Indian J. pure appl. Math., **25** (1&2); 85-94, January & February 1994

SIMULATED ANNEALING FOR SELECTING OPTIMAL INITIAL SEEDS IN THE *K*-MEANS ALGORITHM

G. PHANENDRA BABU AND M. NARASIMHA MURTY

Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012

(Accepted 24 *June* 1993)

In this paper, we explore the applicability of simulated annealing, a probabilistic search method, for finding optimal partition of the data. A new formulation of the clustering problem is investigated. In order to obtain optimal partition, search is undertaken to locate optimal initial seeds, such that the *K*-means algorithm converges to optimal partition. Search space involved in this process is continuous, so decretization is done and simulated annealing is employed for locating optimal initial seeds. Experimental results substantiate the proposed method. Results obtained with the selected data sets are presented.

1. INTRODUCTION

Clustering methods have many potential applications in different areas. Major concern of clustering methods is to form groups such that the data in a group have similar characteristics. Various types of clustering techniques have been proposed in the literature. These can be broadly categorized into 1. Hierarchical and 2. Partitional methods. Hierarchical techniques build a dendogram structure, whereas partitional methods partition the data into a specified or estimated number of clusters. Some partitional methods allow on-line human intervention while clustering process is undertaken. In this paper, we restrict our scope to partitional methods and the number of clusters, C is assumed to be known. Readers are refered to[1, 2, 3] for comprehensive coverage of the clustering algorithms.

Partitional algorithms try to partition the data by minimizing the specified or chosen mathematical criterion function. Most of the clustering methods are gradient descent techniques and converge to local optimal partition, i.e., converge to local optimal solution of the criterion function. Many applications such as image compression, image segmentation, and other engineering problems, require to find optimal partition of the data. The number of possible partitions of a given set of n data items and C clusters is given by stirling approximation[1],

$$S = \frac{1}{C!} * \sum_{k=0}^{C} ((-1)^{C-k} * \binom{C}{k} * k^n). \quad \ldots (1)$$

One way of finding optimal partition is to evaluate the objective function value for each possible partition and select the partition that yields minimum objective function value. This is nothing but exhaustive enumeration and is virtually ruled out as the number of partition increase exponentially with the increase in n and C. Some heuristic procedures such as integer programming, dynamic programming, branch and bound methods have been investigated in this context. But these methods work for small data sets and consume prohibitive amount of time in the case of moderate or large data sets.

Recently probabilistic search techniques that have been modeled based on the natural processes have been extensively used to accomplish the task of finding the global optimal solutions of complex optimization problems. By very nature of the stochasticity involved in the search process these techniques effectively avoid getting stuck in local optima, thus try to converge to global optimal solution. Most important of them are Simulated Annealing (SA)[7,8,16], Neural Networks (NNS[17]), Genetic Algorithms (GA's[18]) Evolution Strategies (ES)[20], and Evolutionary programming (EP)[21]. The first one is modeled using *Annealing* concept in *condensed matter physics*. The second one adopts the principles of neuronal functioning in brain. The last three are the population based models that simulate the natural evolution. As our major concern is to obtain optimal partition of the data, here we present the relevant work done in this area using the above mentioned techniques.

GAs are studied in the context of clustering by various researchers[19]. Successful application of neural networks methods for clustering were undertaken in[17]. Recently ES and EP have been employed for clustering analysis[14, 15]. Klein and Dubes[4] investigate the applicability of SA to clustering problem which searches in the search space of all possible partitions, given by equation 1, optimizing an objective function. In this paper, we confine our discussion to simulated annealing method for obtaining optimal partition of data by finding optimal initial seeds such that the chosen K-means algorithm converges to optimal partition. Comparing the performances of these methods is out of the scope of this paper.

It is clear that finding optimal partition is equal to finding the extrema of the selected criterion function. In this paper, we focus on one of the famous clustering objective criterion function, *total within group sum of squared error*. In general, K-means algorithm is used to optimize that criterion function yielding a partition upon convergence. The K-means algorithm converges in a finite number of steps[10] to a local minimum. It is very rare to obtain optimal partition of data unless sufficient knowledge about inherent natural grouping is available. This knowledge is generally not available for many problems.

In the next section we present the problem formulation. Section 3 presents a brief review of Simulated Annealing and its applications. Simulated Annealing for clustering is disucssed in Section 4. Computational complexity of the algorithm is presented in Section 5. Experimental results are given in Section 6.

