SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 5 No. 5
ISSN 2054-7390

Nonlinear Time Series Prediction Performance Using Constrained Motion Particle Swarm Optimization

[1]Nicholas Sapankevych and [2]Ravi Sankar
[1]Raytheon, St. Petersburg, USA;
[2]Department of Electrical Engineering, University of South Florida, USA;
sankar@usf.edu

ABSTRACT

Constrained Motion Particle Swarm Optimization (CMPSO) is a general framework for optimizing Support Vector Regression (SVR) free parameters for nonlinear time series regression and prediction. CMPSO uses Particle Swarm Optimization (PSO) to determine the SVR free parameters. However, CMPSO attempts to fuse the PSO and SVR algorithms by constraining the SVR Lagrange multipliers such that every PSO epoch yields a candidate solution that meets the SVR constraint criteria. The fusion of these two algorithms provides a numerical efficiency advantage since an SVR Quadratic Program (QP) solver is not necessary for every particle at every epoch. This reduces the search space of the overall optimization problem. It has been demonstrated that CMPSO provides similar (and in some cases superior) performance to other contemporary time series prediction algorithms for nonlinear time series benchmarks such as Mackey-Glass data. This paper details the CMPSO algorithm framework and tests its performance against other SVR time series prediction algorithms and data including the European Network on Intelligent Technologies for Smart Adaptive Systems (EUNITE) competition data and the Competition on Artificial Time Series (CATS) competition data.

Keywords: Support Vector Regression, Constrained Motion Particle Swarm Optimization (CMPSO), Particle Swarm Optimization, Nonlinear Time Series Prediction.

1 Introduction

Nonlinear time series regression and prediction applications range from financial market prediction, electrical load forecasting, dynamic control system design, and a vast array of other real world problems. There are many methods to solve such problems including Auto Regressive Moving Average (ARMA) algorithms (in many different forms), Kalman Filtering (also in many different forms), Artificial Neural Networks (ANNs), Support Vector Machines (SVMs) and Support Vector Regression (SVR), and many others. Any of the above algorithms can be applied to real world problems, some with greater success than others. In many cases, the success of the algorithm for a given application depends heavily on algorithm "tuning" – the process of optimizing the algorithm for the specific problem space. Some examples of "tuning" include model selection (as with Kalman Filtering) and free parameter selection (as with SVR). The employment of some algorithms such as SVR further requires the use of a Quadratic Program (QP) to solve for the given algorithmic parameters, thus increasing the computational complexity of these kinds of approaches.

DOI: 10.14738/tmlai.55.3566
Publication Date: 20th August, 2017
URL: http://dx.doi.org/10.14738/tmlai.55.3566

Although SVMs/SVR algorithms are generally computationally complex, they are effective for time series regression and prediction applications [1]. The challenge remains to optimize free (undefined) parameters associated with SVR. These parameters are essential to the proper operation of the SVR algorithm and are generally user selected in an ad hoc manner. Specifically, three parameters need tuning for proper implementation of SVR: 1. A capacity constant (C) that effects scaling and accuracy (which will ultimately be factored out and eliminated as a free parameter in the Constrained Motion Particle Swarm Optimization (CMPSO) formulation), 2. An error bound (ε that will control the data fit, and 3) a kernel function parameter (σ that is used to help cast data to feature space. The focus of this paper is to detail a unique approach of applying SVR to (almost) any time series regression and prediction problem by using a free parameter optimization framework employing Particle Swarm Optimization (PSO). The CMPSO framework is unique in the sense that it does not require the use of a QP for evaluating a candidate SVR solution associated with each particle of the PSO swarm for every epoch, thus reducing the computational load in solving the problem. CMPSO has been shown to be effective for nonlinear time series regression and prediction problems [2]. As compared to other similar-class algorithmic techniques, CMPSO has met or exceeded performance based on experiments using simulated benchmark time series data referred to as Mackey-Glass time series data. Further, CMPSO has the numerical efficiency advantage. The goal of this paper is to provide a more thorough performance analysis and to evaluate CMPSO against many published benchmark time series data.

The outline of the paper is as follows: Section II gives a brief overview of SVR and PSO theory. Section III details the fusion of SVR and PSO to form CMPSO. Section IV shows how CMPSO improves computationally efficient versus using a separate QP solver and associated PSO optimization scheme. Section V provides CMPSOs performance against published benchmark time series data and Section VI details the advantages of CMPSO and further challenges. This section also provides research areas for future investigations.

2 Methodology Background

2.1 Support Vector Regression

SVR is one of two underlying algorithms used in CMPSO that facilitates the interpolation and extrapolation functions of arbitrary one dimensional time series data. SVR is a supervised learning technique that does not assume any specific, underlying model such as found in Kalman filtering. It has advantages for problems that are non-linear in nature and do not have any defined underlying process. The SVR technique is an extension of SVM classifiers developed by Vapnik *et al.* [3, 4]. This technique has been successfully applied for many time series prediction applications [1].

The fundamental principal behind the SVR approach is to cast the time series data into "feature" space as shown in equation (1). The effect of this process is to transform typically non-linear data (non-linear in the sense of a first order linear fit of the time series) into a first order linear regression in the transformed space. The time series estimate, $\hat{y}$, is the weighted combination of a transformed input variable ϕ(x) (where x can be multi-dimensional) plus a bias term b. This is essentially a linear regression model in feature space. The primal formulation derived by Vapnik and summarized by Smola and Schölkopf [5] attempts to minimize the Euclidian norm of the weights (w) and add a loss function to formulate equation (2). Equation (3), the ε-insensitive loss function in this case, allows for errors within a ε bound around the estimate:

$$\hat{y} = w \cdot \phi(x) + b \tag{1}$$

$$\text{Minimize: } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L(y_i, \hat{y}_i) \tag{2}$$

$$L(y_i, \hat{y}_i) = \begin{cases} |y_i - \hat{y}_i| - \varepsilon & \text{if } |y_i - \hat{y}_i| \geq \varepsilon \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

Where, φ(x): Time series cast in feature space; w, b: Weights and bias term, respectively (to be determined);

L: Loss function; ε: Tube size (free parameter); C: Capacity (free parameter)

With the use of Lagrange multipliers and some mathematical manipulation [6], the dual of the primal quadratic programming (QP) problem for SVR (equation (2)) can be formulated in the Lagrange multiplier equation (4) which is called the dual SVR objective function, and the constraints shown in (5):

$$\text{Maximize } \sum_{i=1}^{N} \alpha_i y_i - \varepsilon \sum_{i=1}^{N} |\alpha_i| - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_i K(x_i, x_j) \tag{4}$$

$$\text{Subject to: } -C \leq \alpha_i \leq C\,,\ \sum_{i=1}^{N} \alpha_i = 0\,, C > 0 \tag{5}$$

Where, α: Lagrange multiplier (to be solved using QP or other technique); $K(A, B)$: Kernel Function (user defined); ε: Tube size (free parameter); C: Capacity (free parameter)

For the data sets analyzed in this paper, the exponential kernel given in equation (6) is used:

$$K(A, B) = exp\left(-\frac{1}{2\sigma^2}\|A - B\|^2\right) \tag{6}$$

Where, σ: Kernel (free parameter); A, B: Input vectors

The resulting time series approximation function is given in equation (7):

$$\hat{y} = \sum_{i=1}^{N} a_i^* K(x, x_i) + b^* \tag{7}$$

Equation (7) essentially replaces equation (1) because in the dual formation, the weights *w* in equation (1) equal the weighted sum of the Lagrange multipliers times the input data [5]. The star (*) notation denotes the data points (and corresponding Lagrange multipliers) that lie on or outside the ε tube (and for the bias term, the notation refers to some optimal bias value). Any data points lying inside the tube will be set to zero per the loss function defined in equation (3). This can lead to a "sparse" solution as data points with associated zero Lagrange multipliers will not be necessary to estimate the function.

