Proceedings of the 2007 American Control Conference
Marriott Marquis Hotel at Times Square
New York City, USA, July 11-13, 2007

A Maximum Entropy Based Scalable Algorithm for Resource Allocation Problems

Puneet Sharma, Srinivasa Salapaka and Carolyn Beck

Abstract— **In this paper, we propose a scalable algorithm for solving resource allocation problems on large datasets. This class of problems is posed as a multi-objective optimization problem in a *Maximum Entropy Principle* framework. This algorithm solves a multi-objective optimization problem that minimizes simultaneously the coverage cost and the computational cost by appropriate recursive prescription of smaller subsets required for a 'divide and conquer' strategy. It provides characterization of the inherent trade-off between reduction in computation time and the coverage cost. Simulations are presented that show significant improvements in the computational time required for solving the coverage problem while maintaining the coverage costs within prespecified tolerance limits.**

I. INTRODUCTION

This paper studies resource allocation problems which require selection of resources (and their locations) for a given set which minimize a *coverage* cost. These problems are also referred to as *locational optimization* problems. Locational optimization algorithms arise in a number of contexts in control, for example, motion coordination algorithms, coverage control [9] 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 2.43×10^{57} possibilities. This combinatorial complexity of such problems is further complicated by their inherent non-convex nature. This makes it important to design algorithms that do not get trapped 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 [11], 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 [9]. 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 drug discovery, we are primarily interested in global solutions.

In [11], we first proposed the use of the *deterministic annealing (DA)* algorithm [7], [8] for catering to the specific constraints and demands of combinatorial chemistry, and demonstrated its effectiveness. The algorithm simultaneously addresses 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. The algorithm presented in [11] does not scale efficiently because the number of computations grows combinatorially with the size of the lead generation library. At the same time, it does not provide any characterization of the trade-off between computational time and error in determining optimal resource locations. In [12], we presented a scalable version of the DA algorithm for solving the basic resource allocation problem and highlighted the inherent trade-off between computation time and accuracy.

In this paper, we develop a framework under which the computational complexity and the inherent non-convexity of these problems are addressed. The framework provides a way of recursively dividing the dataset into smaller data sets and applying repeatedly an algorithm (similar to the DA algorithm) on each of the subsets. This divide and conquer process saves substantial computational expense. The reduction of a given problem to a sequence of subproblems based on finding appropriate subsets is not straightforward as it depends extensively on the distribution of the original dataset. This hurdle is overcome by forming an optimization problem where the computational expense acts as one of the constraints in the problem definition. The algorithm that solves this optimization problem also prescribes how the subsets need to be formed for a given trade-off between the computational expense and the deviation from the solution that acts on the whole set (that is, which does not subdivide the data and therefore does not save computational time). The main principle that the framework relies on is the Maximum Entropy Principle (MEP) developed extensively by E.

This work is partially supported by NSF ITR Collaborative Research grant ECS-0426831 and NSF ECS 0449310 CAR grant. The first and third authors are with the Coordinated Science Lab and the second author is with the Department of Mechanical Science and Engineering, University of Illinois at Urbana Champaign

Emails: psharma2@uiuc.edu, salapaka@uiuc.edu, beck3@uiuc.edu

T. Jaynes [3]. The problem is viewed in a stochastic setting and MEP proves to be a very useful tool for formulating the optimization problem and finding an algorithm that solves it. The main contribution of the paper is the algorithm that automates the computation of the 'divide and conquer rule' (in view of diverse data sets) and at the same time provides tuning parameters that set the trade-off between the computational and coverage costs.

This paper has been organized as follows. We state the basic resource allocation problem in Section II and highlight the key difficulties often encountered while partitioning large datasets. We then present the *Maximum Entropy Principle* [3] in Section III. The proposed scalable algorithm is presented in Section IV. Implementation issues and simulation results are presented in Section V. Finally we conclude the paper by revisiting the most important results obtained herein and identifying future goals.

