REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

HYPERPARAMETER OPTIMIZATION OF SUPPORT VECTOR MACHINES WITH A NEW ALGORITHM COMBINING GRID SEARCH AND PARTICLE SWARM OPTIMIZATION

YUNUS ALTUNKOL
MASTER OF SCIENCE

ADANA 2022

REPUBLIC OF TURKEY

ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

HYPERPARAMETER OPTIMIZATION OF SUPPORT VECTOR MACHINES WITH A NEW ALGORITHM COMBINING GRID SEARCH AND PARTICLE SWARM OPTIMIZATION

Yunus ALTUNKOL

MASTER OF SCIENCE

SUPERVISOR

Mustafa AÇIKKAR

ADANA 2022

ABSTRACT

HYPERPARAMETER OPTIMIZATION OF SUPPORT VECTOR MACHINES WITH A NEW ALGORITHM COMBINING GRID SEARCH AND PARTICLE SWARM OPTIMIZATION

Yunus ALTUNKOL

Department of Electrical and Electronics Engineering

Supervisor: Asst. Prof. Dr. Mustafa AÇIKKAR

February 2022, 65 pages

Hyperparameter optimization is vital in improving the prediction accuracy of Support Vector Machines (SVM) as in all machine learning algorithms. This study introduces a new hybrid optimization algorithm, namely PSOGS, which consolidates two strong and widely used algorithms; Particle Swarm Optimization (PSO) and Grid Search (GS). This hybrid algorithm experimented on twelve benchmark datasets. The speed of PSOGS and the prediction accuracy of PSOGS-optimized SVM models (PSOGS-SVM) were compared to those of its constituent algorithms (PSO and GS) and another hybrid optimization algorithm (PSOGSA) that combines PSO and Gravitational Search Algorithm (GSA). The prediction accuracies were evaluated and compared in terms of Accuracy and F-Score for classification problems and Root Mean Square Error, Mean Absolute Percentage Error, and Multiple Correlation Coefficient for regression problems. For the sake of reliability, the results of the experiments were obtained by performing 10-fold cross-validation on 30 runs. The results showed that PSOGS yields comparable prediction accuracy with GS, performs much faster than GS, provides slightly better results with less execution time than PSO. Besides, PSOGS-SVM presents more effective results than PSOGSA-SVM in terms of both prediction accuracy and execution time. As a result, this study proved that PSOGS is a fast, stable, efficient, and reliable algorithm for optimizing hyperparameters of SVM.

Keywords: Support Vector Machines, Support Vector Regression, Hyperparameter Optimization, Grid Search, Particle Swarm Optimization, PSOGS.

ÖZET

IZGARA ARAMA VE PARÇACIK SÜRÜSÜ OPTİMİZASYONUNU BİRLEŞTİREN YENİ BİR ALGORİTMA İLE DESTEK VEKTÖR MAKİNELERİNİN HİPERPARAMETRE OPTİMİZASYONU

Yunus ALTUNKOL
Elektrik Elektronik Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Mustafa AÇIKKAR
Şubat 2022, 65 sayfa

Hiperparametre optimizasyonu, tüm makine öğrenimi algoritmalarında olduğu gibi Destek Vektör Makinelerinin (SVM) tahmin doğruluğunu iyileştirmede de hayati önem taşır. Bu çalışma; iki güçlü ve yaygın olarak kullanılan algoritmayı, Parçacık Sürü Optimizasyonu (PSO) ve Izgara Araması'nı (GS), birleştiren yeni bir hibrit optimizasyon algoritması olan PSOGS'yi tanıtmaktadır. Bu hibrit algoritma, on iki veri kümesi üzerinde denenmiştir. PSOGS'nin hızı ve PSOGS ile optimize edilmiş SVM modellerinin (PSOGS-SVM) tahmin doğruluğu, onu oluşturan algoritmaların (PSO ve GS) ve PSO ile Yerçekimi Arama Algoritmasını (GSA) birleştiren başka bir hibrit optimizasyon algoritması (PSOGSA) ile karşılaştırılmıştır. Tahmin doğrulukları, sınıflandırma problemleri için Tahmin Doğruluğu ve F-Skoru ve regresyon problemleri için Ortalama Kareler Hatası, Ortalama Mutlak Yüzde Hatası ve Çoklu Korelasyon Katsayısı açısından değerlendirilmiş ve karşılaştırılmıştır. Geliştirilen modellerin güvenilirliğini sağlamak için, 30 farklı deneme ve 10 kat çapraz doğrulama yapılarak deneylerin sonuçları elde edilmiştir. Sonuçlar, PSOGS'nin GS ile karşılaştırılabilir tahmin doğruluğu sağladığını, GS'den çok daha hızlı performans gösterdiğini, PSO'dan daha kısa çalışma süresi ile biraz daha iyi sonuçlar sağladığını göstermiştir. Ayrıca PSOGS-SVM'nin hem tahmin doğruluğu hem de çalışma süresi açısından PSOGSA-SVM'den daha etkili sonuçlar sunduğu görülmüştür. Sonuç olarak bu çalışma, PSOGS'nin SVM hiperparametrelerini optimize etmek için hızlı, kararlı, verimli ve güvenilir bir algoritma olduğunu kanıtlamıştır.

Anahtar Kelimeler: Destek Vektör Makineleri, Destek Vektör Regresyonu, Hiperparametre Optimizasyonu, Izgara Araması, Parçacık Sürü Optimizasyonu, PSOGS

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Asst. Prof. Dr. Mustafa AÇIKKAR for the valuable help and guidance he provided and his patience.

I would also like to thank my beloved family for their invaluable love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

SVM	: Support Vector Machine
SVC	: Support Vector Classification
SVR	: Support Vector Regression
GS	: Grid Search
PSO	: Particle Swarm Optimization
GSA	: Gravitational Search Algorithm
RBF	: Radial Basis Function
γ	: Gamma - RBF width parameter
C	: Capacity (Box) - The regularization parameter
ε	: Epsilon - Insensitivity zone parameter
MSE	: Mean Squared Error
RMSE	: Root Mean Squared Error
R	: Coefficient of Correlation
MAPE	: Mean Absolute Percentage Error
F	: F Score
$Cost_{Eval}$	: Cost function evaluation count for each model
MSE_{Cost}	: MSE result of cost function used in optimization phase of SVR models
F_{Cost}	: F result of cost function used in optimization phase of SVC models
ANOVA	: Analysis of Variance

1. INTRODUCTION

Machine learning algorithms use generalized methods that can be used for different kinds of problems with different ranges of input and output values. For these methods to comply with specific problems and make reliable predictions, they have parameters and hyperparameters that have to be adjusted for each problem separately. Parameters are internal to the model, and their values can be estimated from the learning data, so usually they are not set manually. On the contrary, hyperparameters are external to the model, and their value cannot be estimated from the given data, so they have to be set manually(Kuhn & Johnson, 2013).

Support Vector Machine (SVM) was first identified by Boser, Guyon, and Vapnik in 1992 (Boser et al., 1992) as a supervised algorithm for classification and evolved into a popular machine learning tool for classification and regression problems. It is widely used because of its high accuracy rate and ability to deal with large and diverse datasets (Ben-Hur et al., 2008). Moreover, SVMs can handle linearly inseparable data via mapping data to a higher-dimensional space using the kernel trick (Press et al., 2007).

Although SVMs were first designed to handle classification problems (Support Vector Classification - SVC), they were later reorganized to handle regression problems and this technique is called Support Vector Regression (SVR) (Drucker et al., 1997). As is the case for SVC, studies on SVR have shown that hyperparameter optimization is crucial for SVR to make accurate predictions (Wei et al., 2017; Wu et al., 2009).

The selection of the kernel function also has a big impact on the accuracy rate of SVM. Different kernel functions such as linear, polynomial, radial basis function (RBF), and sigmoid can be used with SVM. RBF was selected as the kernel function in this study because RBF is widely accepted to be working well in practice and is relatively easy to tune (Chang et al., 2010). In this study, different kernels were also tested as preliminary work to justify the selection of RBF, and the results are given in Section 4.2

The accuracy of the developed SVM models is highly related to the optimization of hyperparameters capacity (C) (the regularization parameter). SVR models also need the

parameter epsilon (ε) (the ε-insensitivity zone) to be optimized. The accuracy of the models also depends on the kernel function and its parameters. We selected the RBF kernel function for this study, which uses the RBF width parameter (γ). In order to develop an SVM-based model that achieves high accuracy, the hyperparameters C, ε, and γ have to be optimized.

The optimization of SVM hyperparameters has been investigated by many researchers. So far, Grid Search (GS) (Açıkkar, 2020; Chih-Wei Hsu, Chih-Chung Chang, 2008; Zhang et al., 2018), Bayesian Optimization (Alam et al., 2021), and many heuristic algorithms have been used for the hyperparameter optimization of SVM, such as Particle Swarm Optimization (PSO) (Du et al., 2017; Hoang et al., 2017; Huang & Dun, 2008; Raj & Ray, 2017), Gravitational Search Algorithm (GSA) (Sarafrazi & Nezamabadi-Pour, 2013), Genetic Algorithms (Tao et al., 2019), Ant Colony Optimization (Pan et al., 2020), Fruit Fly Optimization (Hu et al., 2021), and Sine Cosine Algorithm (S. Li et al., 2018). In addition to these algorithms, there are many studies (Al-Thanoon et al., 2019; Fu et al., 2020; Sahu et al., 2020) in the literature that try to improve an existing optimization technique or develop a hybrid one by combining some of these optimization techniques.

Among these algorithms, PSO and GS are two of the most commonly used algorithms for optimizing hyperparameters of SVM. PSO is a fast algorithm which is easy to implement and is considered one of the most powerful algorithms for solving global optimization problems. Nevertheless, PSO has disadvantages, such as premature convergence and falling into a local minimum in high-dimensional space (Larsen et al., 2017). Similarly, GS is also easy to implement and understand, has high optimization accuracy, and has strong global optimization ability. However, unfortunately, GS is an exhaustive approach that requires excessive execution time, especially when the number of parameters to be optimized is large (Liu et al., 2006). These disadvantages of PSO and GS negatively affect their optimization performance.

The contribution of this study is to introduce a new hybrid technique called PSOGS that combines PSO and GS, inheriting their strengths and eliminating their weaknesses. In order to prove this claim, the effect of the proposed algorithm on the prediction performance of the SVR models is compared with that of GS and PSO on the same datasets. Furthermore, another hybrid optimization technique called PSOGSA, which combines the PSO and GSA,

was used for comparison purposes. PSOGSA was introduced by Mirjalili and Mohd Hashim (Mirjalili & Hashim, 2010) with the aim of joining PSO's social thinking ability and GSA's exploration ability. PSOGSA proved to be a good optimization algorithm and was used in other studies to optimize the hyperparameters of SVM (Chen et al., 2019; Zhu et al., 2018).

This study is divided into five sections, and the rest of this study is organized as follows. Similar researches to PSOGS are discussed in Literature Review, Section 2. The methodology and the utilized optimization algorithms are presented in Section 3. In Section 4, descriptive information on the datasets used in this study is given first, then results of preliminary work are given, lastly developed SVM models are compared, and experimental results are discussed. Finally, the conclusions are summarized in Section 5.

2. LITERATURE REVIEW

Optimization of hyperparameters of SVM has been investigated by many researchers and some of them are listed in Introduction Section. This section focuses on similar researches with this study that combines PSO and GS or that changes the characteristic of PSO by modifying it to work on discrete search space.

2.1. PSO and GS Hybrid Researches

PSO and GS have been attracting researchers to work on fusing their power. Other studies investigated combinations of PSO and GS (Demidova & Klyueva, 2018; Hongmei et al., 2019; Khajeh et al., 2017; L. Li et al., 2021; Xiao et al., 2015).

Xiao et al. (Xiao et al., 2015) studied a PSO and GS hybrid method for parameter optimization of SVM. They used GS with a large step size in order to narrow down the search space and used PSO for detailed research in this confined search space. They utilized 5-fold cross-validation for error generalization. They experimented on the ORL face database.

Khajeh et al. (Khajeh et al., 2017) used a combination of GS, Univariate Method, and PSO for optimum design of steel frame structures. They used GS to divide search space into grids and then used a method derived from the Univariate Method for determining a smaller interval for PSO. They refined the outcome of PSO by introducing new intervals when particles reach the limits of the interval. They tested their work on three weight minimization problems of steel frames.

Demidova and Klyueva (Demidova & Klyueva, 2018) also investigated the PSO and GS hybrid method's effect on data classification through optimizing SVM parameters. Unlike the previous two discussed researches, Demidova and Klyueva first used PSO in the flow of their method. At each iteration of PSO, the particle with the best objective function value was considered to be the center of the search space for GS. Then the best result of GS was accepted as a new particle of the swarm and the particle that yielded the worst results at the PSO phase was removed from the swarm.

Hongmei et al. (Hongmei et al., 2019) used PSO-GS-SVM hybrid models to predict photovoltaic power. They first used Feature Selection on collected sample data to eliminate abnormal portions. After selecting and normalizing data, they used PSO to optimize the parameters of SVM and identified a smaller search space for GS to reach the final solution.

Li et al. (L. Li et al., 2021) investigated the prediction of welding deformation and utilized a hybrid PSO-GS-SVR model for their research. They first normalized data, then determined the parameters and search space for PSO. They used a large interval for PSO since it is a fast search algorithm. They used the optimal results of PSO as the central point and created the grids of GS with a small step size. This way they obtained the final results for optimal hyperparameters of SVR.

All these discussed experiments used PSO and GS one after another, with the aim of determining the parameters of the latter. PSOGS differs from these previous studies mainly in two aspects. First is the way PSO and GS combined and the second is that the novel PSOGS works on a discrete search space. This experiment is also separated from the previous ones by the method used to compare the experiment results of PSO-based algorithms.

2.2. Discrete PSO Researches

PSO naturally works on continuous search space, so some adaptations have to be made to make it operate on discrete search space. Discrete PSO version has also been subject to other studies (Clerc, 2004; Kennedy & Eberhart, 1997; Yoshida et al., 2000).

Kennedy and Eberhart (Kennedy & Eberhart, 1997)introduced a discrete binary PSO to solve binary problems. They represented the particle's position as a binary value. They discussed that although a particle's overall velocity can be described by the number of bits that flipped, this is not enough to describe a single bit's rate of change. They solved this problem by defining the velocity of a single bit as the probability of its value turning to one. This changed the concept that particles' being potential solutions in original PSO to the particles' being probabilities. That is because a particle with a particular velocity can have a different position at every iteration. They tested their binary PSO on De Jong's test suite of five functions (de Jong, 1975).

Yoshida et al. (Yoshida et al., 2000) utilized PSO to handle both continuous and discrete variables of the Voltage Control problem which is formulated as a mixed-integer nonlinear optimization problem (MINLP). They used discrete random numbers in PSO formulas in order to obtain discrete results for the required dimensions in search space. They applied their method to different practical power systems and collected promising results. This proved discrete PSO's usability on real-world problems.

Clerc (Clerc, 2004) studied a discrete PSO and observed its strength and limits on Traveling Salesman Problem (TSP). He redefined the basic concepts of PSO, namely position and velocity, and also the operations on these concepts to be able to use them on TSP. He defined the position of a particle as a route traveling all nodes and ending at the start point (N-cycle) and this led to the definition of the search space which is the finite set of N-cycles. He described velocity as a function of transpositions when applied to a position that returns another position. Thus, he defined the movement of the particles by switching the order of nodes. He stated that his aim was not to compete with TSP dedicated algorithms but to prove that PSO can successfully be used for this problem.

PSOGS differs from these studies by the definition of its search space. It is designed to optimize the parameters of SVM. The search space is the indices of grids of GS that correspond to real numbers in the continuous search space and are designed on a logarithmic scale. Another important feature of PSOGS is that it makes use of memory in order to decrease computational cost.

