Proceedings of the 2006 American Control Conference
Minneapolis, Minnesota, USA, June 14-16, 2006

A Scalable Deterministic Annealing Algorithm for Resource Allocation Problems

Puneet Sharma, Srinivasa Salapaka and Carolyn Beck

Abstract— **In this paper, we develop a scalable algorithm for solving resource allocation problems on large datasets. The algorithm is based on the deterministic annealing (DA) algorithm presented in [6]. The capability of the DA algorithm to identify clusters at successive iteration steps is exploited to truncate certain large computations in the algorithm. These truncations are obtained by recursively grouping the clusters into appropriate groups and running the DA algorithm, in parallel, on the groups. This paper develops a notion of *interaction* between groups which is used to measure the deviation of this algorithm from the DA algorithm. Simulations are presented that show significant improvements in the computational time while keeping the error in resource allocations within prespecified tolerance limits.**

I. INTRODUCTION

Many problems which require selection of a subset from a given population can be viewed as resource allocation problems, often referred to as *locational optimization* problems. Locational optimization algorithms arise in a number of contexts in control, for example, motion coordination algorithms, coverage control [7] and mobile sensing networks [1]. These problems share the fundamental goal of aiming to determine an optimal partition of the underlying domain in which they are defined (e.g., a library of compounds for drug discovery, an unknown area of interest for coverage control), and an optimal assignment of values, or elements, from a finite resource set to each *cell* in the partition space.

Computationally, these problems are typically complex and time intensive if not intractable. For example, in the problem of drug discovery, determining 30 representative compounds from an array of 1000 compounds results in approximately 3×10^{25} possibilities. Another factor which adds to the complexity of such problems is their inherent non-convex nature. Thus we require an efficient algorithm that does not get stuck in local minima.

Although all the aforementioned locational optimization problems share similar basic optimization goals, there are a number of features and criteria which are specific to each problem and thus distinguish one from the other. These differentiating characteristics include different distance metrics in the definition of *coverage*, number and types of constraints on the resources in the problem formulation, the necessity of computing global versus local optima, the possibility of elements and resources that exhibit dynamical behavior, and the size and scale of the feasible domain. For example, in the drug discovery problem [8], there are scenarios in which capacity constraints are placed on the experimental resources, thus leading to a *multi-capacity constraint problem* similar to one which arises in the optimal strategic positioning of UAVs [7]. Motion control problems are satisfactorily solved by local optimization solutions, i.e., *distributed* coordination algorithms utilizing only nearest-neighbor information, which are more relevant and are preferred to global coordination schemes, whereas in the drug discovery case we are primarily interested in global solutions.

In [8], we showed that *deterministic annealing (DA)*[6] provided a good framework for catering to the specific constraints and demands of combinatorial chemistry. It simultaneously addressed the key issues of 'diversity' and 'representativeness' in the chosen lead generation library. Due to the inherently large size of the present day combinatorial libraries, scalability is one of the key issues that needs to be addressed by the algorithm. The DA algorithm presented in [8] does not scale efficiently because the number of computations grow exponentially with the size of the lead generation library.

In this paper, we exploit features of the DA algorithm to lower the computational expense. One of the features of DA algorithm is that, as it proceeds, it identifies increasingly finer clusters. We recursively exploit this feature to approximate certain computations in the algorithm. These approximations are obtained by recursively grouping the clusters into appropriate groups and running the DA algorithm, in parallel, on these groups. The approximations are better when these groups are more separated. One of the main contributions of this paper is to quantify the *interaction* between various clusters in the underlying domain and then identify groups (sets of clusters) that are separated from one another. This separation is defined in terms of inter-cluster interactions. That is, if the level of interaction between two regions is below a certain threshold, the regions are assumed separate, and effectively, resource location(s) in these regions is determined by application of the DA algorithm to individual groups. We also characterize the inherent trade off between reduction in computation time and error in the resource locations.

