Bandwidth Prediction based on Nu-Support Vector Regression and Parallel Hybrid Particle Swarm Optimization

Liang Hu [1] , Xilong Che [1]*, Xiaochun Cheng [2]

[1] College of Computer Science and Technology, Jilin University,
No.2699 Qianjin Street, 130012, China.
E-mail: hul@jlu.edu.cn (Liang Hu); paco01008@gmail.com (Xilong Che)
[2] School of Computing Science, Middlesex University,
The Burroughs, Hendon, London NW4 4BT, England, UK.
E-mail: x.cheng@mdx.ac.uk

Received: 18-12-2008
Accepted: 27-11-2009

Abstract

This paper addresses the problem of generating multi-step-ahead bandwidth prediction. Variation of bandwidth is modeled as a Nu-Support Vector Regression (Nu-SVR) procedure. A parallel procedure is proposed to hybridize constant and binary Particle Swarm Optimization (PSO) together for optimizing feature selection and hyper-parameter selection. Experimental results on benchmark data set show that the Nu-SVR model optimized achieves better accuracy than BP neural network and SVR without optimization. As a combination of feature selection and hyper-parameter selection, parallel hybrid PSO achieves better convergence performance than individual ones, and it can improve the accuracy of prediction model efficiently.

Keywords: bandwidth prediction; hyper-parameter selection; feature selection; nu-support vector regression; parallel hybrid particle swarm optimization

1. Introduction

In the context of networks, bandwidth quantifies the data rate that a network link or path can transfer. The bandwidth among computing nodes directly impacts application performance and quality of service. Growing complexity of distributed environment is calling for accurate bandwidth prediction as direction for package routing [1], congestion control [2], bandwidth reservation [3], etc. The prediction step also extends from one-step-ahead [4] to multi-step-ahead [5].

This research focuses on modeling and optimizing methodologies for bandwidth prediction. We consider variation of bandwidth as a time series regression procedure that exploits the relationship between past observation and future prediction. Regression is a feasible way proved by many researches [3,5,6] and our previous investigation [7].

We would like to further discuss the model optimization issues including hyper-parameter selection and feature selection. Main contributions of this paper are as follows:

- to validate nu-support vector regression as method to model multi-step-ahead bandwidth prediction;

*Corresponding Author.

- to propose a parallel hybrid particle swarm optimization algorithm to jointly perform hyper-parameter selection and feature selection;
- to introduce an accuracy and efficiency based criterion for model evaluation of bandwidth prediction.

The rest of this literature is organized as follows: Section 2 provides a brief overview of related works. Section 3 illustrates the modeling method of bandwidth prediction in details. Section 4 gives the parallel hybrid optimizing strategy considering both hyper-parameter selection and feature selection. Section 5 proceeds prediction experiments on benchmark data set and discusses comparative results. Section 6 closes with conclusions as well as directions for future research.

2. Related Works

The importance of bandwidth prediction has attracted efforts from a large number of researchers. Many algorithms have been presented in modeling bandwidth variation. Linear models can be implemented easily and fitted efficiently, therefore they are adopted as prediction methods in many projects. Resource Prediction System (RPS) [6] is a famous project in which bandwidth is modeled using linear time series models, including AR (purely autoregressive), MA (purely moving average), ARMA (autoregressive moving average), ARIMA (autoregressive integrated moving average), and ARFIMA (autoregressive fractionally integrated moving average). These models are also verified in the research of Yao et al. [8]. Another well known system for network prediction is the Network Weather Service (NWS) [4]. Wolski et al. compared the performance of forecasts generated using SNP (Semi_Nonparametric Time Series Analysis), with the techniques implemented by the NWS, such as RUN_AVG (running average), LAST (last measurement), MEDIAN (median filter), GRAD (stochastic gradient) and so on. In the research of Nicholson [9], Markov model is employed in BreadCrumbs project for forecasting connectivity.

However, linear models are incapable of handling nonlinear conditions which exist in most of applications. As a nonlinear pioneer of intelligent methods, Artificial Neural Networks (ANNs) were introduced by many researchers in modeling bandwidth prediction. Liu et al. [2] designed an ANN predictor which can predict the bursty available bandwidth for ABR traffic. Eswaradass et al. [10] proposed an ANN based approach for network performance prediction, tested the ANN mechanism on classical trace files and compared its performance with the NWS system. Experimental results showed the ANN approach always provides an improved prediction over that of NWS. Recurrent neural networks are also employed in several network applications [11,12]. However, the structures of neural networks are given directly by experience in these ANN researches, and there is not analytic mechanism to ensure these structures suitable for other applications. Besides, there is an inherent limitation in ANNs for the reason that ANN models are based on Empirical Risk Minimization (ERM) principle [13], which is short of overcoming under-fitting or over-fitting.

As a promising solution to nonlinear regression problems, Support Vector Machines (SVMs) [14] have recently been winning popularity due to their remarkable characteristics such as good generalization performance, the absence of local minima and sparse representation of the solution. Unlike the ERM principle commonly used in conventional regression techniques, SVM was proposed based on Structural Risk Minimization (SRM) principle, which tries to control model complexity as well as the upper bound of generalization risk, rather than minimizing the training error only, thus is expected to achieve better performance.

In the research of Mariyam et al. [15], SVM is employed to predict TCP throughput. Available bandwidth, queuing delay, and packet loss are chosen as input features and throughput as output label. They also compared the accuracy of SVM predictor to linear History-Based (HB) predictor [16] and indicated superior performance. They emphasized the importance of feature selection on preventing over-fitting, but they only chose features according to experience rather than selection strategy. They used cross-validation [17] to select parameters for prediction models, nevertheless, two parameters were selected separately rather than jointly, which will lead to performance degradation. Huang et al. [3] performed SVM and Particle Swarm Optimization (PSO) separately in building predictors so as to control utility of resource, it is in fact a simple application of the two methods, which means no further improvement is discussed to enhance the model's performance.

Prem and Raghavan [5] have also explored the possibility of applying SVM on bandwidth prediction, and indicated that the SVMs prediction are more accurate and outperform the classical methods such as Autoregressive based ones and Mean/Median based ones. However, there are still shortages in their research that deserve further discussion:

- the number of sample features reaches to 40, but there is no feature selection method employed, thus resulting in a huge storage space and a long training time;
- no pretreatment strategy is announced. Sample values with big metric will bring numerical difficulties during calculation and lead to high regression error;
- they achieve q-step-ahead prediction based on $(q-1)$-step-ahead prediction. It is usually impossible to control prediction accuracy within such iterative mechanism;
- the hyper-parameters of prediction model are given directly without selection procedure, which will lower the generalization performance of prediction model.

