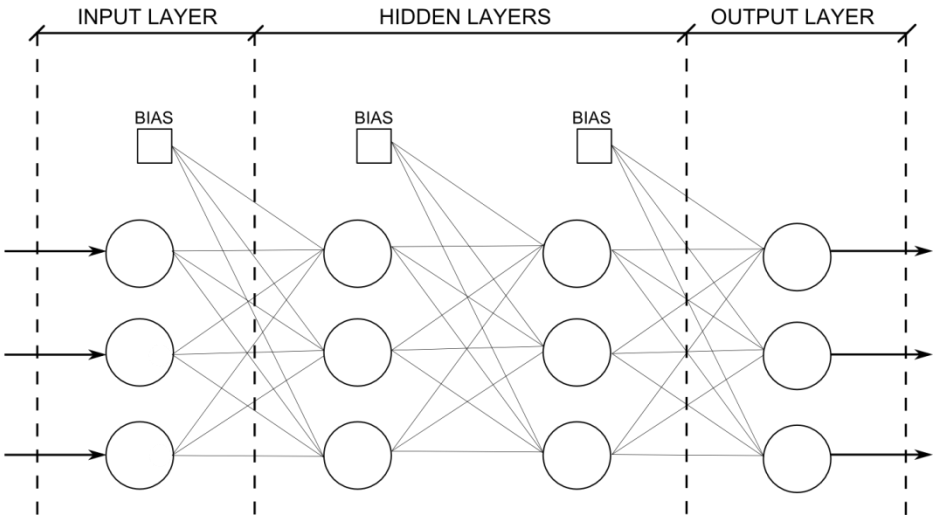


Figure 2.1. Feed-forward Neural Network

2.1.2. Single-hidden Layer Feed-forward neural Network

In Fig. 2.2 the three-layered FNN with one output neuron is shown. These FNNs are also known as Single-hidden Layer Feed-forward neural Networks (SLFNs), and represent the most widely used data-driven model among hydrologist (Maier and Dandy, 2000; Maier et al., 2010). This SLFN can be expressed in mathematical terms as a nonlinear parametrical model of the form

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{L} \beta_i \, G(\boldsymbol{a}_i^T \boldsymbol{x} + b_i) + b_0 \tag{2.1}$$

where $\boldsymbol{x}$ is an input vector of p variables which is fed to the network at a given time; $\hat{y}(\boldsymbol{x})$ is the output returned by the SLFN in response to $\boldsymbol{x}$; $\boldsymbol{a}_i$ is the set of synaptic weights for the connections going from the input layer the i-th hidden neuron; β_i is the connection weight between the i-th hidden neuron and the output neuron; and L is the total number of hidden neurons. All the b_i terms are called biases, and are additional parameters that improve the quality of the mapping by shifting the activations functions in the processing units. The biases can be represented as weights of synapses leaving dummy units which have a constant output equal to one (see Fig. 2.2). $G(\cdot)$ indicates the activation functions in the hidden units, which are generally of the same form across the entire layer. The activation functions are usually chosen as the logistic sigmoid (2.2a) or the hyperbolic tangent (2.2b)

$$f(x) = \frac{1}{1 + e^{-2s(x+t)}} \tag{2.2a}$$

$$f(x) = \frac{e^{2s(x+t)} - 1}{e^{2s(x+t)} + 1} \tag{2.2b}$$

Both functions are smooth and differentiable, and have very similar graphs. The major difference between the two functions lies in the fact that the output of the logistic sigmoid ranges from 0 to 1, while that of the hyperbolic tangent ranges from -1 to 1 (Fig. 2.3). The parameters *s* and *t* identify the *steepness* and *threshold* of the sigmoidal functions. The steepness is the slope of the linear portion of the sigmoid, while the threshold is a scalar that pushes the center of the activation function away from zero. The thresholds actually correspond to the biases *b* in (2.1) when these functions are employed as ANN activation functions. While the steepness is usually set equal to one, the thresholds are adjusted along with the synaptic weights during the ANN calibration process. Although it is generally understood that one hidden layer is usually sufficient to achieve good performances in hydrological applications (Coulibaly et al., 2000), some studies have shown that additional hidden layers may result in better performances (Chen and Chang, 2009; Tomassetti et al., 2009; Trichakis et al., 2009). However, adding hidden layers to a network model may result in an error surface with more local minima that complicates the calibration of the network parameters (Masters, 1993).

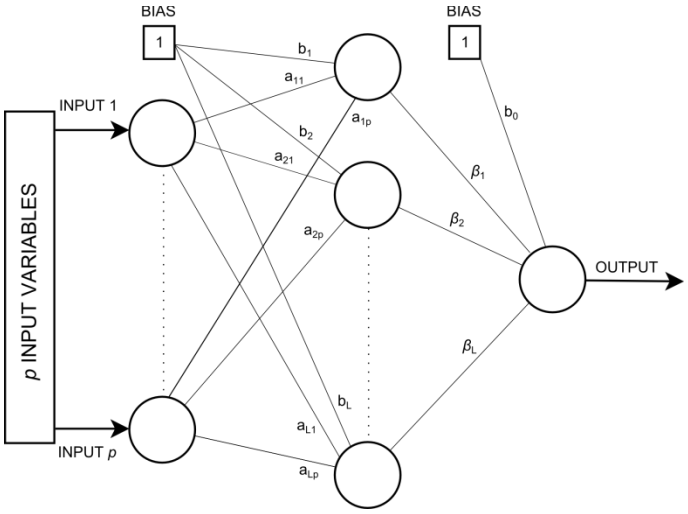


Figure 2.2. Single-hidden Layer Feed-forward neural Network

2.1.3. Neural network training

The optimization of the SLFN parameters, also known as *learning* or *training* in ANN jargon, is carried out to achieve the best approximation of the system output on a set of N input-output pairs $[\boldsymbol{x}(t), y(t)], t = 1,2, \dots, N$. This nonlinear optimization is usually performed by solving a minimization problem of the form

$$\widehat{\boldsymbol{\theta}}_N = argmin \sum_{t=1}^{N} |y(t) - \hat{y}(t|\boldsymbol{\theta})|^2 \tag{2.3}$$

where $y(t)$ is the real output of the system under study at time t; $\hat{y}(t|\boldsymbol{\theta})$ is the output of the SLFN in (2.1), which for a given input data pattern $\boldsymbol{x}(t)$ is a function of the set of network weights and biases $\boldsymbol{\theta} = [\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\beta}]$. $\widehat{\boldsymbol{\theta}}_N$ is the estimation of $\boldsymbol{\theta}$ that minimizes the error function on the N input-output patterns of the training dataset. There is no analytical solution for (2.3); hence the minimization has to be done through a numerical search procedure. First or second order local search techniques, such as the standard back-propagation, the conjugate gradient, or the Levenberg-Marquardt (LM) method are employed to carry out the training process (Haykin, 2008; Masters, 1995, 1993), although global search methods such as Particle Swarm Optimization can also be used (Chau, 2007, 2006), as it will be described in the next Chapter. The following paragraphs will instead provide a short description of the LM algorithm, which has proven to be one of the most efficient training algorithms, and the algorithm of choice for ANN-based hydrological applications (Coulibaly et al., 2000; Nayak et al., 2013; Piotrowski and Napiorkowski, 2011; Sahoo et al., 2006).

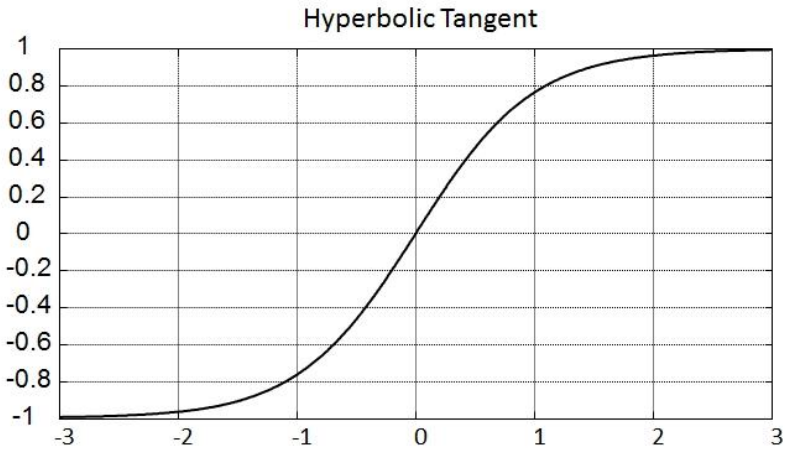


Figure 2.3. Graph of the hyperbolic tangent activation function

2.1.4. The Levenberg-Marquardt algorithm

The nonlinear least squares problem in (2.3) can be expressed as a sum of squared residuals

$$F(\boldsymbol{\theta}) = \frac{1}{2}\sum_{j=1}^{N} r_j^2(\boldsymbol{\theta}) \tag{2.4}$$

For each time *t*, $t = j = 1 \dots N$, the residuals $r_j^2(\boldsymbol{\theta})$ are given by the difference between the real output $y(t)$ and the SLFN predicted output $\hat{y}(t|\boldsymbol{\theta})$ in (2.3)

$$r_j(\boldsymbol{\theta}) = |y(t) - \hat{y}(t|\boldsymbol{\theta})| \tag{2.5}$$

The residual functions r_j can be assembled to form a *residual vector* $r: \mathbb{R}^n \to \mathbb{R}^N$, defined as $\boldsymbol{r}(\boldsymbol{\theta}) = \big(r_1(\boldsymbol{\theta}), r_2(\boldsymbol{\theta}), \dots, r_N(\boldsymbol{\theta})\big)$, where *n* is the number of SLFN parameters and *N* is the length of the training dataset. Using the residual vector, the function $F(\boldsymbol{\theta})$ in (2.4) can be rewritten as

$$F(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{\theta})\|^2, \tag{2.6}$$

and its derivatives are given by the Jacobian matrix $\boldsymbol{J}$ computed with respect to $\boldsymbol{\theta}$

$$\boldsymbol{J}(\boldsymbol{\theta}) = \frac{\partial r_j}{\partial \theta_i} \tag{2.7}$$

where $1 \leq i \leq n$, and $1 \leq j \leq N$. From the Jacobian matrix, the gradient and the Hessian of $F(\boldsymbol{\theta})$ can be obtained as follows

$$\boldsymbol{\nabla} F(\boldsymbol{\theta}) = \sum_{j=1}^{N} r_j(\boldsymbol{\theta}) \boldsymbol{\nabla} r_j(\boldsymbol{\theta}) = \boldsymbol{J}(\boldsymbol{\theta})^T \, \boldsymbol{r}(\boldsymbol{\theta}) \tag{2.8}$$

$$\boldsymbol{H} = \boldsymbol{\nabla^2} F(\boldsymbol{\theta}) = \boldsymbol{J}(\boldsymbol{\theta})^T \boldsymbol{J}(\boldsymbol{\theta}) + \sum_{j=1}^{N} r_j(\boldsymbol{\theta}) \boldsymbol{\nabla^2} r_j(\boldsymbol{\theta}) \tag{2.9}$$

If the residuals are small, or at least the value of their second derivatives is negligible, the Hessian *H* can be written as

$$\boldsymbol{H} = \boldsymbol{\nabla^2} F(\boldsymbol{\theta}) = \boldsymbol{J}(\boldsymbol{\theta})^T \boldsymbol{J}(\boldsymbol{\theta}) \tag{2.10}$$

which is the common approximation of near-linearity of the residuals near the solution. The LM algorithm can be defined as a blend of vanilla gradient descend and the Gauss-Newton method, and employs the information of both first and second order derivatives. Vanilla gradient descent is the simplest and most intuitive method to find minima in a function. The parameter vector is updated by just subtracting the gradient at each step after multiplying it for a scaling coefficient ϑ

$$\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^j - \vartheta \boldsymbol{\nabla} F(\boldsymbol{\theta}) \tag{2.11}$$

This technique suffers from big convergence problems. Where the error surface has a slight slope, the search point is far from the minima. In these cases the method takes small steps and fails to hasten towards the solution. On the other hand, the method

accelerates where the gradient is high, rattling out of the minima while approaching. This is clearly the opposite of what is expected from a good optimization technique. Another problem is that the curvature of the error surface may not be the same in all directions; therefore the search may be misdirected by gradient information only. This issue can be tackled by using curvature information, namely the second order derivatives or the Hessian matrix. The gradient $\boldsymbol{\nabla} F(\boldsymbol{\theta})$ in (2.8) can be expanded using a Taylor series around the current state $\boldsymbol{\theta}^j$, which yields

$$\begin{gathered}\boldsymbol{\nabla} F(\boldsymbol{\theta}) = \boldsymbol{\nabla} F\left(\boldsymbol{\theta}^j\right) + \left(\boldsymbol{\theta} - \boldsymbol{\theta}^j\right)^T \boldsymbol{\nabla^2} F\left(\boldsymbol{\theta}^j\right) + \\ + higher\ order\ terms\ of\ \left(\boldsymbol{\theta} - \boldsymbol{\theta}^j\right)\end{gathered} \tag{2.12}$$

If $F(\boldsymbol{\theta})$ is supposed to be quadratic around $\boldsymbol{\theta}^j$, the higher order terms may be neglected and the value of $\boldsymbol{\theta}$ is obtained by setting the left side of (2.12) equal to zero. This operation yields the update rule for the Newton's method

$$\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^j + \left(\boldsymbol{\nabla^2} F\left(\boldsymbol{\theta}^j\right)\right)^{-1} \boldsymbol{\nabla} F\left(\boldsymbol{\theta}^j\right) \tag{2.13}$$

where the generic $\boldsymbol{\theta}$ has been replaced with $\boldsymbol{\theta}^{j+1}$. The Newton's method uses the approximation (2.10) for Hessian computation because of the implicit quadratic assumption that leads to (2.13) from (2.12). Although this technique shows rapid convergence, it is highly dependent on the linearity around the starting location, thus needs to be improved. This improvement lies in the observation that the gradient descent and the Gauss-Newton iteration are complementary in the advantages they provide. Levenberg (1944) therefore proposed a blend of them as a superior update rule, which can be written as

$$\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^j + (\boldsymbol{H} + \vartheta \boldsymbol{I})^{-1} \boldsymbol{\nabla} F\left(\boldsymbol{\theta}^j\right) \tag{2.14}$$

where $\boldsymbol{H}$ is the Hessian matrix evaluated at $\boldsymbol{\theta}^j$. The algorithm operates by first updating the weights according to (2.14) and measuring the error associated with the new parameter vector $\boldsymbol{\theta}^{j+1}$. If the error gets smaller with respect to the previous one, the near-linearity condition is strengthened, and the parameter ϑ is reduced before performing the next update. When the opposite occurs, the last update is erased and another step from $\boldsymbol{\theta}^j$ is taken after increasing ϑ of some significant factor. The algorithm can still be refined by using second order derivatives even when ϑ is large. By scaling each component of the gradient according to the curvature, larger movement will result along those directions where the gradient is smaller thus solving the biggest issue related to the gradient descent algorithm. This insight has been provided by Marquardt (1963), and as a consequence the final Levenberg-Marquardt parameter update rule can be written as

$$\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^j + (\boldsymbol{H} + \vartheta \cdot diag[\boldsymbol{H}])^{-1} \boldsymbol{\nabla} F(\boldsymbol{\theta}^j) \tag{2.15}$$

The LM algorithm has proven to work extremely well in practice. The only disadvantage compared to other gradient based methods, such as the conjugate gradient, lies in the matrix inversion which has to be performed at each update and can require long time for computation. One problem that this method shares with other gradient based techniques is that it tends to get stuck in areas of local minima which can be far away from the global optimum. That is why several algorithm restarts should be carried out when employing such techniques for ANN training (Piotrowski and Napiorkowski, 2011).

2.1.5. ANN generalization

ANN generalization refers to the ability of the neural network to reproduce the behavior of the system under study in situations which are not represented in the training dataset (Anctil and Lauzon, 2004). Failing to reach good minima of the error function $F(\boldsymbol{\theta})$ may lead to poor generalization due to *underfitting*, i.e. the model failing to emulate the system because it was not able to capture the underlying relationship training dataset. While the chances of underfitting can be reduced with multiple algorithm restarts, this event will always occur when the input set lacks meaningful predictors, or when excessively simple models are used. On the other hand, *overfitting* happens when the model is too complex, or when the calibration process is over-extended causing the ANN to fit the random noise in the training dataset. While both phenomena lead to poor generalization, overfitting is the most common in hydrological data-driven applications. Hereby follows a description of the techniques usually employed to prevent overfitting and grant ANN generalization.

- *Early stopping*. This method is the most widely used in hydrology and water resources, where it is also referred to as stop training or cross-validation (ASCE Task Committee, 2000; Cannon and Whitfield, 2002; Coulibaly et al., 2001, 2000; Leahy et al., 2008; Piotrowski and Napiorkowski, 2013). The early stopping criterion entails splitting the training data set in two parts to form a completely disjoint validation dataset. The calibration of ANN parameters is carried out on the reduced training dataset, but now the model prediction error on the validation dataset is constantly monitored, so that the training process is interrupted when this error start to increase, as shown in Fig. 2.4.

- *K-fold cross-validation*. Although not as popular as early stopping, multifold cross-validation can also be employed to grant model generalization, especially when data is scarce (Chang et al., 2010; Piotrowski and Napiorkowski, 2013). This variant is usually referred to as *k*-fold cross-validation (Kohavi, 1995), where *k* is the number of folds in which the training dataset is divided before model calibration is initiated. The learning process is performed a total of *k* times, and at each time a different fold is used for validation while the remaining $k - 1$ folds are used for training. Values of $k = 5$ or 10 are typically used in data-driven hydrological applications (Galelli and Castelletti, 2013a, 2013b). This approach allows for a better exploitation of the available data, and should theoretically provide a better estimate of the real generalization error. However, the multiple runs of optimization involved in *k*-fold cross-validation may severely slow down the overall ANN learning process compared to early stopping. In addition, while early stopping returns a single ANN which can be used for predictions on new data, *k*-fold cross-validation requires *ensembling* operations to obtain a single output from the multiple ANNs available (Cannon and Whitfield, 2002).

- *Regularization*. This approach reduces the chance of overfitting by preventing the values of the ANN parameters to get large during the calibration process (Anctil et al., 2004). In this way, the ANN produces a smoother response which is less susceptible to noise in the data. Regularization is implemented by modifying the objective function of the nonlinear least square problem to include an additional term which penalizes large synaptic weights. A common form of the modified objective function can be written by adding the squared ℓ^2 norm of the parameters $\boldsymbol{\theta}$ to (2.4), which yields

$$F(\boldsymbol{\theta}) = \frac{1}{2}\left(\sum_{j=1}^{N} r_j^2(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{n} \boldsymbol{\theta^2}\right) \tag{2.16}$$

where the hyper parameter λ controls the strength of the regularization. Setting the optimal value for λ is a problem of this approach, which can be overcome by resorting to Bayesian regularization (Foresee and Hagan, 1997; Khan and Coulibaly, 2006).

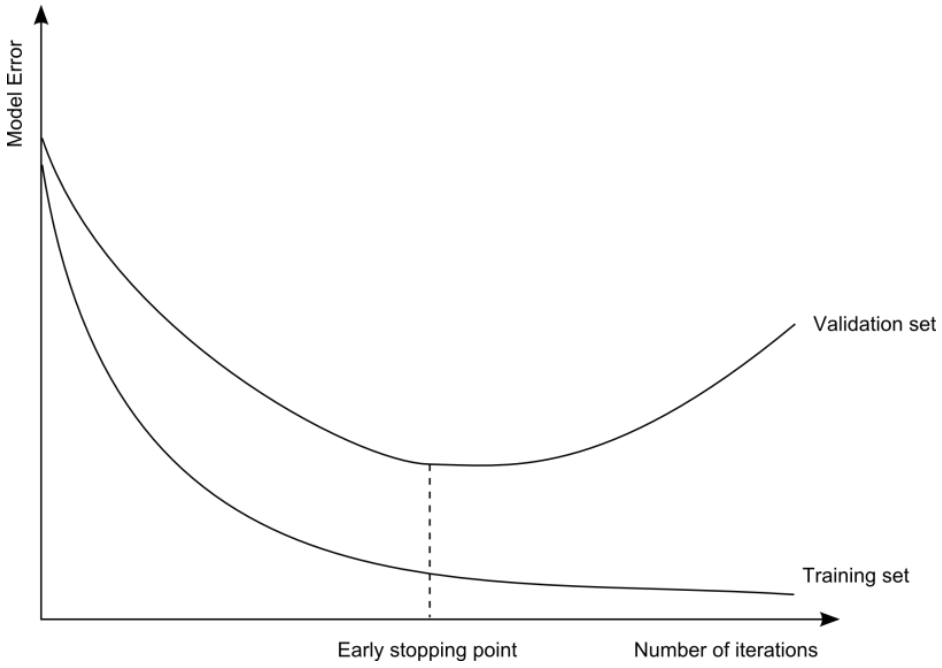


Figure 2.4. The early stopping criterion

2.2. Extreme Learning Machines

The Extreme Learning Machines (ELM) paradigm was initially proposed as a training algorithm for SLFNs in which the input weights and hidden biases are randomly assigned. This operation reduces the SLFNs to a linear system, and allows for the analytical determination of the output weights using common least-squares (Huang et al., 2006b, 2004). These simplifications drastically increase the speed of the learning process, which was also found to grant ELMs better generalization than traditional

SLFNs trained with gradient-based algorithms or Support Vector Machines (SVM). It was successively shown that ELMs are universal approximators that can work with a broad type of activation functions, as long as they are bounded non-constant piecewise continuous (Huang et al., 2011, 2006a). Recently, it was demonstrated that ELMs actually represent a simple unified learning framework for SLFNs, polynomial networks, SVMs and other data-driven techniques which can be applied to both regression and multiclass classification problems (Huang et al., 2012). In addition, ELMs do not require any scheme for improving their generalization capability, since the solution returned by this learning paradigm is not only the one with the smallest training error, but also the one minimizing the norm of output weights (see Section 2.1.5). K-fold cross-validation can be employed to obtain a more accurate estimate of the generalization error. Despite their considerable advantages, there exists only a handful of ELM applications in hydrology and related fields. Acharya et al. (2013) tested ELM for building multi-model ensembles from the products of general circulation models developed to estimate the northeast monsoon rainfall over the southern India. The findings showed that ELM outperformed standard ensembling methods according to several skill metrics. Ortiz-García et al. (2014) used ELMs, as well as several other data-driven techniques, in a problem of daily precipitation prediction formulated as a classification problem. Deo and Şahin (2015) showed that ELMs could substantially outperform ANN models trained with LM backpropagation for predicting future droughts in eastern Australia. The improvements were obtained both in terms of better performance metrics, as well as considerably faster training speeds. In this work, ELM will be employed as a fast and accurate alternative to ANN whenever the task at hand

does not require the use of GO methods to fine tune the neural network parameters. Indeed, an ELM is topologically equal to a SLFN with no biases in the output layer, such as shown in Fig. 2.5 for an ELM with one output neuron. Therefore, when GO is employed as the learning paradigm, it makes no difference whether the underlying model is an ELM or a SLFN.

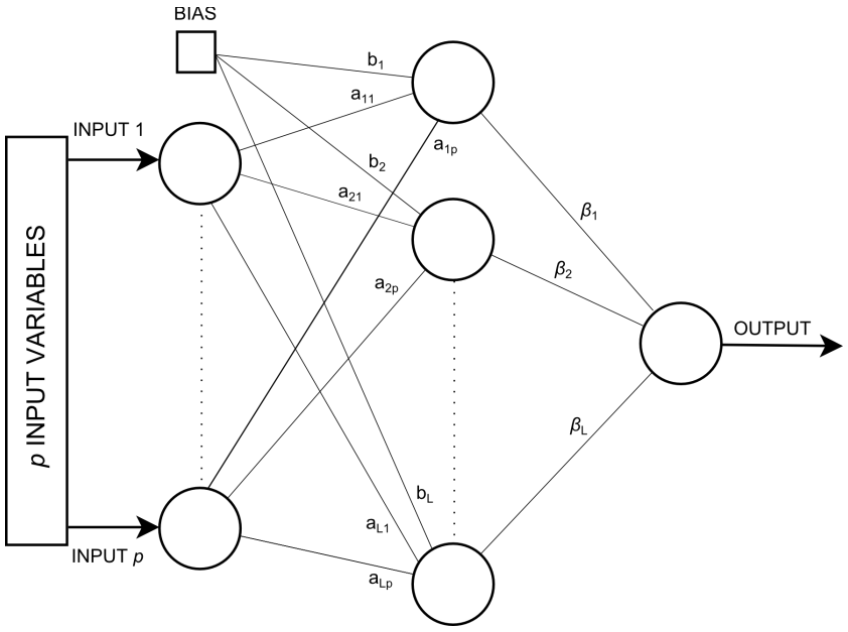


Figure 2.5. Extreme Learning Machine with one output neuron

For a given pattern $\boldsymbol{x}$ of p input variables $\boldsymbol{x} \in \mathbb{R}^p$, the output of an ELM with L hidden nodes and q output nodes is given by

$$f_L(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) \qquad (2.17)$$

where $\boldsymbol{a}_i \in \mathbb{R}^p$, $b_i \in \mathbb{R}$, $\boldsymbol{\beta}_i \in \mathbb{R}^q$, and $G: \mathbb{R}^{p+1} \to \mathbb{R}$ are respectively the input weights, bias, output weights, and activation function of the i-th hidden node. For a dataset of N

input-output patterns $(\boldsymbol{x}_j, \boldsymbol{t}_j) \in \mathbb{R}^{\boldsymbol{p}} \times \mathbb{R}^{\boldsymbol{q}}$, the parameters $\boldsymbol{\beta}$ of the ELM can be estimated by solving the following linear system with ordinary least squares (Huang et al., 2011)

$$\boldsymbol{H\beta} = \boldsymbol{T} \tag{2.18}$$

where

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \cdots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_N) \end{bmatrix}_{N \times L}$$
$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L \times q}, \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N \times q} \tag{2.19}$$

$\boldsymbol{H}$ is called the hidden layer output matrix of the ELM and can be computed after randomly assigning the hidden node parameters $(\boldsymbol{a}_i, b_i)$. The smallest norm least-square solution $\widehat{\boldsymbol{\beta}}$ of (2.18) can be thus found as

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{H}^{+}\boldsymbol{T} \tag{2.20}$$

where $\boldsymbol{H}^{+}$is the Moore-Penrose generalized inverse of $\boldsymbol{H}$, which has to be used instead of the inverse, since generally $L<<N$. A better and stabler solution than (2.20) can be obtained through ridge regression theory (Huang et al., 2012)

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{H}^T \left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H}\boldsymbol{H}^T \right)^{-1} \boldsymbol{T} \tag{2.21}$$

where $1/\lambda$ is a positive value added to the diagonal of $\boldsymbol{H}\boldsymbol{H}^T$. In case $\boldsymbol{H}\boldsymbol{H}^T$ is singular, an alternative version of (2.21) has to be used for the estimation of $\widehat{\boldsymbol{\beta}}$

$$\widehat{\boldsymbol{\beta}} = \left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H}^T\boldsymbol{H} \right)^{-1} \boldsymbol{H}^T\boldsymbol{T} \tag{2.22}$$

The overall complexity m of an ELM is given by the sum of input layer $(\boldsymbol{a}_i, b_i)$ and hidden layer $(\boldsymbol{\beta})$ parameters, which for an ELM with one output neuron is equal to

$m = (p + 1) \cdot L + L$. After the output weights have been determined, the ELM can be employed for prediction on a test dataset. The ELM algorithm can be summarized as follows (Huang et al., 2011):

ALGORITHM I. ELM training

Given a dataset of N input output pairs $(\boldsymbol{x}_j, \boldsymbol{t}_j) \in \mathbb{R}^{\boldsymbol{p}} \times \mathbb{R}^{\boldsymbol{q}}$, for an ELM with L hidden nodes and activation functions $G(\boldsymbol{a}_i, b_i, \boldsymbol{x})$:

1. Assign random values to the connection weights and biases entering the hidden layer $(\boldsymbol{a}_i, b_i)$, i=1,…,L
2. Construct the hidden layer output matrix $\boldsymbol{H}$ according to (2.19)
3. Calculate the output weight vector $\widehat{\boldsymbol{\beta}}$ by employing either (2.21) or (2.22)

3. Swarm Optimization

This chapter illustrates the BFIPS, MOFIPS and MBFIPS algorithms which have been devised to develop the NNRF applications at the core of this thesis work. These methods are presented after some relevant context has been provided. Apart from containing the necessary background on Swarm Optimization, these preliminary sections briefly describe global optimization and how it can be employed to construct NNRF models.

3.1. Introduction to global optimization

Global optimization (GO) entails the minimization, or maximization, of an objective function through a multipoint search on the error surface. Among the most successful techniques, we have purely stochastic search methods, such as Simulated Annealing (Leahy et al., 2008; Masters, 1993) and Pattern Search (Corzo and Solomatine, 2007a, 2007b), as well as heuristic strategies that are usually nature-inspired algorithms. In the last decades, nature-inspired computation has witnessed a tremendous growth, with successful applications in most fields of science and engineering (Brownlee, 2012; Yang, 2014). Many of these techniques have been widely used also in water resources science and engineering, especially those belonging to the families of Evolutionary Computation (EC) and Swarm Intelligence (SI). Genetic Algorithms (GA) is the most well-known EC technique, and arguably the most popular heuristics among hydrologists (Abrahart et al., 1999; Bowden et al., 2005a, 2005b; Chen and Chang, 2009; Dawson et al., 2006; de Vos and Rientjes, 2008; Goldberg, 1989; Jain and Srinivasulu, 2004; Mohan and Vijayalakshmi, 2009; Mohan, 1997; Prasad and Park,

2004; Sedki et al., 2009; Wu et al., 2012; Yapo et al., 1998). Other notable EC examples, that have found applications in hydrology and related fields, are Genetic and Evolutionary Programming (Dawson et al., 2006; Nourani et al., 2012; Parasuraman et al., 2007; Savic et al., 1999), and, more recently, Differential Evolution (Kişi, 2010; Piotrowski and Napiorkowski, 2011; Piotrowski et al., 2012a, 2012b; Storn and Price, 1997). In EC algorithms an initial population of candidate solutions evolves through time following rules adapted from evolutionary biology and Darwin's theory of the survival of the fittest. Successive generations of the initial population are created in order to maximize the overall fitness, i.e. minimize or maximize the objective function of the problem at hand. Typical updating rules for EC algorithms are recombination, or crossover, mutation and selection. On the other hand, the optimization of SI heuristics is carried out through the interaction of a population of agents mimicking the collective behavior of natural systems, such as ant colonies, fish schooling and bird flocking. Individuals in the population strive to improve themselves by imitating traits found in their successful peers. Examples of SI algorithms are Ant Colony Optimization (Afshar, 2007; Dorigo et al., 1996; Kumar and Reddy, 2006), Artificial Bee Colony (Karaboga and Basturk, 2007; Kisi et al., 2012), and Particle Swarm Optimization (PSO), which has probably been the most successful due to its computational efficiency and inherent simplicity (Kennedy and Eberhart, 1995; Poli et al., 2007). Typical applications of PSO in the field of water resources range from soil moisture estimation (Gill et al., 2006; Lü et al., 2011) and the optimization of conceptual rainfall-runoff models (Jiang et al., 2007; Li et al., 2009; Tada and Beven, 2012), to reservoir operations (Gaur et al., 2011; Li and

Zhang, 2009; Reddy and Kumar, 2007a) and supporting decision making in water resources management (Gaur et al., 2011; Guo et al., 2012; Reddy and Kumar, 2007b).

3.2. ANN development with global optimization techniques

When employed for NNRF development, GO techniques perform a parallel multipoint search on the error surface in which each point represents a hypothesis on : 1) the set of ANN model parameters (Chau, 2007, 2006; Chau et al., 2005; de Vos and Rientjes, 2008, 2007; Jain and Srinivasulu, 2004; Kisi et al., 2012; Piotrowski and Napiorkowski, 2011; Sedki et al., 2009; Wu and Chau, 2006); 2) a subset of model inputs to be selected in a pool of candidates (Bowden et al., 2005a, 2005b); 3) some details of the ANN architecture or its connectivity (Abrahart et al., 1999; Chen and Chang, 2009; Corzo and Solomatine, 2007a, 2007b); or 4) the set of model parameters along with network connectivity (Abrahart et al., 2007; Dawson et al., 2006; Leahy et al., 2008; Yao, 1999).

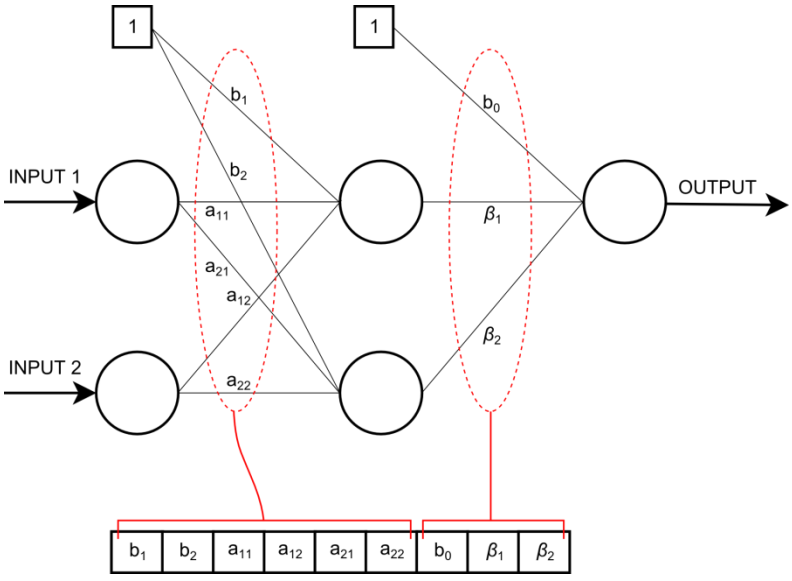


Figure 3.1. Real-valued encoding of a SLFN wih one output neuron

Since these heuristics do not employ derivatives in their optimization process, they can operate on search spaces which are not continuous or differentiable, and can optimize a much wider range of objective functions than the sum of errors in (2.3). Each potential solution of the ANN model is coded on a numerical string, or *genotype*, which constitutes a population member of the global search algorithm. In contrast, the extended mathematical form coded in the string and the relative ANN graph are defined as the *phenotype*. Though these terms belong to EC jargon, they are also employed in the SI context. If the optimization is carried out for the calibration of ANN model parameters, i.e. the synaptic weights and biases, then the genotype is usually made of real values, as shown in Fig. 3.1 for a SLFN with one output neuron. On the other hand, if the optimization process is aimed at input selection and/or network connectivity optimization, the genotype is a binary string, where 1 represents that a connection is established between 2 different nodes of the ANN. Fig. 3.2 shows an example of binary encoding for a SLFN with three inputs and four hidden units. In this particular ANN realization one of the inputs has not been selected and some of the connections between the different layers are missing. When GO techniques are used for the concurrent optimization of model parameters and network connectivity, hybrid real-binary encoding schemes can be employed, although binary codification could be extended also to synaptic weights. In this case the values of the model parameters are obtained by decoding the bits assigned to each of them in the binary string. Regardless of the encoding scheme adopted, the NNRF optimization process is carried out by updated the genotypes according to the rules of the particular algorithm employed. At each iteration, the fitness of population members is assessed by first decoding the genotype into the

phenotype, and then computing the ANN output according to the mathematical form corresponding to the phenotype. The process is usually stopped when a defined number of iterations has been reached. Other common choices are to stop the algorithm if no improvement in the fitness is witnessed over a certain number of iterations, or when an exit criterion on the fitness function has been met.