As mentioned earlier, solving for the Lagrange multipliers requires a quadratic programming algorithm. There are many well-known techniques to solve these types of problems such as Gradient Ascent and Sequential Minimization Optimization (SMO) [7]. The SMO algorithm described in [6] and [7] is designed for SVM classification problems and has been shown to be an efficient method of solving SVM/SVR quadratic programs. CMPSO uses a modified form of SMO specifically designed to solve for the Lagrange

multipliers for the SVR formulation. From the given SVR formulation, one can quickly see that there are several algorithmic choices required to find a suitable solution to any given time series regression or prediction problem. First, a Kernel function must be selected. As documented in [1], the exponential Kernel function is a primary selection for many SVR applications and has been shown to work well in the CMPSO Mackey-Glass benchmark experiment referenced in [2]. Hence, an exponential Kernel was chosen for our experiments. Secondly, there are three free parameters identified in the SVR formulation as user defined. The CMPSO framework attempts to tune these parameters using PSO.

2.2 Particle Swarm Optimization

The selection of C, ε, and σ parameters associated with SVR has been a topic of much research. One candidate approach is Particle Swarm Optimization (PSO) and is the focus of this research. First developed by Kennedy and Eberhart [8] in 1995, PSO mimics the real life process of a swarm of animals or insects searching for food. The goal for each individual of the swarm, where each individual is termed a "particle", is to scavenge in their search region for the place that has the most food. Each particle will remember the location in their search region where they have found the highest density of food and further the entire swarm will remember collectively where the highest density of food was found in the areas that have been searched. As the particles move in their search region, otherwise known as the "search space", they tend to move based on the force, or pull, in three distinct directions:

1) In the direction they are already traveling (inertial)

2) In the direction where the particle remembers the most food being (cognitive)

3) In the direction where the largest amount of food has been found by the entire swarm (social)

As time progresses (and as in real life), the particles of the swarm tend to gravitate towards the place where they collectively have found the most food, assuming there is only one unique place in the entire region that has the most food. Translated to PSO, one would expect every particle to settle on the spot where a maximum (or minimum) solution is found. It should be noted that PSO technical tutorials are found in [9] and [10] with [11] providing a detailed survey of many different PSO applications used in solving real world problems.

The PSO formulation starts with the particle definition, which includes the particles' position and velocity. Each particle definition includes the free parameters and their associated boundaries in the search space. Equations (8a, 8b) are the particle position and velocity definitions for the SVR example given in the previous section.

$$\text{Particle } i \text{ Position:} \quad p_i(C, \varepsilon, \sigma, \alpha_1 \dots \alpha_N, b) \tag{8a}$$

$$\text{Particle } i \text{ Velocity:} \quad v_i(C, \varepsilon, \sigma, \alpha_1 \dots \alpha_N, b) \tag{8b}$$

For CMPSO, each particle i has a position and velocity in search space that includes the three SVR free parameters C, ε, and σ, the N Lagrange multipliers α_1 thru α_N and the bias term b. The boundaries of each of the free parameters are determined by the user before the PSO algorithm is executed (the boundary values of each of the variables will be discussed further in the next section). The next step in the process is to define a "fitness" function. This function (also sometimes referred to as an objective function – not to be confused with the SVR notation of primal and dual objective functions) is the mechanism by which

each particle will evaluate how much food is at its current location in the physical world analogy. Typically the fitness function will generate a single value representing the "level" of fitness relative to some ideal value. In some cases, the position of the best level of fitness may not be unique and particles tending towards "local maxima/minima" can occur. There are no general heuristics for defining fitness functions as they are designed based on the specific application. In practice, identifying an adequate fitness function can be very challenging and may take several iterations to finalize – this is key to a successful optimization process.

After the particles' search space, search boundaries and fitness function have been defined, the next step in the PSO process is to initialize each particle's position and velocity. For each of the variables, a random location and velocity is selected within the boundary of each variable. The random selection is usually from a uniform random distribution within the boundary limits. After particle initialization, the particle simulates real world animal or insect migration by moving through the search space using the simple kinematic equation (9):

Particle Position at Epoch k+1: $$p_{i,k+1} = p_{i,k} + \Delta_t v_{i,k} \quad (9)$$

The position of each particle is updated every epoch (indexed by k) by adding the velocity term with each updated particle position and setting Δt equal to unity. The velocity of each particle i at epoch k is a weighted sum of the current velocity times the "inertial" weighting factor, the direction in which the particle has found its maximum of the objective function times the "cognitive" weighting factor, and the direction in which the maximum is found by all the particles times the "social" weighting factor as given in equation (10):

$$v_{i,k} = \begin{bmatrix} v_{i,k-1} \\ rand \cdot (p_{i,best} - p_{i,k}) \\ rand \cdot (g_{best} - p_{i,k}) \end{bmatrix}^T \cdot \begin{bmatrix} w_I \\ w_c \\ w_s \end{bmatrix} \quad (10)$$

Where, w_I, w_c, w_s: Inertial, Cognitive, and Social Weights;

$rand$: A random number over the closed interval [0,1]; (note the two *rand* functions in (10) are independent); $p_{i,k}$: Particle i's current position;

$p_{i,best}$: Particle i's position where it has found its best result over all k epochs ("local" best)

g_{best}: The position in search space where the swarm's best result was found over all k epochs ("global" best)

The values of $p_{i,best}$ and g_{best} are determined by the evaluation of the fitness function for every particle at every epoch. As each particle moves, the location where the highest score for each particle is remembered ($p_{i,best}$) as well as the location where highest score found for the entire swarm (g_{best}). There are situations where a particle will travel outside the search space for any or all of the variables. There are several techniques available for handling these cases:

Absorbing walls: The velocity of any variable dimension lying outside that dimensional boundary is set to zero. By zeroing the velocity, the particle tends to be "pulled" back into the search space given the p_{best} and g_{best} positions will always be inside the search space. The term "absorbing" for this technique refers to the effect of absorbing the velocity at the limit, or "wall" of the boundary.

Reflecting walls: The sign of the velocity of any dimension lying outside the boundary is reversed. The velocity vector for that dimension causes the particle to "reflect" back into the search space.

Invisible walls: Particles that move outside the boundaries of the search space are allowed without any changes to their velocities; however, the fitness function will not be evaluated until all dimensions of the particles are within the solution boundaries. P_{best} for a particle will not change. The benefit of using this technique is the savings of computational time. Typically, the fitness function is more complex and takes longer to calculate than the equation of motion.

The PSO algorithm is executed until a stopping criterion is reached. There are several different ways to evaluate the completion of the PSO process:

1. *Fitness Function Limit*: The PSO algorithm is stopped when the g_{best} value reaches a user-defined and/or pre-calculated limit.
2. *Function Time-Out:* The PSO algorithm is stopped after a pre-determined amount of time.
3. *Epoch Limit:* The PSO algorithm is stopped after a pre-determined number of epochs.

As stated earlier, it is possible for the PSO algorithm to "stagnate" such that g_{best} value settles on a non-optimal solution (i.e. a "localized" maximum/minimum). One possible solution can be to re-initialize a subset of the particle population to random locations after a certain amount of time or number of epochs has passed. This technique is not guaranteed to succeed as prior knowledge of the search space would have to be known.

