A Stochastic Connectionist Approach for Global Optimization with Application to Pattern Clustering

G. Phanendra Babu, *Member, IEEE*, M. Narasimha Murty, and S. Sathiya Keerthi

***Abstract*—In this paper, a stochastic connectionist approach is proposed for solving function optimization problems with real-valued parameters. With the assumption of increased processing capability of a node in the connectionist network, we show how a broader class of problems can be solved. As the proposed approach is a stochastic search technique, it avoids getting stuck in local optima. Robustness of the approach is demonstrated on several multi-modal functions with different numbers of variables. Optimization of a well-known partitional clustering criterion, the squared-error criterion (SEC), is formulated as a function optimization problem and is solved using the proposed approach. This approach is used to cluster selected data sets and the results obtained are compared with that of the K-means algorithm and a simulated annealing (SA) approach. The amenability of the connectionist approach to parallelization enables effective use of parallel hardware.**

***Index Terms*—Clustering, connectionist approaches, function optimization, global optimization.**

I. Introduction

CLUSTER analysis is very useful when classification information pertaining to the data is not available. The main aim of cluster analysis is to find pattern associations by forming groups of patterns such that a pattern in a group is more similar to other patterns in the same group when compared to patterns in other groups. Many clustering approaches have been proposed in the literature to suit various requirements. These approaches can broadly be classified into: numerical [3], [43], [60]; symbolic [36]; and knowledge-based clustering approaches [61].

Many of the conventional approaches are numerical clustering approaches which assume that patterns are points in a d-dimensional space and perform clustering by defining a (dis)similarity measure. Symbolic clustering approaches are suitable to cluster patterns or objects that are often represented by qualitative or symbolic features. On the other hand, knowledge-based clustering approaches use high-level knowledge pertaining to a set of problems to perform the clustering task. In these approaches, knowledge is embedded into the approach for solving a class of problems. Recently, fuzzy clustering of data with partial supervision has been addressed in [68]. In this approach, unsupervised learning is performed in the presence of some labeled patterns.

Numerical clustering approaches can further be classified into: hierarchical; partitional; graph theoretic; and fuzzy set-theoretic clustering approaches. Hierarchical approaches build a tree structure or a dendrogram of the given data revealing the pattern associations. On the other hand, partitional approaches partition the data set into nonoverlapping clusters. Graph-theoretic methods employ graph algorithms for dendrogram construction or partitioning the data [3], [43]. Fuzzy clustering approaches make use of fuzzy set theoretic concepts in order to find fuzzy clusters [11], [12]. Since the applicability of hierarchical clustering methods is limited by the data size [43], in general many applications employ partitional/fuzzy clustering approaches. Partitional clustering approaches are associated with criterion functions, which can be either global or local type [43]. A global type criterion function considers the entire data set on the whole in the clustering process, whereas approaches using a local type of criterion function operate on the local characteristics of the data. Partitional approaches that use local type criterion functions are especially popular in image segmentation [31], [39]. All partitional algorithms that use the global type of criterion function optimize it to generate a partition of the data set. A straight forward approach is to examine all possible partitions and select the one that extremizes the criterion. This is practically not possible as the number of possible partitions increases exponentially with the increase in either the number of patterns (N) or the number of clusters (K). The total number of partitions of the data into to K clusters is given by Stirling approximation $M(N,K) = (1/K!)\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}i^N$.

In the case of $K = 2$, the number of partitions is $2^{N-1} - 1$ and this clearly indicates the exponential complexity involved in exhaustive enumeration. In this paper, we consider the optimization of one of the most widely used criterion function squared-error criterion (SEC) function.

This paper is divided as follows. Section II presents the SEC function and describes the search techniques used to optimize it. The function optimization formulation is also discussed in Section II. Earlier work related to the connectionist approaches for function optimization is presented in Section III. The stochastic connectionist approach for solving global optimization problems along with its generalized version is described Section IV. In order to test the robustness of the proposed approach, many standard test functions available in the literature are optimized and the performance is compared with that of the existing methods. The scalability aspect of the proposed approach to support large scale function optimization is also presented

Manuscript received June 21, 1998; revised October 31, 1999. This paper was recommended by Associate Editor W. Pedrycz.

G. P. Babu is with Technology Deployment International, Inc., Santa Clara, CA 95054 USA (e-mail: babu@tdiinc.com).

M. N. Murty is with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India (e-mail: mnm@csa.iisc.ernet.in).

S. S. Keerthi is with the Department of Mechanical and Production Engineering, National University of Singapore, Singapore (e-mail: mpessk@nus.edu.sg).

Publisher Item Identifier S 1083-4419(00)01396-0.

in Section IV. Section V deals with the applicability of the proposed approach to solve the clustering problem. Section VI presents results and comparisons for selected data sets. Conclusions are provided in the last section.

II. Squared-Error Criterion Function

Many criterion functions are variants of the SEC and it finds applications in several fields [22], [37]. Several algorithms have been proposed in the literature to optimize the SEC including K-means and ISODATA algorithms. Gordon and Henderson [35] have formulated the minimization of SEC as a nonlinear programming problem (NPP) in the following way: Let

$Y = \{y_1, \ldots, y_N\}$ be a set of d-dimensional patterns,
$C = \{C_1, \ldots, C_K\}$ be K mutually disjoint clusters,
$\mathcal{O} = \{o_1, \ldots, o_K\}$ be K d-dimensional cluster centers.

Optimization problem is

$$\text{minimize } J(\mathcal{W}, \mathcal{O}) = \sum_{i=1}^{N}\sum_{j=1}^{K} w_{ij}\|y_i - o_j\|^2$$

where

$$\|\cdot\| = \|\cdot\|_2, \quad w_{ij} \in \{0,1\};$$
$$\mathcal{W} = [w_{ij}] \text{ is an } N \times K \text{ pattern association matrix;}$$
$$\sum_{j=1}^{K} w_{ij} = 1, \quad \sum_{i=1}^{N} w_{ij} \geq 1;$$
$$o_j = \frac{\sum_{i=1}^{N} w_{ij} y_i}{\sum_{i=1}^{N} w_{ij}}.$$

Many clustering algorithms have been developed to optimize $J(\cdot,\cdot)$ for a given data set with N patterns and K clusters. Generally used hill climbing techniques, such as the switching technique and K-means algorithm [3], [26] start with an initial partition and gradually improve that partition by optimizing the criterion function. These techniques converge to a locally optimal partition (see [59] for a detailed convergence proof). Many engineering applications [37], [60] require an optimal partition to be obtained as long term investments are involved. Finding an optimal partition requires finding an optimal assignment matrix, $\mathcal{W}^*$, such that $J(\mathcal{W}^*, \cdot) \leq J(\mathcal{W}, \cdot), \forall \mathcal{W} \in \{0,1\}^{NK}$. This is a well defined discrete optimization problem [26] and many attempts have been made in this direction using various search techniques. Direct exhaustive enumeration of partitions for partitioning 100 patterns into five clusters requires examining 10^{67} partitions [26], which precludes its use in practice. Some of the deterministic search techniques include dynamic programming [10], integer programming [56] ,[67], and branch and bound techniques [49]. A deterministic annealing approach has been considered by Rose *et al.* [57] which can be viewed as a sequential version of the mean field annealing approach in neural networks [65]. This approach employs an annealing technique in which the error surface gets smoothed, converging to a better local minimum, but the global optimum is not guaranteed.

Simulated Annealing (SA) [46] and Evolutionary Approaches (EA) [40] which belong to the class of stochastic search approaches have been employed in solving the clustering problem. Klein and Dubes [47], Selim and Alsultan [58], and Brown and Huntley [18] have investigated the applicability of the SA algorithm for pattern clustering by solving the discrete optimization formulation. Babu and Murty [9] have applied SA approach for finding an optimal initial seed values such that the chosen algorithm, K-means, converges to an optimal partition.

Evolutionary methods are population based stochastic optimization techniques that naturally fit for parallelization. The clustering problem has been attempted by many researches using evolutionary methods. Raghavan and Birchard [54] have used GA's to optimize within group sum of variances clustering criterion function. Other attempts using GA's include [13], [38], [44], and [53]. An integrated approach that combines K-means and genetic algorithms has been investigated [7] for optimizing SEC. Another evolutionary approach, called evolutionary programming, has been employed by Fogel [29] to find fuzzy min-max clusters in the data by fitting hyper-rectangles in multidimensional space. Evolution strategies have been used to accomplish both hard and fuzzy c-means clustering tasks [8].

Almost all of these approaches try to solve the combinatorial optimization formulation of the clustering problem except [7]–[9]. One of the drawbacks of SA is that it consumes a lot of computational time and is sequential in the nature of its search. On the other hand, evolutionary approaches are a natural fit for parallelization, but are not suitable for large values of N and K. Connectionist approaches [48], [52] are well suited for clustering by providing a scope for massive parallelization. Parsi *et al.* [52] investigated the applicability of Hopfield neural network for clustering. By its deterministic nature of search, it converges to a local optimal partition. A group update approach in the context of Hopfield model has been proposed to improve convergence speed and has been applied to solve a set partition problems [5].

The function optimization formulation of SEC facilitates identifying optimal cluster centers such that optimal pattern assignment can be obtained. This formulation is given by

$$J(\mathcal{O}) = \sum_{i=1}^{N}\sum_{j=1}^{K} w_{ij}\|y_i - o_j\|^2 \tag{1}$$

where

$$w_{ij} = \begin{cases} 1, & \text{if } \|y_i - o_j\| \leq \|y_i - o_q\|, \quad \forall q \neq j \\ 0, & \text{otherwise.} \end{cases}$$

The above criterion function is a real-valued parameter function and the problem turns out to find $\mathcal{O}^*$, i.e., optimal cluster centers, so that $\mathcal{W}^*$ can be found. Note that any ties, i.e., a pattern equidistant from two or more cluster centers, are resolved by assigning associated pattern to one of the equidistant clusters.

III. Connectionist Approaches for Function Optimization

In optimization theory, finding a global optimal solution for a NPP with range constraints is termed as a global optimization problem. There are a number of methods available in the literature to locate local extrema, but the global optimization problem is very difficult to solve in a finite number of steps. Many approaches have been proposed in the literature to solve global optimization problems [24], [25]. Connectionist approaches belong to a class of parallel approaches for solving difficult optimization problems [1]. To the best of our knowledge, no attempt has been made to solve a general function optimization problem using stochastic connectionist approach which tries to find a global extrema of a given function. Some approaches have been proposed in the literature for optimizing quadratic functions.