This paper is organized as follows. In Section II we state the optimization problem and discuss the basics of the DA algorithm. Section III contains a thorough discussion of the proposed algorithm. The notions of cluster interaction and separation, and trade-off between error and computation time

This work has been partially funded by AFOSR URI grant F49620-01-1-0365 and NSF ITR Collaborative Research grant ECS-0426831.
The first and third authors are with the Coordinated Science Lab and the Department of General Engineering and the second author is with the Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign.
Emails: psharma2@uiuc.edu, salapaka@uiuc.edu, beck3@uiuc.edu

1-4244-0210-7/06/$20.00 ©2006 IEEE

are discussed in this section. Section V provides the simulation results and comparisons with the original algorithm. Finally we conclude the paper by revising the important results obtained in this paper and identifying the future goals.

II. DETERMINISTIC ANNEALING ALGORITHM

The fundamental resource allocation problem is stated as:

Given a distribution $p(x)$ of the elements x in a descriptor space Ω, find the best *set of M resource locations r_j that solves the following minimization problem:*

$$\min_{r_j,\ 1\le j\le M} \int_\Omega p(x) \left\{ \min_{1\le j\le M} d(x, r_j) \right\} dx. \quad (1)$$

Here $d(x, r_j)$ represents an appropriate *distance* metric between the resource location r_j and the element x. Alternatively, this problem can also be formulated as finding an *optimal* partition of the descriptor space Ω into M cells R_j and assigning to each cell R_j a resource location r_j such that the following cost function is minimized

$$\sum_j \int_{R_j} d(x, r_j) p(x) dx.$$

The DA algorithm can be viewed as a modification of Lloyd's algorithm [3], [2], in which the initial step consists of randomly choosing resource locations and then successively iterating between the steps: (1) forming Voronoi partitions, and (2) moving the resource locations to respective centroids of cells until the sequence of resource locations converge. It should be noted that the solution depends substantially on the initial allocation as in the successive iterations the locations are influenced only by 'near' points of the domain and are virtually independent of 'far' points. As a result the solutions from this algorithm 'typically' get stuck to local minima.

The DA algorithm [5], [6] eliminates this local influence of domain elements by allowing each element $x \in \Omega$ to be associated with every resource location r_j through a weighting parameter $p(r_j|x)$. Thus this algorithm eliminates the hard partitions of Lloyd's Algorithm. The DA formulation includes a modified distortion term

$$D = \int_\Omega p(x) \sum_j d(x, r_j) p(r_j|x) dx,$$

and an entropy term

$$H = -\int_\Omega p(x) \sum_j p(r_j|x) \log p(r_j|x) dx,$$

which measures the randomness of the distribution of associated weights. Entropy is largest when the distribution of weights over each resource location is identical, i.e. $(p(r_j|x) = 1/M)$ for each x, i.e., when all x have the same influence over every resource location. This algorithm solves the following optimization problem

$$\min_{r_j} \min_{p(r_j|x)} \underbrace{D - T_k H}_{:=F}$$

at the kth iteration where T_k is a parameter called *temperature* which tends to zero as k tends to infinity. The cost function F is called *Free Energy* as this formulation has a parallel in statistical physics [4]. Clearly for large values of T_k, we mainly attempt to maximize the entropy. As T_k is lowered we trade entropy for the reduction in distortion, and as T_k approaches zero, we minimize D directly to obtain a hard (non random) solution. Minimizing the Free Energy term F with respect to association probabilities $p(r_j|x)$ is straightforward and gives the *Gibbs* distribution

$$p(r_j|x) = \frac{e^{-d(x,r_j)/T}}{\sum_i e^{-d(x,r_i)/T}}. \quad (2)$$

The corresponding minimum of F is obtained by substituting for $p(r_j|x)$ from Equation 2,

$$\hat{F} = -T \int_\Omega p(x) \log Z. \quad (3)$$

To minimize $\hat{F}$ with respect to resource locations $\{r_j\}$, we set the corresponding gradients equal to zero $(\frac{\partial \hat{F}}{\partial r_j} = 0)$; this yields the corresponding implicit equations for the locations.

$$r_j = \int_\Omega p(x|r_j) x dx \quad 1 \le j \le M \quad (4)$$

where

$$p(x|r_j) = \frac{p(x) p(r_j|x)}{\int_\Omega p(x') p(r_j|x') dx'}. \quad (5)$$

Note that $p(x|r_j)$ denotes the posterior probability calculated using Bayes's rule and the above equations clearly convey the 'centroid' aspect of the solution.