In our previous work [7], we have proposed a search strategy to select hyper-parameters, however, feature selection is not taken into consideration. In this paper, we attempt to introduce an evolutionary algorithm in optimizing bandwidth prediction model for the expectation of achieving high efficiency and low error. The motivation of choosing and improving the PSO is owing to its superior performance among evolutionary-based optimization algorithms [18] and wide adaptability in many practical applications.

3. Modeling Prediction with Nu-SVR

3.1. *Prediction problem statement*

Predicting bandwidth variation is a kind of regression procedure as far as its essence is concerned. Bandwidth of a link bd can be expressed by $bd(t)$ because its state value varies dynamically, such values are monitored and recorded to form bandwidth time series, denoted as $Z = \{z_u\}_{u=1}^{U}$, where $z_u \in \mathbb{R}, U \in N$. Let z_t stand for value of current time, then $Z_- = \{z_u\}_{u=1}^{t}$, $Z_+ = \{z_u\}_{u=t+1}^{U}$ can be used to represent history set and future set separately, where $t \in (1,U)$. We define $F : Z_- \rightarrow Z_+$ as prediction function set, then any element $f \in F$ is a prediction function. In this research, we focus on q-step-ahead prediction function, its definition is given in Eq. (1).

$$f : z_{t+q} = f(z_t, z_{t-1}, z_{t-2}, \ldots, z_{t-m+1}) . q, m \in N \quad (1)$$

The prediction framework is schematically shown in Fig. 1. The bandwidth data set is divided into three parts: training set, validation set and test set. The training set is used to build prediction model, which is optimized using validation set and evaluated using test set. The model takes historical data as input and generates prediction for future variation.

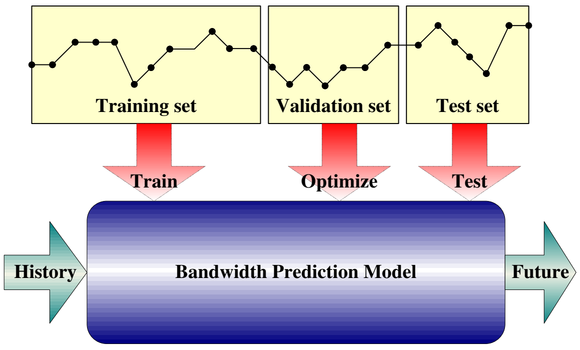


Fig. 1. Prediction framework.

3.2. *Data pretreatment*

Spot set of time series $Z = \{z_u\}_{u=1}^{U}$ can't feed prediction model directly, therefore we transform it into standard sample set (with full features) G, detailed as follows: $G = \{z_t, z_{t-q}, z_{t-q-1}, z_{t-q-2}, \ldots, z_{t-q-m+1}\}_{t=q+m}^{U}$, it can be rewritten as $G = \{z_{i+q+m-1}, z_{i+m-1}, z_{i+m-2}, \ldots, z_i\}_{i=1}^{l}$, $l = U - q - m + 1$, where $z_{i+q+m-1}$ is the labeled output value, $(z_{i+m-1}, z_{i+m-2}, \ldots, z_i)$ is the vector of m input features and l is the number of samples in G. It is in fact a segmentation based on overlapped time windows with equal length, as is shown in Fig. 2.

Furthermore, it is necessary to scale data before applying Nu-SVR method on them: feature values in greater numeric ranges will dominate those in smaller ones; values with large attribute may cause numerical difficulties during calculation and lead to malfunction of Nu-SVR's kernel functions which are usually hypersensitive in a narrow interval. Let $G^{'} = \{y_i, X_i\}_{i=1}^{l}$ denote the scaled sample set, where $y_i = z^{'}_{i+q+m-1}$ is a labeled value, $X_i = (z^{'}_{i+m-1}, z^{'}_{i+m-2}, \ldots, z^{'}_i)$ is a vector containing m features. Scaling and unscaling function are expressed as Eq. (2) and (3).

$$z^{'} = g(z) = \frac{(ub - lb)(z - z^{min})}{z^{max} - z^{min}} + lb \tag{2}$$

$$z = g^{-1}(z^{'}) = \frac{(z^{max} - z^{min})(z^{'} - lb)}{ub - lb} + z^{min} \tag{3}$$

Where z is original value, $z^{'}$ is scaled value in destination interval $[lb, up]$ with lb as lower bound and ub as upper bound; z^{max} is the maximum value of Z, and z^{min} is the minimum value of Z.

3.3. *Modeling with Nu-SVR*

With the pretreated sample set $G^{'} = \{(y_i, X_i)\}_{i=1}^{l}$, where $X_i \in \mathbb{R}^m$ is the i-th vector containing m input features and $y_i \in \mathbb{R}$ is the corresponding desired output label, then modeling is achieved by training a regression function f:

$$f : X_i \mapsto y_i \tag{4}$$

Nu-SVR [19] is a new regression version of SVM. In the following, we will explain in detail how to apply its theory for prediction in the modeling stage. To understand such method, we start from building prediction model using linear regression function:

$$y = f(X) = w^T X + b \tag{5}$$

Where w is the weight vector corresponding to X, and b is the bias. The generalization performance of such linear function $f(X)$ is fairly limited and unable to reflect the true regression procedure. In order to overcome such weakness, a standard mathematical solution is the introduction of $\varphi(X)$, which is a nonlinear mapping function from the input space to a higher dimensional feature space. For example, let's assume that $X = (j, k)$, then we can augment it with $\varphi(X) = (j, k, j^2, k^2, jk)$. By using $\varphi(X)$, we can reach to infinite dimensions for a more expressive f. However, it seems computationally impossible in that case. In fact, we don't have to compute $\varphi(X)$ at all, we just need to compute the inner product $\varphi(X_i)^T \varphi(X_j)$ instead, such artful mechanism will be illustrated latter in this subsection. With the help of $\varphi(X)$, linear regression function Eq. (5) is extended to nonlinear function Eq. (6):

$$y = f(X) = W^T \varphi(X) + b \tag{6}$$

Where W is the weight vector corresponding to $\varphi(X)$. Based on the SRM principle of SVM, our goal is to estimate the coefficients (W and b) following two rules at the same time: first, in order to achieve best performance, $f(X_i)$ should be as close as possible to the truth y_i for all training samples; second, in order to prevent over-fitting, $f(X)$ should be as flat as possible. Then we arrive at how to measure the "closeness" and "flatness".

ε-insensitive loss function L_ε is introduced to measure such "closeness", as in Eq. (7). It measures the absolute error between prediction and true value, but with a tolerance of ε. If $f(X)$ is in the tolerance range of $y \pm \varepsilon$, no loss is considered; if a sample falls out of such range, the distance between sample and ε tube is denoted by slack variables ζ_i and ζ_i^*, as is illustrated by Fig. 3. On the other hand, the parameter norm of $f(X)$ is $||W||^2 = W^T W$, it is used to measure the "flatness": smaller norm means smoother function.