The first approach proposed by Chua and Lin [21] attempts to solve a NPP using a network with small circuits interconnected together. Tank and Hopfield [62] proposed the first neural network model for solving a linear programming problems. Other neural network approaches include [16], [27], [45], and [70]. All these approaches are a type of parallel gradient descent methods and do not guarantee a global optimal solution. Even though much of the literature in the optimization field addresses solving functions that are quadratic in nature, in real life, we encounter many nonquadratic, discontinuous and other types of functions. The existing connectionist approaches are inadequate to solve them. In the following, we present the general formulation of the function with constraints.

Function Optimization: We define the function to be optimized along with the constraints imposed on the search space. The constraints define a feasible search space. The general function with constraints is defined as

$$\begin{array}{lll} \text{Minimize} & f(x) & \forall x \in S \\ \text{Subject to} & h_i(x) \leq 0 & 1 \leq i \leq q \\ & h_j(x) = 0 & q+1 \leq j \leq m \end{array}$$

where $h_1, \ldots, h_q$ are inequality constraints and $h_{q+1}, \ldots, h_m$ are equality constraints. These constraints $h_1, \ldots, h_m$ define a feasible region $\Re_c^n$. Note that x is a vector of size n. In the current context, for obtaining extrema x^* we need to transform this constrained optimization problem into an unconstrained optimization problem using penalty methods. In the penalty function approach, the function is penalized when a solution violates any of the constraints. This forces the solution to lie in the region of interest, $\Re_c^n$. This can be done by minimizing a new function, $f_c()$, given by $f_c(x) = f(x) + \lambda_1(h_1(x)) + \cdots + \lambda_m(h_m(x))$ where λ_i is the ith penalty function which gives no penalty or zero value if the constraint h_i satisfies its equality or inequality. If there is a constraint violation, it returns a positive penalty in order to restrict the scope of x to $\Re_c^n$. In the extreme case, one can consider infinite penalties. In principle, using infinite penalties, minimizing $f_c()$ is equivalent to minimizing $f()$ and the global extrema obtained for $f_c()$ will be the global extrema for $f()$. In practice, incorporation of this type of infinite penalty functions may not be required. Depending on the problem at hand, one can use either a linear or quadratic penalty function expressions as shown at the bottom of the page, where $L(\cdot)$ represents a linear function, $Q(\cdot)$ represents a quadratic function, and coefficients β_i denote positive constants [64]. Some special constraints that are very useful in the search process are range constraints. These constraints define the possible limiting values of variables. Let the bound constraints be $\mu \leq x \leq \nu$ $(\mu_i \leq x_i \leq \nu_i, i = 1, \ldots, n)$, where vectors μ and ν define lower and upper bounds of variables. These constraints can also be incorporated in the function using penalty methods. These constraints can be used explicitly to generate a feasible search point. To ensure that the generated point is a feasible point [16], the following function $g()$ is defined:

$$g(y_i) = \begin{cases} y_i, & \text{if } \mu_i \leq y_i \leq \nu_i \\ \mu_i & \text{if } y_i < \mu_i, \quad \forall i \\ \nu_i, & \text{if } y_i > \nu_i, \quad \forall i. \end{cases} .$$

Now the function turns out to be $f_c(x) = f(g(x)) + \lambda_1(h_1(g(x))) + \cdots + \lambda_m(h_m(g(x)))$. By explicitly forcing x to lie in $\Re_c^n$, the search will be relatively faster and it also reduces the number of penalty functions. Bound constraints can easily be handled by the connectionist approach using the function $g()$. In the next section, we present a connectionist approach for finding a global extremum of a general function with range constraints.

IV. Stochastic Connectionist Approach for Global Optimization

In this section, we present a connectionist approach for general function optimization with range constraints and then investigate the applicability of the proposed approach to solve the clustering problem. The advantages of the proposed approach over the existing connectionist approaches for optimizing NPP are as follows:

1) global optimum is guaranteed asymptotically (restricted case);
2) no constraints on the type of the function being optimized, such as continuity and existence of derivatives;
3) highly parallelizable;
4) scalable to support large scale function optimization.

We are not adhering to the notion of a node as a simple thresholding unit and assume that it has processing capability to support the specific requirements. The idea of a node being a simple processor is quite convincing in the case of learning and representation tasks, where even the failure of a few nodes in the network may not degrade the performance of the network. However, in the context of optimization task, a node failure during the optimization process leads to inappropriate solutions. In fact, this entails an investigation for a compromise

$$\lambda_i(u_i), = \begin{cases} 0, & \text{if } u_i \leq 0 \\ \beta_i L(u_i) \text{ or } \frac{1}{2}\beta_i Q(u_i), & \text{if } u_i > 0 \end{cases} \quad 1 \leq i \leq q$$

$$\lambda_i(u_i), = \begin{cases} 0 & \text{if } u_i = 0, \\ \beta_i |L(u_i)| \text{ or } \frac{1}{2}\beta_i Q(u_i), & \text{if } u_i \neq 0 \end{cases} \quad q+1 \leq i \leq m$$

between the node capability and the parallelization achievable. With this assumption of increased processing capability, we show how a broader range of problems can be solved using connectionist approaches. The approach investigated here can be viewed as an extension of the Boltzmann Machine that is used to solve combinatorial problems [1], [2]. Earlier investigations in this direction include employing the SA algorithm to solve continuous valued parameter optimization problems [14], [15], [23], [66]. We first present the network architecture followed by the node update rules.

The number of nodes is equal to the number of variables to be optimized. Network is fully connected, i.e., each node is connected to every other node and can be viewed as a graph $G = \{V, E\}$, where V is a set of nodes, $|V| = n$, and E represents a set of connections associated with nodes, $\{v_i, v_j\} \in E$. Each connection is associated with a weight called connection weight, w_{ij}. These weights are assumed to be symmetric, i.e., $w_{ij} = w_{ji}$. Each node outputs a real value. All weights are equated to 1 ($w_{ij} = 1$). Connections are used to get the values of other variables from other nodes.

Let the function to be optimized be $f()$. The state of the network is characterized by a vector comprised of output values of nodes. The state of the network is also called the configuration of the network. It is represented by the concatenation of all output values of the nodes as given by $x = x_1 x_2 \ldots x_n$. The value $f^t_{|x_i=a}$ denotes the function value computed at node i, whose output value is a, using other variable values: $f^t_{|x_i=a} = f(x^t_1, \ldots, x^t_{i-1}, a, x^t_{i+1}, \ldots, x^t_n)$. Here, the parameter t represents the time step at which node i is updated. Initially the network is started with some random values that fall in the feasible region. Range constraints can be used explicitly to search in the feasible search space instead of including them into the objective function. Each node generates a neighborhood value, x_{i_1}, by adding Gaussian noise to the current output value, x_{i_0}. This value, x_{i_1}, is computed as $x^t_{i_1} = g(x^t_{i_0} + \mathcal{N}(0, \sigma^2_i))$, where $\mathcal{N}(0, \sigma^2_i)$ generates a normal random value with σ^2_i variance, t denotes the current time step, and function $g()$ checks for the violation of bound constraints. The change in the function value with the change in variable, x_i, value is given by $\Delta f^t_i = f^t_{i_{|x_i = x_{i_1}}} - f^t_{i_{|x_i = x_{i_0}}}$. The new value is set as the current output according to the following update rule:

$$x^{t+1}_{i_0} = \begin{cases} x^t_{i_1}, & \text{if } \Delta f^t_i < 0 \\ x^t_{i_1}, & \text{if } \exp\left(-\frac{\Delta f_i}{T}\right) > \text{rand}(0,1) \\ x^t_{i_0}, & \text{otherwise.} \end{cases}$$

Here, the function $\text{rand}(0, 1)$ generates a uniform random value in the range $[0, 1]$ and the parameter T is called *temperature*. A node is updated at a time step t with a firing probability of p_f. The average number of nodes that are updated at a time step is $n * p_f$ where n is the number of nodes in the network. The temperature, T, controls the stochasticity in node update rule. We can directly see the relevance of this approach to the Boltzmann machine. This update rule is a direct extension of node update rule [2] in the Boltzmann machine for solving discrete optimization problems. Initially, network is started at a high temperature and then is allowed to reach a thermal equilibrium state characterized by the steady state distribution over the solutions in the search space. At thermal equilibrium, the probability that the network is in a state x is given by the Boltzmann distribution [34]

$$\pi_T(x) = \frac{1}{Z_T} \exp\left(-\frac{f(x)}{T}\right) \quad (2)$$

where $Z_T = \int_{\Re^n} \exp(-(f(x)/T))$. As the temperature $T \to 0$, the distribution π_T approximates to global minima. At each temperature and variance, σ^2, the network is allowed to reach the thermal equilibrium. The temperature schedule, i.e., annealing schedule, can be logarithamic [33] and the variance for generating neighborhood values can be kept constant at all temperatures or can be decreased depending on the function to be optimized. Initial variance can be set using the bounds of each variable, i.e., $\sigma^2_i \propto c_i[\nu(i) - \mu(i)]$. From the normal distribution property, we have $P(|x - 2\sigma \leq X \leq x + 2\sigma|) \approx 0.95$. So, approximately 95% of samples generated by $N(x, \sigma^2)$ fall in the range $[-2\sigma, 2\sigma]$. Setting $\sigma = (\nu_i - \mu_i)/4$ will allow 95% of the proposed moves to fall within the bounds of the variable i.

After the network reaches thermal equilibrium at a fixed temperature, both temperature and variance are decreased by multiplying with α_T (cooling rate) and α_σ, respectively. This process is performed until the temperature is sufficiently decreased. Theoretically speaking, only in sequential mode of network execution will this approach guarantee the global optimum asymptotically. However, even in both synchronous and asynchronous parallel modes of network execution, the network converges to a global optimum quite often. In fact, we can exploit the parallelism offered by connectionist approaches only in these two modes of network execution. In order to test the performance of the proposed approach, several standard test functions are optimized using the asynchronous parallel mode of execution.

The problem associated with this approach is that it gets entrapped in a deep local minimum. In order to avoid this problem, the network has to be started at high temperature and should be annealed down slowly. To provide robust search, we generalize this approach by allowing each node to generate more than one value of the corresponding variable to select a new value for updating the node. This reduces the chances of getting stuck in a deep local minimum. Even at zero temperature limit, each node tries to search for a better solution at which the function value is better.