The DA algorithm consists of minimizing $\hat{F}$ with respect to $\{r_j\}$ starting at high values of T_k and tracking the minimum of F while lowering T_k. The steps at each k are:

1) fix $\{r_j\}$ and use Equation 2 to compute the new weights $\{p(r_j|x)\}$.
2) fix $\{p(r_j|x)\}$ and use Equation 4 to compute the new resource locations $\{r_j\}$.

III. SCALABLE DA ALGORITHM

As noted earlier, one of the major problems with combinatorial optimization algorithms is that of scalability, i.e. the number of computations scales up exponentially with an increase in the amount of data. This characteristic is a deterrent to using this algorithm for large (complex) problems. For example, in the process of drug discovery, combinatorial libraries (which consist of extremely large collections of chemical compounds) are used to select subsets of representative compounds for lead generation library design. One of the major deterrents in this selection problem is the huge size of the initial combinatorial library. Thus, in order to address the issue of scalability efficiently, it is necessary to amend the DA algorithm.

Due to the inherent nature of the DA algorithm, the farther an individual data point is from a cluster, the lower is its influence on the cluster. This is evident from the Gibb's distribution laws of the association probabilities $p(r_j|x)$. Hence two clusters that are far apart have very small interaction between them. Thus if we ignore the effect of a separated cluster on the remaining data-points, the resulting error will

not be significant. Here it should be noted that ignoring the effects of separated regions on one another will result in a considerable reduction in the number of calculations (since the points that constitute a separated cluster will not contribute to the distortion and entropy calculations for the remaining points). Thus identification of separated regions in a data-set enables us to process the algorithm much more quickly for large data sets.

The first step required to identify separated regions is to characterize and quantify the level of interaction that exists between the various clusters. The next step is to group together sets of clusters amongst which significant interaction exists, but which have significantly small interactions with other clusters (not in this group). Once these groups of *separate* clusters are identified, the final step is to modify the DA algorithm such that it ignores the effects of separate groups on one another. This modification greatly reduces the calculations to be performed by the algorithm.

A. Cluster Interaction and Separation

In order to characterize the interaction between different clusters, it is necessary to consider the mechanism of cluster formations during the processing of the DA algorithm. As the temperature (T) is reduced, the system undergoes a series of 'phase transitions'[6]. In this case 'phase transition' refers to the splitting of resource locations and the nearest neighbors identify natural clusters in the data as the algorithm proceeds. Successive phase transitions identify finer clusters. Thus an increase in the number of clusters results in these natural splits. This provides us with a tool to control the number of clusters we want from the algorithm. In [6], it is shown that a cluster R_i splits when twice the maximum eigenvalue of the posterior covariance matrix, defined by $C_{x|r_i} = \sum_{x\in\Omega} p(x)p(x|r_i)(x-r_i)(x-r_i)^T$ becomes greater than the temperature value. The temperature at which the split occurs is called the critical temperature ($T_c = 2\lambda_{max}[C_{x|r_i}]$).

In the DA algorithm, the location of a cluster is primarily determined by the data points near to it (with far-away points exerting small influence). The association probabilities $p(r_j|x_i)$ determine the level of interaction between the cluster r_j and the data point x_i. This interaction level decays exponentially with the increase in the 'distance' between r_j and x_i. The total interaction exerted by all the data-points in a given space determines the relative weight of each cluster,

$$p(r_j) =: \sum_{x\in\Omega} p(x, r_j) = \sum_{x\in\Omega} p(r_j|x)p(x) \tag{6}$$

where $p(r_j)$ denotes the weight of cluster R_j.

The value $\sum_{x\in R_i} p(r_j|x)p(x)$ is equal to the level of interaction that data-points in cluster R_i exert on cluster R_j. The higher this value is, the more interaction exists between clusters R_i and R_j. This gives us an effective way to characterize the interaction between various clusters in a dataset. In a probabilistic framework, this interaction can also be interpreted as probability of transition from R_i to R_j.

