A Novel Hybrid Method of Parameters Tuning in Support Vector Regression for Reliability Prediction: Particle Swarm Optimization Combined With Analytical Selection

Wei Zhao, Tao Tao, Enrico Zio, and Wenbin Wang

Abstract—Support vector regression (SVR) is a widely used technique for reliability prediction. The key issue for high prediction accuracy is the selection of SVR parameters, which is essentially an optimization problem. As one of the most effective evolutionary optimization methods, particle swarm optimization (PSO) has been successfully applied to tune SVR parameters and is shown to perform well. However, the inherent drawbacks of PSO, including slow convergence and local optima, have hindered its further application in practical reliability prediction problems. To overcome these drawbacks, many improvement strategies are being developed on the mechanisms of PSO, whereas there is little research exploring a priori information about historical data to improve the PSO performance in the SVR parameter selection task. In this paper, a novel method controlling the inertial weight of PSO is proposed to accelerate its convergence and guide the evolution out of local optima, by utilizing the analytical selection (AS) method based on a priori knowledge about SVR parameters. Experimental results show that the proposed ASPSO method is almost as accurate as the traditional PSO and outperforms it in convergence speed and ability in tuning SVR parameters. Therefore, the proposed ASPSO-SVR shows promising results for practical reliability prediction tasks.

Index Terms—Analytical selection, parameter tuning, particle swarm optimization, reliability prediction, support vector regression.

Acronyms and Abbreviations

ARIMA	Autoregressive integrated moving average.
SVM	Support vector machine.
SRM	Structural risk minimization.
ERM	Empirical risk minimization.
SVR	Support vector regression.
AS	Analytical selection.
SA	Simulated annealing.
GA	Genetic algorithm.
PSO	Particle swarm optimization.

Notation

$\boldsymbol{s}_{\blacksquare}$	$\blacksquare$-dimension input vector.
$=^{\blacksquare}$	Real-valued output.
$\shortparallel$	Number of data patterns.
$\boldsymbol{w}$	Weight vector of regression model.
$\equiv$	Intercept of the regression model.
Φ	Nonlinear mapping that translates the complex nonlinear regression problem to a simple linear regression problem.
$\equiv_{\shortparallel}\equiv^{\shortparallel}$	Slack variables representing the deviation outside the $\equiv$-insensitive zone.
$\equiv$	Width of insensitive zone.
C	Penalty coefficient.
γ	Kernel width, a crucial parameter in Kernel function.
$-$	Mean of output values.
$=_{\shortparallel}$	Standard deviation of output values.
$\shortparallel^{+}$	Noise level of the training dataset.
$\boldsymbol{x}$	Parameters vector $\bar{C}_{\shortparallel}\equiv_{\shortparallel}\gamma$.
Fitness	Index to evaluate the quality of a given chromosome or particle.
$p_{\shortparallel}, p_{\shortparallel\shortparallel}$	Probability of crossover and mutation in the GA methodology.
N	Size of the particle swarm.
$\boldsymbol{p}_{\blacksquare}, \boldsymbol{p}_{\blacksquare}$	Best position based on the individual experience and the group experience.
Iter	Current iteration number.
$\boldsymbol{v}$	Velocity vector in the PSO methodology.
$\shortparallel\shortparallel$	Inertial weight in the PSO methodology.
$\bar{\blacksquare}_{\blacksquare}, \bar{\blacksquare}_{\blacksquare}$	Learning coefficients in the PSO methodology.
$\varphi_{\blacksquare}, \varphi_{\bar{\blacksquare}}$	$\varphi_{\blacksquare}$ $\bar{\blacksquare}_{\blacksquare}\bar{\blacksquare}_{\blacksquare}, \varphi_{\bar{\blacksquare}}$ $\bar{\blacksquare}_{\bar{\blacksquare}}\bar{\blacksquare}_{\bar{\blacksquare}}$.
φ	φ $\varphi_{\blacksquare} + \varphi_{\bar{\blacksquare}}$.

Δ	Δ
M	Transit matrix of system status.
	Eigenvalues of M.
η	Recognition degree measuring the probability that the current best position could be the global optimum.
V	Convergence speed.
T	Monotonically mapping describing the relationship between recognition degree and convergence speed.
	Linear coefficient.
	Global optimal position.
	Optimal position estimated by the AS method.
	Mahalanobis distance.

I. Introduction

Due to the increasing complexity in modern processes, systems, and plants, together with the usual necessities of business profitability, safety of humans' life, and protection of the environment, safe and reliable operation of engineering systems is becoming more and more important and has received increasing attention in research and practice. Reliability analysis and risk assessment offer sound technical frameworks for the study of component and system failures, with quantification of their probabilities and consequences [1].Within these frameworks of analysis, one important task is the reliability prediction. In those cases when the historical trend of reliability state is given as a time series of reliability, reliability prediction can be regarded as a time series prediction problem whose solution entails predicting the future values of reliability based on past data observations.

A widely used time series prediction model is the autoregressive integrated moving average model (ARIMA), with solid foundations in classical probability theory. However, the offline modeling efforts required for model identification and construction limit its usefulness in practical applications [2]. In recent years, neural networks have emerged as a universal approximator for any nonlinear continuous function varying over a time or space domain, and have been applied successfully to various reliability problems such as software reliability prediction [3] and complex system maintenance [4]. However, practical difficulties are encountered due to the need of large datasets for training, no guarantee of convergence to optimality, and the danger of over-fitting [5], [6].

Another powerful machine learning paradigm is the support vector machine (SVM) developed by Vapnik and others in 1995 [7], based on statistics learning theory and VC dimension theory (the Vapnik-Chervonenkis dimension theory, which describes the complexity of the concerned learning function). SVM embodies the idea of minimizing the structural risk minimization (SRM) rather than the empirical risk minimization (ERM) adopted in the neural network training. There are two main categories for SVM: support vector classification (SVC) for partitioning discrete label values and support vector regression (SVR) for fitting ordered real-valued samples [8]. Since the ERM principle is most suited for large training datasets, SVR has been proven to provide superior performances than neural networks on small datasets. For this reason, SVR has been applied to many machine learning tasks including time series prediction and reliability forecasting. For example, Hong [9] applied the SVR method to predict engine reliability and compared the predicting performance with the Duane model, ARIMA model, and general regression neural networks. Experimental results show that the SVR model has better performance over the other models.

Parameters selection is very important in SVR, for obtaining accurate regression/prediction. Existing methods of parameter selection for SVR can be divided into two classes. The first class of methods is based on prior knowledge of the analyst on the problem at hand. For example, Cherkassky [10] proposed an analytical selection (AS) method to choose SVR parameters directly from the training data, based on some existing consensus that the SVR parameters are suitably relative to statistical properties of the training data. This expert experience-type of methodology is simple and effective for determining the parameters, provided that the prior knowledge is sufficiently informative. Obviously, in complex problem settings (high dimensional spaces, very nonlinear functions, little representative data, etc.), these methods are not suitable.

The second class of methods searches for the values of the parameters within an optimization scheme defined on specific performance objectives of the algorithm. In general, there are three types of searching methods that can be used for SVR parameter selection:

1) First are the exhaustive methods for searching the best values of the parameters in the entire parameter space. Because the SVR has real-valued parameters, a discretization operation of the search space is first required before the searching. Therefore, some information may be lost and the method may be very time consuming if a refined grid is adopted. A typical exhaustive method is the Grid-Searching method [11], [12], which discretizes the parameter space uniformly and calculates the SVR generalization performance with the parameters set at the values of each grid point.
2) The second class of searching methods comprises the traditional optimization methods including the gradient descent method [13], ellipsoid method [14], and simultaneous perturbation stochastic approximation method [15]. The methods are not easily generalized and perform well only in specific situations.
3) The third class comprises intelligent optimization methods which are powerful searching methods that have emerged rapidly in recent years and have attracted significant attention because of their good performances in various problem settings, even highly complicated. For example, simulated annealing (SA) [16], genetic algorithm (GA) [5], [17], and particle swarm optimization (PSO) [18], [19] have been proposed to search for optimal values of the parameters of SVR applied to system reliability prediction. These methods have become popular in application and GA is perhaps the most frequently used, because of its demonstrated global search efficacy.