2. PROBLEM FORMULATION

The K-means algorithm[1, 3] starts with some initial seed values[1] and alternates between two steps, one is assigning data items to clusters, and the other is computing cluster centers, until it reaches a stable configuration or a termination condition is satisfied. This is Forgy's[1] version of K-means algorithm, and in fact there are other varieties that have been proposed. It converges to a local minimum in a finite number of steps[10]. Many schemes were suggested for selecting initial seed values. We formalize the square error criterion :

$X = \{x_1, ..., x_n\}$ is a set of n data vectors or items each of d dimensions

C is the number of clusters,

$V = \{v_1, ..., v_C\}$ is the set of cluster centers,

$x_i, v_j \in R^d$, $1 \le i \le n$, $1 \le j \le C$,

Optimization problem can be formulated as

$$\text{minimize } J(W, V) = \sum_{k=1}^{C} \sum_{i=1}^{n} u_{ki} D^2 (x_i, v_k),$$

$$\text{subject to } \sum_{k=1}^{C} u_{ki} = 1,$$

and $u_{ki} \in \{0, 1\}$,

$1 \le k \le C$, $1 \le i \le n$

where $W = [u_{ki}]$, is an $C \times n$ association matrix,

$D(x, v) = \sqrt{\Sigma_{l=1}^{d} (x(l) - v(l))^2}$.

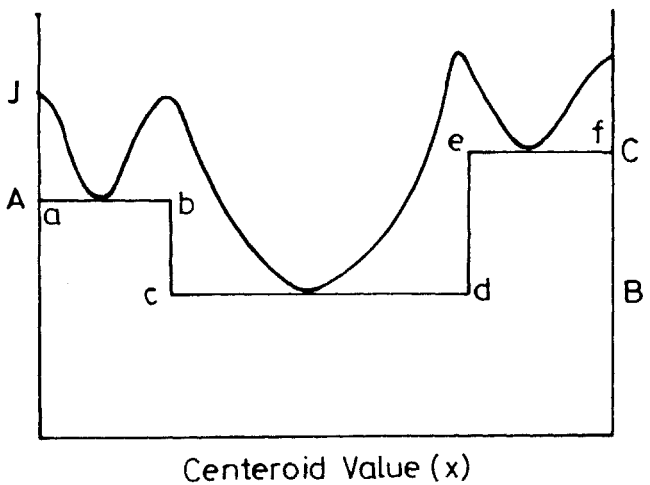

FIG. 1. Multi modal error surface.

The final partition obtained by K-means algorithm is dependent on initial seed values or centroids. Many heuristics have been proposed in the literature to choose better initial seeds such that the K-means algorithm converges to a good partition characterized by the minimum objective function value. We define optimal initial seeds as those which make K-means to converge to extrema of criterion function. Consider a single dimension multi-modal error surface shown in Fig. 1. Any initial value of x in the range $[c, d]$ will guarantee the gradient descent algorithm to converge to global optimum. So all values of x in the vicinity of global minimum are optimal initial seed values. In the above case vicinity is defined by the range $[c, d]$. Now the problem can be formulated as; search for optimal initial seeds such that the chosen algorithm converges to global optimum.

It is to be observed as the search space is continuous, the number of possible candidate solutions is infinite. In order to reduce the complexity, we segment the search space. Each dimension is segmented into $M - 1$ segments. Such that there are d^M possible search points. The number of possible combinations for selecting C grid points as initial seeds is C^M. So the search space increase exponentially as M and C increase. We apply simulated annealing method for locating optimal initial seeds. In the next section we present a brief overview of *SA* and its applications.

3 Simulated Annealing

Simulated annealing approach is based on the idea of a process in metallurgy called *annealing*. Annealing process is performed to attain the low energy states of a solid in heat bath. This involves two steps : in the first step the metal is heated to high temperature till it melts where the particles of the matter arrange in a random fashion and in the second step annealing of metal takes place by decreasing the temperature slowly such that the metal reaches equilibrium state at each temperature decrease. At ground state particles are arranged in an ordered manner forming a lattice strcuture with energy being minimal. This concept of annealing has been simulated by Metropolis *et al.*[16] to study the evolution of soild in heat bath.

The idea of simulating annealing has been adopted by Kirkpatrik *et al.*[6], for solving difficult combinatorial problems with many variables. A configuration of the particles in heat bath is equated to a solution or a configuration of values of the variables in the search space. Each solution will have a corresponding cost associated with it and is computed by defined objective or criterion function. Simulated annealing starts with some random initial solution, S_0 with cost E_0. A neighbourhood solution S_1 is generated by perturbing S_0 and E_1 is computed. If $E_1 < E_0$ then S_1 is accepted as the current solution else it is accepted with the probability $exp(-(E_1 - E_0)/T)$ where T is the control parameter called temperature in the annealing context. This can be veiwed as a Markov chain and is run untill a stable state is reached.