Consider the $m \times m$ matrix ($m \leq M$)

$$A = \begin{pmatrix} \sum_{x\in R_1} p(r_1|x)p(x) & .. & \sum_{x\in R_m} p(r_1|x)p(x) \\ \sum_{x\in R_1} p(r_2|x)p(x) & .. & \sum_{x\in R_m} p(r_2|x)p(x) \\ \vdots & \ddots & \vdots \\ \sum_{x\in R_1} p(r_m|x)p(x) & .. & \sum_{x\in R_m} p(r_m|x)p(x) \end{pmatrix}$$

In a probabilistic framework, this matrix can be considered a finite dimensional Markov operator, with the term $A_{j,i}$ denoting the transition probability from region R_i to R_j. Figure 1 shows the transition probabilities of the associated Markov chain. The higher the transition probability, the greater is the amount of interaction between the two regions.

Once the transition matrix is formed, the next step is to

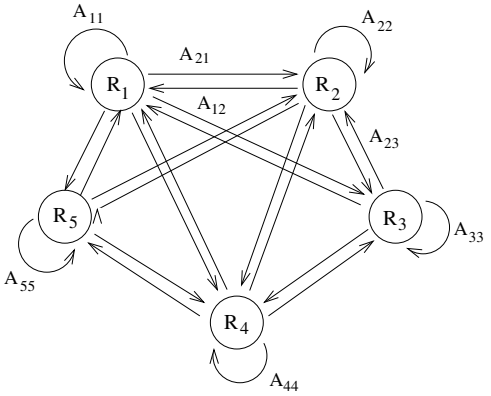


Fig. 1. Markov chain

identify regions (groups of clusters) which are separate from the rest of the data. The separation is characterized by the quantity ϵ. Thus a cluster (R_j) is *ϵ-separate* if the level of its interaction with each of the other clusters ($A_{j,i}, i = 1, 2, \ldots n, i \neq j$) is less than ϵ. Note that ϵ determines the number of separate sets formed (if any), which in turn decides the error in distortion due to the proposed algorithm with respect to that of the original DA algorithm.

B. Trade off between error in resource location and computation time

As was discussed in Section III, the more separate regions we have, the smaller the computation time for the modified algorithm. At the same time, more separate regions result in higher deviation of the distortion of the proposed algorithm from the DA algorithm. This trade-off between reduction in computation time and increase in distortion error is systematically addressed below.

From the DA algorithm, the location of the cluster (r_i) is determined as follows.

$$r_i = \frac{\sum_{x\in\Omega} xp(x)p(r_i|x)}{\sum_{x\in\Omega} p(x)p(r_i|x)} = \frac{G(\Omega, i)}{H(\Omega, i)} \tag{7}$$

where the following notation is used,

$$G(V, j) = \sum_{x\in V} xp(x)p(r_j|x) \tag{8}$$

$$H(V, j) = \sum_{x\in V} p(x)p(r_j|x). \tag{9}$$

Since the cluster Ω_i is separated from all the other clusters, the resource location $r_i^{'}$ will be determined by the modified algorithm as follows:

$$r_i^{'} = \frac{\sum_{x\in\Omega_i} xp(x)p(r_i|x)}{\sum_{x\in\Omega_i} p(x)p(r_i|x)} = \frac{G(\Omega_i,i)}{H(\Omega_i,i)}. \quad (10)$$

Also note that

$$\begin{aligned} G(\Omega_i,i) &= G(\Omega,i) - G(\Omega_{-i},i) \\ H(\Omega_i,i) &= H(\Omega,i) - H(\Omega_{-i},i) \end{aligned} \quad (11)$$

where, $\Omega_{-i} = \Omega \setminus \Omega_i$. Therefore,

$$r_i - r_i^{'} = \frac{G(\Omega,i)}{H(\Omega,i)} - \frac{G(\Omega_i,i)}{H(\Omega_i,i)}, \quad (12)$$

which is the component-wise difference between r_i and $r_i^{'}$. On simplifying the terms and substituting from (11), we have

$$|r_i - r_i^{'}| \leq \frac{\max(G(\Omega_{-i},i)H(\Omega_i,i), G(\Omega_i,i)H(\Omega_{-i},i))}{H(\Omega,i)[H(\Omega,i) - H(\Omega_{-i},i)]}. \quad (13)$$