In this paper, the performance of AS, GA, and PSO methods in tuning SVR parameters is investigated and compared. Besides, to take simultaneously full advantage of prior knowledge on SVR parameter selection and of the search power of intelligent optimization methods, we propose a novel ASPSO-SVR method for reliability prediction which combines the AS and PSO methods. In the newly proposed method, the parameter values obtained by AS are used to guide the search process of PSO to avoid the local optima and accelerate its convergence. The results obtained in a number of experiments illustrate that the PSO method has a better performance both in searching optimal parameters and robustness than the AS method and GA method, and our ASPSO method is superior to PSO in convergence speed and robustness.

The remainder of the paper is organized as follows. Section II introduces the background knowledge about SVR and the parameter tuning methods, including AS, GA, and PSO, necessary to render the paper self-contained. Section III gives a detailed description about our ASPSO-SVR method. The experiments are presented in Section IV and the results are analyzed therein. Section V provides the conclusions that can be drawn from the findings of our research.

II. Theoretical Background

A. Support Vector Regression (SVR)

In brief, for a dataset D $\{(\boldsymbol{s}_i, y_i)\}_{i=1}^{n}$, where $\boldsymbol{s}_i \in R^m$ denotes the m-dimension input vector, y_i denotes the real-valued output, and n is the number of data patterns, the regression task amounts to finding a function between $\boldsymbol{s}_i$ and y_i, which in the linear case can be described as follows [20]:

$$y = f(\boldsymbol{s}) = \boldsymbol{w}^T \boldsymbol{s} + b \tag{1}$$

where $\boldsymbol{w}$ and b are respectively the weight vector and intercept of the regression model, whose values need to be determined so that the linear function built indeed fits at best the linear relation between the input and the output, as represented by the available dataset.

In the nonlinear case, a nonlinear mapping $\Phi: R^m \to F$, where F is the feature space of Φ, is introduced to translate the complex nonlinear regression problem in R^m to a simple linear regression problem in F. The regression function after this transformation reads

$$y = f(\boldsymbol{s}) = \boldsymbol{w}^T \Phi(\boldsymbol{s}) + b. \tag{2}$$

To evaluate the goodness of the regression function, the ε-insensitive loss function is used [21]:

$$L_\varepsilon(y, f(\boldsymbol{s})) = \begin{cases} 0, & |y - f(\boldsymbol{s})| \le \varepsilon \\ |y - f(\boldsymbol{s})| - \varepsilon, & \text{otherwise.} \end{cases} \tag{3}$$

By ignoring the error if the difference between the estimated value obtained by (2) and the real value is smaller than ε, the ε-insensitive loss function measures the empirical risk. The parameter ε is to be tuned. A procedure is set up for minimizing the empirical risk by introducing the slack variables ξ_i, ξ_i' that represent the deviation of the training data outside the ε-insensitive zone.

Besides minimizing the empirical error by the ε-insensitive loss function, we must also minimize the Euclidean norm of the linear weight $\boldsymbol{w}$, i.e., $\|\boldsymbol{w}\|$, which is related to the generalization ability of the SVM model trained. As a result, a compromised quadratic optimization problem to identify the regression model arises as follows:

$$\begin{aligned} \min_{\boldsymbol{w}, b, \xi, \xi'} \; & J(\boldsymbol{w}, \xi, \xi') = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i') \\ \text{s.t.} \; & y_i - \boldsymbol{w}^T \Phi(\boldsymbol{s}_i) - b \le \varepsilon + \xi_i \\ & \boldsymbol{w}^T \Phi(\boldsymbol{s}_i) + b - y_i \le \varepsilon + \xi_i' \\ & \xi_i, \xi_i' \ge 0 \end{aligned} \qquad i = 1, \ldots, n \tag{4}$$

where C denotes the penalty coefficient that determines the trade-off between empirical and generalization errors, whose value needs to be properly set. Through a Lagrangian dual method, we can obtain the solution of this quadratic optimization problem and estimate the output value as

$$f(\boldsymbol{s}) = \boldsymbol{w} \cdot \Phi(\boldsymbol{s}) + b = \sum_{i=1}^{n}(\alpha_i - \alpha_i')K(\boldsymbol{s}_i, \boldsymbol{s}) + b$$
$$K(\boldsymbol{s}_i, \boldsymbol{s}) = \Phi(\boldsymbol{s}_i)^T \Phi(\boldsymbol{s}) \tag{5}$$

where $K(\boldsymbol{s}_i, \boldsymbol{s})$ is a kernel function satisfying the Mercer condition [22]. If not mentioned specifically, the kernel function used in this paper is the radial basis function with width γ:

$$K(\boldsymbol{s}_i, \boldsymbol{s}) = \exp\left(-\frac{\gamma}{m}|\boldsymbol{s}_i - \boldsymbol{s}|^2\right), \gamma \in R. \tag{6}$$

B. AS Method: Prior Knowledge About the SVR Parameters

In many works on SVR, it is mentioned that the selection of the optimal parameters of the SVR is closely related to the statistical characteristics of the training data. For example, Mattera [23] suggested that a "good" value for parameter C can be chosen equal to the maximum in the range of output values in the training data set and Cherkassky [10] translated this in the following implementation:

$$C = \max(|\bar{y} + 3\sigma_y|, |\bar{y} - 3\sigma_y|) \tag{7}$$

where $\bar{y}$ and σ_y are the mean and the standard deviation of the output values in the training data set.

It is also broadly accepted that the value of ε should be proportional to the input noise level [20], [24]. The choice of ε should also be related to the size of the training dataset: intuitively, for data sets of large size small ε-values should be taken, which led Cherkassky to propose the selection of ε as follows [10]:

$$\varepsilon = 3\sigma\sqrt{\frac{\ln n}{n}} \tag{8}$$

where σ is the noise level in the training dataset estimated by the k-nearest neighbors method.

For the value of the width parameter γ of the kernel function, it is well accepted that it should be selected to reflect the variability range of the input in the training dataset. Considering

univariate inputs, for simplicity of illustration, γ could be, for example, set to

$$\gamma = \frac{1}{2\,\mathrm{avg}(|x_i - x_j|)^2} \quad (9)$$

where $\mathrm{avg}(|x_i - x_j|)$ is the mean distance between pairs of training patterns.

Equations (7)–(9) compose a method of analytic selection (AS) of the values of the SVR parameter triplet, $x_{AS} = (C, \varepsilon, \gamma)$, based on the characteristics of the training data and the estimated noise level.

C. Overview of Genetic Algorithm

GAs are a family of evolutionary computational models inspired by the theory of evolution. These algorithms encode each potential solution of the optimization problem in a simple chromosome-like data structure, and then sift the critical information via some recombination operators that imitate biological evolution processes such as survival of the fittest, crossover, and mutation [25]. The basic procedure of GA method adopted in our work is described as follows [5]:

1) Representation: Chromosome x is directly represented as an SVR parameter vector $x = (C, \varepsilon, \gamma)$.
2) Fitness: The fitness value evaluating the quality of chromosome x is defined as follows:

$$f_{fit}(x) = \frac{1}{RMSE}, \quad RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \quad (10)$$

where RMSE is the root mean squared error also used to describe the prediction accuracy in later sections, y_i is the ith reliability value, $\hat{y}_i$ is its estimate, and n is the size of the sample.

3) Initialization and selection: In this study, the initial population is composed of 40 chromosomes randomly generated within the given range of variability of the three parameters to be tuned, and the standard roulette wheel method is employed to select survival chromosomes from the current population, in proportion to their fitness values.
4) Crossover and mutation: As the core operations of GA, crossover and mutation play a fundamental role in the progress of searching the best chromosome. In our study, the simulated binary crossover and polynomial mutation methods are chosen to realize such operations. The probabilities of crossover p_c and of mutation p_m are set to 0.8 and 0.05, respectively.
5) Elitist strategy: The chromosome with the best fitness skips the crossover and mutation procedure and directly survives to the next iteration.
6) Stopping criteria: steps 3–5 are repeated for a predefined number of iterations (in our application, it is set to 100).

