Optimal mix for high performance concrete by evolution strategies combined with neural networks

S Rajasekaran
Department of Infrastructure Engineering, PSG College of Technology, Coimbatore 641 004, India

Received 5 May 2005; accepted 27 September 2005

There is a growing interest in the application of non-traditional methods such as simulated annealing (SA) and genetic algorithm (GA) and evolution strategies (ES) for optimization of structural systems. In this paper, evolution strategies (ES) is used to find the optimal mix design for high performance concrete (HPC) comprised of cement, sand, coarse aggregate, water, silica fume and super plasticizer. It is required to get the optimal mix for strength for 120 MPa and slump of 120 mm. In order to get the equation for strength and slump, the sequential learning neural network (SLNN) proposed by Zhang and Morris is used. The cost function to be minimized is the cost of HPC/unit weight of HPC subjected to strength and slump constraints. It is concluded that the method proposed is highly suitable for getting the optimal mix for high performance concrete in practice.

IPC Code: C04B14/00, G06N5/00

Optimal design methods assist structural engineers to evolve the best possible design in terms of cost, weight, reliability or a combination of these parameters. Development of mathematical programming techniques has paved the way for new approaches to the optimization of structural systems. As these techniques demand large amount of computing resources, the history of development of this field can be directly related to the development of computer hardware and software technologies. Structural optimization using classical mathematical programming techniques were prohibitively expensive in the early days of development. Recent advances in computing scenario have resulted in high performance computing at relatively low cost. This made engineers to choose non-traditional optimization techniques by minimizing the approximations, assumptions and considering engineering aspects such as constructability into the model during formulation stage. Rigorous and realistic formulation providing rational but reliable solution of large practical problems is becoming the focus of present day research in structural optimisation. Decision making features exist predominantly in all fields of human activities including scientific techniques and affects every sphere of our lives. For example an aircraft wing can be made from aluminum or composite and once material and shape are chosen, there are many ways of devising the required internal structure. In civil engineering also, designing an optimal mix for high performance concrete to satisfy strength and slump requirement is a multi-disciplinary activity and it is a trial and error process. The optimization of real life problem requires satisfaction of multiple objectives as in the case of HPC like minimum cost mix having particular strength and slump. The design of HPC is still based on trial mix and as such no rigorous mathematical approach is available. Already Cheng Yeh has designed HPC using neural networks (NN) and non-linear programming. Kasperkiewicz et al. predicted the strength of HPC using Fuzzy ArtMap. But by training the experimental values using NN, it is possible to obtain the equations for strength and slump for any mix. These equations are highly non-linear and it is difficult to apply traditional optimization technique for solving such non-linear equations. Hence, it is necessary to go for non-traditional optimization technique. The aim of this paper is to focus attention on the non-traditional methods such as ES for engineering design optimization such as obtaining the optimal mix for HPC and the potential it offers for practical applications in industry.

The non-traditional methods are not aggressive in the sense that they do not attempt to move rapidly to a minimum using local approximations. The minimum found by these methods is not local minimum but global minimum. These methods are potential search

E-mail: drrajasekaran@gmail.com
and optimization algorithms for complex engineering optimization problem. GA-mimics the principle of natural genetics and natural solution to constitute search and optimization procedure. Dorigo and Gambardella$^3$ have drawn inspiration from the working of natural ant colonies to derive an optimization method known as Ant Colony Optimization (ACO). This method is particularly useful for solving large-scale communication networks. Random cost (RC) originally suggested by Berg$^4$ is based on diffusion equation in one dimension, which is highly stochastic. On the other hand, ES imitates biological evolution of nature and has three characters that make them different from conventional optimization algorithm. They are randomized operators, mutation, selection and recombination. ES achieves a high rate of convergence than GA due to their self-adaptability search mechanism and is considered to be more efficient for solving real world problems. ES were initially applied for continuous optimization problem but recently they have been implemented in discrete and mixed optimization problems. Hence in this paper, the application of ES has been explored.