Many problems have been attempted using this method and the studies prove that this mthod is quite promising to solve Combinatorial optimization (CO) problems. Some of its potential applications are TSP[12], Scheduling[13], Image processing[14]. In the book by Laarhoven and Aarts[7] a detailed discussion about the applications is

presented. For applying SA to CO problems the following points have to be considered :

Problem representation : This involves representation of the solution space and formulating an objective function for computing the merit of a solution. The cost/objective function should be chosen such that it represents the effectiveness of the solution. The problem representation and cost function should be so chosen that their manipulation and computation do not consume much of the CPU time.

Generating neighbourhood solution : For changing current solution, first a new solution has to be generated from the current solution by perturbation. The pertubing mechanism should be designed such that for the given problem, the optimal solution can be attained through a shortest path. After generating the new solution, its cost should be evaluated.

Parameters : For solving a CO problem using the SA method, the following parameters are required.

1. An initial value of the control prameter often referred as initial temperature T_0.
2. A decrement factor α, for decreasing the value of the temperature.
3. A final value of the control parameter, also called final temperature, T_f is specified by the stopping criterion.

At any instant, the value of the control parameter, T determines the probability of the acceptance of a newly generated solution when the cost of it is larger than the cost of the solution from which it is generated.

4. SA FOR SEARCHING OPTIMAL INITIAL SEEDS

In this section, we present SA approach for finding J^*. As discussed in the last section, we need to represent the solution, formulate the objective function, and set control parameters. In order to reduce search effort we use discretized space,as discussed in section 2. The solution S is represented by a vector of size Cxd and each value in it will fall in the range $\{1...M\}$. Decoding mechanism is necessary that converts a given solution into corresponding initial seeds and is performed as follows :

$$v_{ij}(l) = \frac{S((i-1)*d+l)}{M} * (R(l) - L(l)) + L(l),$$

$$1 \le i \le n,\ 1 \le j \le C \qquad \ldots (2)$$

where $L(q) = \min_p (x_p(q))$, $R(q) = \max_p (n_p(q))$,
for all p, q such that $1 \le q \le d$, $1 \le p \le n$.

Cost/Objective function evaluation : As we are incorporating *K*-means algorithm in the search process, the cost of a given solution is computed by running the *K*-means algorithm and is done as follows :

K-means(S)

1. Convert the given solution, S, into initial seed values, eqn. 2,

2. assign all data to the nearest centroids,
3. compute cluster centers,
4. if stable assignment is not obtained go to to step 2 else return $J()$ value.

SA starts with some random initial solution S_0. The algorithm for finding optimal solutions and optimal partition is presented in Fig. 2.

Algorithm : Simulated Annealing

Input : T_0, T_f, initial and final temperature respectively,
Max-Ite, Maximum number of inner loop iterations,
No-Ite, Maximum number of outer loop iterations,
Output : S, optimal or near optimal initial seed value vector,
begin
initialize S^0, E^0 = K–means (S^0), p = *No-Ite*,
while $(T_0 > T_f)$
begin
k = 0;
While (k < *Max-Ite*)
begin
S^1 = *Perturb* (S^0, p),
E^1 = K–means (S^1),
iff $((E^1 < E^0) \vee (\exp(-(E^1 - E^0)/T_0) > random(0, 1)))$
begin
$S^0 = S^1$, $E^0 = E^1$;
end
$k = k + 1$,
end
$p = p + 1$,
$T_0 = \delta * T_0$;
end
output S^0,
end

FIG. 2. Simulated annealing algorithm

Before we discuss about *Perturb*() function in the algorithm, we define *Uunit-Perturb*() function and *Perturb-Factor* parameter :

Unit-Perturb() : This function selects one of the clusters and one of its dimensions randomly and changes the corresponding value in the seed vector, S to any other value in the range [1, *M*].

Perturb-factor is the number of unit perturbations required on the solution vector, S.

Perturb() function performs *Unit-Perturb*() function over solution vector, S, for *Perturb-Factor* times. At high temperatures, *Perturb-Factor* is more and as temperature decreases, it is also decreased. The scheme we adopted is given by

$$Perturb\text{-}Factor = \left[\frac{(No\text{-}Ite\text{-}p)}{No\text{-}Ite} * (No\text{-}Perturbs - 1) + 0.5 \right],$$ where *No-Ite*, *p* are maximum number of outer loop executions and current outer loop iteration number respectively. *No-Perturbs* is the value specified by user and depends on the problem. Polynomial time cooling schedule is considered in this study[7].