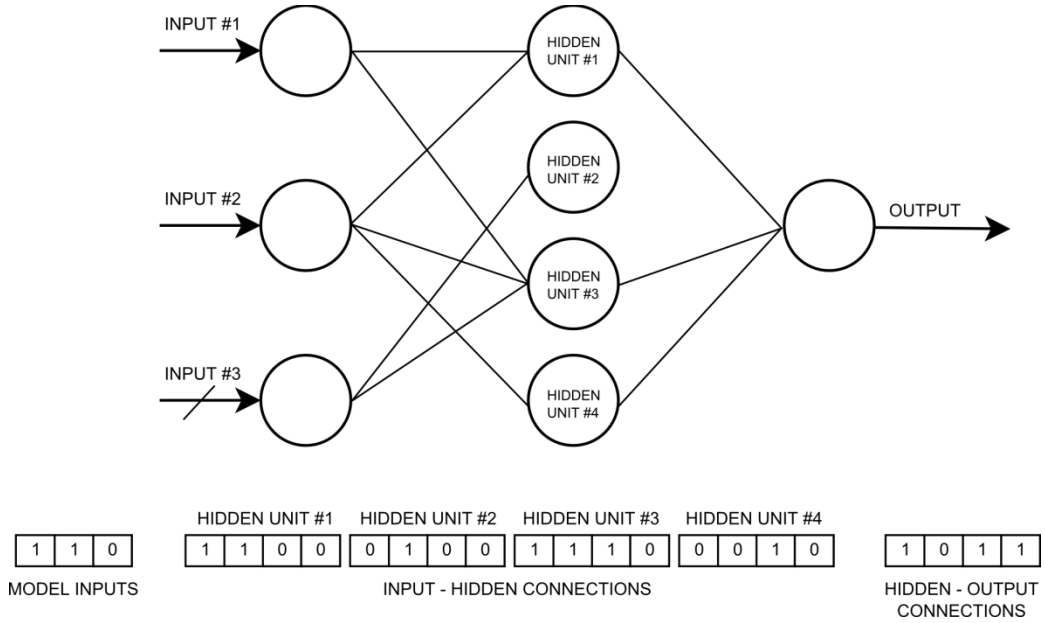


Figure 3.2. Binary encoding of a SLFN wih one output neuron

3.3. The canonical Particle Swarm Optimization algorithm

Recently, the Particle Swarm Optimization (Kennedy and Eberhart, 1995) method has gained popularity due to its computational efficiency and relative ease of implementation compared to other GO approaches. PSO performs its search by exploiting cooperation within a population (swarm) of potential solutions (particles). The swarm flies across the error surface defined by the objective function, and at each time particles are identified by their position and velocity, which are vectors of the same

number of elements as the number of problem dimension *d*. Particle positions are the PSO equivalent to the genotypes of EC algorithms. They represent the hypotheses on the solution of the optimization problem, and each of them is characterized by a different fitness value. At each iteration, a particle flies to a new position according to its velocity, which in the original version of the PSO algorithm is a function of the historical best position achieved by the particle and the historical best position found among all the particles in its neighborhood (see Fig. 3.3). If $\boldsymbol{X}_i$ is the position of the *i*-th particle, $\boldsymbol{V}_i$ is its velocity, $\boldsymbol{P}_i$ its personal best, and $\boldsymbol{G}$ the overall best position in the particle neighborhood, the PSO algorithm is defined by the following equations for velocity and position update (Poli et al., 2007) :

$$\boldsymbol{V}_i \leftarrow \chi\big(\boldsymbol{V}_i + \boldsymbol{U}(0,\varphi_1) \otimes (\boldsymbol{P}_i - \boldsymbol{X}_i) + \boldsymbol{U}(0,\varphi_2) \otimes (\boldsymbol{G} - \boldsymbol{X}_i)\big) \qquad (3.1)$$

$$\boldsymbol{X}_i \leftarrow \boldsymbol{X}_i + \boldsymbol{V}_i \qquad (3.2)$$

where $\otimes$ identifies point-wise multiplication; $\boldsymbol{U}(0,\cdot)$ is a vector of d of uniformly distributed random numbers between 0 and a positive real number; $\varphi = \varphi_1 + \varphi_2 > 4$ and

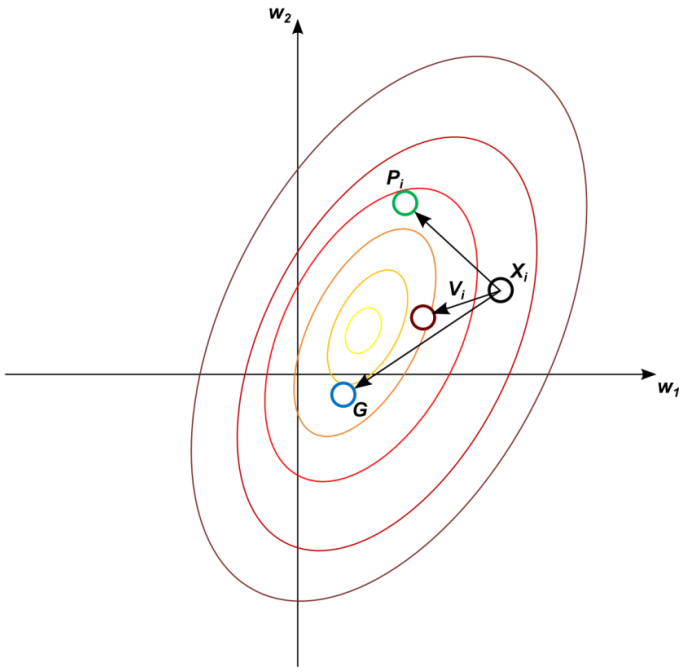
$$\chi = \frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}} \qquad (3.3)$$

Eq. (3.1 – 3.3) above represent Clerc's constriction formulation of canonical PSO (Clerc and Kennedy, 2002). A common choice that guarantees convergence is to set $\varphi = 4.1$ with acceleration constants $\varphi_1 = \varphi_2 = 2.05$. This yields a value of $\chi \cong 0.7298$ for the constriction coefficient. Although these settings force particle convergence and prevent the PSO algorithm from exploding, a better approach is to limit particle positions within an interval $[X_{MIN}, X_{MAX}]$ that depends on the problem, and then constrain particle velocities within $[V_{MIN}, V_{MAX}]$ where $V_{MIN} = X_{MIN}$ and $V_{MAX} = X_{MAX}$ (Eberhart and Shi,

2000). This results in a PSO algorithm with no problem-specific parameters, which is known as the canonical PSO algorithm (Poli et al., 2007), and can be summarized as follows:

ALGORITHM II. Particle Swarm Optimization

1. Randomly initialize particle positions, best positions, velocities, and fitness values.
2. Loop for t = 2 up to a maximum number of iterations.
3. Update particle velocities according to (3.1).
4. Update particle positions according to (3.2) (see Fig. 3.3).
5. Compute particles fitness values for the new positions.
6. Compare fitness values at time t with those at time t-1. Update personal bests where needed.
7. If additional termination criteria are met, exit the loop.
8. Return to step 2.



Particle X_i moves along the velocity V_i to reach the updated position. The velocity is computed based on its previous value, the particle best position P_i and the global best found in its neighborhood G. In this example the error surface is a function of only two parameters w_1 and w_2.

Figure 3.3. Vectorial representation of particle position update.

3.4. Fully Informed Particle Swarm (FIPS) optimization

In canonical PSO only the positions of the neighborhood historical best and personal best are taken into account to update the velocity of each particle. Further studies have argued that discarding all the other information provided by the remaining individuals may actually be detrimental to the whole optimization process, both in terms of reduced convergence speed and final outcome of the search process (Kennedy and Mendes, 2002; Kennedy, 1999; Mendes, 2004; Mendes et al., 2003). This lead to the development of the Fully Informed Particle Swarm (FIPS) optimization method (Mendes et al., 2004), where the term *fully informed* indicates that each particle neighbor contributes in modifying its trajectory. The velocity update equation for the FIPS is a generalization of (3.1), which can be defined as follows (Poli et al., 2007):

$$\boldsymbol{V}_i \leftarrow \chi\left(\boldsymbol{V}_i + \frac{1}{K_i}\sum_{j=1}^{K_i} \boldsymbol{U}(0,\varphi) \otimes \left(\boldsymbol{P}_{nbr_j} - \boldsymbol{X}_i\right)\right) \tag{3.4}$$

where K_i is the number of neighbors for particle i, and nbr_j identifies i's j-th neighbor. It can be seen that this version corresponds to (3.1) if $K_i = 2$ for each particle with $\boldsymbol{P}_{nbr_1} = \boldsymbol{P}_i$, and $\boldsymbol{P}_{nbr_2} = \boldsymbol{G}$. It is important to note that, contrary to canonical PSO, neighborhoods in the FIPS paradigm may or may not include the best position of the i-th particle itself. Although the FIPS algorithm was found to outperform canonical PSO on benchmark case studies (Mendes, 2004; Mendes et al., 2004), its performances depend more on the chosen swarm *topology*, i.e. the graph defining how particles are arranged and interconnected in the swarm to form neighborhoods and perform the collective search.

Alternative FIPS formulation

All the FIPS variants introduced in this thesis work will employ an alternative version of (3.4) for particle velocity update. In particular, the velocity update equation in (3.4) is simplified as follows:

$$\boldsymbol{V}_i \leftarrow \chi\left(\boldsymbol{V}_i + \varphi\left(\left(\sum_{j=1}^{K_i} a_j \boldsymbol{P}_{nbr_j}\right) - \boldsymbol{X}_i\right)\right) \tag{3.5}$$

In (3.5) the contribution of each neighbor is weighted using a normalized random coefficient a_j which is the same for each particle dimension, and is defined as

$$a_j = \frac{b_j}{\sum b_j}, \quad b_j = U(0,1), \quad j = 1, \cdots, K_i \tag{3.6}$$

where $U(0,1)$ identifies a uniform random number between 0 and 1. This version allows for faster computation of particle velocities since only a single uniform random number has to be generated for each neighbor particle instead of one for each problem dimension. In addition, early preliminary trials have shown that this simplification may improve search performances over both PSO and FIPS. The pseudo-code for this alternative FIPS algorithm is equal to ALGORITHM II, with the only difference that equation (3.5) and (3.6) are employed for the particle velocity update in step 3.

3.5. Swarm topologies

The population topology introduced with the original version of the PSO was the fully connected *sphere* in Fig. 3.4a. The configuration of this graph shows that there is only one neighborhood that comprises all the particles in the swarm, which are influenced by a single global-best performer when updating their search direction. This

topology therefore converges rapidly towards a minimum which, however, has good chances of being only a local one. Other topologies were introduced to overcome this problem, and find different balances between the *exploration* and *exploitation* phase of the search (Kennedy and Eberhart, 1997; Kennedy and Mendes, 2002; Mendes, 2004; Mendes et al., 2004, 2003; Poli et al., 2007). Here by exploitation we intend the convergence of the swarm toward a local minimum by sharing information among the particles; while exploration entails free roaming of particles and groups of particles to search for lower minima of the error surface. Some of these alternative topologies are shown in Fig. 3.4b-g. The first alternative topology that was introduced is the *ring* topology in Fig. 3.4b, where each particle has only two neighbors influencing its movements. This entails that if a particle ends up in a good local minimum, it will take time for particles on the opposite end of the ring to receive such information. These particles will have more time to explore thoroughly there area of the error surface before being influenced. The ring topology therefore favors exploration towards exploitation, and results have shown that it outperforms the sphere when searching for the global minimum (Kennedy, 1999). The number of neighbors in the ring-structure can be increased to speed up convergence while maintaining a tendency for superior exploration over the sphere, such as shown in Fig. 3.4c for a ring topology with 4 neighbors. Another alternative topology which has been introduced early on is the *wheel* in Fig. 3.4d, where one particle is connected to all the swarm while the remaining particles cannot communicate between them. In Fig. 3.4e a topology made of four interconnected *clusters* is depicted. In this arrangement, sharing of information is immediate within each cluster, while communications between different clusters is

granted by connector particles. Other configurations are multi-dimensional lattices (Fig. 3.4f), or resemble 3-D shapes, such as the *pyramid* of Fig. 3.4f, or the *Von Neumann* geometry in Fig. 3.4g. The latter one, in which particles are connected to neighbors above, below and on each side of a bi-dimensional lattice, has shown very promising results on a test-bed of well-known functions for nonlinear optimization (Kennedy and Mendes, 2002).

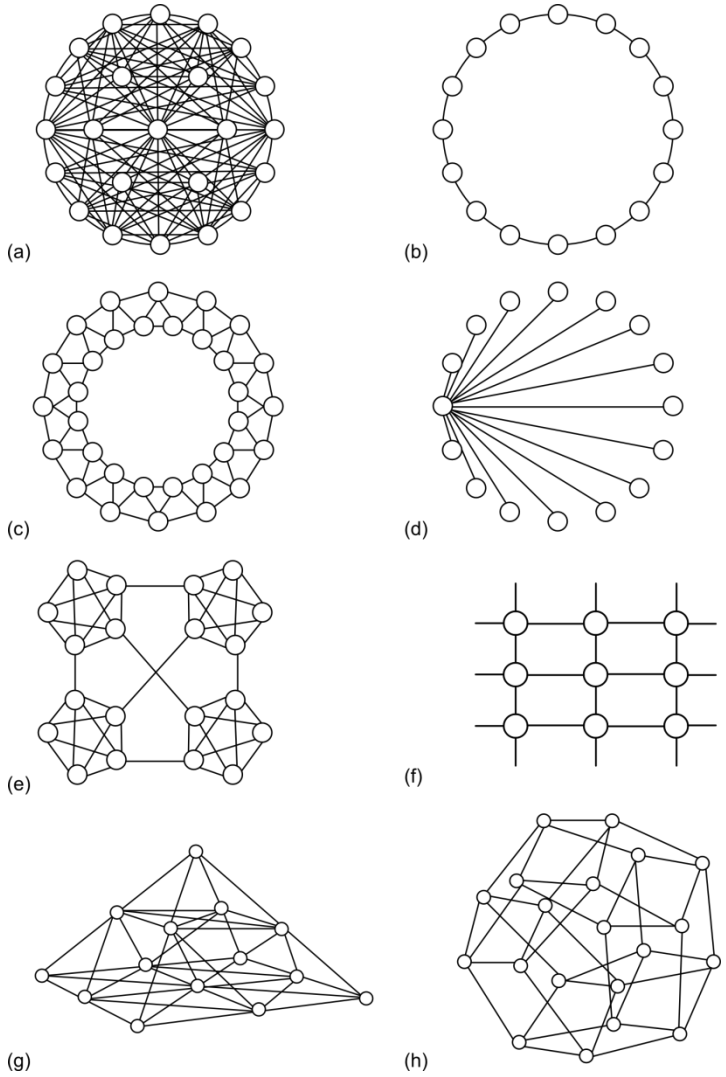
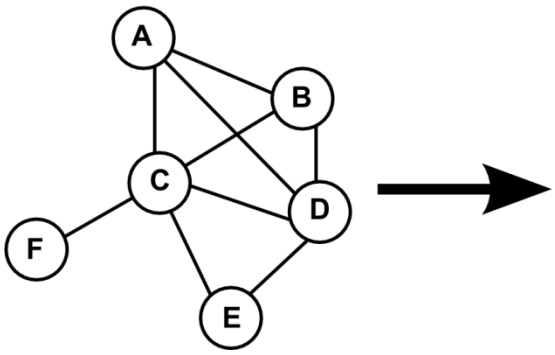


Figure 3.4. Example of swarm topologies: (a) sphere; (b) ring with 2 neighbors; (c) ring with 4 neighbors; (d) wheel; (e) topology with 4 clusters and connectors; (f) bi-dimensional lattice; (g) pyramid; (h) Von Neumann topology.

The implementation of different topologies can be done through binary square *adjacency matrices* in which a 1 defines an existing connection between 2 particles. In Fig. 3.5 a random topology with 6 particles is shown for reference. The ones along the matrix diagonal indicate that the neighborhood of each particle in this topology also includes the particle historical personal best. Since neighborhood relationships are assumed to be reciprocal, the connections between particles are depicted as non-directional, and the resulting adjacency matrix is symmetric. However, in the context of fully informed swarms, it has been pointed out that successful particles can benefit from removing links with the least performing ones in their neighborhood, thus limiting the flow of misguiding information (Mendes, 2004). In this case the matrix representing the topology is asymmetric.



Pt.	A	B	C	D	E	F
A	1	1	1	1	0	0
B	1	1	1	1	0	0
C	1	1	1	1	1	1
D	1	1	1	1	1	0
E	0	0	1	1	1	0
F	0	0	1	0	0	1

Figure 3.5. Matricial encoding of a swarm topology with reciprocal connections.

3.6. Binary-coded Swarm Optimization: the BFIPS algorithm

A paradigm for discrete Binary-coded PSO (BPSO) was devised by Kennedy and Eberhart (1997) shortly after the introduction of the original real-coded PSO. In the BPSO algorithm the trajectories represent changes in the probability that a particular coordinate (bit) will take on a value of zero or one. Although the particle positions are

now binary strings, the equation for velocity update remains unchanged. Particle velocities are still vectors of real values which, in order to express a probability, are constrained in the continuous range [0 1] using a logistic transformation. If $S(V_{ij}) = \frac{1}{1+e^{-V_{ij}}}$ is larger than a uniform random number drawn between 0 and 1, the *j*-th bit of the position array of the *i*-th particle X_{ij} is set to 1; X_{ij} is set to 0 otherwise. The discrete Binary-coded version of the FIPS (BFIPS) is thus obtained by substituting this rule to the equation employed for particle position update in (3.2), while still employing equations (3.5) and (3.6) to perform the velocity update.

According to the binary update rule, a digit in the *j*-th dimension can theoretically change unless V_{ij} is either equal to $-\infty$ or $+\infty$, for which X_{ij} will have a fixed value of 0 or 1, respectively. This implies that the choice of V_{MIN} and V_{MAX} regulates the possibility of further exploration when approaching convergence. Contrary to what happens in real-coded swarm optimization, the logistic transformation in discrete swarms entails that narrow $[V_{MIN}, V_{MAX}]$ ranges will allow for more exploration across the error surface. This range is usually set to [-6, +6] which limits the probabilities $S(V_{ij})$ between 0.0025 and 0.9975, thus preventing the algorithm to stall by always allowing for a minimum chance of discovery. To further foster diversity in the BFIPS, a bit-flipping *mutation operator* is included after particle positions are updated. This operator is implemented by first selecting *m* particles in the swarm with uniform probability. A random uniform number between 0 and 1 is then drawn for each bit of each particle, and the bit is flipped if this number is lower than a preselected scalar known as the mutation rate μ, $\mu \in$ [0 1]. The complete BFIPS optimization algorithm can be thus summarized as follows:

ALGORITHM III. Binary-coded Fully Informed Particle Swarm (BFIPS)

1. Randomly initialize particle positions, best positions, velocities, and fitness values.
2. Loop for t = 2 up to a maximum number of iterations.
3. Update particle velocities according to (3.5-3.6).
4. Constrain velocities with the logistic transformation.
5. Update particle positions based on particle velocities.
6. Compute particles fitness values for the new positions.
7. Compare fitness values at time t with those at time t-1. Update personal bests where needed.
8. Select m particles and perform bit-flipping mutation with mutation rate μ.
9. If additional termination criteria are met, exit the loop.
10. Return to step 2.

3.7. Multi-objective Swarm Optimization: the MOFIPS and MBFIPS algorithms

3.7.1. Pareto-based multi-objective optimization

In Multi-objective Optimization (MO) problems, several conflicting objective functions have to be minimized concurrently (Deb, 2009). The most common strategy employed to solve MO problems is that of Pareto-optimality (Gill et al., 2006; Reddy and Kumar, 2009). Due to the existence of multiple objective functions, the final outcome returned by a Pareto-based algorithm is not a unique solution, but a set of equally good candidates presenting different trade-offs with respect to the objectives. These solutions are said to be non-dominated, or Pareto-efficient, meaning that there is no other candidate showing simultaneously a higher fitness value in all the objective functions defining the problem. When plotting the values of the objective functions against each other, the set of non-dominated solutions describe a frontier known as the

Pareto-front, which is displayed in Fig. 3.6 for a bi-objective problem. The MO algorithm returns an approximation of the optimal, or *true*, Pareto-front, which generally cannot be analytically determined from the problem at hand, and is known only for a restricted number of case-studies usually employed as test functions for MO techniques. A successful MO algorithm should return a large set of solutions which are close to the true Pareto-front, and equally distributed along the frontier. MO generalizations of the PSO (MOPSO) algorithm usually implement the Pareto-based approach by maintaining a set of non-dominated particle positions with respect to the swarm, known as *leading* or *guiding* particles (Coello Coello and Reyes-Sierra, 2006). The positions of these particles are stored separately for reference, and the archive is updated constantly by including new non-dominated solutions, and excluding those which end up being dominated after each optimization step. When the archive grows too big, clustering and trimming operations are carried out to retain those representative solutions offering the most uniform spread along the Pareto-front. The other particles in the swarm can access the archive and direct their search by picking up appropriate leaders through a variety of selection schemes.

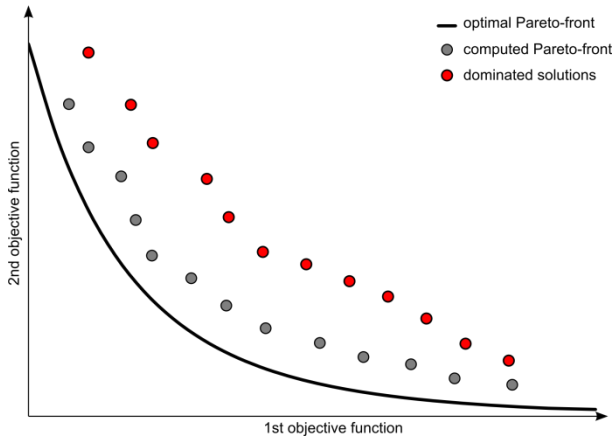
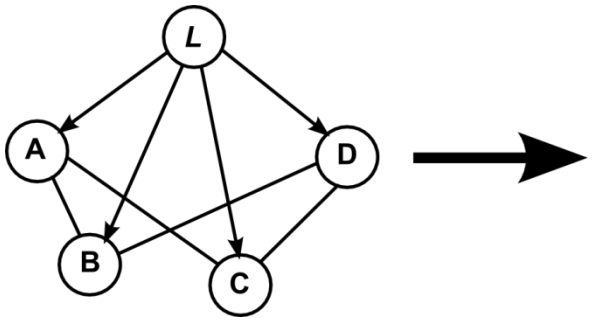


Figure 3.6. Pareto-front of bi-objective optimization problem.

3.7.2. The MOFIPS algorithm

The FIPS paradigm lends itself well for a generalization over multiple objective functions. For the purposes of this thesis work, a novel Multi-objective FIPS (MOFIPS) is introduced based on the alternative FIPS formulation in (3.5-3.6). In MOFIPS, the leading particles constituting the Pareto-front are added to all the neighborhoods in the original swarm topology instead of being stored separately in an archive. This is shown in Fig. 3.7 for the augmented topology of a swarm with four particles and one leader ***L***. The directed connections originating from ***L*** indicate that the particles in the Pareto-front are not subjected to the influence of the other particles, therefore they will not move during the optimization step and their velocity is set to zero. Leading particles are simply instances of the non-dominated positions found during the search, acting as guides, or centers of attraction, for the entire swarm.



Pt.	A	B	C	D	*L*
A	1	1	1	0	1
B	1	1	0	1	1
C	1	0	1	1	1
D	0	1	1	1	1
L	0	0	0	0	0

Figure 3.7. Matricial encoding of a MOFIPS swarm topology with one leader.

The Pareto-front is updated at each iteration by including new non-dominated positions until a maximum size *M* of the front is reached. After this maximum is reached, the update is be carried out by taking into account the *crowding distance* associated with each non-dominated position. The crowding distance is a measure of the density of non-dominated positions in a certain area of the objective function space. It has been firstly

introduced as part of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to foster diversity among possible solutions (Deb et al., 2002). The MOFIPS promotes diversity in the Pareto-optimal set by retaining only the *M* solutions with the highest crowding distances, and discarding the others. To improve swarm convergence and prevent local minima entrapment, a turbulence factor is introduced in the form of a polynomial mutation operator (Deb and Deb, 2012; Deb, 2009). Subject to polynomial mutation, the *j*-th coordinate of particle $\boldsymbol{X}_i$ may change as follows:

$$X_{ij} = \begin{cases} X_{ij} + \delta(X_{ij} - X_{MIN}), & \text{if } u \leq 0.5 \\ X_{ij} + \delta(X_{MAX} - X_{ij}), & \text{otherwise} \end{cases} \tag{3.7}$$

where u is a uniform random number in the [0,1], and δ is a parameter that depends on u and the parameter η_m governing the shape of the polynomial probability distribution. In particular, the value of δ is computed as

$$\delta = \begin{cases} (2u)^{1/(1+\eta_m)} - 1, & \text{if } u \leq 0.5 \\ 1 - (2(1-u))^{1/(1+\eta_m)}, & \text{otherwise} \end{cases} \tag{3.8}$$

This mutation operator can also be applied to all the particles in the Pareto-front. In this case, the frontier is updated by considering the mutated Pareto-front solutions along with the original ones and the updated particle positions $\boldsymbol{X}_i$. The MOFIPS algorithm here described requires the specification of three more parameters with respect to the single-objective version presented in Section 3.4.1. These parameters are: 1) the maximum number *M* of particles in the Pareto-front; 2) the percentage ρ_m of particles coordinates subjected to turbulence; and 3) the shape parameter η_m controlling the probability distribution of the polynomial operator. The pseudo-code for MOFIPS implementation can be written as follows:

ALGORITHM IV. Multi-Objective Fully Informed Particle Swarm (MOFIPS)

1. Initialize particle positions, best positions, velocities, fitness values, and starting Pareto-front.
2. Loop for t = 2 up to a maximum number of iterations.
3. Include Pareto-front positions in all particles neighborhoods.
4. Update particle velocities according to (3.5-3.6).
5. Update particle positions according to (3.2).
6. Compute particles fitness values for the new positions.
7. Compare particles fitness values at time t with those at time t-1 according to non-dominance criterion. Update personal bests where needed.
8. Mutate particles particle according to ρ_m and (3.7 – 3.8).
9. Mutate all the solutions in the Pareto-front as in point 8.
10. Update Pareto-front by considering new particles bests and mutated Pareto-front positions.
11. If needed, remove exceeding non-dominated solutions according to their crowding distance.
12. If additional termination criteria are met, exit the loop.
13. Return to step 2

3.7.3. MOFIPS performances on benchmark tests

MOFIPS performances were assessed on the benchmark functions employed by Deb et al. (2002) to test the NSGA-II algorithm, which is arguably the most widely used MO Evolutionary Algorithm (MOEA). In their work, the authors compare the real and binary-coded version of the NSGA-II against two other common MOEAs, namely the Pareto-Archived Evolution Strategy (PAES) (Knowles and Corne, 1999) and the Strength Pareto-Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999). The comparison is carried out on a set of both unconstrained test problems for which the Pareto-optimal set is known. In particular, the MOFIPS algorithm was tested for the SCH, FON, ZDT1, ZDT2, ZDT3 and ZDT6 mathematical problems, which are reported

in APPENDIX A for reference. The performances were estimated using the same two performance measures employed in Deb et al. (2002), which are particularly effective in directly evaluating both the convergence to a known Pareto-optimal set, and the spread across the solutions returned by the algorithm. These two measures are the the convergence metric Υ, and the diversity metric Δ. A value of zero of Υ entails perfect convergence of the algorithm solutions to a chosen subset of points in the optimal Pareto-front. Accordingly, a zero value for the diversity metric Δ will identify that a set of solutions spans uniformly the entire Pareto-front, including the extremes. The reader is referred to the original NSGA-II paper for information on how these two metrics are actually computed. For fair comparison with the results reported in the study, the same maximum of 25,000 function evaluations is set as the termination criterion for the MOFIPS simulation runs. In Tables 3.1 and 3.2, the mean and variance of the two performance measures are reported for NSGA-II, SPEA and PAES as featured in the work of Deb et al., (2002). The statistics for 20 runs of a successful MOFIPS configuration are also displayed for comparison. The configuration represents a ring-structured swarm with 20 particles, two neighbors per particle, a maximum size of the Pareto-front M of 20 particles, with turbulence parameters ρ_m and η_m of 0.2 and 20, respectively. From the analysis of the results in the tables, it can be seen that MOFIPS compares extremely well against all the MOEAs. The MOFIPS consistently outperforms every other algorithm in terms of the diversity metric Δ for each of the benchmark problems. Significant improvements are not only recorded with respect to the less performing PAES and SPEA, but also with respect to both versions of the NSGA-II. Results are also very promising when the convergence metric Υ is considered.

The MOFIPS shows best convergence in half of the benchmarks problems (FON, ZDT3 and ZDT6), with PAES coming first once (SCH), and the binary version of the NSGA-II twice (ZDT1 and ZDT2). In addition, the MOFIPS appears to be the most balanced among all the algorithms, showing good performances for all benchmarks.

TABLE 3.1. MEAN (FIRST ROW) AND VARIANCE (SECOND ROW) OF THE CONVERGENCE METRIC Υ (IN BOLD THE BEST VALUE RECORDED FOR EACH TEST PROBLEM)

	SCH	FON	ZDT1	ZDT2	ZDT3	ZDT6
NSGA-II	0.003391	0.001931	0.033482	0.072391	0.114500	0.296564
	0.000000	0.000000	0.004750	0.031689	0.007940	0.013100
NSGA-II (binary)	0.002833	0.002571	**0.000894**	**0.000824**	0.043411	7.806798
	0.000001	0.000000	0.000000	0.000000	0.000042	0.001667
SPEA	0.003403	0.125692	0.001799	0.001339	0.047517	0.221138
	0.000000	0.000038	0.000001	0.000000	0.000047	0.000449
PAES	**0.001313**	0.151263	0.082085	1.126276	0.023872	0.085469
	0.000003	0.000905	0.008679	0.036877	0.000010	0.006664
MOFIPS	0.003167	**0.001055**	0.003840	0.002237	**0.003925**	**0.011751**
	0.000000	0.000000	0.000002	0.000001	0.000000	0.000818

TABLE 3.2. MEAN (FIRST ROW) AND VARIANCE (SECOND ROW) OF THE DIVERSITY METRIC Δ (IN BOLD THE BEST VALUE RECORDED FOR EACH TEST PROBLEM)

	SCH	FON	ZDT1	ZDT2	ZDT3	ZDT6
NSGA-II	0.477899	0.378065	0.390307	0.430776	0.738540	0.668025
	0.003471	0.000639	0.001876	0.004721	0.019706	0.009923
NSGA-II (binary)	0.449265	0.395131	0.463292	0.435112	0.575606	0.644477
	0.002062	0.001314	0.041622	0.024607	0.005078	0.035042
SPEA	1.021110	0.792352	0.784525	0.755148	0.672938	0.849389
	0.004372	0.005546	0.004440	0.004521	0.003587	0.002713
PAES	1.063288	1.162528	1.229794	1.165942	0.789920	1.153052
	0.002868	0.008945	0.004839	0.007682	0.001653	0.003916
MOFIPS	**0.254751**	**0.373575**	**0.382408**	**0.375675**	**0.498458**	**0.432655**
	0.003146	0.006032	0.004072	0.004741	0.002196	0.050052

3.7.4. The MBFIPS algorithm

A binary-coded variant of MO Swarm Optimization can be obtained by merging the MOFIPS algorithm with the BFIPS algorithm presented in Section 3.6. This is done

by employing the binary rule of the BFIPS instead of (3.2) for updating particle positions. In the same way, MOFIPS polynomial mutation is substituted with the bit-flipping mutation operator of the BFIPS. The generalization to the MO case is granted by adopting the mechanism of the MOFIPS for 1) selecting leading particles via the Pareto-based non-dominance criterion; 2) adding leaders to particles neighborhoods to form augmented topologies; and 3) discarding exceeding non-dominated solutions according to their crowding distance. These adaptations lead to the Multi-objective Binary-coded Fully Informed Particle Swarm (MBFIPS) optimization algorithm, which can be summarized as follows:

ALGORITHM V. Multi-objective Binary-code Fully Informed Particle Swarm (MBFIPS)

1. Initialize particle positions, best positions, velocities, fitness values, and starting Pareto-front.
2. Loop for t = 2 up to a maximum number of iterations.
3. Include Pareto-front positions in all particles neighborhoods.
4. Update particle velocities according to (3.5-3.6).
5. Constrain velocities with the logistic transformation.
6. Update particle positions based on particle velocities.
7. Compute particles fitness values for the new positions.
8. Compare particles fitness values at time t with those at time t-1 according to non-dominance criterion. Update personal bests where needed.
9. Select m particles and perform bit-flipping mutation with mutation rate μ.
10. Mutate all the solutions in the Pareto-front as in point 9.
11. Update Pareto-front by considering new particles bests and mutated Pareto-front positions.
12. If needed, remove exceeding non-dominated solutions according to their crowding distance.
13. If additional termination criteria are met, exit the loop.
14. Return to step 2

PART III. APPLICATIONS

4. Training NNRF models with MOFIPS

In this first application, swarm optimization algorithms are employed to develop ANN-based NNRF models with better generalization performances than those developed using gradient-based search algorithms. This is done by addressing the cross-validation scheme commonly employed for ANN training (see Section 2.1.5) as a true multi-objective problem. In particular, the calibration dataset is split into two subsets, and the MOFIPS algorithm is employed to tune ANN parameters by concurrently minimizing the residuals on both subsets. The results for the prediction of daily streamflow discharges of the Shenandoah River (US) show that MOFIPS-trained NNRF outperform those developed using standard PSO, as well as advanced gradient-based optimization techniques.

4.1. Introduction

Although extremely fast, even the most advanced local search methods such as the LM algorithm (see Section 2.1.4) are prone to being trapped in local minima, especially in complex problems with a rough error surface and many local optima. This is the main reason why GO algorithms have been tested as alternatives for the development of NNRF models (Maier et al., 2010). Although there exists many applications concerning the use of GA for developing NNRF solutions (Abrahart et al., 1999; Chen and Chang, 2009; Corzo and Solomatine, 2007a, 2007b; Jain and Srinivasulu, 2004; Sedki et al., 2009), only few examples regarding the use of swarm optimization are available in the literature, presenting results which are limited and somewhat contradictory. Chau (2007, 2006) employed PSO-trained SLFNs for real-time

prediction of the water stage in the Shing Mun River, Hong Kong. The results showed that tuning the ANN parameters using the PSO resulted in more accurate NNRF models than those obtained via standard back-propagation or the LM algorithm. In addition, the author proposed the combination of PSO and LM in a split step approach (Chau, 2007). This paradigm combines the advantages of global search capability of PSO algorithm in the first step, and local fast convergence of the LM algorithm in the second step. The mixed approach was able to attain a higher accuracy than the two algorithms on their own. Although these initial studies suggested that the PSO is able to perform better model parameters calibration compared to local search techniques, the opposite conclusions were drawn by Piotrowski and Napiorkowski (2011) for a case study involving streamflow discharge forecasting in the Annapolis River catchment, Nova Scotia (Canada). In their work, the authors show that LM training yielded more accurate FNN models in much less time with respect to several global optimization techniques, including two advanced variants of the PSO which are known to outperform the standard version of the algorithm on benchmark tests. The authors therefore strongly advocate for the use of local search techniques to develop NNRF models by means of differentiable objective functions, as long as a multi-start approach is implemented to reduce the chances of getting stuck in poor minima.

Due to the limited number of reported applications, this first study is an attempt to shed a light on the real effectiveness of swarm optimization techniques for calibrating the synaptic weights and biases of NNRF models. In particular, we focus on how generalization is ensured in PSO-trained neural networks, an aspect which has been overlooked so far, and that could possibly explain the poorer performances of swarm

optimization in some applications. In the aforementioned studies, the cross-validation approach was employed to prevent the PSO-trained NNRF models from overfitting and improve their generalization capability on a validation dataset (see Section 2.1.5). This scheme is implemented by allowing particle position updates only if fitness improvements are concurrently recorded on both the training and the validation dataset (Piotrowski and Napiorkowski, 2011), in similarity with early stopping. We argue that this approach is wrong in two respects. In the first place, it is entirely possible for a particle to move to a location characterized by better generalization while temporarily underperforming on the validation dataset. This can be easily seen by running the PSO to minimize only the training errors while recording the evolution of the validation performances at the same time, as done in Fig. 4.1 for a given particle in a trial experiment. The figure shows how the validation error decreases in the long run although it goes up several times during the optimization process. Therefore, it is likely that preventing those trajectories resulting in temporary deterioration of validation performances might severely hinder the optimization process, especially in the early stages. Most importantly, unlike stopped training, there is no effective difference in how the training and the validation datasets are used in PSO-ANN calibration. Indeed, since the acceptance rule for particle position update employs the objective functions defined on both datasets, the implementation of cross-validated training should be addressed as a multi-objective (MO) optimization problem. The main goal of this first application is thus to employ the MOFIPS algorithm to calibrate NNRF models by concurrently minimizing the error on the training and validation datasets according to the Pareto dominance criterion. The effectiveness of this approach is tested for the prediction of

future streamflow discharges in the North Fork of the Shenandoah River, Virginia (US). The MOFIPS-trained NNRF are compared with those developed using standard cross-validation in conjunction with five other algorithms, comprising the canonical PSO (see Section 3.3), as well as four advanced gradient-based optimization techniques. These are the Scaled Conjugate Gradient (SCG), the Conjugate Gradient with Fletcher-Reeves updates (CGF), the Conjugate Gradient with Polak-Ribiére updates (CGP), and the LM algorithm (Adamowski and Karapataki, 2010; Chen and Chang, 2009; Hamed et al., 2004).

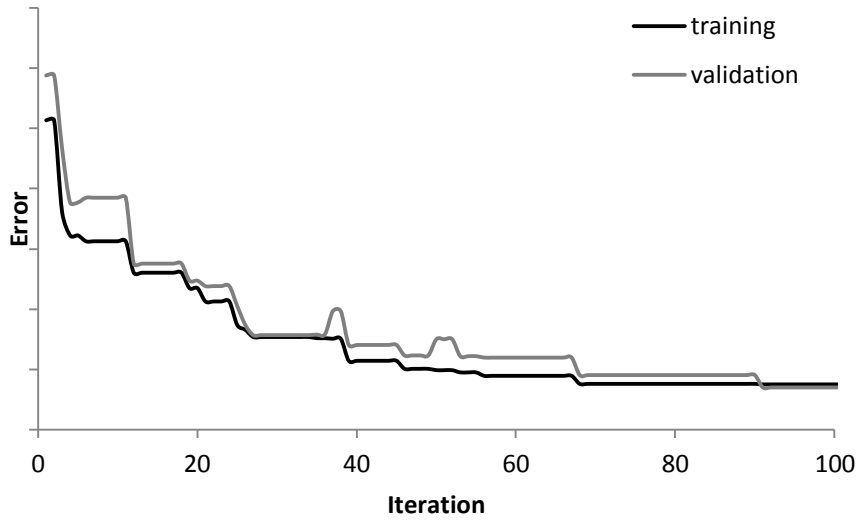


Figure 4.1. Training and validation errors in non-cross-validated PSO-ANN training.

4.2. Case Study

The Shenandoah River is a flood-prone river in Virginia, US, and is the principal tributary of the Potomac River. The Shenandoah River is originated by the confluence of two tributaries, the South Fork and the North Fork, which join their courses northeast of the city of Front Royal in Warren County (Fig. 4.2). In this work we are concerned

with 1-day ahead forecasting of river discharge in the North Fork of the Shenandoah river, a fifth order stream of 169 kilometers that drains an area of around 6930 square kilometers of north eastern Virginia. Daily river discharge observations are available from a gauging station in Strasburg, while daily precipitation data are collected from a meteorological station in Waterloo sited around 35 kilometers upstream from the gauging station. Around 9,000 observations of river discharge and rainfall were retrieved from the US Geological Survey database, ranging from May 1985 to the end of December 2009. A sample of the recorded times series is given in Fig. 4.3.

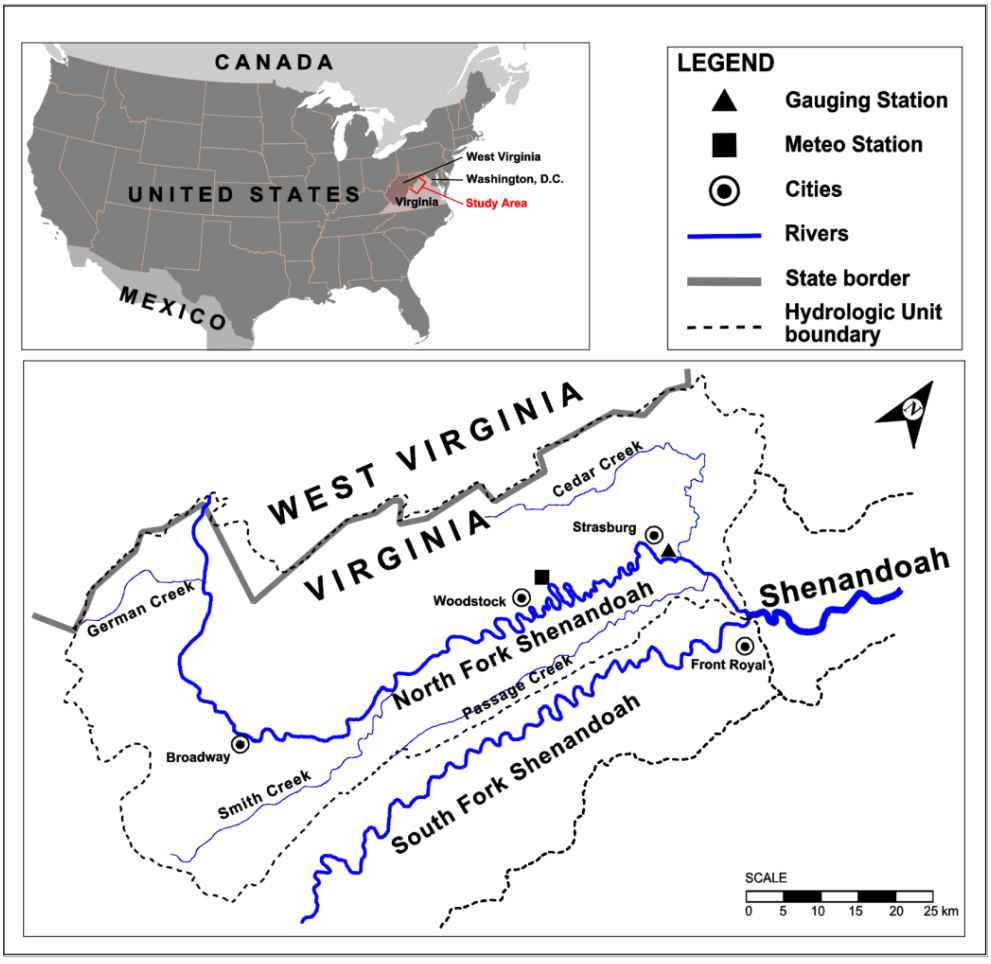


Figure 4.2. Location of the Shenandoah River.

