CI&CEQ

Available on line at
Association of the Chemical Engineers AChE
www.ache.org.rs/CICEQ

Chemical Industry & Chemical Engineering Quarterly 15 (3) 175–187 (2009) CI&CEQ

S.K. LAHIRI
NADEEM M. KHALFE

National institute of technology,
Durgapur, India

SCIENTIFIC PAPER

UDC 519:66:665.6

DOI: 10.2298/CICEQ0903175L

SOFT SENSOR DEVELOPMENT AND OPTIMIZATION OF THE COMMERCIAL PETROCHEMICAL PLANT INTEGRATING SUPPORT VECTOR REGRESSION AND GENETIC ALGORITHM

Soft sensors have been widely used in the industrial process control to improve the quality of the product and assure safety in the production. The core of a soft sensor is to construct a soft sensing model. This paper introduces support vector regression (SVR), a new powerful machine learning method based on a statistical learning theory (SLT) into soft sensor modeling and proposes a new soft sensing modeling method based on SVR. This paper presents an artificial intelligence based hybrid soft sensormodeling and optimazation strategies, namely support vector regression - genetic algorithm (SVR-GA) for modeling and optimization of mono ethylene glycol (MEG) quality variable in a commercial glycol plant. In the SVR-GA approach, a support vector regression model is constructed for correlating the process data comprising values of operating and performance variables. Next, model inputs describing the process operating variables are optimized using genetic algorithm with a view to maximize the process performance. The SVR-GA is a new strategy for soft sensor modeling and optimization. The major advantage of the strategies is that modeling and optimization can be conducted exclusively from the historic process data wherein the detailed knowledge of process phenomenology (reaction mechanism, kinetics etc.) is not required. Using SVR-GA strategy, a number of sets of optimized operating conditions were found. The optimized solutions, when verified in an actual plant, resulted in a significant improvement in the quality.

Key words: SVR; GA; modeling and optimization.

Cut throat competition, open global market and shrinking profit margin forced the process industries to monitor and improve the product quality through a faster and more systamatic way. Although the most reliable approach to quality improvement will be the use of precise first-principle models, such models are not available in most newly developed processes and modeling of a complex industrial process is very difficult and time-consuming. In the glycol industry, for example, the relationship of operating conditions to product quality is not clear. The product qualities have been usually maintained by skilled operators on the basis of their experience and intuition. Although much effort has been devoted to clarify the relationship between operating conditions and product quality, the problem remains unsolved. No industry acceptable first-principle model is available for the product quality improvement. Another option to solve this difficult situation is the use of historic operation data. In the last decade or so, data-based approaches have been widely accepted for process control and monitoring in various industries. In the petrochemical industry, for example, lots of operating and QC variable data are generated in every few seconds from multitude of sensors. To achieve the product quality improvement, we need to develop a system having at least the following functions: 1) to predict the product quality from operating conditions, 2) to derive better operating conditions that can improve the product quality and 3) to detect faults or malfunctions for preventing an undesirable operation. The first function is realized by developing a soft-sensor, which is a mathematical model to relate operating conditions to the product

Corresponding author: S.K. Lahiri, National institute of technology, Durgapur, India.
E-mail: sk_lahiri@hotmail.com
Paper received: 14 November, 2008
Paper revised: 2 February, 2009
Paper accepted: 3 August, 2009

quality. On the basis of the model, the second function is realized by formulating and solving an optimazation problem. The third function is realized by a multivariate statistical process control (MSPC).

When hardware sensors are not available, soft--sensors are key technologies for producing high quality products. Even when hardware sensors can be used, operators and engineers have found the problems listed in Table 1. These problems with hardware sensors were identified as the results of a questionnaire to 26 companies in Japan (PSE 143 Committee, 2004). Soft-sensors are judged to be useful for addressing these problems.

Table 1. Problems with hardware sensors; the results of a questionnaire to 26 companies in Japan

Percentage	Recognized problem
27	Time consuming maintenance
21	Need for calibration
15	Aged deterioration
13	Insufficient accuracy
10	Long dead time, slow dynamics
8	Large noise
2	Low reproducibility
4	Others

For successful monitoring and control of chemical plants, there are important quality variables that are difficult to measure on-line, due to limitations such as cost, reliability, and long dead time. These measurement limitations may cause important problems such as product loss, energy loss, toxic byproduct generation, and safety problem. A soft sensor, an inferential model, can estimate the qualities of interest on-line using other available on-line measurements such as temperatures and pressures. An inferential sensor provides valuable real-time information that is necessary for effective quality control. The major purpose of using soft-sensors is to 1) stabilize the product quality via its online estimation, 2) reduce the energy and material consumption *via* an effective operation close to specifications/constraints and 3) validate online analyzers by comparison with the soft sensors. The soft sensor can be derived from the first principal model when the model offers the sufficient accuracy within the reasonable computation time. However, due to complexity in industrial processes there are cases when the first principle model is not available, or sometimes it takes too much time to compute. As a result, empirical data driven models are the most popular ones to develop soft sensors. Empirical models are usually obtained based on various modeling techniques such as multivariate statistics, artificial neural network and support vector regressions (SVR). In recent years, the support vector regression has been widely used (Vapnik [1,2] and Vapnik, Smola and Golowich [3]) as a useful tool to the nonlinear soft sensing modeling. SVR is a computer modeling approach that learns from examples through iterations without requiring a prior knowledge of the relationships of process parameters and QC variable. It is also capable of dealing with uncertainties, noisy data, and non-linear relationships. SVR modeling has been known as "effortless computation" and readily used extensively due to their model-free approximation capabilities of complex decision-making processes. Once an SVR based process model is developed, it can be used for predicting the important quality variable. Also, it can be interfaced with online DCS and continuous monitoring can be achieved to yield the better process control. This SVR based process model can also be used for the process optimization to obtain the optimal values of the process input variables that maximize the quality of the product. In such situations, an efficient optimization formalism known as Genetic Algorithm can be used. In the recent years, Genetic Algorithms (GAs) that are members of the stochastic optimization formalisms have been used (Cartwright and Long. [5], Polifke, Geng and Dobbeling [4], Garrard and Fraga [6], Goldberg [7] and Hanagandi, Ploehn and M. Nikolaou [8]) with a great success in solving problems involving very large search spaces.