Z	z_U	z_{U-1}	z_{U-2}	$\cdots$	z_{U-q}	z_{U-q-1}	z_{U-q-2}	$\cdots$	$z_{U-q-m+1}$	z_{U-q-m}	$z_{U-q-m-1}$	$\cdots$	z_1
1st sample:	○				●	●	●	●	●				
2nd sample:		○				●	●	●	●	●			
3rd sample:			○				●	●	●	●	●		
..th sample:													

Figure 2: Creation of sample set.

$$L_\varepsilon = \begin{cases} 0 & |y_i - f(X_i)| \leqslant \varepsilon, \\ |y_i - f(X_i)| - \varepsilon & elsewhere. \end{cases} \tag{7}$$

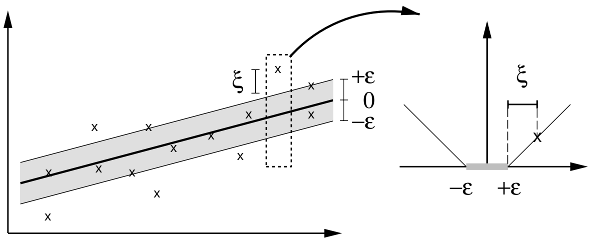


Fig. 3. Theory of slack variable.

Hence we need to minimize the empirical error (overall loss on the training set) $\sum_{i=1}^{l} L_\varepsilon(y_i, f(X_i))$ and the parameter norm $||W||^2$ at the same time, which is equivalent to the following programming problem, namely primal problem of SVR:

$$min \quad \frac{1}{2}W^TW + C\frac{1}{l}\sum_{i=1}^{l}(\zeta_i + \zeta_i^*)$$
$$s.t. \quad \begin{cases} (W^T\varphi(X_i) + b) - y_i \leqslant \varepsilon + \zeta_i, \\ y_i - (W^T\varphi(X_i) + b) \leqslant \varepsilon + \zeta_i^*, \\ \zeta_i, \zeta_i^* \geqslant 0, i = 1, ..., l. \end{cases} \tag{8}$$

Where C is the regularized constant determining the trade-off between the empirical error and the parameter norm. We still face the problem of choosing an adequate parameter ε in order to achieve good performance. Schölkopf [19,20] modified Eq. (8) such that ε also becomes a variable of the problem, and introduced an extra term ν (nu) which attempts to minimize ε. To be more precise, they proved that ν is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors. In addition, with probability 1, asymptotically, ν equals to both fractions. Accordingly Eq. (8) is adapted as follows, namely primal problem of Nu-SVR:

$$min \quad \frac{1}{2}W^TW + C(\nu\varepsilon + \frac{1}{l}\sum_{i=1}^{l}(\zeta_i + \zeta_i^*))$$
$$s.t. \quad \begin{cases} (W^T\varphi(X_i) + b) - y_i \leqslant \varepsilon + \zeta_i, \\ y_i - (W^T\varphi(X_i) + b) \leqslant \varepsilon + \zeta_i^*, \\ \varepsilon, \zeta_i, \zeta_i^* \geqslant 0, i = 1, ..., l. \end{cases} \tag{9}$$

Here $0 \leqslant \nu \leqslant 1$. By introducing Lagrange multipliers $\alpha, \alpha^*, \eta, \eta^*, \beta$, a Lagrange function L is constructed as follows:

$$\begin{aligned} L(\alpha, \alpha^*, \eta, \eta^*, \beta) = \\ \frac{1}{2}W^TW + C\nu\varepsilon + C\frac{1}{l}\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) - \\ \sum_{i=1}^{l}(\eta_i\zeta_i + \eta_i^*\zeta_i^*) - \\ \sum_{i=1}^{l}\alpha_i(\varepsilon + \zeta_i + y_i - W^T\varphi(X_i) - b) - \\ \sum_{i=1}^{l}\alpha_i^*(\varepsilon + \zeta_i^* - y_i + W^T\varphi(X_i) + b) - \beta\varepsilon \end{aligned} \tag{10}$$

It is understood that the Lagrange multipliers in Eq. (10) have to satisfy positive constraints, i.e. $\alpha_i^{(*)}, \eta_i^{(*)}, \beta \geqslant 0$. Here $^{(*)}$ means condition with or without *. It follows from the saddle point condition that the partial derivatives of L with respect to the primal variables $(W, \varepsilon, b, \zeta_i^{(*)})$ have to vanish for optimality:

$$\begin{aligned}
\partial_W L = 0 \rightarrow W = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\varphi(X_i) \\
\partial_b L = 0 \rightarrow \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\
\partial_\varepsilon L = 0 \rightarrow C\nu - \sum_{i=1}^{l}(\alpha_i + \alpha_i^*) - \beta = 0 \\
\partial_{\zeta_i^{(*)}} L = 0 \rightarrow \frac{C}{l} - \alpha_i^{(*)} - \eta_i^{(*)} = 0
\end{aligned} \tag{11}$$

Substituting function Eq. (11) into function Eq. (9) yields the following programming problem, namely dual problem of Nu-SVR:

$$\begin{aligned}
min \quad & \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i)Q_{ij}(\alpha_j^* - \alpha_j) - \\
& \sum_{i=1}^{l} y_i(\alpha_i^* - \alpha_i) \\
s.t. \quad & \begin{cases} \sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i) = 0, \\ 0 \leqslant \sum_{i,j=1}^{l}(\alpha_i^* + \alpha_i) \leqslant C\nu, \\ 0 \leqslant \alpha_i^*, \alpha_i \leqslant \frac{C}{l}, i = 1, ..., l. \end{cases}
\end{aligned} \tag{12}$$

Where Q denotes the matrix of kernel functions, with $Q_{ij} = K(X_i, X_j) = \varphi(X_i)^T\varphi(X_j)$ as kernel function. It can be seen from Eq. (12) that the Lagrange multipliers η_i, η_i^*, β have been eliminated already. In order to solve such dual problem, we just need to compute the inner product $\varphi(X_i)^T\varphi(X_j)$ instead of $\varphi(X)$ itself. Substituting $W = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\varphi(X_i)$ and $Q_{ij} = K(X_i, X_j) = \varphi(X_i)^T\varphi(X_j)$ into Eq. (6) yields the prediction function:

$$y = f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(X_i, X) + b \tag{13}$$

This is the so-called support vector expansion. Based on the Karush-Kuhn-Tucker (KKT) [21,22] conditions of Quadratic Programming (QP) problem, only a number of coefficients $(\alpha_i - \alpha_i^*)$ will be assumed nonzero. Accordingly, the data samples associated with them are referred to as Support Vectors (SVs), which have the approximation errors equal to or larger than ε. According to Eq. (13), it is evident that support vectors are the only samples in training set that are used in determining the prediction function $f(X)$. The prediction function structure of SVM is given in Fig. 4.