II. RESOURCE ALLOCATION PROBLEM

The fundamental resource allocation problem is stated as:

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

$$\min_{y_j,\ 1\le j\le M} \sum_i p(x_i) \left\{ \min_{1\le j\le M} d(x_i, y_j) \right\}. \quad (1)$$

Here $d(x_i, y_j)$ represents an appropriate *distance* metric between the resource location y_j and the element x_i. For example a school allocation problem aims at finding locations for M schools in a city where $p(x_i)$ gives the distribution of the students in the city Ω. The cost function, in this case, the average distance between a student in the city and the nearest school, is referred to as the 'coverage cost'. Alternatively, this problem can also be formulated as finding an *optimal* partition of the space Ω into M cells R_j and assigning to each cell R_j a resource location y_j such that the following cost function is minimized

$$\min_{y_j,\ 1\le j\le M} \sum_j \sum_{x_i\in R_j} d(x_i, y_j) p(x_i). \quad (2)$$

The DA algorithm [7], [8] was developed to solve this resource allocation problem. The DA algorithm can be viewed as a modification of Lloyd's algorithm [6], [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. One key feature of the DA algorithm was that it eliminated local influence of domain elements by allowing each element $x \in \Omega$ to be associated with every resource location y_j through a weighting parameter $p(y_j|x)$.

One of the major problems with combinatorial optimization algorithms is that of scalability, i.e. the number of computations scales up combinatorially with an increase in the amount of data. The DA algorithm achieves better solution (in terms of lower cost function by avoiding local minima) with lower computational cost relative to Lloyd type algorithms by an iterative mechanism where resources are allocated recursively based on a tuning parameter called the *temperature variable*. This temperature is changed exponentially and so the results are obtained in a relatively smaller number of iterates. Thus the DA saves on computational time relative to algorithms such as simulated annealing where analogous temperature variables must be changed logarithmically to obtain comparable results to the DA algorithm. However, at each step of the DA algorithm, terms are computed that require computation of all combinations of distances from each resource to every point in the entire data set. In this paper, we present an algorithm that saves computational expense by devising a 'divide and conquer' strategy at each iteration while keeping intact the few-iterations aspect of the DA algorithm. This becomes especially useful in some applications such as drug discovery where the enormity of the data set makes the DA algorithm slow. We use the MEP to design this algorithm which we present in the next section.

III. MAXIMUM ENTROPY PRINCIPLE

In this section, we provide a brief overview of the *Maximum Entropy Principle* (MEP) presented in [3]. The results from this section have been used in Section IV, where the scalable resource allocation problem is posed in a Maximum Entropy framework.

The MEP deals with ascribing a probability mass function for a vector valued random variable x such that it guarantees that the expected values for a given set of m functions of the random variable $f_1(x), f_2(x), \cdots f_m(x)$ are equal to the m given constants $F_1, F_2, \cdots, F_m$, i.e.,

$$\langle f_k(x) \rangle = F_k,\ 1 \le k \le m.$$

Thus we want to find probabilities $(p_1, \ldots, p_n)$ such that

$$F_k = \langle f_k(x) \rangle = \sum_{i=1}^{n} p_i f_k(x_i), \quad 1 \le i \le n \quad (3)$$

where x_i are the values that the random variable x can take.

MEP states that the probabilities should be such that they maximize the Shannon entropy [10] and at the same time satisfy the constraints in (3). Shannon entropy is defined as

$$H(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log(p_i). \quad (4)$$

The entropy in (4) is maximized under the constraints in (3) by considering the Lagrangian given by

$$L = H - (\lambda_0 - 1) \sum_i p_i - \sum_{j=1}^{m} \lambda_j \sum_i p_i f_j(x_i). \quad (5)$$

Setting $\frac{\partial L}{\partial p_i} = 0$, we get $\frac{\partial H}{\partial p_i} - (\lambda_0 - 1) - \sum_{j=1}^{m} \lambda_j f_j(x_i) = 0$. Substituting $\frac{\partial H}{\partial p_i}$ from (4) and $\sum_i p_i = 1$, this implies

$$p_i = \frac{\exp\left\{ -\sum_{j=1}^{m} \lambda_j f_j(x_i) \right\}}{Z(\lambda_1, \ldots, \lambda_m)}, \quad (6)$$

where the partition function Z is defined by

$$Z(\lambda_1, \ldots, \lambda_m) = \sum_{i=1}^{n} \exp\left\{ -\sum_{j=1}^{m} \lambda_j f_j(x_i) \right\}. \quad (7)$$

Equation (6) defines the probability mass function that maximizes the entropy under the given constraints. The Lagrange multipliers λ_i can be determined by substituting the values for p_i in the constraints (3) and solving the corresponding implicit equations. Practically, however, these equations become too tedious to solve. Therefore, studying the properties of these distributions to obtain some relationships describing the dependence of λ_i on parameters F_k becomes essential. We complete the following analysis.