In the present paper, SVR formalism is inte-grated with Genetic Algorithms to arrive at soft sensor modeling and process optimization strategies. The strategy (henceforth referred to as "SVR-GA") use an SVR as the nonlinear process modeling paradigm for development of soft sensor, and the GA for optimizing the input space of the SVR model so that an improved process performance is realized. This paper describes a systematic approach to the development of inferential measurements of ultraviolet (UV) transmittance (QC variable of monoethylene glycol (MEG) product in glycol plant) using SVR regression analysis. After predicting the UV accurately, model inputs describing process operating variables are optimized using GAs with a view to maximize the UV. The SVR--GA is a new strategy for the chemical process modeling and optimization. The major advantage of the strategies is that modeling and optimization can be conducted exclusively from the historic process data wherein the detailed knowledge of the process phenomenology (reaction mechanism, kinetics etc.) is not required. The optimized solutions when verified in ac-

tual commercial plant resulted in a significant improvement in the MEG quality.

Development of the soft sensor for the product quality in a monoethylene glycol plant

Recently, mono-ethylene glycol (MEG) has emerged as the most important petrochemical product as its demand and price have been considerably rising all over the world in the last few years. It is extensively used as a main feed for the polyester fiber and polyethylene tere-phthalate plastics production. UV is one of the most important quality parameters of MEG and it indirectly represents the impurities level such as aldehyde, nitrogeneous compound and iron in the MEG product. In laboratory, MEG product sample is exposed to UV light of different wavelengths (220, 250,275 and 350 nm) and percentage of the UV light transmitted through the MEG sample is measured. UV transmittance measures the presence of impurities in MEG that absorb light in the ultraviolet region of the spectrum. These undesirable compounds are in trace quantities in the parts per billion (ppb) ranges and primarily unknown in a chemical structure. Samples showing higher transmittance are considered to be of a greater quality grade. In Glycol plant the MEG is drawn off from MEG distillation column as a product, its UV transmittance is affected by many things such as impurity formation in an upstream ethylene oxide reactor, impurity formation and accumulation in MEG column bottoms due to thermal degradation of glycol, non removal and accumulation of aldehyde in the system etc. Since these UV deteriorating impurities are in ppb ranges, they are very difficult to detect during the MEG production process and they have hardly any effect on process parameters. That is why it is very difficult for any phenomenological model for UV prediction to succeed in an industrial scenario. Normally, online UV analyzers are not available to monitor the product MEG UV analysis in an ethylene glycol plant. Off-line methods for MEG quality control is a common practice among the manufacturers, where a sample is withdrawn from the process and product stream for laboratory analysis several times a day and analyzed by time consuming laboratory analysis. In the event of a process malfunction or operating under a suboptimal condition, the plant will continue to produce an off-spec product until lab results become available. For a big world class capacity plant this represents a huge amount of offspec production results in enormous financial losses. This necessitates the online UV sensors or analyzers which can give UV continuously on a real time basis. Accurate, reliable and robust UV soft sensors can be a viable alternative in this scenario. Making of a UV soft sensor is not an easy task as a rigorous mathematical model for the MEG product UV is still not available in literature which can predict UV transmittance to minimize the dependency on a lab analysis. The comprehensive process model is expected to take into account various subjects, such as chemistry, chemical reaction, UV deteriorating compound generation and accumulation which consequently become very complex. Industry needs this mathematical model to predict MEG UV on real time basis so that process parameters can be adjusted before the product goes off specification.

In this study, the SVR-GA strategy has been used to model and optimize the MEG product UV for a commercial plant The optimized operating conditions led to maximized UV of the product (MEG). The best sets of operating conditions obtained thereby when subjected to actual plant validation indeed resulted in significant enhancements in UVs.

HYBRID SVR AND GA BASED MODELING

Process modeling and optimization formalisms

The process optimization objective under consideration is expressed as:

Given the process data comprising values of the multiple process inputs and the corresponding values of the process outputs (MEG UV in this case), find the optimal values of the process inputs such that the prespecified measures of the process performance is maximized.

The SVR-GA strategy fulfills the above-stated objective in two steps. In the first step, an SVR-based process model is developed. This model has the inputs describing process operating parameters and variables such as reflux ratio, reflux flow, MEG column top pressure, MEG column condenser pressure, MEG column control temperature, MEG column feed flow, upstream drying column control temperature, drying column bottom temperature, crude glycol reprocessing flow and its outputs represent the process output variable mono-ethylene glycol UV. In the second step of the SVR-GA procedure, the input space of the SVR model is optimized using a GA algorithm such that the optimized process inputs result in the enhanced values of the output variables.

This optimization problem can be formulated as:

Maximize UV = f (*reflux ratio, reflux flow, MEG column top pressure, MEG column condenser pressure, MEG column control temperature, MEG column feed flow, drying column control temperature, drying*

column bottom temperature, crude glycol reprocessing flow); (1)

SVR-based modelling

i) Linear regression using SVR: Industrial data contains noise. Normally different transmitter, signal transmissions etc. add these noises with process parameters. Normal regression techniques try to reduce the prediction error on noisy training data. This empirical risk minimization (ERM) principle is generally employed in the classical methods such as the least-square methods, the maximum likelihood methods and traditional ANN. The formulation of SVR embodies the structural risk minimization (SRM) principle, which has been shown to be superior, to traditional empirical risk minimization (ERM) principle, employed by conventional neural networks. SRM minimizes an upper bound on the expected risk, as opposed to ERM that minimizes the error on the training data. It is this difference which equips SVM with a greater ability to generalize.The SVR (Cherkassky and Ma [9]) algorithm attempts to position a tube around the data as shown in Figure 1. ε is a precision parameter representing the radius of the tube located around the regression function (see Figure 1); the region enclosed by the tube is known as "e-intensive zone". The diameter of the tube should ideally be the amount of noise in the data. The optimization criterion in SVR penalizes those data points the y values of which lie more than ε distance away from the fitted function, $f(x)$. There are two basic aims in SVR. The first is to find a function $f(x)$ that has at most ε deviation from each of the targets of the training inputs. For the linear case, f is given by:

$f(x) = <w.x> + b$

where $<a.b>$ is the dot product between a and b.