**Sequential Learning Neural Networks (SLNN)**

Given the proportions of high performance concrete, there is no practical design equation available to predict the strength and slump of HPC. But it is possible to train the experimental data using well known backpropagation neural network (BPN) and once the weights of connectivity are determined, given any combination of constituents of HPC matrix, it is possible to infer the strength and slump. One of the main issues encountered in artificial neural network such as BPN is the determination of the structure of the neural network in the number of hidden layers and the number of hidden neurons in each hidden layer. In most reported applications, the number of hidden layers and the number of hidden neurons are determined based on experience. Very often several arbitrary architectures are tried and the one giving the best performance is selected. In some applications, genetic algorithm has been applied to determine the optimum number of hidden layers as well as number of optimum neurons in each hidden layer. Some neural networks could have more number of hidden neurons than necessary, resulting in neural network models, which are over parameterized and thus not parsimonious. Overparameterization could well deteriorate the generalization capability of the network. To counter this, the sequential learning approach for single hidden layer with single hidden neuron neural network has been proposed. This method allows for simultaneous building up and training of neurons. This can be used to train single hidden layer feed forward networks with sigmoidal neurons, radial basis function neurons (RBF) as proposed by Chen and Cowan$^5$ and neural network with mixed type of hidden neurons. Recently, several researchers have tackled this issue in a more systematic manner. In this paper, we apply the sequential learning proposed by Zhang and Morris$^6$.

For finding the strength and slump of HPC, SLNN is applied to both the problems with one single hidden neuron. If one needs more accuracy, one can add further hidden neurons gradually and after adding each hidden neuron the network is tested to see that the model error is minimum. For the application discussed in this paper one hidden neuron itself is sufficient.

Training can be stopped at the point where the network has smallest testing error. The classical Gram-Schmidt as given in Bathe$^7$ orthogonalization technique is used at each training step to build a set of orthogonal bases. Corresponding to the space spanned by hidden neuron output vector, sequential reduction of network error allows the most appropriate number of hidden neurons to be found. In this paper for SLNN training method originally proposed by Zhang and Morris$^6$ is considered with a single hidden neuron for the prediction of strength and slump of HPC.

**Sequential Training**

The sequential learning neural network is shown in Fig. 1. This neural network consists of ‘$m$’ input neurons, ‘$n$’ hidden neurons (in this work $n = 1$) and one output neuron. The weights of the synapses connecting $j^{th}$ neuron in the input layer and $i^{th}$ neuron in the hidden layer is denoted by $c_{ij}$. Similarly the weight of synapse connecting $i^{th}$ neuron in the hidden layer to the one output neuron is denoted by $w_i$. The neurons in the input and output layer are considered as linear neurons and hence

$$O_H$$ - Output of the input neuron = Input of the input neuron = $I_i = X_i$.

In the input layer we assume a bias neuron as $(m+1)^{th}$ neuron with input of 1. $I_{in}$ - Input of the $i^{th}$ hidden neuron is given by...
Fig. 1 — Sequential learning neural network architecture

![Sequential Learning Neural Network Architecture](image)

\[ I_{H_i} = \sum_{j=1}^{m} C_i^j X_j + C_i^{w+1} \] \hspace{1cm} (1)

If the \(i^{th}\) hidden neuron is a linear neuron then the output of the hidden neuron is given by

\[ Q_i = O_{H_i} = I_{H_i} \] \hspace{1cm} (2)

In case if \(i^{th}\) hidden neuron is a sigmoidal neuron then

\[ Q_i = O_{H_i} = \frac{1}{1 + e^{-\lambda X_{in}}} \] \hspace{1cm} (3)

In case if the \(i^{th}\) hidden neuron is a radial basis function (RBF) neuron with Gaussian function

\[ Q_i = O_{H_i} = e^{-\sum_{j=1}^{m} (X_j - C_i^j)^2 / (C_i^{w+1})^2} \] \hspace{1cm} (4)

In the above equations

\[ \{C_i\} = \langle C_i^1, C_i^2, C_i^3, \ldots, C_i^{w+1} \rangle^T \] \hspace{1cm} (5)

where \(\lambda\) is the sigmoidal gain in Eq. (4) and \(Q_i\) is the output of the \(i^{th}\) hidden neuron which is the notation followed by Zhang and Morris\(^6\).

In this paper, we use radial basis neuron for the hidden neuron and linear neuron for input and output layers.

