TELKOMNIKA, Vol.14, No.1, March 2016, pp. 245~253
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v14i1.2812

Hybridizing PSO with SA for Optimizing SVR Applied to Software Effort Estimation

Dinda Novitasari, Imam Cholissodin, Wayan Firdaus Mahmudy
Dept. of Informatics/ Computer Science, Brawijaya University, Malang 65145
e-mail: id.dindanovitasari@gmail.com, imamcs@ub.ac.id, wayanfm@ub.ac.id

Abstract

This study investigates Particle Swarm Optimization (PSO) hybridization with Simulated Annealing (SA) to optimize Support Vector Machine (SVR). The optimized SVR is used for software effort estimation. The optimization of SVR consists of two sub-problems that must be solved simultaneously; the first is input feature selection that influences method accuracy and computing time. The next sub-problem is finding optimal SVR parameter that each parameter gives significant impact to method performance. To deal with a huge number of candidate solutions of the problems, a powerful approach is required. The proposed approach takes advantages of good solution quality from PSO and SA. We introduce SA based acceptance rule to accept new position in PSO. The SA parameter selection is introduced to improve the quality as stochastic algorithm is sensitive to its parameter. The comparative works have been between PSO in quality of solution and computing time. According to the results, the proposed model outperforms PSO SVR in quality of solution.

Keywords: *Particle swarm optimization, simulated annealing, support vector regression, feature selection, parameter optimization*

1. Introduction

The most important part of software project is software effort estimation. It determines how many resources that project needed and must be done accurately. If we have big error rate in estimation, it will lead into big loss such as unpredictable delay time and unexpected budget. To prevent many losses in the future, some approaches are developed to estimate software effort. One of them is machine learning. Support vector machine is machine learning algorithm introduced by Vapnik to solve classification problem. Due to solve real world problems, SVM was developed to solve regression and time series prediction called SVM based regression (SVR). In order to solve nonlinear regression problem, SVR mapped data to high dimensional feature space using kernel function. This kernel must satisfy Mercer condition [1] and one of kernels is radial basis function (RBF).

In machine learning, feature selection introduced as a process of selecting a subset feature for use in model construction. This process is needed for SVR since it can simplify computing process and reducing computing time, especially when computing in high dimensional space. Besides that, proper parameter settings can influence SVR accuracy. SVR-RBF has parameters influenced its performance i.e. error penalty, insensitive loss, and radial basis [2]. Those mentioned above are crucial in SVR-RBF because feature selection influences SVR parameter and vice versa [3]. In the past research, Oliviera investigated the use of SVR in order to do software effort estimation [4]. It gives promising result but cannot guarantee give good result since using predefined number of features and parameter means cannot discover other options that can lead into higher accuracy rate. Numerous candidate of solution can be generated in order to have great number of subset feature combination and vary range of parameter, if we use enumeration. However, it does not utilize a fitness function, and is thus unguided, often failing to find good solution. Due to the complexity of the problem, a powerful approach is required to get a good solution.

Some stochastic optimization methods become alternative to select subset feature and optimize parameter. It generates candidate solutions, involves objective function to evaluate the quality of solution so solution searching could be lead into a good solution. Braga et al proposed genetic algorithm (GA) to optimize SVR in software effort estimation [5]. Our previous research

Received October 7, 2015; Revised December 18, 2015; Accepted January 7, 2016

proposed particle swarm optimization (PSO) to optimize SVR in the same problem domain [6]. Basically, PSO is inspired by flocking bird motion employed parallel search techniques, exploitation and exploration. However, PSO has disadvantage, trapped in local minimum because particles move in high velocity and gain premature convergence [7]. On the other hand, simulated annealing inspired by process of annealing in metallurgy, is good in finding local optimum [8]. Therefore, this study investigates hybridization PSO with SA in order to enhance searching capacity. This proposed model is used to optimize SVR parameter and select subset feature applied to software effort estimation.