4.3. Input and model selection

The original time series are initially pre-processed to remove outliers and form additional inputs formed by means of aggregating operators. In particular, 3-days and 7-days moving average of flow observations, as well as 3-days and 7-days cumulated precipitation are employed to form a total of 6 input variables. Lagged time series up to 3 days were considered so that the total set of inputs comprises a total of 18 potential candidates. The input and output variables are rescaled in the [-1,1] range to facilitate ANN training, and the dataset is then split into a training (40% of available dates), a validation (40% of available dates), and a test dataset (20% of available dates). A constructive forward selection (CFS) scheme (Maier et al., 2010; May et al., 2011) is then implemented to select the optimal number of hidden neurons of the SLFNs, as well as the optimal combination of inputs among all candidates. The CFS entails an incremental trial-and-error strategy where an initial ANN with minimal complexity is trained n times separately, each time having only one of the n candidate variables as the sole input. The most significant input is chosen according to an optimality criterion, and the search continues by looking for the next input among the remaining n - 1 candidates to add to the model. This procedure is repeated iteratively until the inclusion of further inputs does not yield any improvement of the optimality criterion. When this event occurs, the SLFN is first augmented with an additional neuron to its hidden layer, and the search for new inputs is resumed from where it had stopped. The search continues if adding a new input to the augmented model results in improved performances, otherwise the CFS process is terminated and the model obtained at the previous step is returned. To improve the reliability of the CFS scheme, the optimality criterion is

chosen as the median value of the validation Root Mean Square Error (RMSE, see Appendix B) obtained by training each SLFN model with 100 restarts. Due to the computational effort needed to perform the CFS, the SLFNs are trained using the LM method, which is the fastest training technique considered in this work. The results obtained for the LM are then extended to the cases where the other algorithms are employed. The optimal model returned by the CFS scheme has 6 hidden neurons and 5 input variables, namely the streamflow discharges up to three days ahead (t-1, t-2, t-3), the rainfall measured the previous day (t-1), and the 3-days cumulated rainfall computed at time t-3. The total number of weights (including ANN biases) in the optimal model is 43.

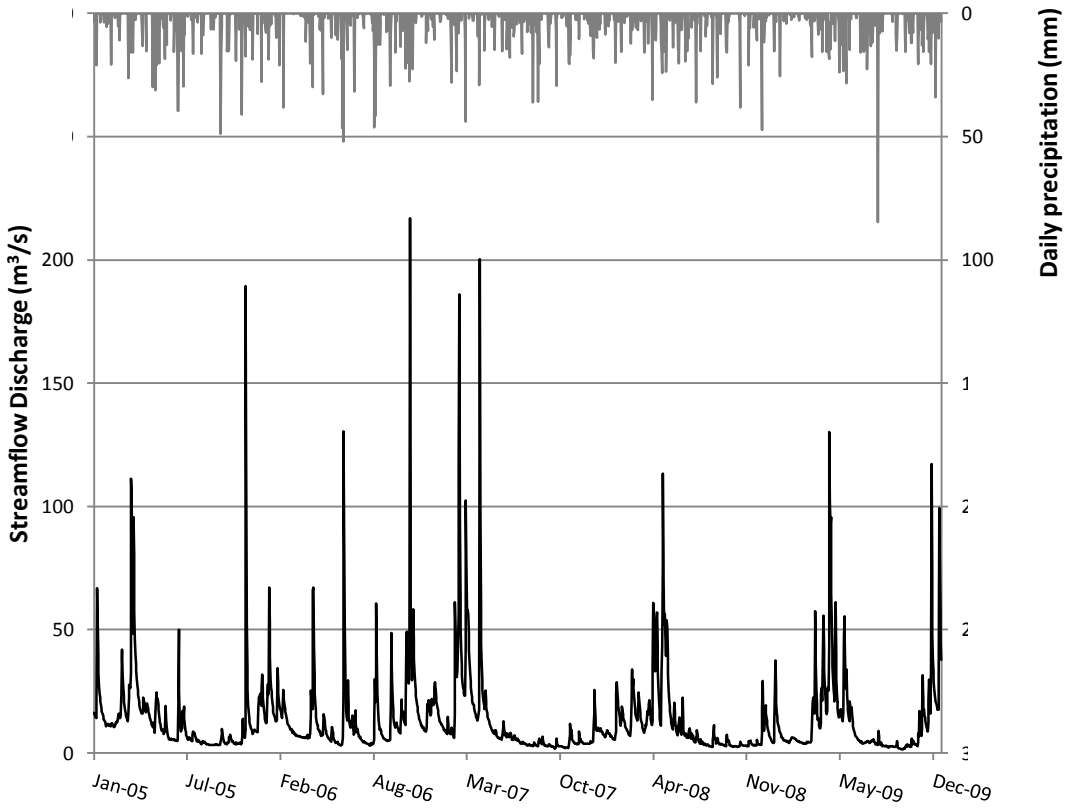


Figure 4.3. Sample of recorded total precipitation and streamflow discharge.

4.4. Comparison of PSO and MOFIPS performances

A first batch of experiments are run to check whether addressing cross-validated ANN training as a MO problem results in improved performances of the NNRF model. This is done by training the optimal model architecture returned by the CFS with both cross-validated single-objective canonical PSO, as well as with the MOFIPS algorithm.

4.4.1. Experimental setup

The comparison is carried out using four different particle arrangements in order to assess the impact of different topologies on the quality of the developed NNRF models. The employed topologies are made of 30 particles which are disposed to form 1) a sphere with 30 neighbors, 2) a ring of 30 particles with 2 neighbors each, 3) a clustered arrangement with 5 niches of 6 particles each, and 4) a 6-by-5 bi-dimensional lattice. The reader is referred to Fig. 3.4 a, b, e, and f for a graphical representation of these arrangements. The first topology was chosen since is the original one proposed for the PSO algorithm, while the remaining three are known to be valid alternatives when the FIPS paradigm is employed (Mendes et al., 2004). For the MOFIPS case, these topologies are tested with or without including each particle in its own neighborhood. These cases are respectively identified as MOFIPS-w and MOFIPS-wo for the remaining of the discussion. The maximum number *M* of Pareto-front particles and the parameter η_m controlling the turbulence probability distribution are set to 30, while the percentage of particle dimensions subjected to turbulence is set to $\rho_m = \frac{1}{number\ of\ weights} \cong 2.5\%$. Each PSO/MOFIPS case is run a total of 20 times for 1,000 iterations, and a distribution of the performances on each dataset is obtained by

extracting the best performing particle of each run. All the experiments are run using the Matlab® computing environment and programming language.

4.4.2. Selection of optimal solutions

The best performing solution for each PSO run is obtained by selecting the particle with the best validation performances. On the other hand, a strategy has to be developed to select a solution from the Pareto-front returned by each MOFIPS run. Each solution in the Pareto-front is by definition equally efficient with respect to the objectives, however the solution at extremes of the frontier are more likely to over-fit one of the two datasets. Although more sophisticated selection schemes could be implemented, in this work the optimal solution of each MOFIPS run was chosen as the one located at the knee of the Pareto-front (Branke et al., 2004), i.e. that solution along the frontier after which a small improvement in one objective leads to a large deterioration in the other (Fig. 4.4).

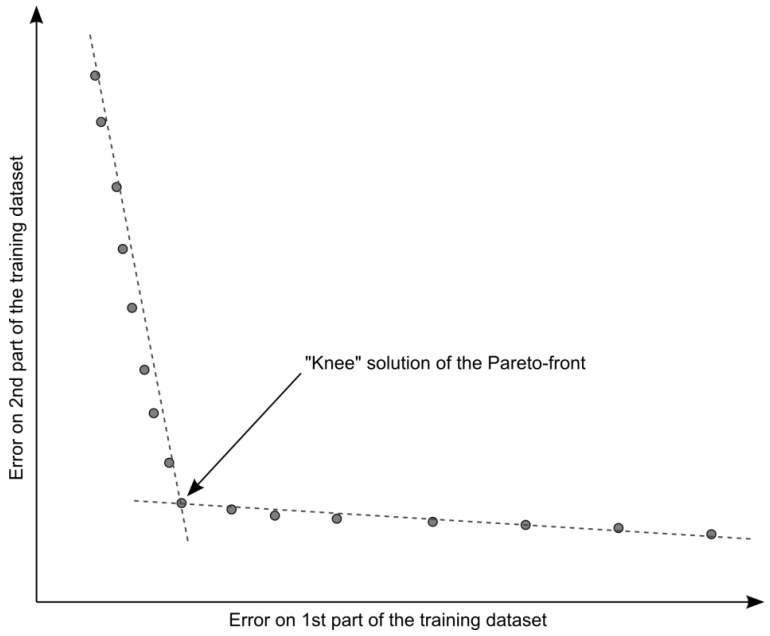


Figure 4.4. Selection of MOFIPS optimal solution.

4.4.3. Results and discussion

The statistical comparison of PSO and MOFIPS performances is reported in Table 4.1 for each employed topology in terms of the Nash-Sutcliffe Coefficient of Efficiency (CE, see Appendix B). Other goodness-of-fit measures were employed for the comparison, but these results were not reported since they do not provide further insights to the analysis. From a first glance at Table 4.1, it appears that the NNRF models trained with the MOFIPS algorithm clearly outperform than those obtained with standard PSO. With exception of one topology, the median values of the CE are around 5% higher on the training and validation datasets, and over 11% higher on the test dataset. These figures suggest that the multi-objective approach develops NNRF models with better generalization capability. The analysis of the results for each topology shows that the efficiency of both PSO and MOFIPS depends on the adopted swarm arrangement. In particular, the PSO seems to be more efficient when the ball topology is employed. This could be attributed to the higher convergence speed provided by this arrangement under the PSO framework with respect to the other topologies. On the other hand, the excessive flow of information among neighbors penalizes the ball topology for both the MOFIPS cases. These results are consistent with those obtained by comparing the PSO and FIPS algorithms on benchmark trials (Mendes et al., 2004), although the MOFIPS employs an alternative formulation of the fully informed velocity update (see Section 3.4.1). All the other candidate topologies are more or less successful, and there are no significant differences between the MOFIPS-w and MOFIPS-wo cases. However, the MOFIPS-wo lattice combination is arguably the one providing the best results overall, with the MOFIPS-wo clustered geometry coming a close second.

4.5. Comparison of MOFIPS and gradient-based algorithms performances

After assessing the superiority of the MOFIPS parading over canonical cross-validated PSO for NNRF development, a second batch of experiments is carried out to assess how the proposed MO approach compares against four of the most advanced local search algorithms, i.e. the SCG, CGF, CGP and LM.

4.5.1. Experimental setup

The four gradient-based algorithms are implemented in MATLAB® using the Neural Network Toolbox®. In addition, the setup of the parameters for each method is chosen based on the suggestions provided along with the employed computing suite. In each case, standard cross-validation (see Section 2.1.5) is employed for improving model generalization, while 100 restarts are used to prevent poor minima entrapment. The best NNRF models developed using each algorithm are selected from the 100 restarts as the ones with the lowest validation error.

4.5.2. Results and discussion

Table 4.2 presents the performances of the NNRF models for the prediction of the Shenandoah River streamflow discharges. The results are given in terms of CE, RMSE, and the ratio of the RMSE to the standard deviation (RMSE/STDEV). The second column of Table 4.2 reports the overall computational time needed by each

TABLE 4.1. PERFORMANCE COMPARISON OF PSO- AND MOFIPS-TRAINED NNRF MODELS

	TRAINING					VALIDATION					TEST				
	Coefficient of Efficiency					Coefficient of Efficiency					Coefficient of Efficiency				
	BEST	WORST	MEDIAN	MEAN	STDEV	BEST	WORST	MEDIAN	MEAN	STDEV	BEST	WORST	MEDIAN	MEAN	STDEV
PSO															
ball	0.791	0.666	0.757	0.75	0.033	0.788	0.663	0.777	0.767	0.032	0.722	0.432	0.663	0.649	0.068
ring	0.768	0.633	0.728	0.725	0.034	0.787	0.648	0.741	0.737	0.032	0.697	0.432	0.615	0.61	0.066
lattice	0.785	0.704	0.746	0.744	0.021	0.784	0.724	0.753	0.753	0.018	0.714	0.531	0.637	0.629	0.045
clustered	0.778	0.696	0.749	0.746	0.023	0.793	0.697	0.764	0.758	0.025	0.719	0.526	0.646	0.633	0.047
MOFIPS-w															
ball	0.771	-0.025	0.753	0.714	0.174	0.772	-0.001	0.757	0.719	0.17	0.678	-0.008	0.634	0.604	0.145
ring	0.805	0.754	0.784	0.783	0.012	0.792	0.765	0.782	0.781	0.008	0.741	0.662	0.704	0.704	0.021
lattice	0.802	-0.029	0.783	0.743	0.182	0.795	-0.001	0.782	0.743	0.175	0.734	-0.004	0.708	0.67	0.16
clustered	0.795	0.761	0.784	0.781	0.011	0.793	0.768	0.782	0.782	0.006	0.73	0.655	0.697	0.696	0.021
MOFIPS-wo															
ball	0.773	-0.022	0.753	0.716	0.174	0.779	-0.003	0.755	0.719	0.17	0.678	-0.004	0.637	0.607	0.146
ring	0.79	0.748	0.777	0.775	0.011	0.794	0.754	0.78	0.779	0.01	0.732	0.629	0.686	0.688	0.027
lattice	0.802	0.752	0.787	0.782	0.015	0.793	0.757	0.784	0.779	0.01	0.732	0.609	0.717	0.702	0.033
clustered	0.803	0.762	0.786	0.784	0.012	0.798	0.76	0.78	0.781	0.009	0.731	0.645	0.71	0.699	0.028

algorithm to perform all the restarts. It can be seen that the model obtained with the LM algorithm outperforms those obtained with the remaining local search methods for each dataset, except for the CGF-trained ANN that shows basically the same performances on the test dataset. Nonetheless, while the LM is the fastest algorithm, the CGF is the slowest and it needs a much longer computational time to develop a NNRF model with similar accuracy. The superiority of the LM over conjugate gradient methods was also reported in other studies (Adamowski and Karapataki, 2010; Hamed et al., 2004). The last row of Table 4.2 shows the performances of a MOFIPS run for comparison. The results pertain to a model obtained with the MOFIPS-wo lattice configuration, after the algorithm was left to run for 4,000 iterations to ensure convergence. The optimal MOFIPS solution was selected according to the scheme presented in Section 4.4.2. From the analysis of the results it emerges that the MOFIPS algorithm compares well against the gradient-based methods both in terms of overall run-time and model accuracy. Although slower than the LM, the MOFIPS is able to provide the NNRF with best generalization ability in one-third to one-half of the time required by the conjugate gradient algorithms to perform 100 restarts. The improvements provided by the MOFIPS on the test dataset range from a minimum of 2% for the LM to a maximum of 10% for the CGP, which is the worst performing algorithm. More insights on the relative NNRF performances can be obtained through a visual inspection of the reconstructed hydrographs, as done in Fig. 4.5 for a sample of the training dataset. It can be seen that the NNRF model obtained by the MOFIPS and the LM tend to better approximate the peaks in streamflow discharge due to rainfall. On the other hand, all the models seem to perform equally well in predicting the falling limbs of the hydrographs

after a storm event. The CGP-trained model seems to suffer from timing errors in predicting the streamflow peaks. This is more likely to happen when there is an excessive imbalance in the relative importance of past streamflow input features over rainfall ones, suggesting that 100 restarts were not sufficient for the CPG algorithm to escape poor minima.

TABLE 4.2. PERFORMANCE COMPARISON OF GRADIENT-BASED AND MOFIPS-TRAINED NNRF MODELS

TRAINING ALGORITHM	TIME (s)	RMSE (m^3/s)			RMSE/STDEV			CE		
		training	validation	test	training	validation	test	training	validation	test
LM	541	12.283	12.917	9.091	0.418	0.475	0.520	0.810	0.800	0.729
SCG	2344	12.919	13.26	9.316	0.440	0.488	0.533	0.790	0.789	0.716
CGP	2387	14.233	13.789	9.989	0.484	0.507	0.572	0.745	0.772	0.673
CGF	3075	12.865	13.138	9.145	0.438	0.483	0.523	0.792	0.793	0.726
MOFIPS	956	12.232	13.147	8.822	0.416	0.484	0.505	0.812	0.793	0.745

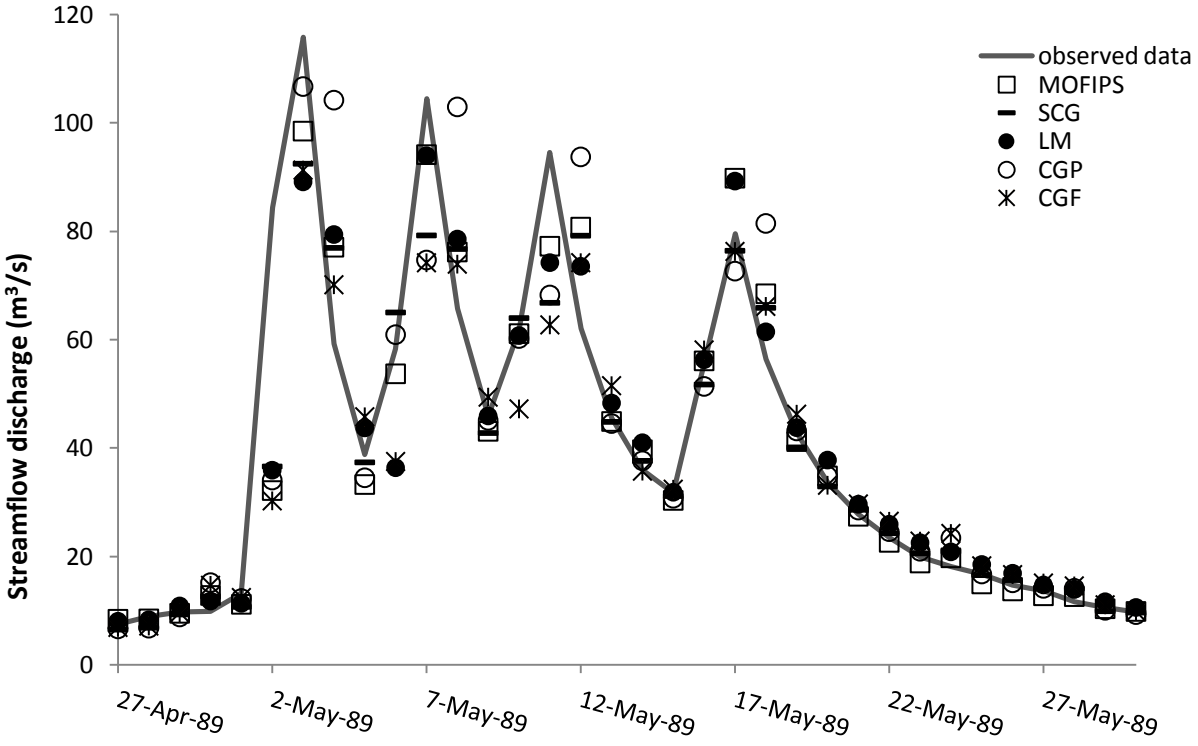


Figure 4.5. Comparison of model predictions.

4.6. Conclusions

In this study we proposed a multi-objective (MO) approach for cross-validated swarm optimization training of ANN to be used for streamflow forecasting purposes. The rationale justifying the use of this paradigm lies in the consideration that developing cross-validated NNRF models with single-objective PSO hinders the optimization process performed by the swarm by penalizing the exploratory behavior of the algorithm. In addition, the specification of two different objective functions in the acceptance rule for particle position update renders the optimization problem intrinsically multi-objective, thus it should be treated as such. Indeed, the experiments run for Shenandoah River watershed demonstrated that the MO approach implemented using the MOFIPS algorithm provides more accurate NNRF models with respect to canonical PSO. In addition, the NNRF model produced by the MOFIPS algorithm is found to outperform those built with the SCG, CGF, CGP and LM methods. With the exclusion of the LM, such improvements are obtained with a significant reduction in the computational costs, especially in comparison to the slower methods of the conjugate gradient family. Although further research is needed to thoroughly assess the effectiveness of the proposed MO training scheme for NNRF development, the findings of this work are certainly encouraging with respect to some results remarking the inferiority of single-objective swarm optimization (Piotrowski and Napiorkowski, 2011).

5. NNRF interval forecasting using the Lower Upper Bound Estimation (LUBE) method and MOFIPS

In the previous Chapter it was shown how the MOFIPS algorithm can be employed to improve the performances of NNRF models developed for deterministic predictions. This chapter demonstrates how MOFIPS can be used to facilitate the estimation of accurate ANN-based Prediction Intervals (PIs), a major issue of NNRF models which limits their use for operational streamflow forecasting. In particular, the MOFIPS algorithm is used in conjunction with the Lower Upper Bound Estimation (LUBE) method, a recent technique that is found to outperform traditional methods for ANN-based PI estimation. The LUBE method construct ANNs with two output neurons that directly approximate the lower and upper bounds of the PIs. The training is performed by minimizing a Coverage Width-based Criterion (CWC), which is a compound, highly nonlinear and discontinuous function. In this work, we test the suitability of the MOFIPS-based LUBE approach in producing PIs at different confidence levels (CL) for the 6-hours ahead streamflow discharges of the Susquehanna and Nehalem Rivers, US. Due to the success of single-objective swarm optimization in other LUBE engineering applications, the PSO and FIPS algorithms are employed for comparison.

5.1. Introduction

Despite the number of successful applications reported in the scientific literature, NNRF solutions still struggle to move from research grade to operational grade level due to a number of unresolved issues that still need to be addressed. A major obstacle is

represented by the difficulties in estimating the uncertainty of the predicted values produced by a NNRF model (Maier and Dandy, 2000). Indeed, the vast majority of NNRF applications so far have been only concerned with providing deterministic forecasts of the modeled hydrological variable, without estimating the degree of confidence associated with them. This is peculiar since hydrological forecasts can be employed for water resource management and natural hazards prevention only if a measure of their reliability is attached to each predicted value (Krzysztofowicz, 2001). The uncertainty characterizing NNRF point forecasts can be overcome by resorting to interval forecasts, or prediction intervals (PIs). A PI is a range of values in which the realization of a predicted random variable is expected to fall with a predefined coverage probability, known as the confidence level (CL). The width of the interval conveys information regarding the uncertainty of the forecast, so that for a given coverage probability, narrower widths entail higher accuracy. PIs are similar to confidence intervals (CIs), with the distinction that CIs are associated to the uncertainty in the prediction of an unknown but fixed value, whereas PIs are assigned to a random variable yet to be observed (De Gooijer and Hyndman, 2006). Since they also account for model misspecification and noise variance, by definition PIs enclose CIs of corresponding CLs. Although deterministic forecasts dominate the field of NNRF, there have been a few noteworthy applications of PI-based streamflow forecasting. The preferred methods involve the use of Bayesian Neural Networks (BNNs), resampling and ensemble techniques, as well as experiments that involve fuzzy theory. BNNs (Khan and Coulibaly, 2006; Kingston et al., 2005; Zhang et al., 2009) are able to return error bars along with their predictions, and have strong probabilistic theory backing them up.

However, they require the computation of the Hessian matrix at each iteration, which in turn causes singularity problems that may harm the quality of the PIs as well as heavy computational costs. Bootstrapping and ensemble modeling (Dawson et al., 2002; Sharma and Tiwari, 2009; Tiwari and Chatterjee, 2010) are also very time consuming as the number of ANN models to be trained has to be large in order to avoid biased estimates of total error variance. Computational costs can be abated with the use of fuzzy-based techniques such as the Local Uncertainty Estimation Model (LUEM) proposed by Shrestha and Solomatine (2006), which is based on fuzzy c-means clustering; or the approach of Alvisi and Franchini (2011) that employs fuzzy ANNs. Despite their differences, methods used in hydrology to estimate of ANN-based PIs usually require complex implementations that may have prevented their widespread application. Therefore, it is likely that resorting to less complicated, faster and yet effective techniques may favor a shift from deterministic NNRF models to more reliable solutions based on prediction intervals. Most importantly, all the aforementioned techniques share a common methodological weakness since they build the PIs indirectly from deterministic point predictions. Indeed, it would be more appropriate to generate the PIs directly through a mechanism that that considers both coverage probability and interval width criteria. This is the main premise that lead to the development of the Lower Upper Bound Estimation (LUBE) method proposed by Khosravi et al. (2011) to generate ANN-based PIs. This technique was found to outperform classic PIs construction techniques such as the delta method, Bayesian methods, and bootstrapping on both synthetic experiments as well as real world regression problems (Khosravi et al., 2011a; Quan et al., 2014a, 2014b, 2014c). The LUBE method constructs an ANN with

two output neurons that directly approximate the lower and upper bounds of the PIs. The training is performed by minimizing a coverage width-based criterion (CWC), which is a compound PI-based objective function accounting for both coverage probability and interval width. The CWC is a highly nonlinear and discontinuous function that requires global optimization techniques for its minimization. In particular, the PSO algorithm has proven very efficient in generating high quality PIs (Quan et al., 2014a). The main objective of this study is to test the suitability of PSO-based LUBE as a straightforward approach to predict streamflow discharges with uncertainty. This is done by producing 90%, 95% and 99% CL PIs for the 6 hours ahead streamflow discharge of the Susquehanna and Nehalem rivers, US. In addition, this work will assess whether PSO-based LUBE can benefit from a multi-objective formulation, as done here using the MOFIPS algorithm. Since. as shown in Chapter 4. MOFIPS-trained NNRF models can substantially outperform those developed using single-objective swarm optimization, similar improvements could be expected for NNRF models producing interval based predictions. This hypothesis is assessed by comparing the performances of MOFIPS-based LUBE NNRF-models against those obtained with single-objective PSO and single-objective FIPS optimization.

The remaining of this chapter is structured as follows: Section 5.2 will discuss the LUBE method and how it is employed in conjunction with swarm-optimization techniques. Section 5.3 will introduce the case studies and the datasets employed for the application. Section 5.4 will discuss the results of the experiments and present a thorough comparison of the LUBE PIs obtained with each of the considered swarm optimization techniques. Conclusions are given at the end of the chapter.

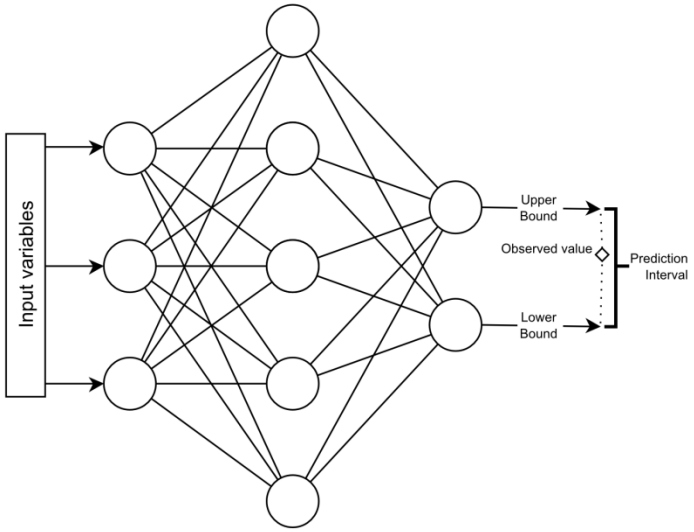


Figure 5.1. LUBE Neural Network model.

5.2. Estimation of ANN-based PIs with LUBE and swarm optimization

5.2.1. Prediction intervals

Prediction intervals (PIs) are random intervals constructed from historical observations that enclose a future observation within a certain range with a given probability, known as the confidence level (CL). If l and u are the lower and upper bounds delimiting the PI, and $(1-\alpha)\%$ is the CL attached to it, for a future unknown observation y_{t+1} of the predicted variable Y we can write $PI = [l, u]$ such that $\Pr(l < y_{t+1} < u) = 1 - \alpha$. Unlike deterministic forecasts, PIs carry information about the accuracy of the prediction, a fundamental requirement for planning, risk assessment and decision making. For valid PIs at a given confidence level, narrower intervals should of course be preferred since they entail less uncertainty and convey more information. Despite the superiority of PIs over point predictions, the latter constitute by far the most

common approach employed when forecasting water resource variables, especially when data-driven methods are chosen as the modeling tool for the hydrological process.

5.2.2. The Lower Upper Bound Estimation method and PI evaluation indices

The Lower Upper Bound Estimation (LUBE) method is a straightforward and efficient technique to produce high quality PIs for ANNs. The LUBE was found to outperform classic PIs construction techniques such as the delta method, Bayesian methods, and bootstrapping on both synthetic experiments as well as real world regression problems (Khosravi et al., 2011a, 2011b; Quan et al., 2014a, 2014b, 2014c). The LUBE method constructs an ANN with two output neurons that directly approximate the lower and upper bounds of the PIs, as depicted in Fig. 5.1. Each actual value of the predicted variable is enclosed in the interval between the two ANN outputs with $(1-\alpha)\%$ probability. This ANN is trained using historical observations of both the predicted variable and a set of relevant inputs. The training is performed by minimizing a Coverage Width-based Criterion (CWC), which is a compound PI-based objective function that accounts for both coverage probability and interval width (Khosravi et al., 2011a). The CWC is defined as a combination of two indices, namely the Prediction Interval Coverage Probability (PICP) (Khosravi et al., 2010) and the Prediction Interval Normalized Root-mean-square Width (PINRW) (Quan et al., 2014a). The value of these indices, including the CWC, can also be expressed in terms of percentages. PICP measures the percentage of target observations which are enclosed in the intervals, and is defined as follows:

$$PICP = \frac{1}{n}\sum_{i=1}^{n} c_i \tag{5.1}$$

where n is the number of observations, c_i is equal to one if the observation $y_i \in [l_i, u_i]$, and zero otherwise. The interval $[l_i, u_i]$ is the range of values delimited by the two outputs produced by the ANN (Fig. 5.1). PINRW provides an estimate of the overall PI width, and is defined as the ratio of the 2-norm of the width of the PIs to the range *R* of the observed variable:

$$PINRW = \frac{1}{R}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(u_i - l_i)^2}\ . \tag{5.2}$$

The CWC cost function is a nonlinear combination of (5.1) and (5.2) defined as (Quan et al., 2014a):

$$CWC = PINRW\left(1 + \gamma(PICP)e^{-\eta(PICP-\mu)}\right) \tag{5.3}$$

where $\gamma(PICP)$ is set equal to 1 during model calibration, while it becomes a step function of PIPC on test dataset

$$\gamma(PICP) = \begin{cases} 0, & if\ PICP \geq \mu\ ; \\ 1, & if\ PICP < \mu\ . \end{cases} \tag{5.4}$$

The value of $\gamma(PICP)$ is forced to 1 during training in order to allow for the construction of more conservative PIs and reduce the risk of violating the CL constraints during test (Khosravi et al., 2011b). The parameters η and μ are two constants used to define the penalty term controlling the balance between coverage probability and width of the developed PIs. The penalty term is needed to synthesize these two conflicting objectives in a single-objective cost function. In particular, the CWC is designed so that the optimization process will first search for valid PIs for which $PICP \geq (1 - \alpha)$ holds;

then the search is refined by gradually giving more importance to the PINRW term in (5.3), so that narrower PIs are constructed. To ensure the calibration is carried out in this way μ is set to $(1-\alpha)$, while literature suggests 80 as an appropriate value for η (Quan et al., 2014a). For a simpler assessment of the width of the developed PIs, the Prediction Interval Normalized Average Width (PINAW) is usually employed instead of PINRW. PINAW is defined as the ratio of the average width of the prediction intervals to the range *R* of the observed variable:

$$PINAW = \frac{1}{nR}\sum_{i=1}^{n}(u_i - l_i)\ .$$

5.2.3. PSO-based and FIPS-based LUBE for constructing streamflow PIs

In the following section the major steps regarding the construction PSO-based LUBE streamflow forecasting models are be discussed according to the general version of the method proposed by Quan et al. (2014b). Although the following steps refer to the PSO algorithm, they are implemented likewise when the FIPS is employed.

Dataset creation

Provided the set of hydrological and meteorological inputs has been already identified, all the input-output patterns are first normalized in the [-0.9,+0.9] range to facilitate model training and avoid saturation of the ANN activation functions. The whole dataset is then divided to form training and a test datasets.

Search for optimal PSO-LUBE model structure

The training dataset is further divided into k subsets to perform k-fold cross-validation (see Section 2.1.5). This procedure is carried out in order to find an optimal ANN structure that will prevent over-fitting and maximize performances. ANN models of increasing complexity, i.e. increasing size of the hidden layer, are trained k times using at each repetition a different subset for validation and the remaining k-1 subsets for model training. The median of the k CWC indices computed for the validation dataset is later used to select the optimal LUBE model.

Training with swarm optimization

The PSO method is used to train an ANN model for each considered model structure and for each repetition entailed by the k-fold cross-validation procedure. The algorithm is first initialized by setting the parameters governing its search, selecting the number of particles in the swarm, and choosing the swarm topology. Particle position and velocity arrays are also initialized before the optimization is started. Each position occupied by the particles during the search process represents a different configuration of synaptic weights and biases of the LUBE ANN model. Particle fitness values are computed based on the CWC of the corresponding LUBE ANN model on the training dataset (CWC_{TRAIN}). The PSO looks for the position in the search space for which CWC_{TRAIN} of the corresponding ANN configuration is the overall lowest.

Gaussian mutation operator

Due to the complexity of the CWC cost function, it is beneficial to include a mutation operator to boost PSO search and facilitate local minima escape. In PSO-based

LUBE, Gaussian mutation is performed after each particle has moved to a new position (Higashi and Iba, 2003; Quan et al., 2014a). If X_{ij} identifies the *j*-th component of the *i*-th particle position array at time *t*, its new value after mutation will be

$$X_{ij} \leftarrow X_{ij} + N\left(0, \sigma\left(X_{ij}\right)\right)$$

where $N\left(0, \sigma\left(X_{ij}\right)\right)$ is a random number from a Gaussian distribution with zero mean and a standard deviation which in the original PSO-based LUBE is set to 10% of the absolute value of X_{ij}. The effects of the mutation operator are also set to decrease exponentially with time.

Training termination and model evaluation

The PSO training is stopped when a maximum number of iterations has been reached, or when the algorithm stalls for a preset number of iterations. When the algorithm exits, a LUBE ANN model is built from the particle with the highest fitness, then the PIs of this model are estimated for the validation dataset and the relative value of CWC (CWC_{VAL}) is computed. Once the CWC_{VAL} for all the k subsets of the cross-validation procedure have been obtained, the median value is computed and stored for comparison with those of other model structures.

Selection of optimal model structure and PIs construction on the test dataset

When the cross-validated training procedure has been performed for all the considered model structures, the optimal model structure is selected as the one with the lowest median CWC_{VAL}. The k trained instances of the optimal structure are then used as

an ensemble to build the final PIs for the test dataset by averaging their lower bounds and the upper bounds outputs.

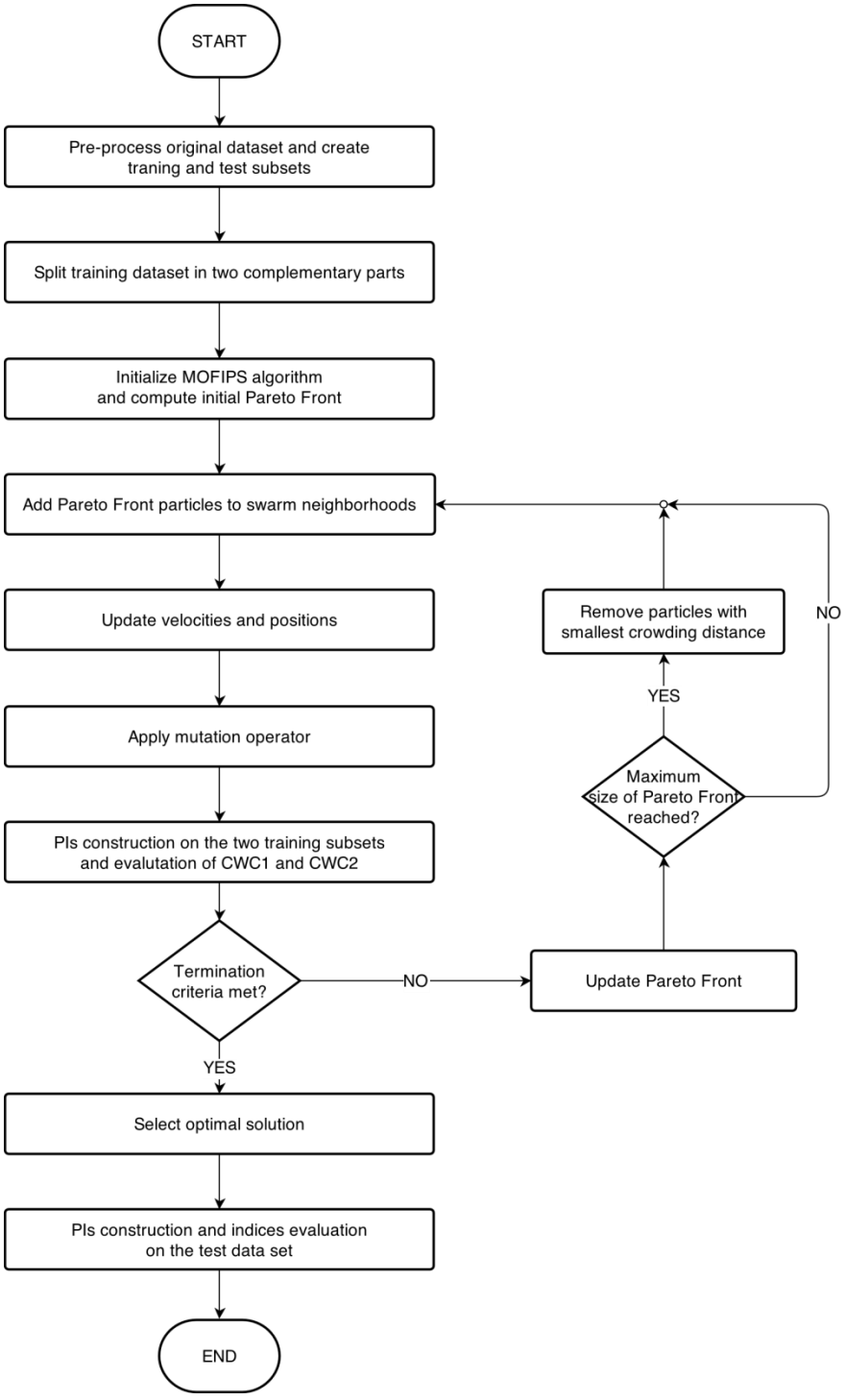


Figure 5.2. Flowchart of the MOFIPS-based LUBE method.