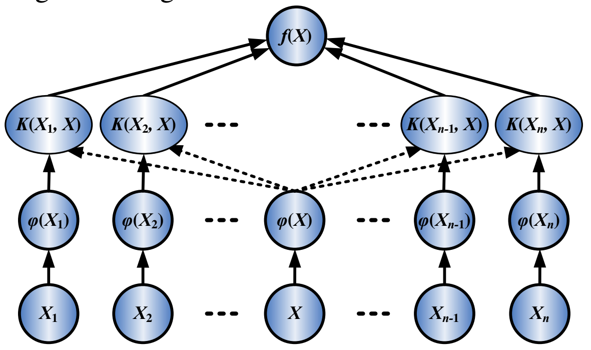


Fig. 4. Prediction function structure of SVM.

3.4. *Kernel functions*

In the prediction model achieved above, kernel function $Q_{ij} = K(X_i, X_j) = \varphi(X_i)^T\varphi(X_j)$ is introduced to compute inner product of augmented samples. It decides the complexity of prediction function, therefore affects the performance of Nu-SVR model. There are four kernel functions frequently used:

(a) Polynomial Function (PF):
$K(X_i, X_j) = (\gamma(X_i^T X_j) + a)^b$,
$\gamma > 0, a \in \mathbb{R}, b \in N$;

(b) Radial Basis Function (RBF):
$K(X_i, X_j) = exp(-\gamma||X_i - X_j||^2), \gamma > 0$;

(c) Sigmoid Function (SF):
$K(X_i, X_j) = tanh(\gamma(X_i^T X_j) - c)$,
$\gamma > 0, c \in \mathbb{R}$;

(d) Linear Function (LF):
$K(X_i, X_j) = X_i^T X_j$.

In general, RBF kernel is a reasonable first choice in training SVM. According to research of Keerthi and Lin [23], RBF kernel with certain parameters can get same performance as LF and SF kernel.

Unlike LF kernel, it nonlinearly maps samples into a higher dimensional space, so it can handle the case when the relation between class labels and attributes is nonlinear. Values of PF kernel may go to infinity or zero when the degree is large, in contrast to RBF kernel of which key point is from zero to one. Therefore, RBF has less numerical difficulties. In addition, it has less hyper-parameters (only γ) which influence the complexity of model selection than PF kernel. Therefore, RBF kernel is our choice in training SVMs. It is clear that once hyper-parameters of model and features of samples are selected, Eq. (13) is achievable by evaluating $(\alpha_i - \alpha_i^*)$ and b. Libsvm toolkit [24] is employed to solve such QP problem.

4. Optimizing Prediction with PH-PSO

Generalization performance of prediction model relies directly on the choice of hyper-parameters. In addition, irrelevant features in samples will also spoil the accuracy and efficiency of model. Besides, hyper-parameter selection and feature selection also correlate with each other. Accordingly, the optimization problem concerning the two is coded with a hybrid vector PR, which consists of real numbers and binary numbers. As Nu-SVR is able to select ε by itself, only C and γ are considered hyper-parameters. The value 1 or 0 for bf_s, respectively, stands for whether the corresponding feature in samples is selected. With $[C_-, C_+], [\gamma_-, \gamma_+]$ as the value intervals and $Fit(\bullet)$ as the target function, the combinational optimization concerning hyper-parameter selection and feature selection jointly can be expressed as:

$$\begin{aligned} & max \quad Fit(PR) \\ \text{s.t.} & \begin{cases} PR = \{C, \gamma, bf_1, bf_2, ..., bf_m\}, \\ C \in [C_-, C_+], \gamma \in [\gamma_-, \gamma_+], \\ bf_s \in \{0,1\}, s = 1,2,...,m. \end{cases} \end{aligned} \tag{14}$$

Because PSO is powerful, easy to implement, and computationally efficient, it is employed and adapted to optimize our prediction model. PSO was proposed by Dr. Kennedy and Dr. Eberhart in 1995 [25], inspired by social behavior of nature system, such as bird flocking or fish schooling. There are mainly two types of PSO distinguished by different updating rules for calculation of particles' position and velocity: continuous version [25,26] and discrete version [27]. Concerning characteristics of our problem, this study proposes a combinational optimization algorithm which hybridizes continuous PSO and discrete PSO together, in hope of improving performance of prediction model, as is explained in details:

The system is initialized with a population of random particles and searches a multi-dimensional solution space for optima by updating particle generations. Each particle moves based on the direction of local best solution discovered by itself and global best solution discovered by the swarm. Each particle calculates its own velocity and updates its position in each iteration until the termination condition is met. Supposing P particles in a D-dimensional search space:

- **(a)** $A_{P\times D}$ denotes the position matrix of all particles, $p = 1,2,...,P, d = 1,2,...,D$, row vector a_p in A denotes the position of the p-th particle, recorded as $a_p = \{a_{p1}, a_{p2}, ..., a_{pD}\}$;
- **(b)** $V_{P\times D}$ denotes the velocity matrix of all particles, row vector v_p in V denotes the velocity of the p-th particle, recorded as $v_p = \{v_{p1}, v_{p2}, ..., v_{pD}\}$;
- **(c)** $LB_{P\times D}$ denotes the local best position of all particles, row vector lb_p in LB denotes the local best position of the p-th particle, recorded as $lb_p = \{lb_{p1}, lb_{p2}, ..., lb_{pD}\}$;
- **(d)** Row vector $gb = \{gb_1, gb_2, ..., gb_D\}$ denotes the global best position shared by all particles.

The particle is represented by the hybrid vector PR. During each iteration the real and binary parts of PR are updated jointly using different rules, namely Eq. (15) and Eq. (16) for real part and Eq. (17) for binary part.

$$\begin{aligned} v_{pd}(t+1) = w \times v_{pd}(t) + \\ c_1 \times rdm1(0,1) \times (lb_{pd}(t) - a_{pd}(t)) + \\ c_2 \times rdm2(0,1) \times (gb_d(t) - a_{pd}(t)). \\ a_{pd}(t+1) = a_{pd}(t) + v_{pd}(t+1). \end{aligned} \tag{15}$$

$$\begin{cases} a_{pd} = \begin{cases} A^{min} & a_{pd} < A^{min}; \\ a_{pd} & A^{min} < a_{pd} < A^{max}; \\ A^{max} & a_{pd} > A^{max}. \end{cases} \\ v_{pd} = \begin{cases} -V^{max} & v_{pd} < -V^{max}; \\ v_{pd} & -V^{max} < v_{pd} < V^{max}; \\ V^{max} & v_{pd} > V^{max}. \end{cases} \end{cases} \tag{16}$$

$$\begin{aligned} v_{pd}(t+1) &= w \times v_{pd}(t) + \\ c_1 \times rdm1(0,1) &\times (lb_{pd}(t) - a_{pd}(t)) + \\ c_2 \times rdm2(0,1) &\times (gb_d(t) - a_{pd}(t)). \\ if \quad &(rdm(0,1) < Sg(v_{pd}(t+1))) \\ then \quad &a_{pd}(t+1) = 1, \\ else \quad &a_{pd}(t+1) = 0; \\ Sg(v) &= \frac{1}{1+e^{-v}}. \end{aligned} \tag{17}$$

The problem-dependent constants A^{min}, A^{max} and V^{max} are defined in order to clamp the excessive roaming of particles, as in Eq. (16). V^{max} is an important parameter. It determines the resolution, or fineness, with which regions between the present position and the target (best so far) position are searched. If V^{max} is too high, particles may fly past good solutions. On the other hand, if V^{max} is too small, particles may be short at exploring ability and trapped in local optima. The relationship between velocity and position is illustrated by Fig. 5.