In order to find a bound on the error $(r_i - r_i^{'})$, note that

$$\begin{aligned} G(\Omega_{-i},i) &= \sum_{x\in\Omega_{-i}} xp(x)p(r_i|x) && (14) \\ &\leq H(\Omega_{-i},i) \sum_{x\in\Omega_{-i}} x && (15) \\ &\leq M_{-i}NH(\Omega_{-i},i) && (16) \end{aligned}$$

where, $M_{-i} = \frac{1}{N}\sum_{x\in\Omega_{-i}} x$. Here we assume that $x > 0$ without any loss of generality (since the resource allocation problem definition is independent of translation or scaling factors). Thus,

$$\begin{aligned} |r_i - r_i^{'}| &\leq \frac{\max[M_{-i}NH(\Omega_i,i), G(\Omega_i,i)]H(\Omega_{-i},i)}{H(\Omega,i)[H(\Omega,i) - H(\Omega_{-i},i)]} \\ &\leq \max\left[M_{-i}N, \frac{G(\Omega_i,i)}{H(\Omega_i,i)}\right]\frac{H(\Omega_{-i},i)}{H(\Omega,i)}; \quad (17) \end{aligned}$$

$$\frac{|r_i - r_i^{'}|}{M} \leq \max\left[\frac{M_{-i}}{M}N, \frac{r_i^{'}}{M}\right]\frac{H(\Omega_{-i},i)}{H(\Omega,i)}; \quad (18)$$

$$\frac{|r_i - r_i^{'}|}{MN} \leq \max\left[\frac{M_{-i}}{M}, \frac{M_i}{M}\right]\eta_i \quad (19)$$

where,

$$\eta_i = \sum_{j\neq i}\epsilon_{ji} / \sum_j \epsilon_{ji}, \quad (20)$$

and the level of interaction (ϵ_{ji}) between cluster Ω_i and Ω_j is

$$\epsilon_{ji} = \sum_{x\in\Omega_i} p(x)p(r_j|x).$$

For a given data-set, the quantities M, M_i and M_{-i} are known apriori. For the error in resource location $|r_i - r_i'|/M$ to be less than δ_i $(\forall\delta_i > 0)$, we should choose η_i such that

$$\eta_i \leq \frac{\delta_i}{N\max\left[\frac{M_{-i}}{M}, \frac{M_i}{M}\right]}. \quad (21)$$

IV. ALGORITHM

The modified algorithm is as follows:

- Step 1: Initiate DA algorithm and determine resource locations together with the association probabilities.
- Step 2: If a split occurs, identify individual clusters and use the association probabilities $(p(r_j|x))$ to construct the transition matrix.
- Step 3: Use the transition matrix to identify separated clusters and group them together to form separated regions. Ω_k will be separated from Ω_j if the quantities $A_{j,k}$ and $A_{k,j}$ are less than a chosen ϵ_{jk}.
- Step 4: Apply the DA algorithm to each region Ω_k, neglecting the effect of separate regions on one another.
- Step 5: Stop if the stopping criterion (such as number of resource locations (M), or computation time etc.) is met, otherwise go to Step 2.

Identification of separate regions in the underlying data provides us with a tool to efficiently scale the DA algorithm. A reduction in the amount of data used for clustering results in exponential savings in computations to be performed. At the same time, the number of such DA algorithms to be performed scales linearly with the number of separated regions formed. Thus the exponential gain prevails over the linear loss thereby resulting in an efficient algorithm to address the issue of scalability.

V. SIMULATIONS AND RESULTS

In order to demonstrate the efficiency of the modified algorithm, we have compared it with the DA algorithm presented in [8]. The three criteria for comparison are:

- Time taken and final distortion achieved when same number of resources are allotted to both algorithms.
- Final distortion achieved when both the algorithms run for the same amount of time.
- Largest size of data set that the algorithms can solve.