5.2.4. The MOFIPS-based LUBE method

A LUBE technique based on MOFIPS is devised in similarity with the bi-objective procedure employed for deterministic NNRFs presented in Chapter 4. The flowchart in in Fig. 5.2 shows how MOFIPS-based LUBE is implemented, with reference to ALGORITHM IV presented in Chapter 3. In MOFIPS-based LUBE the dataset is first split to form separate training and a test datasets. The training dataset is in turn divided in two complementary parts of similar length to create the two subsets of the bi-objective problem. The MOFIPS algorithm is initialized by selecting its parameters, defining the topology to use, and assigning random starting values to particle positions and velocities. Initial PIs are estimated after using the starting particles positions to build the LUBE models. The CWC values are then computed and the starting Pareto-front is generated. If CWC_1 and CWC_2 denote the values of CWC for the two training subsets, an example of the MOFIPS Pareto-front at a given iteration can be illustrated as done in Fig. 5.3. After the initialization, the MOFIPS starts its iterative process to search for better LUBE solutions. According to the MOFIPS paradigm, the non-dominated positions of the Pareto-front are included in particles neighborhoods, and particle velocities are computed based on this augmented set of neighbors. Particles then fly to new positions according to their velocities, and the mutation operator is applied after they have landed. The Pareto-front is then updated and if the updated frontier is larger than a predefined maximum size, the set of solutions is trimmed down to this maximum size by discarding the solutions with lowest crowding distance. The iterative process is terminated either when a maximum number of iterations has been reached, or

when a desired fitness value has been obtained for both objective functions, or when the Pareto-front has not been updated for a given number of consecutive iterations.

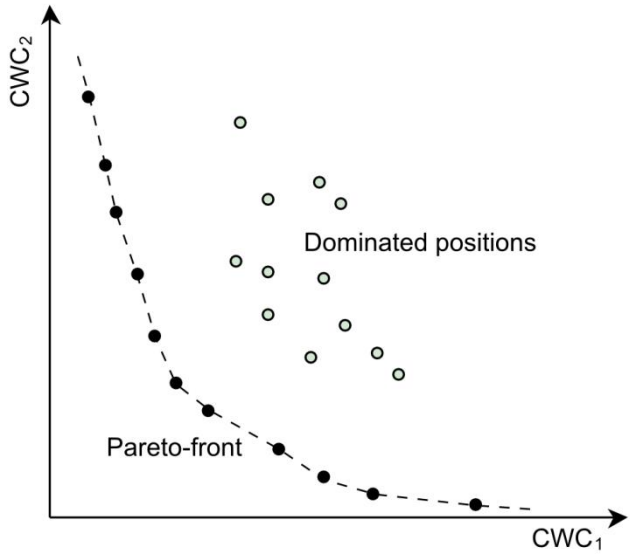


Figure 5.3. Example of MOFIPS Pareto-front for LUBE model development.

5.2.5. Selecting optimal MOFIPS-based LUBE solutions

After the optimization process has been completed, an optimal solution has to be extracted from the Pareto-front and employed to build streamflow PIs on the test dataset. Theoretically, all the non-dominated solutions forming the final Pareto-front are equally good with respect to the chosen CWC objective functions, meaning that none of them outperforms the others on both parts of the training dataset (see Fig. 5.3). However, practical optimal solutions can be identified based on the final goal of the application that is obtaining high quality PIs for the test dataset. A first requirement is that the selected solutions do not overfit either part of the training dataset. This can be reasonably ensured by considering only those solutions which provide valid PIs on both training subsets, thus respecting the constraint $PICP \geq (1 - \alpha)$ for a given confidence

level. On top of this, two different criteria based on the PICP and PINRW indices are proposed to select respectively a Most Precautionary (MP) and a Narrowest Interval (NI) solution. The MP solution is the point on the CWC_1 vs CWC_2 Pareto-front characterized by highest PICP on the training datasets with respect to the medians. This solution is most precautionary in the sense that having the largest coverage probability will increase the odds of future unknown streamflow observations to fall within the PIs built by the LUBE model (Khosravi et al., 2011b). We therefore expect the MP solution to have higher chances of producing valid, although wider, PIs for the test dataset. Given a Pareto-front made of m solutions, if $PICP_{1,i}$ and $PICP_{2,i}$ identify the PICP values of the i-th solution on the two training subsets, the MP solution is chosen as

$$MP = argmax\left(\left(PICP_{1,i} - Median(PICP_1)\right) + \left(PICP_{2,i} - Median(PICP_2)\right)\right) \quad (5.5)$$

where $i = 1,2,\dots,m$. The $argmax$ operator indicates that the MP solution is the member of the Pareto-front maximizing the sum of the differences between the PICP values in the two sets and the respective median values computed from all the points in the frontier. On the other hand, the NI solution is the one generating valid PIs on the training datasets having narrowest widths with respect to the medians. Similarly to (5.5), the NI selecting criterion can be written as

$$NI = argmin\left(\left(PINRW_{1,i} - Median(PINRW_1)\right) + \left(PINRW_{2,i} - Median(PINRW_2)\right)\right) \quad (5.6)$$

where $PINRW_{1,i}$ and $PINRW_{2,i}$ are the PINRW values of the i-th solution on the two training subsets. This time the $argmin$ operator indicates that the solution is the point minimizing the sum of the differences between the PINRW values in the two sets and the respective median values computed from all the points in the frontier. Compared to

the MP solution, the NI solution should likely generate narrower PIs for the test dataset although they might not satisfy the validity condition for the given confidence level. It is important to note that the differences in (5.5) and (5.6) should be normalized with respect to the respective medians if the distributions of the indices vary substantially between the two subsets.

5.3. Case studies

5.3.1. The Susquehanna River

At around 750 km long and with a watershed of over 70,000 km^2, the Susquehanna River is one of the longest rivers in the United States, draining southern New York State (NY), half of Pennsylvania State (PA) and emptying into the Chesapeake Bay in Maryland. For this case study, streamflow discharges PIs with a lead time of 6 hours are produced for the US Geological Survey (USGS) gauging station in Meshoppen, PA, which monitors a catchment area of 22585 km^2. The PIs at Meshoppen are developed using a set of input variables observed at 5 other stations located elsewhere in the area. In particular, hourly rainfall near the cities of Milan, Towanda, Dushore and Montrose, as well as previous hourly streamflow discharges in Towanda are employed. Details regarding these stations are reported in Table 5.1 for reference. Five additional aggregated time series were added to the working dataset by including 6-hours cumulated precipitation (SUM6) at each rainfall station as well as 6-hours moving average streamflow discharges (AVG6) at Towanda. Lag times from a minimum of 6 to a maximum of 11 hours were considered for each raw and aggregated time series. In other words, if *t* indicates the actual time of the prediction, lagged time series from *t*-6

up to *t*-11 are employed as inputs, providing a forecasting lead time of 6 hours. The working dataset for the Susquehanna River thus contains 60 inputs, and after removal of invalid observations, a total of 26555 input/output patterns spanning from January 2004 to April 2008.

TABLE 5.1. DETAILS OF GAUGING AND METEOROLOGICAL STATIONS

Dataset	*Name*	*Observed variable*	*WGS84 Coordinates*		*Distance (km)*
			Latitude	Longitude	
Susquehanna	Meshoppen	Flow discharge	41.61	-76.05	---
	Towanda	Flow discharge	41.77	-76.44	35
	Towanda	Rainfall	41.75	-76.42	35
	Milan	Rainfall	41.93	-76.52	53
	Dushore	Rainfall	41.53	-76.4	31
	Montrose	Rainfall	41.83	-75.87	29
Nehalem	Foss	Flow discharge	45.70	-123.75	---
	Nehalem	Rainfall	45.71	-123.9	53
	Jewell	Rainfall	45.94	-123.53	31
	Vernonia	Rainfall	45.87	-123.19	29

5.3.2. The Nehalem River

The Nehalem River originates in the Northern Oregon Coast Range near the city of Portland, and it ends its 192 Km course in the Pacific Ocean. For this river, PIs with a forecasting lead time of 6 hours ahead are produced for the USGS gauging station near Foss in Tillamook County, OR. The drainage area of this station is of 1728 km^2, which amounts to around 80% of the overall watershed area of 2210 km^2. The input time series for this case study were chosen as the previous streamflow values measured at Foss, plus the hourly rainfall recorded in the meteorological stations of Nehalem, Jewell Wildlife Meadows, and Vernonia (Table 5.1). As done for the first dataset, AVG6 and SUM6 aggregated time series were computed for the streamflow and rainfall inputs,

respectively, and lag times of 6 to 11 hours were considered. The Nehalem River dataset consists of 48 inputs, with 20577 input/output samples recorded between October 2007 and December 2013.

5.4. Results and discussion

5.4.1. Input selection

An optimal set of input features has to be selected from the available candidates of both datasets described in the previous section. In lack of a standard methodology for input selection to develop ANN-based PIs, we resort to the Constructive Forward Selection (CFS) wrapper technique devised for deterministic NNRF models (see Chapter 4). In particular, the CFS is used to determine an optimal set of inputs for NNRF models performing 6 hours ahead point predictions of streamflow discharges for both case studies. These input features are then used to develop LUBE models under the hypothesis that they represent a good approximation of the optimal input set required to develop ANN-based PIs. This assumption is legitimate if one considers that good quality PIs of hydrological variables have been obtained by bootstrapping and ensembling deterministic NNRF models (Dawson et al., 2002; Sharma and Tiwari, 2009; Tiwari and Chatterjee, 2010). Before launching the CFS algorithm, the datasets are first normalized in the [-0.9,0.9] range, and divided in training (40%), validation (40%) and test (20%) subsets as shown in Table 5.2. The Levenberg-Marquardt algorithm with early stopping was employed as the ANN training algorithm, using 50 restarts to prevent local minima entrapment. The CFS method returned an optimal model with 5 hidden neurons and 7 inputs for the Susquehanna dataset, while the best ANN for the Nehalem

River was found to have 3 hidden neurons and 5 inputs. The details of the selected inputs for both case studies are reported in Table 5.3, along with the performances on each dataset expressed in terms of Root Mean Square Error (RMSE) and Nash-Sutcliffe Coefficient f Efficiency (CE) (see APPENDIX B). The high performances shown by the CFS optimal architecture suggest that the selected features are suitable for modeling future streamflow discharges. They will be employed for LUBE model development following the premise made earlier in this section.

TABLE 5.2. DATASETS SUBDIVISION

Dataset	*Subset*	*Intial datetime [mm/dd/yyyy HH:MM]*	*Ending datetime [mm/dd/yyyy HH:MM]*	*Number of observations*	*Streamflow statistics [m3s-1]*		
					Min	*Max*	*Mean*
Susquehanna	Training	01/01/2004 11:00	11/23/2005 02:00	10653	42	5324	604
	Validation	11/23/2005 14:00	04/21/2007 06:00	10582	78	4616	585
	Test	04/21/2007 18:00	05/01/2008 00:00	5320	135	3313	670
Nehalem	Training	10/01/2007 17:00	03/22/2010 18:00	8230	2	1481	108
	Validation	03/22/2010 19:00	06/19/2012 08:00	8231	4	784	121
	Test	06/19/2012 09:00	12/02/2013 17:00	4116	3	580	108

TABLE 5.3. DETERMINISTIC ANN INPUTS AND MODEL PERFORMANCES

SELECTED INPUTS				MODEL PEFORMANCES		
Dataset	*Station*	*Input type*	*Lag*	*Subset*	*RMSE [m3s-1]*	*CE*
Susquehanna	Towanda	Flow RAW	t-6, t-7	Training	50.7	*0.994*
	Towanda	Flow AVG6	t-8, t-11	Validation	69.7	0.985
	Montrose	Rainfall RAW	t-6	Test	70.3	0.983
	Montrose	Rainfall SUM6	t-8, t-9			
Nehalem	Foss	Flow RAW	t-6, t-7	Training	11.1	*0.993*
	Vernonia	Rainfall RAW	t-6	Validation	12.5	0.99
	Jewell	Rainfall RAW	t-6	Test	13.8	0.985
	Jewell	Rainfall SUM6	t-6			

5.4.2. Development of swarm optimization-based LUBE models

After having identified the model inputs, LUBE neural networks can be developed to generate PIs for the 6-hours ahead streamflow discharge for both rivers. For the PSO and FIPS-based LUBE a 10-fold cross-validation procedure is carried out after joining the training and validation subsets in Table 5.2, and dividing the resulting dataset in 10 equal chunks. The two original subsets are left separated when employing the MOFIPS algorithm, where they are regarded as Part 1 and Part 2 of the overall training dataset used for the optimization. PIs with CL of 90%, 95% and 99% are considered for LUBE model development, therefore the values of μ in (5.3) was set to 0.9, 0.95 and 0.99 respectively. The value of η was set to 80 for each confidence level. In order to reduce the computational burden of the experiments, the search for optimal model complexity and swarm topology was performed only for the 90% case, with the results being extended to the other two cases. The search for optimal model structure was carried out trying LUBE models with 3 to 10 hidden neurons for each algorithm employed. For the Susquehanna River the best performances were obtained using ANNs with 6 hidden neurons (62 model parameters), irrespective of the training algorithm. On the other hand, all algorithms provided best results when 4 hidden neurons (34 model parameters) were used for the Nehalem River dataset. For the sake of brevity, full details on algorithms setup are reported in Table 5.4, along with the appropriate references for their implementation. Contrary to PSO- and FIPS-based LUBE, which require 10 runs to implement the *k*-fold cross-validation, MOFIPS is able to produce PIs at the end of a single run. Due to the similarity in the workings of the three algorithms, MOFIPS might therefore require only 1/10 of the computational time of its single-objectives counterparts. However, since MOFIPS entails augmented swarm topologies

and the additional calculation of the Pareto-fronts, this speed-up could be partially reduced. After these considerations, 5 restarts are performed for the MOFIPS technique for fairer comparison. The overall Pareto-frontier obtained from these restarts is considered for the extraction of optimal LUBE solutions according to the MP and NI criteria reported in Section 5.2.5.

TABLE 5.4. DETAILS OF THE SWARM OPTIMIZATION ALGORITHMS EMPLOYED

	PSO	FIPS	MOFIPS
Algorithm formulation	PSO Type 1" constriction (Clerc and Kennedy, 2002; see Section 3.3)	FIPS Type 1" constriction (Mendes et al., 2004; see Section 3.4)	see Section 3.7.2
Number of particles	30 particles		
topology	Von Neumann with "self" included (Mendes et al., 2004; see Section 3.5)	Von Neumann with "self" excluded (Mendes et al., 2004; see Section 3.5)	
minimum and maximum position	-3,+3		
minimum and maximum velocity	-0.5,+0.5		
termination criteria	1000 iterations		
mutation type	Gaussian mutation with exponential decay (Quan et al., 2014a)		Polynomial mutation (Deb and Deb, 2012; Deb, 2009, see Section 3.7.2)
mutation details	Gaussian mean is set to 0, and standard deviation is set to 10% of the absolute value of the ANN parameter		Probability distribution parameter is set to 30, and percentage of particles subjected to mutation is set to 1/num. of ANN parameters

5.4.3. Comparison of generated PIs

Due to the different dataset arrangements employed for the single-objective and multi-objective training, a direct comparison of the PIs generated with the three algorithms can be done only for the test dataset which is the same for each case.

Furthermore, while the optimal MOFIPS-based LUBE models can be univocally identified on the Pareto-front using the MP and NI criteria, PSO and FIPS-based LUBE require the construction of an ensemble from 10 different models. Averaging is thus necessary to produce the final PIs of these ensembles, while the indices employed for the comparison against the MOFIPS solutions are given in terms of medians, as suggested in literature (Khosravi et al., 2011b; Quan et al., 2014a). Table 5.5 reports the comparison of the 90%, 95% and 99% PIs constructed for the 6 hours ahead streamflow discharge of the Susquehanna River at the Meshoppen gauging station, while the results for the Nehalem River at Foss are given in Table 5.6. All the indices described in Section 5.2.2 are shown for comparison, but the discussion is better carried out in terms of PICP and PINAW. From a cursory analysis of the results, it clearly emerges that the MOFIPS solutions substantially outperform the LUBE models built with single objective swarm optimization. For both case studies, the best performer at each confidence level, i.e. the LUBE models providing the narrowest yet valid PIs, is obtained from the MOFIPS Pareto-fronts. It can be seen that the NI models are unsurprisingly those returning the narrowest PIs, with PINAW of around 7.01%, 8.00%, and 12.50% for the Susquehanna River and 8.76%, 11.20% and 14.91% for the Nehalem River. However, the corresponding PICP values of 91.32%, 92.59% and 97.80% for the Susquehanna and of 91.21%, 94.87% and 98.98% for the Nehalem imply that NI solutions return strictly valid PIs only for the 90% case. On the other hand, the models identified by the MP criterion are able to generate valid PIs for all the examined cases. PINAW of 7.9%, 10.7%, and 13% of the streamflow discharge range are slightly greater than those of the NI models, indicating that the MP solutions are the overall best for the

Susquehanna case. Similar conclusions cannot be drawn as easily for the Nehalem case study. Indeed, while the NI solutions fail to ensure the required coverage probability for the 95% and 99% cases, they fall short of only 0.13% and 0.02% with respect to the target CL, and could be regarded as valid. On the other hand, the MP solutions provide PIs which are likely too precautionary for this case, with PICPs considerably larger than the correspondent target CL. Consequently, the PIs of MP solutions are 18% to 45% wider than the NI ones as shown by the PINAW values in Table 5.6. Although the PSO-based LUBE provides valid PIs at each confidence level for both case studies, the produced intervals are too wide for any real practical application. The FIPS-based LUBE shows usually better performances, but it still compares badly against the optimal models returned by the MOFIPS algorithm. This is particularly true the 99% CL case, where the median PICP of the FIPS-based LUBE models is below the requested target for both cases. The results in the last columns of Table 5.5 and 5.6 show that MOFIPS-based LUBE generates better PIs while at the same time providing remarkable speedups. Indeed, despite the 5 restarts, MOFIPS computational times are 40% to 50% smaller than those required by single-objective swarm optimization, indicating that the overhead associated with using augmented swarm topologies and Pareto-fronts calculation is fairly negligible (see end of Section 5.4.2). The superiority of MOFIPS solutions can be also verified from a visual comparison of the PIs generated for the three confidence levels considered, such as shown in Fig. 5.4-5.6 for the last part of the test dataset of the Susquehanna River. It appears that most of the improvements provided by MOFIPS are reflected into a better positioning of the Lower Bound of the PIs, and a more accurate bracketing of peak streamflow discharges. In this regards, of particular interest is Fig.

5.5 showing the PIs at confidence level 95%. For this case, all the algorithms return valid PIs which are also found to comprise the peak discharges for the major storm event occurring on the night of the 21st of March 2008, with a peak flow of 2483 m^3s^{-1}. The interval generated by the MP solution for the peak observation is [2426 ,2724] m^3s^{-1}, which has a width of 298 m^3s^{-1} corresponding to around 12% of the peak discharge itself. On the other hand, the PI of the FIPS-based LUBE is over three times larger including discharges anywhere in the [2005, 2957] m^3s^{-1} range. The accuracy of the PSO-based LUBE models is even worst, with a PI of [772, 3365] m^3s^{-1} which is almost nine times wider than that of the optimal MOFIPS solution, and represents 104% of the peak flow rate.

TABLE 5.5. PERFORMANCES OF LUBE MODELS ON THE TEST DATASET FOR THE SUSQUEHANNA RIVER AND REQUIRED COMPUTATIONAL TIME

		PICP	CWC	PINRW	PINAW	Total computational time [s]
90% confidence level	MOFIPS-based LUBE (NI)	0.9132	0.0719	0.0719	0.0701	3105.2
	MOFIPS-based LUBE (MP)	0.9175	0.0803	0.0803	0.079	
	PSO-based LUBE	0.9143	0.2275	0.2275	0.1992	6132.7
	FIPS-based LUBE	0.913	0.1324	0.1173	0.1129	5952.4
		PICP	CWC	PINRW	PINAW	
95% confidence level	MOFIPS-based LUBE (NI)	0.9259	0.641	0.0816	0.08	3164.6
	MOFIPS-based LUBE (MP)	0.9686	0.1078	0.1078	0.1073	
	PSO-based LUBE	0.9644	0.3757	0.3215	0.2896	6145.1
	FIPS-based LUBE	0.9523	0.1953	0.15	0.1434	5914.1
		PICP	CWC	PINRW	PINAW	
99% confidence level	MOFIPS-based LUBE (NI)	0.978	0.4836	0.134	0.125	3197.3
	MOFIPS-based LUBE (MP)	0.9901	0.1445	0.1445	0.1298	
	PSO-based LUBE	0.9903	0.6947	0.42	0.3628	6181.9
	FIPS-based LUBE	0.9791	0.7377	0.2312	0.205	5879.5

TABLE 5.6. PERFORMANCES OF LUBE MODELS ON THE TEST DATASET FOR THE NEHALEM RIVER AND REQUIRED COMPUTATION TIME

		PICP	CWC	PINRW	PINAW	Total computational time [s]
90% confidence level	MOFIPS-based LUBE (NI)	0.9121	0.0901	0.0901	0.0876	2126.5
	MOFIPS-based LUBE (MP)	0.9208	0.1127	0.1127	0.1033	
	PSO-based LUBE	0.9187	0.2509	0.2509	0.2209	3775.8
	FIPS-based LUBE	0.9175	0.2333	0.2333	0.2106	3743.4
		PICP	CWC	PINRW	PINAW	
95% confidence level	MOFIPS-based LUBE (NI)	0.9487	0.249	0.1182	0.112	2136.7
	MOFIPS-based LUBE (MP)	0.9823	0.1667	0.1667	0.1581	
	PSO-based LUBE	0.9666	0.3189	0.3189	0.2458	4214.8
	FIPS-based LUBE	0.958	0.2399	0.2356	0.2213	3731.1
		PICP	CWC	PINRW	PINAW	
99% confidence level	MOFIPS-based LUBE (NI)	0.9898	0.3444	0.1708	0.1491	2097
	MOFIPS-based LUBE (MP)	0.9947	0.2477	0.2477	0.2158	
	PSO-based LUBE	0.9934	0.4733	0.4088	0.3645	3871
	FIPS-based LUBE	0.9883	0.5783	0.3006	0.2628	3543.1

5.4.4. Wet season vs dry season performances

Further insights on the performances of the MOFIPS-based LUBE methodology can be obtained by decomposing the overall performance indices into those relative to the wet and dry seasons. From an analysis of historical records, it emerges that for both case studies the wet season goes from November to May, while the dry season comprises the remaining 5 months. However, due to the lack of sufficient data samples in the dry season for the test dataset of the Susquehanna River, the analysis will be carried out only for the Nehalem River. The test dataset of the Nehalem River is made for 1/3 by samples recorded during the dry months, while the remaining 2/3 pertains to the wet season. Fig. 5.7 reports the decomposition of the overall test PICP (left) and PINAW (right) indices of the NI solutions for the three CLs. The results shown for the PICP suggest that there are no major differences in the coverage probability across the

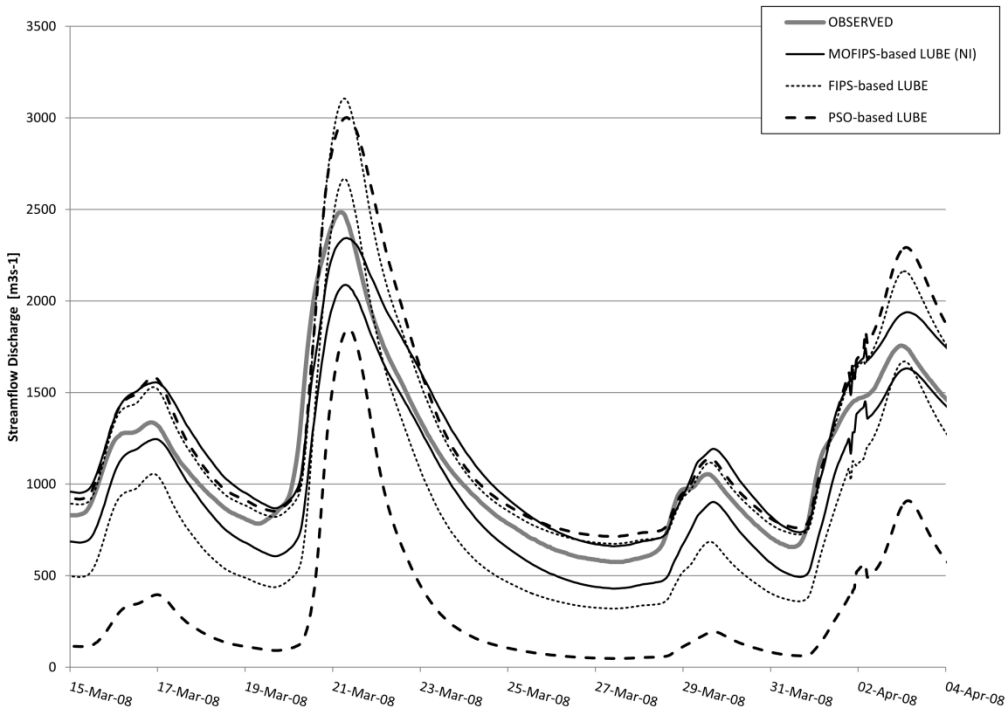


Figure 5.4. LUBE generated prediction intervals at 90% confidence level for the Susquehanna River.

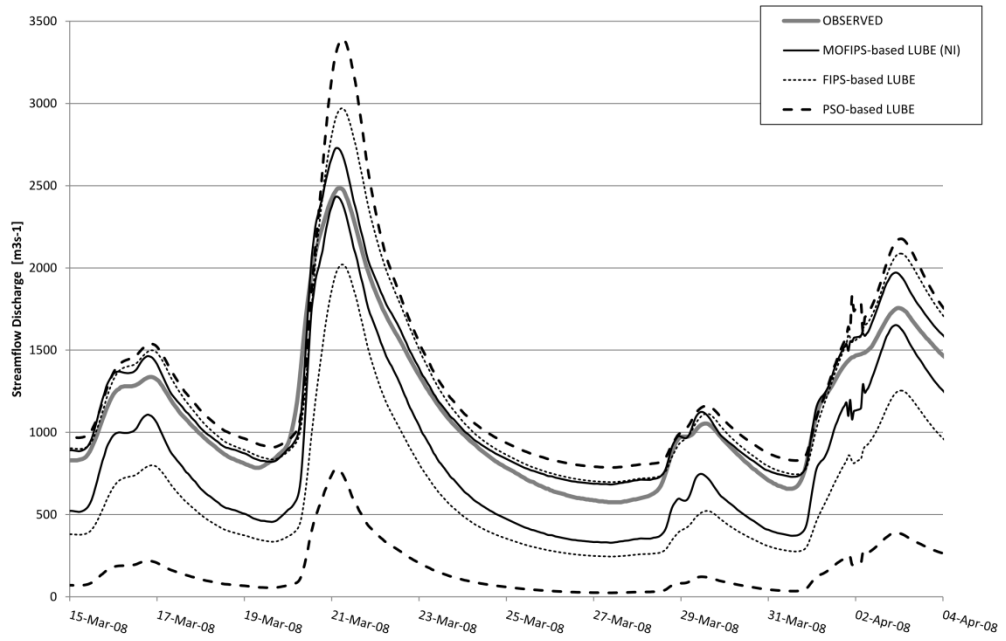


Figure 5.5. LUBE generated prediction intervals at 95% confidence level for the Susquehanna River.

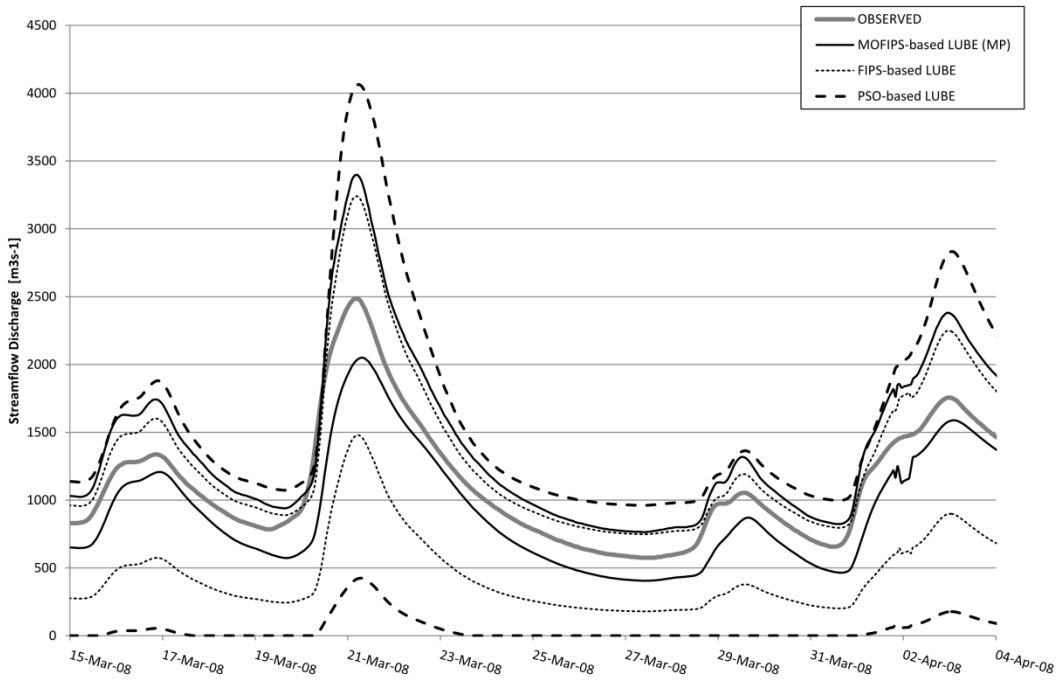


Figure 5.6. LUBE generated prediction intervals at 99% confidence level for the Susquehanna River.

two seasons. On the other hand, the PINAW values are generally lower during the dry season for all the three considered CLs. These results are better examined by taking into account the average flows which are 52.1 and 134.2 m^3s^{-1} for the dry and wet season, respectively. It is interesting to note that, although the average flow during the dry season is 61.2% smaller than that of the wet season, the PINAW values for the dry seasons are at most 16.8% smaller than those of the wet season. This particular value is recorded for the 99% CL case, where the overall PINAW of 14.91% is decomposed into a wet season component of 15.77% and a dry season component equal to 13.13%. The contrast between the difference of these components and that of the seasonal average flows suggests that the width of the PIs is mostly determined by the higher variability of the wet season, which is driven by more frequent and intense rainfall events. This higher

variability forces the LUBE method to widen the PIs in order to increase coverage probability and meet the validity requirement set by the CL. However, since the same ANN model is used for both seasons, the PIs produced during the dry season, albeit narrower than those of the wet season, are wider than what could be expected by analyzing the average flows.

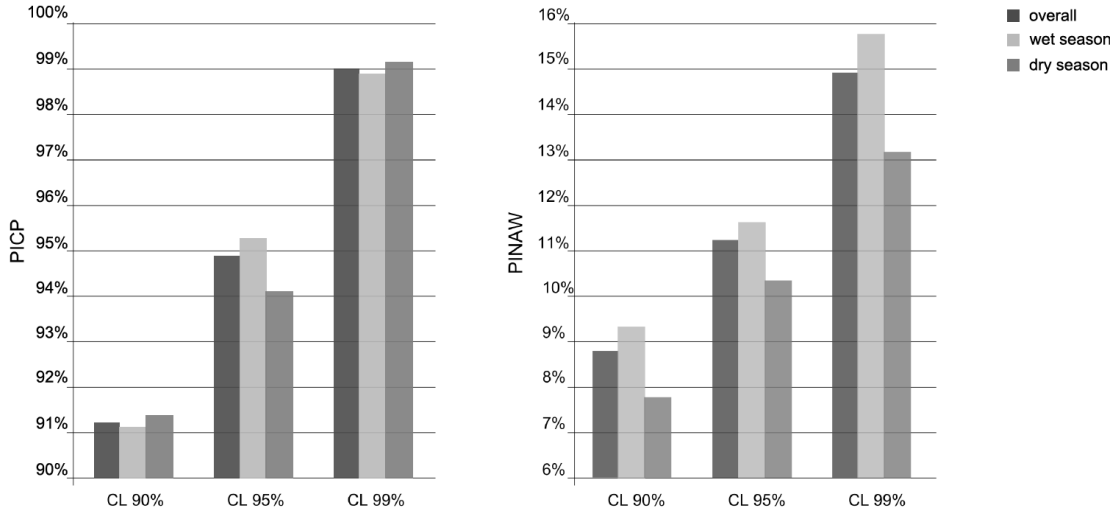


Figure 5.7. Decomposition of test PICP and PINAW for the Nehalem River.

5.5. Conclusions

This study dealt with the application of the LUBE method for the construction of ANN-based PIs of streamflow discharges at 90%, 95% and 99% confidence levels. Single–objective and multi-objective swarm optimization has been employed to develop LUBE models for the prediction of 6 hours ahead streamflow discharges of the for the Susquehanna and Nehalem rivers, US. A novel methodology involving the MOFIPS algorithm was found to provide valid PIs that are substantially narrower than those obtained with single-objective swarm optimization. With average PI widths ranging

from a minimum of 7% to a maximum of 15% of the range of the streamflow data in the test datasets, MOFIPS-based LUBE could be employed for straightforward design of more reliable interval-based streamflow forecasting models. Although the quality of the PIs was found to be significantly affected by the algorithm employed for model development, future studies should be focused on finding more appropriate input selection techniques for interval based hydrological prediction models. In addition, the seasonal decomposition of the overall performance indices encourages seeking for further improvements, which could be obtained, for instance, by resorting to a modular approach where the PIs are produced separately for the dry and wet seasons and joined subsequently.

6. Data-driven input variable selection for rainfall-runoff modeling using binary-coded particle swarm optimization and ELMs

Selecting an adequate set of inputs is a critical step for successful NNRF modeling. In this study, we present a novel approach for Input Variable Selection (IVS) that employs Binary-coded discrete Fully Informed Particle Swarm optimization (BFIPS) and Extreme Learning Machines (ELM) to develop fast and accurate IVS algorithms. A scheme is employed to encode the subset of selected inputs and ELM specifications into the binary particles, which are evolved using single objective and multi-objective BFIPS optimization (MBFIPS). The performances of these ELM-based methods are assessed using the evaluation criteria and the datasets included in the comprehensive IVS evaluation framework proposed by Galelli et al. (2014). From a comparison with 4 major IVS techniques used in their original study it emerges that the proposed methods compare very well in terms of selection accuracy. The best performers were found to be 1) a MBFIPS-ELM algorithm based on the concurrent minimization of an error function and the number of selected inputs, and 2) a BFIPS-ELM algorithm based on the minimization of a variant of the Akaike Information Criterion (AIC). The first technique is arguably the most accurate overall, and is able to reach an almost perfect specification of the optimal input subset for a partially synthetic rainfall-runoff experiment devised for the Kentucky River basin. In addition, MBFIPS-ELM allows for the determination of the relative importance of the selected inputs. On the other hand, the BFIPS-ELM is found to consistently reach high accuracy scores while being considerably faster. By

extrapolating the results obtained on the IVS test-bed, it can be concluded that the proposed techniques are particularly suited for NNRF modeling applications characterized by high nonlinearity in the catchment dynamics.

6.1. Introduction

6.1.1. Input variable selection (IVS) techniques

One of the main issues encountered when developing NNRF applications is represented by the difficulties in identifying the set of inputs to employ for modeling a given hydrological process (Maier and Dandy, 2000; Maier et al., 2010). This task is usually referred to as Input Variable Selection (IVS) (Galelli et al., 2014; Guyon and Elisseeff, 2003; May et al., 2011), and entails the identification of a possibly small set of predictors able to explain the behavior of the output variable. Exclusion of meaningful predictors results in building inaccurate models, no matter how well the other model development steps are carried out. On the other hand, inclusion of irrelevant or redundant inputs hinders the subsequent calibration process by adding more local minima to the error surface defined by the objective function used for ANN training. In addition, unnecessary large input sets result in longer computational time for model development, and undermine post-processing efforts for knowledge discovery and rule extraction (Jain and Kumar, 2009).

6.1.2. Filters, wrappers and embedded IVS techniques

IVS techniques can be broadly divided into *model-free* and *model-based* approaches according to whether or not the selection process is carried out independently from the chosen model (Maier et al., 2010; May et al., 2011). Model-free

approaches are usually known as *filters*, since the significance of the relationship between the output variable and each input candidate is filtered as a preprocessing step, independently of the model employed (Kohavi and John, 1997). Input relevance is estimated through a statistical measure, such as cross-correlation (Coulibaly et al., 2000; Imrie et al., 2000) or Mutual Information (MI) (Bhattacharya and Solomatine, 2005). To prevent redundancy among the selected candidates, partial significance measures can be evaluated by removing the effects of the predictors which have already been selected.

This notion is at the core of efficient automatic filter techniques such as the Partial Correlation Input Selection (PCIS) method (May et al., 2008) and the Partial Mutual Information (PMI) based method (Bowden et al., 2005a; Fernando et al., 2009; May et al., 2008; Sharma, 2000). Another successful approach is that of the Iterative Input variable Selection (IIS) method recently proposed by Galelli and Castelletti (2013a), which employs a ranking scheme based on extremely randomized trees (Galelli and Castelletti, 2013b) to estimate the partial dependence between candidate inputs and the output. Other notable examples of filter techniques developed for hydrological applications may involve dimensionality reduction, stepwise regression, the gamma test, and information theoretic approaches (Noori et al., 2011; Sharma and Mehrotra, 2014; Ssegane et al., 2012; Wan Jaafar et al., 2011). Filter techniques are characterized by good generalization capability since they are not tuned to the specific interaction between the selected inputs and a chosen data-driven model. The lack of an underlying model to be trained also grants them high computational efficiency, and facilitates the physical interpretation of the selected variables. However, these advantages are obtained at the expense of not considering the actual gain in model performances given by each

selected variable. In addition, filters usually treat each candidate variable separately, and therefore do not take into account the information that could be gained by individually irrelevant candidates with high combined explanatory power.