The expected value of $f_k(x)$ over this probability distribution is thus

$$F_k = \sum_{i=1}^{n} f_k(x_i) p_i = -\frac{\partial \log Z(\lambda_1, \ldots, \lambda_m)}{\partial \lambda_k}. \quad (8)$$

Substituting p_i from (6) into (4), we get the maximum value of Entropy (H_{max}) = $\lambda_o + \sum_{j=1}^{m} \lambda_j F_j$. H represents a measure of the 'amount of uncertainty' in the probability distribution. Once H has been maximized, it becomes a function of the data of the problem $\{F_k\}$. Thus

$$H_{max} = S(F_1, .., F_m) = \log Z(\lambda_1, .., \lambda_m) + \sum_{k=1}^{m} \lambda_k F_k \quad (9)$$

A. Sensitivity Analysis

A small change in one of the F_k values will change the maximum attainable H by the amount $\frac{\partial S(F_1,\ldots,F_m)}{\partial F_k}$ Substituting from (8), we get $\lambda_k = \frac{\partial S(F_1,\ldots,F_m)}{\partial F_k}$. Thus, specifying $S(F_1, \ldots, F_m)$ or $Z(\lambda_1, \ldots, \lambda_m)$ is equivalent in the sense that each gives full information about the probability distribution. Differentiating (8) w.r.t λ_j and λ_k w.r.t F_j, we obtain the reciprocity laws

$$\frac{\partial F_k}{\partial \lambda_j} = \frac{\partial^2 \log Z(\lambda_1, \ldots, \lambda_m)}{\partial \lambda_j \partial \lambda_k} = \frac{\partial F_j}{\partial \lambda_k} \quad (10)$$

$$\frac{\partial \lambda_k}{\partial F_j} = \frac{\partial^2 S}{\partial F_j \partial F_k} = \frac{\partial \lambda_j}{\partial F_k}. \quad (11)$$

These reciprocity laws have highly non-trivial physical interpretations in various applications [3], [4], [5].

If one of the functions $f_k(x)$ contains an extra parameter α, then the change in maximum attainable H can be predicted. First, the best estimate of the derivative is sought by computing its mean value over the probability distribution. which can be simplified to yield $\left\langle \frac{\partial f_k}{\partial \alpha} \right\rangle = -\frac{1}{\lambda_k} \frac{\partial}{\partial \alpha} \log Z$. Additionally, if the parameter α appears in all different f_k, then the above can be generalized to $\sum_{k=1}^{m} \lambda_k \left\langle \frac{\partial f_k}{\partial \alpha} \right\rangle = -\frac{\partial}{\partial \alpha} \log Z$. Now the maximum entropy S is also a function of α. On differentiating w.r.t α, we get

$$-\frac{\partial S}{\partial \alpha} = \sum_{k=1}^{m} \lambda_k \left\langle \frac{\partial f_k}{\partial \alpha} \right\rangle = -\frac{\partial}{\partial \alpha} \log Z. \quad (12)$$

IV. PROPOSED 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 combinatorially with an increase in the amount of data. This property is a deterrent to using this algorithm for large (complex) problems.

To diminish the local influence of domain elements, we replace the hard partitions in the cost function (Equation 2) by 'soft' partitions by introducing weighting functions $p(y_j|x_i)$ that associate each element x_i in the domain Ω to every resource location y_j (see Figure 1). Then we design this distribution using MEP by maximizing its entropy under the constraints that expected values of coverage and computational cost functions satisfy prespecified values. By decreasing a parameter (called *temperature*) that comes naturally in the MEP, we formulate an iterative algorithm where the expected value of the coverage cost function is continually decreased. The scalable resource allocation