At the same time, we would like this function to be as flat as possible. Flatness in this sense means a small w. This second aim is not as immediately intuitive as the first, but still important in the formulation of the optimization problem used to construct the SVR approximation:

$$\text{minimize } \frac{1}{2}|w|_2 ;$$

$$\text{subject to } \begin{cases} y_i - < w.x_i > -b \le \varepsilon \\ < w.x_i > +b - y_i \le \varepsilon \end{cases}$$

A key assumption in this formulation is that there is a function $f(x)$ that can approximate all input pairs (x_i,y_i) with ε precession; however, this may not be the case or perhaps some error allowance is desired. Thus the slack variable, ξ_i and ξ_i^*, can be incorporated into the optimization problem to yield the following formulation:

$$\text{minimize } \frac{1}{2}|w|_2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) ;$$

$$\text{subject to } \left\{ \begin{array}{c} y_i - < w.x_i > -b \le \varepsilon + \xi_i \\ < w.x_i > +b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{array} \right\}$$

In Fig. 1, the sizes of the stated excess positive and negative deviations are depicted by ξ_i and ξ_i^*, which are termed "slack" variables. Outside the $[-\varepsilon,\varepsilon]$ region, the slack variables assume nonzero values. The SVR fits $f(x)$ to the data in such a manner that: *i*) the training error is minimized by minimizing ξ_i and ξ_i^* and *ii*) w^2 is minimized to increase the flatness of $f(x)$ or to penalize over complexity of the fitting function.

The constant $C > 0$ determines the tradeoff between flatness (small w) and the degree to which de-

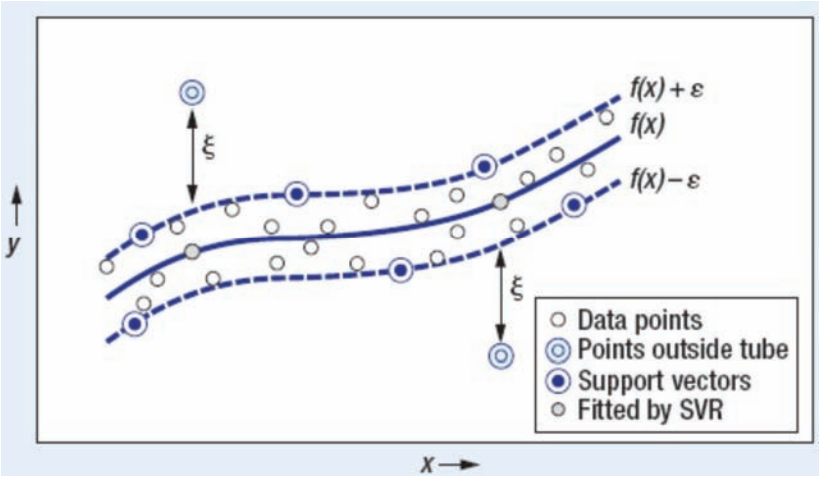

Figure 1. Schematic of SVR using an e-insensitive loss function.

viation larger than ξ are tolerated, and l is the number of samples. This is reffered to as the ε insensitive loss function proposed by Vapnik *et al*. which enables as a sparse set of support vectors to be obtained for the regression.

ii) Nonlinear regression using SVR. Nonlinear function approximation can be achieved by replacing the dot product of input vectors with a nonlinear transformation on the input vectors. This transformation is referred to as the kernel function and is represented by $k(x,x')$, where x and x' are each input vectors.

Applying the kernel function to the dot product of input vectors, we obtain:

$$\text{maximize} \begin{Bmatrix} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{Bmatrix}$$

$$\text{subject to} \begin{Bmatrix} \sum_{i}^{l}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{Bmatrix}$$

By using the Kernel function and corresponding kernel matrix, nonlinear functions approximations can be achieved with SVR while mantaining the simplicity and computational efficiency of linear SVR approximations.

Training and testing

Training a support vector machine consists of an iterative process in which the SVR is given the desired inputs along with the correct outputs for those inputs. It then seeks to alter its margin (w) and bias (b) to try and produce the correct output (within a reasonable error margin). If it succeeds, it has learned the training set and is ready to perform upon previously unseen data. If it fails to produce the correct output it re-reads the input and again tries to produce the correct output. The margins and bias are slightly adjusted during each iteration through the training set (known as a training cycle) until the appropriate margins and bias has been established. Depending upon the complexity of the task to be learned, many thousands of training cycles may be needed for the SVR to correctly identify the training set. Once the output is correct, the margins (w) and bias (b) can be used with the same SVM on unseen data to examine how well it performs. SVM learning is considered successful only if the system can perform well on test data on which the system has not been trained. This capability of a SVM is called generalizability.

Techniques used in GA

For the implementation of GAs, there are certain well defined steps.

Coding

Coding is the method by which the variables xi are coded into string structures. A linear mapping rule is used for the purpose of coding:

$$x_i = x_i^{(l)} + \frac{x_i^{(u)} - x_i^{(l)}}{2^{l_i} - 1} \text{ (decoded value)}$$

The decoded value (of s_i, the binary digit in the coding) is given by the rule:

$$\text{Decoded value} = \sum_{i=0}^{l_i - 1} 2^i s_i, \text{ where } s_i \in (0,1)$$

For example, for a binary code(0111), the decoded value will be:

$(0111) = (1)2^0+(1)2^1+(1)2^2+(0)2^3 = 7$

It is also evident that for a code of length l, there are 2^l combinations or codes possible. The length of the code depends upon the required accuracy for that variable.

$$\text{Accuracy} = \frac{x_i^{(u)} - x_i^{(l)}}{2^{l_i}}$$

So, adhering to the above mentioned rules, it is possible to generate a number of guesses, or, in other words, an initial population of coded points that lie in the given range of the function. Next step is calculation of fitness.

Fitness

As already been mentioned, GAs work on the principles of "survival of the fittest". This means that the good points or the points that yield maximum values for the function are allowed to continue in the next generation, while the less profitable points are discarded from calculations. Depending upon whether the initial objective function needs to be maximized or minimized, the fitness function is hence defined in the following ways:

$F(x) = f(x)$ for a maximization problem

$F(x)=1/(1+f(x))$ for a minimization problem

The fitness function value for a particular coded string is known as the string's fitness. This fitness value is used to decide whether a particular string carries to the next generation or not.

GA operation begins with a population of random strings. These strings are selected from a given range of the function and represent the design or de-

cision variables. To implement our optimization routine, three operations are carried out:

- reproduction;
- crossover;
- mutation.