3. MATERIALS AND METHODS

3.1. Methodology

In this study, twelve different benchmark datasets, described in Section 4.1, were used to develop prediction models; six datasets for classification models and six datasets for regression models. Hyperparameters of each developed model have to be optimized to achieve higher prediction accuracy. To achieve this goal and optimize values of C, ε, and γ, one of the several hyperparameter optimization techniques should be applied to the model. In this study, a new PSO-based hyperparameter optimization algorithm inspired by GS is proposed. Since this study proposes a hybrid of PSO and GS, in order to compare the achieved results, these two hyperparameter optimization algorithms were also used to determine optimal hyperparameters of SVM. The accuracies of the prediction models, cost function evaluation counts, and the execution times of the optimization algorithms were compared with each other for each dataset separately. For further verification, the proposed algorithm results were compared to those of another PSO-based hybrid algorithm, PSOGSA.

3.1.1. Performance Metrics

Classification is the process of identifying and grouping data into predefined categories, or in other words predicting the class of given data points. Therefore, prediction performances of classification methods are evaluated by analyzing the number of correct and wrong predictions. Regression, on the other hand, is used to predict continuous values. Thus, its prediction performance evaluation requires analyzing the proximity of predictions to the real data.

3.1.1.1. Performance Metrics of SVC Models

In this study, the prediction accuracies of developed SVC models were evaluated using Accuracy (Acc) and F-Score (F). Acc and F formulas are derived from Confusion Matrix, which can be described as a table that is used to visualize the prediction performance of an algorithm by comparing the actual and predicted true and false instances of a classification problem.

Table 3. 1 Confusion Matrix

		Predicted Data	
		Positive (PP)	**Negative (PN)**
Actual Data	**Positive (P)**	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

$$Acc = 100 \times (TP + TN)/(P + N) \quad (1)$$

$$F\ = 100 \times ((2 \times TP)) \ / \ ((2 \times TP + FP + FN)) \quad (2)$$

3.1.1.2. Performance Metrics of SVR Models

The prediction performances of regression algorithms are evaluated by analyzing the closeness of their predictions to actual data. For the evaluation of prediction accuracies of SVR models MSE, RMSE, MAPE, and R metrics are used in this study. The formulas of these performance metrics are shown in Equations. (3) - (6).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2 \quad (3)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2} \quad (4)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - Y_i'|}{Y_i} \quad (5)$$

$$R = \sqrt{1 - \frac{\sum_{i=1}^{n}(Y_i - Y_i')^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \quad (6)$$

where *n* is the number of entries for the test set, Y_i and Y_i' are the actual and predicted values of the target variable for each entry and $\bar{Y}$ is the mean of all Y.

As stated in 3.1, besides these metrics, cost function evaluation count ($Cost_{Eval}$) and the execution time of the algorithms (ExecTime) are also presented as an indicator of the performance of the optimization algorithms utilized for both SVC and SVR models. Thus; the values of Acc, F, ExecTime, and $Cost_{Eval}$ are used for the evaluation of SVC models, whereas RMSE, MAPE, R, ExecTime, and $Cost_{Eval}$ are evaluated as the performance measures of each developed SVR model.

3.1.2. One-way Analysis of Variance (ANOVA)

The one-way ANOVA (Montgomery & Runger, 2003) is a test that is used to determine if there are statistically significant differences between means of independent groups. It tests the null hypothesis that all group means are equal ($H_0: \mu_1 = \mu_2 = \mu_3 = \cdots = \mu_n$). The alternative hypothesis H_A states that there is at least one group mean different than the others.

The One-way ANOVA test gives out "p-value" as a result, which is the probability of obtaining results at least as extreme as the observed results of a statistical hypothesis test, assuming that the null hypothesis is correct. This definition indicates that high p-values mean that the H_0 is true, and p-values lower than the significance level mean that the alternative hypothesis is true.

In this study the One-way ANOVA is used to determine if the differences between prediction accuracy results of different models are statistically significant. The significance level is taken 0.05. So, p-values lower than 0.05 indicate that there is a statistically significant difference between prediction accuracy results of different models.

3.1.3. K-fold Cross-Validation

In machine learning, partitioning the dataset into train and test sets is mostly random. The accuracy of the prediction model is affected positively or negatively by this random selection. To avoid this situation, in general, k-fold cross-validation (Anguita et al., n.d.) is used to promote the generalization capability of the prediction models and to achieve more reliable results. For this study, 10-fold cross-validation was utilized to obtain high generalization ability. As the name implies, the original dataset was divided into ten equal-sized subsets (folds) randomly. One of these subsets was kept as the testing subset, and the rest was used as the training subset. This procedure was repeated ten times having each subset used as a testing subset once. In this way, since all subsets were used for both training and testing, the probability of having a misleading result from randomly splitting data is decreased. Finally, the performance metrics were calculated for each fold. Then the mean of the calculated values of each metric was accepted as the final value of the corresponding performance metric.

3.2. Support Vector Machine (SVM)

Support Vector Machines are supervised learning models developed by Vladimir Vapnik and his colleagues in 1992 (Boser et al., 1992). SVM is based on Statistical Learning Theory which is also developed by Vapnik (Vapnik, 1995). It aims to increase predictive accuracy and prevent overfitting to the training data. The basic idea behind SVM is to find a hyperplane that separates different classes of a given problem with the largest distance to the nearest sample data of all classes. Focusing on training data brings the problem of overfitting and SVM addresses this problem by using the structural risk minimization (SRM) principle. SRM tries to minimize the upper limit of the expected risk instead of the error on the training data. This equips SVM with a greater generalization ability and high prediction accuracy.

SVM aims to generate a model from a given training dataset, then predict target values in the test dataset based on that model. Consider given a training dataset of attribute-target pairs:

$$T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n),\} \tag{7}$$

where attributes $x_i \in R^n$, target variables $y_i \in \{-1,1\}$ and $i = 1, 2, 3, \dots, n$. Classifying the given data with SVM requires solving the following optimization problem:

$$min\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i$$
$$\text{subject to } y_i\big((w, x_i) + b\big) - 1 + \xi_i \geq 0,\ \xi_i \geq 0,\ i = 1, 2, 3, \dots, n \tag{8}$$

where ξ_i is a slack variable and $\xi_i \geq 0$ is the relaxation term, $C > 0$ is a constant that is utilized for deciding penalty level for misclassification. The above equations form a constrained convex quadratic programming problem. Lagrange function is used to solve this problem. The dual form of the equation is:

$$\min_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{j=1}^{i} \sum_{i=1}^{n} y_i y_j \alpha_i \alpha_j K\big(x_i, x_j\big)$$
$$\text{subject to } \sum_{i=1}^{n} y_i \alpha_i = 0,\ C \geq \alpha_i \geq 0,\ i = 1, 2, 3, \dots, n \tag{9}$$

where α_i is a Lagrange multiplier for each training sample and $K(x_i, x_j)$ represents the kernel function. As stated before and will be discussed in section 4.2. RBF is selected as the kernel function in this study. RBF has the form of:

$$K(x_i, x_j) = exp(-\gamma \|x - x_i\|^2), y > 0 \quad (10)$$

where γ is the kernel parameter to measure width of RBF kernel function.

These equations also show that two parameters are critical to increase SVM's learning performance and achieve high prediction accuracy. These parameters are the regularization or the penalty parameter C and the RBF width parameter γ.

Although SVM was originally designed for classification problems it can also be used successfully in different kinds of problems. A version of SVM for regression problems was proposed by Vapnik and colleagues in 1996 (Drucker et al., 1997). The regression problem is described by the function:

$$f(x) = \langle w, \varphi(x) \rangle + b \quad (11)$$

where w is the weight coefficient, b is the offset coefficient and $\varphi(x)$ is the kernel function used to transform non-linear problems into a high-dimensional linear problem. Similar to SVM, SVR requires solving the following optimization problem:

$$min \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \varepsilon(y_i - f(x_i))$$

$$\text{subject to } \varepsilon(y_i - f(x_i)) = \begin{cases} 0, & |y_i - f(x_i)| < \varepsilon \\ |y_i - f(x_i)| - \varepsilon, & otherwise \end{cases} \quad (12)$$

where C is the regularization parameter, $\varepsilon(\,)$ is the non-sensitive loss function and ε is the insensitivity zone parameter. In order to make the solving process easier two slack variables ξ_i and ξ_i^* are introduced. Then, the formula becomes:

$$\min_{w,\xi,\xi^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} \langle w, \varphi(x)\rangle + b - y_i \le \varepsilon + \xi_i \\ y_i - \langle w, \varphi(x)\rangle - b \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \tag{13}$$

Again by using Lagrange function, the dual form of the equation is obtained:

$$\min_{\alpha_i,\alpha_i^*} \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\,K(x_i, x_j) + \varepsilon\sum_{i=1}^{n}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{n} y_i(\alpha_i^* + \alpha_i)$$

$$\text{subject to } \begin{cases} \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0 \\ 0 \le \alpha_i^*, \alpha_i \le C \end{cases} \tag{14}$$

where α_i and α_i^* are the Lagrange multipliers for each training sample and $K(x_i, x_j)$ represents the kernel function. As seen from the above formulas, in addition to C and γ SVR has ε the insensitivity zone parameter to be optimized.

In this study, the proposed method was experimented on both classification (SVC) and regression (SVR) models.

3.2.1. Developing Models with SVM

The implementation steps of the SVM-based prediction model are given in detail in Figure 3.1. The algorithms k-fold cross-validation and PSO require some random numbers to be used in the modeling process. In order to get consistent results and compare the optimization algorithms more fairly, these random numbers were generated before the modeling process, and it is ensured that all models with different optimization algorithms use the same random numbers. After initialization, 10-fold cross-validation was applied to the dataset. Then, for each fold, these steps were followed:

- The training and testing sets were generated.
- Hyperparameters of the SVM were optimized by the selected/current algorithm.
- The prediction model was created by using the optimized parameters.
- The target variable was estimated using the prediction model, and performance metrics were calculated and stored.

Then, the results of all 10-folds were used to get an average score. For each dataset, result sets were obtained by using four optimization algorithms.

Training and testing sets were built through 10-fold cross-validation as described in 3.1.2. After that, the training set was divided into sub-training and sub-testing sets via 5-fold cross-validation. These subsets were used in the cost evaluation step of the optimization process. This nested cross-validation technique was used to decrease the effect of randomness and increase generalization capability. All folds and sub folds were built before developing the prediction models so that all models were ensured to be run in exactly the same data portions.

The optimal values of the hyperparameters were found by implementing the optimization algorithms. The optimization algorithms need a cost function to evaluate and compare the error (cost) for each combination of hyperparameters. For this need, F-Score (F) is used as the cost function for classification models (F_{Cost}) and the MSE was used as the cost function (MSE_{Cost}) for regression models. F_{Cost} is calculated as $(100 - F)$ because algorithms are configured to solve minimization problems. In order to build the prediction model at each fold, all optimization algorithms selected the combination of hyperparameters yielding the lowest cost over the training set. The selected combination of hyperparameters was then used to predict the values of target variables in the testing set.

Finally, the accuracy of the prediction model and the speed of the algorithms were evaluated by calculating the prediction accuracy metrics for each fold and then evaluating the mean of them as well as ExecTime and $Cost_{Eval}$.

The computations presented in this study were performed using the MATLAB environment with Statistics and Machine Learning Toolbox (The MathWorks, 2018) on a PC with an 8-core 2.10 GHz CPU and 8 GB memory.

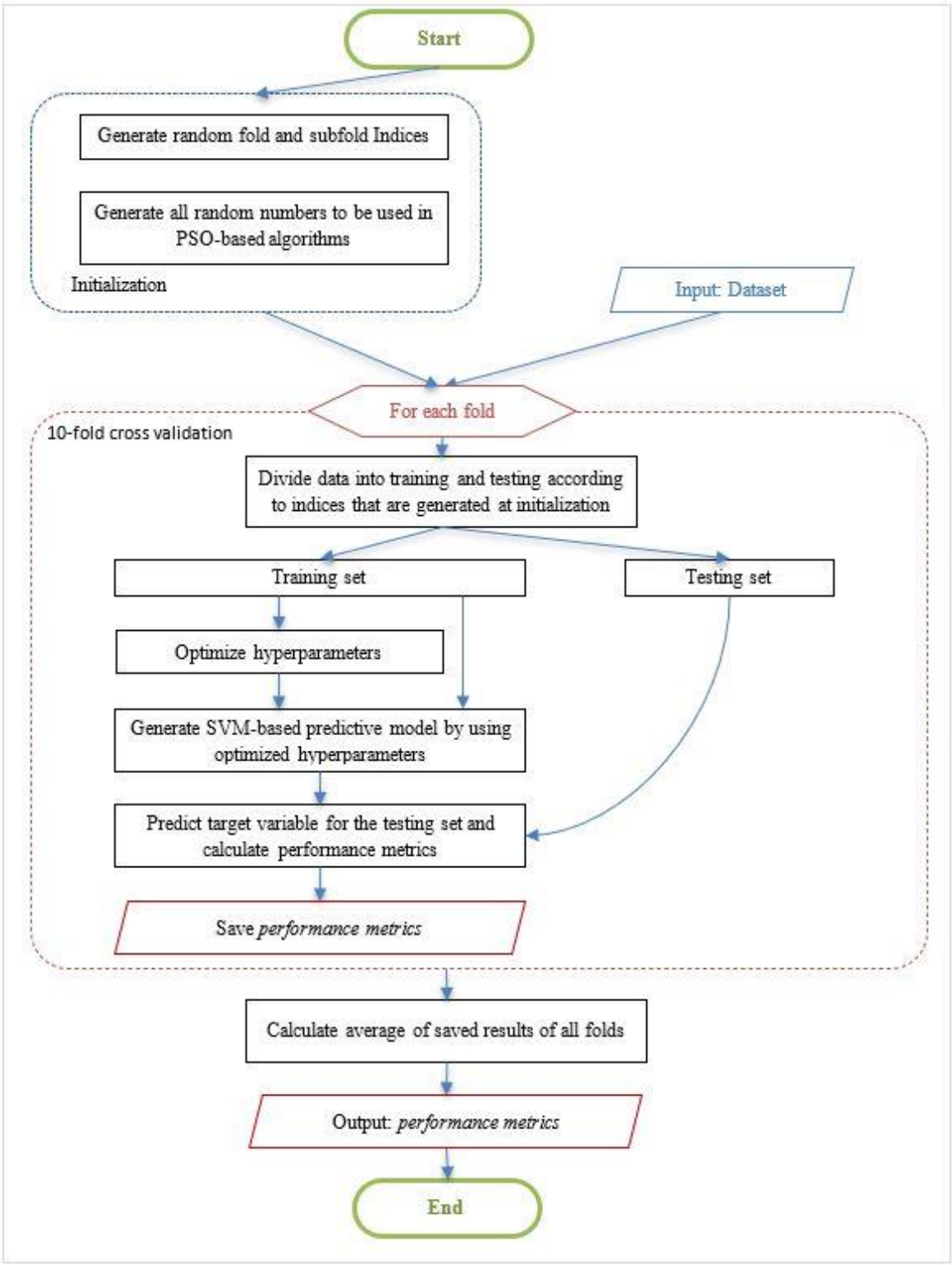


Figure 3. 1 Implementation steps of the SVM-based model

This study focuses on optimizing the hyperparameters of SVM, making use of RBF kernel function, so instead of implementing SVM from scratch, the prediction models were developed by making use of MATLAB's SVM implementation (The MathWorks, 2018).

SVC and SVR were provided by this Machine Learning Toolbox, and the optimization algorithms GS, PSO, and PSOGS were coded by the authors to optimize the hyperparameters of SVM. The source code of PSOGSA was taken from MathWorks' file exchange portal (Mirjalili, 2018) and slightly modified for integration with this study.