D. Overview of Standard Particle Swarm Optimization

PSO is a population-based meta-heuristics that simulates social behavior such as birds flocking to a promising position [18]. PSO performs searches through a population (called swarm) of individual solutions (called particles) that update iteratively. A particle is regarded as a point in an L-dimension space (in this text, $L = 3$ is the size of the hyper-parameter vector), and it can be described by its position and velocity in this space. The L-dimensional position for particle i at iteration t can be represented as $x_i^t = (x_{i1}, x_{i2}, \ldots, x_{iL})^t$ and likewise, the L-dimensional velocity vector for this particle can be represented as $v_i^t = (v_{i1}, v_{i2}, \ldots, v_{iL})^t$. Then, we can define the status of the swarm at iteration t by $X^t = (x_1^t, x_2^t, \ldots x_N^t)$ and $V^t = (v_1^t, v_2^t, \ldots v_N^t)$, where N is the size of the swarm. For searching the optimal position, each particle changes its position according to its current velocity, and changes its velocity according to its individual best previous position (p_i) and the best previous position of the swarm (p_g). The position and velocity of particle i in iteration t are updated as the following equation shows:

$$\begin{aligned} v_{id}^{t+1} &= w v_{id}^t + c_1 r_1 (p_{id} - x_{id}^t) + c_2 r_2 (p_{gd} - x_{id}^t) \\ x_{id}^{t+1} &= x_{id}^t + v_{id}^{t+1} \end{aligned}, \quad d = 1, 2, \ldots, N \quad (11)$$

where w is the inertial weight, r_1, r_2 are random numbers between 0 and 1, and c_1 and c_2 are positive constants called learning coefficients. In the standard PSO paradigm, w linearly varies with the iteration numbers. The following weighting function is usually used:

$$w = w_{max} - \left(\frac{w_{max} - w_{min}}{Iter_{max}} \right) \cdot Iter \quad (12)$$

where 0.9 is the initial weighting and 0.6 is the final weighting, $Iter_{max}$ is the maximum iteration number, and Iter is the current iteration number.

The performance of each particle is also described by the fitness value, as previously defined. If the best local solution has a higher fitness than the current global solution, then the best local solution replaces the best global solution.

The iteration stops when the maximum iteration time is reached or stopping criterion is met. The stopping criterion used in this paper is stated as follows:

$$\frac{|f_{fit}(x_g^{t+1}) - f_{fit}(x_g^t)|}{f_{fit}(x_g^t)} \le \lambda \quad (13)$$

where λ is a given positive number, here set to 0.05.

In the above three sections, we have briefly overviewed the methods of AS, GA, and PSO. In terms of their performance in tuning SVR parameters, PSO-SVR gives superior performances in prediction accuracy, whereas AS-SVR is easy to implement, but gives comparatively low accuracy and GA-SVR is somewhat unstable as experiments will show in later sections. However, PSO is still not an ideal approach in practical application for some inherent drawbacks, including premature and slow convergence, especially when handling problems with high dimension or multiple local optima.

III. Novel ASPSO-SVR Method

In this section, to overcome the drawbacks of PSO, we originally propose a hybrid method called ASPSO to find the optimal SVR parameters. Differently from all other researches on improving the PSO from its algorithm's mechanisms, the ASPSO method combines the PSO optimization progress with the prior knowledge of the handled data. In the following sections, analysis of the convergence behavior of PSO and the relationship

between data prior knowledge and the inertial weight coefficients of PSO are given, which build the core idea of our ASPSO method.

A. Analysis of the Convergence Behavior in PSO

To begin the analysis, we first shrink the PSO algorithm to a simplest form where only one particle's behavior is investigated and the rest of the particles are assumed static [26]. So, we can drop the subscript identifying the particle and the formula of particle evolution can be rewritten as follows:

$$\begin{cases} \boldsymbol{v}(t+1) = \omega \cdot \boldsymbol{v}(t) + \varphi_1 \cdot (\boldsymbol{p}_1 - \boldsymbol{x}(t)) + \varphi_2 \cdot (\boldsymbol{p}_2 - \boldsymbol{x}(t)) \\ \boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \boldsymbol{v}(t+1). \end{cases} \tag{14}$$

From (14), we can see that the PSO algorithm adjusts the velocity $\boldsymbol{v}$ by two terms: a cognition term and a social term [27]. Both of these two terms are of the same form as $\varphi(\boldsymbol{p} - \boldsymbol{x})$, where $\boldsymbol{p}$ is the best position ever found by the particle's experience in the first term or that found by the population consensus in the second term. So, we can redefine $\boldsymbol{p}$ and simplify the formula to a shortened form as follows:

$$\begin{aligned} \boldsymbol{v}(t+1) &= \omega \boldsymbol{v}(t) + \varphi(\boldsymbol{p} - \boldsymbol{x}(t)) \\ \boldsymbol{p} &= \frac{\varphi_1 \cdot \boldsymbol{p}_1 + \varphi_2 \cdot \boldsymbol{p}_2}{\varphi_1 + \varphi_2} \end{aligned} \tag{15}$$

where $\varphi = \varphi_1 + \varphi_2$, $\varphi_1 = r_1 c_1$, and $\varphi_2 = r_2 c_2$. This equation is algebraically identical to the standard two-term form.

When PSO is used in application, the position of $\boldsymbol{p}$ is constantly updated as $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ evolve towards an optimum. In order to further simplify the system, we set $\boldsymbol{p}$ to a constant in the following analysis. We set also φ to a constant to make the system more understandable, whereas normally it is a random number. Thus, the sufficiently reduced system can be described as

$$\begin{cases} \boldsymbol{v}(t+1) = \omega \cdot \boldsymbol{v}(t) + \varphi \cdot (\boldsymbol{p} - \boldsymbol{x}(t)) \\ \boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \boldsymbol{v}(t+1). \end{cases} \tag{16}$$

By substituting $\boldsymbol{p} - \boldsymbol{x}(t)$ with $\boldsymbol{y}(t)$ and rewriting the iteration index as subscript for convenience, the basic simplified dynamic model is defined by

$$\begin{cases} \boldsymbol{v}_{t+1} = \omega \boldsymbol{v}_t + \varphi \boldsymbol{y}_t \\ \boldsymbol{y}_{t+1} = -\omega \boldsymbol{v}_t + (1 - \varphi) \cdot \boldsymbol{y}_t. \end{cases} \tag{17}$$

Now the problem that the particle converges to the current best position, namely $\boldsymbol{x}_t \to \boldsymbol{p}$, is equivalent to the problem that $\boldsymbol{y}_t \to 0$. Stated in matrix form, the system model is

$$P_{t+1} = M P_t \tag{18}$$

where P_t is the system state vector and M is the transition matrix. P_t and M are defined as follows:

$$P_t = \begin{bmatrix} \boldsymbol{v}_t \\ \boldsymbol{y}_t \end{bmatrix}, \quad M = \begin{bmatrix} \omega & \varphi \\ -\omega & 1 - \varphi \end{bmatrix}. \tag{19}$$

Thus, we have

$$P_t = M^t P_0. \tag{20}$$

That is to say, the convergence behavior of the system is defined completely by M, or more accurately, the eigenvalues of M, which can be represented as follows:

$$\lambda_1, \lambda_2 = \begin{cases} \frac{\omega - \varphi + 1}{2} \pm \frac{\sqrt{\Delta}}{2}, & \Delta > 0 \\ \frac{\omega - \varphi + 1}{2} \pm \frac{\sqrt{-\Delta}}{2} i, & \Delta < 0 \end{cases} \tag{21}$$

where

$$\Delta = (\omega - \varphi + 1)^2 - 4\omega. \tag{22}$$

Here, φ is set to 2.99236 as recommended by Clerc [28] and ω is between 0.9 and 0.6. Thus, Δ is a negative number, so that λ_1, λ_2 are complex numbers.

Then, we can define a matrix A subjected to

$$AMA^{-1} = B = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}. \tag{23}$$

We have

$$P_t = M^t P_0 = A^{-1} B^t A P_0 = A^{-1} \begin{bmatrix} \lambda_1^t & 0 \\ 0 & \lambda_2^t \end{bmatrix} A P_0 \tag{24}$$

where P_0 is the state of initial population.

From (24), it is obviously demonstrated that the convergence of the system is simply related to the modulus of λ_1 and λ_2. We can reasonably define the convergence speed V as

$$V = \omega \in (|\lambda_1|, |\lambda_2|) = \sqrt{\omega}. \tag{25}$$

From (25), we can draw the conclusion that the convergence speed of PSO is related to the inertial weight ω. In order to accelerate the convergence speed of PSO, the next section presents a strategy to choose ω according to the AS method.

B. Prior Information and Inertial Weight

In this section, we first define an index η, named recognition degree, to measure the probability that the current best position $\boldsymbol{p}$ could be the global optimum. The range of η is [0,1] and the global optimum has the highest value of 1. It is reasonably assumed that the particle will more easily converge to the current best when the current best position has higher recognition degree, which is mathematically represented as

$$V = T(\eta) \tag{26}$$

where T is a monotonically increasing mapping function. In the simplest situation, we can set T as a linear function:

$$V = T(\eta) = k\eta \tag{27}$$

where k is the linear coefficient.