These issues can be overcome by resorting to model-based approaches, which are generally divided into *wrapper* and *embedded* methods (Blum and Langley, 1997; Guyon and Elisseeff, 2003; Kohavi and John, 1997). The difference between these two classes lies in the fact that the latter techniques perform the IVS task along with model calibration, while the former treat the model as a pure black-box, whose accuracy is maximized by searching for an optimal subset of inputs (Guyon and Elisseeff, 2003). However, the line separating these two classes is not always easy to draw, especially when the most common approach of using global optimization (GO) techniques is adopted to perform the search (May et al., 2011). With reference to Section 3.2, using GO will result in a wrapper technique if the algorithm searches for the optimal set of inputs and, at most, some large scale properties of the ANN model (i.e. number the number of hidden layers and number of units for each hidden layers) (Abrahart et al., 1999; Bowden et al., 2005a, 2005b; Chen and Chang, 2009). On the other hand, the method can be classified as embedded if the EA performs the optimization of ANN weights along with the IVS process (Leahy et al., 2008; Yao, 1999). The distinction is less clear when the EA is employed to directly manipulate links and nodes within the ANN topology but another learning algorithm is used to estimate the value of the synaptic weights (Leahy et al., 2008). It is understood that embedded algorithms are more computationally efficient as they focus on the development of a single final model. However, the increased complexity of the search process may outweigh the benefits of

reduced computational costs, which even for the leanest techniques are usually orders of magnitudes greater than those of the fastest model-free techniques.

6.1.3. Development of wrapper techniques with ELM and binary-coded FIPS

The speed gap between model-based and model-free techniques can be reduced by resorting to fast training algorithms, such as the Extreme Learning Machines (ELM) paradigm introduced in Section 2.2 (Huang et al., 2011, 2006, 2004). Despite their advantages over traditional ANNs and similar techniques, to our best knowledge ELMs have never been employed for NNRF applications at the time of this writing. It is therefore the purpose of this study to explore the potential of ELMs by exploiting their computational efficiency and enhanced generalization capability to develop fast and accurate IVS algorithms. In particular, the proposed techniques are wrappers built by pairing ELMs with the BFIPS and MBFIPS algorithms presented in Section 3.6 and 3.7.4, respectively. BFIPS-ELM wrappers are built using the Root Mean Square Error (RMSE) and several variants of the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) as the objective functions to minimize. The different metrics are employed to check whether penalizing model complexity grants better performances by preventing over-fitting that may occur due to the inclusion of irrelevant or redundant variables. This is a known issue of wrapper methods (Galelli et al., 2014) that could be aggravated when employing ELMs, whose training process may suffer in presence of irrelevant or correlated variables (Miche et al., 2010). This problem is directly addressed when developing MBIFPS-ELM wrappers, since the number of inputs is concurrently minimized with the RMSE through a bi-objective Pareto-based optimization (Xue et al., 2013).

The suitability of the proposed methods is assessed using the evaluation criteria and datasets of the comprehensive evaluation framework proposed by Galelli et al. (2014). In particular, the algorithms are run on 19 fully synthetic datasets that account for a range of properties typical of hydrological data (e.g. nonlinear, non-Gaussian, redundant), as well as on a partially synthetic rainfall-runoff dataset based on the Kentucky River basin. As entailed by the IVS framework, quantitative evaluation of the proposed wrappers is done in terms of input selection accuracy and computational times. In the same way, the criteria defined for explanation capability, flexibility, ease of use and robustness are used to evaluate the techniques from a qualitative point of view. For a thorough assessment of the methods, a comparison with the PMI, PCIS, IIS and GA-ANN techniques tested in the original paper presenting the framework is also carried out. The Chapter is organized as follows. Section 6.2 presents the ELM-based wrapper methods and how they are developed. Section 6.3 briefly describes the IVS framework, while the results of the experiments are shown and discussed in Section 6.4. Conclusions are drawn in Section 6.5.

6.2. ELM-based wrapper development using BFIPS and MBFIPS

6.2.1. Binary particle encoding

As discussed in Section 3.2, when GO techniques are employed to identify the optimal inputs for an ANN, the subsets of selected inputs are encoded in binary strings of the same length of the total number of candidate predictors (Bowden et al., 2005a, 2005b). The *i*-th bit of the string is set to 1 if the *i*-th input has been selected by the searching algorithm, and it is set to 0 otherwise. The selected predictors are then fed to

the ANN, which is first trained and then evaluated to provide a measure of its performances. The characteristics of the data-driven models may be chosen beforehand and remain fixed during the search process, but wrapper performances could be improved by evolving some model specifics along with the input selection (Chen and Chang, 2009). In this work we decided to include the number of hidden neurons of the ELM, the type of the activation function, as well as the value of the parameter λ in the search process (see Section 2.2). The automatic tuning of the first two variables is expected to produce flexible wrappers that can adjust their complexity and functional form to better capture the underlying relationship in the data. In addition, including λ in the encoded parameters will allow using the solutions in (2.21-2.22) to determine the output weights of the ELM more accurately, enhancing its performances. If the BFIPS and MBFIPS algorithms are chosen to develop the ELM-based wrapper, we have that the position vector of a particle is a binary string as the one depicted in Fig. 6.1, where l_{inputs} is the number of bits used to directly encode the selected subset of inputs, $l_{inputs} = "number\ of\ candidate\ variables"$; l_{nodes} is the number of bits used to encode the number of hidden neurons between a maximum (NH_{max}) and a minimum value (NH_{min}), $l_{nodes} = \lceil log_2(NH_{max} - NH_{min}) \rceil$; and l_{act} is the number of bits needed to encode the type activation function used, $2^{l_{act}} \geq "types\ of\ activation\ functions"$. The number of bits l_λ needed to encode the parameter λ is given by $l_\lambda = \lceil log_2(2M) \rceil$, where M is a positive integer so that the value of λ can be defined by increasing powers of 2, $\lambda = 2^i, i \in Z \cap (-M, +M]$. The $\lceil \ \rceil$ brackets employed in the previous lines identify the *ceiling* function which maps a real number to the smallest following integer.

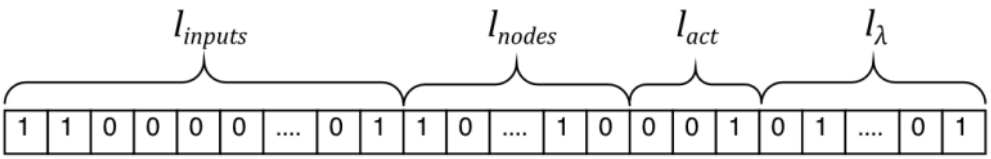


Figure 6.1. Binary encoding scheme for the ELM-based wrappers.

6.2.2. BFIPS-ELM wrappers

The encoding described in the previous paragraphs is used to build different BFIPS-ELM wrappers depending on the objective function chosen to assess ELM accuracy. In particular, this study employs the Root Mean Square Error (RMSE), as well as generalized versions of the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) (Qi and Zhang, 2001). While the version with RMSE represents the basic form of the algorithm, the latter criteria are used to apply different penalties to model complexity in order to obtain parsimonious models. Indeed, this is a prominent issue when developing wrapper methods which are prone to over-fitting due to the inclusion of redundant/irrelevant inputs (Galelli et al., 2014). In addition, ELM are known to have problems when the training dataset presents irrelevant or correlated variables (Miche et al., 2010). If y_i is the i-th observed value of the output variable, $\hat{y}_i$ the model predicted value, and N is the total number of observations, the Mean Squared Error (MSE) can be defined as

$$MSE = \frac{\sum_{i=1}^{T}(y_i - \hat{y}_i)^2}{N}.$$

The expression of the three criteria can be obtained from the MSE

$$RMSE = \sqrt{MSE}, \tag{6.1}$$

$$AIC = \log(MSE) + \frac{2m^g}{N}, \tag{6.2}$$

$$BIC = \log(MSE) + \frac{m^g \log(N)}{N}, \tag{6.3}$$

where m is the model complexity, defined at the end of Section 2.2 for an ELM model, and g is an exponent that can be adjusted to assign different penalties to m. For equal values of the exponent g, the BIC criterion always applies a heavier penalty to model complexity. Notwithstanding the objective function that is minimized during the optimization process, the BFIPS-ELM chosen is always the particle with the lowest value of the employed criterion.

6.2.3. MBFIPS-ELM wrapper

The issue of selecting irrelevant and redundant inputs can be directly tackled by considering the number of inputs as an additional objective function to minimize (Xue et al., 2013). This is done by the MBFIPS-ELM wrapper proposed in this study, where the number of selected inputs, i.e. *cardinality*, and RMSE are the two objective functions employed for the Pareto-based optimization. This approach is expected to 1) force the identification of the most meaningful inputs for each possible subset dimension, since only one non-dominated solution for each cardinality will be at most included in the Pareto-front; and 2) limit the exploration of unnecessarily large subsets, since non-dominant solutions characterized by greater subset dimensions will be included in the Pareto-front only if they have lower RMSE. Since all the solutions in the Pareto-front are equally optimal, a criterion is necessary to select the final wrapper at the end of the optimization process. Ideally, the non-dominant solution with the largest number of

inputs (lowest RMSE) should be chosen, since including more inputs than the real ones should result in lower performances (higher RMSE). However, due to the random component in ELM training, and the problems in identifying an optimal value of the ridge regression constant, it is possible that training a model with a dataset including other inputs besides the real ones might result in lower RMSEs. A simple workaround can be implemented by choosing a cutoff percentage ρ, and then selecting the Pareto-optimal solution with the lowest cardinality for which the following inequality still holds

$$\left(\frac{RMSE^* - RMSE_{MIN}}{RMSE_{MIN}}\right)\% \leq \rho, \qquad (6.4)$$

where $RMSE^*$ is the RMSE value of the selected solution, while $RMSE_{MIN}$ is the lowest of the Pareto-optimal solutions. Setting ρ = 0% will simply return the boundary solution with $RMSE_{MIN}$, while ρ > 0% might select an inward solution with higher RMSE but smaller subset dimension, as illustrated in Fig. 6.2. The shaded area in Fig. 6.2 represents the region of the Pareto-front where (6.4) holds.

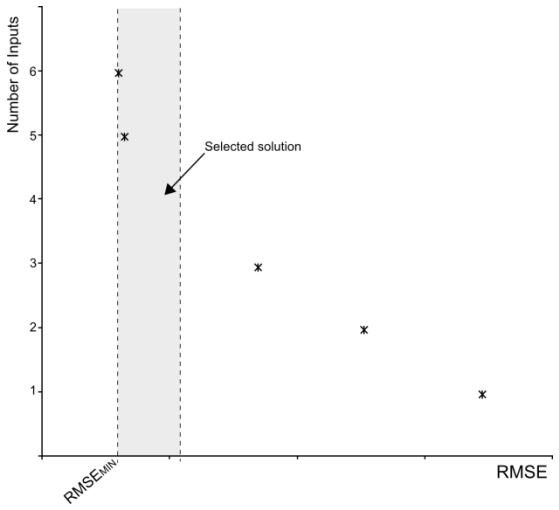


Figure 6.2. Selection of optimal solution for the MBFIPS-ELM.

6.3. The Input Variable Selection evaluation framework

The performances of the wrapper techniques described in the previous section are assessed using the Input Variable Selection (IVS) framework proposed by Galelli et al. (2014). The framework was developed to facilitate the objective evaluation and comparison of IVS algorithms for environmental data-driven modeling. The authors provide a comprehensive test-bed of 26 datasets, as well as several criteria to assess the accuracy and suitability of IVS techniques from both a quantitative and a qualitative point of view.

6.3.1. Benchmark datasets

The test-bed is made of 19 fully synthetic and 7 partially synthetic datasets that account for a comprehensive range of properties typical of environmental data. In addition, 30 replicates of each benchmark dataset are provided to strengthen the statistical significance of the obtained results. Since the focus of this paper is on the development of IVS techniques for hydrological modeling, the study will not employ the 6 benchmark datasets made available for water quality modeling. The test-bed thus comprises the whole set of 19 fully synthetic datasets, as well as a partially synthetic rainfall-runoff experiment designed for the Kentucky River basin. The characteristics of the 20 benchmark datasets employed in this work are summarized in Table 6.1, where the number of observations *N*, number of relevant inputs *K*, and total number of candidate inputs *P* are also given. The ratio of the number of observations over the total number of candidates *N/P* is also highlighted due to its importance in determining the likelihood of the IVS technique to over-fit the data by selecting irrelevant or redundant inputs. Small values of *N/P* denote greater risks of over-fitting, and the risk increases

with increasing correlation between the candidate inputs. The set of comprehensive features of real hydrological data reflected in the test-bed are: non-Gaussian output, high nonlinearity in the input-output relationship, high noise in the data, high collinearity among input variables, inter-dependency of the candidate inputs, and incomplete information in the dataset. Further details on the benchmark datasets could be found in the original paper of Galelli et al. (2014), and on the website of the IVS framework (http://ivs4em.deib.polimi.it/).

TABLE 6.1. CHARACTERISTICS OF THE BENCHMARK DATASETS OF THE IVS FRAMEWORK

Dataset	*N*	*K*	*P*	*N/P*	*Non-Gaussian output*	*Highly nonlinear*	*High noise*	*High collinearity*	*Inter-dependency*	*Incomplete information*
1. AR1	500	1	15	33.3			X	X		
2. AR9_500	500	3	15	33.3			X	X		
3. AR9_70	70	3	15	4.7			X	X		
4. TAR1	500	1	15	33.3			X	X		
5. TAR2	500	2	15	33.3			X	X		
6. NL_500	500	3	15	33.3	X	X				
7. NL_70	70	3	15	4.7	X	X				
8. NL2	500	3	15	33.3	X	X	X	X		
9. Bank_fm	400	8	32	12.5	X					X
10. Bank_fh	400	8	32	12.5	X		X			X
11. Bank_nm	400	8	32	12.5	X	X				X
12. Bank_nh	400	8	32	12.5	X	X	X			X
13. Friedman_c0_10_m	250	5	10	25		X				
14. Friedman_c0_10_h	250	5	10	25		X	X			
15. Friedman_c0_50_m	250	5	50	5		X				
16. Friedman_c0_50_h	250	5	50	5		X	X			
17. Friedman_c25_10_m	250	5	10	25		X		X		
18. Friedman_c25_10_h	250	5	10	25		X	X	X		
25. Kentucky	4739	4	21	225.7	X			X		
26. Miller	200	2	3	66.7	X				X	

6.3.2. Selection accuracy criteria

The Selection Accuracy (SA) score is the metric recommended for the quantitative assessment of the effectiveness of an IVS algorithm

$$SA = \gamma SA_c + (1 - \gamma)SA_e \qquad (6.5)$$

where

$$SA_c = \frac{k}{K}$$

$$SA_e = 1 - \frac{p}{P - K}.$$

SA_c is the ratio of correct input that have been selection k over the total number of correct inputs K. On the other hand, SA_e is based on the proportion of extraneous inputs that have been retained p with respect to the total number of extraneous inputs in the dataset $P - K$. The parameter γ ranges from 0 to 1 and is a tradeoff value weighting the relative importance of the two components. The value of the SA score can range from 0 to 1, where $SA = 1$ denotes a correctly specified model, while $SA = 0$ corresponds to a completely misspecified model, with no relevant inputs and all extraneous inputs selected. The values of SA_c and SA_e are also bounded in the [0,1] range. Apart from the case of perfect specification ($SA_c = 1, SA_e = 1 \rightarrow SA = 1$), we also have that of over-specification of extraneous inputs ($SA_e < 1$) and/or under-specification of relevant inputs ($SA_c < 1$) may occur.

6.3.3. Other evaluation criteria

Apart from the use of the SA metrics defined in the previous section, Galelli et al. recommend a thorough investigation of IVS techniques using other criteria that assess the algorithm 1) computational efficiency, 2) ease of use and robustness, 3) explanation

capability and 4) flexibility. While computational efficiency still entails the estimation of quantitative measures, the remaining three criteria are qualitative in nature. Computational efficiency is a key aspect in determining the success of an algorithm since very accurate but slow techniques might be of no practical use. The analytical determination of the algorithm computational complexity as a function of N and P would be ideal for determining its computational efficiency, allowing for the preemptive estimation of the time required to process a given case study. However, such theoretical solutions are not available for heuristics such as GA and PSO. Therefore total run-time, which depends on software implementation and employed hardware, has to be used in these cases. Ease of use and robustness are essentially related to the number of parameters that need to be set before running the IVS algorithm, the expertise needed by the user to set them appropriately, and how well the algorithm performs using a set of default parameters. The explanation capability of an IVS technique mostly concerns its ability to determine the relevance of each selected input with respect to the others, while its flexibility refers to the ease with which the algorithm components can be interchanged with other methods.

TABLE 6.2. OVERALL MEAN AND MEDIAN SELECTION ACCURACY SCORES OF THE BEST PERFORMING ELM-BASED WRAPPERS

Type of wrapper	*Objective function(s)*	*Complexity penalty*	*Overall mean selection accuracy score*			*Overall median selection accuracy score*		
			SAc	*SAe*	*SA*	*SAc*	*SAe*	*SA*
BFIPS-ELM	RMSE	---	0.896	0.817	0.872	0.948	0.815	0.912
	AIC	logarithmic	0.888	0.857	0.879	0.939	0.888	0.927
	BIC	logarithmic	0.870	0.879	0.873	0.930	0.919	0.909
MBFIPS-ELM	RMSE, num. of inputs	---	0.833	0.98	0.877	0.932	0.988	0.936

6.4. Results and Discussion

6.4.1. Experimental setup

The accuracy and suitability of the proposed ELM-based wrappers are assessed using the IVS framework criteria summarized in the previous chapter. The parameter γ regulating the tradeoff between the selection of relevant inputs and the exclusion of extraneous one was set to 0.7, as suggested by Galelli et al., (2014). Therefore more weight is given to the inclusion of correct inputs when computing the value of SA for each experiment according to (6.5). All the experiments were carried out using MATLAB® on a 2.20 GHz Intel i7-3632QM CPU with 8 GB RAM.

*Type of wrappers tested***.** A total of 10 different configurations are tested using the benchmark datasets. These are 9 BFIPS-ELMs with different objective functions and 1 MBFIPS-ELM. The objective functions for the BFIPS-ELMs are the RMSE (6.1), as well as 4 AIC and 4 BIC criteria that were derived from (6.2-6.3) using different types of complexity penalty. The penalties applied are, in increasing order of penalty, *logarithmic*, *square root*, *linear*, and *quadratic*, which are obtained by setting the exponent g in (6.2-6.3) to $\log_m(\log(m))$, 0.5,1 and 2 respectively. As described in Section 6.2.3, the MBFIPS-ELM wrappers are developed by concurrently minimizing the RMSE and number of selected inputs using a Pareto-based approach. Values of ρ=0%, 1%, 2.5% and 5% were tested to select the optimal model at the end of the MBFIPS optimization process. Five-fold cross-validation is employed to estimate the value of the RMSE and all the AIC/BIC objective functions employed.

Algorithm setup. Both the BFIPS and MBFIPS algorithms are stopped after 1000 iterations, or if no improvements are obtained for 30 consecutive iterations. The swarms

are made of 30 particles arranged to form a lattice topology, such as the one shown in Fig. 3.4f. The mutation rate is set to $\mu = 0.05$, with a maximum of M = 5 randomly selected particles undergoing mutation at each iteration (see Sections 3.6 and 3.7.4). The maximum number of particles in the MBFIPS Pareto-front is set equal to the total number of inputs of each case study.

Binary encoding. The number of ELM hidden nodes is allowed to vary between NH_{min} = 1 and NH_{max} = 250, so that l_{nodes} = 8 bits are required for the encoding. Four different types of activation functions (l_{act} = 2) are considered for the hidden units, namely linear, log-sigmoid (*logsig*), hyperbolic tangent sigmoid (*tansig*), and radial basis function (*rbf*). The number of bits needed to encode the ridge regression parameter is set to l_λ = 6 , so that λ can take any value in the increasing power-of-two sequence going from 2^{-31} = 4.657E-10 to 2^{32}=4.295E09. A total of 16 bits plus $l_{inputs} = P$ are therefore required to encode the wrapper models for each benchmark dataset, with the binary string length varying from a minimum of 19 bits for Miller case to a maximum of 66 bits for the largest Friedman datasets.

6.4.2. Quantitative assessment of wrapper performances

6.4.2.1.Comparison of overall selection accuracy

The overall mean and median selection accuracy scores are reported in Table 6.2, as obtained from the mean SA, SA_c and SA_e scores of the 30 repetitions of all the 20 benchmark datasets. Only values for the best BFIPS-ELM configurations employing the AIC and BIC criteria are reported, which are obtained in both cases using the logarithmic complexity penalty. On the other hand, the overall best MBFIPS-ELM

performances are recorded when the cutoff parameter ρ is set to 1%. The overall mean SA scores are almost equivalent for the reported techniques, with a maximum of 0.879 for the BFIPS-ELM obtained minimizing the AIC. However similar in terms of value, these mean SA scores correspond to different combinations of the SA_c and SA_e scores. It emerges that the BFIPS-ELM obtained without any penalty term, i.e. with the RMSE objective function, is on average the most successful in identifying the relevant inputs with $SA_c = 0.896$, but the least effective in excluding redundant/irrelevant variables with $SA_e = 0.817$. Introducing increasing penalty terms results in the exclusion of some relevant variables to the advantage of retaining fewer extraneous ones, as shown for the BFIPS-ELM obtained with the AIC and BIC criteria. A similar but more pronounced effect is noticed for the MBFIPS-ELM wrapper, for which the lowest mean SA_c of 0.833 and highest mean SA_e of 0.980 are recorded. Fig. 6.3 shows how varying the exponent g in (6.2-6.3) affects the selection accuracy of the wrappers. It can be seen that increasing penalties produce a raise in SA_e but a very steep decline in SA_c that substantially reduces the wrapper effectiveness for $g > 0.5$. These findings suggests that, opposite to ANN modeling where the original AIC and BIC criteria are usually employed ($g = 1$) (Dawson and Wilby, 2001; May et al., 2008), lower penalties should be applied to reduce model complexity when employing ELM. This is reasonable if one considers that the input weights of ELM are randomly assigned, thus the model requires more parameters to capture the underlying relationship in the data. This might represent a problem when using AIC and BIC to build ELM-based IVS techniques, as they penalize overall model complexity rather than the number of inputs directly. Fig. 6.4 shows that similar trends in the change of the SA_c and SA_e scores are also noticed by varying the

cutoff parameter ρ used to determine the final MBFIPS-ELM models. However, the raise in SAe with increasing values of ρ correspond only to a mild decrease of SA_c, so that the values of SA do not drop as much as seen for the BFIPS-ELM wrappers developed using AIC/BIC.

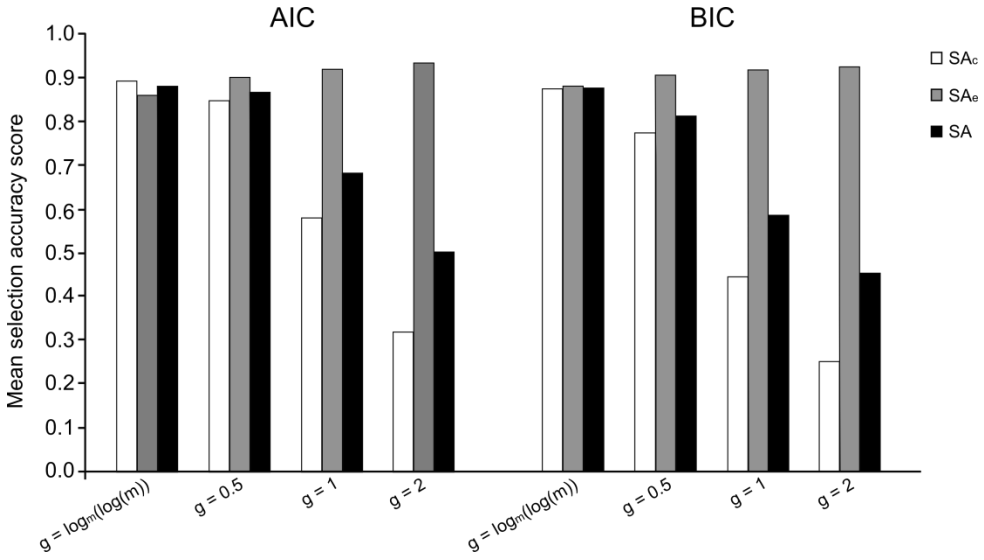


Figure 6.3. Overall mean selection accuracy of BFIPS-ELM for different complexity penalties.

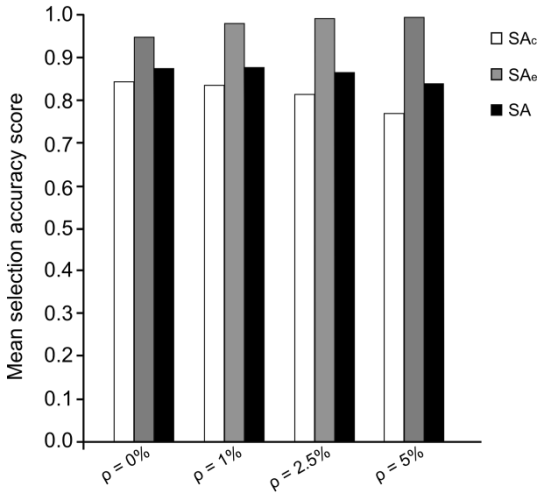


Figure 6.4. Overall mean selection accuracy of MBFIPS-ELM for different values of ρ.

A better comparison of the effectiveness of the wrapper methods in Table 6.2 can be done by considering the overall median selection accuracy scores, which are less sensitive to possible outliers due to particularly weak (or strong) performances in some case studies. Indeed, the median accuracy scores suggest that the MBFIPS-ELM is the overall best performer, with a median SA score of 0.936. The very high values of both mean and median SA_e scores indicate that the MBFIPS-ELM is generally very effective in preventing the over-specification of extraneous inputs. On the other hand, the larger difference between mean SA_c and median SA_c suggest that the MBFIPS-ELM might be sensitive to under-specification of relevant inputs in some particular cases. With a median SA of 0.927, the BFIPS-ELM obtained with AIC comes a close second in terms of overall accuracy, representing the best wrapper among those obtained with single-objective swarm optimization. The performances of this technique along with those of the MBFIPS-ELM will be discussed in detail in the following paragraphs to highlight their strengths and weaknesses of the two approaches on each dataset. For the sake of conciseness, the BFIPS-ELM obtained with AIC will be referred to simply as BFIPS-ELM hereafter, unless otherwise specified.

6.4.2.2.Comparison of selection accuracy on each dataset

The mean selection accuracy scores of the BFIPS-ELM and MBFIPS-ELM as computed from the 30 repetitions of each case study are reported in Table 6.3. Furthermore, Table 6.4 shows the total number of times that the final model employed a certain type of activation function, where *logsig* and *tansig* have been grouped as *sigmoid* functions without any loss of generality. The information provided by this table will facilitate the analysis of wrapper performances on each case study.

TABLE 6.3. MEAN SELECTION ACCURACY SCORES OF BFIPS-ELM AND MBFIPS-ELM FOR EACH BENCHMARK DATASET

Dataset	*BFIPS-ELM*			*MBFIPS-ELM*		
	SAc	*SAe*	*SA*	*SAc*	*SAe*	*SA*
1. AR1	1.000	0.938	0.981	1.000	0.988	0.996
2. AR9_500	1.000	0.847	0.954	1.000	0.992	0.998
3. AR9_70	0.944	0.833	0.911	0.944	0.858	0.919
4. TAR1	1.000	0.895	0.969	1.000	0.988	0.996
5. TAR2	1.000	0.918	0.975	1.000	0.987	0.996
6. NL_500	1.000	1.000	1.000	1.000	1.000	1.000
7. NL_70	1.000	1.000	1.000	0.956	1.000	0.969
8. NL2	0.767	0.911	0.810	0.678	0.981	0.769
9. Bank_fm	0.750	0.714	0.739	0.429	0.994	0.599
10. Bank_fh	0.613	0.760	0.657	0.371	0.976	0.553
11. Bank_nm	0.796	0.754	0.783	0.683	0.988	0.775
12. Bank_nh	0.733	0.726	0.731	0.608	0.961	0.714
13. Friedman_c0_10_m	1.000	1.000	1.000	0.993	1.000	0.995
14. Friedman_c0_10_h	0.933	0.967	0.943	0.907	0.987	0.931
15. Friedman_c0_50_m	0.853	0.684	0.803	0.920	0.989	0.941
16. Friedman_c0_50_h	0.867	0.682	0.811	0.813	0.956	0.856
17. Friedman_c25_10_m	0.933	0.980	0.947	0.873	1.000	0.911
18. Friedman_c25_10_h	0.573	0.940	0.683	0.500	0.973	0.642
25. Kentucky	1.000	0.880	0.964	0.975	0.990	0.980
26. Miller	1.000	0.700	0.910	1.000	1.000	1.000

TABLE 6.4. TYPE OF ACTIVATION FUNCTIONS OF THE OPTIMAL MODELS

Dataset	*BFIPS-ELM*			*MBFIPS-ELM*		
	sigmoid	*linear*	*rbf*	*sigmoid*	*linear*	*rbf*
1. AR1	11	18	1	14	13	3
2. AR9_500	2	28	0	12	18	0
3. AR9_70	3	27	0	12	18	0
4. TAR1	19	3	8	17	0	13
5. TAR2	24	1	5	23	1	6
6. NL_500	5	0	25	13	0	17
7. NL_70	27	0	3	20	0	10
8. NL2	28	0	2	24	0	6
9. Bank_fm	0	30	0	22	0	8
10. Bank_fh	0	30	0	20	1	9
11. Bank_nm	13	17	0	27	0	3
12. Bank_nh	2	28	0	27	2	1
13. Friedman_c0_10_m	30	0	0	26	0	4
14. Friedman_c0_10_h	28	1	1	27	0	3
15. Friedman_c0_50_m	0	30	0	25	0	5
16. Friedman_c0_50_h	0	30	0	19	5	6
17. Friedman_c25_10_m	25	0	5	24	0	6
18. Friedman_c25_10_h	25	0	5	20	0	10
25. Kentucky	28	0	2	20	0	10
26. Miller	0	30	0	2	28	0

AR1 and AR9 datasets. These three datasets are obtained from linear autoregressive models of order 1 and 9, respectively. Despite the high noise and collinearity, the high ratio of observations to number of candidate inputs *N/P* allows both techniques to correctly identify the relevant inputs (SA_c = 1) for the AR1 and AR9_500 cases. However, the MBFIPS-ELM is more efficient in avoiding the selection of extraneous inputs thus obtaining a SA score very close to 1. Albeit a drop in accuracy is noticed for the AR9_70 dataset, both techniques still record SA scores of over 0.91, thus being able to perform the IVS task satisfactorily even when considerably less observations are available. From Table 6. 4 it can be seen that there is a prevalence of linear ELM models, showing that the wrapper techniques can recognize the linear relationship adequately even for the AR9_70 case.

TAR datasets. The TAR datasets are obtained from a threshold autoregressive model. The nonlinearity in the dataset is recognized by both wrapper techniques as the vast majority of the final models are nonlinear. The performances on the these datasets are very similar to that of the AR cases, with the high *N/P* ratio granting the correct specification of relevant inputs, and the MBFIPS wrapper being more efficient in avoiding the over-specification of extraneous ones.

NL datasets. The high nonlinearity in the input-output relationship of these dataset is reflected in the total absence of linear models among the optimal ones found for each repetition of each case study. Both wrappers achieve SA = 1 for the NL_500 case, with the BFIPS-ELM providing perfect input specification also for the NL_70 dataset. Adding high noise and high collinearity in the data is found to reduce the accuracy of the algorithms for the NL2 case. However, the performances of the two

techniques, and of the BFIPS-ELM in particular (SA = 0.81), are still remarkably high if one considers the complexity of this case study. This suggests that the proposed wrappers are well suited for handling highly nonlinear datasets.

Bank datasets. These datasets are representative of cases where there is incomplete information about the output data, which has been obtained from a Bank simulator that used a total of 32 variables of which only 8 are available in the datasets. The original time series of the remaining 24 inputs were randomly shuffled so that they have to be considered as irrelevant. The behavior of the two wrappers is very different for these datasets. The MBFIPS-ELM is generally very accurate in selecting only relevant variables with SA_e close to 1 in most cases, but is prone to under-specification. On the other hand, the BFIPS-ELM is more able to identify the relevant inputs, but it tends to retain some of the irrelevant variables. Overall, the BFIPS-ELM performs better on these benchmark datasets, especially for the linear instances (Bank_fm, Bank_fh) where the MBFIPS-ELM selects on average less than half of the relevant inputs. From the numbers in Table 6.4, it appears that this difference in the performances is due to the fact that the MBFIPS algorithm returns only nonlinear ELM models for these cases, while the BFIPS-ELM final solutions are always linear, thus more appropriate for modeling the linear instances of the dataset. If these two experiments are rerun by forcing the MBFIPS to select only linear activation functions, the values of SA raise to 0.743 for the Bank_fm and to 0.605 for the Bank_fh, showing that for these particular cases the original MBFIPS fails to match the underlying functional form of the dataset. This does not seem to happen for the nonlinear cases where the performances of the two wrappers are similar.

Friedman datasets. These datasets have been generated using the Friedman regression function, which has a highly nonlinear functional form. The BFIPS-ELM method obtains SA = 1 for the case with no collinearity, moderate noise and 10 inputs (Friedman_c0_10_m), with the MBFIPS-ELM also showing a near to perfect score. Adding high noise to this dataset (Friedman_c0_10_h) lowers the accuracy of both techniques which however are still well above 0.9. Good performances are also witnessed in presence of high collinearity (Friedman_c25_10_m), but a consistent drop is noticed when both high collinearity and high noise are present (Friedman_c25_10_h). Albeit the BFIPS-ELM outperforms its multi-objective counterpart in all these examples, it scores noticeably lower for the cases with 50 inputs (Friedman_c0_50_m, Friedman_c0_50_h) due to over-specification. In addition, despite the problem being highly nonlinear, the BFIPS selects linear models for all the repetitions of these datasets, suggesting that single-objective optimization may struggle to obtain high accuracy in these cases. The fact that even the performances of BFIPS-ELM built with other objective function are always substantially lower than those of the MBFIPS-ELM suggest that the latter algorithm might perform better for larger numbers of candidate inputs.

Kentucky dataset. This partially synthetic dataset was created using real streamflow and effective rainfall data from a gauging station and several meteorological stations in the Kentucky River basin. Although this dataset alone cannot represent the wide spectrum of different hydrological scenarios typical of rainfall-runoff modeling applications, it is unlikely that techniques scoring poorly on this benchmark would perform the IVS task accurately for other cases. The predictand for this dataset is the

streamflow discharge Q_t reconstructed by a SLFN for a gaging station along the Kentucky River. The set of candidate variables is given by effective rainfall ER measured at time steps t, t-1, …, t-10, as well as previous streamflow discharges at time steps t-1, t-2, …, t-10. Of the 21 inputs constituting the candidate pool, only 4 have been used as inputs for the SLFN, namely Q_{t-1}, Q_{t-2} , ER_t and ER_{t-1}, therefore

$$Q_t = f(Q_{t-1}, Q_{t-2}, ER_t, ER_{t-1}) + \epsilon \qquad (6.6)$$

where $f(\cdot)$ represents the SLFN, and ϵ a random noise component. The 30 replicates of this dataset were built by reshuffling the original 4739 samples, and regenerating the random noise component ϵ. As it emerges from Table 6.3, both the proposed techniques are able to achieve very high scores on this dataset, with the MBFIPS-ELM reaching almost perfect input variable specification. Therefore, these ELM-based wrappers show a great potential as IVS methods for rainfall-runoff modeling and NNRF applications. Their consistency is also reflected in the fact that they select only sigmoid and radial basis functions, which are known to provide better performances over linear transfer functions in NNRF applications. As it will be shown later in Section 6.4.3, the analysis of the Pareto-fronts returned by the MOFIPS-ELM allows for the determination of the relative importance of each input variable. It emerges that the order of importance, from the most relevant to the least relevant, is Q_{t-1}, Q_{t-2} , ER_t then ER_{t-1}. The increment in predictive accuracy given by including these variables in order of importance is shown in the scatter plots of Fig. 6.5 a-d for the first replicate of the dataset, along with the corresponding values of the coefficient of determination R^2 (see APPENDIX B).

Miller dataset. This dataset was devised as an example where 2 highly correlated variables, with little or no predictive value of their own, have great explanatory power

when combined. A third variable is also present which is highly correlated with the output, but totally irrelevant for its determination. Contrary to other IVS techniques that select one input at a time, wrapper techniques that check the explanatory power of subsets should be able to select the optimal inputs. Indeed, the MBFIPS-ELM reaches perfect specification for all the repetitions, while the BFIPS-ELM, although always able to select the two relevant variables, over-specifies the extraneous input in 30% of the cases. As noted in the previous paragraph, such over-specification problems are due to the fact that the AIC (or the BIC) criterion penalizes model complexity, and not the number of inputs directly. Table 6.4 shows that both techniques are able to recognize the underlying linear input-output relationship.