Several researches investigated SVR optimization have been done and gained promising result. Braga et.al [5] investigated GA application to select subset feature and SVR parameter applied to software effort estimation. They used binary coded chromosome as solution representation for subset feature and SVR parameter. Their research reported success to improve SVR performance. Our previous research [6] investigated PSO application to select subset feature and SVR parameter applied to software effort estimation. We used continuous value type to optimize SVR parameter and discrete value type to select subset feature. Another effort has been done by Adhani [9], who optimized SVR with GAPSO. They are reported success to build SVR model for predicting rainfall in dry season. However, other researches have been done to investigate on how to improve PSO performance. Xue [10] introduced QoS-based hybrid particle swarm optimization (GHPSO) to schedule workflow in cloud computing. It gained better performance than PSO. A research conducted by Shieh et.al [11] in modification PSO with SA. Their research proposed SAPSO to enhance searching capacity algorithm. They reported proposed model could have higher efficiency, better quality and faster convergence than PSO. Therefore, based on past researches, this study proposed SAPSO SVR applied to software effort estimation. By using SAPSO, can be generated more optimize SVR parameter, better selected feature, and low cost value.

2. Support Vector Regression

Given training data $\{x_i,y_i\}$, $i = 1,...,l$; $x_i\in R^d$; $y_i\in R^d$ where x_i, y_i is input (vector) and output (scalar value as target). Other forms of alternative for bias to calculation $f(x)$ is can be build solution like bias as follows [1]:

$$\begin{aligned} b &= y_k - \varepsilon - w^T\Phi(x_k) \\ &= y_k - \varepsilon - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\langle x_i, x_k\rangle \\ &= y_k - \varepsilon - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)k(x_i, x_k) \end{aligned} \quad (1)$$

x_i is support vector where $|\alpha_i - \alpha_i^*|$ isn't zero. Equation $f(x)$ can be written as follows:

$$f(x)=\sum_{i=1}^{l}(\alpha_i - \alpha_i^*)(x \bullet x_i)+b \quad (2)$$

Lambda (λ) is scalar constant, with it's an augmented factor defined as follows [12]:

$$f(x) = \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)(K(x_i,x)+\lambda^2). \quad (3)$$

2.1. Sequential Algorithm for SVR

Vijayakumar has made tactical steps through the process of iteration to obtain the solution of optimization problems of any nature by way of a trade-off on the values of the weights x_i, or called α_i to make the results of the regression becomes closer to actual value. The step by step as follows [12]:

1. Initialize $\alpha_i = 0, \alpha_i^* = 0$. Compute $[R]_{ij} = K(x_i, x_j) + \lambda^2$ (4)
   for $i,j = 1,\ldots,n$
2. For each training point (x_i), $i=1$ to n, compute:

$$E_i = y_i - \sum_{j=1}^{n} (\alpha_i^* - \alpha_i) R_{ij} \tag{5}$$

$$\delta\alpha_i^* = \min\{\max[\ \gamma(E_i - \varepsilon), -\alpha_i^*], C - \alpha_i^*\}. \tag{6}$$

$$\delta\alpha_i = \min\{\max[\ \gamma(-E_i - \varepsilon), -\alpha_i], C - \alpha_i\}. \tag{7}$$

$$\alpha_i^* = \alpha_i^* + \delta\alpha_i^*. \tag{8}$$

$$\alpha_i = \alpha_i + \delta\alpha_i. \tag{9}$$

3. Repeat step 2 until meet stop condition.
   Where learning rate γ is computed from:

$$\frac{\text{learning rate constant}}{\max\left(\text{diagonal of kernel matrice}\right)} \tag{10}$$

3. Particle Swarm Optimization

Particle swarm optimization was introduced by Kennedy and Ebenhart [13], as a nature inspired algorithm. Particles are defined as solution for problem. Developing by Shi and Ebenhart [14], PSO is added by inertia weight to improves performance. Each particle has position and velocity, and updates that in every iterating. The velocity is updated by:

$$v_{ij}(t+1)=wv_{ij}(t)+c_1r_{1j}(t)[y_{ij}(t)-x_{ij}(t)]+c_2r_{2j}(t)[\hat{y}(t)-x_{ij}(t)] \tag{15}$$