If we denote the weights of the synapses connecting \(i^{th}\) neuron in the hidden layer to one neuron in the output layer as \(w_i\), the input to the output neuron is given by

\[ I_o = \sum_{i=1}^{n} w_i O_{H_i} \] \hspace{1cm} (6)

Since the output neuron is a linear neuron, the output of the output neuron is given by

\[ O_o = I_o = \sum_{i=1}^{n} w_i O_{H_i} \] \hspace{1cm} (7)

where \(O_o\) is the model output.

But the actual system output ‘\(y\)’ can be expressed as

\[ y = O_o + E_a = \sum_{i=1}^{n} w_i O_{H_i} + E_a \] \hspace{1cm} (8)

The equations we have derived are for one set of input and output data. If ‘\(n\)’ is the number of training data Eq. (8) can be written as

\[ \{Y\} = \{O_o\} + \{E_a\} = \sum_{i=1}^{n} w_i \{Q_i\} + \{E_a\} \] \hspace{1cm} (9)

\[ \begin{bmatrix} N \times 1 \\ N \times 1 \\ N \times 1 \\ N \times 1 \end{bmatrix} \]

where \(\{Y\}\) is a vector of system outputs, \(\{O_o\}\) is the vector of neural network model outputs, \(\{Q_i\}\) is the vector of outputs of the \(i^{th}\) hidden neuron and, \(\{E_a\}\) is the vector of model residuals and all the vectors are of order \(N \times 1\) where \(N\) is the number of training data.

Eq. (9) can be written as

\[ \{Y\} = w_i \{Q_i\} + \{E_i\} \] \hspace{1cm} (10)

\[ \{E_i\} = w_i \{Q_i\} + \{E_i\} \] \hspace{1cm} (11)

leading to

\[ \{E_{a+1}\} = w_i \{Q_i\} + \{E_a\} \] \hspace{1cm} (12)

Eq. (12) indicates a sequential way of neural network training, the first hidden neuron is used to model the relationship between input, output data, while the rest of the hidden neurons are used to model the relationships between input data and model residuals.
After each hidden neuron, the network is tested on the testing data. Training can be terminated when the model error on testing data cannot be reduced any further.

At the \(n\)th step of this sequential training the \(n\)th hidden neuron is introduced to model the relationship between the input data and the model residuals at the \((n-1)\)th stage \(\{E_{n-1}\}\). The vector \(\{Q_n\}\) can be decomposed into two parts, one is in the space spanned by \(\{Q_1\}, \{Q_2\}...\{Q_{n-1}\}\) and the other part is orthogonal to that space. The orthogonal vector can be denoted by \(\{R_n\}\) and hence Eq. (12) can be written as

\[
\{E_{n-1}\} = w_n \{R_n\} + w_n \{Q_n - R_n\} + \{E_n\} \quad \text{...}(13)
\]

The new information introduced by \(\{Q_n\}\) is due to \(\{R_n\}\) only since the effect of \(w_n (Q_n - R_n)\) in Eq. (13) can be reduced to a small quantity and hence eliminated by changing the previously obtained output layer weights \(w_1, w_2...w_{n-1}\). Hence Eq. (12) can be written as

\[
\{E_{n-1}\} = w_n \{R_n\} + \{E_n\} \quad \text{...}(14)
\]

or

\[
\{E_n\} = \{E_{n-1}\} - w_n \{R_n\} \quad \text{...}(15)
\]

The square of the second norm of \(\{E_n\}\) from Eq. (15) may be written as

\[
\|E_{n}\|^2 = (\{E_{n-1}\}^T - w_n \{R_n\}^T)(\{E_{n-1}\} - w_n \{R_n\}) \quad \text{...}(16a)
\]
\[
= \|E_{n-1}\|^2 - 2w_n \{E_{n-1}\}^T \{R_n\} - w_n^2 \{R_n\}^T \{R_n\} \quad \text{...}(16b)
\]

The objective at the \(n\)th training stage is then to find \(w_n, C_n\) such that \(\|E_n\|^2\) is minimized. Geometrically, this is equivalent to find the number of hidden neurons so as to produce an \(\{R_n\}\) which is most aligned with \(\{E_{n-1}\}\). Hence the weights are calculated as

\[
w_n = w_n^{k-1} - \eta \frac{\partial \|E_n\|^2}{\partial w_n} \quad \text{...}(17a)
\]
\[
C_n = C_n^{k-1} - \eta \frac{\partial \|E_n\|^2}{\partial C_n} \quad \text{...}(17b)
\]