A. Case 1: Same Number of Resources

As was pointed out earlier, the identification of separate regions speeds up the algorithm (as it requires lesser amount of computation). In order to demonstrate this fact, the algorithm was tested on many different datasets. For the purpose of simulation, datasets were created in the following manner.

The first set was obtained by identifying ten random locations in a square region of size 400×400. These locations were then chosen as the cluster centers. Next, the size of each of these clusters was chosen and all points in the cluster were generated by a normal distribution of randomly chosen variance. A total of 5000 points comprised this data set. Figure 2 shows the data-set and the resource locations obtained by the old DA algorithm. The blue crosses denote the resource locations and the pie-chart at the bottom gives the relative weight of each resource. The modified DA algorithm starts with one cluster at the centroid of the data set. As the temperature is reduced, the clusters are split and separated regions are determined at each such split. Figure 3 shows the four separate regions identified

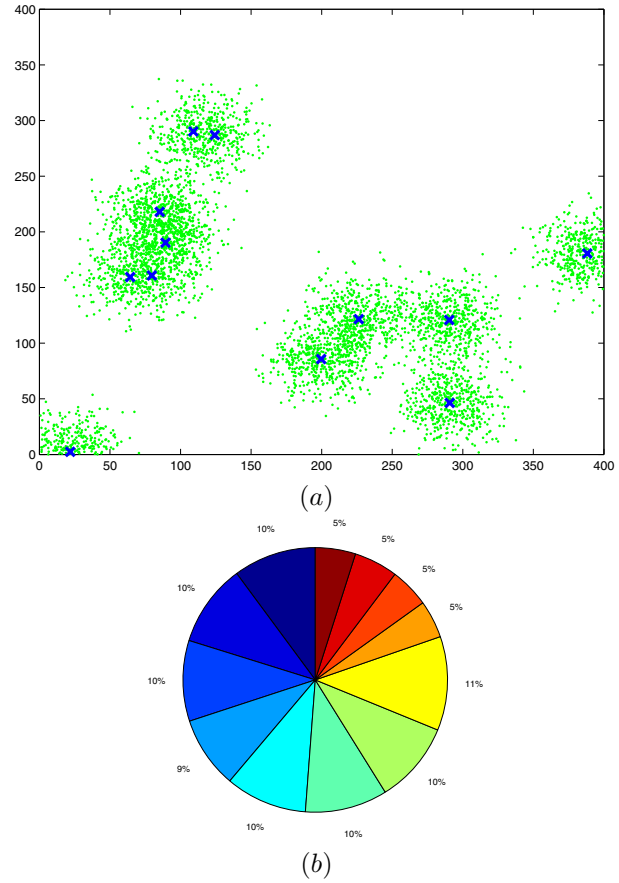


Fig. 2. (a.) Simulated data-set (green dots) and resource locations (blue crosses) determined from DA algorithm (b.) Relative weights of resources

by the algorithm (as described in Section III-A) at the instant when 12 resources have been allotted. Figure 4 offers a comparison between the two algorithms. Here, the blue crosses represent the resource locations determined by the DA algorithm presented in [8]. The circles represent the locations determined by the modified algorithm that we have proposed. Note that the data-set was partitioned into four separated regions (represented by the four colors of the circles). As can be seen from the figure, there is not much difference between the locations obtained by the two algorithms. The main advantage of the modified algorithm is in terms of the computation time of the algorithm. The results from the two algorithms have been tabulated below.

Algorithm	Distortion	Computation Time (sec)
DA	300.80	129.41
Scalable DA	316.51	21.53

As can be seen from the table, the modified algorithm uses just one-sixth of the time taken by the original DA algorithm and results in only a 5.2% error in terms of distortion. This was obtained for $\epsilon = 0.005$. Both the algorithms were terminated when the number of resources reached 12. It should be noted here that in case of the modified algorithm, the computation time can be further reduced (by changing ϵ), but at the expense of error in distortion.