3.3. Optimization Algorithms

An effective search algorithm is needed to find the proper values of the hyperparameters of SVM because, as discussed before, SVM requires an optimal selection of the hyperparameters in order to perform accurately. Various optimization algorithms have been proposed to optimize hyperparameters of SVM for improved prediction accuracy in the literature. This study proposed a new hybrid optimization algorithm, PSOGS, based on PSO and GS.

In this section, adequate information is given about the optimization algorithms used in this study.

3.3.1. Grid Search (GS)

GS is one of the most used algorithms for optimizing the hyperparameters of a given SVM model (Açıkkar, 2020; Chih-Wei Hsu, Chih-Chung Chang, 2008; Zhang et al., 2018). Although GS is an exhaustive algorithm, researchers use GS not only for its high accuracy rate but also for its relatively easy implementation and capability of being easily parallelized (Feurer & Hutter, 2019).

In the GS, the predefined range of values of each hyperparameter is divided evenly by a fixed step-size leading to a certain number (GridSize) of discrete points on a logarithmic scale, resulting in a discretized hyperparameter search space. Thus, the sets of hyperparameters are obtained using every single combination of each hyperparameter. Then GS iterates through every hyperparameter set for the given model. At the end of this search process, the hyperparameter set yielding the minimum cost is marked as the optimized hyperparameters of the SVM model.

The pseudo-code of GS is given in Algorithm 3.1.

1: Divide ranges of hyperparameters into logarithmically scaled intervals
2: Combine all values of all hyperparameters to get the search space (*S*)
3: Initialize $minError$ to infinity
4: **foreach** combination of hyperparameters p_i in *S* **do**
5: Calculate cost for the combination ($f(p_i)$)
6: **if** $f(p_i) < minError$ **then**
7: $minError = f(p_i)$
8: $optimum\ combination = p_i$
9: **end**
10: **end**

Algorithm 3. 1 GS Pseudo-code

3.3.2. Particle Swarm Optimization (PSO)

PSO is an evolutionary algorithm that emerged in recent years. Kennedy and Eberhart first proposed it in 1995 (Kennedy & Eberhart, 1995). The idea of the PSO algorithms inspired by the movement of bird flocking or fish schooling. The particles representing the birds or the fishes collaborate with and learn from each other to find an optimal solution. Every particle in the swarm can be a solution to the problem. The particles have three main properties, which are: position (X), velocity (V), and cost value. The cost is obtained from the calculation of the cost function.

Firstly, in the implementation of the PSO, particles are initialized to random positions in the solution space, and the cost of each particle's position is calculated via the cost function. Initial velocities of particles are set randomly within the range of (V_{min}, V_{max}). Then, at each iteration, each particle is moved towards a path that is the vector sum of the vector representing its previous velocity, the vector from the particle's position to its best position, and the vector from the particle's position to the swarm's best position, all multiplied by different coefficients. What is meant by "the best position" is the position that has the least cost. Each particle's best position and swarm's best position are recorded since they are needed for this calculation.

Suppose $P = (p_1, p_2, p_3, \dots, p_n)$ represents the particles of the swarm (S), and the vector representation of the position of the i[th] particle in *d*-dimensional space is the transpose of $X_i = [X_{i1,} X_{i2}, X_{i3}, \dots, X_{id}]$. Similarly, the velocity of the i[th] particle is the transpose of $V_i = [V_{i1,} V_{i2}, V_{i3}, \dots, V_{id}]$; the best position of the i[th] particle is the transpose of the vector $B_i = [B_{i1,} B_{i2}, B_{i3}, \dots, B_{id}]$ and finally, the best position of the swarm is the transpose of $B_g = [B_{g1,} B_{g2}, B_{g3}, \dots, B_{gd}]$. The velocity and the new position of the i[th] particle at (k+1)[th] iteration are calculated by the formulas:

$$V_{id}^{k+1} = w \cdot V_{id}^{k} + c_1 r_1 \left(B_{id}^{k} - X_{id}^{k}\right) + c_2 r_2 \left(B_{gd}^{k} - X_{id}^{k}\right) \quad (15)$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \quad (16)$$

In Equation 15, w is the inertia weight, c_1 and c_2 are the acceleration coefficients, and r_1 and r_2 are random numbers between 0 and 1. In order to keep the particles in the search space, the new position of each particle is compared to the maximum and minimum possible values of the search area at each dimension, namely X_{min} and X_{max}. In order to prevent particles from jumping too much, the velocity is limited with a V_{min} and V_{max}. "Linear Decreasing Inertia Weight" (Bansal et al., 2011) strategy was used to balance PSO's exploration and exploitation. This strategy suggests reducing the inertia weight at the end of every iteration using Equation 17.

$$w_{i+1} = w_{max} - \left((w_{max} - w_{min}) * \left(\frac{i}{MaxIter}\right)\right) \quad (17)$$

where w_{max} is 1.1 and w_{min} is 0.1.

Finally, at the end of the last iteration, the position with the least cost is accepted as the best position, representing the optimal values of the hyperparameters.

The pseudo-code of the PSO algorithm is given in Algorithm 3.2.

1: Input Parameters: *swarm size*, *MaxIter*, X_{max} , X_{min}
2: Initialize parameters: c_1, c_2, w_{max} , w, V_{max} , V_{min}
3: **foreach** particle p_i in S **do**
4: Initialize particle's position (X_i) randomly
5: Initialize particle's velocity (V_i) randomly
6: Calculate cost of the position (f(X_i))
7: Set particle's best position ($p_{i,best}$) to current position
8: **if** f($p_{i,best}$) < f(G_{best}) **then**
9: G_{best}= $p_{i,best}$
10: **end**
11: **end**
12: **for** iter = (1 to *MaxIter*) **do**
13: **foreach** particle p_i in S **do**
14: Update V_i using equation (15)
15: Update X_i using equation (16)
16: Calculate f(X_i)
17: **if** f(X_i) < f($p_{i\ best}$) **then**
18: $p_{i\ best}$= X_i
19: **if** f($p_{i\ best}$) < f(G_{best}) **then**
20: G_{best}= $p_{i\ best}$
21: **end**
22: **end**
23: **end**
24: Update w using equation (17)
25: **end**

Algorithm 3. 2 PSO Pseudo-code

3.3.3. PSOGSA

PSOGSA is a hybrid swarm intelligence algorithm introduced by Mirjalili and Hashim (Mirjalili & Hashim, 2010), which combines PSO and GSA. GSA is a heuristic intelligent optimization algorithm proposed by Rashedi et al. in 2009 (Rashedi et al., 2009). It is inspired by physical laws, mainly by Newtonian gravity and the laws of motion.

Similar to PSO, GSA also uses agents called masses or agents. Each agent is a candidate solution, and agents transfer information between each other by attracting each other via gravitational force. Suppose *N* agents are deployed randomly in a d-dimensional search space for the algorithm. The gravitational force that j^{th} agent applies to i^{th} agent at time *t* is calculated with the formula:

$$F_{ij}^{d}(t) = G(t)\frac{M_i(t)\times M_j(t)}{R_{ij}(t)+\epsilon}(x_j^d(t) - x_i^d(t)) \tag{18}$$

where $G(t)$ represents the gravitational constant, $M_i(t)$ and $M_j(t)$ are masses of i^{th} and j^{th} agents. Masses of agents are calculated in proportion to their fitness value. This suggests that the masses of agents may change over time. $R_{ij}(t)$ is the distance between agents and ϵ is a small constant. The total force applied to an agent is calculated by the formula:

$$F_i^d(t) = \sum_{j=1,j\neq i}^{N} rand_j F_{ij}^d(t) \tag{19}$$

where $rand_j$ is a random number in the interval [0, 1]. Then, acceleration (*ac*), velocity (*V*), and position (*X*) of agents are calculated using these formulas respectively:

$$ac_i^d(t) = F_i^d(t)/M_i^d(t) \tag{20}$$

$$V_i^d(t+1) = rand_i \times V_i^d(t) + ac_i^d(t) \tag{21}$$

$$X_i^d(t) = X_i^d(t) + V_i^d(t+1) \tag{22}$$

where $rand_i$ is a random number in the interval [0, 1]. These equations simulate the movement of agents in the search space. The better an agent's fitness value gets the more its attractive force increases. This approach is used to pull agents towards the optimal solution.

GSA has proven itself to be a good optimization algorithm and has been involved in studies (Sabri et al., 2013; Sarafrazi & Nezamabadi-Pour, 2013; Siddique & Adelib, 2016).

Mirjalili and Hashim introduced PSOGSA with the aim of balancing the exploration and exploitation of the optimization process by combining PSO's social thinking ability and GSA's strength in local search. They combined these methods not by calling one after another but by running them in parallel. They managed this by fusing PSO's and GSA's calculation of particles' velocity. The constructed formula of a particle's velocity is:

$$V_i(t+1) = w \times V_i(t) + c_1 r_1 \times ac_i(t) + c_2 r_2 \times (gbest - X_i(t)) \quad (23)$$

where w, c_1, c_2, r_1, r_2 are the parameters and constants used in PSO.
The pseudo-code of the PSOGSA algorithm is given in Algorithm 3.3.

Mirjalili and Hashim tested their proposed hybrid method on benchmark functions and proved that PSOGSA can perform better than both PSO and GSA. Later, Mirjalili et. al. (Mirjalili et al., 2012) also experimented on training Feedforward Neural Networks (FNN) by using PSOGSA and concluded that it serves well the aim of increasing convergence speed and avoiding falling into a local minimum.

Input Parameters: *swarm size*, *MaxIter*, X_{max} , X_{min}
Initialize parameters: c_1, c_2, w_{max} , w, V_{max} , V_{min}
foreach particle p_i in S **do**
    Initialize particle's position (X_i) randomly
    Initialize particle's velocity (V_i) randomly
end
for iter = (1 to *MaxIter*) **do**
    foreach particle p_i in S **do**
        If particle is outside limits of search space, get it back to boundary
        Calculate f(X_i)
        if f(X_i) < f($p_{i\ best}$) **then**
            $p_{i\ best} = X_i$
            if f($p_{i\ best}$) < f(G_{best}) **then**
                $G_{best} = p_{i\ best}$
            end
        end
        Update F_i using equation (19)
        Update ac_i using equation (20)
        Update V_i using equation (23)
        Update X_i using equation (22)
        end
    end
    Update w using equation (17)
end
foreach particle p_i in S **do** (calculate fitness for the last time)
    Calculate f(X_i)
    if f(X_i) < f($p_{i\ best}$) **then**
        $p_{i\ best} = X_i$
        if f($p_{i\ best}$) < f(G_{best}) **then**
            $G_{best} = p_{i\ best}$
        end
    end
end

Algorithm 3. 3 PSOGSA Pseudo-code

3.3.4. Proposed Algorithm (PSOGS)

PSO is one of the most widely used biomimicry algorithms for optimization problems. Since its introduction in 1995 (Kennedy & Eberhart, 1995), it has received increasing attention from both researchers and academicians and has been used to solve various optimization problems. PSO's advantages include being easy to implement, having only a few parameters to be set, having a fast convergence speed, being effective in global search, and being computationally inexpensive in terms of both memory requirements and CPU speed. However, even though PSO is an efficient and fast search algorithm, it also has some disadvantages, such as premature convergence problem, that is, it can be trapped into a local minimum in high-dimensional space, especially with complex problems (Larsen et al., 2017).

GS in machine learning is the simplest form of tuning hyperparameters for a given model. It simply recalculates the given model for each possible combination of the hyperparameters and selects the set of hyperparameters where the model performance is most accurate. This approach ensures that GS has a high prediction accuracy, but this also brings a disadvantage of high computational time due to the potentially massive number of combinations to be tested. Moreover, cross-validation further increases the execution time and complexity.

In this study, we constructed a novel hybrid algorithm called PSOGS that inherits the superiorities of PSO and GS and overcomes the flaws of PSO and GS mentioned above. Before examining the implementation of PSOGS, the following remarks should be observed. These are the adjustments that are used to integrate PSO with GS.

- Although PSO is particularly suitable for optimizing continuous variables, it has been redesigned to run in a discrete search space, as the GS algorithm does. PSOGS, basically, is the PSO algorithm executed on the grids of GS. The search area is limited to the discrete points that GS evaluates.
- PSOGS operates on indices of hyperparameter values. These indices are created at the first step of GS, dividing value ranges of hyperparameters evenly. The points in the search space of PSOGS are the combinations of these indices.
- The cost function uses the values of hyperparameters corresponding to the points in search space when developing sub-models in the optimization phase of the SVM model (i.e., Point (i, j, k) represents the i^{th} value of C, the j^{th} value of ε, and the k^{th} value of γ).

The other differences of PSOGS implementation from PSO are;

- In PSOGS, the particles are initially located at distinct points at every dimension. So, each particle is a different combination of values of hyperparameters.
- PSOGS operates on indices, but the value of the velocity is not necessarily an integer. For this reason, the evaluated position was rounded to the nearest integer at each dimension.
- If a particle's velocity has a non-zero value in one direction, but its position is not changed in that direction because of the rounding, then it is forced to move one step in that direction.
- Since PSOGS runs in a discrete search space, it is most likely that the particles will cross the same points more than once throughout their search for the best. During the optimization phase, PSOGS uses memory to store previously used hyperparameter sets (points) to prevent the production of identical sub-models and re-evaluation of the cost function. Every time a particle changes its position, PSOGS looks up from the memory to determine whether the corresponding hyperparameters were used before or not. If the hyperparameter set is used before, the process simply uses the previously stored cost and jumps to the next cycle to calculate the new hyperparameter set.

The main difference of PSOGS from GS is the number of times the cost function is evaluated during the SVM optimization process. GS examines all possibilities within the predefined search space, whereas PSOGS examines each particle's position at initialization and for each iteration. So, $Cost_{Eval}$ for GS is the square of GridSize for classification models and cube of GridSize for regression models, while it is at most (#particles × (#iterations + 1)) for PSOGS. Since PSOGS uses memory to prevent recalculation when a point is visited more than once, $Cost_{Eval}$ is expected (and proved) to be much lower.

As stated in the Introduction section PSOGS differs from the previous researches that combine PSO and GS, by the way, PSO and GS combined. In all four of the discussed experiments, PSO and GS are used in turn and the continuous nature of PSO was untouched. Whereas, PSOGS is a redesign of PSO that can operate on discrete search space. In this study, PSOGS was executed on the grids of GS, this way PSO and GS are integrated.

The steps of the PSOGS algorithm are described in Algorithm 3.4.

Input Parameters: *swarm size*, *MaxIter*, X_{max} , X_{min}
Initialize parameters: c_1, c_2, w_{max} , w, V_{max} , V_{min}
foreach particle p_i in S **do**
    Initialize particle's position in all dimensions ($X_{i,d}$) to a random unique point
    Initialize particle's velocity (V_i) randomly
    Calculate cost of the position (f(X_i))
    Store $X_i$and f(X_i) to memory
    Set particle's best position ($p_{i,best}$) to current position
    if f($p_{i,best}$) < f(G_{best}) **then**
        G_{best}= $p_{i,best}$
    end
end
for iter = (1 to *MaxIter*) **do**
    foreach particle p_i in S **do**
        Update V_i using equation (15)
        Update X_i using equation (16)
        Round X_i to nearest integer
        If $V_{i,d}$ had a non-zero value before rounding but rounded to zero **then**
            Force particle to move one step in that direction
        end
        If X_i exists in memory **then**
            Get f(X_i) from memory
        else
            Calculate f(X_i)
        end
        Store X_i and f(X_i) to memory
        if f(X_i) < f($p_{i\ best}$) **then**
            $p_{i\ best}$= X_i
            if f($p_{i\ best}$) < f(G_{best}) **then**
                G_{best}= $p_{i\ best}$
            end
        end
    end
    Update w using equation (17)
end

Algorithm 3. 4 PSOGS Pseudo-code

4. EXPERIMENTAL ANALYSIS

In this section, firstly, benchmark datasets used in this study were introduced. Then results of some preliminary work were conducted to verify kernel function selection for SVM and parameters selection for optimization algorithms were given. Finally, the results of the actual experiments to evaluate the effect of the four optimization algorithms on SVM performance were compared.