The details of this have been reported elsewhere.6,8-11

### Optimal Mix for High Performance Concrete\(^{12}\)

Sabir\(^{13}\) defines high performance concrete (HPC) as cement concrete, in which each ingredient performs effectively to contribute towards fresh concrete as well as hardened concrete properties. HPC consists of all ingredients of conventional cement concrete (CCC) with chemical admixtures as superplasticizers and mineral admixtures like fly ash, silica fume. Superplasticizers and mineral admixtures are improving the utility of HPC. The performance of cement concrete can be improved from both strength and durability considering impermeability characteristics, which is achieved by adopting lower water-cement ratio and using pozzolanic admixtures such as fly ash and silica fume.

Materials used for HPC: Cement-43 grade OPC, fine aggregate-naturally available sand, coarse aggregate-locally available blue granite, mineral admixtures-silica fume imported from Australia, chemical admixture-superplasticiser (Conplast SP 337), potable water.

In this paper ES approach is used for optimal mix design of HPC mix composed of cement, sand, coarse aggregate, water, silica fume and superplasticizers. It is required to get the optimal mix for the strength greater than 120 MPa and the slump nearly equal to 120 mm. The optimal mix can be designed for any other mix and slump.

### Optimization of High Performance Concrete

HPC usually consists of cement, sand, coarse aggregate, water, silica fumes and superplasticizer. It is required to find the sand (FA)/cement (C), coarse aggregate CA/C, water (W)/C, silica fume (SF) in percentage and superplasticizer (SP) in percentage for optimization of HPC.

A general discrete sizing structural optimization problem for HPC is posed as:

\[
\begin{align*}
\text{Minimize} & \quad F(X) \\
\text{Satisfying} & \quad g_j(x) \leq 0 \quad \ldots j = 1, NC
\end{align*}
\]

where ‘mg’ represents the number of independent design variables and ‘NC’ number of constraints and
\( W = F(X) \) is the objective function, \( g_j \) is the \( j^{th} \) constraint and the inequalities in Eq. (19) are known as side constraints on the design variables.

In case of high performance concrete \( W \) is the cost of the mix per unit weight and is called cost function given by

\[
W = \frac{\sum_{i=1}^{i_{mg}} \text{Cost}_i \times X_i}{\sum_{i=1}^{i_{mg}} X_i} \quad \ldots(20)
\]

In addition to the side constraints, the following strength and the slump constraints are considered in this study. They include

1. The strength of concrete (28 days cube strength) should not be less than the required strength given by

\[
f > f_{all} \quad \ldots(21)
\]

and the constraint equation is written as

\[
C_1 = (1 - f / f_{all}) \quad \text{if } f < f_{all} = 0 \quad \text{if } f \geq f_{all} \quad \ldots(22)
\]

2. Workability should not exceed the 1.1 times the allowable slump given by

\[
\text{slump} < 1.1 \text{ slump}_{all}\quad \ldots(23)
\]

and the constraint equation is written as

\[
C_2 = \begin{cases} \frac{\text{slump}}{1.1 \text{ slump}_{all}} - 1 & \text{if } \text{slump} > 1.1 \text{ slump}_{all} \\ 0 & \text{if } \text{slump} \leq \text{slump}_{all} \end{cases} \quad \ldots(24)
\]

and workability should not be less than 0.9 allowable slump given by

\[
\text{slump} > 0.9 \text{ slump}_{all} \quad \ldots(25)
\]

and the constraint equation is written as

\[
C_2 = \begin{cases} 1 - \frac{\text{slump}}{0.9 \text{ slump}_{all}} & \text{if } \text{slump} < 0.9 \text{ slump}_{all} \\ 0 & \text{if } \text{slump} \geq 0.9 \text{ slump}_{all} \end{cases} \quad \ldots(26)
\]

**Transformation of Constrained Optimisation to Unconstrained Optimisation**

GA initially developed, employed penalty function approach both static and dynamic in majority of the cases for treating constrained optimisation problems. In static penalty approach

\[
\varphi_x = f_x, \text{ if constraints are satisfied} \quad \ldots(27a)
\]

\[
\varphi_x = f_x(1 + pC), \text{ otherwise} \quad \ldots(27b)
\]

where \( C \) is the sum of the constraints given by Eqs (22)-(26) and ‘\( p \)’ is the static penalty parameter in which functions and constraints are normalized between 0 and 1. The main advantage of the method is its simplicity. There is no guidance how to choose single penalty parameter ‘\( p \)’. If it is chosen too small, the search will converge to an infeasible solution, otherwise a feasible solution may be located but it would be far from the global optimum. This approach is used in this paper for treating constrained optimisation problem using evolution strategies (ES).