Results obtained from another dataset are presented in the

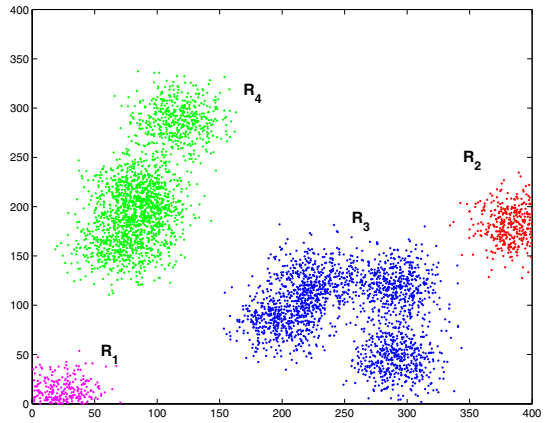


Fig. 3. Separated regions R_1, R_2, R_3 and R_4 (denoted by different colors) as determined by the scalable DA algorithm

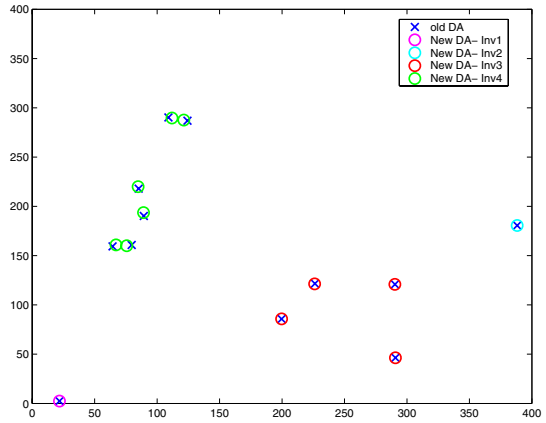


Fig. 4. Comparison of resource locations obtained from two algorithms

table below. This set had three clusters which were distinctly separated from each other, containing a total of 9000 points.

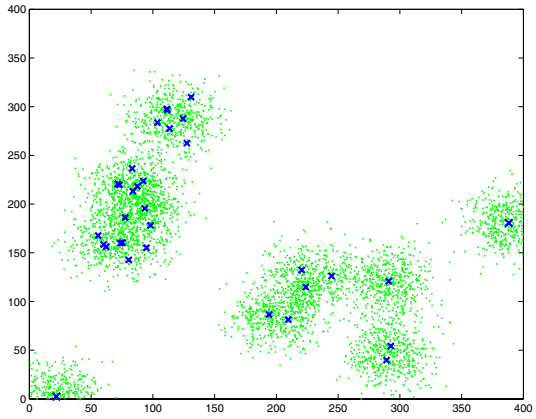
Algorithm	Distortion	Computation Time (sec)
DA	1480.9	887.21
Scalable DA	1593.1	148.14

As seen in the table, the original DA algorithm uses 887 seconds to determine 36 resource locations in the underlying dataset. On the other hand, the scalable DA algorithm uses just 148 seconds and results in a 7.6% error in distortion.

B. Case 2: Same Amount of Computation Time

The next comparison between the two algorithms is to demonstrate the efficiency of the modified algorithm in terms of number of resources allotted. For the purpose of simulation, the same data-set (Case 1) was used. Here, both the algorithms were executed on the given data set for the same amount of time. Figure 5 shows the resource locations as computed by the modified algorithm. The resource locations computed by the original algorithm have already been shown in Figure 2. As can be seen from Figure 5, the top left cluster has been allotted 23 resources. Here it should be noted that the relative weight of these resources is small when compared to other resources. This relative weight is automatically calculated by the DA algorithm. The pie

chart at the bottom shows the relative importance of each resource location. The two dark blue regions on the pie chart



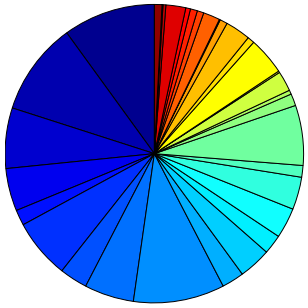


Fig. 5. (a) Resource locations determined by modified DA (b) Relative weights of resources

approximately represent 9% weight each. This corresponds to the two clusters which have been allotted one resource each. The results obtained have been tabulated below