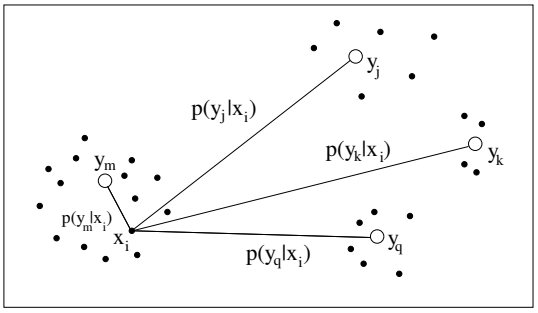

Fig. 1. Soft partitions, Weighting functions $p(y_j|x_i)$ and dimnishing local influence $p(y_m|x_i) >> p(y_j|x_i)$

problem can be stated as a Maximum Entropy problem as follows:

$$\max_{y_j, p(y_j|x)} H = -\sum_j \sum_i p(x_i, y_j) \log p(x_i, y_j) \quad (13)$$

$$\text{such that, } \sum_j \sum_i p(x_i) p(y_j|x_i) d(x_i, y_j) = F_1 \quad (14)$$

$$\sum_j \sum_i p(x_i) p(y_j|x_i) d(x_i, y_j) M_{ij} = F_2 \quad (15)$$

$$\sum_j \sum_i p(x_i) p(y_j|x_i) {N_j}^2 = F_3 \quad (16)$$

$$\text{where, } H = H(X, Y) = H(X) + H(Y|X) \quad (17)$$

$$H(Y|X) = -\sum_j \sum_i p(x_i) p(y_j|x_i) \log p(y_j|x_i), \quad (18)$$

$$M_{ij} = \begin{cases} 0 & \text{if } x_i \in R_j \\ 1 & \text{if } x_i \notin R_j \end{cases}, \quad (19)$$

and N_j is the number of elements in cluster R_j, given by $N_j = \sum_i (1 - M_{ij})$. Here (14) represents the average coverage function where we have replaced the hard partitions by a distribution function. Thus the original decision variables of partitions $\{R_j\}$ are replaced by probability

mass functions $\{p(y_j|x)\}$. By introducing these distribution functions, as in the DA algorithm, each resource location is influenced by every element of Ω and thereby reduces the probability of 'getting stuck to local minima'. The problem setup (13-19) reflects one iteration of the algorithm where β_1 (which can be thought of as reciprocal of the temperature variable mentioned earlier) is fixed. As we increase the β_1 variable (and appropriately change β_2 and β_3 along with it), we obtain the iterations analogous to the DA algorithm. Optimal partitioning requires individual clusters to contain elements which are maximally 'similar' to each other and at the same time are maximally diverse/dissimilar from elements of other clusters. Equation 15 represents the sum of interaction between members of a particular cluster with the outside members. For an optimal partition, we would like to minimize this interaction. This will then enable us to ignore the inter-cluster interactions, thereby resulting in a lower number of computations for determining the coverage F_1 and the entropy H. Equation 16 represents the computational expense associated with a particular partitioning. For each cluster, the number of computations is proportional to the number of elements in that cluster. Thus this term constraints the size of each cluster and penalizes big clusters (which result in higher number of net computations).

The first part of the entropy term $H(X)$ is independent of the resource allocations. The remaining entropy term $H(Y|X)$ measures the randomness of the distribution of associated weights. Entropy is the largest when the distribution of weights over each resource location is identical, when all x have the same influence over every resource location (i.e. $p(y_j|x_i) = 1/M$ for each $x \in \Omega$). We further relax the problem by approximating the indicator variables M_{ij} and N_j by smooth functions. They are redefined as:

$$M_{ij} = 1 - e^{-\frac{d_{ij}}{\sigma_j}}, \quad N_j = \textstyle\sum_i e^{-\frac{d_{ij}}{\sigma_j}} \tag{20}$$

where σ_j is a parameter that reflects the square of the radius of the subset of points around the resource location y_j which are included in computation of y_j.

The multi-objective problem involves maximizing the Entropy term $H(Y|X)$ with respect to $p(y_j|x)$ and y_j while keeping F_1, F_2 and F_3 at pre-specified levels.

Define the Lagrangian L by

$$L(y_j, p(y_j|x), \beta_1, \beta_2, \beta_3) = H - \beta_1 F_1 - \beta_2 F_2 - \beta_3 F_3 \tag{21}$$

where β_1, β_2, and β_3 are the Lagrange multipliers for equality constraints in (14), (15), and (16).