Operators in GA

Reproduction

The reproduction operator is also called the selection operator. This is because it is this operator that decides the strings to be selected for the next generation. The end results of this operation are the formation of a "mating pool", where the above average strings are copied in a probabilistic manner. The rule can be represented as:

(Probability of selection into mating pool) $\propto$ (Fitness of string)

The probability of the selection of the i^{th} string into the mating pool is given by:

$$p_i = \frac{F_i}{\sum_{i=1}^{n} F_j}$$

where F_i is the fitness of the i^{th} string. F_j is the fitness of the j^{th} string and n is the population size.

The average fitness of all the strings is calculated by summing the fitness of individual strings and dividing by the population size, and is represented by the symbol $\overline{F}$:

$$\overline{F} = \frac{\sum_{i=1}^{n} F_i}{n}$$

It is obvious that the string with the maximum fitness will have the most number of copies in the mating pool. This is implemented using "roulette wheel selection". The algorithm of this procedure is as follows:

Step 1: using F_i, calculate p_i.

Step 2: calculate the cumulative probability P_i.

Step 3: generate n random numbers (between 0 and 1).

Step 4: copy the string that represents the chosen random number in the cumulative probability range into the mating pool. A string with higher fitness will have a larger range in cumulative probability and so has more probability of getting copied into the mating pool.

At the end of this implementation, all the strings that are fit enough would have been copied into the mating pool and this marks the end of the reproduction operation.

Crossover

After the selection operator has been implemented, there is a need to introduce some amount of randomness into the population in order to avoid getting trapped in local searches. To achieve this, a cross over operation was performed. In the crossover operation, new strings are formed by the exchange of the information among strings of the mating pool. For example:

00|000 $\rightarrow$ 00|111

11|111 11|000 $\leftarrow$ Crossover point

Parents Children

The strings are chosen at random and a random crossover point is decided, and crossover is performed in the method shown above. It is evident that, using this method, better strings or worse strings may be formed. If worse strings are formed, then they will not survive for long, since the reproduction will eliminate them. But what if the majority of the new strings formed are worse? This undermines the purpose of reproduction. To avoid this situation, we do not select all the strings in population for crossover. We introduce a crossover probability (p_c). Therefore, ($100p_c$), %, of the strings are used in crossover. (1-p_c), %, of the strings is not used in crossover. Through this, we have ensured that some of the good strings from the mating pool remain unchanged. The procedure can be summarized as follows:

Step 1: select ($100p_c$) of the strings out of the mating pool at random.

Step 2: select pairs of strings at random.

Step 3: decide a crossover point in each pair of strings (again this done by a random number generation over the length of the string and the appropriate position is decided according to the value of the random number).

Step 4: perform the crossover on the pairs of the strings by exchanging the appropriate bits.

Mutation

Mutation involves making changes in the population members directly, that is, by flipping randomly selected bits in certain strings. The aim of the mutation is to change the population members by a small amount to promote local searches when the optimum is nearby. Mutation is performed by deciding a mutation probability, p_m, and selecting strings, on which mutation is performed, at random. The procedure can be summarized as follows:

Step 1: calculate the approximate number of mutations to be performed by:

Approximate number of mutations = n/p_m

Step 2: generate random numbers to decide whether mutation is to be performed on a particular population member or not. This is decided by a "coin toss". If the generated random number of a particular population member is lower than the prespecified number (again chosen randomly) then that represent the "true" case. If it is not lower than the pre specified number, then that will be considered as "false" case. If the outcome is true, perform mutation, if false, do not.

Step 3: if the outcome found in step 2 is true for a particular population member, then generate another random number to decide the mutation point over the length of the string. Once decided, flip the bit corresponding to the mutation point.

At the end of mutation, the strings obtained represent the next generation. The same operations are carried out on this generation until the optimum value is encountered.

GA-based optimization of SVR models

Having developed an SVR-based process model, a GA algorithm is used to optimize the N- dimensional input space of the SVR model. Conventionally, various deterministic gradient-based methods are used for performing optimization of the phenomenological models. Most of these methods require that the objective function should simultaneously satisfy the smoothness, continuity, and differentiability criteria. Although the nonlinear relationships approximated by an SVR model can be expressed in the form of generic closed-form expressions, the objective function(s) cannot be guaranteed to satisfy the smoothness criteria. Thus, the gradient-based methods cannot be efficiently used for optimizing the input space of an SVR model and, therefore, it becomes necessary to explore alternative optimization formalisms, which are lenient towards the form of the objective function.

Genetic algorithms combine the "survival of the fittest" principle of natural evolution with the genetic propagation of characteristics, to arrive at a robust search and optimization technique. The principal difference between the widely used deterministic gradient-based optimization schemes and the stochastic ones such as GA is that the latter class of methodologies involves a random component at some stage in their implementation. For instance, GAs manipulate a set of candidate solutions at random with the objective of sampling the search (solution) space as widely as possible while at the same time trying to locate promising regions for further exploration.

Principal features possessed by the GAs are: *i*) they are zeroth order optimization methods requiring only the scalar values of the objective function, *ii*) capability to handle nonlinear, complex and noisy objective functions, *iii*) they perform global search and thus are more likely to arrive at or near the global optimum and *iv*) their search procedure being stochastic, GAs do not impose pre-conditions, such as smoothness, differentiability and continuity on the objective function form.

The optimization objective underlying the GA-based optimization of an SVR model is defined as: Find the N-dimensional optimal decision variable vector, $x^* = [x_1^*, x_2^*, \ldots, x_N^*]^T$ representing optimal process conditions so that it maximizes process outputs, yk ; $k = 1,2,\ldots,K$. The corresponding single objective function f to be maximized by the GA is defined in Eq. (1). In the GA procedure, the search for an optimal solution (decision) vector, $\mathbf{x}^*$, begins from a randomly initialized population of probable (candidate) solutions. The solutions, usually coded in the form of binary strings (*chromosomes*), are then tested to measure their fitness in fulfilling the optimization objective. Subsequently, a main loop comprising following operations is performed: *i*) selection of better (fitter) parent chromosomes, *ii*) production of an offspring solution population by crossing over the genetic material between pairs of the fitter parent chromosomes and *iii*) mutation (bit-flipping) of the offspring strings. The implementation of this loop generates a new population of candidate solutions, which when compared to the previous population, usually fares better at fulfilling the optimization objective. The best string that evolves after repeating the above described loop till convergence, forms the solution to the optimization problem (refer Appendix 1). The stepwise procedure for the GA-based optimization of an SVR model is provided in a flowchart in Figure 2.