A. *Generalized Stochastic Connectionist Approach*

Each node generates a set of neighborhood points by adding Gaussian noise to the current output value. Let the generated points at node i be $x_{i_1}, x_{i_2}, \ldots, x_{i_W}$ and the current output be x_{i_0}

$$\begin{aligned} x^t_{i_j} &= g\left(x^t_{i_0} + \mathcal{N}\left(0, \sigma^2_i\right)\right), \quad 1 \leq j \leq W \\ f^t_{i_j} &= f^t_{|x_i = x_{i_j}}, \quad 1 \leq j \leq W. \end{aligned}$$

Node output is updated using the following rules:

$$f^t_{i_l} = \min_j \{f^t_{i_j}\}, \quad \forall j, \quad \text{if } \left(f^t_{i_l} < f^t_{i_0}\right) \text{ set } x_{i_0} = x_{i_l}$$

```
$T_0$ and $T_f$ are the initial and final temperatures, and $\alpha_T$ is the cooling rate
while ($T_0 > T_f$) do
begin
  $r = 0$;
  while( $r < Max\_Ite$) do /* $Max\_Ite$ is the maximum number of iterations */
  begin
    for each node $i$ do
    begin
      if ($p_f > rand(0,1)$) do /* $p_f$ is the probability of firing a node*/
        generate $x_{i_j} = g(\mathcal{N}(x_{i_0}, \sigma_i^2)), 1 \le j \le W$;
        compute $f_{i_j} = f_{|x_i = x_{i_j}}$, $1 \le j \le W$; $f_{i_l} = \min_j\{f_{i_j}\}$;
        if ($f_{i_l} < f_{i_0}$) $x_i = x_{i_l}, f_i = f_{i_l}$;
        else $\mathcal{P}(l) = \frac{\exp(-f_{i_l}/T_0)}{\sum_{q=0}^{W} \exp(-f_{i_q}/T_0)}$
            set $x_i = x_{i_l}, f_i = f_{i_l}$ with probability $\mathcal{P}(l)$;
    end
    $r = r + 1$
  end
  $T_0 = \alpha_T * T_0, \sigma_i^2 = \alpha_\sigma * \sigma_i^2, 1 \le i \le K$;
end
```

Fig. 1. Sequential algorithm.

else select an output value using $\mathcal{P}$, a probability vector of size W where

$$\mathcal{P}(l) = \frac{\exp\left(-f_{i_l}^t/T\right)}{\sum_{k=0}^{W}\exp\left(-f_{i_k}^t/T\right)}, \quad 0 \le l \le W.$$

The approach discussed earlier is a special case of this method, when $W = 1$. The probability vector maintains the probabilities of transition from the current state to each of the other W states. For example, consider, $W = 3$ and $\mathcal{P} = \{0.3, 0.2, 0.2, 0.3\}$. The probability of remaining in the same state is 0.3, and the probabilities of accepting the states 1, 2, and 3 are 0.2, 0.2, and 0.3, respectively. One way of selecting a state is to generate a random number, r, in the range [0–1], and select a state, l, such that the sum of all probabilities $\sum_{i=0}^{l}\mathcal{P}(l) \ge r$. In the limit temperature $(T \rightarrow \infty)$, $\mathcal{P}(l) \rightarrow 1/(W+1), \forall l$. As temperature and variance tends to zero, the probability vector becomes $\{1.0, \ldots, 0.0\}$. Algorithm for the generalized stochastic connectionist network is presented in Fig. 1.

This approach aims at avoiding the deep local minima problem. We analyze the advantages of setting $W > 1$. The problem with deep local minima will come when the temperature is near zero and variance is small. At this temperature, each node tries to locate a neighborhood point better than the current local minimum. Consider a one-dimensional function possessing deep local minima as shown in Fig. 2, and let the current solution be v_i. The only way of coming out of this local minimum is to search for a point in the range $[a, b]$. For $W = 1$, the probability of obtaining a neighborhood point in the range $[a, b]$, is $P_{ab_{(W=1)}} = |\int_a^b (1/\sqrt{2\pi}\sigma)\exp(-((v_i - x)^2/2\sigma^2))\,dx|$.

By generating W samples, the probability of generating at least one point in the range $[a, b]$ is $(1 - (1 - P_{ab_{(W=1)}})^W)$.

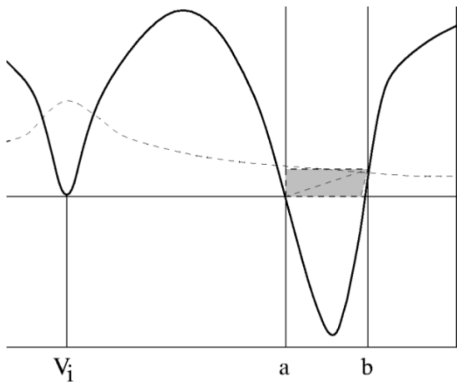


Fig. 2. A function (thick line) along with normal distribution function (dotted line) with mean v_i.

This clearly indicates that this probability increases with the increase in W value and approaches 1.0 as $W \rightarrow \infty$. The value of W should be chosen carefully. Large values of W increases the number of function evaluations, whereas small values of W often lead to local optima entrapment. In our experiments, we observed that setting $W = 5$ for general function optimization gives an excellent rate of progress with a high success rate (≥ 0.95) and a small number of function evaluations. It was observed that W can be set to small values for the functions without deep local minima, but has to be set to higher values for functions with deep local minima. In this paper, we compare the performance of the proposed approach for different values of W (1 and 5). The success rate is defined as a ratio of the number of times the global optimal solution is obtained to the total number of trials. In the Appendix I, we provide asymptotic convergence proof assuming that variance is constant and $W = 1$. The possibility of extending this proof to work for fixed values of W (>1)

TABLE I
COMPARISON OF AVERAGE NUMBER OF FUNCTION EVALUATIONS (AVERAGE TIME OF EXECUTION) TAKEN BY DIFFERENT APPROACHES

Function	GP	RCOS	H3	H6	S5	S7	S10
	(F13)	(F2)	(F11a)	(F11b)	(F12a)	(F12b)	(F12c)
Bremmermann	210L[0.5]	250L[1]	505L[2]	L	3401L[1]	1700L[8]	500L[17]
Mod. Brem.	300[0.7]	160[0.5]	420L[2]	515[3]	375L[1.5]	405L[1.5]	336L[2]
Zilinskas		5129[80]	8641[175]		L	12121L[282]	8892L[214]
Torn	2499[4]	1558[4]	2584[8]	3447[16]	3679[10]	3606[13]	3874[15]
Gomulka/Torn					6654[17]	6084[15]	6144[20]
Gomulka/V.M	1495[2]	1318[3]	6766[17]	11125[48]	7085[19]	6684[23]	7352[23]
Price	2500[3]	1800[4]	2400[8]	7600[46]	3800[14]	4900[20]	4400[20]
Fagiuoli	158[0.7]	1600[5]	513[5]	2916[100]	2514[7]	2519[9]	2518[13]
INTEROPT	6375[6.3]	4172[10.4]	1113[1.4]	17262[46.4]	3700[11(0.4)]	2426[5.8(.6)]	3463[8.6(.5)]
Vanderbilt/ Louie	557[1(1)]	–	1224[4(1)]	1914[12(.62)]	3910[16(.54)]	3421[15(.64)]	3078[15(.81)]
Connectionist approach (W=1)	321[0.2] (1)	782[.51] (.96)	1097[2.2] (0.95)	1276[3.3] (.81)	2569[4] (.48)	2571[5] (.59)	2437[6.1] (.72)
Connectionist approach (W=5)	772[0.5] (.95)	605[0.4] (1)	1436[2.9] (1)	1101[2.85] (.88)	9062[14] (.96)	4315[8.5] (.98)	5010[12.5] (1)

L: indicates that a local optimum is obtained by the approach.
Values in brackets () indicate the success rate. Some approaches don't have these values as they were not published or not relevant in the context of the associated approach.
Values in square brackets [] indicate average time of execution taken by different approaches

is also discussed. In the following section, we present results related to the optimization of the standard test functions along with relevant comparisons.

B. Optimization of Standard Test Functions

In order to establish the characteristics of a global optimization technique, researchers have used a variety of test functions. The choice of test functions is crucial as each method has its own merits and drawbacks in solving different types of functions. A set of seven widely used standard test functions are selected to test and compare the performance of global optimization approaches. We use these functions along with some other functions taken from Renpu Ge and Qin [32], and Breiman and Cutler [17]. Results are compared with those obtained with other approaches. Generally, global optimization approaches are compared based on the number of objective function evaluations and the amount of time taken for optimization. The execution time is defined in terms of standard time units [17]. Many methods proposed do not consider the scalability of the approach to solve large scale function optimization problems. We argue that the scalability of an approach is also an important criterion in establishing the efficacy of an approach over others. Results obtained with other approaches have been taken from [24]. The results obtained with SA are taken from [14] and [66].

The number of variables in these standard functions are in the range [2–6]. The functions are referred using the following abbreviations:

1) Branin function (RCOS);
2) Goldstein and Price (GP) function;
3) Hartman's family of functions (H3, H6);
4) Shekel's family of functions (S5, S7, S10).

Different test functions have been selected from [32]. A function number or a function abbreviation is used to refer to the corresponding function. References to these functions are provided in Appendix II.

TABLE II
RESULTS AND CONTROL PARAMETERS FOR THE SELECTED FUNCTIONS

FN	N	R	M_I	N_f	Temperature		Variances		SR
			$x10$		T_0	T_f	σ_0^2	σ_f^2	
1	2	10	2	398	100	5x10^{-2}	2.0	5x10^{-3}	1.00
2	2	10	2	605	1	10^{-5}	0.75	5x10^{-3}	1.00
3	2	5	5	540	1	10^{-5}	0.75	10^{-3}	0.99
4	2	10	5	867	1	10^{-4}	4.00	5x10^{-3}	1.00
5	2	10	5	446	10	10^{-4}	0.75	10^{-2}	1.00
6	2	100	5	13410	100	10^{-2}	0.6	5x10^{-3}	1.00
7	2	10	5	638	0.5	10^{-5}	0.75	5x10^{-3}	1.00
8	2	10	5	2079	10	10^{-5}	2.0	5x10^{-3}	1.00
9	2	10	20	6915	100	10^{-3}	2.0	10^{-3}	0.98
10	2	10	20	6649	100	10^{-3}	2.0	10^{-3}	0.98
11	3	30	20	1436	0.5	10^{-5}	0.3	10^{-3}	1.00
11a	6	30	20	1101	1	10^{-5}	0.3	10^{-3}	0.88
12	4	10	20	9062	0.5	10^{-5}	0.3	10^{-5}	0.96
12a	4	10	20	4315	0.5	10^{-5}	0.3	10^{-5}	0.98
12b	4	10	20	5010	0.5	10^{-5}	0.3	10^{-5}	1.00
13	2	10	10	722	10^4	10^{-2}	0.5	10^{-4}	0.95

FN and SR stand for function number, and success rate
N_f stands for the average number of function evaluations
$M_I(Max\_Ite)$ stands for the number of inner loop iterations
$R = \frac{\log T_f - \log T_0}{\log \alpha}$ stands for the number of outer loop iterations

As the proposed approach belongs to a class of probabilistic search techniques, we performed 100 trials for each of the selected functions on a PC 386. Each trial has been started with a different starting point that was selected randomly with uniform distribution in the search space. The average number of function evaluations along with the control parameters are presented. The network was run in the asynchronous parallel mode, parallel execution, and at any time instant half of the randomly, with uniform distribution, selected nodes are allowed to change their output values. All function evaluations computed in parallel are considered as a single function evaluation. The program was run until a global/local optimum is obtained with a precision of 10^{-4}. The initial temperature T_0 is set greater than or equal to the variance of function values obtained for some randomly selected solutions. The cooling rate α_T is set to 0.95. The average number of function evaluations required for the seven functions, F2, F13, F11a, F11b, F12a, F12b, and F12c are compared with the earlier reported results obtained by other procedures. Table I presents a comparison of the number of function evaluations for both $W = 1$ and $W = 5$ cases.