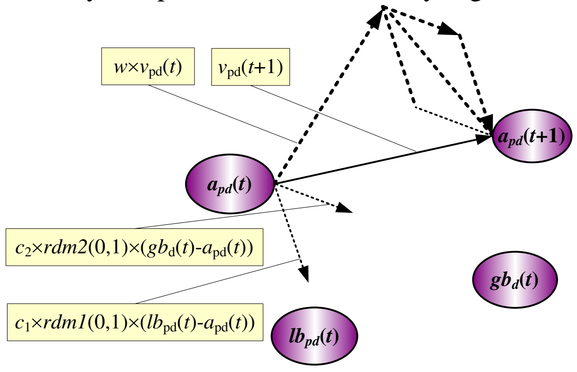

w×v_pd(t)
v_pd(t+1)
a_pd(t+1)
a_pd(t)
c₂×rdm2(0,1)×(gb_d(t)-a_pd(t))
gb_d(t)
c₁×rdm1(0,1)×(lb_pd(t)-a_pd(t))
lb_pd(t)

Fig. 5. Relationship between velocity and position.

$rdm(0,1)$, $rdm1(0,1)$ and $rdm2(0,1)$ are random numbers evenly distributed, respectively, in $[0,1]$. t denotes the step of iteration. Inertia weight w plays the role of balancing global search and local search, it can be a positive constant or even a positive linear/nonlinear function of time. Acceleration constant c_1 and c_2 represent personal and social learning factors, respectively; if $c_1 = 0$, then the particle only has social experience, it may converge fast but fall into local minima easily; if $c_2 = 0$, then the particle only has personal experience, all particles in the swarm become moving by themselves without interaction, thus the probability of finding best solution is very little; if $c_1 = c_2 = 0$, then the particle does not have any experience, all particles in the swarm become disorderly and unsystematic. $Sg(\bullet)$ is a sigmoid function limiting transformation.

Fitness definition. The definition of fitness function is crucial in that it determines what a PSO should optimize. Besides, the particle with high fitness value has high probability to effect other's positions during iterations. Accuracy and efficiency are both concerned in evaluating the fitness of prediction model, in other words, a model is better (with larger fitness) only if it has lower prediction error as well as less training time, thus comes to a relationship of symmetrical inverse proportion. Moreover, when the training time is acceptable, accuracy is considered prior, so we design the fitness function using Eq. (18). Where MSE_t is the training mean squared error, h is a constant controlling the bound of fitness and T_t denotes the model's training time.

$$\begin{aligned} Fitness &= \frac{h}{MSE_t \times \ln T_t} \\ MSE_t &= \frac{1}{l}\sum_{i=1}^{l}(y_i - f(X_i))^2 \end{aligned} \tag{18}$$

Cross-Validation. There are three common ways of calculating MSE_t: cross-validation [17], full-validation [7] and leave-one-out (LOO) [28]. In r-fold cross-validation, the validation set is divided into r subsets of equal size. Sequentially one subset is tested using the model trained on the remaining $(r-1)$ subsets. Thus, each instance of the whole validation set is predicted once so the cross-validation accuracy is the mean squared error in total.

Full-validation is an extreme of the r-fold cross-validation when r is 1. In full-validation, the entire validation set is used as training set, and also used as test set. Whereas, LOO is another extreme of the r-fold cross-validation when r equals l, l is the total sample number of validation set.

In full-validation, model training is performed once only so that computational time is short. However, prediction error is estimated using seen samples, which makes it impossible to reflect model's generalization performance on unseen samples. On the contrary, LOO makes full use of seen samples to predict unseen ones, yet computational time is too long in that model training has to be performed l times. Traditional 5-fold cross-validation is a trade-off between the two, so that it is chosen as error estimation method in fitness calculation.

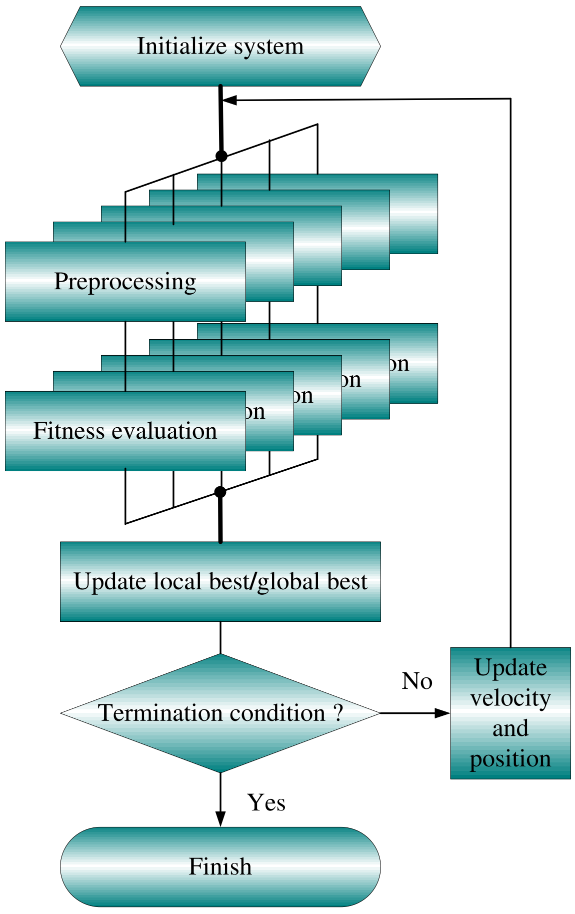


Fig. 6. Flowchart of optimization.

Termination condition. Many conditions can be used to judge termination of algorithm, as is given below, we simply choose the first one for the reason that it is suitable for comparing results among different models.

- **(a)** upper number limit for the whole iteration;
- **(b)** upper time limit for the whole iteration;
- **(c)** upper fitness limit for the whole iteration;
- **(d)** upper number limit for successive iterative steps during which fitness must obviously rise.