CASE STUDY OF MONO-ETHYLENE GLYCOL PRODUCT UV TRANSMITTANCE

Figure 3 describes a brief process description of glycol section of mono-ethylene glycol plant (MEG) where glycol (90%) and water solution (10%) fed to the drying column to remove the water from the drying column top. The bottom of the drying column fed to the MEG column to distil MEG from heavier glycols (namely diethylene glycol and triethylene glycol). MEG product (99.9 mass% purity) is withdrawn from the MEG column below the top packing bed. An over-

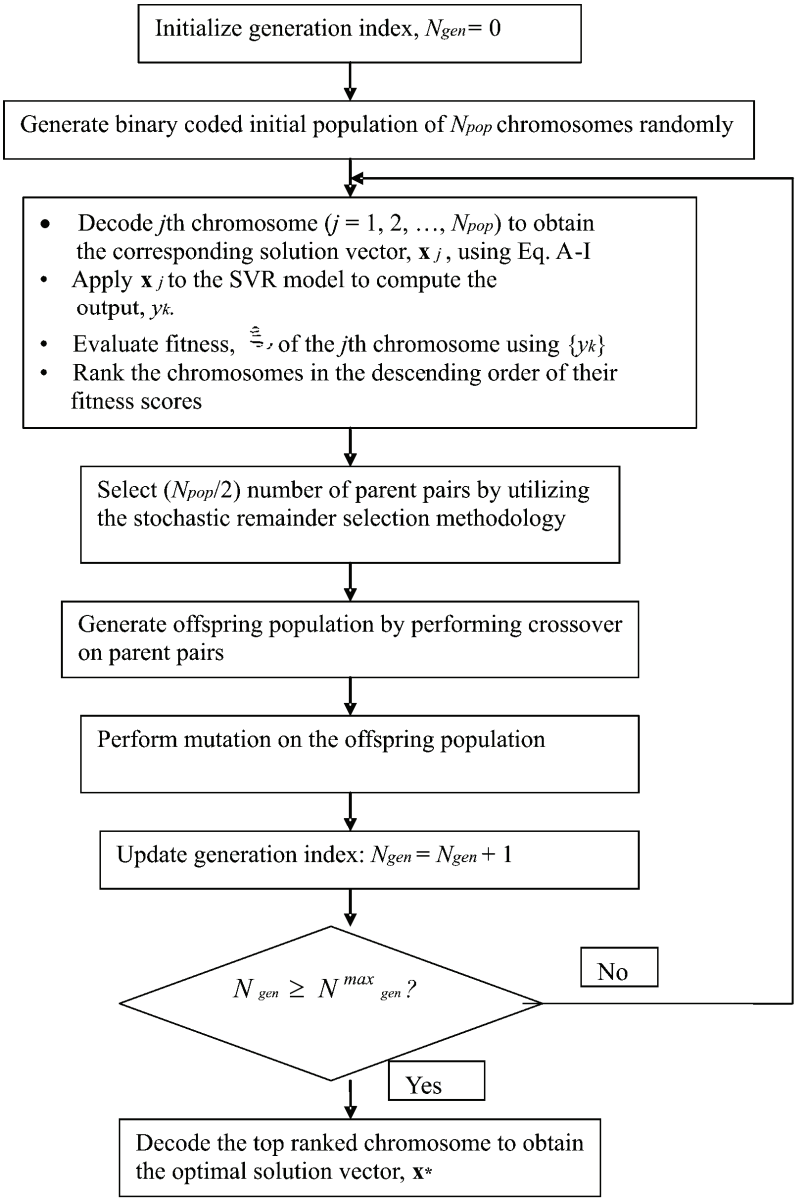

Figure 2. Flowchart for GA based optimization of SVR model.

head vapor purge of up to 10% of the product is taken out to purge the light compounds. Figure 3 shows the location of input parameters from the drying column and MEG column which were used to build the model of UV.

Development of the SVR-based correlation

The development of the SVR-based correlation had been started with the collection of a large data-bank. The next step was to perform a neural regression, and to validate it statistically.

Collection of data

The quality and quantity of data is very crucial in SVR modeling as learning is primarily based on these data. Hourly average of actual plant operating data at the steady state was collected for approximately one year. Data were checked and cleaned for obvious inaccuracy and retain those data when plant operation was in a steady state and smooth. Finally 6273 records are qualified for neural regression. This wide range of database includes plant operation data at various capacities starting from 75% capacity to 110% of design capacity.

Identification of input and output parameters

Based on the operating experience in a glycol plant, all physical parameters that influence UV are put in a so-called “wish-list”. Out of the number of inputs in a “wish list”, several sets of inputs were made