It can be observed that in the case of $W = 1$, the success rate is small for functions H6 (F11b), S5 (F12a), S7 (F12b) and S10 (F12c) as they contain deep local minima. Even though the number of function evaluations is less compared to that of the other methods, the problems with deep local minima are severe. For $W = 5$, high success rates can be observed. The average time of execution taken by each method for each of these functions is presented in Table I. These values are provided in square ([]) brackets.

The time is measured in standard units and one unit of time is equal to the time consumed for 1000 evaluations of the S5 function on the PC 386. It can be observed that in most of the cases, the time taken by the proposed approach is less compared to that of the other strategies. These results bring out the effectiveness of the connectionist approach over conventional deterministic or random strategies. The results of all functions, F1–F13, along with control parameters, are listed in Table II.

From these results, we empirically establish the robustness of the proposed connectionist approach in terms of its ability to find global optima, execution time taken and the success rate obtained. We test the scalability of the connectionist approach with respect to the increased number of variables. For testing scalability, we consider functions F14 and F15. The number of variables is increased from 25 to 400 in steps of 25 and the network was run in both asynchronous (Sequential Execution) and asynchronous parallel (Parallel Execution) modes by setting $W = 1$ as these functions do not have deep local minima. In asynchronous mode a node is selected randomly with uniform distribution and is updated in a time step, and in asynchronous parallel mode a set of nodes is selected randomly with uniform distribution and are updated in a time step. Figs. 3 and 4 present the average number of function evaluations, J, and the average time of execution for different values of n for functions F14 and F15, respectively. It can be observed that the number of function evaluations remain almost constant even as the number of variables increases. For function F14 with $n = 400$, the number of local minima is 2^{1950}. It is very difficult for conventional approaches to optimize these type of functions because of several local minima. From these experimental results, it can be inferred that the proposed connectionist approach always gets global optimum, and has an excellent scalability property.

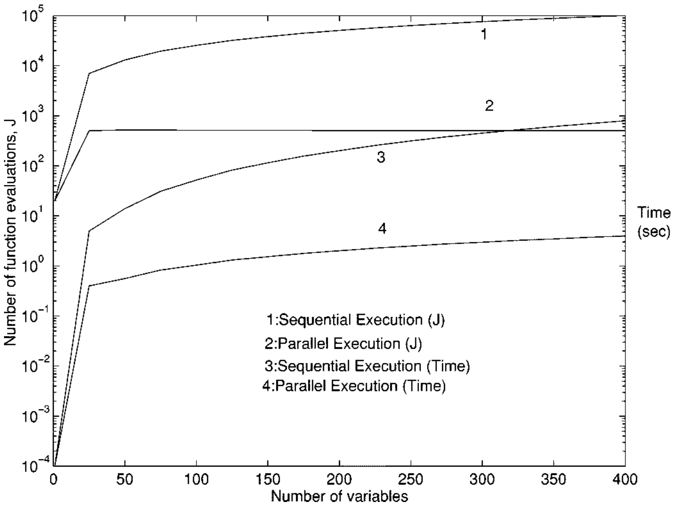


Fig. 3. Number of function evaluations, J, and computation time, Time, (in seconds) required to obtain global optimum for function F14 with different number of variables for both sequential and parallel network executions.

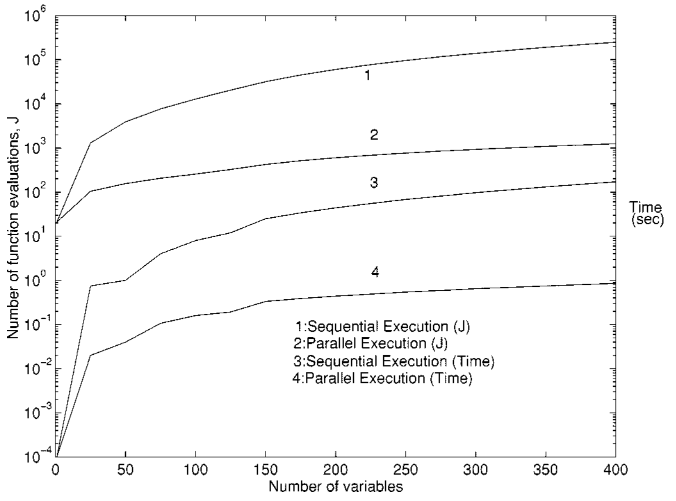


Fig. 4. Number of function evaluations, J, and computation time, Time, (in seconds) required to obtain global optimum for function F15 with different number of variables for both sequential and parallel network executions.

This approach is employed to solve the clustering problem which is formulated as a function optimization problem. In the next section, we discuss the network architecture for solving clustering problem and the use of gradient knowledge in the search process.

V. Clustering with the Connectionist Approach

We use the proposed connectionist approach to locate optimal cluster centers, thus solving real-parameter function optimization problem.

In the proposed network architecture, each node corresponds to a d dimensional cluster center. Variable o in (1) maps to x_i of a node in the network. This indicates that the output of each node is a real-valued vector. The vectorized node output values can be

```
$T_0$ and $T_f$ are the initial and final temperatures, and
$\alpha_T$ is the cooling rate
while ($T_0 > T_f$) do
begin
  $r = 0$;
  while( $r < Max\_Ite$) do
  begin
    for each node $i, (1 \leq i \leq K)$, do (in parallel), $1 \leq i \leq K$
    begin
      if ($p_f > rand(0,1)$) do
        compute new cluster center $o_i'$ by assigning the nearest patterns to the cluster $i$
        compute gradient $\bar{b}_i = (o_i' - o_i), x_{i0} = o_i$;
        generate $x_{ij}(l) = g(\mathcal{N}(x_{i0}(l), \frac{b_i(l)}{||b_i||}\sigma^2)), 1 \leq j \leq W, 1 \leq l \leq d$;
        compute $f_{ij} = J_{|o_i = x_{ij}}, 1 \leq j \leq W$; $f_{il} = \min_j\{f_{ij}\}$;
        if ($f_{il} < f_{i0}$) $o_i = x_{il}, J_{|o_i} = f_{il}$;
        else $\mathcal{P}(l) = \frac{\exp(-f_{il}/T_0)}{\sum_{q=0}^{W} \exp(-f_{iq}/T_0)}$
            set $o_i = x_{il}, J_{f_{|o_i}} = f_{il}$ with probability $\mathcal{P}(l)$,
    end
    $r = r + 1$;
  end
  $T_0 = \alpha_T * T_0, \sigma^2 = \alpha_\sigma * \sigma^2$;
end
```

Fig. 5. Sequential algorithm for clustering.

communicated to other nodes with the available network communication facilities or with the help of a specialized hardware to support parallel transfer of vector output. The values of variables can be mapped onto $[0, 1]$ range to facilitate analog circuit realization [16]. The node update rule discussed earlier has to be extended to accommodate vector valued states. The neighborhood values are generated using $x_{i_j}(l) = g(x_{i_0}(l) + \mathcal{N}(0, \sigma_l^2))$, $1 \leq l \leq d, 1 \leq j \leq W$ where σ_l^2 is the variance associated with the lth component. Note that x_{i_j} is a d-dimensional vector and $x_{i_j}(l)$ is the lth component of the d-dimensional vector. It is evident that stochastic search is performed by sampling the neighborhood. This process consumes excessive computational time which increases with the increase in the number of dimensions. We can make use of knowledge about the gradient in order to increase the speed of search, and the stochastic search is done using the gradient direction. Assume that the gradient at x_{i_0} is $\vec{b}_{i_0} = -(\partial J/\partial x_{i_0})$. The neighborhood points generated are given by $x_{i_j} = g(x_{i_0} + (\vec{b}_{i_o}/||\vec{b}_{i_0}||)B)$, where $B = \mathcal{N}(0, \sigma^2)$. The variance, σ^2, is estimated using the normal distribution property $P(x - 2\sigma \leq X \leq x + 2\sigma) \approx 0.95$, so $\sigma = ||\vec{b}_{i_0}||/4$.

In the case of the clustering problem, we can obtain the partial derivative with respect to a cluster center. The gradient direction $\vec{b}_i$ at a node i can easily be computed by assigning patterns to the nearest cluster centers and computing new cluster center $\vec{b}_{i_0}'$. The gradient now is $\vec{b}_{i0} = \vec{b}_{i_0}' - x_{i_0}$. The network is initialized with random cluster centers and is annealed down slowly by starting with a high initial temperature T_0, and an initial variance σ_0^2. We use an exponentially decreasing temperature schedule. The temperature and variance are decreased by multiplying with

```
$\mathcal{S}_0$ is the initial solution,
$\mathcal{E}_{S_0}$ is the function value (squared-error) of the solution $\mathcal{S}_0$
$T_0$ and $T_f$ are initial and final temperatures
while ($T_0 > T_f$)
begin
  $l = 0$;
  while ($l < Max\_Ite$)
  begin
    generate neighborhood solution $\mathcal{S}_1$ from $\mathcal{S}_0$
    compute $\mathcal{E}_{S_1}$ value
    if $\mathcal{E}_{S_1} > \mathcal{E}_{S_0}$
    begin
      if (exp ($-\frac{(\mathcal{E}_{S_1} - \mathcal{E}_{S_0})}{T}$) $> rand(0,1)$)
        $\mathcal{S}_0 = \mathcal{S}_1; \mathcal{E}_{S_0} = \mathcal{E}_{S_1}$;
    end
    else $\mathcal{S}_0 = \mathcal{S}_1; \mathcal{E}_{S_0} = \mathcal{E}_{S_1}$;
    $l = l + 1$;
  end
  $T_0 = \alpha * T_0$;
end
```

Fig. 6. Simulated annealing algorithm.