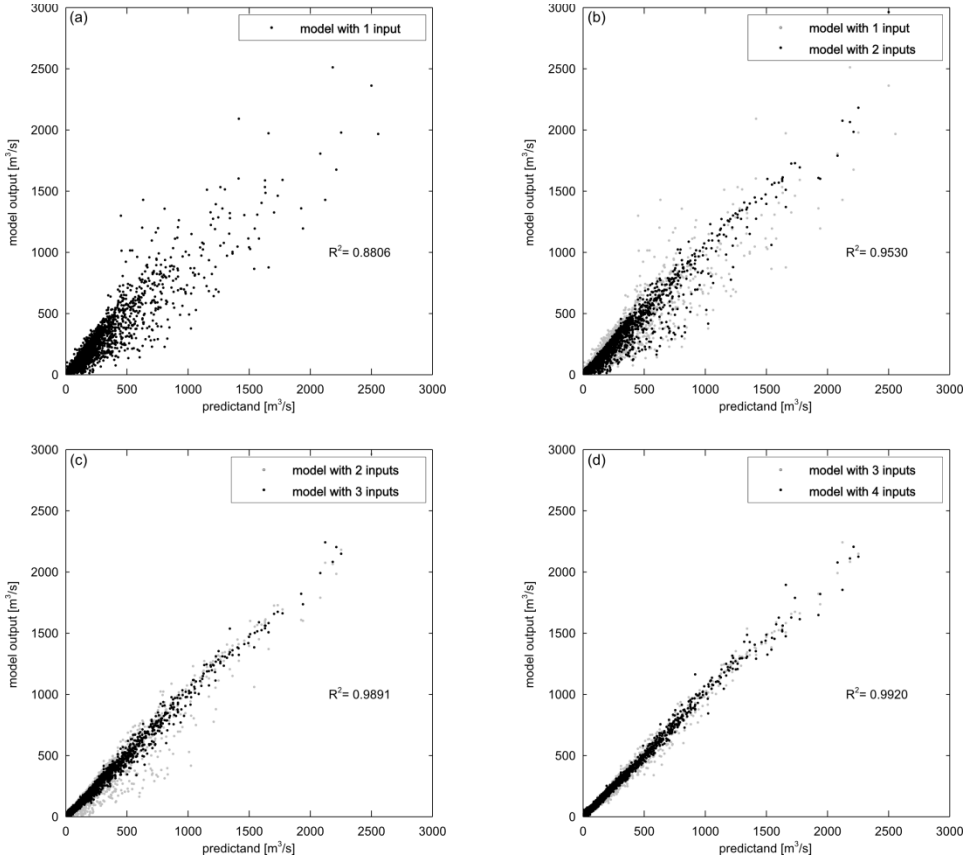


Figure 6.5. Scatter plots of Kentucky River streamflow vs ELM model output for progressively better-specified input subsets.

6.4.2.3.Computational efficiency

An accurate IVS algorithm might be of no practical use if it requires too much time for processing a dataset. This is usually a major drawback of wrapper techniques which explore a very large number of feasible solutions before reaching convergence. The average run-time of the two methods for each dataset is shown in Table 6.5, along with the average number of iterations and the average number of hidden units of the final models. It appears that, thanks to the fast convergence of swarm optimization and to the exceptional reduction in training time provided by the ELM, both methods can be employed in practical IVS applications. With average run-time of around 2 minutes over the synthetic datasets and of 40 minutes for the Kentucky dataset, the BFIPS-ELM is the fastest between the two proposed techniques. These figures go up respectively to 5 minutes and 70 minutes for the MBFIPS-ELM, due to the increased number of iterations needed to reach convergence as well as to larger size of the final models. This was expected since multi-objective problems are generally more complex and require more iterations to converge, and since the MBFIPS-ELM are developed by minimizing the number of inputs directly, rather than model complexity. The computational times of both techniques can be reduced by limiting the maximum number NH_{max} of hidden neurons of the ELM. Fig. 6.6 show how the wrappers overall mean accuracy scores and average run-time change when reducing NH_{max} to 30, 60 and 120 units. It can be seen that the performances on the whole test-bed are basically unchanged for the BFIPS-ELM, unless the minimum value of NH_{max} is considered. Most importantly, these similar performances are obtained at much lower run-times, which are found to vary linearly with NH_{max}. Lower computational requirements are also witnessed for the MBFIPS-

ELM, albeit the average performances decrease much quicker for this wrapper. Fig. 6.7 shows the details for the Kentucky dataset. It can be seen that what happens for the overall accuracy is reflected on this dataset, with the MBFIPS-ELM underperforming the BFIPS-ELM for all cases apart from $NH_{max} = 250$, where it reaches the highest SA score.

TABLE 6.5. AVERAGES AND STANDARD DEVIATIONS OF NUMBER OF HIDDEN UNITS, NUMBER OF ITERATIONS AND RUN-TIMES.

Dataset	*Average num. of hidden units*		*Average num. of iterations*		*Run-time [sec]*	
	BFIPS-ELM	*MBFIPS-ELM*	*BFIPS-ELM*	*MBFIPS-ELM*	*BFIPS-ELM*	*MBFIPS-ELM*
1. AR1	6.7 ± 6.7	98.8 ± 67.7	140.5 ± 31.3	190.6 ± 74.0	99.8 ± 20.6	277.7 ± 147.2
2. AR9_500	8.4 ± 4.9	136.9 ± 87.4	128.7 ± 34.6	292.2 ± 95.8	92.8 ± 19.9	438.4 ± 178.0
3. AR9_70	7.4 ± 2.5	79.0 ± 74.8	138.1 ± 25.9	304.8 ± 91.8	39.0 ± 7.3	155.1 ± 58.7
4. TAR1	13.3 ± 8.6	108.0 ± 79.0	134.9 ± 35.7	175.8 ± 56.6	106.4 ± 31.5	266.6 ± 105.7
5. TAR2	32.0 ± 35.7	113.0 ± 83.7	138.0 ± 39.5	187.1 ± 57.5	109.7 ± 35.8	267.7 ± 108.8
6. NL_500	219.6 ± 27.8	205.5 ± 35.1	146.2 ± 38.5	215.9 ± 52.5	309.6 ± 78.0	393.9 ± 121.9
7. NL_70	183.6 ± 57.2	137.9 ± 60.0	136.5 ± 27.4	188.7 ± 41.2	92.4 ± 19.4	109.0 ± 29.5
8. NL2	87.5 ± 61.1	121.9 ± 66.6	134.7 ± 36.7	201.1 ± 62.5	200.8 ± 49.9	312.4 ± 130.2
9. Bank_fm	23.3 ± 10.3	133.8 ± 71.8	151.3 ± 37.3	366.2 ± 97.3	124.8 ± 29.8	472.9 ± 171.4
10. Bank_fh	20.0 ± 8.9	136.1 ± 76.2	142.1 ± 36.8	409.2 ± 140.3	106.7 ± 23.2	567.0 ± 239.3
11. Bank_nm	69.5 ± 68.3	167.8 ± 70.2	155.2 ± 37.7	424.2 ± 84.8	135.7 ± 33.7	644.1 ± 188.6
12. Bank_nh	29.2 ± 35.5	138.1 ± 75.4	147.9 ± 31.9	483.4 ± 116.2	121.2 ± 25.3	727.7 ± 228.1
13. Friedman_c0_10_m	185.7 ± 51.5	172.9 ± 65.0	144.6 ± 28.0	239.4 ± 50.1	150.9 ± 36.8	238.0 ± 64.6
14. Friedman_c0_10_h	102.9 ± 67.8	157.2 ± 61.7	137.2 ± 37.4	242.5 ± 48.2	93.2 ± 31.7	246.5 ± 65.2
15. Friedman_c0_50_m	43.9 ± 31.7	169.0 ± 79.2	149.9 ± 40.2	513.6 ± 135.0	112.6 ± 27.1	534.3 ± 184.1
16. Friedman_c0_50_h	61.4 ± 35.6	139.5 ± 92.1	141.2 ± 29.3	730.0 ± 167.6	106.7 ± 20.5	797.4 ± 236.2
17. Friedman_c25_10_m	112.4 ± 49.0	128.9 ± 68.8	122.0 ± 29.5	228.6 ± 56.6	126.5 ± 31.4	228.5 ± 73.4
18. Friedman_c25_10_h	24.2 ± 18.2	111.1 ± 72.0	122.3 ± 37.1	176.4 ± 47.2	90.2 ± 23.7	162.1 ± 72.3
25. Kentucky	222.6 ± 28.4	205.4 ± 43.3	145.2 ± 45.3	258.9 ± 71.5	2458.9 ± 874.1	4168.1 ± 1655.5
26. Miller	117.0 ± 89.5	157.9 ± 85.6	110.6 ± 25.0	166.8 ± 37.2	104.5 ± 25.7	158.2 ± 37.4

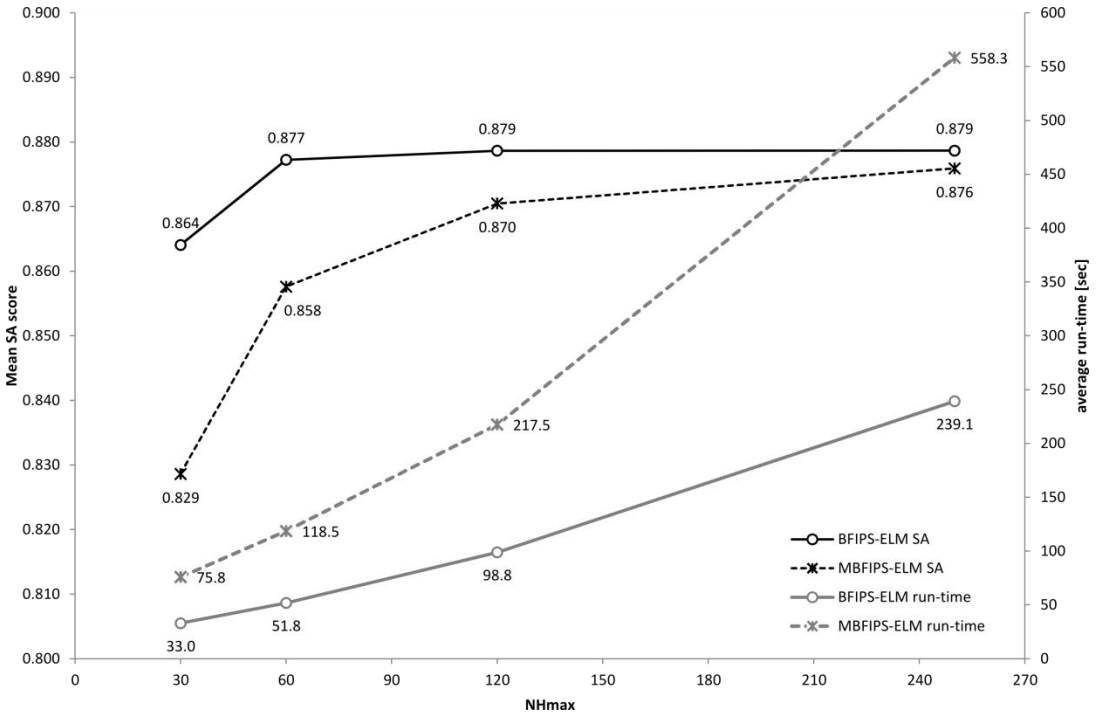


Figure 6.6. Overall mean selection accuracy for different values of NH_{max}.

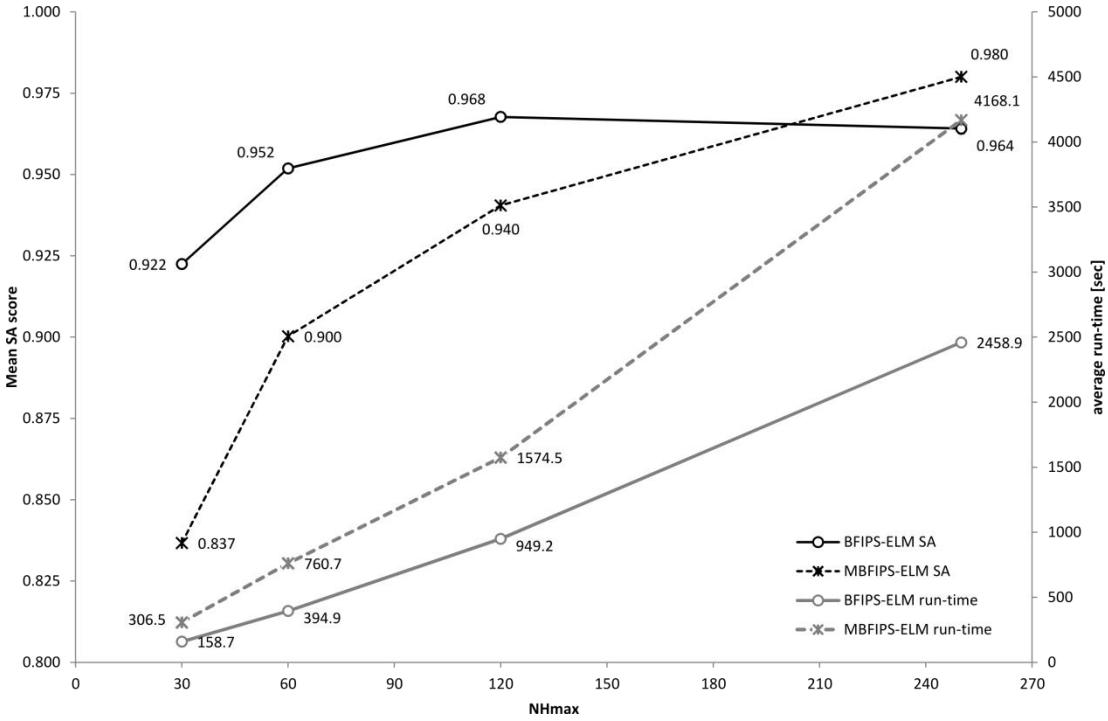


Figure 6.7. Mean selection accuracy in the Kentucky dataset for different values of NH_{max}.

6.4.2.4.Comparison with other IVS techniques

A more thorough evaluation of the effectiveness of the BFIPS-ELM and MBFIPS-ELM can be done by comparing their performances against those of the 4 IVS techniques tested in the original work of Galelli et el. (2014). The reader is referred to their paper and references therein for further information on these techniques and their implementation. The mean SA scores reported in Table 6.6 were retrieved from the website (http://ivs4em.deib.polimi.it) included in the IVS framework. These figures can be reliably compared against those in this study since they are computed using the same value of 0.7 for γ. On the other hand, the analysis regarding the average run-times is only approximative since the experiments were carried out on different software environments and hardware. From a comparison with Table 6.2-6.3, it can be seen that the proposed wrappers generally provide higher accuracy than these methods, with increases of overall SA that goes from a minimum of 11.1% to a maximum of 15.8% for the mean scores, and from 13.4% to 20.6% for the median scores. These improvements tend to occur when the dataset is characterized by high nonlinearities, non-Gaussian output, reduced number of observations, and inter-dependency of the input variables (see Table 6.1). Although each technique has its merits and drawbacks, the measure of these improvements suggests that BFIPS-ELM and MBFIPS-ELM are very good candidates for carrying out the IVS task, especially if one considers the high accuracy and reduced computational costs of the ELM-based wrappers obtained with lower NH_{max}, as shown in Fig. 6.6-6.7. Furthermore, even with all the caution due to the different implementations, the proposed techniques appear to be much faster than the GA-ANN wrapper. Indeed, the average 30 hours of run-time needed by the GA-ANN to process

the Kentucky dataset drop to 70 minutes for the largest MBFIPS-ELM and to less than 3 minutes for the BFIPS-ELM with NH_{max} = 30, which still outperforms the GA-ANN with a SA of 0.922. The speedups provided by ELM training over backpropagation are even more significant if one considers that this particular GA-ANN was built around an ANN with one single hidden-neuron. Similarly, from the good results of smaller ELMs on the Kentucky dataset (Fig. 6.7) it appears that the PMIS and IIS might lose their computational speed advantage for long datasets such as those typically employed in data-driven streamflow forecasting applications. With the exception of the PCIS that achieves high accuracy in very short computational times for the Kentucky dataset, there are strong indications that the proposed methods are superior in carrying out the IVS task for NNRF applications. In addition, if one considers the poor results on the NL benchmarks, it is likely that PCIS accuracy could substantially decrease for applications in highly nonlinear watersheds, such as those of arid and semi-arid regions, mountainous regions, and areas with high climate variability (Borga et al., 2007; Ye et al., 1997). These nonlinearities are particularly severe during storm events, thus IVS techniques based on linear correlation measures might not be suitable for the development of flood forecasting and warning systems. On the other hand, the extrapolation of the benchmark results suggests that employing the ELM-based wrappers should provide additional benefits for applications in poorly-gauged watershed, characterized by short datasets with higher noise. Similarly, the remarkable performances on the Miller benchmark suggest that the proposed techniques might offer consistent advantages when processing input variables which are truly informative only when considered together. This latter case is particularly relevant for those applications where hydro-meteorological

parameters are used to estimate evapotranspiration in order to model the evolution of soil moisture content affecting runoff generation (Zanetti et al., 2007).

TABLE 6.6. MEAN SA SCORES AND AVERAGE RUN-TIME OF PMIS, IIS, PCIS AND GA-ANN (TAKEN FROM GALELLI ET EL., 2014)

Datasets	*Mean selection accuracy SA*				*Average run-time [sec]*			
	PMIS	*IIS*	*PCIS*	*GA-ANN*	*PMIS*	*IIS*	*PCIS*	*GA-ANN*
1. AR1	0.999	0.994	0.999	1.000	16.80 ± 2.87	9.49 ± 2.70	0.16 ± 0.05	1491.2 ± 560.2
2. AR9_500	0.999	0.985	0.998	1.000	38.84 ± 3.29	26.73 ± 6.64	0.38 ± 0.13	1973.3 ± 864.3
3. AR9_70	0.823	0.696	0.937	0.851	2.22 ± 0.43	3.39 ± 0.91	0.38 ± 0.13	378.4 ± 199.7
4. TAR1	1.000	0.992	0.998	1.000	14.02 ± 1.22	11.66 ± 3.92	0.23 ± 0.09	841.7 ± 330.9
5. TAR2	0.999	0.999	0.986	0.953	26.13 ± 3.46	8.08 ± 0.16	0.40 ± 0.27	1630.4 ± 860.6
6. NL_500	0.743	0.983	0.575	0.576	23.41 ± 3.72	24.51 ± 3.39	0.20 ± 0.08	878.6 ± 272.3
7. NL_70	0.743	0.729	0.536	0.520	1.82 ± 0.66	4.93 ± 1.79	0.25 ± 0.25	185.8 ± 117.4
8. NL2	0.681	0.671	0.450	0.558	24.26 ± 4.94	15.66 ± 5.02	0.33 ± 0.15	850.2 ± 467.0
9. Bank_fm	0.484	0.475	0.743	0.752	31.03 ± 6.18	44.41 ± 6.22	1.23 ± 0.36	1544.4 ± 341.9
10. Bank_fh	0.498	0.480	0.589	0.548	35.07 ± 7.45	25.52 ± 4.08	0.99 ± 0.27	1754.1 ± 775.6
11. Bank_nm	0.606	0.681	0.785	0.762	48.33 ± 17.61	41.94 ± 2.12	1.55 ± 0.50	1732.6 ± 288.6
12. Bank_nh	0.504	0.580	0.728	0.619	34.50 ± 12.82	30.59 ± 3.65	1.29 ± 0.37	1667.8 ± 634.6
13. Friedman_c0_10_m	0.995	0.986	0.859	0.860	13.36 ± 1.10	9.64 ± 0.45	0.31 ± 0.07	609.9 ± 215.7
14. Friedman_c0_10_h	0.865	0.891	0.850	0.851	10.79 ± 1.41	6.68 ± 0.44	0.56 ± 0.58	710.5 ± 332.0
15. Friedman_c0_50_m	0.995	1.000	0.856	0.860	46.61 ± 3.42	60.59 ± 4.85	1.26 ± 0.35	2074.7 ± 564.2
16. Friedman_c0_50_h	0.860	0.888	0.851	0.855	38.72 ± 6.58	57.39 ± 6.45	1.33 ± 0.55	1832.8 ± 593.0
17. Friedman_c25_10_m	0.832	0.720	0.647	0.627	10.31 ± 2.62	10.40 ± 2.47	0.26 ± 0.18	325.5 ± 87.3
18. Friedman_c25_10_h	0.660	0.594	0.577	0.571	7.78 ± 1.87	3.17 ± 0.56	0.22 ± 0.05	303.4 ± 146.6
25. Kentucky	0.650	0.822	0.944	0.915	1860.37 ± 107.22	800.58 ± 92.86	7.65 ± 1.88	106,725.6 ± 27,214.6
26. Miller	0.350	0.618	0.280	0.710	2.65 ± 0.27	0.98 ± 0.55	0.14 ± 0.08	5664.9 ± 1561.8
Overall mean SA	0.783	0.776	0.818	0.806				
Overall median SA	0.764	0.789	0.759	0.769				

6.4.3. Qualitative assessment of the proposed wrappers

The qualitative criteria proposed in the IVS framework and briefly described in Section 6.3.3 are employed to perform a final step of evaluation on the BFIPS-ELM and MBFIPS-ELM techniques.

Ease of use and robustness. The proposed techniques require the specifications of several parameters that 1) govern the swarm optimization process, 2) determine the binary encoding of the wrappers, and 3) define the optimization function used or the selection of the final models. However, the optimal choices for most of these parameters have already been investigated in previous studies or have been examined in the detail in this one. For instance, employing 30 particles for the swarm has been a standard for most applications regarding the use of PSO, and as mentioned earlier in text, the use of von Neumann topologies has been advocated in several studies. We suggest setting the maximum number of particles in the MBFIPS Pareto-front as the number of input candidates so that all of them could be selected in case they are all relevant. Although a thorough investigation on the effect of mutation has not been carried out in this study, the values reported for the mutation parameters seemed to be most appropriate after some preliminary runs (results not shown here). The parameters l_{inputs} and l_{act} defining encoding scheme are directly determined by the size of the candidate pool and the number of different activation function that one wants to test. As emerges from Section 6.4.2.3, appropriate values of NH_{max} can be set to around 100 for the BFIPS-ELM and 250 for the MBFIPS-ELM. Run-time could be shortened at the expense of slightly lower performances by reducing the maximum number of hidden units to around 50 for the BFIPS-ELM. On the other hand, values of NH_{max} below 100 should not be employed for the MBFIPS-ELM. Is important to allow the ridge regression constant λ to take a wide variety of possible values, therefore we suggest to use 6 or 7 bits for l_{λ}. More bits could be assigned for the coding of λ, however an alternative method to the power-of-two sequence employed here will be needed to fully exploit the denser discretization. As

shown in the previous sections, the best results for the BFIPS-ELM were obtained using the AIC criterion with a logarithmic penalty. The use of this objective function is therefore suggested for the development of ELM-based wrappers. A square root penalty could be preferred if more importance is given to the exclusion of extraneous inputs. On the other hand, the RMSE might be preferred if one wants to avoid under-specification of meaningful inputs. In the same way, the value of ρ could be changed to obtain similar effects for the MBFIPS-ELM case, taking into account that ρ=1% provided the best tradeoff for the test-bed employed in this study.

Explanation capability. While filters technique such as the PMIS, IIS and PCIS directly provide information about the relative importance of each selected input, wrapper methods usually require post-processing analysis for such information to be retrieved. Although the BFIPS-ELM is no exception to this rule, the set of Pareto-particles returned by MBFIPS-ELM could be extremely informative with respect to the relative importance of the selected inputs. As a consequence of the optimization approach employed for the MBFIPS-ELM, the variable defining the subset for the Pareto solution with *cardinality* = 1 is overall the most relevant. In the same way, the variable added to form the subset of the solution with *cardinality* = 2 is the second most informative. Similar steps can be performed until the actual final solution selected with the criterion in (6.4) is considered, and the last variable added is regarded the least informative. For this procedure to be fully informative the Pareto-front has to be consistent, meaning that variables included in subsets with lower cardinalities should always be part of those with greater cardinalities. Although the analysis of relative variable importance for the fully synthetic datasets is beyond the scope of this study, an

in-depth analysis is carried out for the Kentucky dataset. The input selected at each cardinality for the 30 replicates of the Kentucky dataset are shown in the *selection matrix* of Fig. 6.8. Darker colors in the matrix identify inputs which have been selected at lower cardinalities and always retained at higher cardinalities. The selection matrix shows that all the returned Pareto-fronts are consistent, even when the MBFIPS-ELM under-specifies (row 9, 25 and 30) or over-specifies (row 6, 11, 17, and 18) the optimal subset. Furthermore, there is a remarkable uniformity across the replicates, since Q_{t-1} is always selected at cardinality 1, Q_{t-2} at cardinality 2, ER_t at cardinality 3, and lastly ER_{t-1} at cardinality 4 or 5 when the Pareto-front does not include a solution with 4 inputs (row 11). It is also interesting to note that all the under-specifications concern this latter input, which is the least relevant.

Flexibility. The optimization algorithms and reference model of the proposed wrappers can be easily interchanged with other suitable alternatives, provided the binary encoding is modified accordingly in the latter case. However, it is important to underline the intrinsic flexibility of the ELM-wrappers presented in this study, where models with different complexity and different types of activation functions are concurrently developed to match the underlying relationship in the datasets. In addition, the convenient formulation of the BFIPS allows for a straightforward implementation of its multi-objective generalization.

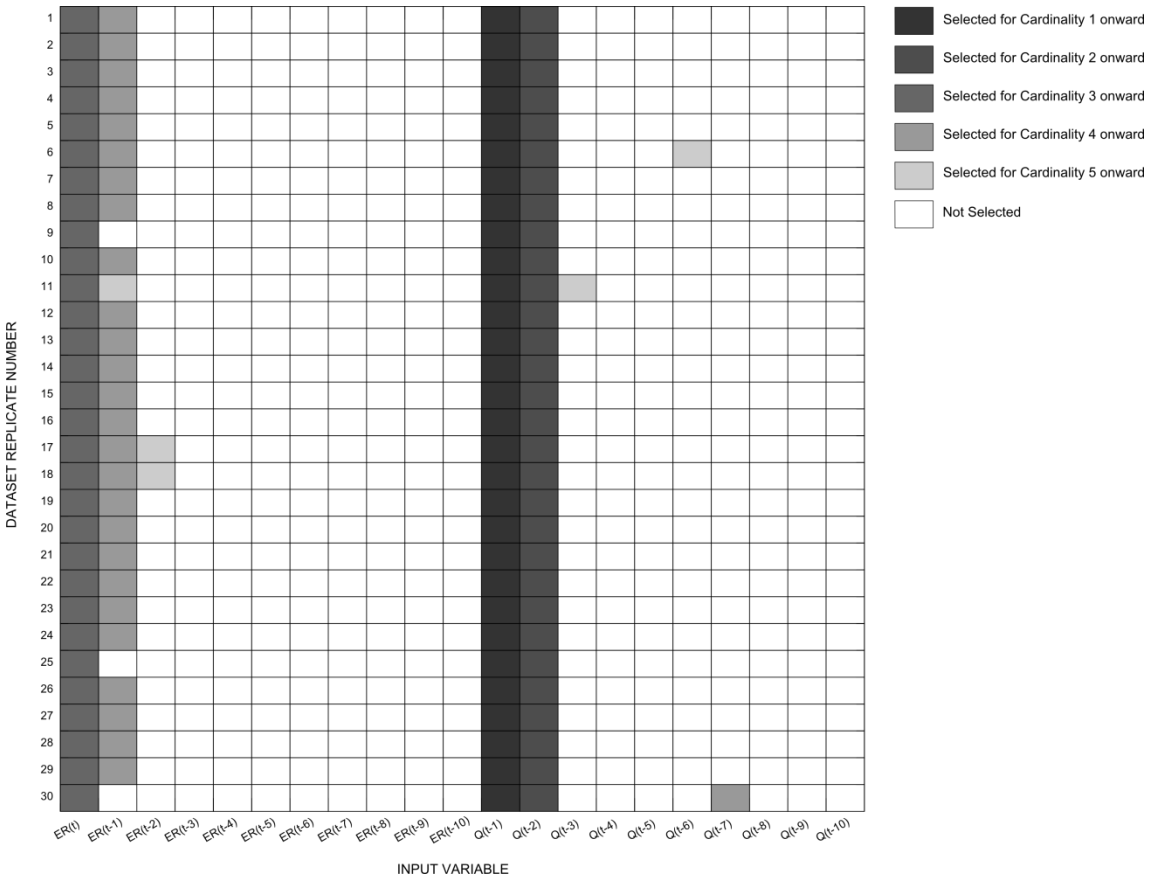


Figure 6.8. Selection matrix for the Kentucky River dataset.

6.5. Conclusions

Fast and accurate wrapper IVS techniques for application in rainfall-runoff data-driven modeling were developed using ELM and binary-coded discrete swarm optimization. The effectiveness of the proposed methods was assessed using the criteria and datasets of a comprehensive IVS evaluation framework, and compared with that of 4 existing IVS methods. The results obtained showed that the proposed wrapper techniques provided overall best performances at run-times which are comparable with those of some fast model-free approaches, such as PMIS and IIS. The best performers were found to be 1) the MBFIPS-ELM algorithm developed based on the concurrent

minimization of an error function and the number of selected inputs, and 2) the BFIPS-ELM algorithm based on the minimization of the AIC variant with logarithmic complexity penalty. The first technique was arguably the most accurate overall, and was able to reach an almost perfect specification of the optimal input subset on a partially synthetic rainfall-runoff experiment. These high performances are obtained at the expense of longer run-times, since the MBFIPS-ELM requires larger underlying ELM models. However, the analysis of the returned Pareto-fronts allows for the determination of the relative importance of the selected inputs, which is usually unattainable with other wrapper methods such as the BFIPS-ELM. On the other hand, the latter technique is found to consistently reach high accuracy scores while being considerably faster as it performs well even with small ELM models. Further studies should verify whether these techniques could provide additional benefits when employed in real-world applications characterized by high nonlinearity in the catchment dynamics, as suggested by the extrapolation of the results obtained on the synthetic datasets. Improved performances over other IVS techniques should also be expected for applications in poorly-gauged watershed or when there is significant inter-dependence among the input variables forming the candidate pool.

7. Neural Network River Forecasting through baseflow separation and binary-coded swarm optimization

The inclusion of expert knowledge in NNRF applications is expected to yield better performances in modeling and more reliable estimates of river quantities. Modular techniques designed to work on different flow regimes and hydrological conditions are preferred ways to incorporate such hydrological knowledge in data-driven models. Previous studies have suggested that more accurate prediction of total streamflow could be achieved through modular ANNs trained to perform an implicit baseflow separation. These models fit separately the BaseFlow (BF) and Excess Flow (EF) components as obtained by a digital filter, and reconstruct the Total Flow (TF) by adding these two signals at the output. The optimization of the filter parameters and ANN architectures is carried out through global search techniques that minimize a weighted function of the errors of the TF, BF, and EF components. Despite the favorable premises, the real effectiveness of these modular models (MM) has been tested only on a few case studies, and the quality of the baseflow separation performed by this technique has never been thoroughly assessed. In this study, we compare the performance of MM against global models (GM) for nine different gaging stations in the northern United States. Binary-coded swarm optimization is employed for the identification of filter parameters and model structure, while ELMs, instead of ANN, are used to drastically reduce the large computational times required to perform the experiments. The results show that there is no evidence that MMs outperform global GMs for predicting the TF. In addition, the baseflow produced by the MM largely underestimates the actual baseflow component expected for most of the considered gages. This occurs because the values of the filter

parameters maximizing overall accuracy do not reflect the geological characteristics of the river basins. The results indeed show that setting the filter parameters according to expert knowledge results in accurate baseflow separation but lower accuracy of TF predictions, suggesting that these two objectives are intrinsically conflicting rather than compatible.

7.1. Introduction

The black-box nature of NNRF models is a target for widespread criticism, and likely the major reason why many hydrologists advise against their use in real-world problems in favor of physically sound conceptual models. To overcome this issue, recent efforts have been made by the research community to explain the internal workings of NNRF models, and link the processes taking place within the network to the processes in the watershed (Fernando and Shamseldin, 2009; Jain and Kumar, 2009; Jain et al., 2004; Wilby et al., 2003). Others studies have focused on the incorporation of expert knowledge into data-driven models in order to improve their hydrological plausibility and overall modeling performances with respect to those of a single global model (GM). A preferred track is represented by the use of modular models (MM) designed to work on different flow regimes, specific parts of the hydrograph, as well as different hydrological conditions. Zhang and Govindaraju (2000) employed Bayesian concepts and a committee of three specialized ANN responsible for the forecasting of low-, medium-, and high-runoff events for three medium watersheds in Kansas, US. An additional gating-network was delegated to weigh the output of these three expert modules depending on the hydro-meteorological conditions. A similar approach was

later devised by Parasuraman et al. (2006), where only one of the ANN modules was selected at a time by an embedded spiking layer developed with unsupervised learning, i.e. either trained by competitive learning or represented by a Self-Organizing Map (SOM). Another example combining unsupervised learning and MM was proposed by Toth (2009) for the Sieve watershed, Italy. Clustering of data with similar hydrological and meteorological conditions was first performed using SOM, and an ANN was subsequently trained for each cluster. Clusters' association obtained by exploiting SOM indications on the similarity between classes later resulted in remarkable performance improvements of the MM over the GM used for comparison.

Jain and Srinivasulu (2006) proposed an integrated approach where the flow hydrograph was decomposed into different segments based on physical concepts in a catchment, and each segment was modeled using different ANN and/or conceptual techniques. Application of this hybrid MM approach to the Kentucky River, US, catchment showed that it was able to outperform the GM made of a single ANN. Further improvements were later obtained for the same case study with a similar approach that employed Genetic Algorithms (GA) for training the ANN modules of the integrated system (Srinivasulu and Jain, 2009). Corzo and Solomatine (2007a, 2007b, 2006) proposed a more process-based approach where the BaseFlow (BF) and Excess Flow (EF) components of the Total Flow (TF) are first separated using the Eckhardt digital filter (Eckhardt, 2005), and then modeled separately by two ANNs. This Baseflow Separation-based MM (BS-MM) reconstructs the TF by summing the output produced by the two modules trained to fit the BF and EF components through the LM method. The unknown filter parameters are optimized along with the number of hidden neurons

of the two ANNs using GA and Generalized Pattern Search (GPS) to minimize a weighted sum of the errors of the TF, BF and EF predictions. Although this technique was reported to outperform the GM and other modular techniques on three different watersheds, the study did not thoroughly assess how well the BF and EF sub-processes were actually reproduced by the two modules. Indeed, the physical plausibility of this approach would be greatly enhanced if the two specialized modules were found to correctly fit the BF and EF components, making the BS-MM a strong candidate for real-world operational purposes.

To our knowledge, these important propositions have not been verified and, hence, filling this gap remains a largely unexplored area of research. This provides the motivation for the present study to test the BS-MM approach for separation of BF and EF. To improve the statistical significance of our study, we test the BS-MM nine watersheds with porous aquifers sited in the northern United States. Reliable estimates of the BaseFlow Index (BFI) and the optimal Eckhardt filter parameters are already available for each employed gage (Eckhardt, 2008). The availability of these values allows verifying whether the implicit baseflow separation performed by the modular approach is consistent with expert findings, as well as check if using the recommended values for the filter parameters will yield improved performances. The BS-MM approach will also be tested against the GM built for each of the nine gages, so as to provide a more statistically-robust comparison of these two different approaches. Due to the large computational costs required to develop the models for each case study, the Extreme Learning Machine (ELM) paradigm is used instead gradient-based ANN training. In addition, the BFIPS algorithm is employed to perform the search of optimal

model structures and values of the filter parameters, while the automatic selection of the optimal set of inputs is done using the MBFIPS as shown in Chapter 6.

The rest of this Chapter is organized as follows. Section 7.2 presents the necessary background information on the BS-MM technique and offers a detailed description of the models used for the experiments. Section 7.3 describes the employed case studies and the experimental setup. The results of the experiments are reported and discussed in section 7.4. Conclusions are drawn in Section 7.5.

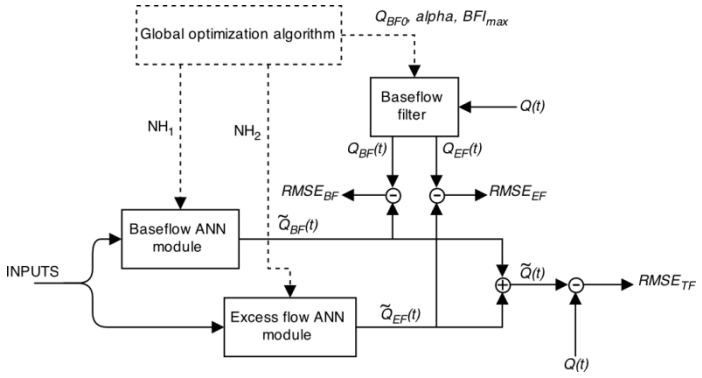


Figure 7.1. The BS-MM model of Corzo and Solomatine.

7.2. Baseflow Separation-based Modular Models (BS-MM)

7.2.1. Original BS-MM

The BS-MM proposed by Corzo and Solomatine (2007a, 2007b, 2006) is displayed in Fig. 7.1. The Eckhardt baseflow filter produces the baseflow $Q_{BF}(t)$ and excess flow $Q_{EF}(t)$ components from the observed total flow $Q(t)$. These signals are used as targets for the training of the two specialized ANN modules, whose outputs $\tilde{Q}_{BF}(t)$ and $\tilde{Q}_{EF}(t)$ are added up to give the estimation of the total flow $\tilde{Q}(t)$. Since the

base- and excess-flow components are initially unknown, the most relevant inputs are selected with respect to the observed TF using correlation and mutual information analysis. A GO algorithm, the GA and GPS in their original works, is employed to optimize the number of hidden neurons of both ANN modules NH_1 and NH_2, as well as the values of the filter parameters. These are: 1) the initial baseflow value Q_{BF0}; 2) the recession constant a; and 3) the maximum value BFI_{max} of the baseflow index *BFI*, which is the long-term ratio of baseflow to total streamflow. The value of $Q_{BF}(t)$ given by the Eckhardt filter can be thus written as

$$Q_{BF}(t) = \frac{(1 - BFI_{max})aQ_{BF}(t-1) + (1-a)BFI_{max}Q(t)}{1 - aBFI_{max}} \qquad (7.1)$$

subject to $Q_{BF}(t) \leq Q(t)$. The value of $Q_{EF}(t)$ is obtained by subtracting the results of (7.1) from $Q(t)$. The parameters composing the filter are supposed to be unknown, and the GO algorithm finds them along with NH_1 and NH_2 by minimizing a weighted sum of the Root Mean Square Errors (*RMSE*) computed for the total flow $RMSE_{TF}$, and for the predictions of two branches of the modular model $RMSE_{BF}$ and $RMSE_{EF}$, respectively. Since the BS-MM technique was originally devised for operational purposes in flood prediction, Corzo and Solomatine (2007a, 2007b) reasonably assigned greater weights to the overall and excess flow terms, resulting in the objective function in (7.2) for the GO algorithm

$$E_T = 0.6RMSE_{TF} + 0.3RMSE_{EF} + 0.1RMSE_{BF} \qquad (7.2)$$

Although the three weights could be also subject to optimization, the same E_T objective function is also employed in the present study.

7.2.2. Modifications to the BS-MM

This work features some modifications to the original BS-MM, mostly in order to increase its speed and automation, which would facilitate the analysis across several case studies. In the first place, ELMs are used, instead of ANNs, for both GM and MM model development. As reported in Section 2.2, this recently introduced class of neural models is known to be more accurate than ANN while at the same time providing remarkable speedups during the training process. Secondly, the BFIPS algorithm is employed, instead of GA and GPS, for the determination of optimal ELM properties and values of the filter parameters. However, the value of the recession constant a is not searched during the optimization process but determined beforehand through recession analysis, which can be easily performed and provides more reliable results (Eckhardt, 2012, 2008; Li et al., 2014). The selection of optimal model predictors is automatized using a wrapper based on the MBFIPS algorithm, which according to the results presented in Chapter 6 has been found to outperform other input variable selection techniques for NNRF applications.