Setting $\frac{\partial L}{\partial p(y_j|x_i)} = 0$, and solving for $p(y_j|x_i)$ yields

$$p(y_j|x_i) = e^{-1-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2} \tag{22}$$

Since the sum of associated weights $\sum_j p(y_j|x_i) = 1$, we get $e^{-1} \sum_j e^{-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2} = 1$, implying

$$p(y_j|x_i) = \frac{e^{-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2}}{Z(\beta_1, \beta_2, \beta_3; \sigma_j)} \tag{23}$$

where the partition function Z is defined by

$$Z_i(\beta_1, \beta_2, \beta_3; \sigma_j) = \sum_j e^{-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2} \tag{24}$$

Substituting $p(y_j|x_i)$ from (23) in (21), gives

$$L^* = \sum_i p(x_i) \log Z_i(\beta_1, \beta_2, \beta_3) \tag{25}$$

To obtain the optimal resource locations, set $\frac{\partial L^*}{\partial y_j} = 0$.

$$y_j = \frac{\sum_i p_{ij} \left\{ W_{ij} x_i - \frac{2\beta_3 N_j}{\sigma_j} \left[\sum_k e^{\frac{-d_{kj}}{\sigma_j}} x_k \right] \right\}}{\sum_i p_{ij} \left\{ W_{ij} - \frac{2\beta_3}{\sigma_j} N_j^2 \right\}} \tag{26}$$

where

$$W_{ij} = \left[\beta_1 + \beta_2 \left(1 + e^{\frac{-d_{ij}}{\sigma_j}} (\frac{d_{ij}}{\sigma_j} - 1) \right) \right]. \tag{27}$$

To obtain the optimal subsets around each resource location y_j, (parameterized by σ_j), set $\frac{\partial L^*}{\partial \sigma_j} = 0$, which gives

$$\sum_i p_{ij} \beta_2 {d_{ij}}^2 e^{-\frac{d_{ij}}{\sigma_j}} = 2\beta_3 N_j \left(\sum_k e^{-\frac{d_{kj}}{\sigma_j}} d_{kj} \right) \sum_i p_{ij} \tag{28}$$

We need to solve (28) to determine the parameter σ_j.

Re-writing (28) in vector form, we obtain

$$\beta_2 \begin{bmatrix} p_{1j} d_{1j}^2 \\ \vdots \\ p_{Nj} d_{Nj}^2 \end{bmatrix}^T \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_N \end{bmatrix} = \beta_3' u^T \rho \begin{bmatrix} d_{1j} \\ \vdots \\ d_{Nj} \end{bmatrix}^T \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_N \end{bmatrix} \tag{29}$$

which can be written as

$$(\boldsymbol{\mu}^T \boldsymbol{\rho}) = (\boldsymbol{u}^T \boldsymbol{\rho})(\boldsymbol{b}^T \boldsymbol{\rho}) \tag{30}$$

where, $\boldsymbol{b} = \beta_3' \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}^T = \beta_3' \boldsymbol{d}$

$$\boldsymbol{\rho} = \begin{bmatrix} \rho_1 & \cdots & \rho_N \end{bmatrix}^T = \begin{bmatrix} e^{-\frac{d_{1j}}{\sigma_j}} & \cdots & e^{-\frac{d_{Nj}}{\sigma_j}} \end{bmatrix}^T \tag{31}$$

$$\boldsymbol{u} = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T ; \beta_3' = 2\beta_3 \sum_i p_{ij} = 2\beta_3 \boldsymbol{u}^T \boldsymbol{p}$$

$$\boldsymbol{\mu} = \beta_2 \begin{bmatrix} p_{1j} d_{1j}^2 & \cdots & p_{Nj} d_{Nj}^2 \end{bmatrix}^T$$

Solving for σ_j (see Appendix VII for details), we obtain

$$\frac{1}{\sigma_j} = -\frac{d^T \nu}{d^T d} = -\frac{d^T u}{d^T d} \log \left[\frac{\boldsymbol{\mu}^T b}{u^T b b^T b} \right] - \frac{d^T \log b}{d^T d} \tag{32}$$

Thus the solution of the Lagrangian maximization problem in (21), and hence that of the scalable resource allocation problem, is given by $p(y_j|x)$, y_j, and σ_j computed in (23), (26), and (32) respectively.