And its position updated by:

$$x_i(t+1)=x_i(t)+v_i(t+1) \tag{16}$$

Where $v_{ij}(t)$ is velocity of particle i in dimension j=1,...n at time t, $x_{ij}(t)$ is position of particle i in dimension j at time t, c_1 and c_2 are acceleration constants used to scale contribution of the cognitive and social components, r_{1j} and r_{2j} are random values in the range [0,1]. W is inertia weight obtained by:

$$\omega = (\omega_{\max} - \omega_{\min})\frac{iter_{\max} - iter}{iter_{\max}} + \omega_{\max} \tag{17}$$

Where w_{max} and w_{min} are maximal and minimum inertia weight, $iter_{max}$ is maximum number of iterations, $iter$ is current iteration number. Y_i is personal best position of particle i obtained by:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f\left(x_i(t+1)\right) \geq f\left(y_i(t)\right) \\ x_i(t+1) & \text{if } f\left(x_i(t+1)\right) < f\left(y_i(t)\right) \end{cases} \tag{18}$$

And ŷ represents global best position of particle i obtained by:

$$\hat{Y}(t) \in \{y_0(t),\ldots,y_{ns}(t)\} | f(\hat{y}(t)) = \min\{f(y_0(t)),\ldots, f(y_{ns}(t))\} \tag{19}$$

3.1. Binary PSO

Some optimization problems are set in a space featuring discrete. Kennedy and Ebenhart [15] proposed binary PSO in which each element of particle's position vector can take on the binary value 0 or 1. New velocity of particle is normalized by sigmoid function:

$$v'_{ij}(t) = sig(v_{ij}(t)) = \frac{1}{1+e^{-v_{ij}(t)}} \quad (20)$$

Where *v_{ij}*(*t*) is obtained from Equation (15). Using Equation (16), the position update changes to:

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r_{3j}(t) < sig(v_{ij}(t+1)) \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

Where $r_{3j}(t) \sim U(0,1)$.

4. Hybridizing PSO with SA

A searching algorithm has two important components, exploration and exploitation. Exploration means algorithm search in different region of searching space to find global optimum. Exploitation means algorithm localize promising area to find best solution in that area. A good searching algorithm must able to balance its exploration and exploitation, able to search entire space and jump out of local optimum solution. By that means, it must able to improve probability and ability of finding global optimum solution.

Initial random position PSO can lead into premature convergence, entire particle move toward local optimum solution and cause weakening exploration because particle can't jump out of area. It is characteristic and weakness of PSO. Meanwhile, SA with low variation of temperature parameter and searching solution reach equilibrium condition, able to guarantee to find global optimum. It is enhanced by metropolis process, ability to jump out from local optimum. However, it costs high computing time.

Based on PSO and SA characteristic above, this study hybridizes PSO with SA, combines PSO parallel process and movement mechanism and SA searching procedure. By combining that, this proposed model able to find good solution in local and global optimum with low computing time.

4.1. Simulated Annealing Algorithm

Simulated annealing is an optimization process based on the annealing process; the cooling process of a liquid or solid and the analysis of the behavior of substances as they cool. This algorithm is introduced by Kirkpatrick [16] and inspired by Metropolis work about energy distribution [17]. In SA algorithm, metropolis process does searching solution. During the process, disturbance mechanism (metropolis acceptance rule) determines quality of solution by searching around existing solution and comparing neighbor solution and current solution. This procedure affects SA ability to jump out from local optimum solution. If neighbor solution is better than current solution then neighboring solution is accepted as the new current solution. If neighbor solution is worse than current solution then SA will use a probability to determine whether accept this neighboring solution as new current solution or not, or regenerate for a new neighboring solution. The probability mechanism for metropolis acceptance rule is defined as follows:

$$P = \begin{cases} 1 & \text{if } f(x_j) < f(x_i) \\ e^{-\frac{f(x_j)-f(x_i)}{c_b T}} & \text{otherwise} \end{cases} \quad (22)$$

Where *P* is probability, $f(x_j)$ is neighbor solution, $f(x_i)$ is current solution, $c_b>0$ is Boltzmann constant and *T* is temperature of the system. T is derived from:

$$T_{k+1} = \alpha \times T_0 \quad (23)$$

Where α is cooling rate, T_{k+1} is temperature at time *k*, and T_0 is initial temperature. While SA is quite simple, it has been successfully implemented to solve various combinatorial problem [18].