In general PSO algorithms, it is not easy to estimate the recognition degree for lack of prior knowledge about the global optimum. However, when PSO is applied to tune SVR parameters, the prior estimation of the optimum deduced from the training data (in this paper, we get it by the AS method) gives an intuitive measurement of the recognition degree.

Considering the simplest and the most pervasive case, we assume that the current best position could be characterized by a Gaussian distribution with mean $\boldsymbol{x}_{best}$ and covariance Σ. $\boldsymbol{x}_{best}$ is the global optimum. The recognition degree of a position could

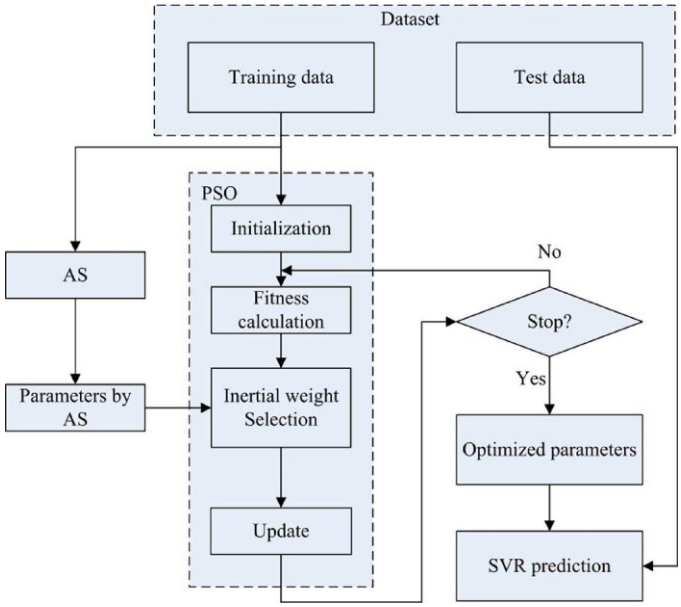

Fig. 1. Flowchart of ASPSO-SVR.

be measured by this probability distribution, which is expressed as follows:

$$\eta(\boldsymbol{p}) \triangleq f(\boldsymbol{p}) \quad \text{where } \eta \text{ is a constant number to make sure } \eta \text{ is between 0 and 1.} \quad (28)$$

In order to calculate η, $\boldsymbol{x}_{\text{initial}}$ could be estimated by the AS method. That is to say, $\boldsymbol{x}_{\text{initial}}$ is set to $\boldsymbol{x}_{AS}$ obtained by (7)–(9). However, there is no theoretical approach to obtain the covariance Σ. Empirically, we set Σ as a unit matrix for normalized $\boldsymbol{p}$ and $\boldsymbol{x}_{\text{initial}}$.

Considering (26)–(28), thus, we have

$$V(\boldsymbol{p}) = \eta(\boldsymbol{p}) \quad \text{where } \eta \text{ is a constant number to make sure } \eta \text{ is between 0 and 1.} \quad (29)$$

where η is a constant number to make sure η is between 0 and 1. Then, we can deduce the expression selecting the inertial weight according to the difference between $\boldsymbol{p}$ and $\boldsymbol{x}_{AS}$:

$$w = \eta \quad \text{where } \eta \text{ is a constant number to make sure } \eta \text{ is between 0 and 1.} \quad (30)$$

where η is a constant number defined by the user, and d_M is the Mahalanobis distance between $\boldsymbol{p}$ and $\boldsymbol{x}_{AS}$.

C. Implementation of ASPSO-SVR

On the basis of (14) and (30), the basic flowchart of ASPSO-SVR is depicted in Fig. 1 and its main steps are as follows:

1) AS estimation. By applying (7)–(9) to the training data, the AS method gives a prior estimate of the optimal SVR parameters, $\boldsymbol{x}_{AS}$.
2) PSO initialization. The initial PSO particles $\boldsymbol{X}^{(0)} = \{\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}, \ldots, \boldsymbol{x}_N^{(0)}\}$ are randomly set within a preset range $C \in [C_{\min}, C_{\max}]$, $\varepsilon \in [\varepsilon_{\min}, \varepsilon_{\max}]$, $\gamma \in [\gamma_{\min}, \gamma_{\max}]$, where N is the population size.
3) Calculation of optimal parameters by PSO.
   For $t = 1, 2, \ldots$:
   a) Fitness calculation. For the population $\boldsymbol{X}^{(t)}$ in the tth iteration, the fitness value of each particle is calculated analogously to what is done for the standard PSO-SVR. Then, we also have the current best positions $\boldsymbol{P}^{(t)} = \{\boldsymbol{p}_1^{(t)}, \boldsymbol{p}_2^{(t)}, \ldots, \boldsymbol{p}_N^{(t)}\}$, where each best position is composed of an individual term $\boldsymbol{p}_i$ with the highest fitness value traversing one particle's experience and a group term $\boldsymbol{p}_g$ with the highest fitness value considering all current particles.
   b) Inertial weight selection. By substituting $\boldsymbol{p}_i^{(t)}$ and $\boldsymbol{x}_{AS}$ to (30), the inertial weight vector, $\boldsymbol{W}^{(t)} = \{w_1^{(t)}, w_2^{(t)}, \ldots, w_N^{(t)}\}$, is selected.
   c) Update. Based on (14), the particles' state of current iteration $\boldsymbol{X}^{(t)}$ is updated to $\boldsymbol{X}^{(t)} + \boldsymbol{v}^{(t)}$.
   d) Interception. Steps a)–c) are repeated until t reaches the defined maximum number of iterations (in our application this threshold is set to 100) or a defined stopping criterion as (13) is met.
4) SVR prediction: With the optimal parameters obtained by PSO, we can predict the future reliability values by the SVR model.

IV. Experimental Studies

In this section, some experiments are analyzed to verify the performance of our proposed ASPSO-SVR method. Firstly, we look at a problem of regression of an artificial function, which holds similar characteristics to the problem of reliability prediction while, on the other hand, is easily implemented and controllable for experimentation; then, we apply our method to forecast real reliability data. A single-step-ahead prediction model is considered in this case: that is, each sample y_t is regarded as a function of the sample in the previous time step. Then, the relationship between the prediction at the current time and the input at the previous time is described by the following equation.

$$y_t = f(y_{t-1}). \quad (31)$$

Though different form of data is concerned in the reliability forecasting case, essentially, it is a generalized application of regression problems. It is to say, the ability of SVR in addressing reliability forecasting problems is essentially a generalization of its basic ability in regression. That is the reason that the regression performance of the various methods is investigated first in the experimental section.

Through these experiments, we can systematically compare the prediction performance of the proposed ASPSO method with benchmark methods like AS, GA, and PSO; the comparison includes aspects of accuracy, stability, and convergence speed.

A. Study of Artificial Function Data

The first group of experiments is based on artificial uni-dimensional datasets. Each dataset used in the experiments described in this section is built by taking output values of a chosen function in correspondence of inputs with additive noise. The experiments are here presented progressively.

Case 1: In the first case, we consider a sinc function as

$$f(x) = \frac{\sin(x)}{x}. \quad (32)$$

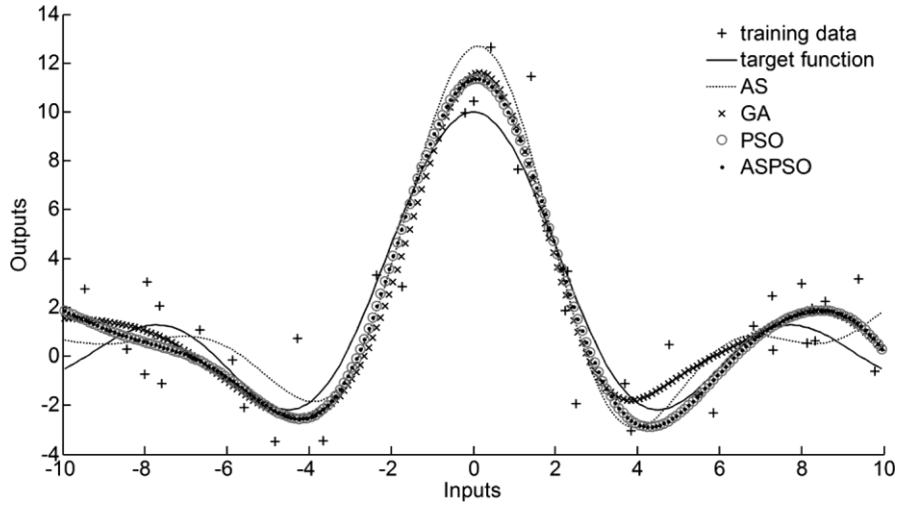

Fig. 2. Regression results of different SVR parameters selection method.