TABLE III
RESULTS OBTAINED WITH K-MEANS AND SA APPROACHES FOR GTD

	K-Means						SA					
K	MiJ	MaJ	AvJ	SD	SR	AvT	MiJ	MaJ	AvJ	SD	SR	AvT
2	121425.8	2296520.5	124660.5	18392.85	97	0.005	121425.8	121425.8	121425.8	0.0	100	76.9
3	77008.6	91126.9	83064.2	662.62	19	0.007	77008.6	77008.6	77008.6	0.0	100	77.9
4	49600.6	84030.4	58298.9	12398.7	10	0.006	49600.6	49600.6	49600.6	0.0	100	78.9
5	38716.0	78354.2	42348.2	6914.2	11	0.008	38716.0	38716.0	38716.0	0.0	100	79.8
6	30535.4	53354.2	34554.6	3379.8	3	0.009	30535.4	32531.5	31072.3	835.6	50	80.7
7	24432.6	37766.3	29477.5	2843.7	2	0.011	24432.6	24748.5	24464.2	94.7	40	81.6
8	21835.8	37470.8	26480.5	2782.3	0	0.012	21483.0	22063.2	21720.5	251.7	50	82.4
9	18970.5	29337.4	23172.9	2312.5	0	0.013	18550.4	19323.3	19076.3	286.8	30	83.8
10	16843.3	32085.8	21150.8	2673.8	0	0.015	16307.9	17600.8	16828.6	447.6	10	84.2

TABLE IV
RESULTS OBTAINED WITH SCA FOR GTD

	$W = 1$						$W = 3$					
K	MiJ	MaJ	AvJ	SD	SR	AvT	MiJ	MaJ	AvJ	SD	SR	AvT
2	121425.8	121425.8	121425.8	0.0	100	0.030	121425.8	121425.8	121425.8	0.0	100	0.11
3	77008.6	77008.6	77008.6	0.0	100	0.13	77008.6	77008.6	77008.6	0.0	100	0.36
4	49600.6	49600.6	49600.6	0.0	100	0.27	49600.6	49600.6	49600.6	0.0	100	0.80
5	38716.0	38716.0	38716.0	0.0	100	0.42	38716.0	38716.0	38716.0	0.0	100	1.20
6	30535.4	30535.4	30535.4	0.0	100	0.51	30535.4	30535.4	30535.4	0.0	100	1.55
7	24432.6	24432.6	24432.6	0.0	100	2.34	24432.6	24432.6	24432.6	0.0	100	4.35
8	21483.0	21507.1	21487.1	8.37	80	3.57	21483.0	21483.0	21483.0	0.0	100	11.37
9	18550.4	19053.3	18616.9	149.58	60	5.75	18550.4	18550.4	18550.4	0.0	100	16.53
10	16307.9	16578.8	16441.8	102.56	30	8.84	16307.9	16578.8	16441.8	102.56	30	23.75

α_T (<1) and α_σ (<1), respectively. The sequential algorithm for clustering is presented in Fig. 5.

Computational Complexity: In the sequential implementation of the algorithm, the complexity of the algorithm depends on the values of W, N, K, d, *Max_Ite*, T_0, T_f, and α_T. Let R $(= {}^{\alpha_T}\sqrt{T_f/T_0})$ be the number of external loop iterations. Here the time complexity of each function evaluation is given by $O(NKd)$. So the expected run complexity of the algorithm is $O(p_f WNKd)$ where p_f is firing probability of node. Note that the value of J for $x_{io} = o_i$ is the same for all nodes. So once it is computed, all nodes can make use of it. The expected run time complexity of the parallel algorithm is presented later.

VI. EXPERIMENTAL RESULTS

Several data sets have been extensively tested using the proposed approach for $W = 1$ and 3. In most of the cases (near) optimal solutions are obtained. All simulations are carried out on a PC/AT 386 machine. The network was run in asynchronous parallel mode by allowing randomly selected half of the nodes in the network to update their output values in each iteration. There are several ways to select nodes randomly. In our experimentation, we allowed each node to fire with a probability, p_f, set to 0.5. If the number of nodes selected for firing using p_f exceed half of the total number of nodes, we removed some of the nodes randomly from the list to ensure that total nodes that will be updated are half $(n/2)$. In case the number nodes selected for firing are less than half, then new nodes are selected randomly

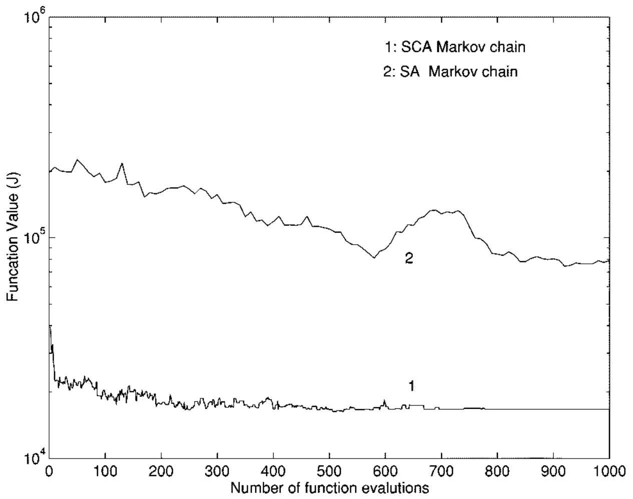


Fig. 7. Squared-error corresponding to the states in Markov chain.

from the unselected list. Note that the selected nodes are updated without changing the optimization value until all are done. All function evaluations performed in parallel are accounted as a single function evaluation while computing the time of execution. Here an iteration is defined as one complete asynchronous update. The value of *Max_Ite* is set to 100. In our experiments, with regard to the clustering problem, we tested with $W = 1$ and $W = 3$. It should be noted that the data has to be made available with every node. This is possible with MIMD message passing

TABLE V
RESULTS OBTAINED WITH K-MEANS AND SA APPROACHES FOR BTD

	K-Means						SA					
K	MiJ	MaJ	AvJ	SD	SR	AvT	MiJ	MaJ	AvJ	SD	SR	AvT
2	336.12	337.24	336.26	0.22	67	0.027	336.12	336.12	336.12	0.0	100	105.72
3	227.22	319.67	230.05	9.38	56	0.039	227.22	227.22	227.22	0.0	100	107.36
4	180.91	221.81	193.24	13.69	23	0.054	180.91	180.91	180.91	0.0	100	108.81
5	160.23	211.80	170.03	9.56	2	0.062	160.23	160.55	160.29	0.13	80	109.38
6	143.27	183.16	154.34	7.30	0	0.071	141.46	144.31	142.64	1.26	70	110.95
7	128.77	178.64	139.96	6.38	0	0.084	126.27	131.21	127.23	1.47	60	112.51
8	117.74	148.07	120.87	7.47	0	0.092	113.50	118.20	116.35	1.58	40	113.08
9	105.34	144.07	120.87	7.47	0	0.105	102.74	109.56	105.33	1.95	20	115.68
10	96.08	137.41	114.01	8.70	0	0.109	92.72	99.39	96.01	2.63	0	117.24

machines such as PARAM [63]. These type of machines provide a suitable environment to parallelize the proposed approach and to handle data of size upto 4 MB.

We present the results obtained for the four standard data sets:

1) German towns data (GTD);
2) British towns data (BTD);
3) Fossil data;
4) 8OX data.

Experiments were carried out for various numbers of clusters, $2 \leq K \leq 10$. Results are compared with that obtained with the SA approach and K-Means algorithm.

The SA algorithm used for comparison is provided in Fig. 6. In SA, the solution is represented as a vector of size N with each value representing the corresponding cluster label associated with that pattern. The neighborhood is generated by simply reassigning a randomly selected pattern to a different cluster. The initial, T_o and final, T_f, temperature values in SA are estimated by using the acceptance ratio, $\mathcal{X}$ which is given by $\mathcal{X} = (m_1 + m_2 * \exp(-\Delta\mathcal{E}^+/T))/(m_1 + m_2)$, where m_1 and m_2 are the number of perturbations for which there is a reduction and increase in the objective function value, respectively. Observe that the perturbations are accepted with a probability $\exp(-\Delta\mathcal{E}^+)$, where $\Delta\mathcal{E}^+$ is the average value of all increments in objective function value due to m_2 moves. Initial and final temperatures are set such that the value of $\mathcal{X}$ is 0.75 and $(\geq)$ 0.999, respectively. In order to allow sufficient time to converge to high quality solutions, the cooling rate, α_T is set to 0.99.

Similarly, the acceptance ratio, $\mathcal{X}_r$ associated with a node r in SCA can be determined as follows: plus3ptminus6ptplus3pt-minus6pt

$$\mathcal{X}_r = \frac{m_1 + m_2\left(1 - \frac{1}{1+\sum_{q=1}^{W}\exp(-\Delta J_q/T)}\right)}{m_1 + m_2} \tag{3}$$

where $\mathcal{P}(l) = \exp(-\Delta J_l/T)/(1 + \sum_{q=1}^{W}\exp(-\Delta J_q/T))$; $\Delta J_0 = 0$, $\Delta J_q = -(J_{r_q} - J_{r_0})$. Here m_1 and m_2 are the same as above. One can estimate the average acceptance ratio $\mathcal{X}_{\text{avg}}$ by using individual acceptance ratios of all nodes in the network when operated in asynchronous parallel mode. This is estimated (approximately) by $\mathcal{X}_{\text{avg}} = \sum_{q=1}^{n}\mathcal{X}_q/n$. Starting temperature

TABLE VI
RESULTS OBTAINED WITH SCA FOR BTD

	$W = 3$					
K	MiJ	MaJ	AvJ	SD	SR	AvT
2	336.12	336.12	336.12	0.0	100	0.09
3	227.22	227.22	227.22	0.0	100	1.77
4	180.91	180.91	180.91	0.0	100	5.92
5	160.23	160.55	160.26	0.10	90	11.23
6	141.46	142.97	141.80	0.55	70	12.26
7	126.28	127.16	126.45	0.27	70	13.13
8	113.50	113.59	113.53	0.04	70	14.61
9	102.74	103.44	103.04	0.26	40	16.27
10	92.68	93.41	92.86	0.22	30	17.36

is estimated by setting $\mathcal{X}_{\text{avg}}$ to 0.75 and final temperature is estimated by setting it to $(\geq)$ 0.999.

In all trials for SCA, we used the asynchronous parallel mode of network execution (Parallel Execution) ensuring that half of the nodes are fired in the network at each time step. All function evaluations performed in parallel are accounted as a single function evaluation while computing the time of execution. The K-Means algorithm was executed for 100 times for each of the data sets. Initial cluster centers are selected randomly with uniform distribution in the bounding search space. Both SA and SCA were run for ten times for all data sets by setting appropriate temperature values. The cooling rate is set to 0.95 in all cases.