Initial Temperature — Initial temperature, T_0, has to be estimated correctly. Let X be the desired inital acceptance probability[7] and $m1$, $m2$ be the number of perturbations for which reduction in objective function value and increase in objective function value are observed respectively. According to the algorithm, $m2$ perturbations are accepted with the probability $exp(-\Delta E/T_0)$. So the acceptance ratio will be

$$X = m1 + m2 * \exp(-(\Delta E)/T_0)/(m1 + m2). \quad \ldots (3)$$

From this equation T_0 is

$$T_0 = \Delta E^+/\ln(m2/m2 * X - (1 - X) * m1)) \quad \ldots (4)$$

where ΔE^+ is the average value of all increments in objective function value due to $m2$ moves.

Final Temperature — Final temperature should be small and is selected using the following equation.

$$T_f = -\beta \Delta E^+/\log \theta$$

In this study we use $\beta = 0.1$, $\theta = 0.00001$.

Temperature Decrement — The temperature decrement parameter δ is computed from, *No-Ite* and final temperature, T_f, as shown below :

$$\delta = \sqrt[No\text{-}Ite]{\frac{T_f}{T_0}}$$

5. Complexity of the Algorithm

It is very straight forward to estimate the complexity of the algorithm presented in Section 3. Here our concern is to find the number of *K*-means algorithm runs required for computing the cost of solutions. From this stand point, we can estimate the number of objective function evaluations required.

Number of inner loop iterations : *Max-Ite*;

Number of outer loop iterations : *No-Ite.*

It can easily be found that a total number of *No-Ite* * *Max-Ite* *K*-means algorithm runs are required. Computational complexity of single *K*-means run is $O(ndCK_g)$ where n is the number of data items, d is the number of dimensions, C is the number of clusters and K_g is the number of iterations required for *K*-means algorithm to converge to local optimal partition. Generally *K*-means algorithm converges in few iterations (< 10). So an upper bound can be assumed for K_n to analyze the algorithms complexity. Now the total complexity turns out to be $O(ndCK_nN_k)$ where $N_k = No\text{-}Ite * Max\text{-}Ite$.

6. Experimental Study and Results

In our study, we used many data sets for testing the proposed scheme and in most of the cases it converged to optimal clusters for data sets having well separated clusters. Here we present our experimental results obtained using two data sets. One is the British Towns Data,[9] and the other is artificial data with 500 samples and 50 clusters.

Both these data sets have been used in experimenting with our approach and a comparision is made with the standard *K*-means algorithm which is run with random selection of initial seed values from the data. It is observed that if the number of clusters we test is smaller, then *K*-means algorithm is also reaching (near) optimal solution after selected number of runs. The *K*-means algorithm was run many times as the number of objective function evaluations performed in simulated annealing and best out of those results was considered.

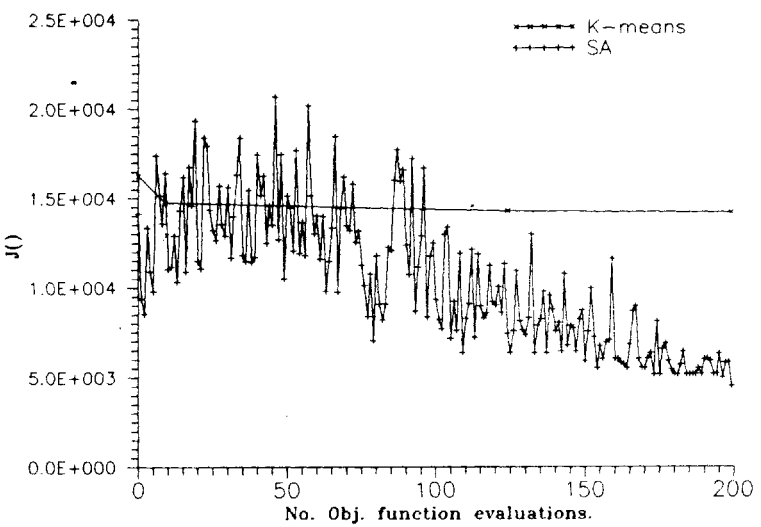

Fig. 3. Comparison of simulated annealing and *K*-means algorithms performances

Table 1

Data	No. Clusters	Simulated Annealing (J)	K-Means (J)	No. Obj. Fun. Eval.s
British Towns	6	141.46	143.19	75
Data	8	113.5041	116.66	100
	10	92.6827	100.59	100
Simulated Data	50	4497.1474	14494.06	200

We used the British Towns data, in order to test the effectiveness of the SA over the standard *K*-means algorithm as the numebr of clusters,i.e. the value of C, increases. Table 1 lists the minimum objective function values obtained for 6, 8, and 10 clusters respectively. As the number of clusters increase the performance of standard *K*-means algorithm decreases to find the (near) optimal solution using random initial seed values.