An important ingredient in the algorithm is designing the *cooling law*, which dictates how the variables $\beta = [\beta_1 \ \beta_2 \ \beta_3]^T$ are changed in the annealing process. In comparison to the basic DA algorithm, the design of the cooling law for the multi-constraint optimization problem considered in this paper is more complex. This necessitates

a careful study of the sensitivity of the algorithm to the parameter β. In what follows, we show (1) presence of critical values of β where the annealing process is very sensitive to the changes in β and (2) study of the relative sensitivity of constraints on the changes in the vector β.

A. Critical Temperature Values for the Annealing Vector β.

At the instant when a split happens, the Hessian of L^* loses its positive definite property. To study this, we place coincident resources at the same location y_k and perturb it by an amount $\epsilon\Psi_k$. The necessary condition for optimality is $\frac{\partial}{\partial\epsilon}L^*_\epsilon|_{\epsilon=0} = 0$

$$\Rightarrow \sum_i p(x_i) \sum_k p(y_k + \epsilon\Psi_k|x_i)\beta^T\Delta_{ki\epsilon}\Psi_k = 0, \text{where}$$

$$\Delta_{ji\epsilon} = \frac{\partial d}{\partial y_j}|_{y_j+\epsilon\Psi}\Psi_j, d = \left[\begin{array}{cccc} d_1(x_i,y_j) & .. & d_3(x_i,y_j) \end{array}\right]^T$$

Together with this, we also need to ensure that the second partial derivative is positive ($\frac{\partial^2 L_\epsilon}{\partial\epsilon^2} > 0$). Note that $\frac{\partial^2 L^*_\epsilon}{\partial\epsilon^2}$ is given by

$$\sum_i\sum_k p(x_i,y_k)\Psi_k^T\left[\left(\beta^T\otimes\Delta^2_{ki0}\right)I_L - \Delta^T_{ki0}\beta\beta^T\Delta_{ki0}\right]\Psi_k + \sum_i p(x_i)\left(\sum_k p(y_k|x_i)\beta^T\Delta_{ki0}\Psi_k\right)^2.$$

where $\beta^T\otimes\Delta^2_{ki0} = \sum_l \beta_l\frac{\partial^2 d_l(x_i,y_k)}{\partial y_k^2}$.

It is easy to show that the LHS is positive for all perturbations Ψ, if and only if the first term is positive. Thus $\frac{\partial^2 L_\epsilon}{\partial\epsilon^2} > 0 \Leftrightarrow$ the first term is positive. Thus the condition for the bifurcation occurs when

$$\left(\beta^T\otimes\Delta^2_{ki0}\right)I_L - \Delta^T_{ki0}\beta\beta^T\Delta_{ki0} = 0.$$

Solution of this equation provides us with an expression for the critical temperatures. These critical values play a significant role in the algorithm. As the vector β is changed during the annealing algorithm, the system undergoes a sequence of *phase transitions* at these critical values, which consists of natural cluster splits whereby the number of clusters become larger. This information about the phase transition allows us to accelerate the algorithm in between phase transitions without compromising performance.

B. Sensitivity Analysis

The maximum value of entropy (H_{max}) is

$$H_{max} = \sum_i p(x_i)\log Z_i + \beta_1F_1 + \beta_2F_2 + \beta_3F_3. \quad (33)$$

The maximum attainable H depends on the values F_1, F_2 and F_3. Thus $H_{max} = S(F_1,F_2,F_3)$

$$= \sum_i p(x_i)\log Z_i(\beta_1,\beta_2,\beta_3) + \beta_1F_1 + \beta_2F_2 + \beta_3F_3 \quad (34)$$

A small change in either F_1, F_2 or F_3 will change the maximum attainable H by $\frac{\partial S(F_1,F_2,F_3)}{\partial F_k}$ After simplification, we explicitly get $\beta_k = \frac{\partial S(F_1,F_2,F_3)}{\partial F_k}$. Thus specifying $S(F_1,F_2,F_3)$ or $Z_i(\beta_1,\beta_2,\beta_3)$ are equivalent in the sense that each gives full information about the probability distribution. As before, we can derive the reciprocity law

$$\frac{\partial F_k}{\partial\beta_j} = \frac{\partial^2}{\partial\beta_j\partial\beta_k}\sum_i[p(x_i)\log Z_i(\beta_1,\beta_2,\beta_3)] = \frac{\partial F_j}{\partial\beta_k}$$
$$\frac{\partial\beta_k}{\partial F_j} = \frac{\partial^2 S}{\partial F_j\partial F_k} = \frac{\partial\beta_j}{\partial F_k}. \quad (35)$$

The sensitivity analysis in this section provides us with a framework to study the tradeoff between coverage cost (F_1) and computational expense (F_3). At the same time, it can also be used to obtain the Lagrange multipliers β_k.