4.1. Overview of Datasets

In this study, twelve medium-scaled datasets from UCI Machine Learning Repository (UCI Machine Learning Repository: Data Sets) were used to test the proposed algorithm and compare its performance with other used hyperparameter optimization algorithms. In order to evaluate the algorithm's effect on classification models, Breast Cancer Wisconsin (BC), Mammographic Mass (MM), SPECT Heart Data (SPH), Heart Disease (HD), Statlog Heart (SH), and Ionosphere (ION) datasets are used. For regression models, QSAR Fish Toxicity (FTX), Concrete Compressive Strength (CON), Concrete Slump Test (CST), Energy Efficiency, and Real Estate Valuation (REV) datasets are used. Two datasets were derived from the Energy Efficiency dataset because it includes two target variables; Heating Load and Cooling Load. The EEH and EEC datasets refer to Energy Efficiency dataset with Heating Load and Cooling Load, respectively. Table 4.1 and 4.2 show the main properties and the descriptive statistics of the datasets used for SVC and SVR respectively.

Table 4. 1 Descriptive statistics of the datasets for SVC

Dataset			Target Description		
Name	**Instance Count**	**Attributes**	**Field Name**	**Positive Instance Count**	**Negative Instance Count**
BC	683	Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nuclei, Mitoses	Class (2 = benign, 4 = malignant)	(4 = malignant) 239	(2 = benign) 444
MM	830	BI-RADS, Age, Shape, Margin, Density	Severity (0 = benign, 1 = malignant)	(1 = malignant) 403	(0 = benign) 427
SPH	267	22 Binary features (partial diagnosis)	Overall Diagnosis (0 = normal, 1 = abnormal)	(1 = abnormal) 110	(0 = normal) 157
HD	303	Age, Sex, Chest Pain Type, Resting Blood Pressure, Serum Cholesterol, Fasting Blood Sugar, Resting ECG, Max. Heart Rate, Exercise-Induced Angina, Oldpeak, Slope, Number of Major Vessels, Thal	Diagnosis of heart disease (0 = no presence, 1 = presence)	(1 = presence) 165	(0 = no presence) 138
SH	270	Age, Sex, Chest Pain Type, Resting Blood Pressure, Serum Cholesterol (mg/dl), Fasting Blood Sugar, Resting EC Results, Max Heart Rate, Exercise-Induced Angina, Oldpeak, Slope, No of Major Vessels, Thal	Heart Disease (1 = absence, 2 = presence)	(2 = presence) 120	(1 = absence) 150
ION	351	34 continuous valued attributes	Output (g = good, b = bad)	(b = bad) 126	(g = good) 225

Table 4. 2 Descriptive statistics of the datasets for SVR

Dataset			Target Description			
Name	**Instance Count**	**Attributes**	**Field Name**	**Min.**	**Max.**	**Mean ± Standard Deviation**
CON	1030	Cement (kg), Blast Furnace Slag (kg), Fly Ash (kg), Water (kg), Superplasticizer (kg), Coarse Aggregate (kg), Fine Aggregate (kg), Age (kg)	Compressive Strength (MPa)	2.33	82.60	35.8 ± 16.7
FTX	908	CIC0, SM1_Dz(Z), GATS1i, NdsCH, NdssC, MLOGP	LC50 (mol/L)	0.05	9.61	4.06 ± 1.5
EEH	768	Relative Compactness, Surface Area (m^2), Wall Area (m^2), Roof Area (m^2), Overall Height (m), Orientation, Glazing Area (m^2), Glazing Area Distribution	Heating Load (kW)	6.01	43.01	22.3 ± 10.1
EEC			Cooling Load (kW)	10.90	48.03	24.6 ± 9.5
REV	414	Transaction Date, House Age (year), Distance to the Nearest MRT Station (m), Convenience Stores, Latitude (degree), Longitude (degree)	House Price of Unit Area (NT$)	7.60	117.50	38.0 ± 13.6
CST	103	Cement (kg), Slag (kg), Fly ash (kg), Water (kg), SP (kg), Coarse Aggr. (kg), Fine Aggr. (kg)	28-day Compressive Strength (Mpa)	17.19	58.53	36.0 ±7.8

4.2. Kernel Function Selection

As mentioned in the Introduction Section, SVM can be used with different kernel functions, and among these functions, RBF was selected for this study. Different kernel functions were tested as preliminary work to rationalize this selection. GS-SVM models with GridSize of 10 were run on BC, MM, SPH, HD, SH, and ION datasets for classification models and on CST, REV, EEC, EEH, FTX, and CON datasets for regression models. For each kernel function, GS-SVM was run 5 times and the average of the experiment results are accepted as final for preliminary work. The results can be found in Tables 4.3 and 4.4. For each dataset and performance metric, the best results are written in bold.

Table 4. 3 Kernel Selection Test Results for SVC Datasets

Dataset	Kernel Function	F_{Cost} (%)	Acc (%)	ExecTime (m)
BC	Polynomial	4.13	**96.93**	35.64
	RBF	**3.99**	96.90	**0.20**
	Linear	4.20	96.49	17.88
	Sigmoid	4.40	96.34	**0.20**
MM	Polynomial	**14.98**	83.25	133.74
	RBF	15.04	**83.49**	11.16
	Linear	15.70	82.29	20.21
	Sigmoid	16.72	82.05	**0.50**
SPH	Polynomial	33.58	68.55	19.89
	RBF	**33.13**	**70.13**	0.20
	Linear	33.20	68.22	11.50
	Sigmoid	38.45	63.02	**0.13**
HD	Polynomial	13.39	83.48	20.23
	RBF	**13.30**	**83.49**	**0.15**
	Linear	13.46	82.84	12.43
	Sigmoid	15.14	78.17	0.16
SH	Polynomial	18.93	83.33	16.32
	RBF	18.50	84.02	**0.14**
	Linear	**18.44**	**84.07**	10.47
	Sigmoid	19.62	79.63	0.15
ION	Polynomial	9.48	**86.03**	12.70
	RBF	**9.46**	**86.03**	**0.14**
	Linear	9.78	84.60	0.16
	Sigmoid	9.87	85.75	**0.14**

Table 4.3 shows that RBF has led to similar or better accuracy in less time compared to Polynomial and Linear kernel functions for classification models. Execution time for Sigmoid kernel function is less than RBF, but regarding prediction accuracy, RBF is a better choice.

Table 4.4 shows that RBF has led to slightly better accuracy than the Polynomial kernel function and much better than Linear and Sigmoid kernel functions on regression models. Considering the accuracy and execution time of the models, these results favor the selection of the RBF kernel function.

Table 4. 4 Kernel Selection Test Results for SVR Datasets

Dataset	**Kernel Function**	**MSE_{Cost}**	**RMSE**	**ExecTime (m)**
CST (*MPa*)	Polynomial	0.33	0.53	203.81
	RBF	**0.27**	**0.47**	**2.05**
	Linear	7.22	2.61	38.40
	Sigmoid	2.51	1.86	5.00
REV (*NT$*)	Polynomial	53.85	7.16	971.62
	RBF	**53.81**	**7.12**	**26.46**
	Linear	77.75	8.62	183.76
	Sigmoid	73.63	8.37	149.95
EEC (*kW*)	Polynomial	**0.21**	**0.46**	2021.57
	RBF	**0.21**	**0.46**	**141.57**
	Linear	9.12	2.97	409.70
	Sigmoid	10.34	3.09	222.63
EEH (*kW*)	Polynomial	2.57	1.52	2087.73
	RBF	**1.71**	**1.18**	**124.09**
	Linear	11.14	3.27	421.51
	Sigmoid	7.58	2.71	305.56
FTX (*mol/L*)	Polynomial	0.83	0.93	2353.36
	RBF	**0.75**	**0.87**	**152.58**
	Linear	0.91	0.97	554.85
	Sigmoid	2.00	1.38	255.90
CON (*MPa*)	Polynomial	32.54	6.07	2988.53
	RBF	**30.82**	**5.65**	**90.57**
	Linear	111.21	10.47	588.54
	Sigmoid	119.24	10.78	128.09

4.3. Parameter Settings

In this study, the goal of compared algorithms was to find the optimal values of three hyperparameters of SVM. Thus, in order to run GS, the ranges of these hyperparameters had to be predefined. Table 4.5 shows the ranges of hyperparameters (search space) along with the number of intervals (GridSize) and the scale type of the hyperparameters utilized in each GS-SVM implementation. The hyperparameters C and γ are used both in SVC and SVR models, ε is used only in SVR models. The range of each hyperparameter given in Table 3 was also used in the PSO, PSOGS, and PSOGSA implementations.

Table 4. 5 Overview of the utilized hyperparameters for SVM-based prediction models

Hyperparameter	Range	GridSize	Scale Type
Capacity (C)	[1, 50000]	10, 20	Logarithmic-scaled
Epsilon (ε)	[0.0001, 100]		
Gamma (γ)	[0.1, 1000]		

PSO algorithm is defined in section 3.3.2. In Eq. 15, there are three constants (w, c_1, c_2) and two random numbers (r_1, r_2). In this study, the values of c_1 and c_2 were taken from the widely accepted study of M. Clerc and J. Kennedy in 2002 (Clerc & Kennedy, 2002), while the values of r_1 and r_2 were calculated separately for each particle, at every iteration. In order to balance PSO's exploration and exploitation "Linear Decreasing Inertia Weight" (Bansal et al., 2011) strategy was used.

In this study, V_{max} was accepted to be equal to 0.75 times (X_{max} - X_{min}), and V_{min} was accepted to be equal to minus V_{max}. Additionally, the number of particles (swarm size) and MaxIter were decided by preliminary work. The results of this preliminary work are given in Tables 4.6 – 4.17. The models created by PSO and PSOGSA are represented by the number of iterations and number of particles used. Therefore, those models are named *[algorithm]–[#iterations]–[#particles]*. Besides these two parameters PSOGS also uses GridSize so PSOGS models are named *PSOGS[GridSize]–[#iterations]–[#particles]*.

Table 4. 6 Performance of algorithms with different parameters on BC dataset

Model	F_{Cost} (%)	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
PSO_10_15	4.00	96.48	94.94	**0.38**	**165**
PSO_10_20	3.99	96.48	94.94	0.49	220
PSO_10_25	**3.98**	**96.66**	**95.28**	0.60	275
PSO_15_20	3.99	96.60	95.14	0.67	320
PSO_15_25	3.99	96.64	95.18	0.80	400
PSOGS10_10_15	4.04	96.34	94.78	**0.13**	**41**
PSOGS10_10_20	4.03	**96.36**	**94.82**	0.15	50
PSOGS10_10_25	4.00	96.28	94.74	0.17	57
PSOGS10_15_20	3.99	96.28	94.74	0.17	51
PSOGS10_15_25	**3.98**	96.28	94.74	0.18	60
PSOGS20_10_15	3.86	96.10	94.58	**0.21**	**59**
PSOGS20_10_20	3.86	96.14	94.62	0.23	69
PSOGS20_10_25	**3.84**	**96.24**	**94.72**	0.27	80
PSOGS20_15_20	3.86	96.20	94.70	0.27	72
PSOGS20_15_25	**3.84**	96.20	94.68	0.30	84
PSOGSA_10_15	3.98	**96.54**	**95.06**	**0.41**	**165**
PSOGSA_10_20	3.97	96.52	95.02	0.52	220
PSOGSA_10_25	**3.94**	96.50	95.02	0.65	275
PSOGSA_15_20	3.98	96.46	94.90	0.72	320
PSOGSA_15_25	3.96	96.52	94.98	0.86	400

Table 4. 7 Performance of algorithms with different parameters on MM dataset

Model	F_{Cost} (%)	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
PSO_10_15	15.82	**83.30**	**82.50**	**10.55**	**165**
PSO_10_20	15.76	82.88	82.12	12.97	220
PSO_10_25	15.74	82.96	82.18	15.67	275
PSO_15_20	15.73	82.92	82.12	21.82	320
PSO_15_25	**15.70**	82.78	82.00	25.28	400
PSOGS10_10_15	16.03	83.04	82.28	**8.44**	**41**
PSOGS10_10_20	**15.97**	**83.28**	**82.50**	9.18	50
PSOGS10_10_25	15.98	83.20	82.42	9.30	57
PSOGS10_15_20	16.08	82.78	82.06	8.90	51
PSOGS10_15_25	16.00	82.90	82.14	9.43	60
PSOGS20_10_15	15.94	83.04	82.20	**12.97**	**59**
PSOGS20_10_20	**15.91**	**83.10**	**82.30**	14.65	69
PSOGS20_10_25	15.92	83.02	82.20	15.94	80
PSOGS20_15_20	15.92	83.02	82.22	16.56	72
PSOGS20_15_25	15.89	83.04	82.24	17.45	84
PSOGSA_10_15	15.82	82.96	82.12	**12.54**	**165**
PSOGSA_10_20	15.76	82.80	82.00	16.78	220
PSOGSA_10_25	15.72	82.92	82.10	21.36	275
PSOGSA_15_20	**15.71**	83.00	82.18	26.07	320
PSOGSA_15_25	15.73	**83.06**	**82.24**	30.60	400

Table 4. 8 Performance of algorithms with different parameters on SPH dataset

Model	F_{Cost} (%)	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
PSO_10_15	33.33	70.04	60.54	**0.47**	**165**
PSO_10_20	**33.30**	70.26	60.94	0.62	220
PSO_10_25	**33.30**	**70.52**	**61.20**	0.78	275
PSO_15_20	33.31	70.46	61.18	0.87	320
PSO_15_25	33.35	70.42	60.76	1.07	400
PSOGS10_10_15	34.17	70.56	60.98	**0.13**	**32**
PSOGS10_10_20	34.17	70.56	60.98	0.15	38
PSOGS10_10_25	34.12	70.64	61.06	0.17	46
PSOGS10_15_20	**34.04**	70.80	**61.26**	0.16	40
PSOGS10_15_25	34.05	**70.88**	61.34	0.16	48
PSOGS20_10_15	33.67	70.70	61.52	**0.18**	**48**
PSOGS20_10_20	33.64	**71.06**	**61.80**	0.21	56
PSOGS20_10_25	**33.63**	70.46	61.08	0.24	68
PSOGS20_15_20	33.64	70.58	60.80	0.23	61
PSOGS20_15_25	33.68	70.04	59.98	0.27	77
PSOGSA_10_15	33.43	**71.16**	**61.80**	**0.47**	**165**
PSOGSA_10_20	**33.28**	70.50	61.08	0.63	220
PSOGSA_10_25	33.30	70.50	61.30	0.76	275
PSOGSA_15_20	33.36	70.64	61.12	0.89	320
PSOGSA_15_25	33.35	70.72	61.30	1.06	400

Table 4. 9 Performance of algorithms with different parameters on HD dataset

Model	F_{Cost} (%)	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
PSO_10_15	13.42	82.74	84.60	**0.40**	**165**
PSO_10_20	**13.37**	82.40	84.44	0.56	220
PSO_10_25	**13.37**	82.46	84.46	0.64	275
PSO_15_20	13.39	82.86	84.72	0.78	320
PSO_15_25	13.38	**82.90**	**84.82**	0.89	400
PSOGS10_10_15	13.81	82.74	84.72	**0.10**	**37**
PSOGS10_10_20	13.76	82.88	84.82	0.13	44
PSOGS10_10_25	**13.66**	**82.94**	**84.86**	0.15	55
PSOGS10_15_20	13.73	82.74	84.68	0.13	47
PSOGS10_15_25	13.67	82.74	84.66	0.16	58
PSOGS20_10_15	13.59	82.80	84.58	**0.17**	**55**
PSOGS20_10_20	13.55	**83.16**	**84.90**	0.21	65
PSOGS20_10_25	**13.51**	83.02	84.76	0.24	83
PSOGS20_15_20	13.53	83.02	84.82	0.21	67
PSOGS20_15_25	13.52	83.08	84.86	0.23	79
PSOGSA_10_15	13.44	82.14	84.12	**0.42**	**165**
PSOGSA_10_20	13.38	82.02	83.90	0.58	220
PSOGSA_10_25	13.37	81.98	84.00	0.65	275
PSOGSA_15_20	**13.34**	**82.20**	84.18	0.84	320
PSOGSA_15_25	13.37	82.18	**84.22**	0.91	400