We generated an artifical data set with 50 non-overlapping clusters and each cluster having 10 samples. Fig. 3. shows how simulated annealing is able to reach global minimum in this large problem with the search space of size $3.4 * 10^{43}$. Here we plot number of objective function evaluations vs. error value (J). For *K*-means algorithm best available result till that run is taken as its representative. The value of *M* also plays an important role in the search. As the value of *M* increases, search space increases exponentially. We used $M = 20$ in our study.

7. CONCLUSIONS

We have introduced a way of finding optimal clusters characterized by the extrema of the objective function using simulated annealing. Several data sets have been tested and results are encouraging. In our formulation, search space is dependent on *M* and *C*. In most of the problems moderate value of *M* is sufficient ($M = 20$ to 30). Here the search space is not dependent on the size of the data set, *N*. Large data sets consume much time to execute *K*-means algorithm itself. In such cases searching for optimal cluster centers is recommended. Testing with various adaptive cooling schedules and different perturbation schemes is under progress.

REFERENCES

1. M.R. Anderberg, (1973), Cluster Analysis for Applications. Academic Press, London.
2. R.O. Duda and P.E. Hart (1973), Pattern Classification and Scene Analysis. Wiley, New York.
3. A. K. Jain, and R. C. Dubes (1988), Algorithms for Clustering Data. Prentice-Hall, Engle-wood, Cliffs, NJ.
4. W. Raymond Klein and R. C. Dubes (1989). Experiments in Projection and Clustering by Simulated Annealing. Pattern Recognition, Vol. 22, Bo.2 (1989), pp. 213-220.
5. Yi-tzuu Chien, (1978), Interactive Pattern Recognition. Marcel Dekker, Inc. New York.

6. S. Kirkpatrick, C. D., Gelatt Jr. and M. P. Vecchi, Optimization by simulated annealing, Science, N.Y.220, p. 671-680(1983).
7. P. J. M. van Laarhoven and E. H. L. Aarts, Simulated Annealing: Theory and Applications, D. Reidel, Hinggham, MA(1987).
8. D. Vanderbilt and S. G. Louie, A Monte Carlo approach to optimization over continuous variables, J. Comput. Phys. 56, p.259-271(1984).
9. S. Z. Selim and K. Alsultan, A simulated Annealing algorithm for the clustering problem, Pattern Recognition, Vol. 24, No. 10, pp. 1003-1008, 1991.
10. S. Z. Selim and M. A. Ismail, *K*-means type algorithms: generalized convergence theorem and characterization of local optimality, *IEEE Trans. Pattern Annal. Mach. Intell.* 6, p.81-87, 1984.
11. E. H. L. Aarts, J. H. M. Korst and P. J. M. Van Laarhoven, A quantitative analysis of the simulated annealing algorithm: a case study for traveling salesman problem, Journal of Statistical Physics 50, pp.189-206, 1988.
12. W.-M. Chen, Y.-X. Wong and X. Ping, Flow-shop scheduling by the knowledge of statistical mechanics and annealing, Proc. 26th IEEE Conf. on Decision and Control 1, Los Angeles, pp. 642-643, 1987.
13. S. Geman and D. Geman, Stochastic relaxation, Gibbs distribution, and the Bayasian restoration of images, IEEE Proc. of Pattern Analysis and Machine Intelligence, pp. 721-741, 1984.
14. G. P. Babu and M. N. Murty. "A note on Evolutionary Programming for cluster analysis", working paper, CSA, IISc.
15. G. P. Babu and M. N. Murty. "Clustering with Evolution Strategies", submitted to *Pattern Recognition Journal.*
16. K. Binder, *Monte Carlo methods in statistical Physics.* Springer-Verlag, New York.
17. G. P. Babu and M. N. Murty. "Neural Networks and Clustering", sumbitted to *IEEE trans. on Neural Networks.*
18. H. John Holland, *Adaptation in natural and artifical systems.* The University of Michigan Press, Ann Arbour, 1975.
19. V. V. Raghavan and K. Birchand, A clustering strategy based on a formalism of the reproduction process in natural system., *SIGIR Form,* vol. 14, pp.10-22, (1979).
20. H.-P. Schwefel, *Numerical Optimization of computer models.* John Wiley, New York, London, (1981).
21. L. J. Fogel, A. J. Owens and M. J. Walsh, (1966), *Artificial Intelligence through Simulated Evolution,* John Wiley, NY.