5. SAPSO SVR Model

5.1. Particle Representation

In this study, SVR RBF is defined by the parameter *C* – complexity parameter, *ε* - the extent to which deviations are tolerated, *λ* - augmenting factor, *σ* – width of RBF kernel, *cLR* – learning rate constant. The particle is comprised of six parts: *C, ε, λ, σ, cLR* (continuous-valued) and features mask (discrete-valued). Table 1 shows the representation of particle *i* with dimension n_f+5 where n_f is the number of features. The feature mask is Boolean that "1" indicates the feature is selected and "0" indicates feature is not selected.

Table 1. Particle *i* is composed of six parts: c, *ε, λ, ε, cLR* and feature mask

Continuous-valued					Discrete-valued
C	***ε***	***λ***	***σ***	***cLR***	***Feature mask***
$X_{i,1}$	$X_{i,2}$	$X_{i,3}$	$X_{i,4}$	$X_{i,5}$	$X_{i,6}, X_{i,7},\ldots,X_{i,nf}$

5.2. Objective Function

Objective function is used to measure how optimal the generated solution. There are two types of objective function: fitness and cost. The higher fitness value means better solution. The lower cost value means better solution. In this study, cost typed is used as objective function because the purpose of this algorithm is to minimize error. Accuracy of prediction and number of selected features are criteria used to design cost function. Thus, the particle with high accuracy of prediction and small number of features produces a low prediction error. The prediction error has two predefined weights: W_A for accuracy of prediction (95%) and W_F for the selected feature (5%) [19].

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \tag{24}$$

$$error = \left(w_A \times MAPE\right) + \left(w_F \times \frac{\sum_{j=1}^{n_F} f_j}{n_F}\right) \tag{25}$$

Where *n* is number of data, A_i is actual value and F_i is prediction value for data, f_j is value of feature mask where "1" represents that feature *j* is selected and "0" represents that feature *j* is not selected and n_f is total number of features.

5.3. SAPSO SVR Algorithms

The SAPSO SVR algorithm is started by initialization of particle. Then, calculate cost and determine personal best position (*pBest*) and global best position (*gBest*). After that, update velocity and position. Usually, PSO automatically accept new position, however SAPSO SVR introduces SA metropolis acceptance rule in this step. This rule determines whether to accept new position or regenerate another candidate position based on cost function difference between new and old positions. This enables PSO to jump out from local optimum, improve quality of solution, and increase rate of convergence. Simulated annealing explores solution towards direction of *pBest* and *gBest*. The acceptance rule accepts or rejects new solution based on current temperature parameter and cost value difference. If candidate solution unable pass criteria then a new position generated using PSO and repeated until metropolis acceptance rule accept new position or upper bound of disturbance is reached. By this way, the model explores solution, improve exploration and spend low computing time since using PSO parallel processing.

Based on Figure 1, the whole procedure of SAPSO SVR is described as follows:

1. Normalizing data using

$$x_n = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{26}$$

Where *x* is the original data from dataset, x_{min} and x_{max} is the minimum and maximum value of original data, and x_n is normalized value.

2. Dividing data into k to determine training and testing data.
3. Initializing a population of particle randomly s_1^0, $v_1^0 = 0$ i=1,2,…n, *iter*=0.
4. Calculating cost of s_1^0 by averaging error over *k* SVR training.
5. Updating *pBest* and *gBest* of each particle.
6. Updating inertia weight.
7. Repeat these steps until meet stopping condition

a) Updating velocity v_1^{iter+1} and position of each particle.

b) Calculating cost of s_1^{iter+1} by averaging error over *k* SVR training.

c) Evaluate Δcost= $\text{cost}(s_1^{iter+1}) - \text{cost}(s_1^{iter})$ and generate random number R [0,1]. If *Δcost*≤0, then accept new position with probability ONE. Otherwise, s_1^{iter+1} is accepted based on following criterion: $\text{probability} = e^{\frac{-\Delta \cos t}{temp}}$ ≥R. Proceed to next step if all new positions are accepted or repeat step 7.1 until 7.3 for those particles failed to be accepted. Too many failures (i.e. 100 in our study) for same particle will force the last position will be accepted.

d) Updating *pBest* and *gBest* of each particle.

e) Updating inertia weight and temperature, set *iter*=*iter*+1.