V. IMPLEMENTATION AND SIMULATIONS

In the previous section, we prescribed an iterative scheme to determine the optimal resource locations y_j, weighting functions $p(y_j|x)$, and the cluster parameter σ_j. The reciprocal of the Lagrange multiplier β_1 is referred to as the 'temperature' variable because of its parallels to the physical process of annealing. The proposed algorithm is hierarchial in nature as it starts with one resource location at the centroid of the dataset and gradually splits into multiple locations which are optimal in the sense of coverage and computational expense. We start with a low value of β_k and gradually increase it as the algorithm progresses. For low values of β_k, we essentially maximize the entropy without taking into account the coverage cost and computational expense. As β_k increases, we trade the maximization of entropy with the minimization of the coverage cost and computational expense.

- Step 1: Initialize the resource location (y_j) at the centroid of the dataset and choose σ_j such that for every point $x_i \in \Omega, M_{ij} \simeq 0$.
- Step 2: Fix y_j and compute the weighting functions $p(y_j|x)$ according to (23).
- Step 3: Compute the optimal σ_j according to (42)
- Step 4: Compute the new locations y_j, given by (26)
- Step 5: Stop if the stopping criterion (such as number of resource locations, or computation time etc.) is met, otherwise go to Step 2.

To implement the splitting process, we perturb each resource locations y_j by a small amount δ and use the values y_j and $y_j + \delta$ in Step 2 and 3. Next, we analyze the new values of resource locations computed in Step 5 to determine the number of distinct locations. Once it reaches the pre-specified limit, we terminate the algorithm.

It should be noted that the value σ_j prescribes the number of terms that constitute each cluster. As a result, all the computations in Step 2, 3 and 4 are truncated to exclude the points that lie outside a given cluster. This reduces the computational time required for clustering a given dataset. For the case of large datasets, this reduction is quite drastic.

For the purpose of simulation, we first used the basic DA algorithm on a dataset which 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. The proposed recursive 'divide and conquer' algorithm starts with one cluster at the centroid. As the algorithm progresses, the clusters are split and association probabilities are assigned according to Equation 23. Figure 2 shows the dataset and the final resource locations. The resource locations (y_j) are computed according to Equation 26. The parameter σ_j prescribes the terms in each cluster (and hence the region where computations are truncated). The parameters β_2 and β_3 were prescribed while β_1 was varied. The main advantage of the proposed algorithm is in terms of the computation time of the algorithm. The results from the two algorithms have been tabulated below. As can be seen from Table I, the recursive 'divide and conquer' type algorithm uses just one-fifth of the time taken by the DA algorithm and results in only a 6.4% error in terms of coverage cost. Both the algorithms were terminated when the number of resources reached 12.

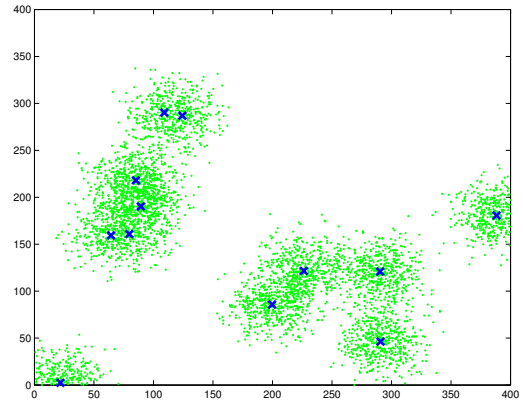

Fig. 2. Simulated dataset, $x_i,\ i = 1, \ldots 5000$ (green dots) and resource locations, $y_j\ j = 1, \ldots, 12$ (blue crosses).

Algorithm	Coverage Cost	Computation Time (sec)
DA	300.80	129.4
Scalable Algorithm	321.05	24.8

TABLE I

COVERAGE COSTS AND COMPUTATION TIMES FOR TWO ALGORITHMS

The critical temperature analysis in Section IV provides us with a framework to automate the process of varying β_k. It also identifies the critical values of β_k at which the resources are split.