2.3 Combining Particle Swarm Optimization and Support Vector Regression

For the SVR problem as stated in section 2.1, there are essentially two optimization objectives:

1. Find the Lagrange multipliers and bias term (typically accomplished with a QP program)

2. Find optimal values for the SVR free parameters (capacity term C, tube size ε, and kernel free parameter σ)

Multiple Objective optimization techniques must be used for applications that have more than one objective. *Multiple Objective* Particle Swarm Optimization (MOPSO) is a candidate technique that is considered for this application since more than one "fitness" criteria is required. A survey of MOPSOs can be found in [12] along with their implementations and applications. Reference [13] also details the use of MOPSOs and qualitative performance results associated with applications with more than one fitness criteria. Many of the MOPSO techniques reviewed and referenced above were considered for solving this specific SVR problem. However, the solution of the QP program is dependent on $C, \varepsilon, \text{and } \sigma$, making the MOPSO techniques cited difficult to implement for this type of problem. In other words, objective number two stated above cannot be found without evaluating objective number one with a candidate set of $C, \varepsilon, \text{and } \sigma$.

PSO based approaches not based on multiple objectives have been proposed for SVR free parameters ($C, \varepsilon, \text{and } \sigma$,) estimation. Hong [14] proposed the use of Chaotic PSO in use with SVR for electrical load forecasting. The technique uses a parallel approach in applying PSO to find the SVR free parameters, but the technique still requires an SVR solution algorithm for the QP problem. Guo *et al*. [15] uses PSO for finding the free parameters for a Least Squares Support Vector Machine (LS-SVM) application using

medical related data for benchmarking (note that this application was specifically for hyper-parameter selection for SVM classification, not SVR based regression). Other PSO related applications involving SVR and linear constraint problems are being studied. Yuan *et al*. [16] introduced a modified PSO algorithm for SVM training based on linearly-constrained optimization using PSO and techniques proposed by Paquet and Engelbrecht [17, 18]. The method presented by Yuan, the "Modified Linear PSO - MLPSO", initializes the Lagrange multipliers randomly, but relies on re-initialization of the Lagrange multipliers should they go beyond the capacity value boundary C. The focus of this research is the PSO applied process for solving linearly constrained optimization problems. Wang *et al.* [19] uses PSO to solve for the three user-defined SVR parameters (referred to as C, ε, and σ, respectively) for a real world application relating to coal working face gas concentration forecasting. This is a similar approach to the presented research in this paper, although the SVR optimization in this citation still requires a separate QP algorithm. Although Wang presents a feasible approach, there is still the computational overhead associated with using SVR and PSO in a non-integrated methodology.

There are many SVR and PSO combined approaches that have been published recently that use PSO to optimize SVR free parameters [19-28]. The applications in these publications range from power load forecasting, traffic flow optimization, to benchmark data estimation, and many others. There are also other general modifications to the algorithms in the citations, but the core processing algorithms are based on both SVM/SVR and PSO. All of these publications show that a PSO based approach to SVR parameter tuning is viable and effective for many different problems. However, none of the researched publications address the computational overhead involved with using a QP solver to find an appropriate SVR solution. The examples stated above that combine both PSO and SVR approaches illustrate the utility and test validity of this type of technique to solve real world problems. It should be pointed out that the PSO/SVR is relatively new area of research and the work outlined in this paper attempts to advance these concepts by using PSO to simultaneously solve the QP problem and optimize the SVR free parameters (C, ε, and σ).

3 Constrained Motion Particle Swarm Optimization

Current research has shown examples of how PSO can be applied to SVR applications; however, there are many complex design and implementation issues. One of the biggest challenges associated with implementing PSO for optimizing SVR, specifically optimizing C, ε, and σ, is the requirement to re-compute the Lagrange multipliers and bias term via a QP solver for every particle position update. The requirement to execute a QP solver for every particle and for every epoch leads to computational inefficiencies and longer optimization times. Another challenge is adapting a PSO/SVR scheme to different applications. Every time series has different numerical bounds, in both the time (independent) and measured value (dependent) dimensions. One could also consider a multiple dimension input to the Support Vector, as SVR itself is designed to accommodate such input schemes. Flexibility in the PSO/SVR design is necessary to accommodate a wide variety of applications. CMPSO is a design that uses PSO as both the SVR free parameter optimizer as well as the QP solving algorithm. Additionally, this optimization framework is able to adapt (scale) the data of interest in order to extend its functionality to multiple applications without the need for “hand” tuning.

3.1 The CMPSO Framework

The goal of CMPSO is to estimate/predict an arbitrary one-dimensional time series using SVR estimation given in equation (7) in section 2.1. The data inputs to CMPSO are N real valued discrete inputs, $x_1 \ldots x_N$, which represent the independent time samples and a corresponding real valued discrete sample for each time step, $y_1 \ldots y_N$. Note the difference between each time step does not have to be uniform. The output of CMPSO is the estimate ($\hat{y}$) of an arbitrary time sequence. For a regression application, the minimum and maximum values of an arbitrary time variable would lie in the interval $[x_1, x_N]$ where $x_1 < x_2 < \ldots < x_N$. For a prediction application, the arbitrary time variable would have values greater than x_N.

In order to adapt CMPSO to multiple applications without the need for offline data pre-processing, the framework automatically adjusts the input time series such that [min(x), max(x)] lie in the interval [0.0, 0.9] for time series prediction past the last known time sample. For prediction applications, time values in the interval (0.9, 1.0] are used. For time series estimation (regression) applications, the input time series lies in the interval [0.0, 1.0]. The framework also automatically scales the independent variable samples such that $|y_i|$ is less than or equal to 1.0. After CMPSO has optimized the free parameters and estimated the scaled time series, the estimate will be re-scaled back to the original x and y dimensions at which point statistical analysis can be performed between the input values (or reference values) and the CMPSO generated estimate.

3.2 The PSO Algorithm Parameters

The PSO parameters are determined (partially) on the computational environment used for this research, specifically the number of particles that are used in the optimization process. More powerful or parallel processing systems may allow for more particles, less computational processing time and/or larger time series sample sets to be processed at once. A summary of CMPSO computational efficiency and computing environment is given in Section 4. For this research, a summary of PSO specific parameters are given in Table I. It should be mentioned that the determination of the three velocity weighting factors can be adjusted by the user and these values were chosen empirically for the data researched in this paper.

3.3 The SVR Algorithm Parameters

Given the objective of CMPSO to eliminate the need for a QP solver for each epoch k, the Lagrange multipliers and the bias term need to be added to the three free parameter for optimization (one Lagrange multiplier for each data sample and one bias term as outlined in section 2.1). Each particle's position in search space was defined in equation (8a) based on the given SVR variables above. One problem associated with using PSO to solve the SVR optimization problem is defining the Capacity term C. PSO (for CMPSO) requires constant limits for each of the variables, including the Lagrange multipliers. However, C is a variable that constrains the limit of the Lagrange multiplier as seen in the constraint equation in (5). This capacity term C can be viewed as a Lagrange multiplier scale factor. If C is viewed as such, it would be advantageous to remove this factor from the SVR constraint equation if possible, and as it turns out, the elimination of this variable as a constraint is part of the motivation to establish the CMPSO generic framework that scales the data to the same numerical bounds regardless of application.