8. If stopping criteria is satisfied, and then end iteration. If not, repeat step 7. In this study, stopping criteria is a given number of iterations.
9. Output the best solution *gBest* and its cost value.

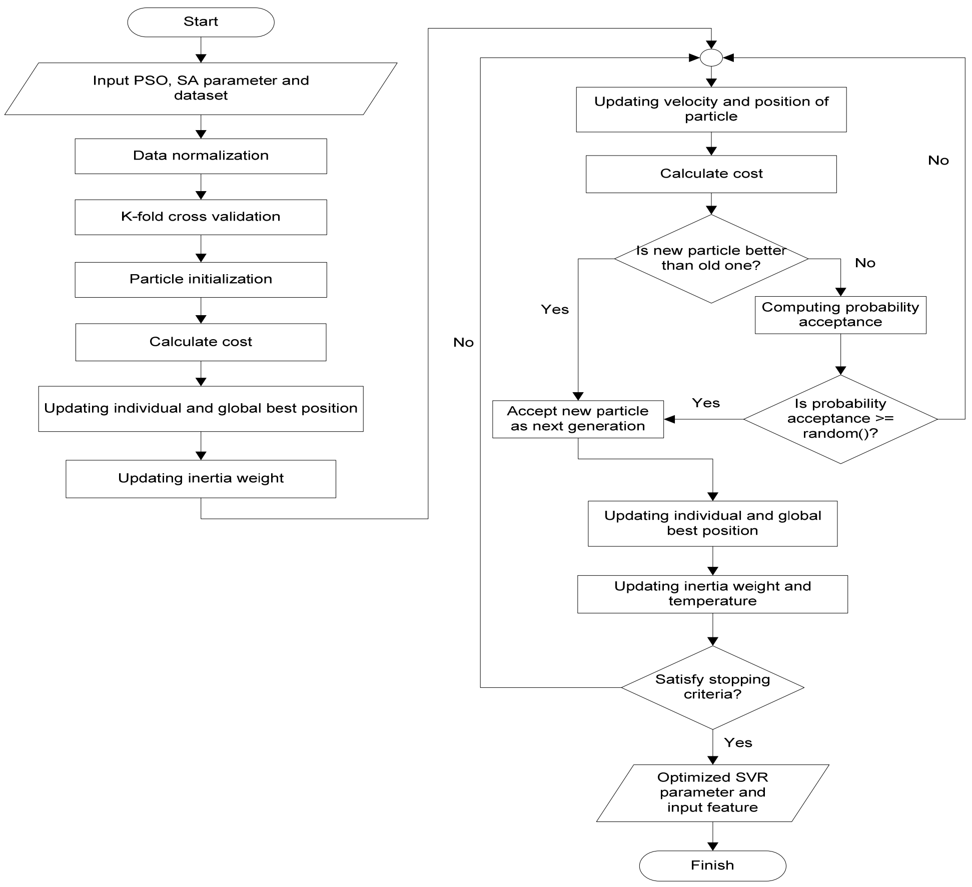
Start
Input PSO, SA parameter and dataset
Data normalization
K-fold cross validation
Particle initialization
Calculate cost
Updating individual and global best position
Updating inertia weight
Updating velocity and position of particle
Calculate cost
Is new particle better than old one?
Yes
No
Computing probability acceptance
Is probability acceptance >= random()?
Yes
No
Accept new particle as next generation
Updating individual and global best position
Updating inertia weight and temperature
Satisfy stopping criteria?
No
Yes
Optimized SVR parameter and input feature
Finish

Figure. 1 Flowchart of SAPSO SVR algorithm

6. Application SAPSO SVR in Software Effort Estimation

6.1. Simulation Settings

This study simulates 2 algorithms: PSO SVR and SAPSO SVR programmed using C#. For SAPSO SVR simulation, we use the same parameter and dataset that is obtained from [6] that conducted PSO SVR simulation. For software effort estimation, the inputs of SVR are Desharnais dataset [20]. The Desharnais dataset consists of 81 software projects described by 11 variables, 9 independent variables and 2 dependent variables. For the simulation, we decide to use 77 projects due to incomplete provided features and 7 independent variables (*TeamExp*, *ManagerExp*, *Transactions*, *Entities*, *PointsAdjust*, *Envergure*, and *PointsNonAdjust*) and 1 dependent variable (*Effort*). The PSO parameters were set as in Table 2.. Firstly, we run test to determine best parameter for SA (T_0 and α) then simulations is performed and compared to other algorithms.