In dynamic penalty approach proposed by Joines and Houck

\[
\varphi = f + (pg)^a C \quad \ldots(28a)
\]

and

\[
C = \sum_{j=1}^{2} C_j^p \quad \ldots(28b)
\]

where \( p \), \( \alpha \), \( \beta \) are constants. A reasonable choice for these parameters proposed by Joines and Houck are 0.5-2 for \( C \) and 1 for the other two and ‘\( g \)’ is the generation number. The second term of Eq. (28a) in the penalty equation takes extreme large values, which makes even the slight violation design not to be selected in subsequent generation. There are many more penalty approaches recently developed as (i) annealing penalty, (ii) adaptive penalty, (iii) co-evolutionary penalty, (iv) death penalty and (v) use of non-dominance. Static penalty is a traditional technique and it is still used in practice since it exploits the information from the infeasible points to guide the search. For the problems considered in this paper “Static Penalty Approach” of Rajeev and Krishnamoorthy is used to obtain the modified objective function in ES as

\[
\Phi_x = F(X)\{1 + pC\} \quad \ldots(29)
\]

\[
C = \sum_{j=1}^{2} C_j \quad \ldots(30)
\]
where $p$, the penalty parameter can be taken for all practical purposes as 10. $p$ is selected from experience after trying various values such as $p = 1, 10, 100$ and so on.

In order to get the equation for strength and slump the (SLNN) proposed by Zhang and Morris is trained for first fourteen data of the Table 1 and tested for the remaining 9 data. The architecture of SLNN shown in Fig. 2, consists of 5 input neurons for S/C, CA/C, W/C, SF, SP and one bias neuron and one neuron in the hidden layer and one neuron in the output layer representing either the strength or slump. Both the inputs and outputs are normalized such that the values lie between 0 and 1. Radial basis function is used for hidden neuron and linear law is used for both input and output neurons. The network is trained for 200,000 iterations with learning rate of 0.6 and gamma value of 0.000001 and once we get the convergence we can find the weights connecting the input neurons to the hidden neuron and the hidden neuron to output neuron. The graph of error rate with respect to ln (iterations) is shown in Fig. 3 and it is observed that monotonic convergence is obtained. The weights of the connectivity of the network are shown in Fig. 4. In the calculation of compressive strength, the minimum, the maximum and the average errors for training patterns are 1.5, 11.2 and 4.48 percentage respectively and for the test patterns 1.59, 11.2 and 4.77% respectively. It is observed for the slump calculations, the minimum, the maximum and the average percentage of errors are 0.4, 12.5 and 3.19% respectively for training data and 0.57, 7.09 and 3.77% for test data respectively. The strength of HPC is given by the equation (after training from SLNN)

\[ f = 8.2628 e^{-1} \]  

\[ \ldots(31a) \]
\[ I_1 = \frac{1}{2.6808^2} [(X_1 + 2.3357)^2 + (X_2 + 1.7148)^2 + (X_3 - 1.5897)^2 + (X_4 - 1.2737)^2 + (X_5 - 1.9704)^2]. \]  

and the slump is given by the equation is given by

\[ sl = 12.2872 e^{-x_2} \]  

\[ I_2 = \frac{1}{1.41657^2} [(X_1 - 1.2086)^2 + (X_2 - 3.0106)^2 + (X_3 - 0.23047)^2 + (X_4 - 0.84016)^2 + (X_5 - 0.71905)^2] \]

The cost function to be minimized is given as

\[ 3.0X_1 + 2X_2 + 0.05X_3 + 13.5X_4 + 5X_5 + 3.8 = F(X) \]  

for minimize

\[ \text{Evolutionary Strategies (ES)} \]