Table 1: CMPSO – PSO Algorithm Specific Parameters

Parameter	Value	Notes
Number of Particles	500	Limited by computational hardware and software implementation
Number of Particles to Reset	490	Number of particles periodically reset to avoid stagnation
Inertial Weight w_I	0.75	Particle velocity weighting factor based on current particle velocity
Cognitive Weight w_C	0.75	Particle velocity weighting factor based on direction where particle found optimal result
Social Weight w_S	0.25	Particle velocity weighting factor based on direction where optimal result found for entire swarm
Epoch Limit	2000	One of two limits that when reached, halt CMPSO

The Capacity term C can be factored out of equation (4) and the constraints in (5) as follows:

$$\text{Maximize: } C \cdot \left(\sum_{i=1}^{N} \alpha_i y_i - \varepsilon \sum_{i=1}^{N} |\alpha_i| - \frac{C}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_i K(x_i, x_j) \right) \tag{11}$$

$$\text{Subject to: } -1 \leq \alpha_i \leq 1 \, , \sum_{i=1}^{N} \alpha_i = 0, C > 0 \tag{12}$$

As seen in (11), each Lagrange multiplier is scaled by a factor C (illustrated in the equation for completeness), and in (12), the constraint limits of the α_i variables have been set to constants value of +/- 1.0. This will allow for CMPSO to fix the bounds of the Lagrange multipliers and to treat the Capacity value C as a free parameter to optimize in a way that fits the CMPSO framework. This optimization problem is the same as the referenced SVR optimization problem, just with a change in the scaling of the Lagrange multipliers. Note in equations (11) and (12) that the bias term *b* is not present in solving for the Lagrange multipliers. For CMPSO (and SVR), it is not necessary to have the bias term as part of the free parameters to optimize. Assume from equation (7) that some optimal bias term b^* exists. One mathematical model to optimize this free parameter would be to minimize the error mean between the training values *y* and each resulting particle output as seen in equation (13):

$$\text{Minimize:} \quad \mathrm{E}[y - \hat{y}_i] = b_i^* \tag{13}$$

This now allows for a single calculation for each particle at each epoch for the bias term *b*. The equations that define each particle position and velocity given in (8a) and (8b) then become equations (14a) and (14b) respectively:

$$\text{Particle } i \text{ Position:} \quad p_i(C, \varepsilon, \sigma, \alpha_1 \ldots \alpha_N) \tag{14a}$$

$$\text{Particle } i \text{ Velocity:} \quad v_i(C, \varepsilon, \sigma, \alpha_1 \ldots \alpha_N) \tag{14b}$$

The last two variables to consider for the SVR parameters are the gap value ε and the Kernel Function (an exponential kernel function for this research – see equation (6)) free parameter σ. Through empirical analysis, the boundaries for these parameters as well as the other variables defined in this section are detailed in Table II.

3.4 Particle Initialization and Constrained Motion

Each particle position and velocity defined in equations (14a) and (14b) must be initialized to some value. For capacity C, gap ε, and kernel constant σ, the initial values are selected from a uniform distribution within the bounds defined in Table II. However the bounds for the Lagrange multipliers have to be dealt with in a separate way. Although the bounds are set within the [-1.0, 1.0] limits, there is the additional constraint that the Lagrange multipliers must sum to zero (see dual constraint equation (5)). CMPSO will randomly select N-1 Lagrange multipliers to initialize. The values will be randomly selected from a uniform distribution between the [-1.0, 1.0] limits. The N^{th} Lagrange multiplier will be the negative of the sum of the N-1 randomly selected Lagrange multiplier values, satisfying the constraint. The exact same process is repeated for the initial velocity values (times a uniform scale value). For each particle, the candidate time series fit is now a feasible (but not necessarily optimal) candidate based on the SVR dual optimization constraints in equation (5). At this point, a fitness function is evaluated (see section E below) and the p_{best} value for each particle is stored. The maximum p_{best} value is then selected as the g_{best} value for the optimization.

As described in section 2.2, each particle must now move in search space searching for an optimal solution. The motion is based on a weighted sum of a particle's initial velocity, its direction towards its pbest value, and its direction towards the optimal place found by the swarm for each variable (gbest). The constraint equation in (5) must still hold after a particle moves. Since a particle's $\alpha_1 \dots \alpha_N$ velocity is equal to zero, and the pbest and gbest locations for the Lagrange multipliers also sum to zero, any movement of the weighted sum of these values also sums to zero. The particle motion for $\alpha_1 \dots \alpha_N$ is constrained to meet the constraint equation in (5). Therefore, every particle at every epoch will always contain a feasible solution to the SVR optimization problem. To ensure this constraint holds when a particle moves outside the Lagrange multiplier (or other variable) bounds, the Invisible Walls technique is used.

Table 2: CMPSO – SVR Specific Parameters

Parameter	Value Range	Notes
Capacity, C	[0.001, 2.0]	Set based on the scaling of the input data
Gap, ε	[0.02, 0.10]	Notional values – can be user defined
Kernel Constant, σ	[1e-5, 1e-3]	Set based on the scaling of the input data
Lagrange Multipliers , $\alpha_1 \dots \alpha_N$	[-1.0, 1.0]	Bounded by the factoring of Capacity C
Bias Term b	{unbounded}	Can be determined after Lagrange multipliers have been found

3.5 The Fitness Function and Iteration Bounds

The fitness function can be one of the most difficult tasks to complete in defining a PSO framework. For time series prediction and regression, there are several choices in evaluation metrics to select such as error mean or mean squared error, etc. However, the CMPSO framework is set up not only to find the best fit for time series data, but also to be a QP solver. This research has found that using the dual gap measure associated with SVR optimization convergence is sufficient to produce good results as published

in [2]. Equation (15) is the ratio of the difference of the primal and dual objectives defined in equations (2) and (4) respectively to the primal objective output plus one. This equation is referenced in [5] and [6] as finding the optimal Lagrange multipliers for the SVR problem:

$$\text{Fitness: } \frac{Primal\ Objective - Dual\ Objective}{Primal\ Objective + 1} \tag{15}$$

As mentioned earlier, one of the iteration stopping points is the number of epochs completed, which is set to 2000 for CMPSO (but can be changed based on user needs). This bound is essentially a computational time limit in case there are issues with any particular data series converging. For CMPSO, there are two numeric limits that will stop processing and produce a final output – both are based on the fitness function in equation (15). The ideal limit based on published recommendations in [5] and [6] is set to 0.001. If equation (15) is evaluated at this value or less, the entire optimization process is complete.

In the results published in [2], it has been found that a value of 0.25 for the fitness function limit is sufficient to stop the PSO optimization process in finding the values of C, ε, and σ. At this point these values are fixed and a separate QP program is executed once. For this research, Sequential Minimization Optimization (SMO) for SVR (an implementation based on [5] and [6]) is implemented. Since the Lagrange multipliers are feasible at every point in the CMPSO process, the SMO can be initially seeded with these values, although there is no guarantee of increased SMO performance between nonzero seeded SMO execution vs. zero seeding the Lagrange multipliers. Nevertheless, the worst case scenario means CMPSO will execute a QP program only once. It turns out that the computational impact is negligible as explained in the next section. Figure 1 illustrates the block diagram for the entire CMPSO process.

4 Computational Efficiency

One of the CMPSO objectives is to increase computational efficiency by eliminating the need for a QP solver. The CMPSO framework achieves this by constraining the motion of the particles in search space such that the SVR objective function constraints are always met for every particle at all times. This yields a feasible candidate result for every particle at every epoch and essentially reduces the search space. A simple experiment was set up to illustrate CMPSO computational efficiency. A small data set was used to estimate the best regression fit by a) CMPSO and b) a "separated" PSO and SVR implementation where each particle's Lagrange multipliers were computed to reach the duality gap limit of 0.001 for every epoch. The data set is a 41-point sample of the function defined in equation (16):

$$f(x) = \sin(0.2\pi x)\, e^{(0.1x)} + 0.5N(0,1) \tag{16}$$

Where, N(.) is a Normal distribution with zero mean and unit variance.