Table 2. PSO parameter settings

Number of fold	10
Population of particles	20
Number of iterations	40
Inertia weight(w_{max}, w_{min})	(0,9, 0,4)
Acceleration coefficient(c_1, c_2)	(2, 2)
Parameter searching space	C (0,1-1500), ε (0,001-0,009), σ (0,1-4), λ(0,01-3), *cLR* (0,01-1,75)

6.2. Best Parameter

In stochastic algorithms, parameters have effect to quality of generated solution. In SA, initial temperature and cooling rate influence its performance. By observing parameters, we choose best parameter has lowest cost in each simulation.

Table 3 showed simulation to choose best initial temperature (T_0). This simulation conducted by increasing temperature by 10% from 50 up to 90 in each simulation and use cooling rate at 0,5. If T_0 is too low then algorithm has possibility to not explore, makes converge at local optimum. If T_0 is too high, then it can increase computing time. This table showed that T_0 at 90 give lowest cost.

Table 4 showed simulation to choose best cooling rate (α). This simulation conducted by increasing temperature by 10% from 0,5 up to 0,9 in each simulation If α is too low then algorithm has possibility to fail into local optimum solution, repeats calculation, and increase computing time. If α is too high, then it increase computing time. This table showed that α at 0,9 give lowest cost.

Table 3. Parameter setting for initial temperature

i-th simulation	T_0				
	50	60	70	80	90
1	0,6084	0,8336	0,5801	0,6096	**0,5690**
2	0,5706	0,5760	0,7265	0,5975	**0,5734**
3	0,5992	0,5898	0,5886	0,5848	**0,5682**
4	0,5610	0,6177	0,6223	0,5875	**0,5935**
5	0,7475	0,7579	0,6161	0,5795	**0,5761**
Average Cost	0,6174	0,6750	0,6267	0,5918	**0,5760**

Table 4. Parameter setting for cooling rate

i-th simulation	α				
	0,5	0,6	0,7	0,8	**0,9**
1	0,5690	0,5861	0,5776	0,5769	**0,5659**
2	0,5734	0,5957	0,5994	0,6163	**0,5563**
3	0,5682	0,5898	0,6182	0,6003	**0,5748**
4	0,5935	0,5765	0,6007	0,5753	**0,5575**
5	0,5761	0,6013	0,6026	0,5778	**0,5628**
Average Cost	0,5760	0,5899	0,5997	0,5893	**0,5635**

6.3. Comparison Works

By using best parameter, we compare SAPSO SVR and PSO SVR performance. Figure 2 showed comparison of convergence between PSO SVR and SAPSO. It showed that SAPSO have faster convergence than PSO SVR. In Table 5, we can see that SAPSO has higher computing time faster than PSO SVR but on the other hand, SAPSO also can have faster convergence and lower cost than PSO SVR. The error difference of error is big, but the high computing time can be compromised. The computing time is high because the model must repeat searching candidate position if they fail meet the acceptance rule criteria and this is different with PSO automatically accept candidate position.

Table 5. Simulation result

Model	Time (ms)	Optimal (*C, ε, σ, cLR, λ*)	Selected features	Error
PSO-SVR	50238	393,04, 0,09, 0,1, 0,01, 1,6669	2 (*PointsAdjust* and *PointsNonAdjust*)	0,5981
SAPSO-SVR	83756	1500, 0,0401, 0,3996, 0,01, 0,01	2 (*Envergure,* and *PointsNonAdjust*)	0,5575