Algorithm	Resources	Distortion
DA	12	300.80
Scalable DA	33	260.08

Both the algorithms were executed for 129 seconds. As can be seen from the table, the modified algorithm results in a lower value for the distortion. There is a 13.5% reduction in distortion for the same amount of computational time. This is due to the fact that the modified algorithm was able to allot a higher number of resources in the given time. A total of 33 resources were used by the modified algorithm. On the other hand, the original DA algorithm could only allot 12 resources in the given dataset. The separated regions are the same as before (Figure 3). As was expected, the identification of separated regions reduces the number of computations to be performed, which in turn increases the number of resource locations that can be computed in a given amount of time.

C. Case 3: Large Data-sets

The scalable DA algorithm not only speeds up the processing of the algorithm, but also lets us solve datasets with large numbers of points. In order to demonstrate this, the following dataset was created. Three cluster locations were identified (as shown in Figure 6) and the resulting system was solved for different cluster sizes. Simulations were carried

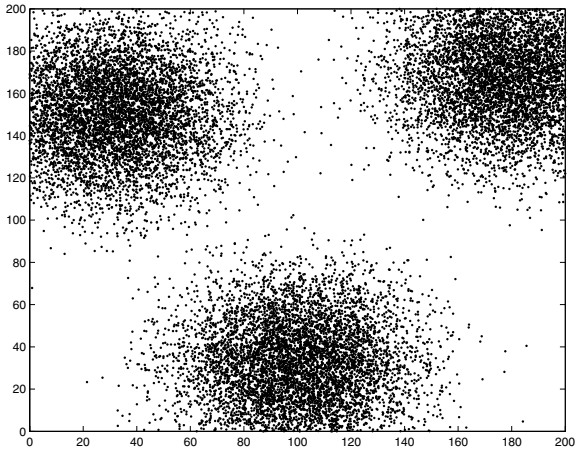


Fig. 6. Simualted Data-set containing 30000 points

out in MATLAB on a 1.5 GHz Centrino processor with 512 MB RAM. It was observed that the DA algorithm did not successfully terminate when the total number of points in the data-set was more than 15000 (with 36 resource locations). On the other hand, the scalable algorithm performed efficiently while identifying 36 resource locations in a data-set containing 40000 points.

VI. CONCLUSIONS

In this paper, we have presented a modified version of the DA algorithm. This modified algorithm effectively addresses the issue of scalability in the original DA algorithm. It is based on recursively identifying separated regions in a dataset (for different number of resource locations), and using this information to reduce the number of computations. The interaction between various clusters is characterized and an analysis is presented to address the trade-off between error and computation time for the modified algorithm. Simulation results were presented to demonstrate the efficiency of the modified algorithm, and comparisons with the original algorithm were made to further highlight this fact.

REFERENCES

[1] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage Control for Mobile Sensing Networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[2] A. Gersho and R. Gray. *Vector Quantization and Signal Compression*. Kluwer, Boston, Massachusetts, 1st edition, 1991.

[3] S.P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[4] K. Rose. Statistical mechanics and phase transitions in clustering. *Physics Review Letters*, 65(8):945–948, 1990.

[5] K. Rose. Constrained Clustering as an Optimization Method. *IEEE Trans. Pattern Anal. Machine Intell.*, 15:785–794, August 1993.

[6] K. Rose. Deterministic Annealing for Clustering, Compression, Classification, Regression and Related Optimization Problems. *Proceedings of the IEEE*, 86(11):2210–39, November 1998.

[7] S. Salapaka and A. Khalak. Constraints on Locational Optimization Problems. *Proc. IEEE Control and Decision Conference*, pages 1741–1746, December 2003.

[8] P. Sharma, S. Salapaka, and C. Beck. A deterministic annealing approach to combinatorial library design for drug discovery. *Proc. of American Control Conference*, pages 979–984, 2005.