Table 4. 10 Performance of algorithms with different parameters on SH dataset

Model	F_{Cost} (%)	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
PSO_10_15	17.66	84.00	80.76	**0.30**	**165**
PSO_10_20	17.62	**84.08**	**80.86**	0.39	220
PSO_10_25	**17.45**	83.94	80.60	0.48	275
PSO_15_20	**17.45**	84.00	80.62	0.53	320
PSO_15_25	17.47	83.84	80.40	0.62	400
PSOGS10_10_15	18.01	**84.42**	**81.34**	**0.09**	**39**
PSOGS10_10_20	18.01	84.36	81.20	0.11	48
PSOGS10_10_25	17.96	84.36	81.20	0.12	56
PSOGS10_15_20	**17.91**	84.08	80.88	0.11	50
PSOGS10_15_25	17.93	84.30	81.12	0.12	58
PSOGS20_10_15	17.71	**83.92**	**80.56**	**0.12**	**56**
PSOGS20_10_20	17.61	83.40	79.98	0.17	72
PSOGS20_10_25	**17.54**	83.86	80.52	0.18	82
PSOGS20_15_20	17.68	83.70	80.26	0.16	72
PSOGS20_15_25	17.61	83.70	80.26	0.17	83
PSOGSA_10_15	17.53	83.92	**80.70**	**0.31**	**165**
PSOGSA_10_20	**17.50**	83.86	80.62	0.38	220
PSOGSA_10_25	17.57	**84.00**	**80.70**	0.44	275
PSOGSA_15_20	17.55	83.56	80.24	0.53	320
PSOGSA_15_25	17.51	83.84	80.50	0.59	400

Table 4. 11 Performance of algorithms with different parameters on ION dataset

Model	F_{Cost} (%)	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
PSO_10_15	9.35	**85.10**	**88.96**	**0.32**	**165**
PSO_10_20	9.37	84.92	88.86	0.41	220
PSO_10_25	**9.33**	84.74	88.72	0.50	275
PSO_15_20	9.34	84.86	88.84	0.57	320
PSO_15_25	9.34	84.96	88.92	0.68	400
PSOGS10_10_15	9.51	84.56	88.62	**0.08**	**40**
PSOGS10_10_20	9.49	84.58	88.63	0.10	47
PSOGS10_10_25	**9.43**	**84.63**	**88.64**	0.11	56
PSOGS10_15_20	9.49	84.56	88.62	0.10	50
PSOGS10_15_25	9.47	84.62	88.63	0.11	59
PSOGS20_10_15	9.39	**85.02**	**88.92**	**0.11**	**54**
PSOGS20_10_20	9.40	84.84	88.80	0.13	67
PSOGS20_10_25	9.38	84.84	88.80	0.15	80
PSOGS20_15_20	**9.34**	84.76	88.74	0.14	72
PSOGS20_15_25	9.35	84.72	88.72	0.16	84
PSOGSA_10_15	9.39	**85.14**	**88.98**	**0.32**	**165**
PSOGSA_10_20	9.32	84.84	88.80	0.41	220
PSOGSA_10_25	**9.28**	84.90	88.82	0.50	275
PSOGSA_15_20	9.34	84.50	88.56	0.57	320
PSOGSA_15_25	9.30	84.86	88.78	0.67	400

Table 4. 12 Performance of algorithms with different parameters on CST dataset

Model	MSE_{Cost} (MPa)	RMSE (MPa)	ExecTime (m)	$Cost_{Eval}$
PSO_10_15	0.56	0.69	**1.55**	**165**
PSO_10_20	0.56	0.69	1.83	220
PSO_10_25	**0.55**	**0.68**	2.35	275
PSO_15_20	**0.55**	**0.68**	4.49	320
PSO_15_25	**0.55**	**0.68**	5.65	400
PSOGS10_10_15	0.60	0.70	**0.46**	**84**
PSOGS10_10_20	**0.59**	0.69	0.52	96
PSOGS10_10_25	**0.59**	**0.68**	0.55	112
PSOGS10_15_20	0.60	0.71	0.55	100
PSOGS10_15_25	**0.59**	0.70	0.61	117
PSOGS20_10_15	0.56	0.70	**0.83**	**102**
PSOGS20_10_20	**0.55**	**0.68**	1.29	138
PSOGS20_10_25	**0.55**	0.69	1.34	161
PSOGS20_15_20	**0.55**	0.69	1.46	162
PSOGS20_15_25	**0.55**	0.69	1.53	184
PSOGSA_10_15	1.05	0.88	**1.08**	**165**
PSOGSA_10_20	0.97	0.81	1.77	220
PSOGSA_10_25	0.57	**0.70**	2.03	275
PSOGSA_15_20	0.59	0.71	4.97	320
PSOGSA_15_25	**0.56**	0.71	5.78	400

Table 4. 13 Performance of algorithms with different parameters on REV dataset

Model	MSE_{Cost} (MPa)	RMSE (MPa)	ExecTime (m)	$Cost_{Eval}$
PSO_10_15	60.83	7.72	**5.40**	**165**
PSO_10_20	60.01	7.70	7.23	220
PSO_10_25	59.55	**7.64**	9.36	275
PSO_15_20	59.45	7.68	11.41	320
PSO_15_25	**59.38**	7.69	14.86	400
PSOGS10_10_15	60.93	7.61	**4.31**	**88**
PSOGS10_10_20	60.29	7.63	4.49	98
PSOGS10_10_25	**59.54**	**7.56**	5.16	114
PSOGS10_15_20	59.96	7.58	5.35	114
PSOGS10_15_25	59.92	**7.56**	5.63	128
PSOGS20_10_15	60.23	7.65	**6.50**	**126**
PSOGS20_10_20	59.80	7.62	8.83	166
PSOGS20_10_25	59.55	7.64	9.73	199
PSOGS20_15_20	59.69	7.64	10.22	207
PSOGS20_15_25	**59.53**	**7.60**	11.79	253
PSOGSA_10_15	61.78	**7.68**	**5.78**	**165**
PSOGSA_10_20	61.49	7.73	7.43	220
PSOGSA_10_25	**60.48**	**7.68**	10.06	275
PSOGSA_15_20	60.59	**7.68**	10.63	320
PSOGSA_15_25	60.81	7.71	14.36	400

Table 4. 14 Performance of algorithms with different parameters on EEC dataset

Model	MSE_{Cost} (MPa)	RMSE (MPa)	ExecTime (m)	$Cost_{Eval}$
PSO_10_15	1.85	1.33	**44.80**	**165**
PSO_10_20	1.82	**1.30**	65.61	220
PSO_10_25	1.82	**1.30**	74.74	275
PSO_15_20	1.75	1.33	219.27	320
PSO_15_25	**1.74**	1.34	255.49	400
PSOGS10_10_15	1.90	1.34	**36.52**	**101**
PSOGS10_10_20	1.85	1.32	48.44	129
PSOGS10_10_25	**1.80**	**1.30**	55.94	153
PSOGS10_15_20	1.85	1.34	56.75	142
PSOGS10_15_25	**1.80**	1.32	61.44	163
PSOGS20_10_15	1.73	1.28	**80.94**	**124**
PSOGS20_10_20	1.75	1.30	92.80	149
PSOGS20_10_25	1.73	1.28	95.34	187
PSOGS20_15_20	**1.70**	**1.26**	140.59	195
PSOGS20_15_25	1.71	1.27	145.76	218
PSOGSA_10_15	1.88	1.37	**58.94**	**165**
PSOGSA_10_20	1.93	1.41	73.78	220
PSOGSA_10_25	1.82	1.40	79.60	275
PSOGSA_15_20	**1.78**	**1.36**	220.75	320
PSOGSA_15_25	2.36	1.46	213.81	400

Table 4. 15 Performance of algorithms with different parameters on EEH dataset

Model	MSE_{Cost} (MPa)	RMSE (MPa)	ExecTime (m)	$Cost_{Eval}$
PSO_10_15	0.25	0.49	**35.69**	**165**
PSO_10_20	0.24	0.48	61.45	220
PSO_10_25	**0.23**	0.48	76.08	275
PSO_15_20	**0.23**	0.48	195.99	320
PSO_15_25	**0.23**	**0.47**	230.41	400
PSOGS10_10_15	**0.23**	0.48	41.28	**86**
PSOGS10_10_20	**0.23**	**0.47**	**40.06**	99
PSOGS10_10_25	**0.23**	**0.47**	41.35	109
PSOGS10_15_20	**0.23**	**0.47**	43.89	106
PSOGS10_15_25	**0.23**	**0.47**	42.19	114
PSOGS20_10_15	0.24	0.48	**71.89**	**108**
PSOGS20_10_20	0.24	0.48	100.43	148
PSOGS20_10_25	**0.23**	0.48	104.09	164
PSOGS20_15_20	**0.23**	0.48	106.87	167
PSOGS20_15_25	**0.23**	**0.47**	129.37	197
PSOGSA_10_15	0.26	0.50	**57.48**	**165**
PSOGSA_10_20	**0.23**	0.49	80.41	220
PSOGSA_10_25	0.28	0.52	87.05	275
PSOGSA_15_20	0.28	0.52	216.29	320
PSOGSA_15_25	**0.23**	**0.48**	277.19	400

Table 4. 16 Performance of algorithms with different parameters on FTX dataset

Model	MSE_{Cost} (MPa)	RMSE (MPa)	ExecTime (m)	$Cost_{Eval}$
PSO_10_15	0.86	0.94	**19.42**	**165**
PSO_10_20	**0.85**	**0.93**	31.78	220
PSO_10_25	**0.85**	0.94	36.87	275
PSO_15_20	**0.85**	**0.93**	56.94	320
PSO_15_25	**0.85**	0.94	65.39	400
PSOGS10_10_15	0.78	0.89	**15.16**	**94**
PSOGS10_10_20	0.78	0.89	16.38	123
PSOGS10_10_25	0.78	**0.88**	20.01	146
PSOGS10_15_20	0.78	0.89	17.83	129
PSOGS10_15_25	**0.77**	0.89	19.94	155
PSOGS20_10_15	0.79	**0.89**	**20.88**	**113**
PSOGS20_10_20	**0.78**	**0.89**	24.90	149
PSOGS20_10_25	**0.78**	0.90	28.07	186
PSOGS20_15_20	**0.78**	**0.89**	28.78	179
PSOGS20_15_25	**0.78**	**0.89**	35.69	225
PSOGSA_10_15	0.85	**0.93**	**17.60**	**165**
PSOGSA_10_20	0.85	0.94	42.53	220
PSOGSA_10_25	0.85	**0.93**	41.12	275
PSOGSA_15_20	0.85	0.94	66.75	320
PSOGSA_15_25	**0.84**	**0.93**	65.73	400

Table 4. 17 Performance of algorithms with different parameters on CON dataset

Model	MSE_{Cost} (MPa)	RMSE (MPa)	ExecTime (m)	$Cost_{Eval}$
PSO_10_15	34.48	6.09	**39.05**	**165**
PSO_10_20	31.77	5.88	54.77	220
PSO_10_25	31.65	**5.77**	80.05	275
PSO_15_20	31.54	5.83	137.46	320
PSO_15_25	**31.49**	5.79	204.11	400
PSOGS10_10_15	31.35	5.73	**36.95**	**105**
PSOGS10_10_20	31.12	5.71	42.88	127
PSOGS10_10_25	**30.76**	5.73	48.80	147
PSOGS10_15_20	31.30	**5.60**	49.10	132
PSOGS10_15_25	30.83	5.68	53.45	154
PSOGS20_10_15	30.96	5.69	**58.83**	**131**
PSOGS20_10_20	30.24	5.68	63.68	175
PSOGS20_10_25	30.24	5.65	74.39	210
PSOGS20_15_20	**29.71**	**5.63**	75.68	223
PSOGS20_15_25	29.89	5.66	88.68	261
PSOGSA_10_15	38.69	6.22	**51.46**	**165**
PSOGSA_10_20	32.83	**5.77**	55.78	220
PSOGSA_10_25	33.21	5.87	62.76	275
PSOGSA_15_20	32.17	**5.77**	143.59	320
PSOGSA_15_25	**31.36**	5.79	158.22	400

The best results of the 5 models of each algorithm are indicated in bold. Regarding the results of the preliminary work, in order to have optimum performance in a reasonable time, swarm size was decided to be 25 particles, and the process was decided to be carried out in 10 iterations. These numbers were also observed to be suitable for PSOGS and PSOGSA.

As previously expressed in the Introduction section this research differs from the aforementioned PSO-GS hybridization researches by the technique that is used to compare the results of the proposed method to the results of other algorithms. In this study, at the beginning of the experiments, the random variables used by PSO-based algorithms were determined prior to the execution of the algorithms so that the comparison results would be

free from the effects of randomness. For further elimination of the random nature of PSO from comparison, 10-fold cross-validation was used, moreover, in the final experiments, algorithms were run 30 times and results of each algorithm were given by an average of 30 runs. Additionally, in order to be able to make a fair comparison, the same parameter values are used for PSO, PSOGS, and PSOGSA where applicable.

4.4. Experimental Results

This study has proposed a new hybrid optimization algorithm combining PSO and GS, acquiring PSO's speed and GS's accuracy while eliminating PSO's premature convergence problem and GS's excess time consumption problem. In order to compare the performance of the models created by the new algorithm with those of PSO and GS, all three of them were run on the previously defined datasets. Additionally, the new algorithm was also compared to another PSO-based hybrid algorithm, PSOGSA.

For each dataset, GS and PSOGS were run with two different configurations regarding the GridSize, whereas PSOGS and PSOGSA were run once with the configurations decided by the preliminary work. Thus, a total of six SVM models were created for each dataset. As expressed in 4.3, in order to reduce the effects of randomness on the comparison, 10-fold cross-validation was utilized and each model was run 30 times. The average of 30 runs was taken as final results.

GridSize represents the models created with GS, so they are named in GS[GridSize] form. The models created with other algorithms are named as described in Section 4.3.

When reading and comparing results, all measures should be interpreted according to how they assess the data. For SVC experiments higher Acc and F results, whereas for SVR experiments lower RMSE and MAPE results mean better predictive accuracy is achieved by the optimization algorithm. Similarly, the more R results converge to one, the better the prediction accuracy obtained by the optimization algorithm. Also, less ExecTime and lower $Cost_{Eval}$ means faster execution, better performance.

4.4.1. Results of SVC Models

As stated before, Acc, F, ExecTime, and $Cost_{Eval}$ are the metrics for prediction accuracy comparison of classification models. Tables 4.18 – 4.23 show the results of experiments on datasets selected for classification analysis. Acc and F results were given with a standard deviation of 30 runs' results in *[Average]* ± *[StandardDeviation]* format. For each dataset, the best Acc, F, ExecTime, and $Cost_{Eval}$ results of all six models were indicated in bold and with a green background color. The second-best Acc and F results achieved by a different algorithm than the best result are shown with a blue background. If the second-best results are also results of the same algorithm with the best results, then the second-best results are also shown with a green background, and the following best results are indicated with blue background. Statistically significance of the differences in F results are tested by utilizing One-way ANOVA and the p-value results are given in Table 4.24. Statistically significant differences of algorithms are displayed in bold in this table. Additionally, PSO-based methods' convergences to the best cost in the training phase are given in Figure 4.1.