Evolution strategies (ES) were proposed for parameter optimization problems in the seventies by Rechenberg\textsuperscript{17} and Schwefel\textsuperscript{18}. ES imitate biological evolution in nature and have three characteristics that make them differ from other conventional optimization algorithms: (i) in place of the usual deterministic operators, they use randomized operators: mutation, selection as well as recombination; (ii) instead of single design point, they work simultaneously with a population of design points in the space variables; (iii) they can handle continuous, discrete or mixed optimization problems. The second characteristic allows for a natural implementation of ES on parallel computing environments. The ES, however, achieve a high rate of convergence than GA due to their self-adaptation search mechanism and are considered more efficient for solving real world problems. The ES were initially applied for continuous optimization problems, but recently they have also been implemented in discrete and mixed optimization problems.

\[ \text{ES for discrete optimization problems} \]

In engineering practice, the design variables are not continuous because usually the structural parts are constructed with certain variation of their dimensions. Thus design variables can only take values from a predefined discrete set. For the solution of discrete optimization problems, Cai and Thierauf\textsuperscript{19} have proposed a modified algorithm. The basic differences between discrete and continuous variables are focused on the mutation and recombination operators. The mutation operator ensures that each parent \( s_p^{(g)} \) of the current generation ‘g’ produces an offspring \( s_o^{(g)} \) whose genotype is slightly different from that of parent.

\[ s_o^{(g)} = s_p^{(g)} + z^{(g)} \]  

where \( z^{(g)} = [z_1^{(g)}, z_2^{(g)}, ..., z_n^{(g)}]^T \) is a random vector. The mutation operator in the continuous version of ES produces a normally distributed random change vector \( z^{(g)} \). Each component of this vector has small standard deviation value \( \sigma_i \) and zero mean value. As a
result of this, there is a possibility that all components of the parent vector may be changed but usually the changes are small. In the discrete version of ES the random vector \( z^{(g)} \) is properly generated in order to force the offspring vector to move to another set of discrete values. The fact that the difference between any two adjacent values can be relatively large is against the requirement that the variance \( \sigma_i^2 \) should be small. For this reason it is suggested that not all the components of the parent vector but only a few of them (e.g. 1) should be randomly changed in every generation. This means that \((n-1)\) components of the randomly changed vector \( z^{(g)} \) will have zero value. In other words the terms of the vector \( z^{(g)} \) are derived from

\[
z_i^{(g)} = \begin{cases} 
(k+1)\delta_{ai} & \text{for } (k+1) \text{ randomly chosen components} \\
0 & \text{for } (n-k) \text{ other components}
\end{cases}
\]  

where \( \delta_{ai} \) is a small change in the design variable (say 1/10 of design variable). This of course violates the discreteness of the sections and once the final optimal design is arrived at we can make necessary corrections by choosing the available sections.

\( k \) is a random integer, which follows the Poisson distribution

\[
p(\kappa) = \frac{(\gamma)^\kappa}{\kappa!}e^{-\gamma}
\]

\( \gamma \) is the standard deviation as well as the mean value of the random number \( k \). The choice of \( k \) depends on the size of the problem and here half of the total number of design variables is considered. The \( n \) components are selected randomly in every generation from the set of design variables. In both versions of continuous and discrete optimisation of multi-membered ES there are two different types of selection:

- \((\mu+\lambda)\)-ES: The best \( \mu \) individuals are selected from a temporary population of \((\mu+\lambda)\) individuals to form the parents of the next generation.
- \((\mu, \lambda)\)-ES: The \( \mu \) individuals produce \( \lambda \) offspring \((\mu \leq \lambda)\) and the selection process defines a new population of \( \mu \) individuals from the set of \( \lambda \) offspring only.

In the second type, the life of each individual is limited to one generation. This allows the \((\mu, \lambda)\)-ES selection to perform better on dynamic problems where the optimum is not fixed, or on the problems where the objective function is noisy. The next important thing in the optimization procedure is the termination criteria. The optimization procedure terminates the best value of the objective function in the last \( 4^n n^n \mu/\lambda \) generations remains unchanged.

**Steps in ES in structural optimization problems**

There are four main components in the operations of ES: (i) creation of initial pool of designs, (ii) combination of the designs in a pool in order to produce better designs, (iii) mutation for giving a small change for the offspring from the parents and (iv) obtain a new generation of designs.