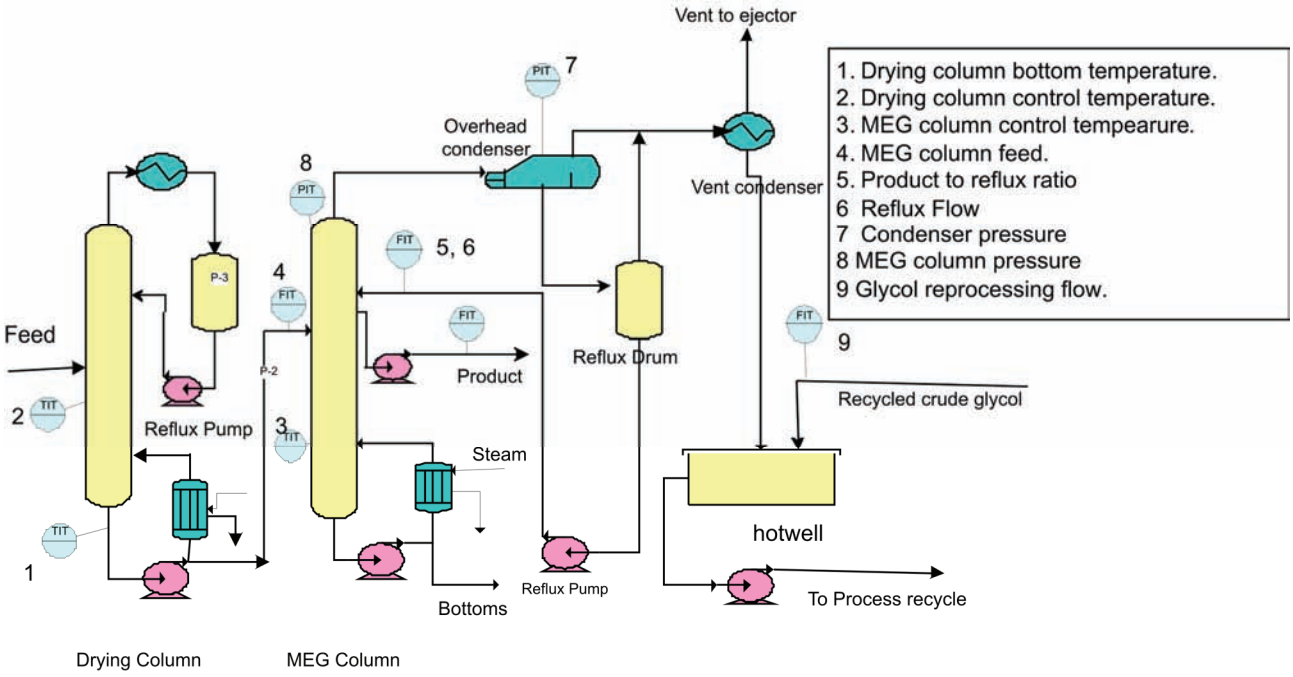

Figure 3. Process flow diagram of drying and MEG column.

and tested *via* rigorous trial-and-error on the SVR. Finally, the nine input variables (Table 2) have been finalized to predict UV.

Support vector regression

For modeling purposes, the column operating conditions data (see table 2) can be viewed as an example input matrix (**X**) of size (6273×9), and the corresponding UV data as the example output matrix (**Y**) of size (6273×1). For SVR training, each row of **X** represents a nine-dimensional input vector $\mathbf{x} = = [x_1,x_2,...,x_9]^T$, and the corresponding row of matrix **Y** denotes the one-dimensional desired (target) output vector $\mathbf{y} = [y_1]^T$. As the magnitude of inputs and outputs greatly differ from each other, they are normalized in -1 to +1 scale. 80% of total dataset was chosen randomly for training and the rest 20% was selected for validation and testing.

There are five different parameters to be evaluated to design a successful regression model. These five parameters are 1) kernel type, 2) type of loss function, 3) kernel parameter, *i.e.* degree of polynomial etc., 4) C (represents the trade-off between the model-complexity and the approximation error) and 5) ε (signifies the width of the ε-insensitive zone used to fit the training data).

Since the prior knowledge is not there regarding the suitability of a particular value of any of the above five parameters, the strategy adopted here is holistic and summarized in Figure 4. The SVR performance was evaluated exhaustively for all combinations of above parameters. All the kernel types available in literature (namely linear, polynomial, Gaussian radial basis function, exponential radial basis function, splines, B-splines) is tested with all combinations of the loss function (namely ε - insensitive loss function, quadratic loss function). The degree of kernel was varied from 1 to 6, capacity control varied from 10000

Table 2. Input and output variable for SVR model

Input Variables	Output variables
Reflux ratio (Product flow / Reflux flow)	Mono-ethylene glycol UV
Reflux flow (Mt/h)	
MEG Column top pressure (mmHg)	
MEG Column condenser pressure (Barg)	
MEG column control temperature (°C)	
MEG column feed flow (Mt/h)	
Drying column control temperature (°C)	
Drying column bottom temperature (°C)	
Crude glycol reprocessing flow	