Table 4. 18 Comparison of algorithms' performance on BC dataset

Model	**Acc (%)**	**F (%)**	**ExecTime (*m*)**	**$Cost_{Eval}$**
GS10	96.60 ± 0.29	95.08 ± 0.44	0.28	100
GS20	96.58 ± 0.32	95.08 ± 0.47	1.11	400
PSO_10_25	**96.67 ± 0.18**	**95.16 ± 0.28**	0.62	275
PSOGS10_10_25	96.59 ± 0.30	95.06 ± 0.42	**0.13**	**51**
PSOGS20_10_25	96.55 ± 0.28	95.02 ± 0.36	0.21	82
PSOGSA_10_25	96.65 ± 0.24	95.14 ± 0.36	0.58	275

Table 4. 19 Comparison of algorithms' performance on MM dataset

Model	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
GS10	**83.43 ± 0.65**	**82.80 ± 0.72**	14.81	100
GS20	**83.38 ± 0.62**	82.75 ± 0.64	58.74	400
PSO_10_25	83.26 ± 0.57	82.73 ± 0.65	13.19	275
PSOGS10_10_25	83.34 ± 0.74	82.79 ± 0.85	**6.94**	**56**
PSOGS20_10_25	83.26 ± 0.53	82.66 ± 0.62	7.66	60
PSOGSA_10_25	83.18 ± 0.64	82.67 ± 0.63	14.18	275

Table 4. 20 Comparison of algorithms' performance on SPH dataset

Model	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
GS10	69.97 ± 1.51	60.72 ± 2.07	0.20	100
GS20	69.58 ± 1.70	60.00 ± 2.61	0.76	400
PSO_10_25	**70.31 ± 1.50**	61.19 ± 2.14	0.62	275
PSOGS10_10_25	**70.31 ± 1.66**	**61.30 ± 2.43**	**0.13**	**48**
PSOGS20_10_25	70.26 ± 1.61	**61.24 ± 2.15**	0.17	68
PSOGSA_10_25	70.25 ± 1.52	61.17 ± 2.01	0.61	275

Table 4. 21 Comparison of algorithms' performance on HD dataset

Model	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
GS10	82.75 ± 0.93	84.83 ± 0.96	0.23	100
GS20	82.50 ± 1.01	84.58 ± 1.01	0.89	400
PSO_10_25	82.57 ± 0.98	84.69 ± 0.92	0.54	275
PSOGS10_10_25	82.71 ± 0.90	84.83 ± 0.88	**0.12**	**52**
PSOGS20_10_25	**82.80 ± 0.86**	**84.86 ± 0.93**	0.17	73
PSOGSA_10_25	82.55 ± 1.02	84.64 ± 1.06	0.55	275

Table 4. 22 Comparison of algorithms' performance on SH dataset

Model	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
GS10	83.64 ± 1.13	80.31 ± 1.50	0.20	100
GS20	83.28 ± 1.27	79.92 ± 1.68	0.78	400
PSO_10_25	83.82 ± 1.05	80.46 ± 1.36	0.50	275
PSOGS10_10_25	83.85 ± 0.85	**80.60 ± 1.16**	**0.10**	**52**
PSOGS20_10_25	83.85 ± 1.03	**80.56 ± 1.44**	0.14	72
PSOGSA_10_25	**83.90 ± 0.94**	80.55 ± 1.31	0.50	275

Table 4. 23 Comparison of algorithms' performance on ION dataset

Model	Acc (%)	F (%)	ExecTime (*m*)	$Cost_{Eval}$
GS10	**85.19 ± 0.94**	**89.08 ± 0.68**	0.17	100
GS20	**85.15 ± 0.78**	**89.03 ± 0.57**	0.68	400
PSO_10_25	84.99 ± 0.89	88.93 ± 0.68	0.51	275
PSOGS10_10_25	84.90 ± 0.67	88.86 ± 0.48	**0.08**	**54**
PSOGS20_10_25	84.99 ± 0.79	88.93 ± 0.55	0.12	78
PSOGSA_10_25	84.89 ± 0.67	88.86 ± 0.54	0.51	275

Tables 4.19 to 4.23 show that all four algorithms yielded close results regarding the prediction accuracy, however, PSOGS-SVC models were much faster than the other algorithms.

Compared to GS-SVC, PSOGS-SVC achieved very close Acc and F results on the BC dataset, and slightly better results on SPH, HD, and SH datasets. PSOGS-SVC achieved these prediction performance results in less time and with fewer $Cost_{Eval}$ than GS-SVC. These results prove the claim that PSOGS-SVC can achieve the prediction performance of GS-SVC in a much shorter optimization time.

Compared to PSO-SVC, PSOGS-SVC achieved the same Acc and F results on SPH and ION datasets, and slightly better results for MM, HD, and SH datasets. PSOGS-SVC achieved these results with less $Cost_{Eval}$ thus in less time than PSO-SVC models. These results showed

that PSOGS-SVC can achieve moderately better prediction accuracy than PSO-SVC in less time.

PSOGS-SVC also achieved better prediction accuracy than PSOGSA-SVC on MM, SPH, HD, and ION datasets. Moreover, PSOGS-SVC used less $Cost_{Eval}$ to reach the final solution thus it was faster than PSOGSA-SVC.

In order to determine statistical significance of the achieved results, F Score of PSOGS models are compared to that of other algorithms pair-wise by using One-way ANOVA. The p-value results are given in Table 4.24.

Table 4. 24 One-way ANOVA test results (p-value) for F results of SVC models

	Datasets					
Algorithm Pair	**BC**	**MM**	**SPH**	**HD**	**SH**	**ION**
PSOGS10 – GS10	0.835	0.987	0.324	0.989	0.405	0.148
PSOGS10 – PSO	0.280	0.759	0.853	0.569	0.676	0.646
PSOGS10 - PSOGSA	0.431	0.513	0.813	0.453	0.868	0.980
PSOGS20 - GS20	0.625	0.597	**0.049**	0.276	0.118	0.493
PSOGS20 - PSO	0.104	0.671	0.933	0.489	0.797	1.000
PSOGS20 - PSOGSA	0.214	0.984	0.892	0.389	0.978	0.637

One-way ANOVA results indicate that the only statistically significant difference between F Score results of PSOGS and other algorithms is the difference of F Score results of PSOGS20 and GS20 on SPH dataset.

Analyzing $Cost_{Eval}$ results in detail shows that PSOGS10 evaluated the cost function 47.8% fewer times than GS10, and 81% fewer times than PSO and PSOGSA on average. PSOGS20 made 82% fewer cost function evaluations than GS20, and 73.76% fewer cost function evaluations than PSO and PSOGSA on average. These results point out that PSOGS decreased the number of cost function evaluations remarkably.

Figure 4.1 shows that in the training phase PSOGS reached the best result it could find with fewer $Cost_{Eval}$ than PSO and PSOGSA.

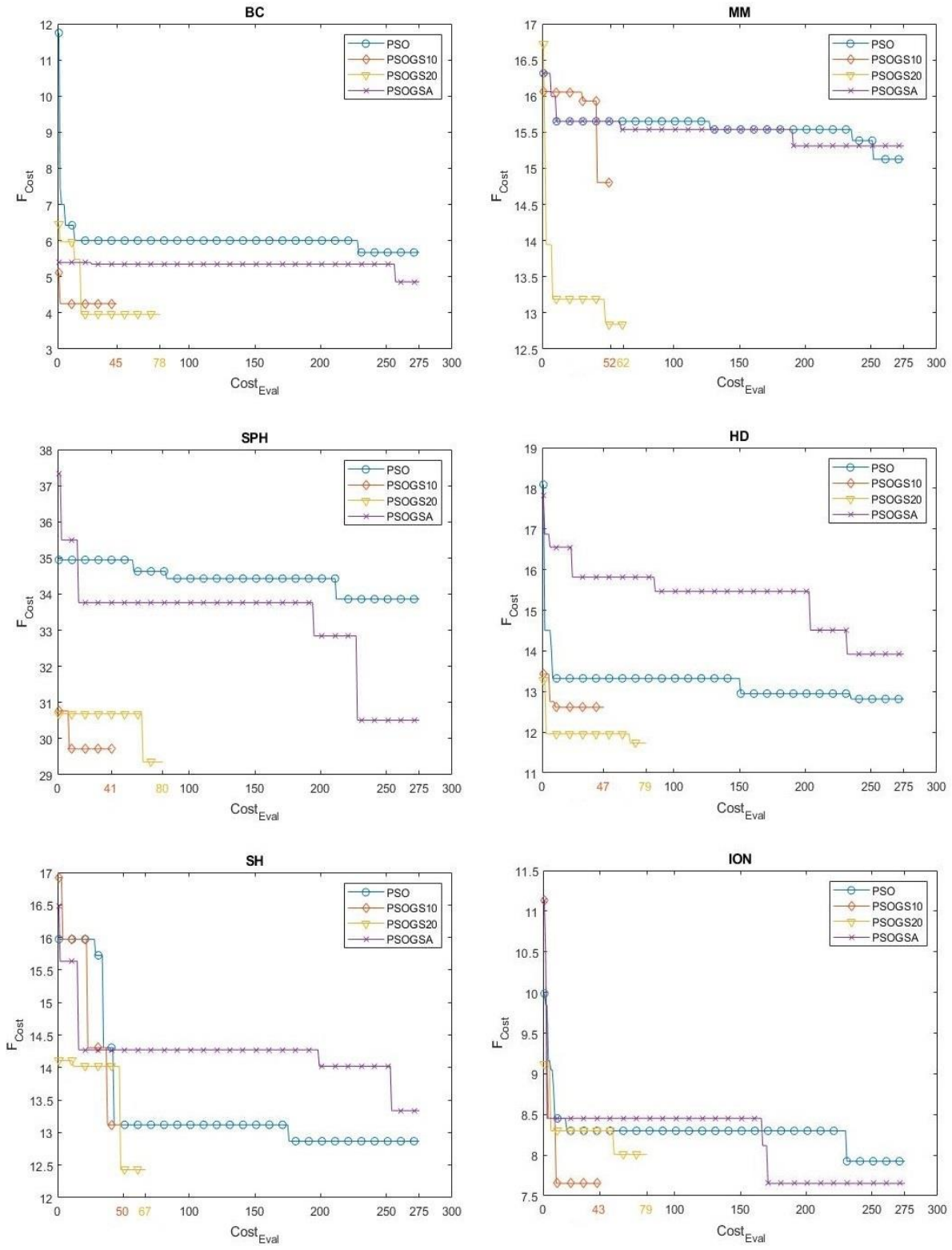


Figure 4. 1 Algorithms' convergence to best cost in the training phase (SVC)

4.4.2. Results of SVR Models

Algorithms' effects on prediction performance are compared with regard to RMSE, MAPE, and R, ExecTime, and $Cost_{Eval}$ for regression models. Tables 4.25 – 4.30 show the results of experiments on datasets selected for regression analysis. RMSE and MAPE results were given with a standard deviation of 30 runs' results in *[Average]* ± *[StandardDeviation]* format. For each dataset, the best RMSE, MAPE, ExecTime, and $Cost_{Eval}$ results of all six models were indicated in bold and with a green background color. The second-best RMSE, MAPE, and R results achieved by a different algorithm than the best result are shown with a blue background. If the second-best results are also results of the same algorithm with the best results, then the second-best results are also shown with a green background, and the following best results are indicated with blue background. Statistically significance of the differences in RMSE results are tested by utilizing One-way ANOVA and the p-value results are given in Table 4.31. Statistically significant differences of algorithms are displayed in bold in this table. In addition to these results, PSO-based methods' convergences to the best cost in the training phase are given in Figure 4.2.

Table 4. 25 Comparison of algorithms' performance on CST dataset

Model	RMSE (*MPa*)	MAPE (%)	R	ExecTime (*m*)	$Cost_{Eval}$
GS10	0.57 ± 0.06	1.20 ± 0.10	**0.996**	2.8	1000
GS20	0.60 ± 0.07	1.25 ± 0.10	0.995	21.7	8000
PSO_10_25	0.57 ± 0.05	1.19 ± 0.09	**0.996**	4.0	275
PSOGS10_10_25	**0.56 ± 0.06**	**1.16 ± 0.11**	**0.996**	**1.0**	**113**
PSOGS20_10_25	0.60 ± 0.06	1.27 ± 0.11	0.995	2.6	165
PSOGSA_10_25	0.59 ± 0.07	1.23 ± 0.15	**0.996**	3.7	275

Table 4. 26 Comparison of algorithms' performance on REV dataset

Model	RMSE (*NT$*)	MAPE (%)	R	ExecTime (*m*)	$Cost_{Eval}$
GS10	**7.35 ± 0.15**	**14.57 ± 0.34**	**0.829**	35.4	1000
GS20	7.40 ± 0.16	14.76 ± 0.29	0.826	302.2	8000
PSO_10_25	7.50 ± 0.17	14.86 ± 0.33	0.819	8.9	275
PSOGS10_10_25	7.43 ± 0.15	14.75 ± 0.27	0.822	**7.1**	**134**
PSOGS20_10_25	7.41 ± 0.15	14.74 ± 0.40	0.822	10.5	206
PSOGSA_10_25	7.62 ± 0.14	15.26 ± 0.48	0.812	10.0	275

Table 4. 27 Comparison of algorithms' performance on EEC dataset

Model	RMSE (*kW*)	MAPE (%)	R	ExecTime (*m*)	$Cost_{Eval}$
GS10	**1.17 ± 0.03**	**3.04 ± 0.07**	**0.992**	278.5	1000
GS20	1.24 ± 0.05	3.21 ± 0.12	0.991	2213.0	8000
PSO_10_25	1.33 ± 0.06	3.21 ± 0.12	0.990	83.9	275
PSOGS10_10_25	1.28 ± 0.10	3.28 ± 0.25	0.990	**48.7**	**136**
PSOGS20_10_25	1.26 ± 0.06	3.21 ± 0.13	0.990	115.0	184
PSOGSA_10_25	1.37 ± 0.07	3.39 ± 0.29	0.989	74.0	275

Table 4. 28 Comparison of algorithms' performance on EEH dataset

Model	RMSE (*kW*)	MAPE (%)	R	ExecTime (*m*)	$Cost_{Eval}$
GS10	0.47 ± 0.01	1.66 ± 0.04	**0.999**	318.6	1000
GS20	**0.46 ± 0.01**	**1.54 ± 0.02**	**0.999**	2474.6	8000
PSO_10_25	0.49 ± 0.01	1.69 ± 0.07	**0.999**	103.6	275
PSOGS10_10_25	0.48 ± 0.01	1.68 ± 0.05	**0.999**	**42.1**	**111**
PSOGS20_10_25	0.48 ± 0.02	1.66 ± 0.10	**0.999**	124.2	177
PSOGSA_10_25	0.55 ± 0.17	1.89 ± 0.46	0.998	89.3	275

Table 4. 29 Comparison of algorithms' performance on FTX dataset

Model	RMSE (*mol / L*)	MAPE (%)	R	ExecTime (*m*)	$Cost_{Eval}$
GS10	**0.87 ± 0.01**	**25.72 ± 0.30**	**0.791**	141.1	1000
GS20	0.88 ± 0.01	25.95 ± 0.35	0.785	1159.1	8000
PSO_10_25	0.95 ± 0.02	27.16 ± 0.63	0.744	30.4	275
PSOGS10_10_25	0.91 ± 0.03	26.54 ± 0.61	0.767	**19.4**	**133**
PSOGS20_10_25	0.91 ± 0.03	26.45 ± 0.63	0.769	27.3	175
PSOGSA_10_25	0.94 ± 0.02	27.07 ± 0.76	0.748	37.4	275

Table 4. 30 Comparison of algorithms' performance on CON dataset

Model	RMSE (*MPa*)	MAPE (%)	R	ExecTime (*m*)	$Cost_{Eval}$
GS10	5.55 ± 0.10	13.12 ± 0.21	0.940	87.9	1000
GS20	**5.49 ± 0.12**	**12.88 ± 0.40**	**0.941**	660.6	8000
PSO_10_25	5.81 ± 0.23	13.37 ± 0.58	0.934	71.2	275
PSOGS10_10_25	5.55 ± 0.16	13.27 ± 0.45	0.940	**43.2**	**138**
PSOGS20_10_25	5.61 ± 0.22	13.01 ± 0.48	0.939	78.9	215
PSOGSA_10_25	5.87 ± 0.28	13.26 ± 0.52	0.932	89.6	275

Considering Tables 4.25 to 4.30, the following remarks can be observed from the results of the prediction models on the datasets:

Comparing PSOGS-SVR with GS-SVR, it can be observed, RMSE, MAPE, and R results for PSOGS-SVR are very close to GS-SVR. PSOGS-SVR slightly passed GS-SVR only for the CST dataset. These results mean that PSOGS-SVR can be an alternative to GS-SVR regarding prediction performance. In addition, PSOGS-SVR achieved these results in far lower time than GS-SVR did and outclassed GS-SVR in terms of ExecTime as expected. These results prove the claim that PSOGS-SVR can achieve the prediction performance of GS-SVR in a much shorter optimization time.