The same computing environment and functional software components were used for both CMPSO and the PSO/SVR "separated" approach – a Windows 7 (Intel) based stand-alone PC. The CMPSO framework, PSO and SVR functions are written in MATLAB and C ("mex" files called by MATLAB), where the exact same software was used for both approaches. MATLAB has the ability to measure the time to execute any particular block of code and this functionality was used in this experiment. For the "separated" approach, the particles would only optimize the SVR free parameters and SMO (for SVR) was used to solve for the Lagrange multipliers. Also, a range of number of particles, down to a minimum of 25, was used to compare results. A total of 300 runs were executed for both methodologies. Several key metrics were analyzed:

1. Time to execute entire algorithm
2. Number of runs that converged to dual gap value of 0.001
3. Normalized Mean Square Error (NMSE) of results

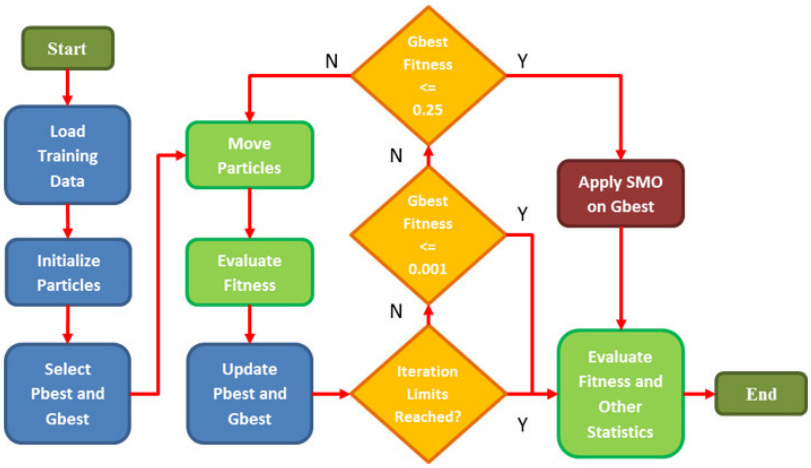

Figure 1 CMPSO Functional Block Diagram

Table III illustrates the results of the experiment. The experimental results show that 99% of the time CMPSO converged to a solution in approximately seven seconds, typically with a very small NMSE of less than 0.005. By contrast, running PSO and SVR separately, the solution never converged with the minimum number of particles (25) and it took more than five times longer for the PSO+SVR method to settle on incorrect results (essentially stagnate). If more particles are used in this method to try and improve the results, the processing time would only increase. This simple experiment shows that not only is CMPSO a viable algorithm for approximating time series functions, it is also more computationally efficient than functionally separated methods.

Table 3: CMPSO – Computational Efficiency

Measured Parameter	CMPSO	PSO+SVR	Notes
Mean Algorithm Execution Time (sec)	7.2749	39.0826	PSO+SVR used minimum of 25 particles
Percentage of Runs that Converged	99.0%	0.0%	PSO+SVR never converged to dual gap limit
NMSE	Typically less than 0.005	{N/A}	Since PSO+SVR never converged, NMSE was not calculated

5 Performance Results

CMPSO algorithm performance was measured against benchmark Mackey-Glass time series data as referenced in [2]. This data is highly nonlinear, but synthesized. In this research, we compared the CMPSO performance against a real world electrical load forecasting application as well as another nonlinear, synthesized data set from the Competition on Artificial Time Series (CATS). CMPSO performance was compared to the original competition contestants as well as more recent SVR and non-SVR based

algorithms that used the data sets post-competition. It is important to note that the CMPSO framework, by design, was not altered for both the presented results.

5.1 EUNITE Competition Data

In 2001, the European Network on Intelligent Technologies for Smart Adaptive Systems (EUNITE – see [29]) initiated a competition [30] to predict maximum electrical loads for the East-Slovakia Power Distribution Company. The goal is to estimate maximum daily power loads for the month of January 1999 based on the daily values of 1997 and 1998 of electrical load and temperature. Accurate predictions are measured by the Maximum Average Percent Error (MAPE) and Maximum Error (MAXE) as shown in equations (17) and (18):

$$MAPE = \frac{100}{31}\sum_{i=1}^{3\ 1}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{17}$$

$$MAXE = \max(|y_i - \hat{y}_i|)_{i=1\ldots3\ 1} \tag{18}$$

Where,
y_i= Actual Maximum Load for Month of January 1999
$\hat{y}_i$= Predicted Maximum Load

For this data set, the maximum number of data points is 31 as there are 31 days in the month of January. This data set requires some form of strategy for using CMPSO, as there are many data sets available for training (two years of power and temperature data respectively as well as the temperature for January 1999). For the purposes of showing the utility of CMPSO, the CMPSO framework not been altered, but two simple strategies were used on the input data based on distinct factors associated with the EUNITE data set:

1. There appears to be loading trends based on what day of the week it is. It is assumed that weekends tend to have less power demand than the Monday through Friday work week.
2. There is a relationship between the temperature and power consumption for any given day. As temperature decreases, power consumption increases.

Figure 2 shows the January 1997 and January 1998 maximum power loads, time aligned such that the two data sets are aligned by day of the week. There are trends surrounding the weekends vs. the work week, as noted by the relative maximums of five days duration at work week vs. the general minimums of two days duration at weekend that are clearly visible. This figure illustrates the first factor in the prediction strategy. To address the second factor, it is necessary to eliminate the trending in the data as a function of the day of week (assuming there is a relationship between temperature and day of week). This can be achieved by looking at the daily changes in power load and temperature between 1997 and 1998. Figure 3 is an illustration of this relationship. It is a scatter plot of the change in power load vs. the change in temperature for 360 days, time aligned to the day of week. A correlation coefficient of -0.5041 was computed on the 360 data points. This result means there is a moderate inverse correlation between power consumption and temperature. Given the relationship between power consumption and temperature, a simple linear approximation can be generated as shown in equation (19):

$$\Delta_P = -3.4668\Delta_T + 9.9573 \tag{19}$$

Where,

Δ_P = Change in maximum load for same day of the week between years (MW)
Δ_T = Change in temperature for the same day of the week between years (C)

This offset can now be applied to each day of the prediction, if the temperature of Jan 1999 is known (or in reality can be predicted reliably in the short term).

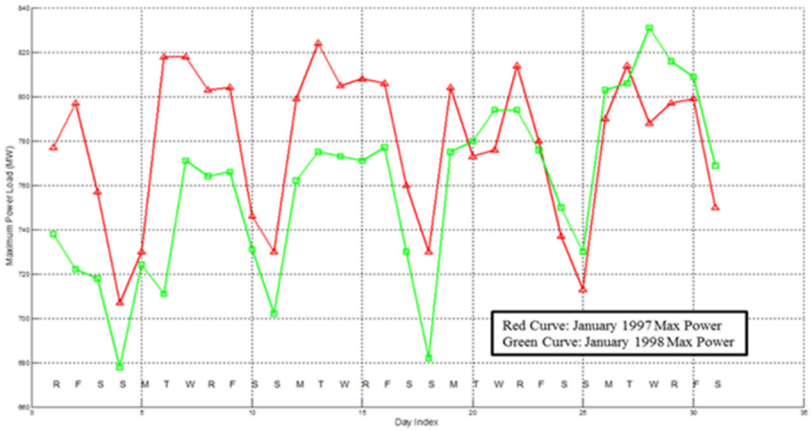

Figure 2: Maximum Power Load for January 1997 and 1998

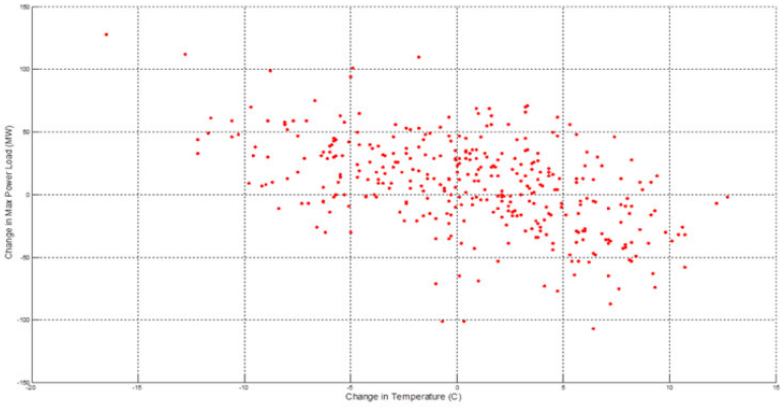

Figure 3: Relationship between changes in Power Load and Temperature for January 1997 and 1998

To predict the trending of the data, CMPSO was used to fit the mean value of the day-of-week aligned maximum power loads of January 1997 and January 1998. In addition, the output of CMPSO was offset by the linear approximation described in equation (19) on a day by day basis based on the difference in temperature between 1998 and 1999. Table IV shows the numerical results of the experiment.