Parallel procedure. The parallel optimization for prediction model is shown in Fig.6, with major steps explained as follows:

- **(a)** Initialize system: set parameters for PSO system, including population P, iteration number IT, and hyper-parameter intervals $[C_-, C_+]$ and $[\gamma_-, \gamma_+]$; set parameters for particles such as inertia weight w, personal learning factor c_1 and social learning factor c_2; randomly generate position a_p and velocity v_p for each particle;
- **(b)** Preprocessing in parallel: prepare sample set with corresponding features as well as candidate model with corresponding hyper-parameters according to particle representation;
- **(c)** Fitness evaluation in parallel: use validation set to evaluate candidate model, then calculate fitness for particle according to Eq. (18);
- **(d)** Update local best/global best: if a particle's fitness is better than its local best, update corresponding local best; if a particle's fitness is better than global best, update global best;
- **(e)** Termination judgement: turn to step **(g)** if termination condition is met, otherwise turn to step **(f)**;
- **(f)** Update velocity and position of each particle according to Eq. (15) and Eq. (17), then turn to step **(b)** for next iteration;

(g) Finish: output global best, prepare sample set with selected features and prediction model with selected hyper-parameters according to representation of global best.

5. Experiments and Discussions

After the presentation of modeling and optimizing mechanisms, the following questions have further motivated us for experiments.

○ Is the model feasible for one-step-ahead prediction or multi-step-ahead prediction ?

• *We compare different models for q-step-ahead bandwidth prediction,* $q = 1,2,3,4,5$ *are concerned.*

○ Are the efficiency and accuracy acceptable ?

• *We log parallel/serial CPU time for optimizing models, and employ Mean Absolute Error (MAE) to measure prediction accuracy, as in Eq. (19), where z and* z^* *denote true value and predicted value in original interval. We also implement Back Propagation Neural Network (BPNN) for comparison.*

$$MAE = \frac{1}{l}\sum_{i=1}^{l}|z-z^*| \tag{19}$$

○ Are there remarkable differences among optimizing strategies ?

• *We implement four different strategies including feature selection with hyper-parameter selection (FH), feature selection without hyper-parameter selection (F0), hyper-parameter selection without feature selection (0H), and parameters given directly without any optimization mechanism, same way as in* [5] *(00).*

○ Does the optimizing procedure converge during proper iterations ?

• *We record the iteration number and corresponding global best fitness to evaluate convergence of optimizing procedure.*

5.1. Preparations for experiments

Experiment nodes are running under Fedora Core Linux 9.0 system and connected by 100MB switch-hub. Each node is equipped with single Intel Pentium IV 3.0GHz CPU and 1GB-DDR400Hz MEMORY. One node is used to control the overall optimizing procedure, and the rest nodes are used for fitness evaluation in parallel. The number of nodes used for fitness evaluation is equal to the number of particles in PH-PSO algorithm. The control program and optimizing programs are coded in java, and deployed on different nodes in the form of web service. All the tests are implemented through dynamic collaboration of such services.

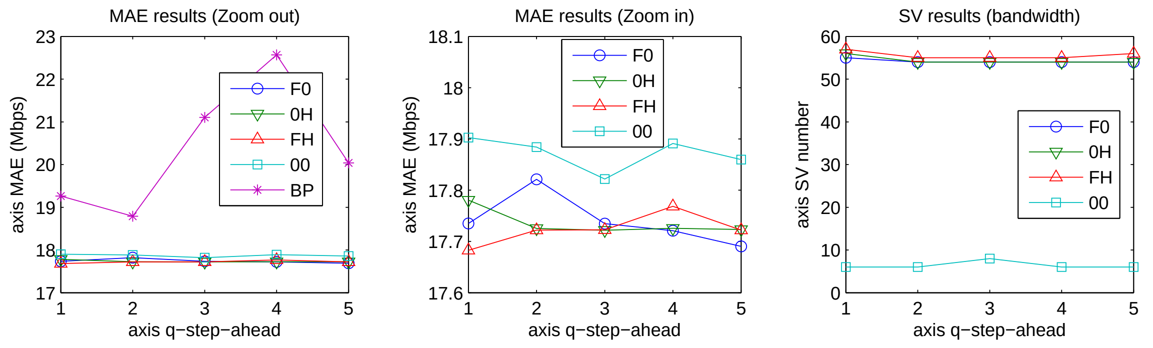
Table 1. Statistics of data sets.

General statistic	value
Set size	1511
Minimum	3.05
Maximum	334.0
Mean	85.14908008
Variance	1932.83648856

For the purpose of giving comparable and reproducible results, we prefer using public data rather than historical data recorded by ourselves. We chose “iepm-bw.bnl.gov.iperf” [29] as benchmark data set. It is published by the Stanford Linear Accelerator Center, University of Stanford. After pretreatment to original data, the latest 200 samples are sequentially chosen to form experiment data set, which is then divided into training set, validation set and test set, with a proportion of 100:50:50. Summary statistics for data set are listed in Table. 1.

Table 2. Parameters initialization.

parameter	value	description
$[lb, up]$	$[0,1]$	destination interval
m	10	number of full features
ν	0.54	extra term for minimizing ε
$[\gamma_-, \gamma_+]$	$[2^{-10}, 2^{10}]$	parameter for RBF kernel
$[C_-, C_+]$	$[2^{-10}, 2^{10}]$	regularized term for SVR
w	$1.4 \rightarrow 0.5$	inertia weight for PSO
c_1, c_2	$2,2$	acceleration constants for PSO
IT	100	iteration times for PSO
P	10	number of particles for PSO
h	0.01	fitness constant for PSO
$bpIn$	10	input number for BPNN
$bpHid$	5	hidden number for BPNN
$bpOut$	1	output number for BPNN
$steps$	500	train steps for BPNN



(a) MAE(Zoom Out) (b) MAE(Zoom In) (c) number of SVs

Figure 7: prediction models.

Parameters are also initialized with values that are commonly used: fix ν based on the research of [30], set acceleration constants c_1 and c_2 according to [25], decrease inertia weight w linearly with time as proposed in [26], and change C, γ exponentially during optimization [31], detailed in Table.2.