In the recent studies by Lagaros *et al.* and Papadrakakis *et al.*, it was found that probabilistic search algorithms are computationally efficient even if greater number of analyses are needed to reach the optimum. These analyses are computationally less expensive than in the case of mathematical programming algorithms since they do not need gradient information. Furthermore, probabilistic methodologies were found to converge to global optimum in the due course whereas mathematical programming algorithms may be trapped in local optima. Finally, the natural parallelism inherent in probabilistic search algorithms makes them very attractive for application in parallel computer architectures.

To start with the algorithm, the initial population is created randomly. In HPC there are five inputs such as S/C, CA/C, W/C, SF/C, SP/C. Each input is represented by 4 bits and any one individual population contains 20 bits and there will be \( n \) such individuals. Care is taken that all parents are selected in the vicinity of the best parent. Mating the best parent with all other parents through crossing over operation generates the offspring. Offspring thus generated are mutated and checked if they are in the feasible region. According to \((\mu+\lambda)\) selection scheme, in every generation the values of the objective function of the parent and the offspring vectors are compared and the worst vectors are rejected, while the remaining ones are considered to be the parent vectors of the new generation. On the other hand, according to \((\mu, \lambda)\) selection scheme, only the offspring vectors of each generation are used to produce the new generation. This procedure is repeated until the chosen termination criterion is satisfied. The ES algorithm for structural optimization applications can be stated as follows:
Evolution strategies algorithm
(i) Selection step: Selection of \( S_i \) \((I = 1, 2, \ldots, \mu)\) parent vectors of the design variables.
(ii) Analysis step: Solve: Objective function \( f = f(I = 1, 2, 3, \ldots, \mu) \).
(iii) Constraints check: all parent vectors become feasible.
(iv) Offspring generation: Generate \( S_j \), \((j = 1, 2, \ldots, \lambda)\) offspring vectors of the design variables.
(v) Mutation: The offspring thus generated are mutated to give a small variation from the parents.
(vi) Analysis step: solve: \( f(j = 1, 2, \ldots, \lambda) \)
(vii) Nominal convergence check: Nominal convergence occurs when the mean value of the objective function of the designs of the current population is relatively close to the best design achieved until the current generation. If satisfied according to the current level of violation, continue. Else change \( S_j \) and return to step 4.
(viii) Selection step: Selection of the next generation parents according to \((\mu+\lambda)\) or \((\mu, \lambda)\) selection schemes.
(ix) Convergence check: If satisfied stop, else go to step 4.

Multilevel optimization
In this work, a multilevel optimization approach is implemented and proved to eliminate the effect of a drawback such as trapping in the local optimum. In this approach, an initial optimization named the first level optimization is carried out with \((X_i)_{\text{min}} < X_i < (X_i)_{\text{max}}\) by judiciously assuming the values of minimum and maximum values for the design parameters. After a few iterations, one can get \( X_i \) for near optimum solution. In the second level optimization, we can narrowdown the search space for each variable \( X_i \) as

\[
(X_i)_{\text{min}} < X_i < (X_i)_{\text{max}}
\]

where the minimum value of the design variable of the present iteration is greater than that used in the first level and the maximum value of the design variable must be less than that used in the first level thus narrowing down the search space.

The process is continued in a similar fashion by narrowing down the search space. The proposed approach has two important characteristics: firstly it encourages the optimization process to investigate better solutions in more restricted favourable regions of search space and secondly each level may be interpreted as one step climbing down the hill towards the foot of the hill (minima). During the tests it is observed that two to three levels of optimization are adequate for the convergence to the true optimum. This multi-level optimization procedure encourages the optimization process to investigate better solution in more restricted favourable regions of search space.

The cost function to be minimized is given as

\[
3.0X_1 + 2X_2 + 0.05X_3 + 13.5X_4 + 5X_5
+ 3.8 = F(X)\text{...for minimize.} \quad \text{(37)}
\]

where

\[
X_1 = S/(2.0C); \quad X_2 = CA/(4C); \quad X_3 = W/(0.5C);
X_4 = SF/(30); \quad X_5 = SP/5 \quad \text{(38)}
\]

\(C=1, \quad S=1.4; \quad CA=3.147; \quad W/C=0.456; \quad SF=1.9%\); \(SP=3.87%\) giving the cost of the HPC per kg of concrete (as per the prevailing rates in India). Multilevel optimization is shown in Table 2. Figure 5 shows the variation of cost of optimum mix of HPC with respect to each generation and monotonic convergence is obtained. Table 3 shows the mix design and the cost of HPC per kg of concrete for various strengths and slumps.