Table 4: CMPSO – EUNITE Competition Results

Measured Parameter	CMPSO	CMPSO + ⍰P Offset	Notes
MAPE	3.176	2.628	% Error
MAXE	84.931	66.653	Max Error in MW

As shown in Table IV, there is significant improvement by using the temperature data to offset the values. Figure 4 shows the final results of this experiment. As can be seen, CMPSO gives a good approximation

to the actual January 1999 load data (Blue Curve). The simple linear approximation is shown to improve with the offset in power due to temperature (Magenta Curve). In addition, Figure 5 is an approximation of where CMPSO (and the offset addition) would have ranked in the competition based on the values shown in Table V (referenced from [30]).

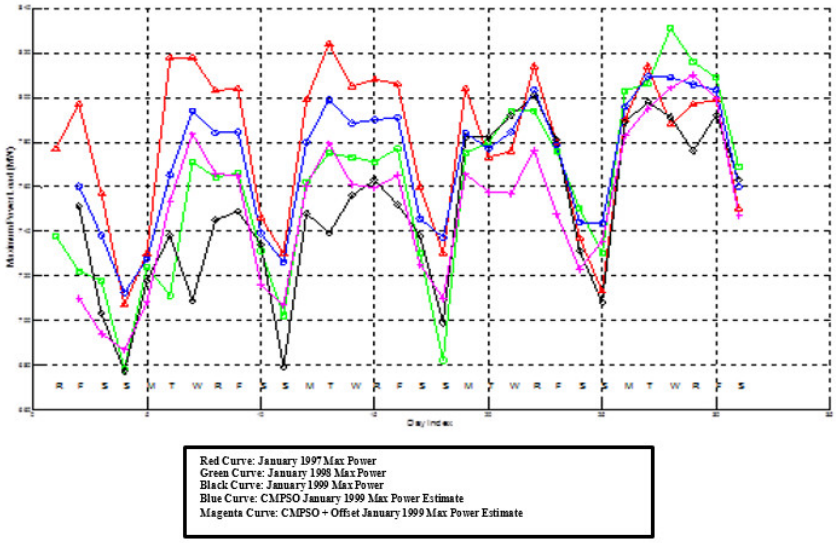

Figure 4: CMPSO EUNITE Results

As seen in Figure 5, an attempt was made to rank CMPSO and CMPSO with the delta offset with the other competitors (in the absence of the actual result values from each competitor). There are two sets of highlighted text boxes and lines for each of the CMPSO results. Based on MAPE, it appears CMPSO alone would have ranked at least sixth. In the case where the January 1999 temperature is included, CMPSO (plus the offset) would have ranked no worse than third based on MAPE. Note that the MAXE values were also close to the other competitors (refer to Table V). The winners of this competition, Chen *et al.* [31] used an SVM/SVR approach with multi-dimensional inputs to predict load. They also took into consideration additional attributes such as if the day being predicted is a holiday. The authors further reported details of their approach in [32], noting that there is no added value in using temperature data to predict the load. Although the weighting of electrical load for a holiday was not taken into consideration for this CMPSO benchmark evaluation, there was clearly an improvement in CMPSO prediction performance by modeling temperature and including the offset in the results.

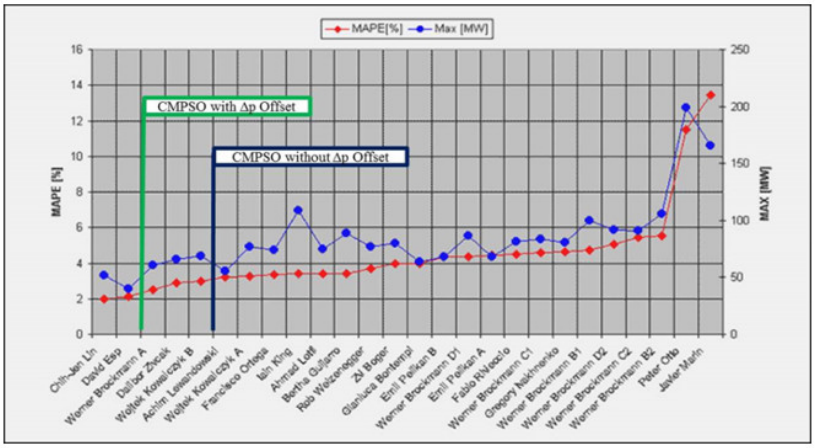

Figure 5: CMPSO EUNITE Results Compared to Other Contestants

In addition, CMPSO is compared to other recent SVM/SVR based approaches ([33] through [37]) that used EUNITE data as a benchmark. Table V shows the algorithm comparison by metric.

Table 5: CMPSO and SVR Based Algorithms – EUNITE Benchmark

Method	Metric		
	MAPE	MAXE	MSE
LS-SVM w/ k-nearest neighbor [37]	1.71	40.99	N/A
LS-SVM [34]	1.97	39.778	N/A
EMD-SVR [33]	1.98	N/A	284.3
CMSPO w/ Linear Offset	2.628	74.663	594.3
CMPSO	3.176	84.931	908.8
SVM Model [36]	3.67	N/A	N/A
AR Model [35]	6.69	N/A	N/A

5.2 CATS Data

In 2004 at the International Joint Conference on Neural Networks, a competition was organized to compare different prediction methods using a proposed benchmark CATS (Competition on Artificial Time Series) time series data of 5000 samples, among which 100 samples are missing [38]. The goal of the competition was to predict five sets of 20 missing data points each from the given data of 5000 samples and compare the methods using two criteria. The first was the Mean Square Error (MSE) - E1 for the four missing data gaps at sample indices 981 to 1000, 1981 to 2000, 2981 to 3000, and 3981 to 4000 in addition to the last omitted 20 data points with sample indices 4981 to 5000. The second was the MSE - E2 for the first four missing data gaps only. E2 was meant to only evaluate the effectiveness of the interpolation of a given algorithm versus the prediction estimate in E1 (the last 20 omitted samples).

Two different winners were selected based on the lowest E1 and E2 MSE scores. Sarkka *et al.* [39] used a Kalman Filter based approach with the lowest E1 MSE for all five missing data sets while Wichard and Ogorzalek [48] used an ensemble Neural Network based approach with the lowest E2 MSE for the first four missing data sets. The strategy for adapting CMPSO was straight forward for this particular application. For each of the first four data gaps, every tenth data sample was used. Half of the samples

were before the gap and the other half after the gap. In addition, ten consecutive samples were used just before and after the data gap for a total of 94 data points per set. For the last data set, a similar strategy of every 10 data samples was selected prior to the end of the data set along with the last 20 sequential samples for a total of 138 samples. Every sample was estimated between the first and last index of the data set and compared to the given results for evaluation.

CMPSO is compared to the top ten entries in the competition ([39] through [48]) as well as several other recent algorithm publications ([49] through [53]) that use CATS data as a benchmark. In all cases, CMPSO outperformed all of the other algorithms for this data set. Table VI shows the comparison of all the algorithms considered. Note the entries in Table VI with an (*) are recently developed approaches that used the benchmark CATS data as a performance indicator. Figures 6 and 7 illustrate CMPSO performance for two of the five data sets for estimation. Figure 6 is the second data set (gap) and Figure 7 is the last data set (prediction). The red curve is the provided competition data, the blue curve is the CMPSO estimation and the red circles are the actual CATS missing data for comparison. As seen from the results, CMPSO provides a very close fit to the missing competition data.