German Towns Data: This data set is taken from Spath [60] and consists of cartesian coordinates of 59 towns in Germany. The K-means algorithm [30] results are shown in Table III. In table, MiJ, MaJ, AvJ, SD, SR, and AvT are the minimum, maximum, average, standard deviation, success rate and average execution time of the results obtained for the selected number of runs. Observe that the success rate decreases rapidly with the increase of the number of clusters K. The SA temperature values found for $K = 2$ and 10 are $T_o = 15\,000.0$ and $T_f = 300.0$, and $T_o = 2500.0$ and $T_f = 40.0$, respectively. Results are shown in Table III. The SCA values found for $K = 2$ and 10 are $T_o = 5000.0$ and $T_f = 4500.0$, and $T_o = 2000.0$ and $T_f = 50.0$, respectively, with $W = 1$. As the value of R increases with the increase of K, the average run time increases.

TABLE VII
RESULTS OBTAINED WITH K-MEANS AND SA APPROACHES FOR FOSSIL DATA

	K-Means						SA					
K	MiJ	MaJ	AvJ	SD	SR	AvT	MiJ	MaJ	AvJ	SD	SR	AvT
2	4464.59	4464.59	4464.59	0.0	100	0.028	4464.59	4464.59	4464.59	0.0	100	223.2
3	3408.52	4319.79	3613.01	209.9	10	0.076	3408.52	3408.52	3408.52	0.0	100	225.5
4	2760.85	3764.94	3088.55	326.74	3	0.109	2760.85	2959.80	2860.32	99.45	40	227.7
5	2313.20	3702.34	2650.56	285.33	0	0.146	2312.13	2312.13	2312.13	0.0	100	229.8
6	2022.82	3629.90	2390.96	319.70	0	0.163	2021.75	2191.81	2062.21	65.44	40	231.9
7	1826.22	3553.49	2174.10	274.13	0	0.192	1809.55	1943.25	1901.32	36.21	10	234.1
8	1653.24	2656.27	1971.76	235.62	0	0.231	1689.24	1846.17	1741.75	60.48	0	236.1
9	1517.91	2619.49	1818.55	204.27	0	0.241	1520.21	1778.77	1595.71	76.46	0	238.2
10	1432.76	2406.27	1687.33	202.87	0	0.282	1391.53	1485.80	1449.73	23.92	10	240.3

The statistics of the results are listed in Table IV for $W = 1$ and $W = 3$. Observe that SCA with $W = 3$ could find solutions with 100% success rates for all values of K except for $K = 10$.

It can be observed that the solutions obtained in both the cases, i.e., for $W = 1$ and $W = 3$ are almost equally good. We do not know whether error surface for a given data set consists of deep local minima or not. Testing with different values of W will be helpful in these cases. We may assume that a function does not contain deep local minima if results of the same quality are obtained for different values of W. One advantage of operating with a value of $W > 1$ is that it reduces much of the communication cost as well as the computational cost involved in mapping and re-mapping the variable values. In the subsequent experiments W is set to 3. Fig. 7 depicts the output state sequence of a network in a trial. It can be clearly observed that the near optimal solution(s) is reached within the first 100 iterations. This indicates that the proposed connectionist approach is capable of obtaining near-optimal results very quickly.

We obtained better results than reported by Ismail and Kamel [42] using heuristic methods. For GTD, for clusters $K = 9$ and $K = 10$, the proposed approach could obtain new lower bounds 18 550.43 and 16 307.96 with high success rates against the reported results 18 970.45 and 16 578.80. These results clearly substantiate the advantage of the proposed approach.

British Towns Data: This data set is taken from Andrews [4] and consists of 50 four-dimensional samples. The oiginal data set consists of 155 samples each with 57 variables. The present data set is the first four principle components of the first 50 samples forming four clusters [20].

The K-means algorithm results are shown in Table V. The SA temperature values found for $K = 2$ and 10 are $T_o = 21.0$ and $T_f = 1.0$, and $T_o = 15.0$ and $T_f = 0.4$, respectively. Results are shown in Table V. The SCA temperature values found for $K = 2$ and 10 are $T_o = 100.0$ and $T_f = 80.0$, and $T_o = 35.0$ and $T_f = 1.0$, respectively. The statistics of the results are listed in Table VI. For this data set, in the cases $K = 7, 8, 9$ and 10, we found better quality results than the reported results of this data set using hybrid breadth/depth first search technique [42]. The squared-errors obtained using SCA are 126.27, 113.50, 102.74 and 92.68 as against the published values 127.18, 114.09, 103.49 and 97.43.

TABLE VIII
RESULTS OBTAINED WITH SCA FOR FOSSIL DATA

	$W = 3$					
K	MiJ	MaJ	AvJ	SD	SR (%)	AvT
2	4464.59	4464.59	4464.59	0.0	100	0.07
3	3408.52	3408.52	3408.52	0.0	100	0.52
4	2760.85	2760.85	2760.85	0.0	100	0.68
5	2312.13	2312.13	2312.13	0.0	100	2.31
6	2021.75	2021.75	2021.75	0.0	100	6.52
7	1809.55	1809.55	1809.55	0.0	100	12.43
8	1640.52	1646.99	1643.76	2.93	50	19.47
9	1515.72	1520.19	1518.92	1.56	10	24.91
10	1391.53	1403.03	1397.64	3.71	10	29.34

Fossil Data: This data set is taken from [19] and consists of 87 Nummlitidae Speciments from the Eocene Yellow limestone formation of north western Jamaica. Each specimen is a six-dimensional vector and this data set consists of three clusters [20].

The K-means algorithm results are shown in Table VII. Observe that the success rate decreases rapidly with the increase of the number of clusters K. The temperature values found for $K = 2$ and 10 are $T_o = 1000.0$ and $T_f = 40.0$, and $T_o = 25.0$ and $T_f = 2.0$, respectively. Results are shown in Table VII. The SCA temperature values found for $K = 2$ and 10 are $T_o = 850.0$ and $T_f = 500.0$, and $T_o = 95.0$ and $T_f = 1.0$, respectively. The statistics of the results are listed in Table VIII.

8OX data: This data set is taken from [43] and consists of 45 patterns each with eight features. This data set is derived from the Munson hand printed FORTRAN character set. This data set consists of features extracted from a digitized, on a 24×24 grid, hand written characters 8, O, and X. Each character has 15 representative patters, thus forming a data set of size 45.

The K-means results are shown in Table IX. The SA temperature values found for $K = 2$ and 10 are $T_o = 1000.0$ and $T_f = 2.0$, and $T_o = 15.0$ and $T_f = 1.0$, respectively. Results are shown in Table IX. The SCA temperature values found for $K = 2$ and 10 are $T_o = 150.0$ and $T_f = 80.0$, and $T_o = 50.0$ and $T_f = 1.0$, respectively. The statistics of the results are listed in Table X. These results clearly demonstrate the

TABLE IX
RESULTS OBTAINED WITH K-MEANS AND SA APPROACHES FOR 8OX

	K-Means						SA					
K	MiJ	MaJ	AvJ	SD	SR	AvT	MiJ	MaJ	AvJ	SD	SR	AvT
2	1507.49	2011.35	1549.69	109.07	50	0.045	1507.49	1507.49	1507.49	0.0	100	86.1
3	1201.21	1441.21	1289.83	62.65	8	0.059	1201.21	1201.21	1201.21	0.0	100	87.8
4	1026.21	1312.27	1128.65	82.95	7	0.077	1026.21	1026.21	1026.21	0.0	100	89.3
5	859.68	1232.54	981.37	72.69	1	0.096	859.68	868.72	861.49	3.62	80	90.9
6	733.86	1192.76	887.62	76.00	0	0.109	729.26	780.92	751.48	22.36	50	92.6
7	650.02	1092.17	810.34	78.80	0	0.127	641.47	641.47	641.47	0.0	100	94.2
8	586.35	961.05	739.56	84.60	0	0.151	580.72	591.51	585.87	3.63	20	95.8
9	544.89	972.15	697.61	87.00	0	0.163	524.28	537.51	528.63	4.57	30	97.5
10	485.04	817.01	634.59	80.00	0	0.181	471.73	480.67	477.81	2.89	0	99.3

TABLE X
RESULTS OBTAINED WITH SCA FOR 8OX.

	$W = 3$					
K	MiJ	MaJ	AvJ	SD	SR	AvT
2	1507.49	1507.49	1507.49	0.0	100	0.12
3	1201.21	1201.21	1201.21	0.0	100	0.40
4	1026.21	1026.21	1026.21	0.0	100	2.79
5	859.68	859.68	859.68	0.0	100	4.78
6	729.26	729.26	729.26	0.0	100	9.04
7	641.47	642.98	641.61	0.45	90	18.12
8	580.72	587.46	582.39	2.43	60	22.46
9	524.28	529.73	526.57	2.05	30	23.90
10	470.05	476.80	474.85	2.76	10	25.49

superiority of SCA in obtaining better clusters characterized by the squared-error value for eight-dimensional data.

The connectionist approach presented in this paper is suitable for clustering small or medium sized data sets with a moderate or large number of clusters [6]. Parallel algorithms can be developed for SCA to make use of parallel hardware for optimizing large scale functions.

VII. CONCLUSION

In this paper, we have investigated the applicability of a new stochastic connectionist approach for solving the clustering problem. A stochastic connectionist approach along with its generalized version has been proposed and its efficacy is showed over a variety of test functions. The clustering problem has been formulated as a function optimization problem and the proposed connectionist approach has been employed to solve it. The results obtained with the proposed approach for four data sets demonstrate superiority over K-means and SA approach. One of the major advantages of this approach over other stochastic methods is that, it utilizes gradient knowledge and performs stochastic search along the gradient direction. This facilitates faster convergence to the desired partition. The parallelism that can be obtained with the proposed approach is proportional to the number of clusters.