Compared to PSO-SVR, PSOGS-SVR achieved slightly better RMSE and MAPE results, whereas it achieved slightly better or the same R results for all datasets. Regarding the ExecTime measure, PSOGS-SVR is also faster than PSO-SVR. The difference in ExecTime is mainly due to the number of repeating sub-models during the optimization phase. For which PSOGS does not reevaluate the cost; instead, it just looks up the previous result. These results showed that PSOGS-SVR can achieve moderately better prediction accuracy than PSO-SVR in less time.

PSOGS-SVR results were also better than PSOGSA-SVR in both better prediction accuracy and ExecTime.

To determine statistical significance of the achieved results, RMSE result of PSOGS models are compared to that of other algorithms pair-wise by using One-way ANOVA. The p-value results are given in Table 4.31.

Table 4. 31 One-way ANOVA test results (p-value) for RMSE results of SVR models

	Datasets					
Algorithm Pair	**CST**	**REV**	**EEC**	**EEH**	**FTX**	**CON**
PSOGS10 – GS10	0.689	0.212	**3×10^{-12}**	**0.001**	**4×10^{-8}**	0.952
PSOGS10 – PSO	0.391	0.376	0.542	0.414	**2×10^{-6}**	**9×10^{-6}**
PSOGS10 - PSOGSA	0.079	**0.031**	0.146	**0.031**	**5×10^{-5}**	**2×10^{-6}**
PSOGS20 - GS20	**0.027**	0.428	**3×10^{-11}**	**0.019**	**4×10^{-9}**	**0.183**
PSOGS20 - PSO	**0.041**	0.153	**0.008**	0.191	**8×10^{-8}**	**0.001**
PSOGS20 - PSOGSA	0.521	**0.007**	**3×10^{-6}**	**0.026**	**4×10^{-6}**	**0.001**

One-way ANOVA test indicates that the difference of RMSE results of PSOGS and GS are statistically significant for CST, EEC, EEH and FTX datasets. On CST dataset PSOGS achieved better accuracy but on the other datasets GS achieved better accuracy. Considering PSO and PSOGS, the difference of RMSE results is statistically significant for CST, EEC, FTX and CON datasets. On EEC, FTX and CON datasets PSOGS achieved better prediction accuracy. PSOGS10 achieved better prediction accuracy than PSO on CST dataset but the difference is statistically insignificant, but for the same dataset PSO achieved better prediction

accuracy than PSOGS20 and the difference is statistically significant. Compared to PSOGSA, PSOGS's better results are statistically significant on all datasets except CST.

In view of ExecTime results, all tests on all datasets show that PSOGS-SVR runs faster than all three algorithms. It can clearly be observed from the results that the objective of PSOGS being remarkably faster than GS is achieved without a significant loss in the accuracy of the produced models. PSOGS10 and PSOGS20 decreased GS10's and GS20's ExecTime by factors of 5 and 21 respectively. In most cases, PSOGS managed to result in similar accuracy performance results, and as in the case of the CST dataset PSOGS even produced slightly better results. The difference in ExecTime mainly results from $Cost_{Eval}$.

Inspecting $Cost_{Eval}$ results of SVR experiments, it can be perceived that PSOGS10 evaluated the cost function 87.3% fewer times than GS10, and 53.6% fewer times than PSO and PSOGSA on average. Whereas PSOGS20 evaluated the cost function 97.7% fewer times than GS20, and 32% fewer times than PSO and PSOGSA on average. Similar to SVC experiments, these results also favor PSOGS's improvement regarding $Cost_{Eval}$.

Figure 4.2 shows that, as is the case for training SVC models, PSOGS reached the best result it could find with fewer $Cost_{Eval}$ than PSO and PSOGSA when working with SVR models.

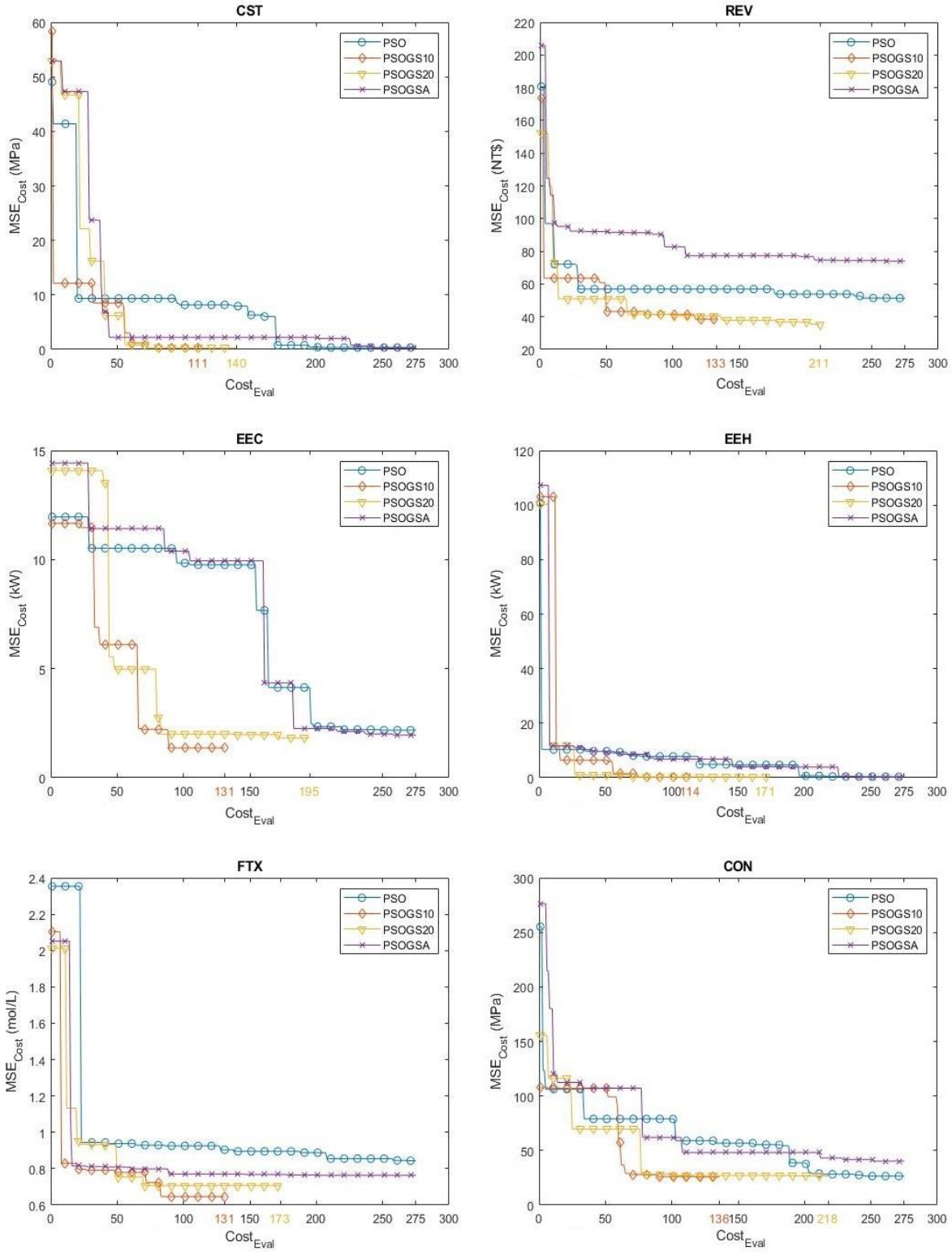


Figure 4. 2 Algorithms' convergence to best cost in the training phase (SVR)

4.4.3. Comparison of SVC and SVR Results

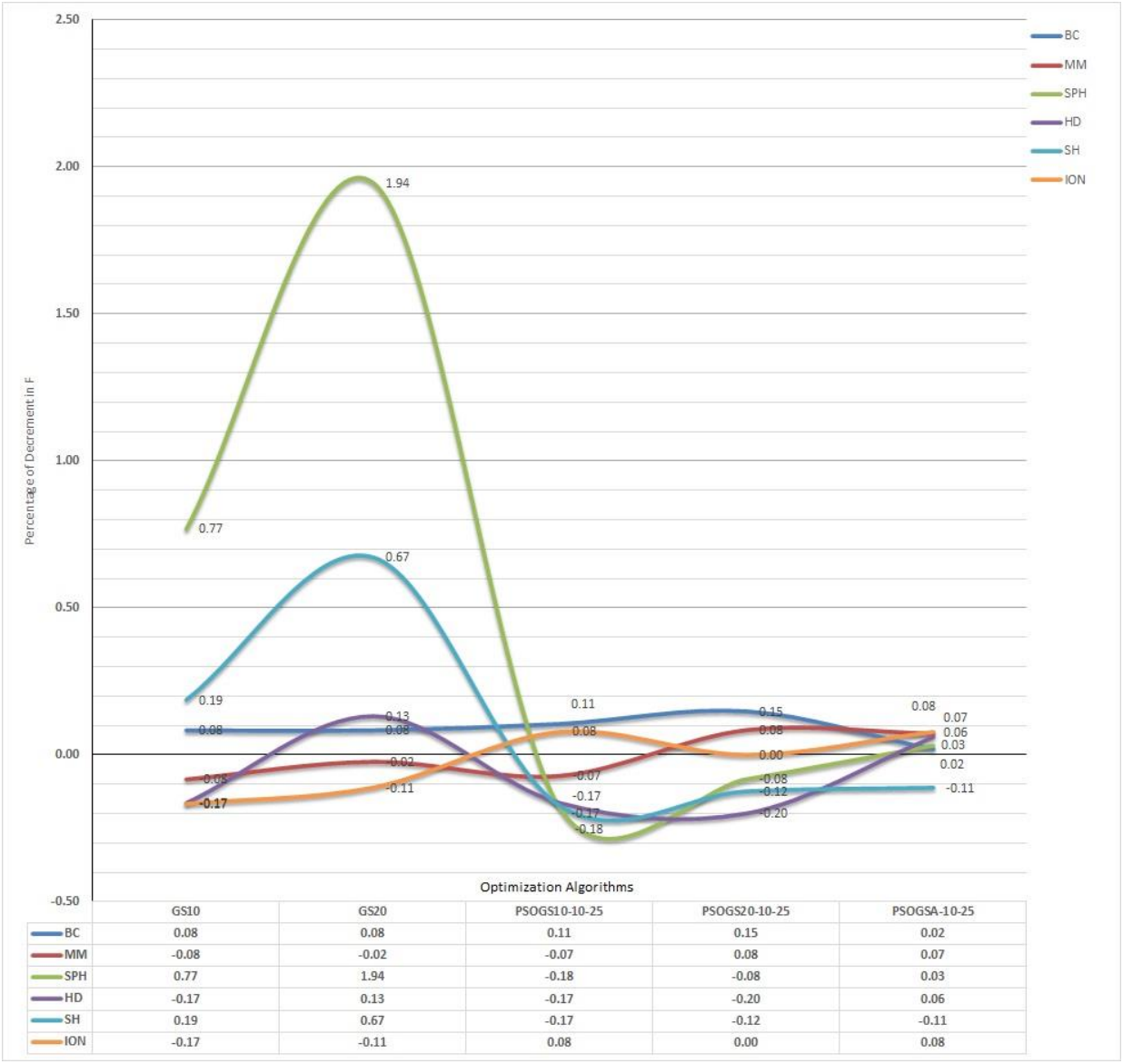
Experimental results of both SVC models and SVR models are in favor of the claim that PSOGS can be a good alternative to both PSO and GS. PSOGS was successful in reaching the aim of attaining the prediction accuracy of PSO and GS in less time and with less $Cost_{Eval}$.

When SVC results are compared with SVR results, it is clear that PSOGS's effect on SVM efficiency is more observable on regression models. This is mainly because SVR has three parameters to optimize whereas SVC has only two parameters. The higher number of hyperparameters results in more dimensions, a bigger search space.

As it can be observed from Figure 4.3, experimental results on prediction accuracies of created SVC models are very close. One-way ANOVA test results also showed that the differences between prediction accuracy results of different SVC models are statistically insignificant. Whereas, there are statistically significant differences between prediction accuracy results of SVR models. Whether the prediction accuracy differences are statistically significant or not, PSOGS proved to work faster than other algorithms.

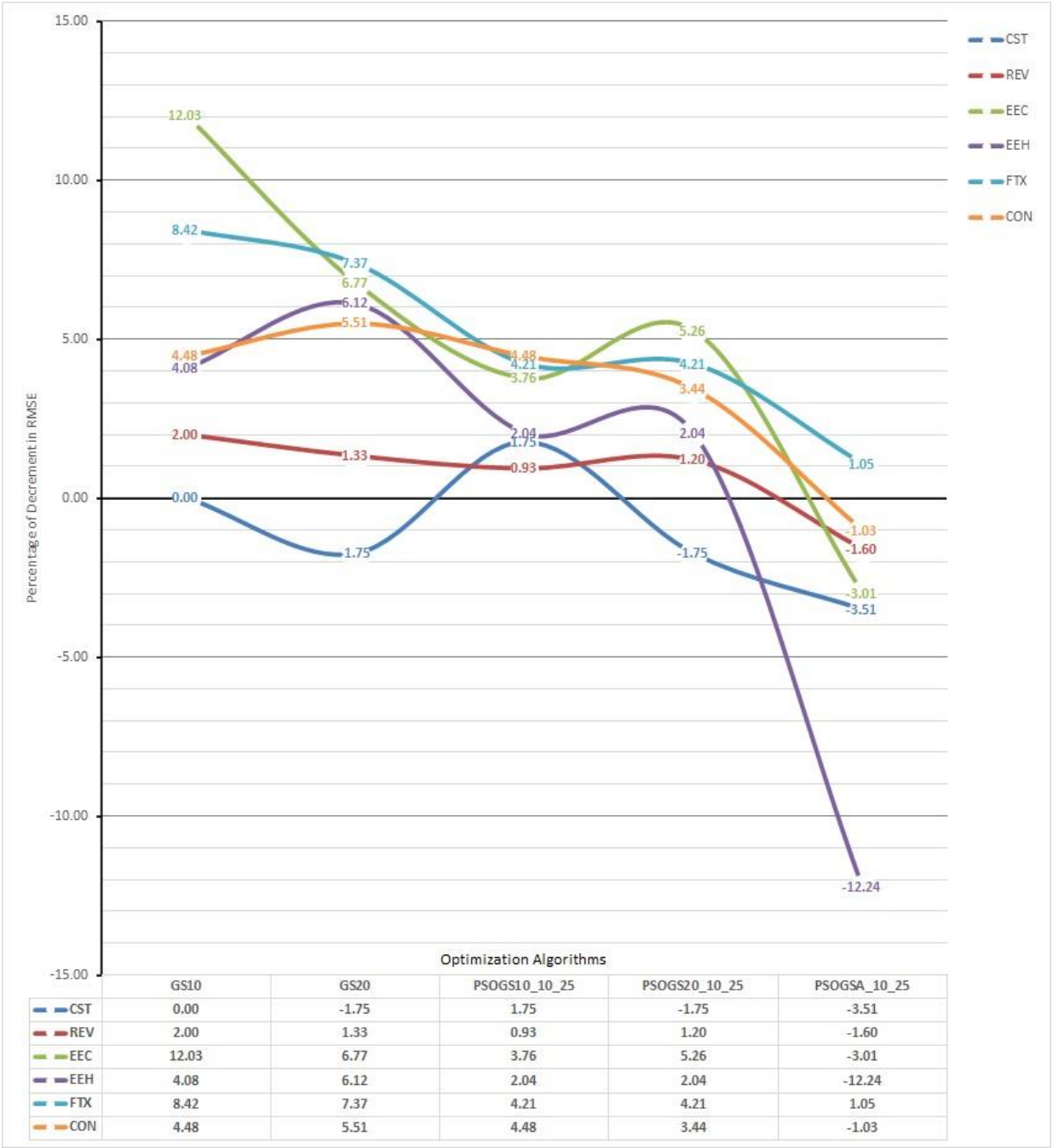
As discussed, PSOGS's effect can better be observed on SVR models. Figure 4.4 shows the comparison of RMSE results with respect to PSO-SVR models. In this chart, it can be observed that PSOGS10 increased PSO's effect on prediction accuracy by an average of 2.86% and up to 4.48%. PSOGS20, as well, yielded on average 2.86% and up to 5.26% better prediction accuracy regarding RMSE results.