Table 6 CMPSO – CATS Benchmark

E1 MSE	E2 MSE	Model
113	140	CMPSO
143	129	ANN and AdaBoost* [50]; Single Global Model
262	239	ANN and AdaBoost* [50]; Multiple Local Models
287	N/A	Variance Minimization LS-SVM* [52]
390	288	Dynamic Factor Graphs* [53]
408	346	Kalman Smoother [39]
441	402	Recurrent Neural Networks [40]
502	418	Competitive Associative Net [41]
530	370	Weighted Bidirectional Multi-stream Extended Kalman Filter [42]
577	395	SVCA Model [43]
644	542	MultiGrid-Based Fuzzy System [44]
653	351	Double Quantized Forecasting Method [45]
660	442	Time-reversal Symmetry Method [46]
676	677	BYY Harmony Learning Based Mixture of Experts Model [47]
725	222	Ensemble Models [48]
1215	979	Deep Belief Network with Boltzmann Machines* (best value) [51]
2510	2450	ANN Based Approach* [49]

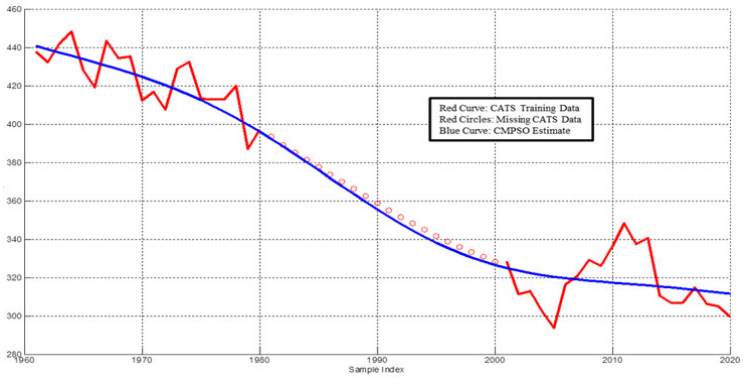

Figure 6: CMPSO Estimate for CATS Data Set 2

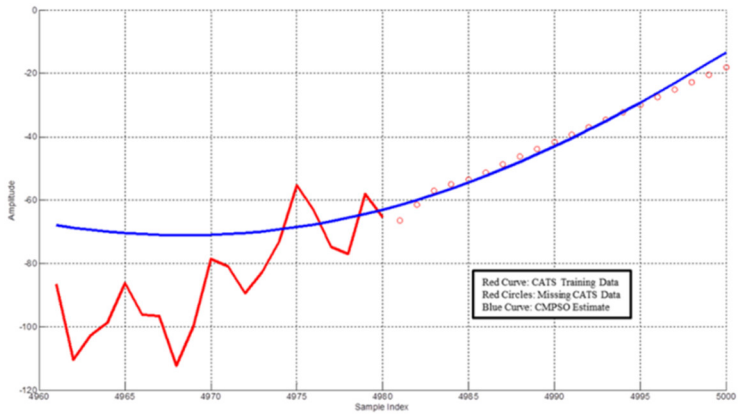

Figure 7: CMPSO Estimate for CATS Data Set 5

6 Conclusion

Support Vector Regression has been shown to be an effective algorithm for nonlinear time series regression and prediction applications across many different domains [1-2]. One of the major challenges associated with SVR is the ability to "tune" SVR to yield adequate results from application to application. Constrained Motion Particle Swarm Optimization (CMPSO) is a unique fusion of Support Vector Regression and Particle Swarm Optimization algorithms that attempts to optimize SVR free parameters while minimizing the need for excess computational resources by ensuring a feasible solution for every particle at every epoch. In addition, CMPSO is a standalone application that does not require user intervention regardless of data type. For the presented data, no algorithmic parameters were ever altered by the user.

CMPSO has performed well for a wide variety of nonlinear time series regression and prediction applications, including benchmark Mackey-Glass time series data [2]. EUNITE and CATS benchmark data results offer further proof of CMPSO performance as shown by comparison to other similar techniques. Based on the results presented in this paper, CMPSO is a viable, generalized approach to time series estimation and regression as compared to both SVM/SVR approaches as well as other Neural Network based algorithms. One unique feature associated with CMPSO is the reduction of the optimization space as the constraints associated with particle motion ensure a feasible solution for every particle at every epoch. Other PSO based SVR optimization techniques will require some form of QP solver for every candidate solution (i.e. every particle) at every epoch. This is an increase in computational efficiency.

Future research should include optimization of the PSO free parameters, choices of SVR Kernel Functions, different loss functions (such as used in LS-SVM), and the adaptation to multi-dimensional input data. It could also be conceivable to extend the CMPSO framework to include multiple models beyond SVR, where each particle could have sub-particles dedicated to any given regression/prediction algorithm.

ACKNOWLEDGEMENT

The authors would like to thank Dr. A. D. Snider, *Professor Emeritus* at the University of South Florida for his help in reviewing the manuscript and providing detailed comments to improve the quality of presentation.

REFERENCES

[1] N. Sapankevych and R. Sankar, "Time Series Prediction Using Support Vector Machines: A Survey", *IEEE Computational Intelligence*, vol. 4, no. 2, pp. 24-38, May 1999.

[2] N. Sapankevych and R. Sankar, "Constrained Motion Particle Swarm Optimization and Support Vector Regression for Non-Linear Time Series Regression and Prediction Applications", *12th International Conference on Machine Learning and Applications (ICMLA 2013)*, December 2013.

[3] V. N. Vapnik, "The Nature of Statistical Learning Theory", Springer, 1995.

[4] V. N. Vapnik, "Statistical Learning Theory", *John Wiley and Sons*, 1998.

[5] A. J. Smola and B. Schölkopf, "A Tutorial on Support Vector Regression", *NeuroCOLT Technical Report*, Royal Holloway College, London, UK, 1998.

[6] N. Cristianini and J. Shawe-Taylor, "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods", *Cambridge University Press*, 2000.

[7] John C. Platt, "Sequential Minimization Optimization: A Fast Algorithm for Training Support Vector Machines", Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[8] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[9] R. Poli, J. Kennedy, and T. Blackwell, "Particle Swarm Optimization: An Overview", *Swarm Intell*, vol. 1, pp. 33-57, 2007.

[10] J. Robinson and Y. Rahmat-Samii, "Particle Swarm Optimization in Electromagnetics", *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, February 2004.

[11] R. Poli, "An Analysis of Publications on Particle Swarm Optimization Applications", *University of Essex, UK, Department of Computer Science, Technical Report CSM-469*, May 2007, Revised November 2007.

[12] M. Reyes-Sierra and C. A. C. Coello, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287-308, 2006

[13] J. Fieldsend, "Multi–Objective Particle Swarm Optimization Methods", *Technical Report No. 419, Department of Computer Science, University of Exeter*, 2004.

[14] W.-C. Hong, "Chaotic Particle Swarm Optimization Algorithm in a Support Vector Regression Electric Load Forecasting Model", Energy Conversion and Management, vol. 50, no. 1, pp. 105-117, 2009.

[15] X. C. Guo, J. H. Yang, C. G. Wu, C. Y. Wang, and Y. C. Liang, "A Novel LS-SVMs Hyper-Parameter Selection Based on Particle Swarm Optimization", Neurocomputing, vol. 71, no. 16-18, pp. 3211-3215, 2008.

[16] H. Yuan, Y. Zhang, D. Zhang, and G. Yang, "A Modified Particle Swarm Optimization Algorithm for Support Vector Machine Training", *Proceedings of the 6th World Congress on Intelligent Control and Automation*, June 21-23, 2006.

[17] U. Paquest and A. P. Engelbrecht, "A New Particle Swarm Optimizer for Linearly Constrained Optimization", *Proceedings IEEE Congress on Evolutionary Computation*, pp. 227-233, 2003.

[18] U. Paquest and A. P. Engelbrecht, "Training Support Vector Machines with Particle Swarms", *Proceedings International Joint Conference on Neural Networks*, pp. 1593-1598, 2003.