TABLE I
RMSE FOR DIFFERENT FUNCTION TYPES WITH DIFFERENT NOISE LEVELS

Noise level	Methods	RMSE										Mean	Standard deviation
Target function : $y = s$													
$\sigma = 1$	AS	0.2209	1.2387	0.6475	1.3284	0.13303	0.4534	0.1034	0.7326	0.5691	0.2631	0.5690	0.4337
	GA	0.2163	0.9306	0.7536	0.9631	0.1657	0.3780	0.2173	0.2887	0.5903	0.9614	0.5465	0.3325
	PSO	0.1602	0.8115	0.4454	0.7543	0.2547	0.3529	0.2810	0.2143	0.6128	0.1906	0.4077	0.2391
	ASPSO	0.1608	0.7931	0.4603	0.7544	0.2554	0.3817	0.2724	0.2097	0.5977	0.2136	0.4099	0.2327
$\sigma = 2$	AS	**0.83162**	1.9473	2.6628	1.9854	2.5502	2.2687	0.0935	1.5936	2.0364	**1.5419**	**1.751142**	0.7869
	GA	4.3132	1.8693	2.5218	0.2298	6.2979	7.1492	2.9968	3.7956	4.0238	1.9943	3.51917	2.0815
	PSO	0.3101	0.1397	1.2607	0.0648	0.4941	0.2232	0.0097	0.4963	0.1321	0.2031	**0.33338**	0.3646
	ASPSO	0.3095	0.1411	1.0135	0.0852	0.4991	0.2238	0.0109	0.5107	0.1317	0.1978	0.31233	0.2965
Target function : $y = s^2 + s + 1$													
$\sigma = 5$	AS	7.6928	49.5629	10.8916	17.3639	12.9373	22.2204	42.8112	9.1254	**19.2354**	**20.3694**	**21.2210**	14.1208
	GA	11.3244	11.6291	4.4968	9.2407	18.4272	12.5531	31.2114	9.3786	**16.3451**	**11.4197**	13.6026	7.2694
	PSO	8.5105	7.3955	2.2536	2.7347	14.5867	10.8193	10.6673	7.9917	3.5328	11.2160	7.9708	4.0839
	ASPSO	7.9836	7.5567	3.0125	2.4912	**8.9963**	8.7954	**8.8852**	9.6321	8.7427	5.6566	7.1752	2.5762
$\sigma = 10$	AS	72.7756	37.8415	23.4862	39.4547	**66.1863**	129.980	**55.4059**	**49.5631**	**56.2038**	**33.7417**	**56.4638**	29.9146
	GA	70.7827	**18.6030**	12.7345	**16.2283**	128.013	64.0742	5.4549	**13.8893**	**56.3145**	**62.1987**	**44.8293**	38.6445
	PSO	**81.6381**	27.4305	15.6488	11.9962	42.3675	61.2897	8.5596	**20.3741**	**40.8896**	**45.1744**	**35.5368**	23.4042
	ASPSO	75.2345	30.0126	16.7832	12.0362	40.1973	58.1364	9.2082	**12.9826**	**25.3647**	**30.1796**	**31.0135**	21.5311
Target function : $y = \sin(s)$													
$\sigma = 0.5$	AS	0.3734	0.3047	0.3123	0.3381	0.4152	0.3976	0.3241	0.3126	0.3354	0.3099	0.3423	0.0394
	GA	0.0842	0.08595	0.1429	0.0672	0.1814	0.1289	0.0849	0.0712	0.1230	0.9643	0.1933	0.2732
	PSO	0.1190	0.0904	0.1620	**0.08235**	0.1557	0.1307	0.0847	0.1023	0.0951	0.1273	0.1149	0.0287
	ASPSO	0.1056	0.0917	0.1539	0.0904	0.1467	0.1356	0.0835	0.0739	0.1011	0.1301	0.1112	0.0281
$\sigma = 0.25$	AS	0.3274	0.1761	0.2637	0.4132	0.1814	0.2005	0.2142	0.3961	0.3026	0.2718	0.2747	0.0851
	GA	0.0723	0.0248	0.0125	0.1101	0.0338	0.0308	0.0640	0.0958	0.0233	0.0495	0.0516	0.0329
	PSO	0.0379	0.0168	0.0085	0.0675	0.0490	0.0227	0.0671	0.0377	0.0812	0.0231	0.0411	0.0244
	ASPSO	0.0411	0.0201	0.0094	0.0723	0.0506	0.0215	0.0649	0.0361	0.0213	0.037	0.0374	0.0204

The simulated training data are pairs (s_i, y_i), where the input values s_i are random-uniformly sampled in the range $[-10, 10]$ and the output values y_i are generated as

$$y_i = f(s_i) + \varepsilon_i \tag{33}$$

where ε_i is the Gaussian additive white noise. In this case 1, we set the noise standard deviation (standard) and . The test data are also random-uniformly sampled as the training data. Fig. 2 shows the regression results of the four methods considered. We can see that the outputs of all methods approximate the target function values, with GA, PSO, and ASPSO yielding better generalization.

Case 2: In the second case, we consider different target functions and noise levels. To compare the prediction accuracy of the four methods objectively, we use the RMSE between the SVR estimates and the corresponding true values of the target function for the test input values. To account for the randomness of the estimation process, we perform the regression ten times and calculate the mean value and the variance of RMSE for each target function and noise level. The results of the repeated experiments are reported in Table I.

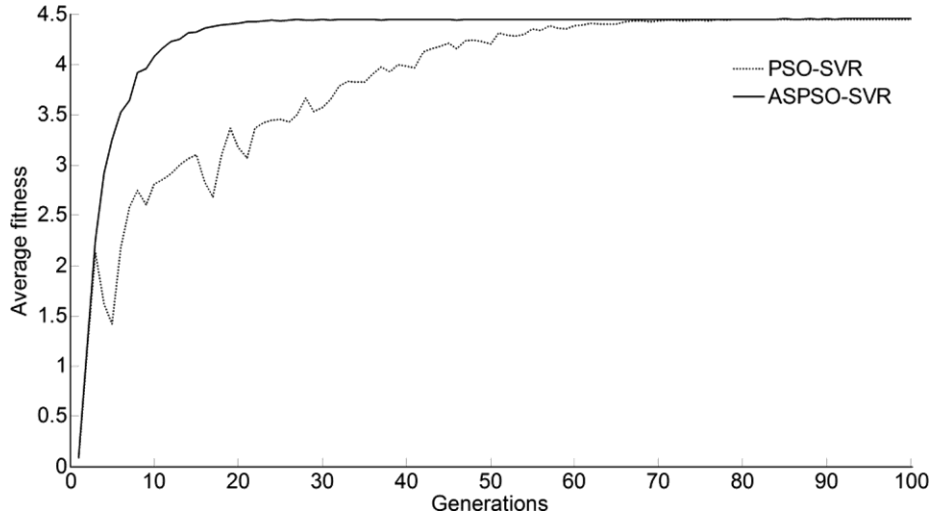

Fig. 3. Average fitness curves of PSO and ASPSO.

TABLE II
COMPARISON OF CONVERGENCE PERFORMANCE OF PSO-SVR AND ASPSO-SVR

Experiment Index	Iteration Number	
	PSO-SVR	ASPSO-SVR
1	55	21
2	58	23
3	71	19
4	48	11
5	51	22
Mean	56.6	19.2
Standard Deviation	8.9050	4.8167

Generally, as expected, the GA, PSO, and ASPSO methods perform better than the AS method. Further, the RMSE mean value and standard deviation of the GA method tends to become large as the noise level increases: this shows the GA method instability and sensitivity to noise. On the contrary, for all function types and noise levels considered, the PSO method performs satisfactorily in both mean value and standard deviation. The ASPSO-SVR has almost the same performance of the PSO-SVR, due to the similarity of the optimization mechanisms.

Case 3: Although PSO possesses a comparative advantage over AS and GA in generalization ability and stability as above illustrated, the slow convergence speed of PSO is still a challenge for its practical application. In this case 3, the convergence performance of PSO and ASPSO is investigated. The dataset used in this case is the same as that used in case 1.