APPENDIX I
ASYMPTOTIC CONVERGENCE

For all practical purposes, without loss of generality, we can assume that the continuous search space is discretized, as a digital computer storage has only finite precision. Let Δx be the precision offered. The number of values a variable x can take within the range $a \leq x \leq b$, is $\lfloor (b-a)/\Delta x \rfloor$, where $\lfloor y \rfloor$ denotes the greatest integer smaller than or equal to y. This assumption of the search space being discretized facilitates interpretation of the state sequence of network as a Markov chain. The proof of convergence to a global optimal configuration requires the following assumptions: The search space is finite; At least one of the configurations in the discretized search space is a global optimal configuration; Variance σ^2 is assumed to be constant. From bound constraints, $\mu \leq x \leq \nu$, we can compute the size of the search space, $|S| = \prod_{i=1}^{n} \lfloor (\nu(i) - \mu(i))/\Delta x \rfloor$. Each variable x_i can take a value from the set $S_{x_i} = \{\mu(i), \mu(i) + \Delta x, \ldots, \nu(i)\}$. Now the search space S is the cartesian product of $S_{x_1}, \ldots, S_{x_n}$, $S = S_{x_1} \times S_{x_2} \cdots \times S_{x_n}$. The output values of nodes in the network form a configuration or solution in S. Let $S_* \subset S$ be the set of all optimal solutions. The steady state distribution over solutions in the discretized search space with finite size is given by the Boltzmann distribution

$$\pi_T = \frac{1}{Z_T} \exp(-f(x)/T) \tag{4}$$

where

$$Z_T = \sum_{x_1 \in S_{x_1}} \sum_{x_2 \in S_{x_2}} \cdots \sum_{x_n \in S_{x_n}} \exp(-f(x)/T)$$
$$Z_T = \sum_{\forall x \in S} \exp(-f(x)/T).$$

The state transition of the network from state i to state j is defined by transitional probabilities from one state to the other. The transitional probabilities are defined as follows:

$$P_{ij}^T = \begin{cases} G_{ij} A_{ij}^T, & \text{if } i \neq j \\ 1 - \sum_{x \in \aleph_i, x \neq i} G_{ix} A_{ix}^T, & \text{if } i = j \end{cases}$$

where $\aleph_i$ represents a set of neighborhood configurations of i, G_{ij} is the probability of generating the configuration j from i

and A_{ij}^T is the acceptance probability. In the proposed connectionist approach with $W = 1$, configuration j is a neighborhood configuration of i , if and only if they satisfy the criterion: $\exists r$ such that $i(m) = j(m), \forall m \neq r$ and $i(r) \neq j(r)$. This implies that two configurations i and j differ at rth node output value. The probability of generating a configuration j from i by updating node r is given by

$$G_{ij}^r = \int_{j(r)-\frac{\Delta x}{2}}^{j(r)+\frac{\Delta x}{2}} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(i(r)-y)^2}{2\sigma^2}\right) dy \qquad (5)$$

and $G_{ij} = (1/n)G_{ij}^r$, where $1/n$ is the probability of selecting rth node for updating.

The acceptance probability is given by

$$A_{ij}^T = \exp(-(f_j - f_i)^+/T)$$

where

$$(f_j - f_i)^+ = \begin{cases} 0, & \text{if } (f_j - f_i) \leq 0 \\ f_j - f_i, & \text{otherwise.} \end{cases}$$

Theorem [28]: For any irreducible and aperiodic Markov chain with transition matrix P , the steady state distribution, $\mathbf{q}$, exists and is independent of the initial distribution q_0. The components of $\mathbf{q}$ can uniquely be determined by the $\sum_i q_i P_{ij}^T = q_i, \forall j$.

After infinite number of iterations of the network at a fixed temperature, T , the probability distribution of solutions reaches the stationary distribution. In order to prove that the Markov chain given by a state transitional matrix P reaches the stationary distribution, we use the following Lemma.

Lemma 1 [1]: If the Markov chain is irreducible and aperiodic and the components of $\mathbf{q}$ satisfy the property

$$q_i P_{ij}^T = q_j P_{ji}^T$$

then $\mathbf{q}$ is called the stationary distribution.

In order to prove the existence of stationary distribution, we need to show that the Markov chain is irreducible and aperiodic.

1) Irreducibility: The Markov chain of P is irreducible if $P_{ij}(q) > 0, \forall i, j$, i.e., there exists a nonzero probability of reaching a state from any other state in a finite number of steps (q).

Proof: Let the state j be reachable from i. A state is said to be reachable from another state, if and only if, there exists a valid sequence of state transitions from one state to the other. So, the probability of reaching state j from state i in q transitions is given by $(T > 0)$

$$\begin{aligned} P_{ij}^T(q) &= \sum_{\forall i_1, i_2, \ldots, i_{q-1} \in S} P_{ii_1}^T \cdots P_{i_{q-1}j}^T \\ &\geq G_{ii_1} A_{ii_1}^T \cdots G_{i_{q-1}j} A_{i_{q-1}j}^T \end{aligned}$$

we have $G_{ij} > 0$ and $A_{ij}^T > 0, j \in \aleph_i, i \neq j$, so, $P_{ij}^T(q) > 0$ and this is true for all states i and j. So every state is reachable from every other state. Thus proving that the Markov chain is irreducible. □

2) Aperiodicity: If the Markov chain associated with the transition matrix P is irreducible, then if any one state is aperiodic, then all states in the Markov chain are aperiodic. It is enough to prove that one state is aperiodic. Let the function values of configurations are $f_i < f_j$, and $G_{ij} > 0$. If $i, j \notin S_*$, then $A_{ij}^T < 1$. The aperiodicity of state i can be proved, if we can show that $P_{ii}^T > 0$.

Proof:

$$\begin{aligned} P_{ii}^T &= 1 - \sum_{m \in S, m \neq i} G_{im} A_{im}^T \\ &> 1 - \sum_{m \in S, m \neq i} G_{im} \quad \text{as } A_{ij} < 1 \\ P_{ii}^T &> 0 \end{aligned}$$

Hence, all states in the Markov chain under consideration are irreducible and aperiodic. In order to prove that the distribution of configurations after infinite number of iterations of the network is stationary, we need to prove that the components of $\mathbf{q}$ satisfy the equation $q_i P_{ij}^T = q_j P_{ji}^T$. □

Proof:

$$\begin{aligned} q_i^T P_{ij}^T &= \frac{1}{Z_T} \exp(-f_i/T) G_{ij} A_{ij}^T \\ &= \frac{1}{Z_T} G_{ij} \exp(-f_i/T) \exp\left(-\frac{(f_j - f_i)^+}{T}\right) \\ &= \frac{1}{Z_T} G_{ij} \exp(-f_j/T) \exp\left(-\frac{(f_i - f_j)^+}{T}\right). \end{aligned}$$

We have to prove that $G_{ij} = G_{ji}$. As the configurations i and j differ at position r and $i \in \aleph_j, j \in \aleph_i$, we have:

$$\begin{aligned} G_{ij} &= \frac{1}{\sqrt{2\pi}\sigma} \int_{j(r)-\frac{\Delta x}{2}}^{j(r)+\frac{\Delta x}{2}} \exp\left(-\frac{(i(r)-y)^2}{2\sigma^2}\right) dy \\ &\approx \frac{1}{\sqrt{2\pi}\sigma} \frac{1}{2} \left[\exp\left(-\frac{\left(i(r) - j(r) - \frac{\Delta x}{2}\right)^2}{2\sigma^2}\right) \right. \\ &\quad \left. + \exp\left(-\frac{\left(i(r) - j(r) + \frac{\Delta x}{2}\right)^2}{2\sigma^2}\right) \right] \Delta x \\ &= \frac{1}{\sqrt{2\pi}\sigma} \int_{i(r)-\frac{\Delta x}{2}}^{i(r)+\frac{\Delta x}{2}} \exp\left(-\frac{(y - j(r))^2}{2\sigma^2}\right) dy \\ &= G_{ji}. \end{aligned}$$

We assume that $G_{ij} = G_{ji}$ and the error due to approximation can be ignored. So we have

$$\begin{aligned} q_i^T P_{ij}^T &= \frac{1}{Z_T} G_{ji} \exp(-f_j/T) \exp\left(-\frac{(f_i - f_j)^+}{T}\right) \\ &= q_j P_{ji}^T. \end{aligned}$$

This proves that the distribution of solutions or configurations after infinite iterations reach the stationary distribution. Now, we need to prove that network converges to optimal solutions asymptotically at the zero temperature

$$\lim_{T \to 0} q_T(i) = \frac{1}{|S_*|} \mathrm{Opt}(i), \quad \forall i$$

where

$$\mathrm{Opt}(i) = \begin{cases} 1, & \text{if } i \in S_* \\ 0, & \text{otherwise.} \end{cases}$$

□

Proof:

$$\lim_{T\to 0} q_T(i) = \lim_{T\to 0} \frac{\exp\left(-\frac{f_i}{T}\right)}{Z_T}$$
$$= \lim_{T\to 0} \frac{\exp\left(\frac{f^*-f_i}{T}\right)}{\sum_{j\in S}\exp\left(\frac{(f^*-f_j)}{T}\right)}$$

For all states $i \notin S_*, \lim_{T\to 0}\exp((f^* - f_i)/T) = 0$, and for all other states, $i \in S_*, \lim_{T\to 0}\exp((f^* - f_i)/T) = 1$. So $\lim_{T\to 0} q_T(i) = \mathrm{Opt}(i)/|S_*|$. Hence at zero temperature, the stationary distribution converges to global optimal solutions. The asymptotic convergence proof for $W > 1$ is not undertaken in this paper. We can have different value of W for different variables. Let W_i be the number of samples being considered by ith node. If we assume that $W_i = \lfloor(\mu_i - \nu_i)/\Delta x\rfloor, \forall i$, then with a similar Markov chain analysis, we can prove the asymptotic convergence to global optimal solution(s).

Appendix II

Functions 1–12 are taken from [32, pp. 146–150].

Function 13. [F13] [32, p. 146, Prob. 14] Goldstein-Price (GP) function. .

Function 14. [F14] [32, p. 146, Prob. 19].

Function 15. [F15] [17, COS function: p. 197, Cosine mixture problem.]

Acknowledgment

The authors wish to thank the anonymous reviewers for their valuable comments on the earlier draft of this paper.

References

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machine: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. New York: Wiley, 1989.

[2] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cogn. Sci.*, vol. 9, pp. 147–169, 1985.

[3] M. R. Anderberg, *Cluster Analysis for Applications*. New York: Academic, 1973.

[4] D. F. Andrews, "Plots of high dimensional data," *Biometrics*, vol. 28, pp. 125–136, 1972.

[5] L. Aristidis and S. L. Andreas, "Group updates and multi scaling: An efficient neural network approach to combinatorial optimization," *IEEE Trans. Syst., Man, Cybern. B.*, vol. 26, pp. 222–232, Apr. 1996.

[6] G. P. Babu, "Pattern Clustering with Connectionist and Evolutionary Approaches," Ph.D. dissertation, Comput. Sci. Automat., Ind. Inst. Sci., Bangalore, 1994.

[7] G. P. Babu and M. N. Murty, "A near-optimal initial seed value selection in k-means algorithm using a genetic algorithm," *Pattern Recognit. Lett.*, vol. 14, pp. 763–769, 1993.

[8] G. P. Babu and M. N. Murty, "Clustering with evolution strategies," *Pattern Recognit.*, vol. 27, no. 2, pp. 321–329, 1994.