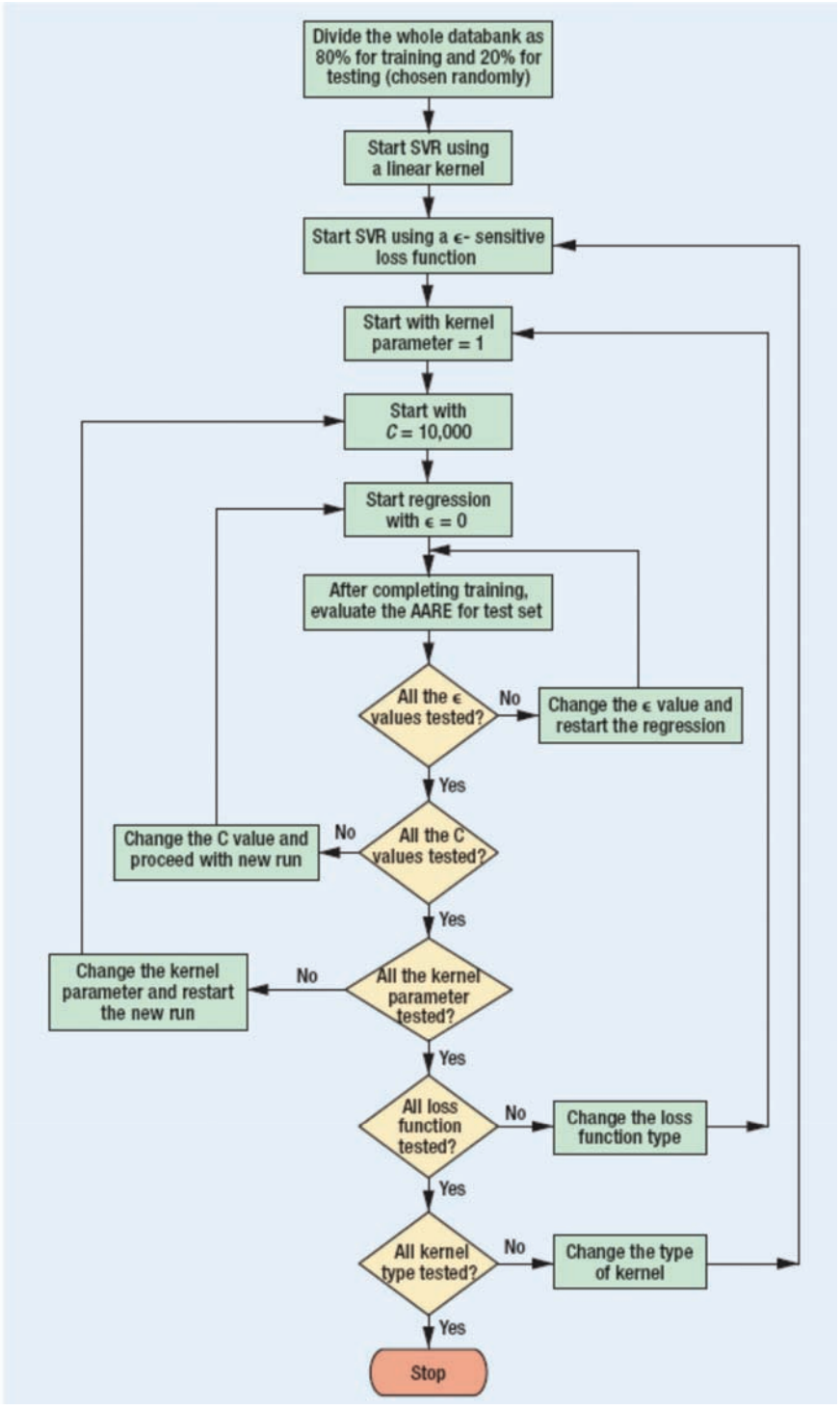

Figure 4. Flow chart of SVR algorithm implementation.

to 0.1 (typically six values: 10000, 1000, 100, 10, 1, 0.1) and epsilon varies from 0 to 25 (typically six values: 0, 0.1, 1, 10, 25). Each run was exposed with the same training and testing data and AARE and sigma was calculated for each run.

The statistical analysis of SVR prediction is based on the following performance criteria:

1. The average absolute relative error (*AARE*) on a test set should be minimum:

$$AARE = \frac{1}{N}\sum_{1}^{N}\left|\left(\frac{y_{\text{predicted}} - y_{\text{experimental}}}{y_{\text{experimental}}}\right)\right|$$

2. The standard deviation of error(σ) should be minimum:

$$\sigma = \sqrt{\sum_{1}^{N} \frac{1}{N-1} (|(y_{\text{predicted}(i)} - y_{\text{experimental}(i)})/y_{\text{experimental}(i)}| - AARE)^2}$$

RESULTS AND DISCUSSIONS

SVR model developments for UV soft sensor

While the training set was utilized for the iterative updation of the SVR parameters (*b* and *w*), the test set was used for simultaneously monitoring the generalization ability of the SVR model. For developing an optimal SVR model, its five-structural parameter described above was varied systematically. For choosing an overall optimal model, the criterion used was the least *AARE* for the test set. The optimal model that satisfied this criterion has Exponential radial basis function(erb *f*), ε insensitive loss function, width of erb $f = 2, C = 1000$, and $\varepsilon = 0.1$.The average error (*AARE*) for training and test set is calculated as 0.04 and 0.042% and corresponding cross correlation coefficient (*R*) calculated as 0.84 and 0.83, respectively. The low and comparable training and test error *AARE* values indicate good prediction and generalization ability of the trained SVR model. Good prediction and generalization performance of the model is also evident from the high and comparable *R* values corresponding to both the outputs of training and test sets. To validate the reliability of the model, the actual plant data were taken from DCS at different plant load at different point of time and actual lab measured UV was compared with the model predicted UV.

Figure 5 describes a comparison of the outputs as predicted by the SVR model and their target values. Considering the fact that all the input output data are from a real plant with their inherent noise, the very low prediction error can be considered as an excellent SVR model. Once developed, this SVR model can be used to quantitatively predict the effects of all input parameters on the MEG product UV transmittance.

GA-based optimization of the SVR model

After development of a successful SVR model of glycol column, the next step is to find out the best set of operating conditions which lead to maximum UV. GA based hybrid model was run and optimum parameters were evaluated (within their permissible operating limit). Figure 6 describes the actual *versus* the optimum UV. From Figure 6 it is clear that by making

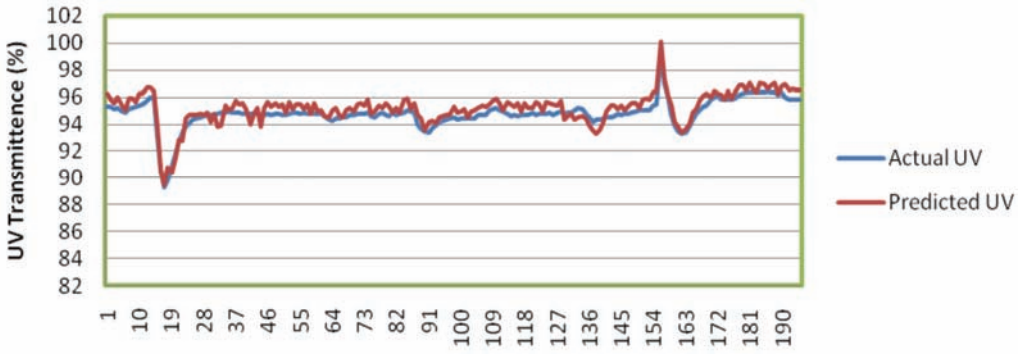

Figure 5. Actual vs. predicted UV.

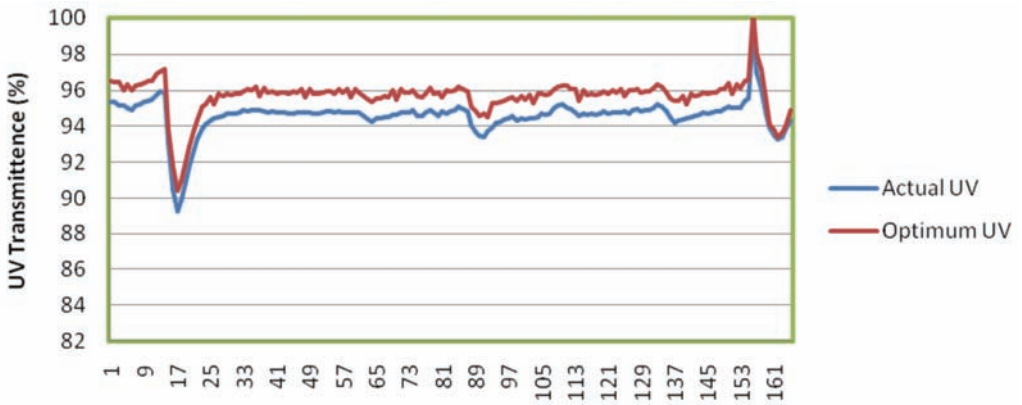

Figure 6. Actual vs. optimum UV.

a small change in the nine input parameters, the 1 to 2% rise in UV can be made. Refer to Table 3 for the optimum value of input variables calculated by GA algorithm. Drying column control temperature was found to have a significant effect on MEG product UV. This temperature will help to strip out UV deteriorating compounds from the drying column itself before they enter MEG column. Three cases were run with three different limits of this temperature. The optimum value is shown in Table 3.The program was made online where it gives the operator what should be the nine input parameters at different time to maximize the UV in real time basis. After verifying all the calculations, the optimum input parameters were maintained in an actual plant and benefit found was exactly the same as calculated. This ensures the validation and accuracy of this calculation.