[19] X. Wang, J. Lu and J. Liu, "Wavelet Transform and PSO Support Vector Machine Based approach for Time Series Forecasting", *International Conference on Artificial Intelligence and Computational Intelligence*, vol. 1, pp. 46-50, Nov 2009.

[20] J. Hu, P. Gao, Y. Yao, and X. Xie, "Traffic Flow Forecasting with Particle Swarm Optimization and Support Vector Regression", *IEEE 17th International Conference on Intelligent Transportation Systems,* pp. 2267-2268, October 2014.

[21] Y. Wen and Y. Chen, "Modified Parallel Cat Swarm Optimization in SVM Modeling for Short-term Cooling Load Forecasting", *Journal of Software*, vol. 9, no. 8, August 2014.

[22] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble Deep Learning for Regression and Time Series Forecasting", *IEEE Symposium on Computational Intelligence in Ensemble Learning,* pp. 1-6, December 2014.

[23] C. Rajeswari, B. Sathiyabhama, S. Devendiran, and K. Manivannan, "Bearing Fault Diagnosis Using Wavelet Packet Transform, Hybrid PSO, and Support Vector Machine", *12th Global Congress on Manufacturing and Management – Procedia Engineering*, vol. 97, 2014.

[24] P. Shinde, T. Parvat, "Analysis on: Intrusions Detection Based on Support Vector Machine Optimized with Swarm Intelligence", *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 12, pp. 559-566, December 2014.

[25] W. Wenhai, D. Jiandong, "Short-term Wind Power Forecasting Based on Maximum Correntropy Criterion", *International Conference on Power System Technology*, pp. 2800-2805, October 2014.

[26] Z. Hu, Q. Liu, Y. Tian, Y. Liao, "A Short-term Wind Speed Forecasting Model Based on Improved QPSO Optimizing LSSVM", *International Conference on Power System Technology*, pp. 2806-2811, October 2014.

[27] H. Dong, X. Zhu, Y. Liu, F. He, and G. Huo, "Iris Recognition Based on CPSO Algorithm for Optimizing Multichannel Gabor Parameters", *Journal of Computational Information Systems*, vol. 11, pp. 333-340, 2015.

[28] J. F. L. de Oliveira, T. B. Ludermir, "A Distributed PSO-ARIMA-SVR Hybrid System for Time Series Forecasting", *IEEE International Conferences on Systems, Man, and Cybernetics*, pp. 3867-3872, October 2014.

[29] http://www.eunite.org (January 2015)

[30] http://neuron-ai.tuke.sk/competition/ (January 2015)

[31] M.-W. Chang, B.-J. Chen, and C.-J. Lin, "EUNITE Network Competition: Electricity Load Forecasting", November 2001. (see [19]).

[32] B.-J. Chen, M.-W. Chang, and C.-J. Lin, "Load Forecasting Using Support Vector Machines: A Study on EUNITE Competition 2001", *IEEE Transactions on Power Systems*, vol. 19, no. 4, pp. 1821-1830, November 2004.

[33] B. Bican, Y. Yaslan, "A Hybrid Method for Time Series Prediction Using EMD and SVR", *6th International Conference on Communications, Control, and Signal Processing*, pp. 566-569, May 2014.

[34] X. Yang, "Comparison of the LS-SVM Based Load Forecasting Models", *International Conference on Electronic & Mechanical Engineering and Information Technology*, pp. 2942-2945, Aug 2011.

[35] I. Fernandez, C. E. Borges, and Y. K. Penya, "Efficient Building Load Forecasting", *IEEE 16th Conference on Emerging Technologies & Factory Automation*, pp. 1-8, September 2011.

[36] B. E. Turkay and D. Demren, "Electrical Load Forecasting Using Support Vector Machines", *7th International Conference on Electrical and Electronics Engineering*, vol. I, pp. 49-53, December 2011.

[37] T. T. Chen, S. J. Lee, "A Weighted LS-SVM Based Learning System for Time Series Forecasting", *Information Sciences*, pp. 99-116, 2015.

[38] Lendasse, E. Oja, O. Simula, and M. Verleysen, "Time Series Prediction Competition: The CATS Benchmark", *Proc. IEEE International Joint Conference on Neural Networks*, pp. 1615-1620, 25-29 July 2004.

[39] S. Sarkka, A. Vehtari, and J. Lampinen, "Time Series Prediction by Kalman Smoother with Cross Validated Noise Density", *Proc. IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 1653-1657, 25-29 July 2004.

[40] X. Cai, N. Zhang, G. Venayagamoorthy and D. Wunsch, "Time Series Prediction with Recurrent Neural Networks Using a Hybrid PSO-EA Algorithm", *International Joint Conference on Neural Networks,* vol. 2, pp. 1647-1652, 2004.

[41] S. Kurogi, T. Ueno and M. Sawa, "Batch Learning Competitive Associative Net and Its Application to Time Series Prediction", *International Joint Conference on Neural Networks,* vol. 2, pp. 1591-1596, 2004.

[42] X. Hu and D. Wunsch, "IJCNN 2004 Challenge Problem: Time Series Prediction with a Weighted Bidirectional Multi-stream Extended Kalman Filter", *International Joint Conference on Neural Networks*, vol. 2, pp. 1641-1645, 2004.

[43] F. Palacios-Gonzalez, "A SVCA Model for The Competition on Artificial Time Series", *International Joint Conference on Neural Networks,* vol. 4, pp. 2777-2782, 2004.

[44] L. J. Herrera Maldonado, H. Pomares, I. Rojas, J. Gonzalez and M. Awad, "MultiGrid-Based Fuzzy Systems for Time Series Forecasting: CATS Benchmark IJCNN Competition", *International Joint Conference on Neural Networks*, vol. 2, pp. 1603-1608, 2004.

[45] G. Simon, J. A. Lee, M. Verleysen and M. Cottrell, "Double Quantization Forecasting Method for Filling Missing Data in the CATS Time Series", *International Joint Conference on Neural Networks*, vol. 2, pp. 1635-1640, 2004.

[46] P. F. Verdes, P. M. Granitto, M. I. Szeliga, A. Rebola and H. A. Ceccatto, "Prediction of the CATS benchmark exploiting time-reversal symmetry", *International Joint Conference on Neural Networks*, vol. 2, pp. 1631-1634, 2004.

[47] H.-W. Chan, W.-C. Leung, K.-C. Chiu and L. Xu, "BYY Harmony Learning Based Mixture of Experts Model for Non-stationary Time Series Prediction", *International Joint Conference on Neural Networks*, 2004.

[48] J. Wichard and M. Ogorzalek, "Time Series Prediction with Ensemble Models", *International Joint Conference on Neural Networks*, vol. 2, pp. 1625-1630, 2004.

[49] D. Samek and D. Manas, "Comparison of Artificial Neural Networks Using Prediction Benchmarking", *Proceedings of the 13th International Conference on Automatic Control, Modelling & Simulation*, pp. 152-157, 2011.

[50] Y. Dong, J. Zhang, and J. M. Garibaldi, "Neural Networks and AdaBoost Algorithm Based Ensemble Models for Enhance Forecasting of Nonlinear Time Series", *International Joint Conference on Neural Networks*, vol. 1, pp. 149-156, July 2014.

[51] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi, "Time Series Forecasting Using a Deep Belief Network with Restricted Boltzmann Machines", *Neurocomputing*, pp. 47-56, 2014.

[52] R. Ormandi, "Applications of Support Vector-Based Learning", Research Group on Artificial Intelligence of the University of Szeged and the Hungarian Academy of Sciences, Ph. D. Dissertation, 2013.

[53] P. Mirowski, "Time Series Modeling with Hidden Variables and Gradient-Based Algorithms", Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, Ph. D. Dissertation, 2011.