[9] ——, "Simulated annealing for selecting initial seeds in k-means algorithm," *Ind. J. Pure Appl. Math.*, vol. 24, no. 1 and 2, pp. 85–94, 1994.

[10] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1962.

[11] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[12] J. C. Bezdek and S. K. Pal, Eds., *Fuzzy Models for Pattern Recognition*. New York: IEEE Press, 1992.

[13] J. N. Bhuyan, V. V. Raghavan, and K. E. Venkatesh, "Genetic algorithm for clustering with an ordered representation," in *Proc . 4th Int. Conf. Genetic Algorithms*, 1991, pp. 408–415.

[14] G. L. Bilbro and W. E. Synder, "Optimization of functions with many minima," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 4, pp. 840–849, 1991.

[15] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, "Generalized simulated annealing for function optimization," *Technometrics*, vol. 28, no. 3, pp. 209–217, 1986.

[16] A. Bouzerdoum and T. R Pattison, "Neural network for quadratic optimization with bound constraints," *IEEE Trans. Neural Networks*, vol. 4, no. 2, pp. 293–304, 1993.

[17] L. Breiman and A. Cutler, "Deterministic algorithm for global optimization," *J. Math. Programm.*, vol. 58, pp. 179–199, 1993.

[18] D. E. Brown and C. L. Huntley, "A practical application of simulated annealing to clustering," *Pattern Recognit.*, vol. 25, no. 4, pp. 401–412, 1992.

[19] D. F. Chernoff, "The use of faces to represent points in k-dimensional space graphically," *J. Amer. Stat. Assoc.*, vol. 58, no. 342, pp. 361–368, 1973.

[20] Y. T. Chien, *Interactive Pattern Recognition*. New York: Marcel Dekker, 1978.

[21] L. O. Chua and G. Lin, "Nonlinear programming without computation," *IEEE Trans. Circuits Syst.*, vol. CAS-31, no. 2, pp. 182–188, 1984.

[22] G. B. Coleman and H. C. Andrews, "Image segmentation by clustering," in *Proc. IEEE*, 1967, pp. 773–785.

[23] A. Corana, M. Marchesi, C. Martini, and S. Ridella, "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm," *ACM Trans. Math. Softw.*, vol. 13, no. 3, pp. 262–280, 1987.

[24] L. C. W. Dixon and G. P. Szego, Eds., *Toward Global Optimization*. Amsterdam, The Netherlands: North-Holland, 1975.

[25] ——, *Toward Global Optimization 2*. Amsterdam, The Netherlands: North-Holland, 1978.

[26] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[27] A. Rodriguez-Vazquez *et al.*, "Nonlinear switched-capacitor neural networks for optimization problems," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 384–398, Apr. 1990.

[28] W. Feller, *An Introduction to Probability Theory and Its Applications 1*. New York: Wiley, 1950.

[29] D. B. Fogel and P. K. Simpson, "Experiments with evolving fuzzy clusters," in *Proc. 2nd Annu. Conf. Evol. Prog.*, CA, 1993, pp. 90–97.

[30] E. W. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," *Biometrics*, vol. 21, no. 768, 1965.

[31] Y. Fukada, "Spatial clustering procedures for region analysis," *Pattern Recognit.*, vol. 12, pp. 395–403, 1980.

[32] R. Ge and Y. Qin, "The globally convexized filled functions for global optimization," *Appl. Math. Comput.*, vol. 35, pp. 131–158, 1990.

[33] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution and Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721–741, 1984.

[34] S. Geman and C. Hwang, "Diffusions for global optimization," *SIAM J. Contr. Optim.*, vol. 24, no. 5, pp. 1031–1043, 1986.

[35] A. D. Gordon and J. T. Henderson, "Algorithm for Euclidean sum of squares classification," *Biometrics*, vol. 33, pp. 355–362, 1977.

[36] K. C. Gowda and E. Diday, "Symbolic clustering using a new dissimilarity measure," *Pattern Recognit.*, vol. 24, no. 6, pp. 567–578, 1991.

[37] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, pp. 4–29, 1984.

[38] L. O. Hall, I. B. Ozyurt, and J. C. Bezdek, "Clustering with a genetically optimized approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 103–112, 1999.

[39] R. M. Haralick and G. L. Kelly, "Pattern recognition with measurement space and spatial clustering," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 440–450, 1975.

[40] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

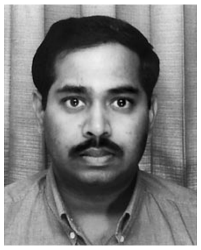
[41] K. Hwang and D. Kim, "Parallel pattern clustering on a multiprocessor with orthogonally shared memory," presented at the Int. Conf. Parallel Processing, 1987.

[42] M. A. Ismail and M. S. Kamel, "Multi dimensional data clustering utilizing hybrid search strategies," *Pattern Recognit.*, vol. 22, pp. 75–89, 1989.

[43] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[44] D. Jones and M. A. Beltramo, "Clustering with Genetic Algorithm," General Motors Research Labs., Warren, MI, GMR-7156, 1990. Research Publication.

[45] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 554–562, May 1988.
[46] C. D. Kirkpatrick, Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.
[47] R. W. Klein and R. C. Dubes, "Experiments in projection and clustering by simulated annealing," *Pattern Recognit.*, vol. 22, no. 2, pp. 213–220, 1989.
[48] T. Kohonen, *Self-Organization and Associative Memory*. New York: Spinger-Verlag, 1984.
[49] W. L. G. Koontz, P. M. Narendra, and K. Fukunaga, "A branch and bound clustering algorithm," *IEEE Trans. Comput.*, vol. C-23, pp. 908–914, 1975.
[50] X. Li and Z. Fang, "Parallel clustering algorithms," *Parallel Comput.*, vol. 11, pp. 275–290, 1989.
[51] L. M. Ni and A. K. Jain, "A VLSI systolic architecture for pattern clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, Jan. 1985.
[52] B. K. Parsi, J. A. Gualtieri, J. A. Devaney, and K. K. Parsi, "Clustering with neural networks," *Biol. Cybern.*, vol. 63, pp. 201–208, 1990.
[53] V. V. Raghavan and B. Agarwal, "Optimal determination of user-oriented clusters: An application for the reproductive plan," in *Genetic Algorithms and Their Applications: Proc. 2nd Int. Conf. Genetic Algorithms*, 1987, pp. 241–246.
[54] V. V. Raghavan and K. Birchand, "A clustering strategy based on a formalism of the reproductive process in natural system," in *Proc. 2nd Int. Conf. Information Storage and Retrieval*, 1979, pp. 10–22.
[55] S. Ranka and S. Sahni, "Clustering on hypercubes," *IEEE Trans. Parallel Distrib. Processing*, pp. 128–138, 1991.
[56] M. R. Rao, "Cluster analysis and mathematical programming," *J. Amer. Stat. Assoc.*, vol. 66, pp. 622–626, 1971.
[57] K. Rose, E. Gurewitz, and G. Fox, "A deterministic annealing approach to clustering," *Pattern Recognit. Lett.*, vol. 11, pp. 589–594, 1990.
[58] S. Z. Selim and K. Alsulton, "A simulated annealing algorithm for the clustering problem," *Pattern Recognit.*, vol. 10, no. 24, pp. 1003–1008, 1991.
[59] S. Z. Selim and M. A. Ismah, "Soft clustering of multi-dimensional data: A semifuzzy approach," *Pattern Recognit.*, vol. 17, no. 5, pp. 559–568, 1984.
[60] H. Spath, *Cluster Analysis Algorithms for Data Reduction and Classification*. West Sussex, U.K.: Ellis Horwood, 1980.
[61] V. Sridhar and M. N. Murty, "Clustering algorithms for library comparison," *Pattern Recognit.*, vol. 24, no. 9, pp. 815–823, 1991.
[62] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 533–541, 1986.
[63] Center for development of advanced computing, in C-DAC, 1993. PARAS Tutorial.
[64] R. T. Rockafeller, "Lagrange multipliers and optimality," *SIAM Rev.*, pp. 183–238, 1993.
[65] D. E. Van den bout and T. K. Miller, "Graph partitioning using annealed neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 2, pp. 192–203, 1990.
[66] D. Vanderbilt and S. G. Louie, "A Monte-Carlo simulated annealing approach to optimize over continuous variables," *J. Comput. Phys.*, vol. 56, pp. 259–271.
[67] H. D. Vinod, "Integer programming and theory of grouping," *J. Amer. Stat. Assoc.*, pp. 506–519, 1969.
[68] P. Withold, "Fuzzy clustering with partial supervision," *IEEE Trans. Syst., Man, Cybern. B*, vol. 27, pp. 787–795, Oct. 1997.
[69] E. L. Zapata, F. F. Rivera, O. G. Plata, and M. A. Ismail, "Parallel fuzzy clustering on fixed size hypercube simd computers," *Parallel Comput.*, vol. 11, pp. 291–303, 1989.
[70] S. Zhang, X. Zhu, and L. Zou, "Second-order neural nets for constrained optimization," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 1021–1024, 1992.



G. Phanendra Babu (M'96) received the B.E. degree in computer science from the Andhra University in 1989, the M.Tech. degree in computer science from the University of Hyderabad, India, in 1991, and the Ph.D. degree in computer science from the Indian Institute of Science, Bangalore, in 1994.

He was a Senior Systems Engineer at Wipro Systems Ltd., Bangalore, from 1991 to 1992. From 1994 to 1997, he was a Research Staff Member at the Institute of Systems Science, now Kent Ridge Digital Laboratories, National University of Singapore, Singapore. He is currently a Senior Member of Technical Staff at the Technology Deployment International Inc., Santa Clara, CA. His research interests include multimedia information systems, pattern recognition, neural networks, genetic algorithms, intelligent information filtering, image databases, and machine learning.

M. Narasimha Murty received the B.E. and M.E. degrees in electrical engineering and the Ph.D. degree in pattern recognition in 1982 from the Indian Institute of Science, Bangalore,

From 1982 to 1983, he was a Project Leader in the Processor Systems India, Bangalore. He is currently a Professor in the Department of Computer Science and Automation. His research interests are in pattern clustering, data mining, and neural networks.

Dr. Murty is a member of the Indian Unit of Pattern Recognition and Artificial Intelligence.

S. Sathiya Keerthi received the Ph.D. degree in control engineering from The University of Michigan, Ann Arbor, in 1987.

In April 1987, he joined the faculty of the Department of Computer Science and Automation, Indian Instiitute of Science, Bangalore. He has recently joined the Control Division of the Department of Mechanical and Production Engineering, National University of Singapore, as Associate Professor. His research interests are in the areas of neural networks, support vector machines, numerical optimization, and geometric algorithms in robotics. He has published over 50 papers in leading international journals and conferences.