Table 3. Optimum value of input variables calculated by GA algorithm

Parameters		Case 1			Case 2			Case 3		
		Min value	Optimum value	Max value	Min value	Optimum value	Max value	Min value	Optimum value	Max value
Input	Reflux ratio (product flow/reflux flow)	0.70	0.78	0.78	0.70	0.78	0.78	0.77	0.77	0.78
	Reflux flow (Mt/h)	110.0	114.52	115.0	110	110.00	115	112	112	115
	MEG column top pressure (mmHg)	92.0	94.75	97.0	92	92	97	92	92	97
	MEG column condenser pressure (Barg)	1.60	1.66	1.67	1.60	1.60	1.68	1.67	1.67	1.70
	MEG column control temperature (°C)	168.80	169.0	169.0	168.80	168.8	169.0	168.80	168.80	169.0
	MEG column feed flow (Mt/h)	98.0	98.0	98.0	100	100	101	99	99	101
	Drying column control temperature (°C)	85.0	90.92	100.0	82	82	90	90	90	95
	Drying column bottom temperature (°C)	165.0	166.0	166.0	165	166	166	165	165	166
	Crude glycol reprocessing flow (Output)	0.0	0.0	8.0	0.0	0.05	8.0	0.0	0.0	8.0
Output	Optimized UV	-	97.60	-	-	95.0	-	-	95.40	-
	Actual UV	-	96.0	-	-	93.41	-	-	94.78	-

CONCLUSION

This paper introduces SVR into soft sensing modeling and proposes a new soft sensing modeling method based on SVR. In the strategy, a soft sensor model is developed using SVR method followed by the input space of that model being optimized using GAs so that the process performance is maximized. The major advantage of the SVR-GA strategy is that modeling and optimization can be conducted exclusively from the historic process data wherein the detailed knowledge of the process phenomenology (reaction mechanism, kinetics etc.) is not required. Effective results indicate that SVR modeling method provides a new tool for soft sensing modeling and has a promising application in the industrial process applications. Using SVR-GA strategy, a number of sets of optimized operating conditions leading to the maximized product UV was obtained. The optimized solutions, when verified in an actual plant, resulted in a significant improvement in the product UV.

REFERENCES

[1] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, New York, 1995

[2] V. Vapnik, Statistical Learning Theory, John Wiley, New York, 1998

[3] V. Vapnik, A. Smola, ,S. Golowich, Adv. Neural Inform. Proc. Syst. **9** (1996) 281-287

[4] W. Polifke, W. Geng, K. Dobbeling, Combus. Flame **113** (1988) 119-120

[5] H.M. Cartwright, R. A. Long, Ind. Eng. Chem. Res. **32** (1993) 2706-2713

[6] A. Garrard, E. S. Fraga, Comput. Chem. Eng. **22** (1998) 1837-1850

[7] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, New York, 1989

[8] V. Hanagandi, H. Ploehn, M. Nikolaou, Chem. Eng. Sci. **51** (1996) 1071-1078.

[9] V. Cherkassky, Y. Ma, Neural Networks **17**(1) (2004) 113-126.

APPENDIX 1

Steps performed in GA

Step 1 (initialization). Set generation index (N_{gen}) to zero and generate a population of N_{pop} binary strings (chromosomes) randomly; each string consisting of l_{chr} bits is divided into N segments equal to the number of decision (input) variables to be optimized.

Step 2 (fitness computation). Decode j^{th} binary-coded chromosome ($j = 1,2,\ldots,N_{pop}$) to obtain its equivalent decimal-valued solution vector ($\mathbf{x}_j$) using:

$$\mathbf{x}_{j,n} = x_n^L + \frac{(x_n^U - x_n^L)\times S_n}{2^{l_n}-1}; \quad \sum_{n=1}^{N} l_n = l_{chr} \quad [x_1^*, x_2^*, \ldots, x_N^*]^T$$

where x_n^L and x_n^U refer to the lower and upper bounds on x_n, respectively; l_n is the length of n^{th} binary segment and S_n denotes the decimal equivalent of the nth binary segment. Next, depending upon the model to be optimized, vector $\mathbf{x}_j$ is used to compute the output of an SVR model; this output is subsequently used to calculate the fitness value ($j\xi$) of the j^{th} candidate solution. Upon computing the fitness scores of N_{pop} candidate solutions in the current population, the solutions are ranked in the decreasing order of their fitness scores.

Step 3 (parent selection). From the current population, choose N_{pop} number of parent chromosomes to form the mating pool. The members of this pool, which are used to produce offspring population, possess relatively high fitness scores. The commonly used parent selection techniques are the Roulette-Wheel (RW) method or its more stable variant known as the stochastic remainder selection (SRS).

Step 4 (crossover). Randomly select $N_{pop}/2$ number of parent pairs from the mating pool and perform a crossover operation on each pair with the probability equal to pcross ($0 < p_{cross} \leq 1$). In crossover, parent strings are cut at the same randomly chosen crossover point to obtain two substrings per parent. The substrings are then mutually exchanged between the parents and combined to form two off spring chromosomes. This crossover operation is performed on all the parent pairs in the mating pool to obtain an offspring population of the size of the mating pool.

Step 5 (mutation). Flip (mutate) the bits of the offspring strings where the probability of a bit getting flipped (zero to one or vice versa) is equal to p_{mut}. The recommended range of p_{mut} is [0.01-0.05].

Step 6. Increment the generation index:

$$N_{gen} = N_{gen} + 1$$

Step 7. Repeat Steps 2-6 on the newly generated offspring strings until convergence is achieved. The criteria for the convergence are: N_{gen} exceeds its maximum limit (N_{maxgen}), or the fitness score of the best (fittest) string in the offspring population undergoes a very small or no change over successive generations. After convergence, the string possessing the highest fitness value is decoded to obtain the optimized decision variable vector, $\mathbf{x}^*$.