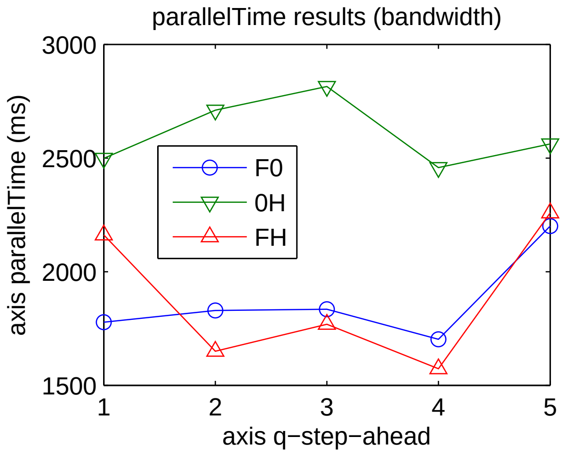
5.2. *Results and discussions*

The MAE results of different models are shown in Fig. 7(a) and Fig. 7(b). For all q-cases, the SVMs achieve better accuracy than BPNN. MAE of SVMs stays below 17.9 Mbps, there is no remarkable difference between one-step-ahead prediction and multi-step-ahead prediction. As prediction step q increases, there is not an obvious ascending trend in MAE on experiment data set, which means that our modeling method is suitable for both one-step-ahead and multi-step-ahead bandwidth prediction. Furthermore, comparing four strategies, the introduction of optimizing mechanism helps to enhance the accuracy of prediction model, especially the combinational optimization FH which achieves lower error in most of the q-cases.

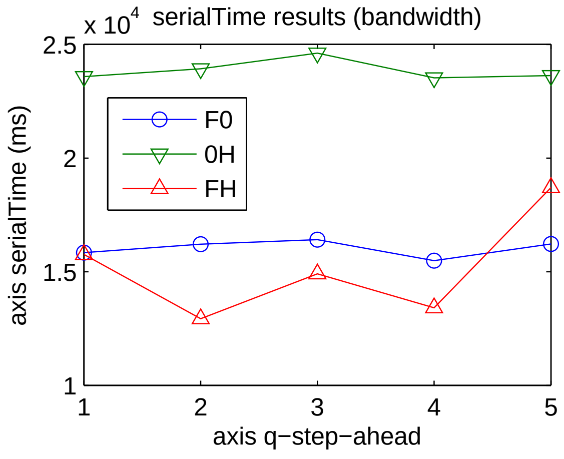
A remarkable characteristic of SVR/Nu-SVR is the sparse representation of the solution, namely model with less support vectors is better in achieving same accuracy. It can be seen from Fig. 7(b) and 7(c) that Nu-SVR models being optimized have higher accuracy than SVR model without optimizing strategy, whereas support vector numbers of Nu-SVR models are over 50 compared to SVR whose number is less than 10. We can see there is a tradeoff between model accuracy and solution sparseness: model with more support vectors are more complicated as well as more capable in characterization.

Four strategies are different in efficiency, the parallel/serial CPU time of each is compared in Fig. 8. From each sub-figure, we can see that the CPU time does not show a remarkable tendency as step q increases. 0H costs more time than FH and F0, which means the model's training time can be obviously reduced by feature selection rather than hyperparameter selection. From comparison between parallel and serial time, it is clear that the introduction of parallelization can remarkably speed up the optimizing procedure, especially the combinational optimization FH within 3 seconds for experiment data set.

The global best fitness during each iteration is logged for convergence comparison, as is shown in Fig. 9. FH wins the best fitness in all the q-cases with 0H as the last, which implies that combinational optimization FH as a whole outperforms individual optimization F0 or 0H. Landscape comparison among different q-cases are shown in Fig. 9(f), no obvious trend on fitness is found when prediction step q increases. From Sub-figures in Fig. 9 we can count times that prematurity happens: 3 times in F0, 5 times in 0H, and 2 times in FH. It is implied that the combinational optimization converges during proper iterations in most of the q cases.

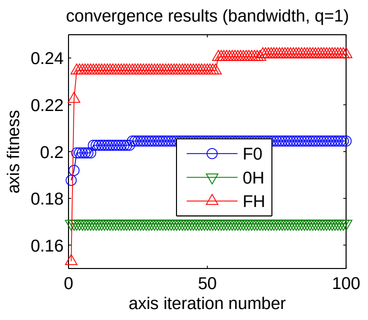


(a) parallelTime

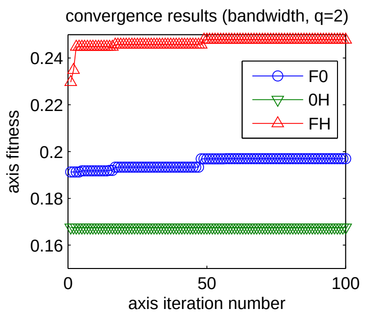


(b) serialTime

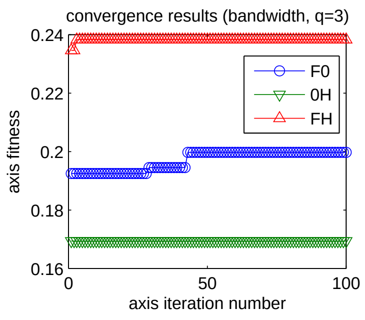
Figure 8: Optimizing time.



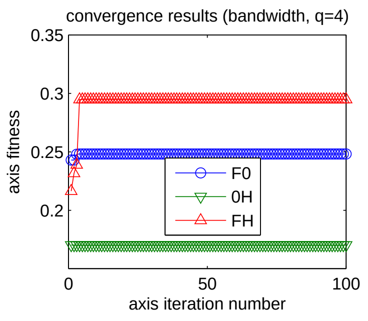
(a) q=1



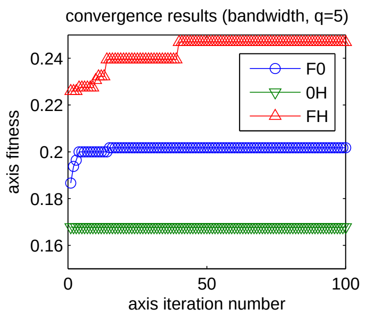
(b) q=2



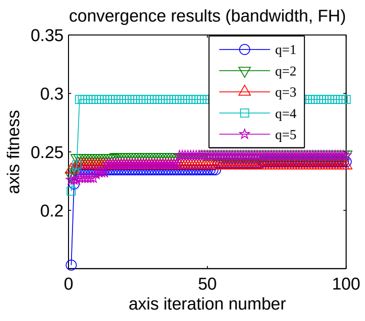
(c) q=3



(d) q=4



(e) q=5



(f) FH landscape

Figure 9: Convergence results.

6. Conclusions and Future works

In this paper, Nu-Support Vector Regression (Nu-SVR) is employed to model one-step-ahead and multi-step-ahead bandwidth prediction. Model optimization issues including hyper-parameter selection and feature selection are also discussed. A Parallel Hybrid Particle Swarm Optimization (PH-PSO) algorithm is proposed to improve the accuracy and efficiency of prediction model.