7.2.3. Models employed

The analysis presented in this study concerns three different model typologies. These are: 1) an overall global model built using a single ELM (GM); 2) a BS-MM derived from the original model (Corzo and Solomatine, 2007a, 2007b) after the modifications described in the previous section have been applied (BS-MM1); and 3) a variant of BS-MM1, where all the filter parameters are determined beforehand according to expert-knowledge (BS-MM2). The analysis on the latter model allows checking whether the inclusion of expert-knowledge in BS-MM development results in

better discharge predictions independently from failures in the filter parameters' optimization process.

GM. This model consists of a single ELM, as shown in Fig. 7.2. The MBFIPS is used to determine the optimal subset of inputs, as well as the optimal number of hidden neurons NH of the ELM model, and the value of the ridge regression constant λ. The value of the constant ρ used to select the final model is set to 1%.

BS-MM1. The modified version of the BS-MM is displayed in Fig. 7.3. The inputs fed to the two ELM modules are those obtained for the GM using the MBFIPS. The BFIPS is employed to search for the optimal number of hidden neurons NH_1 and NH_2 of the two ELMs, the respective ridge regression constants λ_1 and λ_2, as well as the values of the baseflow filter parameters BFI_{max} and Q_{BF0}. As mentioned before, the recession constant a will be instead determined through recession analysis. The BFIPS performs the optimization by minimizing the weighted error function in (7.2).

BS-MM2. This model (Fig. 7.4) differs from the BS-MM1, as the baseflow separation is performed in advance using fixed filter parameters. In particular, the value of BFI_{max} is assigned based on the geologic characteristics of the watershed, while Q_{BF0} is set equal to the total streamflow discharge observed at the beginning of the time series. This latter simplification is justified, since it is known from sensitivity analysis that Q_{BF0} has little impact on the quality of the baseflow separation performed by the Eckhardt filter, especially for long data series (Eckhardt, 2012). Performing the baseflow

separation in advance allows determining the inputs for the BF and TF modules separately using the MBFIPS-based wrapper as done for the GM. Once this pre-processing step has been completed, the BFIPS is employed to optimize the ELM structure of the final BS-MM2 model by minimizing (7.2).

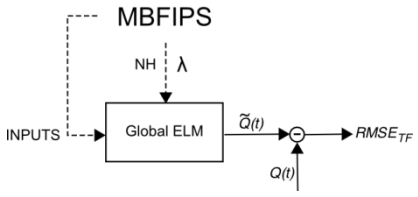


Figure 7.2. The Global Model (GM).

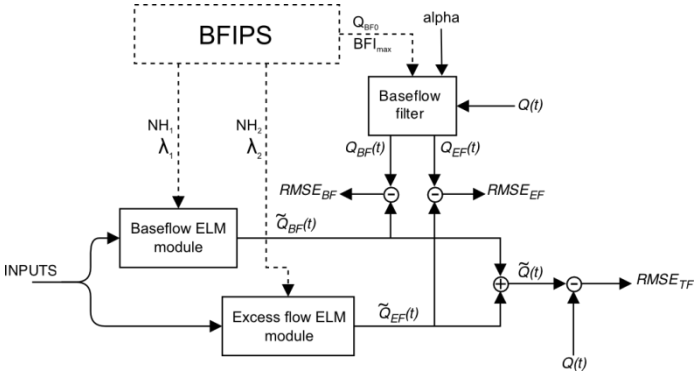


Figure 7.3. The BS-MM1 model.

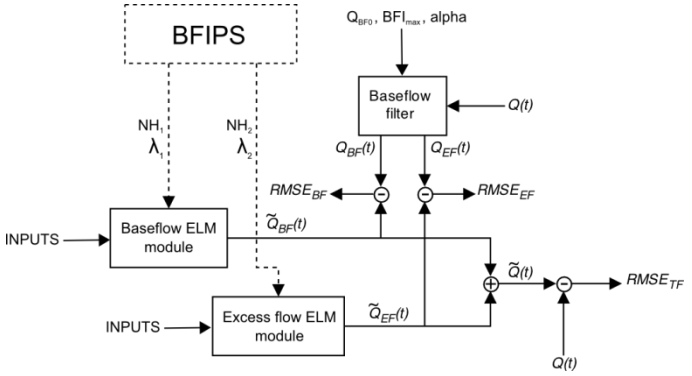


Figure 7.4. The BS-MM2 model.

7.3. Experimental setup

7.3.1. Working datasets

The models presented in the previous section are now tested on nine different small- to medium-sized watersheds in the northern United States. Table 7.1 reports the details of these case studies, including their location, the USGS gaging station ID, the drainage area at the station, the average streamflow in the considered period, as well as the values of the filter parameters a and BFI_{max}. As mentioned before, the recession constant is obtained through recession analysis of the streamflow discharge (Eckhardt, 2008). On the other hand, the reported values of BFI_{max} are those recommended based on empirical results and on the geologic characteristics of the watersheds (Eckhardt, 2012, 2008, 2005). The value of 0.8 is thus used for all the gaging stations, as the cases examined are all relative to perennial streams with porous aquifers. Table 7.1 also shows the BFI estimates as computed with the Eckhardt filter using the reported values of the parameters. These estimates will be later employed along with goodness-of-fit measures to assess the quality of the baseflow separation performed by the BS-MMs. The candidate modelling inputs for each case were obtained from 1 up to a maximum of 3 nearby stations of NOAA's National Climatic Data Center (NCDC). The type of variables available are identified as RAIN (rainfall), SNOW (snowfall), SNWD (snow depth), TMAX (maximum temperature) and TMIN (minimum temperature), respectively. The datasets are divided into training, validation and test subsets, accounting for 50%, 25% and 25% of the observations, respectively. The observations are shuffled in order to grant statistical similarity of streamflow across the three subsets, while at the same time including the highest peak in the training dataset to improve

model generalization. Data shuffling requires reconstructing the flow series sequentially in time before applying the baseflow filter. The BF and EF signals obtained from the filter are then reordered to match the initial shuffled arrangement before being used as target data for the ELM modules. Five lagged predictors are employed for both autoregressive (FLOW) and exogenous inputs. In particular, discharges up to 5 days ahead are considered for streamflow observations, while lags from 0 to 4 days ahead are used for the meteorological variables.

7.3.2. Algorithm *setup*

The same setup is employed for each of the nine case studies. Both the BFIPS and the MBFIPS algorithms are run for a maximum of 200 iterations, or stopped earlier if no improvement in the search is witnessed for over 30 consecutive iterations. In both cases, the search is performed using 30 particles arranged to form a von Neumann topology (Fig. 3.4h). The number of maximum particles in the MBFIPS Pareto-frontier is also set to 30, while the bit-flipping mutation probability is set to 5% and a maximum of 5 particles are allowed to undergo mutation at each iteration. In order to reduce the chance of bad optimization minima, 30 different runs of the algorithms are carried out for each case study and each model typology. The optimal models for the BS-MM cases are thus chosen as the ones corresponding to the lowest value of E_T on the validation dataset, while the final GM models are identified on the overall Pareto-front computed from the frontiers of the 30 runs by setting the cutoff parameter ρ = 1%.

7.3.3. Binary particle encoding

Since binary-coded swarm optimization is used to evolve ELM model structures, determine the value of problem-specific parameters and identify the optimal set of

inputs, the encoding scheme described in Section 6.2.1 is employed for particle positions. For those cases where inputs are to be selected using the MBFIPS one bit is used for each potential input, with a value of 1 indicating a selected predictor and 0 otherwise. Seven bits are allocated for the number of hidden neurons of each modular ELM, whose complexity could therefore range between 1 and 128 hidden neurons. On the other hand, 8 bits are allotted for the GM-ELM so that the maximum possible size of the three model topologies would be the same. Six bits are destined for the values of the ridge regression constants λ, which are allowed to vary within a series of increasing powers of 2 between 2^{-31} and 2^{32}. Eight bits (256 steps) are used to code each of the two filter parameters, with the value of Q_{BF0} going from 0 to the maximum value of the observed Q, while BFI_{max} is allowed to vary between 0.25 and 0.8. These values are chosen as they are the suggested extremes for Eckhardt filter implementation, with 0.25 corresponding to the case of perennial stream with rocky aquifer and 0.8 being the recommended value for the case of perennial stream with porous aquifer.

7.3.4. Evaluation metrics

For a thorough assessment of model prediction accuracy, five evaluation metrics are employed: the Root Mean Square Error (RMSE), the Coefficient of Efficiency (CE), the Median Absolute Percentage Error (MdAPE), the Fourth Root Mean Quadrupled Error (R4MS4E), and the Mean Squared Logarithmic Error (MSLE). As reported in APPENDIX B, these 5 metrics evaluate model performances with respect to overall goodness-of-fit of the predictions (RMSE, CE and MdAPE), as well as during low flow (MSLE, and in second place MdAPE) and high flow (R4MS4E) sections of the hydrographs.

TABLE 7.1. CASE STUDIES DETAILS

USGS ID	*Latitude*	*Longitude*	*Drainage Area (sq. km)*	*Mean Discharge (cumecs)*	*Period investigated*	*Total observations (used observations)*	*a*	*BFImax*	*BFI*	*Variables available*
04015475	47°31'38"	92°07'21"	259	2.33	19-09-78 to 30-09-82	1473 (1468)	0.971	0.8	0.69	RAIN, SNOW, SNWD, TMAX, TMIN
04067958	45°23'16"	88°18'18"	1157.7	8.8	01-06-98 to 31-12-13	5693 (4225)	0.977	0.8	0.76	RAIN, SNOW
04069416	45°08'36"	87°48'02"	2641.8	19.37	01-06-98 to 31-12-13	5693 (5000)	0.976	0.8	0.74	RAIN, SNOW, SNWD
04072150	44°32'00"	88°07'47"	279.7	1.76	01-01-01 to 31-12-13	4748 (2068)	0.970	0.8	0.57	RAIN, SNOW, TMAX, TMIN
04085395	44°01'29"	88°07'05"	282.3	1.44	01-07-93 to 30-09-05	4475 (4214)	0.970	0.8	0.71	RAIN, SNOW, SNWD, TMAX, TMIN
04232046	43°06'22"	77°27'43"	71.5	0.37	01-12-87 to 21-02-90	814 (809)	0.967	0.8	0.66	RAIN, SNOW
01333000	42°42'32"	73°11'50"	110.3	2.86	01-01-00 to 31-12-13	5114 (4378)	0.972	0.8	0.69	RAIN, SNOW, TMAX, TMIN
01101000	42°45'10"	70°56'46"	55.2	1.07	01-01-94 to 31-12-13	7305 (4060)	0.974	0.8	0.69	RAIN, SNOW, SNWD, TMAX, TMIN
01176000	42°10'56"	72°15'51"	388.5	7.04	01-01-94 to 31-12-13	7305 (4065)	0.977	0.8	0.75	RAIN, SNOW, SNWD, TMAX, TMIN

TABLE 7.2. PERFORMANCE METRICS FOR TF PREDICTIONS ON THE TEST DATASET

	GM					BS-MM1					BS-MM2				
USGS ID	RMSE	CE	MdAPE	R4MS4E	MSLE	RMSE	CE	MdAPE	R4MS4E	MSLE	RMSE	CE	MdAPE	R4MS4E	MSLE
04015475	0.3658	0.9935	**7.3176**	0.9504	**0.0336**	**0.3615**	**0.9936**	7.7244	**0.9319**	0.0556	0.6855	0.9770	11.7335	2.3141	0.1442
04067958	**0.7372**	**0.9895**	**2.3360**	**1.5086**	**0.0042**	0.8323	0.9866	2.5477	2.0231	0.0043	0.7638	0.9887	2.7890	1.5607	0.0043
04069416	**2.9265**	**0.9633**	7.2930	**5.9245**	0.0183	3.0311	0.9606	7.4488	7.1715	0.0188	3.0759	0.9594	**7.2002**	7.5334	**0.0181**
04072150	**1.7978**	**0.8574**	**41.6663**	**4.5282**	**0.4749**	1.8428	0.8502	67.9088	4.7429	0.7259	1.8428	0.8502	54.8926	4.8468	0.5011
04085395	**0.3939**	**0.9731**	9.2608	1.0862	0.0737	0.3976	0.9726	9.7810	**1.0145**	0.0671	0.4144	0.9702	**8.2725**	1.0160	**0.0564**
04232046	**0.1719**	**0.8797**	**12.4469**	**0.3356**	0.1151	0.1751	0.8753	14.2391	0.3436	0.1037	0.1875	0.8569	15.8467	0.3688	**0.1034**
01333000	**1.4232**	**0.8254**	**11.3982**	**3.3494**	**0.0938**	1.4782	0.8117	13.9758	3.7272	0.1216	1.6167	0.7747	13.3884	4.0191	0.1000
01101000	0.2434	0.9804	**6.4697**	0.8253	0.0493	**0.1629**	**0.9912**	7.7400	**0.3551**	**0.0566**	0.1826	0.9890	8.6668	0.5303	0.0561
01176000	0.8524	0.9856	**4.0211**	1.8848	0.0153	0.8470	0.9858	4.4233	**1.7395**	**0.0125**	**0.8361**	**0.9861**	4.6431	1.8183	0.0176
MEAN	**0.9902**	**0.9386**	**11.3566**	**2.2659**	**0.0976**	1.0143	0.9364	15.0877	2.4499	0.1296	1.0673	0.9280	14.1592	2.6675	0.1112
MEDIAN	**0.7372**	**0.9731**	**7.3176**	**1.5086**	**0.0493**	0.8323	0.9726	7.7400	1.7395	0.0566	0.7638	0.9702	8.6668	1.8183	0.0564

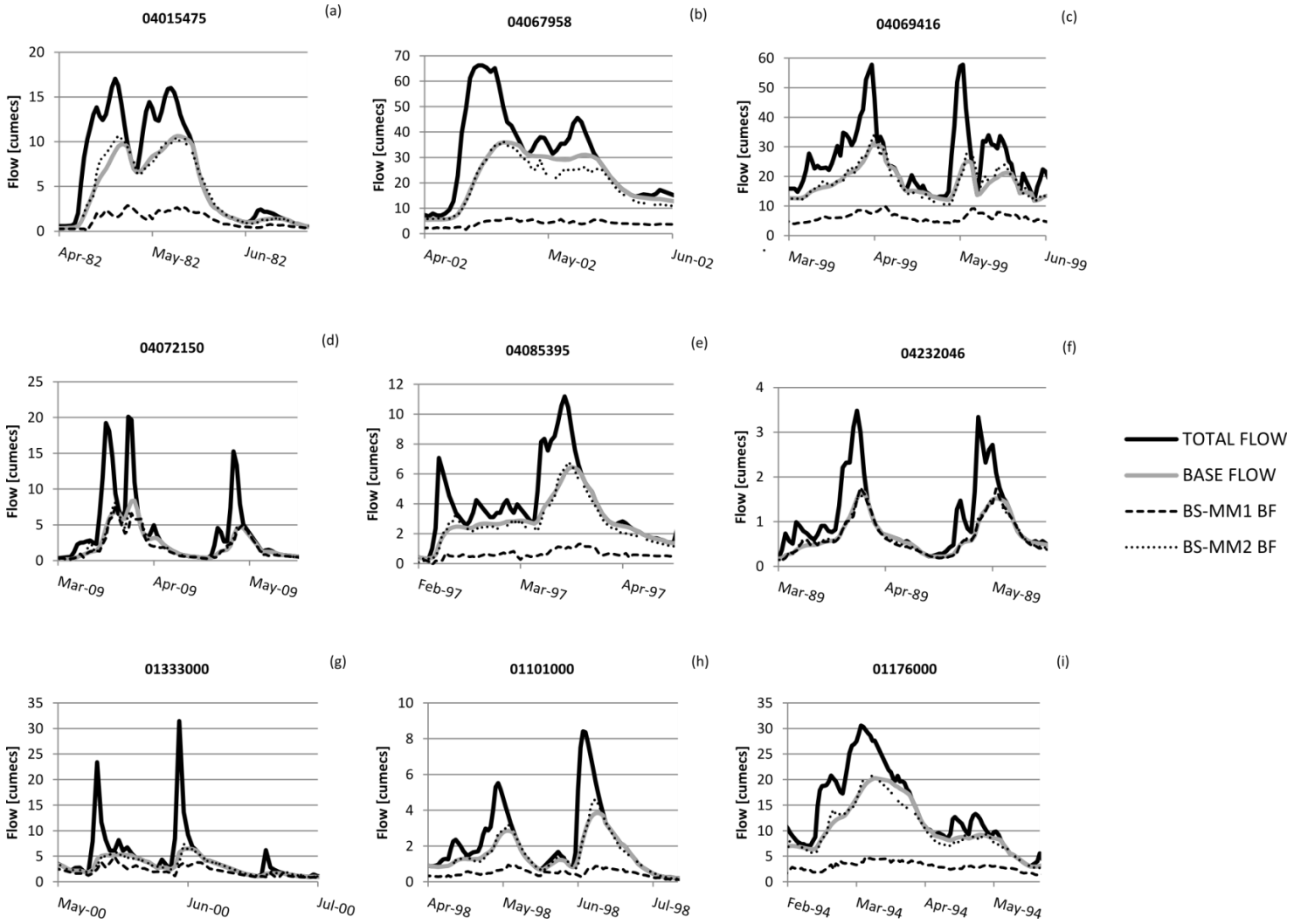


Figure 7.5. Comparison of BF signals produced by modular models for each watershed

7.4. Results and discussion

7.4.1. Total streamflow prediction

Table 7.2 reports the values of the evaluation metrics as computed for the test datasets of each watershed. The results show no evidence that BS-MM approaches outperform the GM for total streamflow prediction. The opposite is more likely to be true, as global solutions show better performances on most of the watersheds employed for the comparison. The GM models are the only ones that concurrently rank the highest with respect to all the metrics for a given watershed, as it happens for the gaging stations #04067958, #04072150 and #01333000. If one considers each metric separately, the GM models score the best on six out of nine cases according to RMSE and CE, seven cases for MdAPE, five cases for R4MS4E, and 4 cases for MSLE. Similar conclusions could be drawn by looking at the mean and median values for the nine watersheds, also reported in Table 7.2. Although these values should be taken with a grain of salt since they are relative to basins in different hydro-meteorological conditions, they strongly suggest that modular solutions perform worse than global ones for TF prediction. This is particularly true for the BS-MM2 models, which, despite being built to perform baseflow separation according to expert-knowledge (BFI_{max} = 0.8), are arguably the worst performers in terms of overall goodness-of-fit of the modeled TF signal. This conclusion seems more likely if one considers that all the streams examined have a strong BF component, which should in theory favor the BS-MM approaches over GM (Corzo and Solomatine, 2007a, 2007b). On the contrary, the GM models appear to have better performances also for the low-flow component of the hydrographs, as indicated

by lower values of MSLE. This is somewhat unexpected, as low-flow is mainly BF which is directly modeled in modular solutions.

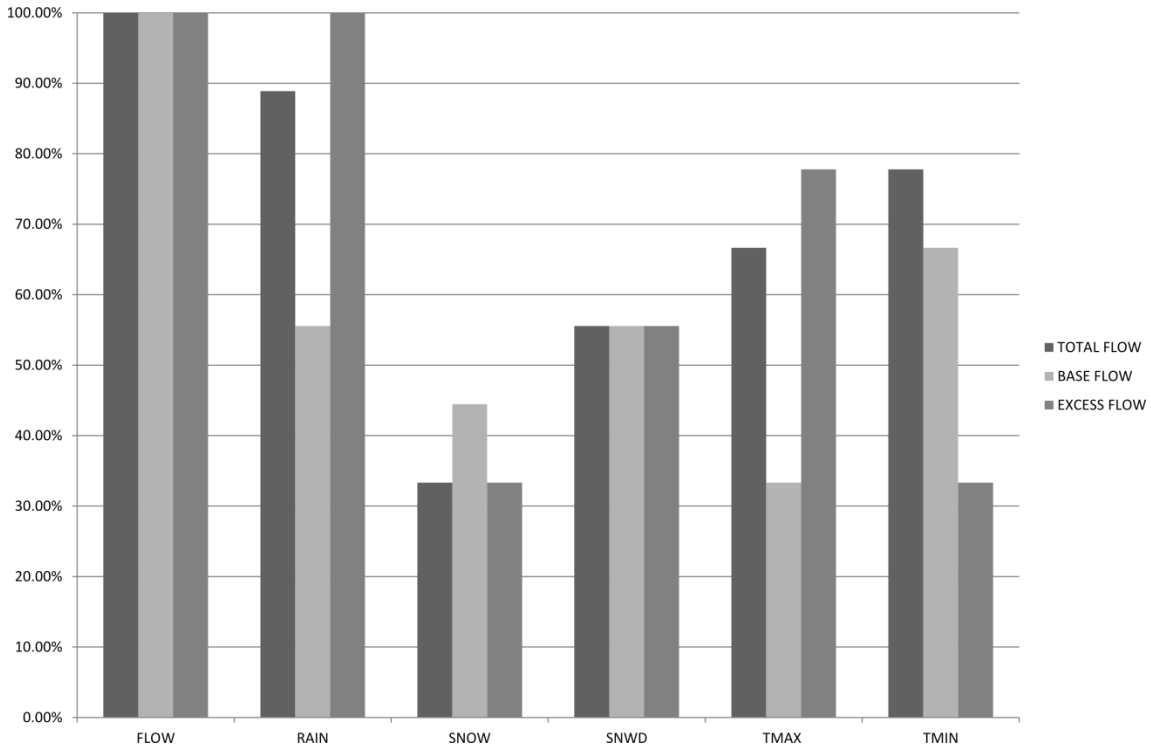


Figure 7.6. Selection frequency for each type of variable with respect to modeled signal.

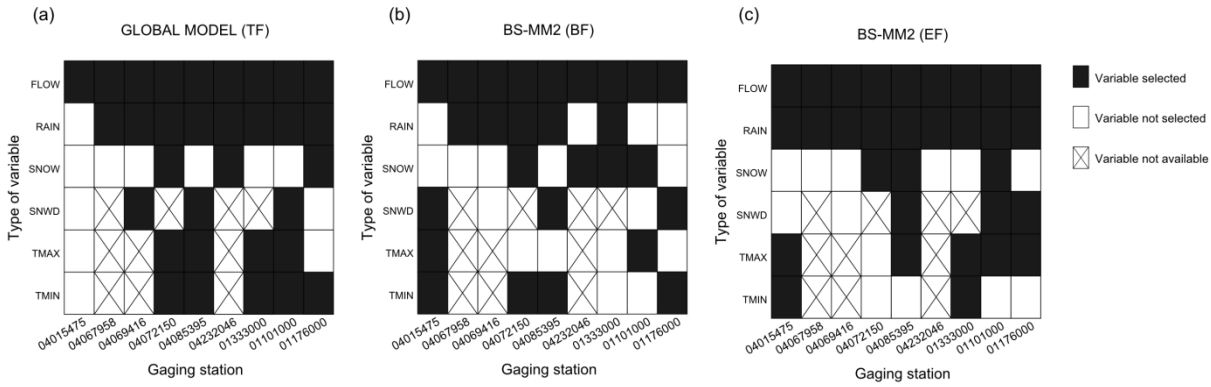


Figure 7.7. Selected variables for each watershed.

7.4.2. Analysis of baseflow separation

Table 7.3 reports the optimal values of Q_{BF0} and BFI_{max} parameters of the BS-MM1 models developed for each stream. It emerges that the values of BFI_{max} returned by the optimization algorithm are substantially smaller than the recommended value of 0.8 for most of the watersheds, and very close to the lower bound of 0.25. Consequently, the BF signals are very different from those returned by the Eckhardt filter with BFI_{max} = 0.8 (BF_{ref}), as shown by the poor values of the CE metric computed using these two time series. In the same way, the BFI estimated for BS-MM1 on the test datasets are generally much lower than the reference values reported in Table I. On the other hand, the estimated BFIs of BS-MM2 are consistent with the values in Table I, suggesting that this approach is more physically-based than BS-MM1. Indeed, although one could question whether 0.8 is the best choice of BFI_{max} for all the analyzed cases, most of the BFIs estimated from BS-MM1 predictions are too low for perennial streams with porous aquifers. The lack of hydrological significance of the BF signals from the BS-MM1 models can be also evaluated graphically, as done in Fig. 7.5, where the BF produced by the two modular solutions are compared with BF_{ref} on part of the dataset of each watershed. The figures show that while the BF signals from the BS-MM2 models closely resemble that of the reference baseflow, the BS-MM1 solutions significantly underestimate BF_{ref} for all the gaging stations except for #04072150 (Fig. 7.5d) and #04232046 (Fig. 7.5f). This behavior may depend on the coefficients of the E_T objective function in (7.2) used to develop the modular models, which weigh more the errors of the EF component than those of the BF component. This may lead the optimization process for BS-MM1 development to maximize EF by reducing BFI_{max}, and minimize

$RMSE_{ET}$ while disregarding the damped BF signal. In these cases, the values of BFI_{max} corresponding to better minima of E_T will not reflect the hydrological processes taking place in the river basins. It is interesting to note that, according to the results in Table 7.2, the BS-MM1 models underperforms GM models for TF predictions even when they produce a well-grounded baseflow separation.

TABLE 7.3. BASEFLOW SEPARATION OF THE BS-MM MODELS

		BS-MM1			BS-MM2	
USGS ID	*QBF0 [cumecs]*	*BFImax*	*Estimated BFI*	*Baseflow CE*	*Estimated BFI*	*Baseflow CE*
4015475	54.00	0.265	0.245	0.0711	0.693	0.9865
4067958	3.40	0.254	0.255	-0.8064	0.758	0.9600
4069416	7.20	0.252	0.253	-0.6367	0.733	0.9620
4072150	11.90	0.738	0.502	0.9064	0.573	0.9576
4085395	21.10	0.252	0.256	0.1566	0.714	0.9807
4232046	1.90	0.794	0.667	0.9789	0.671	0.9708
1333000	40.40	0.592	0.527	0.7152	0.69	0.9515
1101000	0.00	0.256	0.238	0.0643	0.694	0.9704
1176000	4.20	0.25	0.253	-0.1537	0.748	0.9689

7.4.3. Analysis of selected inputs

A final analysis is carried out to check for similarities and differences in the optimal subset of inputs selected for modeling the TF, BF and EF time series. This is done by comparing the predictors selected by the MBFIPS for the GM and BS-MM2 models. For the sake of conciseness, results are presented only for the type of variables rather than for each lagged predictor itself. In particular, Fig. 7.6 shows the frequency with which each type of variable is selected with respect to the total number of times this variable is available across the datasets (see Table 7.1), while the variable selection matrices in Fig. 7.7 present the details for each watershed. As expected, it emerges that past streamflow data (FLOW) and RAIN are the most important inputs of the lot.

FLOW variables are selected for all the watersheds and for both TF and its components. RAIN seems to be less important for modeling the BF, but is still selected in five out of the nine cases, indicating that this variable should also be fed to the BF module, unless proven differently by detailed pre-processing analysis. Indeed, groundwater is displaced by precipitation during a rainfall event and generates baseflow by flowing into the stream. SNOW also seems to carry explanatory power for both TF and its components, however is the least relevant among the meteorological variables. It also appears that, whenever SNWD is available, SNOW predictors are less likely to be selected, suggesting that the former variable carries more information than the latter. This can be observed especially for TF (Fig. 7.7a) and BF (Fig. 7.7b), where for the 5 cases in which both SNWD and SNOW are available, the former is selected in three cases, while the latter only once. It is interesting to note the contrast between TMAX and TMIN for the BF and EF components, with TMAX being substantially more relevant for EF and vice versa. While it can be argued that TMAX is more likely to explain snowmelt, which affects mostly the EF component, the high correlation between these two variables would require further studies to shed light on this matter.

7.5. Conclusions

In this study, we investigated the effectiveness of data-driven Baseflow Separation-based Modular Models (BS-MM) in predicting total streamflow discharge as well as its baseflow and excess flow components. These models estimate the TF by adding the predictions of the BF and EF components. Two different modular solutions were developed and compared against a global model used for reference. The BS-MM1

model consisted of two ELM trained to fit the BF and EF signals produced by a baseflow filter whose parameters were optimized during model development. On the other hand, the BS-MM2 was developed using a filter whose parameters were fixed according to the geological characteristics of the watershed. Contrary to previous studies, experiments run for nine perennial streams in porous aquifers in the northern United States show significant evidence of modular models underperforming global ones for predicting overall streamflow discharge. In addition, the baseflow separation performed by the BS-MM1 was not consistent with the partitioning expected for most of the streams due to substantial underestimation of the baseflow component. The BS-MM2 was found to be more physically grounded, with BFI estimates close to the expected ones and satisfactory reconstruction of the BF signal. However, this model was arguably the worst in predicting overall streamflow discharge, suggesting that this objective and that of performing an accurate baseflow separation might be conflicting rather than compatible. Further studies could probe whether a different formulation of the objective function that weights differently the errors of the TF, BF and EF predictions might solve this issue. Another future line of work could extend the analysis to watersheds with intermittent streamflow, although the difficulties in performing the baseflow separation for this class of streams might hinder the development of the modular models (Aksoy and Wittenberg, 2011).

PART IV. CONCLUSIONS

8. Conclusive remarks and future developments

This thesis has dealt with the use of data-driven techniques for the prediction of water quantities, a field now known also as Neural Network River Forecasting (NNRF). Despite the considerable amount of research on these methods, the existence of several unresolved issues has hindered the adoption of NNRF in operational real-world contexts. The work done in this thesis contributed towards addressing some of these problems by harnessing real- and binary-coded variants of PSO, a flexible and efficient nature-inspired optimization technique which has been only marginally used in data-driven hydrological modelling. In particular, three new PSO variants have been introduced in this manuscript, namely the MOFIPS, BFIPS and MBFIPS optimization algorithms, which have all been derived from the Fully Informed Particle Swarm (FIPS) paradigm. These algorithms have been employed to devise novel applications aimed at investigating the feasibility of

i. developing efficient interval-based NNRF models,
ii. producing a fast and accurate scheme for the automatic selection of optimal inputs and NNRF model structure,
iii. improving NNRF model performances by embedding expert-knowledge to enhance their physical plausibility.

The applications presented in Part III of this manuscript provided concrete solutions to the first two issues highlighted above. In Chapter 5 it was demonstrated that accurate and fast estimation of streamflow prediction intervals can be obtained with the MOFIPS-based LUBE methodology, as shown for the case studies of the Susquehanna and Nehalem rivers. Improvements over the original PSO-based LUBE models were

gained using the bi-objective optimization paradigm illustrated in Chapter 4 for deterministic point predictions.

On the other hand, the BFIPS- and MBFIPS-ELM wrappers presented in Chapter 6 are two examples of successful automatic input and model structure selection techniques. The underlying ELM models grant these methods enhanced speed and great flexibility, suggesting that ELMs are ideal candidates for NNRF applications and should be taken into account by the research community. However, their partially-randomized and extremely distributed nature severely compromise any effort of tracing back the internal workings of the ELM to the processes occurring in the watersheds, as done for other NNRF models (Fernando and Shamseldin, 2009; Jain and Kumar, 2009; Sudheer and Jain, 2004; Wilby et al., 2003). This could hinder their adoption by practitioners who might still prefer less performant but more physically-sound models. Further concerns could arise from the results in Chapter 7, which highlighted the difficulties of including expert-knowledge directly into the modular NNRF models performing implicit baseflow separation. However, it is certainly worth exploring whether ELM could amplify the benefits of other modular approaches which have been devised to increase the physical-plausibility of NNRF (Jain and Srinivasulu, 2006; Srinivasulu and Jain, 2009; Toth, 2009) .

In addition, practitioners might be willing to downplay the lack of physical interpretation of pure black-box NNRF solutions in case workable and consistent methodologies are implemented for estimating the uncertainty of the predictions. In this regard, it is important to note that ELM may substantially increase the computational efficiency of time demanding bootstrap approaches for estimating prediction intervals

(Sharma and Tiwari, 2009; Tiwari and Chatterjee, 2010), especially if one resorts to parallel computing (He et al., 2013). It will be interesting to assess how these ELM-based solutions compare with MOFIPS-LUBE both in terms of quality of the produced predictive intervals and computational demands.

A final remark concerning the work presented in this manuscript is that, although all the presented applications involve novel PSO variants, the presented techniques are *non-specific* with respect to the Global Optimization (GO) algorithm employed. The PSO algorithm was chosen due to its flexibility and ease of implementation of both real- and binary-coded problems, involving one or more objective functions. Although this certainly facilitated the development of ad-hoc techniques targeted to address different NNRF issues, it is entirely possible that other GO techniques might outperform the MOFIPS, BFIPS and MBFIPS algorithms introduced in this work. Further studies could be thus directed towards redeveloping the proposed methodologies using other GO algorithms, such as Borg (Hadka and Reed, 2012; Reed et al., 2013) and variants Differential Evolution (Das and Suganthan, 2011; Piotrowski and Napiorkowski, 2011; Piotrowski et al., 2012), which have proven very successful in other water resources and environmental modelling applications.

APPENDIX A. Benchmark functions

The following functions taken from Deb et al. (2002) have been employed to test the performances of the MOFIPS algorithm in 3.7.3.

Problem	n	Variable bounds	Objective functions	Optimal solutions	Comments
SCH	1	$[-10^3, 10^3]$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	$x \in [0, 2]$	convex
FON	3	$[-4, 4]$	$f_1(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right)$ $f_2(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right)$	$x_1 = x_2 = x_3$ $\in [-1/\sqrt{3}, 1/\sqrt{3}]$	nonconvex
ZDT1	30	$[0, 1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \ldots, n$	convex
ZDT2	30	$[0, 1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (x_1/g(\mathbf{x}))^2\right]$ $g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \ldots, n$	nonconvex
ZDT3	30	$[0, 1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(\mathbf{x})} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x_1)\right]$ $g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \ldots, n$	convex, disconnected
ZDT6	10	$[0, 1]$	$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (f_1(\mathbf{x})/g(\mathbf{x}))^2\right]$ $g(\mathbf{x}) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25}$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \ldots, n$	nonconvex, nonuniformly spaced

APPENDIX B. Statistical metrics for assessment of hydrological models

The following statistical metrics have been employed for the assessment of the proposed NNRF models. These are (Dawson et al., 2010, 2007): the root mean square error (RMSE), the coefficient of efficiency (CE), the median absolute percentage error (MdAPE), fourth root mean quadrupled error (R4MS4E), and mean squared logarithmic error (MSLE), given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T}\left(Q(t)-\tilde{Q}(t)\right)^2}{N}},$$

$$CE = 1 - \frac{\sum\left((Q(t)-\tilde{Q}(t))^2\right)}{\sum\left((Q(t)-\bar{Q}(t))^2\right)},$$

$$MdAPE = median\left(\left|\frac{Q(t)-\tilde{Q}(t)}{Q(t)}\right| \times 100\right),$$

$$R4MS4E = \sqrt[4]{\frac{\sum_{i=1}^{T}\left(Q(t)-\tilde{Q}(t)\right)^4}{N}},$$

$$MSLE = \frac{1}{N}\sum_{i=1}^{T}\left(ln\left(Q(t)\right) - ln\left(\tilde{Q}(t)\right)\right)^2.$$

where $Q(t)$ is the observed discharge value, $\tilde{Q}(t)$ is the modeled discharge value, $\bar{Q}(t)$ is the median observed value, and *N* is the total number of observations. The first three evaluation metrics are used to assess the level of overall agreement between the observed and modeled output variables. The root mean square error is a non-negative metric expressed in real units with no upper bound and equal to 0 for a perfect model. On the other hand, the coefficient of efficiency is dimensionless with a value of 1 for a perfect model and no lower bound. The median absolute percentage error is a dimensionless non-negative ratio, which is equal to 0 for a perfect model and has no

upper bound. Although less popular than RMSE and CE, the MdAPE, based on the median of the absolute residuals, is less affected by skewed error distributions and less sensitive to the larger errors that occur at high flows (Dawson et al., 2007). The fourth root mean quadrupled error is used to better evaluate model goodness-of-fit on peak and high flows. It is a non-negative metric expressed in real units that has no upper bound and is equal to 0 for a perfect model. Somewhat complementary to the R4MS4E is the mean squared logarithmic error, which, due to the logarithmic transformations involved in its computation, is a preferred measure for assessing model performances when predicting low flows. MSLE is non-negative and takes a value of 0 for a perfect model.

Bibliography

Abrahart, R. J., Anctil, F., Coulibaly, P., Dawson, C. W., Mount, N. J., See, L. M., … Wilby, R. L. (2012). Two decades of anarchy? Emerging themes and outstanding challenges for neural network river forecasting. *Progress in Physical Geography*, *36*(4), 480–513. doi:10.1177/0309133312444943

Abrahart, R. J., Heppenstall, A. J., & See, L. M. (2007). Timing error correction procedure applied to neural network rainfall—runoff modelling. *Hydrological Sciences Journal*, *52*(3), 414–431. doi:10.1623/hysj.52.3.414

Abrahart, R. J., See, L. M., & Kneale, P. E. (1999). Using pruning algorithms and genetic algorithms to optimise network architectures and forecasting inputs in a neural network rainfall-runoff model. *Journal of Hydroinformatics*, *1*(2), 103–114.