To look into the convergence performance, the curve of the average fitness (regarded as a function of the iteration number) is given in Fig. 3. Here, the stop criterion in (13) is not applied because what we are investigating is the trend with which the average fitness varies with the iteration number, and each searching process is left to last until reaching the maximum iteration number, 100. The results show that our ASPSO can effectively improve the convergence progress of standard PSO: the two methods have a similar fitness level at convergence, which means similar generalization ability, but ASPSO converges to "a good solution" much faster than standard PSO method.

Table II shows the experimental results that the stop criterion is concerned. It gives the iteration numbers until the stop criterion is met for five repeated experiments, which fully illustrates that our ASPSO-SVR has significantly reduced the time for optimally tuning the SVR parameters: considering that the time for AS estimation and III selection in ASPSO is negligible compared to that of one searching iteration, we can reasonably deduce that less iterations lead to less computational burden.

B. Reliability Prediction

In this section, we apply the ASPSO-SVR for predicting real reliability data taken from literature cases, and compare its prediction accuracy with GA-SVR and standard PSO-SVR.

Case 1: The first case study concerns the forecasting of the time-to-failure of turbochargers of a specific type. The data comprise the time-to-failure for 40 suits of turbo chargers. Out of the set of 40 data, 35 samples are used as training data and the remaining five samples as test data, as adopted in previous studies [18], [19], [29], [30]. The initial population is randomly generated for GA, PSO, and ASPSO. The corresponding predictive outputs of the three search methods are illustrated in Fig. 4. To account for the stochasticity inherent in the GA and PSO search, the optimization is repeated 10 times and the RMSE of the results are listed in Table III.

The results of Fig. 4 and Table III show that the two PSO methods are comparable in prediction accuracy and superior to the GA method, on average. Also, the PSO-based methods are more stable than GA, if we look at the dispersion of listed ten RMSEs.

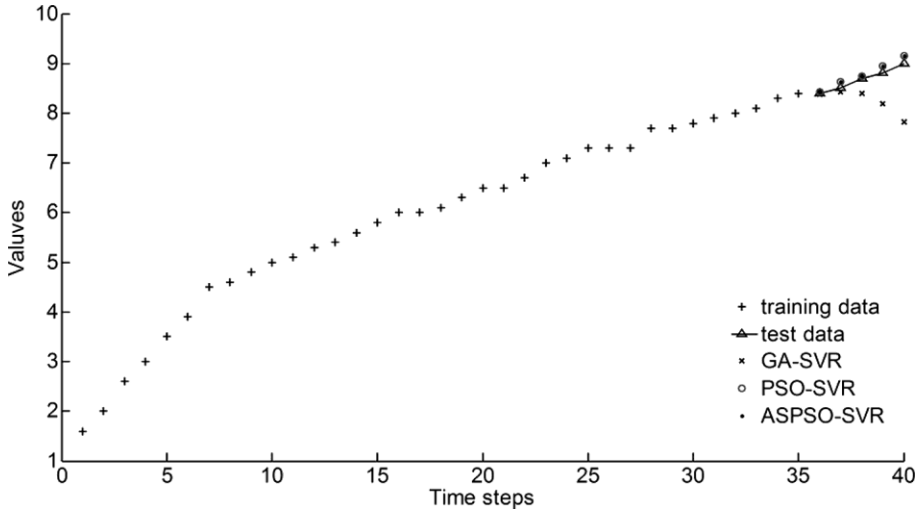

Fig. 4. Prediction results of GA-SVR, PSO-SVR, and ASPSO-SVR in forecasting the turbochargers failure data.

TABLE III
PREDICTION ACCURACY OF GA-SVR, PSO-SVR, AND ASPSO-SVR IN RELIABILITY PREDICTION CASE 1

Experiments Index	*RMSE*		
	GA-SVR	PSO-SVR	**ASPSO-SVR**
1	0.0106	0.0127	0.0124
2	0.3659	0.0131	0.0125
3	0.5280	0.0128	0.0128
4	0.0253	0.0126	0.0126
5	0.0068	0.0128	0.0125
6	0.9592	0.0129	0.0127
7	0.0159	0.0132	0.0123
8	0.0451	0.0126	0.0119
9	0.2839	0.0131	0.0126
10	0.1179	0.0128	0.0127
Mean	0.2359	0.0129	0.0125
Standard Deviation	0.3116	0.00021	0.00026

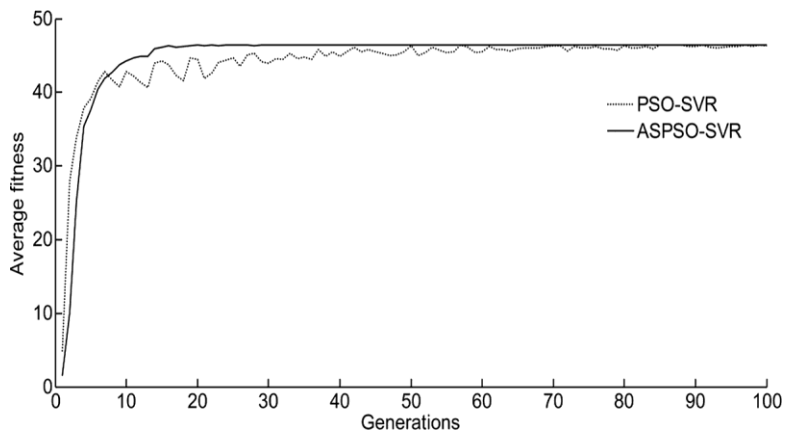

Fig. 5. Average fitness curves of PSO-SVR and ASPSO-SVR in reliability prediction case 1.

Next, we want to compare the convergence speed of the two PSO methods. Fig. 5 represents the average fitness of these two methods: it is seen that, again, the average fitness of ASPSO reaches an optimal and stable value faster than PSO. The running times of GA-SVR, PSO-SVR, and ASPSO-SVR when the premise criterion is met are reported in Table IV. We repeated the simulation for 10 times with Microsoft Windows 7, Matlab 7.9.0 on Intel 2.4 GHz. From Table IV, the evident improvement in the running time of our ASPSO-SVR is shown. The reason for this improvement is that the ASPSO method can effectively accelerate the convergence of PSO, which means less iterations and this compensates some computationally negli-

TABLE IV
RUNNING TIME OF GA-SVR, PSO-SVR, AND ASPSO-SVR IN RELIABILITY PREDICTION CASE 1

Experiments Index	Running time (seconds)		
	GA-SVR	PSO-SVR	**ASPSO-SVR**
1	823.1	513.4	79.3
2	396.7	490.8	88.5
3	528.5	499.0	76.8
4	677.9	528.9	82.3
5	1021.6	505.7	84.9
6	633.8	521.3	87.8
7	752.1	506.8	81.6
8	596.4	477.6	84.1
9	917.2	489.5	93.3
10	846.5	499.7	87.9
Mean	719.4	503.3	84.7

TABLE V
PREDICTION ACCURACY OF GA-SVR, PSO-SVR, AND ASPSO-SVR IN RELIABILITY PREDICTION CASE 2

Experiments Index	*RMSE*		
	GA-SVR	PSO-SVR	**ASPSO-SVR**
1	3.2773	0.1025	0.0877
2	5.2770	0.0873	0.0881
3	0.1149	0.1028	0.0896
4	1.2547	0.0874	0.0901
5	3.3135	0.1009	0.0876
6	2.7140	0.1033	0.1006
7	3.9856	0.1524	0.0874
8	0.2533	0.0889	0.0877
9	0.6039	0.0875	0.0880
10	1.8492	0.1073	0.0769
Mean	2.2643	0.1020	0.0884
Standard Deviation	1.7314	0.0194	0.0057

TABLE VI
RUNNING TIME OF GA-SVR, PSO-SVR, AND ASPSO-SVR IN RELIABILITY PREDICTION CASE 2

Experiments Index	Running time (seconds)		
	GA-SVR	PSO-SVR	**ASPSO-SVR**
1	1206.8	1018.4	256.7
2	1336.5	983.8	341.6
3	976.4	917.5	329.4
4	1562.8	1430.7	316.0
5	1021.6	1128.9	405.8
Mean	1220.82	1095.86	329.9

gible time spent for AS operations. With such time-saving advantage, the ASPSO-SVR can be fitter for practical reliability prediction tasks.