Prediction results shows that the SVMs achieve better accuracy than BPNN. Mean Absolute Error (MAE) does not show remarkable ascending tendency while prediction step is growing, therefore Nu-SVR is feasible for modeling bandwidth prediction in not only one-step-ahead but also multi-step-ahead settings. Comparative results also indicate that optimizing time can be obviously reduced by feature selection rather than hyper-parameter selection. As a combination of feature selection and hyper-parameter selection, PH-PSO achieves better convergence performance than individual ones. It can improve the accuracy of prediction model in rather short time, namely less than 3 seconds.

Our future work will extend in two ways. First, there are still unsatisfied results that are calling for further improvements; for example, to prevent prematurity in optimizing procedure. Second, considering the diversity of networks, more elements and application instances should be verified to support the feasibility of our methods.

Acknowledgments

This work is funded by: National Natural Science Foundation of China under Grant No. 60873235&60473099 and by Science-Technology Development Key Project of Jilin Province of China under Grant No. 20080318 and by Program of New Century Excellent Talents in University of China under Grant No. NCET-06-0300.

References

1. L. Dai, Y. Xue, B. Chang, Y. Cao, and Y. Cui, "Optimal Routing for Wireless Mesh Networks With Dynamic Traffic Demand," *Mobile Netw Appl.*, **13**, 97-116 (2008).
2. Z. X. Liu, X. P. Guan, and H. H. Wu, "Bandwidth prediction and congestion control for abr traffic based on neural networks," *Lecture Notes in Computer Science*, 3973/2006, 202–207 (2006).
3. C. J. Huang, Y. T. Chuang, W. K. Lai, Y. H. Sun, and C. T. Guan, "Adaptive resource reservation schemes for proportional DiffServ enabled fourth-generation mobile communications system," *Comput. Commun.*, **30(7)**, 1613–1623 (2007).
4. R. Wolski, L. Miller, G. Obertelli, and M. Swany, "Performance Information Services for Computational Grids," *In: Resource Management for Grid Computing, J. Nabrzyski, J. Schopf, and J. Weglarz, editors, Kluwer Publishers, Fall* (2003).
5. H. Prem and N. R. S. Raghavan, "A Support Vector Machine Based Approach for Forecasting of Network Weather Services," *J. Grid Comput.*, **4(1)**, 89–114 (2006).
6. P. A. Dinda, "Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems," *IEEE Trans. Parallel Distrib. Syst.*, **17(2)**, 160–173 (2006).
7. L. Hu and X. L. Che, "Design and Implementation of Bandwidth Prediction based on Grid Service," *Proc. 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08, September 25-27, 2008, Dalian, China)*, 45–52 (2008).
8. J. Yao, S. S. Kanhere, and M. Hassan, "An Empirical Study of Bandwidth Predictability in Mobile Computing," *Proc. 3rd ACM International Workshop on Wireless Network Tesbeds, Experimental Evaluations and Characterization (WINTECH2008) in conjunction with ACM MOBICOM2008, San Francisco, USA, September* (2008).
9. A. J. Nicholson and B. D. Noble, "BreadCrumbs: forecasting mobile connectivity," *Proc. MOBICOM08*, 46–57 (2008).
10. A. Eswaradass, X. H. Sun, and M. Wu, "A neural network based predictive mechanism for available bandwidth," *Proc. 19th International Parallel and Distributed Processing Symposium (IPDPS05, April)*, 33a (2005).
11. J. Zhang and I. Marsic, "Link quality and signal-to-noise Ratio in 802.11 WLAN with fading: A Time-Series Analysis," *Proc. IEEE 64th Vehicular Technology Conference (VTC-2006-Fall), Montreal, Canada, September, "Modeling and Simulation of Mobile Wireless Systems"*, 1–5 (2006).
12. Gowrishankar and P. S. Satyanarayana, "Recurrent neural network based BER prediction for NLOS channels," *Proc. 4th International Conference on Mobile Technology, Applications, and Systems and the 1st In-*

ternational Symposium on Computer Human interaction in Mobile Technology (Mobility'07, Singapore, September 10-12), 410–416 (2007).

13. P. F. Pai, W. C. Hong, P. T. Chang, and C. T. Chen, "The application of support vector machines to forecast tourist arrivals in Barbados: An empirical study," *International Journal of Management*, **23(2)**, 375–385 (2006).
14. V. N. Vapnik, "The Nature of Statistical Learning Theory," *2nd ed, Springer-Verlag, New York* (1999).
15. M. Mariyam, S. Joel, B. Paul, and X. J. Zhu, "A machine learning approach to TCP throughput prediction," *Proc. 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (San Diego, California, USA, June 12-16), SIGMETRICS '07*, 97–108 (2007).
16. Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer TCP throughput," *Proc. SIGCOMM'05 (Philadelphia, Pennsylvania, USA, August 21-26)*, **35(4)**, 145–156 (2005).
17. M. W. Browne, "Cross-validation methods," *Journal of Mathematical Psychology*, **44(1)**, 108–132 (2000).
18. E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, **19(1)**, 43–53 (2005).
19. B. Sch$\ddot{o}$lkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, **12(5)**, 1207–1245 (2000).
20. A. J. Smola and B. Sch$\ddot{o}$lkopf, "A tutorial on support vector regression," *Statistics and Computing*, **14(3)**, 199–222 (2004).
21. W. Karush, "Minima of functions of several variables with inequalities as side constraints," *Masters thesis, Dept. of Mathematics, Univ. of Chicago* (1939).
22. H. W. Kuhn and A. W. Tucker, "Nonlinear programming," *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics (Berkeley)*, 481–492 (1951).
23. S. S. Keerthi and C. J. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel," *Neural Computation*, **15(7)**, 1667–1689 (2003).
24. C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," Available: http://www.csie.ntu.edu.tw/˜cjlin/libsvm/ , May 1 (2008).
25. J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proc. IEEE International Conference on Neural Networks (Perth,Australia)*, **4**, 1942–1948 (1995).
26. Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," *Proc. IEEE International Conference on Evolutionary Computation*, 69–73 (1998).
27. J. Kennedy and R. C. Eberhart, "A Discrete Binary Version of the Particle Swarm Optimization," *Proc. IEEE International Conference on Neural Networks (Perth, Australia)*, **5**, 4104–4108 (1997).
28. K. Fukunaga and D. M. Hummels, "Leave-one-out procedures for nonparametric error estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11(4)**, 421–423 (1989).
29. Bandwidth data set, Available: http://www.slac.stanford.edu/comp/net/iepm-bw.slac.stanford.edu/combinedata/. Aug 01 (2006).
30. A. Chalimourda, B. Sch$\ddot{o}$lkopf, and A. J. Smola, "Experimentally optimal nu in support vector regression for different noise models and parameter settings," *Neural Networks*, **17**, 127–141 (2004).
31. C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," *Department of Computer Science and Information Engineering, National Taiwan University*, Available: http://www.csie.ntu.edu.tw/˜cjlin/papers/guide/guide.pdf (2003).