Acharya, N., Shrivastava, N. A., Panigrahi, B. K., & Mohanty, U. C. (2013). Development of an artificial neural network based multi-model ensemble to estimate the northeast monsoon rainfall over south peninsular India: an application of extreme learning machine. *Climate Dynamics*, *43*(5-6), 1303–1310. doi:10.1007/s00382-013-1942-2

Adamowski, J., & Chan, H. F. (2011). A wavelet neural network conjunction model for groundwater level forecasting. *Journal of Hydrology*, *407*(1-4), 28–40. doi:10.1016/j.jhydrol.2011.06.013

Adamowski, J., & Karapataki, C. (2010). Comparison of multivariate regression and artificial neural networks for peak urban water-demand forecasting: Evaluation of different ANN learning algorithms. *Journal of Hydrologic Engineering*, *15*(10), 729–743. doi:10.1061/(ASCE)HE.1943-5584.0000245

Adamowski, J., & Sun, K. (2010). Development of a coupled wavelet transform and neural network method for flow forecasting of non-perennial rivers in semi-arid watersheds. *Journal of Hydrology*, *390*(1-2), 85–91. doi:10.1016/j.jhydrol.2010.06.033

Afshar, M. H. (2007). Partially constrained ant colony optimization algorithm for the solution of constrained optimization problems: Application to storm water network design. *Advances in Water Resources*, *30*(4), 954–965. doi:10.1016/j.advwatres.2006.08.004

Aksoy, H., & Wittenberg, H. (2011). Nonlinear baseflow recession analysis in watersheds with intermittent streamflow. *Hydrological Sciences Journal*, *56*(2), 226–237. doi:10.1080/02626667.2011.553614

Alvisi, S., & Franchini, M. (2011). Fuzzy neural networks for water level and discharge forecasting with uncertainty. *Environmental Modelling & Software*, *26*(4), 523–537. doi:10.1016/j.envsoft.2010.10.016

Anctil, F., & Lauzon, N. (2004). Generalisation for neural networks through data sampling and training procedures, with applications to streamflow predictions. *Hydrology and Earth System Sciences*, *8*(5), 940–958. doi:10.5194/hess-8-940-2004

Anctil, F., Perrin, C., & Andréassian, V. (2004). Impact of the length of observed records on the performance of ANN and of conceptual parsimonious rainfall-runoff forecasting models. *Environmental Modelling & Software*, *19*(4), 357–368. doi:10.1016/S1364-8152(03)00135-X

Bhattacharya, B., & Solomatine, D. P. (2005). Neural networks and M5 model trees in modelling water level–discharge relationship. *Neurocomputing*, *63*, 381–396. doi:10.1016/j.neucom.2004.04.016

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine. *Artificial Intelligence*, *97*, 245–271.

Borga, M., Boscolo, P., Zanon, F., & Sangati, M. (2007). Hydrometeorological Analysis of the 29 August 2003 Flash Flood in the Eastern Italian Alps. *Journal of Hydrometeorology*, *8*(5), 1049–1067. doi:10.1175/JHM593.1

Bowden, G. J., Dandy, G. C., & Maier, H. R. (2005). Input determination for neural network models in water resources applications. Part 1—background and methodology. *Journal of Hydrology*, *301*(1-4), 75–92. doi:10.1016/j.jhydrol.2004.06.021

Bowden, G. J., Maier, H. R., & Dandy, G. C. (2005). Input determination for neural network models in water resources applications. Part 2. Case study: forecasting salinity in a river. *Journal of Hydrology*, *301*(1-4), 93–107. doi:10.1016/j.jhydrol.2004.06.020

Branke, J., Deb, K., Dierolf, H., & Osswald, M. (2004). Finding knees in multi-objective optimization. In *Parallel Problem Solving from Nature-PPSN VIII* (pp. 722–731). Springer. doi:10.1007/b100601

Brownlee, J. (2012). *Clever Algorithms: Nature-Inspired Programming Recipes*.

Cannon, A. J., & Whitfield, P. H. (2002). Downscaling recent streamflow conditions in British Columbia , Canada using ensemble neural network models. *Journal of Hydrology*, *259*, 136–151.

Carcano, E. C., Bartolini, P., Muselli, M., & Piroddi, L. (2008). Jordan recurrent neural network versus IHACRES in modelling daily streamflows. *Journal of Hydrology*, *362*(3-4), 291–307. doi:10.1016/j.jhydrol.2008.08.026

Chang, F. J., Kao, L. S., Kuo, Y. M., & Liu, C. W. (2010). Artificial neural networks for estimating regional arsenic concentrations in a blackfoot disease area in Taiwan. *Journal of Hydrology*, *388*(1-2), 65–76. doi:10.1016/j.jhydrol.2010.04.029

Chau, K. W. (2006). Particle swarm optimization training algorithm for ANNs in stage prediction of Shing Mun River. *Journal of Hydrology*, *329*(3-4), 363–367. doi:10.1016/j.jhydrol.2006.02.025

Chau, K. W. (2007). A split-step particle swarm optimization algorithm in river stage forecasting. *Journal of Hydrology*, *346*(3-4), 131–135. doi:10.1016/j.jhydrol.2007.09.004

Chau, K.W., Wu, C. L., & Li, Y. S. (2005). Comparison of Several Flood Forecasting Models in Yangtze River. *Journal of Hydrologic Engineering*, *10*(6) 485–491. doi:10.1061/(ASCE)1084-0699(2005)10:6(485)

Chaves, P., & Chang, F.J. (2008). Intelligent reservoir operation system based on evolving artificial neural networks. *Advances in Water Resources*, *31*(6), 926–936. doi:10.1016/j.advwatres.2008.03.002

Chen, Y., & Chang, F.J. (2009). Evolutionary artificial neural networks for hydrological systems forecasting. *Journal of Hydrology*, *367*(1-2), 125–137. doi:10.1016/j.jhydrol.2009.01.009

Clerc, M., & Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73. doi:10.1109/4235.985692

Coello Coello, C. A., & Reyes-Sierra, M. (2006). Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, *2*(3), 287–308. doi:10.5019/j.ijcir.2006.68

Corzo, G. A., & Solomatine, D. P. (2006). Optimization of base flow separation algorithm for modular data-driven hydrologic models. In *Proceedings of the 7th International Conference on Hydroinformatics*. Nice, France.

Corzo, G., & Solomatine, D. (2007a). Baseflow separation techniques for modular artificial neural network modelling in flow forecasting. *Hydrological Sciences Journal*, *52*(3), 491–507. doi:10.1623/hysj.52.3.491

Corzo, G., & Solomatine, D. (2007b). Knowledge-based modularization and global optimization of artificial neural network models in hydrological forecasting. *Neural Networks*, *20*(4), 528–36. doi:10.1016/j.neunet.2007.04.019

Coulibaly, P., Anctil, F., Aravena, R., & Bobde, B. (2001). Artificial neural network modeling of water table depth fluctuations, *Water Resources Research 37*(4), 885–896. doi:10.1029/2000WR900368

Coulibaly, P., Anctil, F., & Bobée, B. (2000). Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. *Journal of Hydrology*, *230*(3-4), 244–257. doi:10.1016/S0022-1694(00)00214-6

Danandeh Mehr, A., Kahya, E., & Olyaie, E. (2013). Streamflow prediction using linear genetic programming in comparison with a neuro-wavelet technique. *Journal of Hydrology*, *505*, 240–249. doi:10.1016/j.jhydrol.2013.10.003

Das, S., & Suganthan, P. N. (2011). Differential Evolution : A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation*, *15*(1), 4–31.

Dawson, C. W., Abrahart, R. J., & See, L. M. (2007). HydroTest: A web-based toolbox of evaluation metrics for the standardised assessment of hydrological forecasts. *Environmental Modelling & Software*, *22*(7), 1034–1052. doi:10.1016/j.envsoft.2006.06.008

Dawson, C. W., Abrahart, R. J., & See, L. M. (2010). HydroTest: Further development of a web resource for the standardised assessment of hydrological models. *Environmental Modelling & Software*, *25*(11), 1481–1482. doi:10.1016/j.envsoft.2009.01.001

Dawson, C. W., Harpham, C., Wilby, R. L., & Chen, Y. (2002). Evaluation of artificial neural network techniques for flow forecasting in the River Yangtze, China. *Hydrology and Earth System Sciences*, *6*(4), 619–626. doi:10.5194/hess-6-619-2002

Dawson, C. W., See, L. M., Abrahart, R. J., & Heppenstall, A. J. (2006). Symbiotic adaptive neuro-evolution applied to rainfall-runoff modelling in northern England. *Neural Networks*, *19*(2), 236–47. doi:10.1016/j.neunet.2006.01.009

Dawson, C. W., & Wilby, R. (1998). An artificial neural network approach to rainfall-runoff modelling. *Hydrological Sciences Journal*, *43*(1), 47–66. doi:10.1080/02626669809492102

Dawson, C. W., & Wilby, R. L. (2001). Hydrological modelling using artificial neural networks. *Progress in Physical Geography*, *25*(1), 80–108. doi:10.1191/030913301674775671

De Gooijer, J. G., & Hyndman, R. J. (2006). 25 Years of Time Series Forecasting. *International Journal of Forecasting*, *22*(3), 443–473. doi:10.1016/j.ijforecast.2006.01.001

De Vos, N. J., & Rientjes, T. H. M. (2007). Multi-objective performance comparison of an artificial neural network and a conceptual rainfall—runoff model. *Hydrological Sciences Journal*, *52*(3), 397–413. doi:10.1623/hysj.52.3.397

De Vos, N. J., & Rientjes, T. H. M. (2008). Multiobjective training of artificial neural networks for rainfall-runoff modeling. *Water Resources Research*, *44*(8), W08434. doi:10.1029/2007WR006734

Deb, K. (2009). *Multi-objective optimization using evolutionary algorithms*. New York: John Wiley & Sons.

Deb, K., & Deb, D. (2012). *Analyzing Mutation Schemes for Real-Parameter Genetic Algorithms. KanGAL, Report No. 2012016.*

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. doi:10.1109/4235.996017

Deo, R. C., & Şahin, M. (2015). Application of the extreme learning machine algorithm for the prediction of monthly Effective Drought Index in eastern Australia. *Atmospheric Research*, *153*, 512–525. doi:10.1016/j.atmosres.2014.10.016

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant systems: optimization by a colony of cooperative agents. *IEEE Transactions on Man, Machine and Cybernetics-Part B*, *26*(1), 29 – 41.

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the 2000 IEEE Congress on Evolutionary Computation - CEC'00*, 84–88.

Eckhardt, K. (2005). How to construct recursive digital filters for baseflow separation. *Hydrological Processes*, *19*(2), 507–515. doi:10.1002/hyp.5675

Eckhardt, K. (2008). A comparison of baseflow indices, which were calculated with seven different baseflow separation methods. *Journal of Hydrology*, *352*(1-2), 168–173. doi:10.1016/j.jhydrol.2008.01.005

Eckhardt, K. (2012). Technical Note: Analytical sensitivity analysis of a two parameter recursive digital baseflow separation filter. *Hydrology and Earth System Sciences*, *16*(2), 451–455. doi:10.5194/hess-16-451-2012

Fernando, D. A. K., & Shamseldin, A. Y. (2009). Investigation of Internal Functioning of the Radial-Basis-Function Neural Network River Flow Forecasting Models. *Journal of Hydrologic Engineering*, *14*(3), 286–292. doi: 10.1061/(ASCE)1084-0699(2009)14:3(286)

Fernando, T. M. K. G., Maier, H. R., & Dandy, G. C. (2009). Selection of input variables for data driven models: An average shifted histogram partial mutual information estimator approach. *Journal of Hydrology*, *367*(3-4), 165–176. doi:10.1016/j.jhydrol.2008.10.019

Foresee, F. D., & Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. In *1997 International Joint Conferenence on Neural Networks* (pp. 1930–1935).

Galelli, S., & Castelletti, A. (2013a). Assessing the predictive capability of randomized tree-based ensembles in streamflow modelling. *Hydrology and Earth System Sciences Discussions*, *10*(2), 1617–1655. doi:10.5194/hessd-10-1617-2013

Galelli, S., & Castelletti, A. (2013b). Tree-based iterative input variable selection for hydrological modeling. *Water Resources Research*, *49*(7), 4295–4310. doi:10.1002/wrcr.20339

Galelli, S., Humphrey, G. B., Maier, H. R., Castelletti, A., Dandy, G. C., & Gibbs, M. S. (2014). An evaluation framework for input variable selection algorithms for environmental data-driven models. *Environmental Modelling & Software*, *62*, 33–51. doi:10.1016/j.envsoft.2014.08.015

Gaur, S., Chahar, B. R., & Graillot, D. (2011). Analytic elements method and particle swarm optimization based simulation-optimization model for groundwater management. *Journal of Hydrology*, *402*(3-4), 217–227. doi:10.1016/j.jhydrol.2011.03.016

Gill, M. K., Kaheil, Y. H., Khalil, A., McKee, M., & Bastidas, L. (2006). Multiobjective particle swarm optimization for parameter estimation in hydrology. *Water Resources Research*, *42*(7), W07417. doi:10.1029/2005WR004528

Goldberg, D. E. (1989). *Genetic Algorithms in Search , Optimization , and Machine Learning*. Addison-wesley.

Govindaraju, R. S. (2000a). Artificial Neural Networks in Hydrology. I: Preliminary Concepts. *Journal of Hydrologic Engineering*, *5*(2), 115–123.

Govindaraju, R. S. (2000b). Artificial Neural Networks in Hydrology. II: Hydrologic Applications. *Journal of Hydrologic Engineering*, *5*(2), 124–137.

Guo, X., Hu, T., Zhang, T., & Lv, Y. (2012). Bilevel model for multi-reservoir operating policy in inter-basin water transfer-supply project. *Journal of Hydrology*, *424-425*, 252–263. doi:10.1016/j.jhydrol.2012.01.006

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hadka, D., & Reed, P. (2012). Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework. *Evolutionary Computation*, *21*(2), 1–30. doi:10.1162/EVCO_a_00075

Hamed, M. M., Khalafallah, M. G., & Hassanien, E. a. (2004). Prediction of wastewater treatment plant performance using artificial neural networks. *Environmental Modelling Software*, *19*(10), 919–928. doi:10.1016/j.envsoft.2003.10.005

Haykin, S. (2008). *Neural Networks and Learning Machines* (3rd ed.). Prentice Hall.

He, Q., Shang, T., Zhuang, F., & Shi, Z. (2013). Parallel extreme learning machine for regression based on MapReduce. *Neurocomputing*, *102*, 52–58. doi:10.1016/j.neucom.2012.01.040

Higashi, N., & Iba, H. (2003). Particle swarm optimization with Gaussian mutation. In *Proceedings of the 2003 Swarm Intelligence Symposium - SIS'03* (pp. 72–79). IEEE.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359 – 366.

Huang, G.B., Chen, L., & Siew, C. (2006). Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes. *IEEE Transactions on Neural Networks*, *17*(4), 879–892.

Huang, G.B., Wang, D. H., & Lan, Y. (2011). Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, *2*(2), 107–122. doi:10.1007/s13042-011-0019-y

Huang, G.B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, *42*(2), 513–529.

Huang, G.B., Zhu, Q.Y., & Siew, C.K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In IEEE (Ed.), *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on (Vol. 2)* (pp. 985–990). doi:http://dx.doi.org/10.1109/IJCNN.2004.1380068

Huang, G.B., Zhu, Q.Y., & Siew, C.K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, *70*(1-3), 489–501. doi:10.1016/j.neucom.2005.12.126

Imrie, C. E., Durucan, S., & Korre, A. (2000). River flow prediction using artificial neural networks: generalisation beyond the calibration range. *Journal of Hydrology*, *233*, 138–153. doi:10.1016/S0022-1694(00)00228-6

Jain, A., & Kumar, S. (2009). Dissection of trained neural network hydrologic models for knowledge extraction. *Water Resources Research*, *45*(7), W07420. doi:10.1029/2008WR007194

Jain, A., & Srinivasulu, S. (2004). Development of effective and efficient rainfall-runoff models using integration of deterministic, real-coded genetic algorithms and artificial neural network techniques. *Water Resources Research*, *40*(4), W04302. doi:10.1029/2003WR002355

Jain, A., & Srinivasulu, S. (2006). Integrated approach to model decomposed flow hydrograph using artificial neural network and conceptual techniques. *Journal of Hydrology*, *317*(3-4), 291–306. doi:10.1016/j.jhydrol.2005.05.022

Jain, A., Sudheer, K. P., & Srinivasulu, S. (2004). Identification of physical processes inherent in artificial neural network rainfall runoff models. *Hydrological Processes*, *18*(3), 571–581. doi:10.1002/hyp.5502

Jayawardena, A. W., & Fernando, D. A. K. (1998). Use of Radial Basis Function Type Artificial Neural Networks for Runoff Simulation. *Computer-Aided Civil and Infrastructure Engineering*, *13*(2), 91–99. doi:10.1111/0885-9507.00089

Jiang, Y., Hu, T., Huang, C., & Wu, X. (2007). An improved particle swarm optimization algorithm. *Applied Mathematics and Computation*, *193*(1), 231–239. doi:10.1016/j.amc.2007.03.047

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. doi:10.1007/s10898-007-9149-x

Kennedy, J. (1999). Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, 1931–1938. doi:10.1109/CEC.1999.785509

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*(2), 1942–1948. doi:10.1109/ICNN.1995.488968

Kennedy, J., & Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation., 5*, 4104 –4108.

Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02., 2*, 1671–1676. doi:10.1109/CEC.2002.1004493

Khan, M. S., & Coulibaly, P. (2006). Bayesian neural network for rainfall-runoff modeling. *Water Resources Research*, *42*(7). W07409. doi:10.1029/2005WR003971

Khosravi, A., Nahavandi, S., & Creighton, D. (2010). Construction of Optimal Prediction Intervals for Load Forecasting Problems. *IEEE Transactions on Power Systems*, *25*(3), 1496–1503.

Khosravi, A., Nahavandi, S., & Creighton, D. (2011). Prediction interval construction and optimization for adaptive neurofuzzy inference systems. *IEEE Transactions on Fuzzy Systems*, *19*(5), 983–988.

Khosravi, A., Nahavandi, S., Creighton, D., & Atiya, A. F. (2011). Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE Transactions on Neural Networks*, *22*(3), 337–46. doi:10.1109/TNN.2010.2096824

Kingston, G. B., Lambert, M. F., & Maier, H. R. (2005). Bayesian training of artificial neural networks used for water resources modeling. *Water Resources Research*, *41*(12), W04419. doi:10.1029/2005WR004152

Kişi, Ö. (2004). River Flow Modeling Using Artificial Neural Networks. *Journal of Hydrologic Engineering*, *9*(1), 60–63. doi:10.1061/(ASCE)1084-0699(2004)9:1(60)

Kişi, Ö. (2009). Neural Networks and Wavelet Conjunction Model for Intermittent Streamflow Forecasting. *Journal of Hydrologic Engineering*, *14*(8), 773–782. doi:10.1061/(ASCE)HE.1943-5584.0000053

Kişi, Ö. (2010). River suspended sediment concentration modeling using a neural differential evolution approach. *Journal of Hydrology*, *389*(1-2), 227–235. doi:10.1016/j.jhydrol.2010.06.003

Kisi, O., Ozkan, C., & Akay, B. (2012). Modeling discharge–sediment relationship using neural networks with artificial bee colony algorithm. *Journal of Hydrology*, *428-429*, 94–103. doi:10.1016/j.jhydrol.2012.01.026

Knowles, J., & Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. *Proceedings of the 1999*

Congress on Evolutionary Computation, CEC 1999, *1*, 98–105. doi:10.1109/CEC.1999.781913

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artificial Intelligence*, *14*(12), 1137–1143. doi:10.1067/mod.2000.109031

Kohavi, R., & John, H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Krzysztofowicz, R. (2001). The case for probabilistic forecasting in hydrology. *Journal of Hydrology*, *249*(1-4), 2–9. doi:10.1016/S0022-1694(01)00420-6

Kumar, D. N., & Reddy, M. J. (2006). Ant Colony Optimization for Multi-Purpose Reservoir Operation. *Water Resources Management*, *20*(6), 879–898.

Labadie, J. W. (2004). Optimal Operation of Multireservoir Systems: State-of-the-Art Review. *Journal of Water Resources Planning and Management*, *130*(2), 93–111. doi:10.1061/(ASCE)0733-9496(2004)130:2(93)

Leahy, P., Kiely, G., & Corcoran, G. (2008). Structural optimisation and input selection of an artificial neural network for river level prediction. *Journal of Hydrology*, *355*(1-4), 192–201. doi:10.1016/j.jhydrol.2008.03.017

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, *2*, 164–168.

Li, H., & Zhang, Q. (2009). Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, *13*(2), 284–302. doi: 10.1109/TEVC.2008.925798

Li, H., Zhang, Y., Chiew, F. H. S., & Xu, S. (2009). Predicting runoff in ungauged catchments by using Xinanjiang model with MODIS leaf area index. *Journal of Hydrology*, *370*(1-4), 155–162. doi:10.1016/j.jhydrol.2009.03.003

Li, L., Maier, H. R., Partington, D., Lambert, M. F., & Simmons, C. T. (2014). Performance assessment and improvement of recursive digital baseflow filters for catchments with different physical characteristics and hydrological inputs. *Environmental Modelling & Software*, *54*, 39–52. doi:10.1016/j.envsoft.2013.12.011

Lin, J.Y., Cheng, C.T., & Chau, K.W. (2006). Using support vector machines for long-term discharge prediction. *Hydrological Sciences Journal*, *51*(4), 599–612. doi:10.1623/hysj.51.4.599

Lü, H., Yu, Z., Zhu, Y., Drake, S., Hao, Z., & Sudicky, E. A. (2011). Dual state-parameter estimation of root zone soil moisture by optimal parameter estimation and extended Kalman filter data assimilation. *Advances in Water Resources*, *34*(3), 395–406. doi:10.1016/j.advwatres.2010.12.005

Maier, H. R., & Dandy, G. C. (2000). Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environmental Modelling & Software*, *15*(1), 101–124. doi:10.1016/S1364-8152(99)00007-9

Maier, H. R., Jain, A., Dandy, G. C., & Sudheer, K. P. (2010). Methods used for the development of neural networks for the prediction of water resource variables in river systems: Current status and future directions. *Environmental Modelling & Software*, *25*(8), 891–909. doi:10.1016/j.envsoft.2010.02.003

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, *11*(2), 431–441.

Masters, T. (1993). *Practical Neural Networks Recipes C++*. San Diego: Academic Press.

Masters, T. (1995). *Advanced Algorithms for Neural Networks*. New York: Wiley.

May, R., Dandy, G., & Maier, H. (2011). Review of Input Variable Selection Methods for Artificial Neural Networks. *Artificial Neural Networks—methodological Advances and Biomedical Applications*, pp. 19–44.

May, R. J., Maier, H. R., Dandy, G. C., & Fernando, T. M. K. G. (2008). Non-linear variable selection for artificial neural networks using partial mutual information. *Environmental Modelling & Software*, *23*(10-11), 1312–1326. doi:10.1016/j.envsoft.2008.03.007

Mendes, R. (2004). *Population Topologies and Their Influence in Particle Swarm Performance* (Doctoral dissertation, Universidade do Minho).

Mendes, R., Kennedy, J., & Neves, J. (2003). Watch thy neighbor or how the swarm can learn from its environment. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium - SIS'03*, 88–94. doi:10.1109/SIS.2003.1202252

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, *8*(3), 204–210. doi: 10.1109/TEVC.2004.826074

Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., & Lendasse, A. (2010). OP-ELM : Optimally Pruned Extreme Learning Machine. *IEEE Transactions on Neural Networks*, *21*(1), 158–162.

Minns, A. W., & Hall, M. J. (1996). Artificial neural networks as rainfall-runoff models. *Hydrological Sciences Journal*, *41*(3), 399–417. doi:10.1080/02626669609491511

Mohan, S. (1997). Parameter Estimation of Nonlinear Muskingum Models Using Genetic Algorithm. *Journal of Hydraulic Engineering*, *123*(2), 137–142.

Mohan, S., & Vijayalakshmi, D. P. (2009). Genetic Algorithm Applications in Water Resources. *ISH Journal of Hydraulic Engineering*, *15*(sup1), 97–128. doi:10.1080/09715010.2009.10514971

Muttil, N., & Chau, K. W. (2006). Neural network and genetic programming for modelling coastal algal blooms. *International Journal of Environment and Pollution*, *28*(3/4), 223. doi:10.1504/IJEP.2006.011208

Muttil, N., & Chau, K. W. (2007). Machine-learning paradigms for selecting ecologically significant input variables. *Engineering Applications of Artificial Intelligence*, *20*(6), 735–744. doi:10.1016/j.engappai.2006.11.016

Nayak, P. C., Venkatesh, B., Krishna, B., & Jain, S. K. (2013a). Rainfall-runoff modeling using conceptual, data driven, and wavelet based computing approach. *Journal of Hydrology*, *493*, 57–67. doi:10.1016/j.jhydrol.2013.04.016

Nayak, P. C., Venkatesh, B., Krishna, B., & Jain, S. K. (2013b). Rainfall-runoff modeling using conceptual, data driven, and wavelet based computing approach. *Journal of Hydrology*, *493*, 57–67. doi:10.1016/j.jhydrol.2013.04.016

Noori, R., Karbassi, A. R., Moghaddamnia, A., Han, D., Zokaei-Ashtiani, M. H., Farokhnia, a., & Gousheh, M. G. (2011). Assessment of input variables determination on the SVM model performance using PCA, Gamma test, and forward selection techniques for monthly stream flow prediction. *Journal of Hydrology*, *401*(3-4), 177–189. doi:10.1016/j.jhydrol.2011.02.021

Nourani, V., Hosseini Baghanam, A., Adamowski, J., & Kisi, O. (2014). Applications of hybrid wavelet-Artificial Intelligence models in hydrology: A review. *Journal of Hydrology*, *514*, 358–377. doi:10.1016/j.jhydrol.2014.03.057

Nourani, V., Komasi, M., & Alami, M. T. (2012). Hybrid Wavelet – Genetic Programming Approach to Optimize ANN Modeling of Rainfall – Runoff Process. *Journal of Hydrologic Engineering*, *17*(6), 724–741. doi:10.1061/(ASCE)HE.1943-5584.0000506.

Nourani, V., Komasi, M., & Mano, A. (2009). A Multivariate ANN-Wavelet Approach for Rainfall–Runoff Modeling. *Water Resources Management*, *23*(14), 2877–2894. doi:10.1007/s11269-009-9414-5

Ortiz-García, E. G., Salcedo-Sanz, S., & Casanova-Mateo, C. (2014). Accurate precipitation prediction with support vector classifiers: A study including novel predictive variables and observational data. *Atmospheric Research*, *139*, 128–136. doi:10.1016/j.atmosres.2014.01.012

Parasuraman, K., Elshorbagy, A., & Carey, S. K. (2006). Spiking modular neural networks: A neural network modeling approach for hydrological processes. *Water Resources Research*, *42*(5), W05412. doi:10.1029/2005WR004317

Parasuraman, K., Elshorbagy, A., & Carey, S. K. (2007). Modelling the dynamics of the evapotranspiration process using genetic programming. *Hydrological Sciences Journal*, *52*(3), 563–578. doi:10.1623/hysj.52.3.563

Piotrowski, A. P., & Napiorkowski, J. J. (2011). Optimizing neural networks for river flow forecasting – Evolutionary Computation methods versus the Levenberg–Marquardt approach. *Journal of Hydrology*, *407*(1-4), 12–27. doi:10.1016/j.jhydrol.2011.06.019

Piotrowski, A. P., & Napiorkowski, J. J. (2013). A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. *Journal of Hydrology*, *476*, 97–111. doi:10.1016/j.jhydrol.2012.10.019

Piotrowski, A. P., Rowinski, P. M., & Napiorkowski, J. J. (2012a). Comparison of evolutionary computation techniques for noise injected neural network training to estimate longitudinal dispersion coefficients in rivers. *Expert Systems with Applications*, *39*(1), 1354–1361. doi:10.1016/j.eswa.2011.08.016

Piotrowski, A. P., Rowinski, P. M., & Napiorkowski, J. J. (2012b). Expert Systems with Applications Comparison of evolutionary computation techniques for noise injected neural network training to estimate longitudinal dispersion coefficients in rivers. *Expert Systems With Applications*, *39*(1), 1354–1361. doi:10.1016/j.eswa.2011.08.016

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, *1*(1), 33–57. doi:10.1007/s11721-007-0002-0

Pramanik, N., & Panda, R. K. (2009). Application of neural network and adaptive neuro-fuzzy inference systems for river flow prediction. *Hydrological Sciences Journal*, *54*(2), 247–260. doi:10.1623/hysj.54.2.247

Prasad, T. D., & Park, N.-S. (2004). Multiobjective Genetic Algorithms for Design of Water Distribution Networks. *Journal of Water Resources Planning and Management*, *130*(1), 73–82. doi:10.1061/(ASCE)0733-9496(2004)130:1(73)

Qi, M., & Zhang, G. P. (2001). An investigation of model selection criteria for neural network time series forecasting. *European Journal of Operational Research*, *132*, 668-680.

Quan, H., Srinivasan, D., & Khosravi, A. (2014a). Particle swarm optimization for construction of neural network-based prediction intervals. *Neurocomputing*, *127*, 172–180. doi:10.1016/j.neucom.2013.08.020

Quan, H., Srinivasan, D., & Khosravi, A. (2014b). Short-term load and wind power forecasting using neural network-based prediction intervals. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(2), 303–15. doi:10.1109/TNNLS.2013.2276053

Quan, H., Srinivasan, D., & Khosravi, A. (2014c). Uncertainty handling using neural network-based prediction intervals for electrical load forecasting. *Energy*, *73*, 916–925. doi:10.1016/j.energy.2014.06.104

Raman, H., & Chandramouli, V. (1996). Deriving a General Operating Policy for Reservoirs Using Neural Network. *Journal of Water Resources Planning and Management*, *122*(5), 342–347.

Reddy, M. J., & Kumar, D. N. (2007a). Multi-objective particle swarm optimization for generating optimal trade-offs in reservoir operation. *Hydrological Processes*, *21*(21), 2897–2909. doi:10.1002/hyp

Reddy, M. J., & Kumar, D. N. (2007b). Optimal reservoir operation for irrigation of multiple crops using elitist-mutated particle swarm optimization. *Hydrological Sciences Journal-Journal Des Sciences Hydrologiques*, *52*(4), 686–701. doi:10.1623/hysj.52.4.686

Reddy, M. J., & Kumar, D. N. (2009). Performance evaluation of elitist-mutated multi-objective particle swarm optimization for integrated water resources management. *Journal of Hydroinformatics*, *11*(1), 79. doi:10.2166/hydro.2009.042

Reed, P. M., Hadka, D., Herman, J. D., Kasprzyk, J. R., & Kollat, J. B. (2013). Evolutionary multiobjective optimization in water resources: The past, present, and future. *Advances in Water Resources*, *51*, 438–456. doi:10.1016/j.advwatres.2012.01.005

Sahoo, G. B., Ray, C., & De Carlo, E. H. (2006). Use of neural network to predict flash flood and attendant water qualities of a mountainous stream on Oahu, Hawaii. *Journal of Hydrology*, *327*(3-4), 525–538. doi:10.1016/j.jhydrol.2005.11.059

Sanikhani, H., & Kisi, O. (2012). River Flow Estimation and Forecasting by Using Two Different Adaptive Neuro-Fuzzy Approaches. *Water Resources Management*, *26*(6), 1715–1729. doi:10.1007/s11269-012-9982-7

Savic, D. A., Walters, G. A., & Davidson, J. W. (1999). A Genetic Programming Approach to Rainfall-Runoff Modelling. *Water Resources Management, 13*, 219–231.

Sedki, A., Ouazar, D., & El Mazoudi, E. (2009). Evolving neural network using real coded genetic algorithm for daily rainfall–runoff forecasting. *Expert Systems with Applications*, *36*(3), 4523–4527. doi:10.1016/j.eswa.2008.05.024

Senthil Kumar, A. R., Sudheer, K. P., Jain, S. K., & Agarwal, P. K. (2005). Rainfall-runoff modelling using artificial neural networks: comparison of network types. *Hydrological Processes*, *19*(6), 1277–1291. doi:10.1002/hyp.5581

Shamseldin, A. Y. (1997). Application of a neural network technique to rainfall-runoff modelling. *Journal of Hydrology*, *199*(3-4), 272–294. doi:10.1016/S0022-1694(96)03330-6

Sharma, A. (2000). Seasonal to interannual rainfall probabilistic forecasts for improved water supply management: Part 1 — A strategy for system predictor identification. *Journal of Hydrology*, *239*(1-4), 232–239. doi:10.1016/S0022-1694(00)00346-2

Sharma, A., & Mehrotra, R. (2014). An information theoretic alternative to model a natural system using observational information alone. *Water Resources Research*, *50*(1), 650–660. doi:10.1002/2013WR013845

Sharma, S. K., & Tiwari, K. N. (2009). Bootstrap based artificial neural network (BANN) analysis for hierarchical prediction of monthly runoff in Upper Damodar Valley Catchment. *Journal of Hydrology*, *374*(3-4), 209–222. doi:10.1016/j.jhydrol.2009.06.003

Shrestha, D. L., & Solomatine, D. P. (2006). Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, *19*(2), 225–35. doi:10.1016/j.neunet.2006.01.012

Srinivasulu, S., & Jain, A. (2009). River Flow Prediction Using an Integrated Approach. *Journal of Hydrologic Engineering*, *14*(1), 75–83. doi: 10.1061/(ASCE)1084-0699(2009)14:1(75)

Ssegane, H., Tollner, E. W., Mohamoud, Y. M., Rasmussen, T. C., & Dowd, J. F. (2012). Advances in variable selection methods I: Causal selection methods versus stepwise regression and principal component analysis on data of known and unknown functional relationships. *Journal of Hydrology*, *438-439*, 16–25. doi:10.1016/j.jhydrol.2012.01.008

Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(3), 341–359.

Sudheer, K. P., & Jain, A. (2004). Explaining the internal behaviour of artificial neural network river flow models. *Hydrological Processes*, *18*(4), 833–844. doi:10.1002/hyp.5517

Tada, T., & Beven, K. J. (2012). Hydrological model calibration using a short period of observations. *Hydrological Processes*, *26*(6), 883–892. doi:10.1002/hyp.8302

Thirumalaiah, K., & Deo, M. C. (1998). River Stage Forecasting Using Artificial Neural Networks. *Journal of Hydrologic Engineering*, *3*(1), 26–32.

Tiwari, M. K., & Chatterjee, C. (2010). Uncertainty assessment and ensemble flood forecasting using bootstrap based artificial neural networks (BANNs). *Journal of Hydrology*, *382*(1-4), 20–33. doi:10.1016/j.jhydrol.2009.12.013

Tokar, A. S., & Johnson, P. A. (1999). Rainfall-Runoff Modeling Using Artificial Neural Networks. *Journal of Hydrologic Engineering*, *4*(3), 232–239.

Tokar, A. S., & Markus, M. (2000). Precipitation-Runoff Modeling Using Artificial Neural Networks and Conceptual Models. *Journal of Hydrologic Engineering*, (April), 156–161.

Tomassetti, B., Verdecchia, M., & Giorgi, F. (2009). NN5: A neural network based approach for the downscaling of precipitation fields - Model description and preliminary results. *Journal of Hydrology*, *367*(1-2), 14–26. doi:10.1016/j.jhydrol.2008.12.017

Toth, E. (2009). Classification of hydro-meteorological conditions and multiple artificial neural networks for streamflow forecasting. *Hydrology and Earth System Sciences*, *13*, 1555–1566.

Toth, E., Brath, A., & Montanari, A. (2000). Comparison of short-term rainfall prediction models for real-time flood forecasting. *Journal of Hydrology*, *239*, 132–147.

Trichakis, I. C., Nikolos, I. K., & Karatzas, G. P. (2009). Optimal selection of artificial neural network parameters for the prediction of a karstic aquifer's response. *Hydrological Processes*, *23*(20), 2956–2969.

Wan Jaafar, W. Z., Liu, J., & Han, D. (2011). Input variable selection for median flood regionalization. *Water Resources Research*, *47*(7). doi:10.1029/2011WR010436

Wilby, R. L., Abrahart, R. J., & Dawson, C. W. (2003). Detection of conceptual model rainfall-runoff processes inside an artificial neural network. *Hydrological Sciences JournalJournal Des Sciences Hydrologiques*, *48*(2), 163–181.

Wu, C. L., & Chau, K. W. (2006). A flood forecasting neural network model with genetic algorithm. *International Journal of Environment and Pollution*, *28*(3/4), 261. doi:10.1504/IJEP.2006.011211

Wu, C. L., & Chau, K. W. (2011). Rainfall–runoff modeling using artificial neural network coupled with singular spectrum analysis. *Journal of Hydrology*, *399*(3-4), 394–409. doi:10.1016/j.jhydrol.2011.01.017

Wu, C. L., & Chau, K. W. (2013). Prediction of rainfall time series using modular soft computing methods. *Engineering Applications of Artificial Intelligence*, *26*(3), 997–1007. doi:10.1016/j.engappai.2012.05.023

Wu, C. L., Chau, K. W., & Fan, C. (2010). Prediction of rainfall time series using modular artificial neural networks coupled with data-preprocessing techniques. *Journal of Hydrology*, *389*(1-2), 146–167. doi:10.1016/j.jhydrol.2010.05.040

Wu, C. L., Chau, K. W., & Li, Y. S. (2009). Predicting monthly streamflow using data-driven models coupled with data-preprocessing techniques. *Water Resources Research*, *45*(8), W08432. doi:10.1029/2007WR006737

Wu, S.J., Lien, H.C., & Chang, C.-H. (2012). Calibration of a conceptual rainfall–runoff model using a genetic algorithm integrated with runoff estimation sensitivity to parameters. *Journal of Hydroinformatics*, *14*(2), 497 – 511. doi:10.2166/hydro.2011.010

Wu, W., Dandy, G. C., & Maier, H. R. (2014). Protocol for developing ANN models and its application to the assessment of the quality of the ANN model development process in drinking water quality modelling. *Environmental Modelling & Software*, *54*, 108–127. doi:10.1016/j.envsoft.2013.12.016

Xue, B., Zhang, M., Member, S., & Browne, W. N. (2013). Particle Swarm Optimization for Feature Selection in Classification : A Multi-Objective Approach. *IEEE Transactions on Cybernetics*, *43*(6), 1656–1671.

Yang, X.-S. (2014). *Nature-Inspired Optimization Algorithms*. Elsevier.

Yao, X. (1999). Evolving Artificial Neural Networks. *Proocedings of the IEEE*, *87*(9), 1423–1447.

Yapo, P. O., Gupta, H. V., & Sorooshian, S. (1998). Multi-objective global optimization for hydrologic models. *Journal of Hydrology*, *204*(1-4), 83–97. doi:10.1016/S0022-1694(97)00107-8

Ye, W., Bates, B. C., Viney, N. R., & Sivapalan, M. (1997). Performance of conceptual rainfall-runoff models in low-yielding ephemeral catchments. *Water Resources Research*, *33*(1), 153–166.

Zanetti, S. S., Sousa, E. F., Oliveira, V. P. S., Almeida, F. T., & Bernardo, S. (2007). Estimating Evapotranspiration Using Artificial Neural Network and Minimum Climatological Data. *Journal of Irrigation and Drainage Engineering*, *133*(2), 83–89.

Zhang, B., & Govindaraju, R. S. (2000). Prediction of watershed runoff using Bayesian concepts networks. *Water Resources Research*, *36*(3), 753–762.

Zhang, X., Liang, F., Srinivasan, R., & Van Liew, M. (2009). Estimating uncertainty of streamflow simulation using Bayesian neural networks. *Water Resources Research*, *45*(2). doi:10.1029/2008WR007030

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *Evolutionary Computation, IEEE Transactions on*, *3*(4), 257–271. doi:10.1109/4235.797969