Case 2: The second literature case study comes from the observation of unscheduled maintenance actions for a submarine diesel engine undergoing a deterioration process [31]. The first 60 samples are used as training data and the remaining 10 samples as test data. Other settings are same as in case 1. Experimental results about prediction accuracy and running time are listed in Tables V and VI.

From Tables V and VI, we can draw the same conclusion as in case 1, which is that the ASPSO-SVR can accelerate the parameters tuning with no less of accuracy and more robustness.

Case 3: In the previous two real cases, both the concerned reliability data are of obviously linear trend. For more comprehensive illustration, the case concerning the reliability data without clear trend is required. Therefore, a reliability series consisting of the reliability data of a car engine is introduced. In this case, distance between two unscheduled and consecutive corrective maintenance times is considered as a reliability indicator of the car engine. The data of 100 engines are treated as a time series. Figs. 6 and 7 give the prediction results and average fitness curves when the first 90 samples are used as training data and the rest as test data. Then, the prediction accuracy and running time of plenty of repeated experiments are also listed in Tables VII and VIII. In this case, though the data concerned

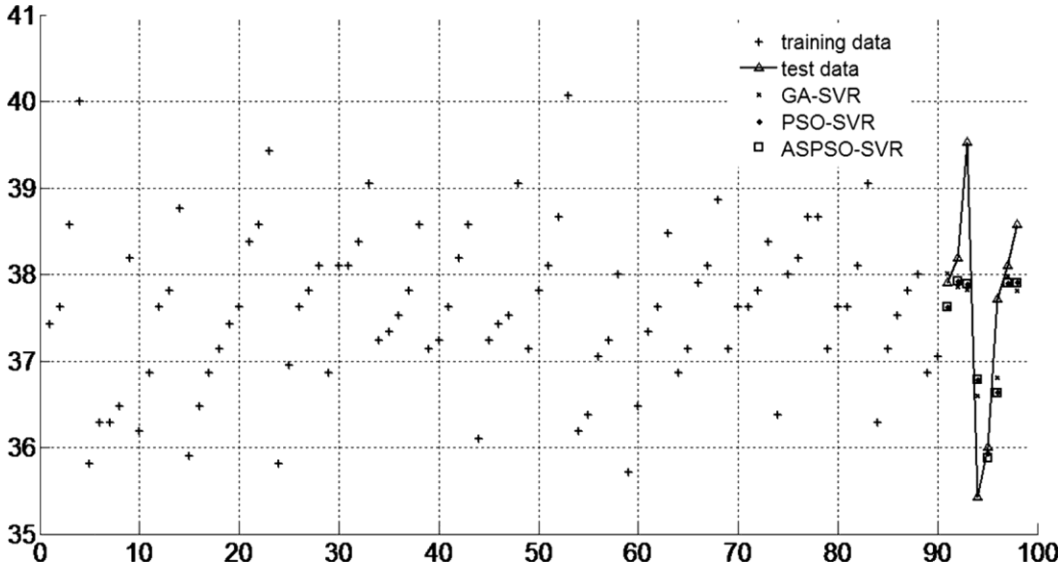

Fig. 6. Prediction results of GA-SVR, PSO-SVR, and ASPSO-SVR in reliability prediction case 3.

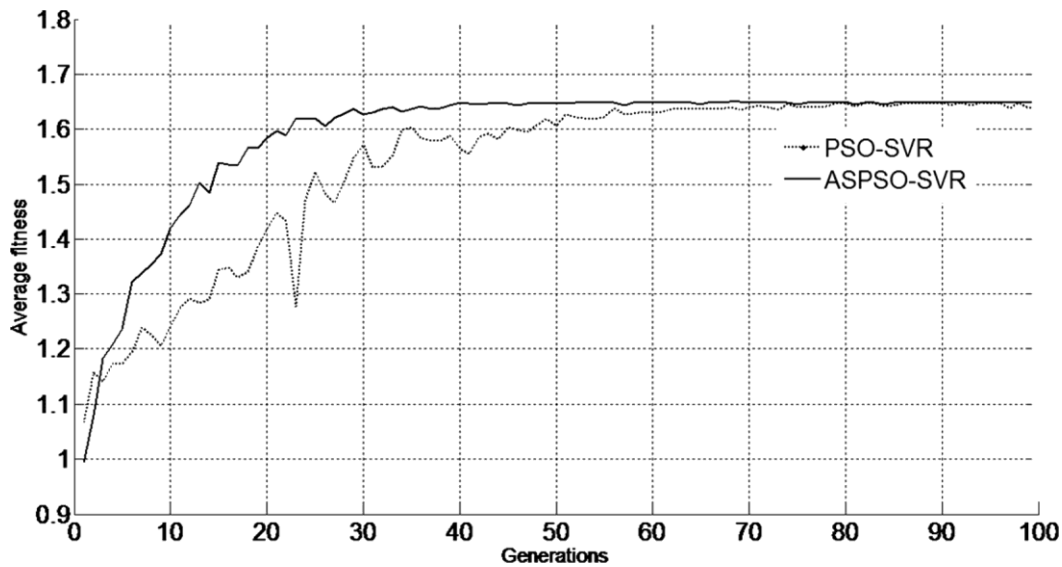

Fig. 7. Average fitness curves of PSO-SVR and ASPSO-SVR in forecasting the reliability data of a car engine.

TABLE VII
PREDICTION ACCURACY OF GA-SVR, PSO-SVR, AND ASPSO-SVR IN RELIABILITY PREDICTION CASE 3

Experiments Index	*RMSE*		
	GA-SVR	PSO-SVR	**ASPSO-SVR**
1	0.7703	0.8199	0.7582
2	1.2737	0.8865	0.8628
3	1.3585	0.8872	0.8865
4	1.0231	0.8881	0.8763
5	1.1258	0.7998	0.8129
6	1.7933	0.8237	0.8433
7	0.9384	0.8699	0.8216
8	1.0276	0.9124	0.7843
9	1.2183	0.8671	0.7965
10	0.6119	0.8890	**0.8933**
Mean	1.1141	0.8644	**0.8336**
Standard Deviation	0.3291	0.0370	0.0462

are with no longer linear trend, it also shows the similar experimental results.

It is worth noting that we have also performed the experiments for larger number of generations and individuals for GA

TABLE VIII
RUNNING TIME OF GA-SVR, PSO-SVR, AND ASPSO-SVR IN RELIABILITY PREDICTION CASE 3

Experiments Index	Running time (seconds)		
	GA-SVR	PSO-SVR	ASPSO-SVR
1	1398.5	1217.3	412.8
2	1571.3	1472.7	349.7
3	1459.9	1385.1	515.4
4	1613.8	1430.7	498.6
5	1415.8	1517.4	503.5
Mean	1491.86	1404.64	456
Standard Deviation	95.790	115.655	72.051

TABLE IX
PREDICTION ACCURACY OF GA-SVR WITH LARGE INDIVIDUALS AND GENERATIONS IN RELIABILITY PREDICTION CASE 1, 2, 3, COMPARED WITH PSO-SVR AND ASPSO-SVR METHOD

Experiments Index		*RMSE*		
		Case 1	*Case 2*	*Case 3*
GA-SVR	1	0.2202	0.2373	0.8545
	2	0.1860	0.2074	0.8897
	3	0.2396	0.2088	0.8336
	4	0.1261	0.2074	0.8813
	5	0.2287	0.2328	0.7972
	6	0.1588	0.2267	0.8786
	7	0.2193	0.3768	0.9192
	8	0.1334	0.3283	0.8639
	9	0.1308	0.2476	0.8506
	10	0.1594	0.2685	0.9608
		Means		
GA-SVR		0.1802	0.2542	0.8729
PSO-SVR		0.0129	0.1020	0.8644
ASPSO-SVR		0.0125	0.0884	0.8171

in all three above-mentioned reliability case studies. The experimental results show that when these parameters change, GA becomes much more stable and accurate, at cost of much heavier computational burden, however. After all, the performance of GA is still poor compared with PSO and ASPSO methods in tuning SVR parameters. That is the reason why the number of individuals and iterations are not very large in our experiments. Table IX lists the RMSE of GA with 100 individuals and 500 generations for 10 repeated experiments, which is quite large for real-time requirement.