<table>
<thead>
<tr>
<th>Level</th>
<th>Max Min obtained</th>
<th>FA</th>
<th>CA</th>
<th>W</th>
<th>SF</th>
<th>SP</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Min</td>
<td>0.5</td>
<td>2</td>
<td>0.2</td>
<td>0</td>
<td>0.5</td>
<td>3.83</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>2</td>
<td>4</td>
<td>0.5</td>
<td>30</td>
<td>5.0</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>Optimum</td>
<td>1.5</td>
<td>2.8</td>
<td>0.45</td>
<td>26</td>
<td>14.5</td>
<td>3.119</td>
</tr>
<tr>
<td>II</td>
<td>Min</td>
<td>0.5</td>
<td>3</td>
<td>0.3</td>
<td>0</td>
<td>3</td>
<td>2.89</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>2</td>
<td>4</td>
<td>0.5</td>
<td>15</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Optimum</td>
<td>1.5</td>
<td>2.8</td>
<td>0.45</td>
<td>14.5</td>
<td>3</td>
<td>2.70</td>
</tr>
<tr>
<td>III</td>
<td>Min</td>
<td>1.0</td>
<td>3.0</td>
<td>0.4</td>
<td>0</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>1.5</td>
<td>3.6</td>
<td>0.5</td>
<td>15</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Optimum</td>
<td>1.4</td>
<td>3.147</td>
<td>0.456</td>
<td>3.87</td>
<td>2.70</td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 5 — Monotonic convergence in ES](image-url)
Multi-criteria optimization

Figure 6 shows the curve between slump and strength of HPC. So far our goal has been to get the optimal mix to satisfy the requirements of strength and slump. In Fig. 6 we consider two objectives—slump and strength—both of which are important in the optimal design. The point A represents the solution, which gives maximum slump and minimum strength and the point B represents the solution, which gives maximum strength and minimum slump. If both objectives (strength and slump) are important goals of design, one cannot really say whether solution A is better than solution B, or vice versa. One solution is better than other in one objective, but is worse in the other. In fact, there exists many such solutions (like many points on the curve drawn in Fig. 5) which also belongs to the Pareto-front set as defined by Deb and one cannot conclude about an absolute hierarchy of solution A, B or any point on the curve in the set without any further information. All these solutions (in the black curve) are known as Pareto-front solutions, which is very informative and guides the designer to choose the proper design. Fig. 7 shows the cost of HPC for different strengths and slumps.

Conclusions

Evolution strategies ES is very effectively used to find the optimal mix for HPC. The technique for handling the constraints in ES is based on the use of penalty functions, which transform the constraint optimization problem into an unconstrained one. Since the optimization procedure involves the analysis many times, computer time is prohibitive. For a practical problem like HPC, there is no method of mix design available to get the strength and slump. SLNN with one single hidden neuron is used to obtain the relationship between strength and slump with other input parameters. The program developed can be used to find the optimal mix of HPC subjected to strength and workability constraints in any geographical location of the world provided one uses the prevailing rates for the constituents of HPC.

Acknowledgements

The author thanks the management and Dr R Rudramoorthy, Principal, PSG College of Technology for providing necessary facilities to carry out this study.
Nomenclature

\[ C^j_i \] = weight of connectivity between \( j \)th input neuron to \( i \)th hidden neuron

\[ E \] = error

\[ F \] = function

\[ H \] = Hessian Matrix

\[ I \] = input

\[ O \] = output

\[ Q \] = output of hidden node

\[ s^p \] = parent

\[ s^o \] = offspring

\[ w \] = weight of synapse connecting hidden neuron to output neuron

\[ W \] = objective function

\[ X \] = variables

\[ Y \] = actual system output

\[ Z \] = random vector

\[ \eta \] = learning rate

\[ \mu, \lambda \] = individuals

References