Summing up all the results, PSOGS is a real candidate to be a new hybrid optimization technique that can be used to optimize the parameters of SVM.



	GS10	GS20	PSOGS10-10-25	PSOGS20-10-25	PSOGSA-10-25
BC	0.08	0.08	0.11	0.15	0.02
MM	-0.08	-0.02	-0.07	0.08	0.07
SPH	0.77	1.94	-0.18	-0.08	0.03
HD	-0.17	0.13	-0.17	-0.20	0.06
SH	0.19	0.67	-0.17	-0.12	-0.11
ION	-0.17	-0.11	0.08	0.00	0.08

Figure 4. 3 Percent decrement rates in F in models with respect to PSO-optimized SVC models



	GS10	GS20	PSOGS10_10_25	PSOGS20_10_25	PSOGSA_10_25
CST	0.00	-1.75	1.75	-1.75	-3.51
REV	2.00	1.33	0.93	1.20	-1.60
EEC	12.03	6.77	3.76	5.26	-3.01
EEH	4.08	6.12	2.04	2.04	-12.24
FTX	8.42	7.37	4.21	4.21	1.05
CON	4.48	5.51	4.48	3.44	-1.03

Figure 4. 4 Percent decrement rates in RMSE in models with respect to PSO-optimized SVR models

5. CONCLUSIONS

Hyperparameter optimization is a crucial process for increasing the prediction accuracy of the SVM-based models. In order to ensure that this process is handled properly, a convenient optimization algorithm must be utilized. In this study, a new hybrid optimization algorithm called PSOGS is proposed to fulfill this purpose. This new algorithm combines two widely used optimization techniques, GS and PSO.

In order to put forth the validity and stability of the proposed algorithm, six different benchmark datasets with classification characteristics and five different benchmark datasets with regression characteristics were used to determine the optimization performance of the PSOGS by comparison against its constituent algorithms (GS and PSO) and another PSO-based hybrid algorithm PSOGSA.

Preliminary works have been conducted to decide the kernel function for SVM to be used, as well as swarm size and the number of iterations of PSO-based algorithms.

Within the scope of this study, six models with different optimization algorithms and configurations were developed for each dataset to optimize the hyperparameters of the SVM. All developed models were compared with each other in terms of prediction performance, ExecTime, and $Cost_{Eval}$. These results were obtained by taking the average of all 30 runs and all folds' results of the 10-fold cross-validation process for each run.

The experimental results showed improvements of the proposed algorithm in reducing GS's excessive ExecTime and increasing PSO's prediction accuracy, especially on regression models. Regarding SVR models PSOGS10 and PSOGS20 evaluated the cost function 87.3% and 97.7% fewer times than GS10 and GS20 respectively. Compared to PSO; PSOGS10 and PSOGS20 evaluated the cost function 53.6% and 32% fewer times and yielded up to 4.48% and 5.26% better prediction accuracies respectively. As for SVC models, PSOGS10 evaluated the cost function 47.8% and 81% fewer times than GS10 and PSO; PSOGS20 made 82% and 73.76% fewer cost function evaluations than GS20 and PSO respectively.

The prediction performances of created models were also compared using One-way ANOVA test in order to determine if the differences between PSOGS-SVM models and other models are statistically significant. The tests showed that PSOGS's effect on results are significant for SVR models.

As a result, PSOGS has proved itself as a fast, stable, efficient, and robust algorithm that can safely be utilized for hyperparameter optimization of SVM.

REFERENCES

Açıkkar, M. (2020). Prediction Of Gross Calorific Value Of Coal From Proximate And Ultimate Analysis Variables Using Support Vector Machines With Feature Selection. *Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi*, *9*(2), 1129–1141. https://doi.org/10.28948/ngumuh.585596

Alam, M., Sultana, N., & Hossain, S. M. (2021). Bayesian optimization algorithm based support vector regression analysis for estimation of shear capacity of FRP reinforced concrete members. *Applied Soft Computing*, *105*, 107281. https://doi.org/10.1016/j.asoc.2021.107281

Al-Thanoon, N. A., Qasim, O. S., & Algamal, Z. Y. (2019). A new hybrid firefly algorithm and particle swarm optimization for tuning parameter estimation in penalized support vector machine with application in chemometrics. *Chemometrics and Intelligent Laboratory Systems*, *184*, 142–152. https://doi.org/10.1016/j.chemolab.2018.12.003

Anguita, D., Ridella, S., & Rivieccio, F. (n.d.). *K-Fold Generalization Capability Assessment for Support Vector Classifiers*.

Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011). Inertia weight strategies in particle swarm optimization. *Proceedings of the 2011 3rd World Congress on Nature and Biologically Inspired Computing, NaBIC 2011*, 633–640. https://doi.org/10.1109/NaBIC.2011.6089659

Ben-Hur, A., Ong, C. S., Sonnenburg, S., Schölkopf, B., & Rätsch, G. (2008). Support vector machines and kernels for computational biology. *PLoS Computational Biology*, *4*(10). https://doi.org/10.1371/journal.pcbi.1000173

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–152. https://doi.org/10.1145/130385.130401

Chang, Y. W., Hsieh, C. J., Chang, K. W., Ringgaard, M., & Lin, C. J. (2010). Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, *11*, 1471–1490.

Chen, B., Fu, X., Guo, X., Gu, C., Shao, C., & Qin, X. (2019). Zoning Elastic Modulus Inversion for High Arch Dams Based on the PSOGSA-SVM Method. *Advances in Civil Engineering*, *2019*. https://doi.org/10.1155/2019/7936513

Chih-Wei Hsu, Chih-Chung Chang, and C.-J. L. (2008). A Practical Guide to Support Vector Classification. *BJU International*, *101*(1), 1396–1400. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Clerc, M. (2004). Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem. In *New Optimization Techniques in Engineering* (pp. 219–239). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-39930-8_8

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

Demidova, L., & Klyueva, I. (2018). Data classification based on the hybrid versions of the particle swarm optimization algorithm. *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, 1–4. https://doi.org/10.1109/MECO.2018.8406069

Drucker, H., Surges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems*, *1*, 155–161.

Du, J., Liu, Y., Yu, Y., & Yan, W. (2017). *A Prediction of Precipitation Data Based on Support Vector Machine and Particle Swarm Optimization (PSO-SVM) Algorithms*. https://doi.org/10.3390/a10020057

Feurer, M., & Hutter, F. (2019). *Hyperparameter Optimization* (pp. 3–33). Springer, Cham. https://doi.org/10.1007/978-3-030-05318-5_1

Fu, W., Shao, K., Tan, J., & Wang, K. (2020). Fault Diagnosis for Rolling Bearings Based on Composite Multiscale Fine-Sorted Dispersion Entropy and SVM with Hybrid Mutation SCA-HHO Algorithm Optimization. *IEEE Access*, *8*, 13086–13104. https://doi.org/10.1109/ACCESS.2020.2966582

Hoang, T., Cho, M.-Y., Alam, M., & Vu, Q. (2017). A Novel Differential Particle Swarm Optimization for Parameter Selection of Support Vector Machines for Monitoring Metal-oxide Surge Arrester Conditions. *Swarm and Evolutionary Computation*, *38*. https://doi.org/10.1016/j.swevo.2017.07.006

Hongmei, K., Huibin, S. U. I., & Peng, Z. (2019). PV Prediction based on PSO-GS-SVM Hybrid Model. *Journal of Physics: Conference Series*, *1346*, 12028. https://doi.org/10.1088/1742-6596/1346/1/012028

Hu, G., Xu, Z., Wang, G., Zeng, B., Liu, Y., & Lei, Y. (2021). Forecasting Energy Consumption of Long-distance Oil Products Pipeline Based on Improved Fruit Fly Optimization Algorithm and Support Vector Regression. *Energy*, *224*, 120153. https://doi.org/10.1016/j.energy.2021.120153

Huang, C. L., & Dun, J. F. (2008). A distributed PSO-SVM hybrid system with feature selection and parameter optimization. *Applied Soft Computing Journal*, *8*(4), 1381–1391. https://doi.org/10.1016/j.asoc.2007.10.007

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization Proceedings., IEEE International Conference. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *11*(1), 111–117.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, *5*, 4104–4108 vol.5. https://doi.org/10.1109/ICSMC.1997.637339

Khajeh, A., Ghasemi, M. R., & Ghohani Arab, H. (2017). Hybrid Particle Swarm Optimization, Grid Search Method And Univariate Method To Optimally Design Steel Frame Structures. *International Journal of Optimization in Civil Engineering*, *7*, 171–189.

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer. https://doi.org/10.1007/978-1-4614-6849-3

Larsen, R. B., Jouffroy, J., & Lassen, B. (2017). On the premature convergence of particle swarm optimization. *2016 European Control Conference, ECC 2016*, 1922–1927. https://doi.org/10.1109/ECC.2016.7810572

Li, L., Liu, D., Ren, S., Zhou, H.-G., & Zhou, J. (2021). Prediction of Welding Deformation and Residual Stress of a Thin Plate by Improved Support Vector Regression. *Scanning*, *2021*, 8892128. https://doi.org/10.1155/2021/8892128

Li, S., Fang, H., & Liu, X. (2018). Parameter optimization of support vector regression based on sine cosine algorithm. *Expert Systems with Applications*, *91*, 63–77. https://doi.org/10.1016/J.ESWA.2017.08.038

Liu, R., Liu, E., Yang, J., Li, M., & Wang, F. (2006). Optimizing the Hyper-parameters for SVM by Combining Evolution Strategies with a Grid Search. In *Intelligent Control and Automation* (pp. 712–721). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-37256-1_87

Mirjalili, S. (2018). *Hybrid Particle Swarm Optimization and Gravitational Search Algorithm (PSOGSA)*.

Mirjalili, S., & Hashim, S. Z. M. (2010). A new hybrid PSOGSA algorithm for function optimization. *Proceedings of ICCIA 2010 - 2010 International Conference on Computer and Information Application*, *1*, 374–377. https://doi.org/10.1109/ICCIA.2010.6141614

Mirjalili, S., Mohd Hashim, S. Z., & Moradian Sardroudi, H. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*, *218*(22), 11125–11137. https://doi.org/https://doi.org/10.1016/j.amc.2012.04.069

Montgomery, D. C., & Runger, G. C. (2003). *Applied Statistics and Probability for Engineers*. John Wiley and Sons.

Pan, M., Li, C., Gao, R., Huang, Y., You, H., Gu, T., & Qin, F. (2020). Photovoltaic power forecasting based on a support vector machine with improved ant colony optimization. *Journal of Cleaner Production*, *277*, 123948. https://doi.org/10.1016/j.jclepro.2020.123948

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Section 16.5. Support Vector Machines. In *Numerical Recipes: The Art of Scientific Computing* (3rd ed., p. 1262). New York: Cambridge University Press. https://doi.org/10.1137/1031025

Raj, S., & Ray, K. C. (2017). ECG Signal Analysis Using DCT-Based DOST and PSO Optimized SVM. *IEEE Transactions on Instrumentation and Measurement*, *66*(3), 470–478. https://doi.org/10.1109/TIM.2016.2642758

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. *Information Sciences*, *179*(13), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Sabri, N. M., Puteh, M., & Mahmood, M. R. (2013). A review of gravitational search algorithm. *International Journal of Advances in Soft Computing and Its Applications*, *5*(3).

Sahu, B., Panigrahi, A., Mohanty, S., & Panigrahi, S. S. (2020). A hybrid Cancer Classification Based on SVM Optimized by PSO and Reverse Firefly Algorithm. *International Journal of Control and Automation*, *13*(4), 506–517.

Sarafrazi, S., & Nezamabadi-Pour, H. (2013). Facing the classification of binary problems with a GSA-SVM hybrid system. *Mathematical and Computer Modelling*, *57*(1–2), 270–278. https://doi.org/10.1016/j.mcm.2011.06.048

Siddique, N., & Adelib, H. (2016). Applications of gravitational search algorithm in engineering. *Journal of Civil Engineering and Management*, *22*, 981–990. https://doi.org/10.3846/13923730.2016.1232306

Tao, Z., Huiling, L., Wenwen, W., & Xia, Y. (2019). GA-SVM based feature selection and parameter optimization in hospitalization expense modeling. *Applied Soft Computing Journal*, *75*, 323–332. https://doi.org/10.1016/j.asoc.2018.11.001

The MathWorks. (2018). *MATLAB Statistics and Machine Learning Toolbox User's Guide* (No. 2018b). The MathWorks, Inc.

UCI Machine Learning Repository: Data Sets. (n.d.). Retrieved April 1, 2021, from https://archive.ics.uci.edu/ml/datasets.php

Vapnik, V. (1995). *The Nature of Statistical Learning Theory* (1st ed.). Springer. https://doi.org/https://doi.org/10.1007/978-1-4757-2440-0

Wei, J., Zhang, R., Yu, Z., Hu, R., Tang, J., Gui, C., & Yuan, Y. (2017). A BPSO-SVM algorithm based on memory renewal and enhanced mutation mechanisms for feature selection. *Applied Soft Computing Journal*, *58*, 176–192. https://doi.org/10.1016/j.asoc.2017.04.061

Wu, C. H., Tzeng, G. H., & Lin, R. H. (2009). A Novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert Systems with Applications*, *36*(3 PART 1), 4725–4735. https://doi.org/10.1016/j.eswa.2008.06.046

Xiao, T., Ren, D., Lei, S., Zhang, J., & Liu, X. (2015). Based on grid-search and PSO parameter optimization for Support Vector Machine. *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, *2015-March*(March), 1529–1533. https://doi.org/10.1109/WCICA.2014.7052946

Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., & Nakanishi, Y. (2000). A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems*, *15*(4), 1232–1239. https://doi.org/10.1109/59.898095

Zhang, J., Song, W., Jiang, B., & Li, M. (2018). Measurement of lumber moisture content based on PCA and GS-SVM. *Journal of Forestry Research*, *29*(2), 557–564. https://doi.org/10.1007/s11676-017-0448-x

Zhu, S., Lian, X., Wei, L., Che, J., Shen, X., Yang, L., Qiu, X., Liu, X., Gao, W., Ren, X., & Li, J. (2018). PM2.5 forecasting using SVR with PSOGSA algorithm based on CEEMD, GRNN and GCA considering meteorological factors. *Atmospheric Environment*, *183*(April), 20–32. https://doi.org/10.1016/j.atmosenv.2018.04.004