V. CONCLUSION

In this paper, a novel ASPSO-SVR scheme is proposed for solving reliability prediction problems. Differently from other improved PSO algorithms, the proposed scheme utilizes the prior knowledge of SVR for the selection of inertial weight in the PSO method. Based on mathematical deductions, a strategy of adapting the inertial weight by comparing the current particles knowledge with the prior SVR knowledge is proposed. Because of the adaptability of the inertial weight, the ASPSO-SVR scheme has superior prediction performance over that of traditional GA-SVR and standard PSO-SVR, as demonstrated in the case studies based on both artificial and real data. The results obtained in these case studies show that the standard PSO-SVR and ASPSO-SVR have comparable performances in prediction accuracy and robustness ability, both of which are better than GA-SVR. But in terms of convergence speed, our ASPSO-SVR shows a significant advantage. Due to the properties of computational speed and robustness, the ASPSO method is fitter for the practical reliability prediction tasks than the GA method or the standard PSO method for tuning SVR parameters. In the future research, more reliability applications will be considered to further investigate the detailed performance of the ASPSO-SVR method, and improvements in describing the prior knowledge of SVR and integrating it within intelligent searching processes will be explored.

REFERENCES

[1] E. Zio, "Reliability engineering: Old problems and new challenges," Rel. Eng. Syst. Safety, vol. 94, pp. 125–141, 2009.

[2] H. Lu, W. J. Kolarik, and S. S. Lu, "Real-time performance reliability prediction," IEEE Trans. Rel., vol. 50, no. 4, pp. 353–357, Dec. 2001.

[3] W. Adnan and M. Yaacob, "An integrated neural-fuzzy system of software reliability prediction," in Proc. 1st Int. Conf. Software Testing, Reliability and Quality Assurance, 1994, Dec. 21–22, 1994, pp. 154–158.

[4] N. Amjady and M. Ehsan, "Evaluation of power systems reliability by an artificial neural network," IEEE Trans. Power Syst., vol. 14, no. 1, pp. 287–292, Feb. 1999.

[5] K. Y. Chen, "Forecasting systems reliability based on support vector regression with genetic algorithms," Rel. Eng. Syst. Safety, vol. 92, pp. 423–432, 2007.

[6] N. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," IEEE Computat. Intell. Mag., vol. 4, no. 2, pp. 24–38, May 2009.
[7] V. Vapnik, The Nature of Statistical Learning Theory. New York, NY, USA: Springer-Verlag, 1995.
[8] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," Statist. Comput., vol. 14, pp. 199–222, 2004.
[9] W. C. Hong and P. F. Pai, "Predicting engine reliability by support vector machines," Int. J. Adv. Manuf. Technol., vol. 28, pp. 154–161, 2006.
[10] V. Cherkassky and Y. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," Neural Netw., vol. 17, pp. 113–126, 2004.
[11] C.-C. Chang and C.-J. Lin, "Training v-support vector classifiers: Theory and algorithms," Neural Computat., vol. 13, pp. 2119–2147, 2001.
[12] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, A Practical Guide to Support Vector Classification, 2003 [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.224.4115
[13] F. Guely and P. Siarry, "Gradient descent method for optimizing various fuzzy rule bases," in Proc. 2nd IEEE Int. Conf. Fuzzy Systems, 1993, 1993, vol. 2, pp. 1241–1246.
[14] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," Combinatorica, vol. 1, pp. 169–197, 1981.
[15] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," IEEE Trans. Aerosp. Electron. Syst., vol. 34, no. 3, pp. 817–823, Jul. 1998.
[16] P. F. Pai and W. C. Hong, "Software reliability forecasting by support vector machines with simulated annealing algorithms," J. Syst. Softw., vol. 79, pp. 747–755, 2006.
[17] W.-C. Hong, Y. Dong, L.-Y. Chen, and S.-Y. Wei, "SVR with hybrid chaotic genetic algorithms for tourism demand forecasting," Appl. Soft Comput., vol. 11, pp. 1881–1890, 2011.
[18] S. W. Lin, K. C. Ying, S. C. Chen, and Z. J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," Expert Syst. Applicat., vol. 35, pp. 1817–1824, 2008.
[19] I. D. Lins, M. C. Moura, E. Zio, and E. L. Droguett, "A particle swarm-optimized support vector machine for reliability prediction," Qual. Rel. Eng. Int., vol. 28, pp. 141–158, 2012.
[20] A. Smola and B. Schölkopf, "A tutorial on support vector regression," Statist. Comput., vol. 14, pp. 199–222, 2004.
[21] V. N. Vapnik, The Nature of Statistical Learning Theory. New York, NY, USA: Springer-Verlag, 1995.
[22] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in Proc. 5th Annu. Workshop Computational Learning Theory, 1992, pp. 144–152.
[23] D. Mattera and S. Haykin, "Support vector machines for dynamic reconstruction of a chaotic system," in Advances in Kernel Methods. Cambridge, MA, USA: MIT Press, 1999, pp. 211–241.
[24] J. Kwok, "Linear dependency between ε and the input noise in ε-support vector regression," in Artificial Neural Networks — ICANN 2001, G. Dorffner, H. Bischof, and K. Hornik, Eds. Berlin, Germany: Springer, 2001, pp. 405–410.
[25] D. Whitley, Genetic Alg. Tutorial. Statist. Comput., vol. 4, pp. 65–85, 1994.
[26] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," IEEE Trans. Evol. Computat., vol. 6, no. 1, pp. 58–73, Feb. 2002.
[27] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in Proc. 1998 IEEE Int. Conf. Evolutionary Computation, 1998 IEEE World Congr. Computational Intelligence, 1998, pp. 69–73.
[28] M. Clerc, "The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization," in Proc. Proc. 1999 Congr. Evolutionary Computation, 1999 (CEC 99), 1999.
[29] M. C. Moura, E. Zio, I. Didier Lins, and E. L. Droguett, "Failure and reliability prediction by support vector machines regression of time series data," Rel. Eng. Syst. Safety, vol. 96, pp. 1527–1534, 2011.
[30] E. Zio, M. Broggi, and L. Golea, "Predicting reliability by recurrent neural networks," in Proc. 8th World Congr. Computational Mechanics" WCCM8 & "5th Eur. Congr. Computational Methods in Applied Science and Engineering" ECCOMAS, Venice, Italy, 2008.
[31] W. C. Hong and P. F. Pai, "Predicting engine reliability by support vector machines," Int. J. Adv. Manuf. Technol., vol. 28, pp. 154–161, 2006.

Wei Zhao received the B.S., M.S., and Ph.D. degrees, all in the School of Automatic Control, from Northwestern Polytechnical University, Xi'an, China.

Then she did postdoctoral research in Beihang University, China, where she is now an Associate Professor. Her main research interests are signal processing and information fusion.

Tao Tao received the B.Sc. degree in physics from Beihang University, China, in 2011. He is now working toward the Ph.D. degree in signal and information processing in the School of Electronic and Information Engineering, Beihang University.

His current research interests concern the reliability prediction methods about the complex systems and components.

Enrico Zio received the B.Sc. degree in nuclear engineering from the Politecnico di Milano, Italy, in 1991; the M.Sc. degree in mechanical engineering from the University of California, Los Angeles, CA, USA, in 1995; the Ph.D. degree in nuclear engineering from the Politecnico di Milano in 1995; and the Ph.D. degree in nuclear engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1998.

He is the Director of the Chair on Systems Science and the Energetic Challenge, European Foundation for New Energy – EDF, CentraleSupélec, France, and the full Professor of Computational Methods for Safety and Risk Analysis in the Politecnico di Milano, Italy. He is President of Advanced Reliability, Availability and Maintainability of Industries and Services (ARAMIS) Ltd., and Chairman of the European Safety and Reliability Association, ESRA. His research topics are analysis of the reliability, safety, and security of complex systems under stationary and dynamic conditions, particularly by Monte Carlo simulation methods; development of soft computing techniques (neural networks, support vector machines, fuzzy and neuro-fuzzy logic systems, genetic algorithms, differential evolution) for safety, reliability, and maintenance applications, system monitoring, fault diagnosis and prognosis, and optimal design. He is co-author of five international books and more than 200 papers in international journals.

Dr. Zio serves as a referee of more than 20 international journals.

Wenbin Wang is a Professor in Operational Research (OR), and the dean of the Dongling School of Economics and Management at the University of Science and Technology Beijing (USTB), China. Prior to joining USTB in 2011, he was professor in OR at Salford Business School, University of Salford, U.K. His main research area is in maintenance modeling, particularly with applications to inspection and condition based maintenance. He has published 9 book chapters and over 150 papers.

Prof. Wang has chaired many international conferences.