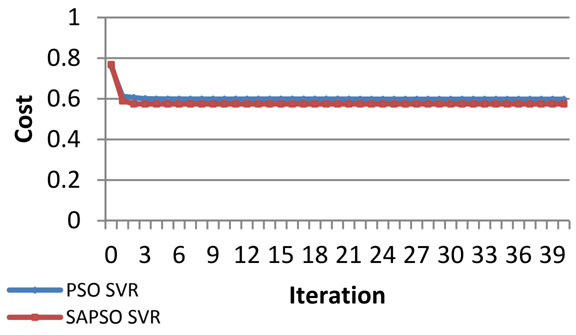

Figure 2. Convergence curve PSO SVR and SAPSO

7. Conclusion

This study investigated the use of SAPSO for optimal feature subset selection and SVR parameters optimization in the problem of software effort estimation. In our simulations, we used Desharnais dataset. We compared our results to PSO-SVR. From the experiment results, using SA can improve performance of PSO. The proposed model can combine the advantage of both algorithms and gain lower cost than PSO.

References

[1] Smola AJ, Scholkopf B. A Tutorial on Support Vector Regression. *Statistics and Computing*. 2004; 14(3): 199-222.
[2] Wang W, Xu Z, Lu W, Zhang X. Determination of The Spread Parameter in the Gaussian Kernel For Classification and Regression. *Neurocomputing*. 2003; 55: 643–663.
[3] Frohlich H, Chapelle O, Scholkopf B. *Feature Selection for Support Vector Machines by Means of Genetic Algorithm*. In Proceedings 15th IEEE International Conference on Tools with Artificial Intelligence. 2003: 142-148.
[4] Oliveira ALI. Estimation of software project effort with support vector regression. *Neurocomputing*. 2006; 69(13-15): 1749–53.
[5] Braga PL, Oliveira ALI, Meira SRL. *A GA-based Feature Selection and Parameters Optimization for Support Vector Regression Applied to Software Effort Estimation*. In Proceedings of the 2008 ACM Symposium on Applied Computing. Fortaleza, Ceará, Brazil, ACM. 2008: 1788–1792.
[6] Novitasari D, Cholissodin I, Mahmudy WF. Optimizing support vector regression using particle swarm optimization for software effort estimation. *Submitted to: IAENG International Journal of Computer Science*. 2015
[7] Shi Y, Eberhart RC. *Empirical study of particle swarm optimization*. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99. 1999: 1945–1950.
[8] Locatelli M. Convergence properties of simulated annealing for continuous global optimization. *Journal of Applied Probability*. 1996; 33(4): 1127–1140.
[9] Adhani G, Buono A, Faqih A. Optimization of Support Vector Regression using Genetic Algorithm and

Particle Swarm Optimization for Rainfall Prediction in Dry Season. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(11): 7912–7919.

[10] Xue S, Wu W. Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(7): 1560–1566.

[11] Shieh H-L, Kuo C-C, Chiang C-M. Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification. *Applied Mathematics and Computation*. 2011; 218(8): 4365–4383.

[12] Vijayakumar S, Wu S. *Sequential Support Vector Classifiers and Regression*. In Proceedings of International Conference on Soft Computing (SOCO '99). 1999: 610–619.

[13] Kennedy J, Eberhart R. *Particle Swarm Optimization*. In Proceedings of IEEE International Conference on Neural Networks. Piscataway, NJ. 1995: 1942–1948.

[14] Shi Y, Eberhart R. *A Modified Particle Swarm Optimizer*. In 1998 IEEE International Conference on Evolutionary Computation Proceedings IEEE World Congress on Computational Intelligence. 1998: 69–73.

[15] Kennedy J, Eberhart RC. *A discrete binary version of the particle swarm algorithm*. In Proceedings of the World Miulticonference on Systemics, Cybernetics and Informatics. Piscataway, NJ. 1997: 4104–4109.

[16] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. *Science*. 1983; 220(4598): 671–680.

[17] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*. 1953 ;1087(21).

[18] Mahmudy WF. Improved simulated annealing for optimization of vehicle routing problem with time windows (VRPTW). *Kursor*. 2014; 7(3): 109–116.

[19] Guo Y. *An Integrated PSO for Parameter Determination and Feature Selection of SVR and Its Application in STLF*. In Proceedings of the Eighth International Conference on Machine Learning and Cybernetics. Baoding. 2009: 12–15.

[20] Sayyad Shirabad J, Menzies TJ. The PROMISE Repository of Software Engineering Databases [Internet]. School of Information Technology and Engineering, University of Ottawa, Canada. 2005.