VI. CONCLUSIONS

We have developed a framework to simultaneously address the issues of non-convexity and computational complexity for resource allocation problems. The recursive algorithm which we have proposed saves substantial computational expense and thus scales efficiently with an increase in the size of the given dataset. The computational cost has directly been incorporated in the cost function of the optimization problem. Using the MEP framework, we have also characterized the trade-off between the computational expense and the coverage cost.

VII. APPENDIX

Define $\boldsymbol{\nu} = \log \boldsymbol{\rho}$. Thus, $\boldsymbol{\nu} = -\frac{1}{\sigma_j}\left[\begin{array}{ccc} d_1 & \cdots & d_N \end{array}\right]^T$ Define $\gamma_j = \frac{1}{\sigma_j}$. Note that if we knew $\boldsymbol{\nu} = \log \boldsymbol{\rho}$, then $\gamma_j = -\frac{\boldsymbol{d}^T\boldsymbol{\nu}}{\boldsymbol{d}^T\boldsymbol{d}}$. Then (30) holds iff

$$\boldsymbol{\rho}^T(\boldsymbol{u}b^T\boldsymbol{\rho} - \boldsymbol{\mu}) = 0 \Leftrightarrow \boldsymbol{u}b^T\boldsymbol{\rho} - \boldsymbol{\mu} = \left(I - \frac{\boldsymbol{\rho}\boldsymbol{\rho}^T}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right) y \quad (36)$$

Now, $q = \boldsymbol{\mu} + \left(I - \frac{\boldsymbol{\rho}\boldsymbol{\rho}^T}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right) y$ is in the range-space of $S = \boldsymbol{u}b^T$ if $SS^\dagger q = q$, i.e.

$$\left(I - \frac{\boldsymbol{u}\boldsymbol{u}^T}{\|u\|^2}\right)\left[\boldsymbol{\mu} + \left(I - \frac{\boldsymbol{\rho}\boldsymbol{\rho}^T}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right) y\right] = 0 \quad (37)$$

If we assume that the solution to (30) (and (37)) exits, then the solution with least norm is given by $\boldsymbol{\rho} = S^\dagger q$,

$$\boldsymbol{\rho} \quad = \quad (ub^T)^\dagger\left[\boldsymbol{\mu} + \left(I - \frac{\boldsymbol{\rho}\boldsymbol{\rho}^T}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right) y\right] \text{ for some } y \quad (38)$$

where $(ub^T)^\dagger = \frac{bu^T}{\|u\|^2\|b\|^2}$. After substitution, we get

$$\boldsymbol{\rho} \quad = \quad \frac{1}{\|u\|^2\|b\|^2}\left[u^T\boldsymbol{\mu} + u^T y - \frac{u^T\boldsymbol{\rho}\boldsymbol{\rho}^T y}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right] b \quad (39)$$

$$= \quad \Theta b \text{ where } \Theta(\boldsymbol{\rho}) \text{is a scalar} \quad (40)$$

Substituting $\boldsymbol{\rho}$ in (30) gives $\boldsymbol{\mu}^T\Theta b = u^T\Theta b b^T\Theta b$

$$\Rightarrow \Theta = \frac{\boldsymbol{\mu}^T b}{u^T b b^T b} \text{ and } \boldsymbol{\nu} = \log\left[\frac{\boldsymbol{\mu}^T b}{u^T b b^T b}\right] u + \log b \quad (41)$$

We now substitute ν to solve for γ_j

$$\gamma_j = \frac{1}{\sigma_j} = -\frac{d^T\nu}{d^T d} = -\frac{d^T u}{d^T d}\log\left[\frac{\boldsymbol{\mu}^T b}{u^T b b^T b}\right] - \frac{d^T \log b}{d^T d} \quad (42)$$

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] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, (4):620–630, 1957.

[4] E. T. Jaynes. Information theory and statistical mechanics II. *Physical Review*, (2):171–190, 1957.

[5] E. T. Jaynes. *Probability Theory - The Logic of Science*. Cambridge University Press, 2003.

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

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

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

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

[10] C. E. Shannon and W. Weaver. The mathematical theory of communication. *Univ. of Illinois Press, Urbana IL.*

[11] 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.

[12] P. Sharma, S. Salapaka, and C. Beck. A scalable deterministic annealing algorithm for resource allocation problems. *Proc. of American Control Conference*, pages 3092 – 3097, 2006.