ENTROPY BASED FRAMEWORK FOR STATIC AND DYNAMIC COVERAGE AND CLUSTERING PROBLEMS

BY

PUNEET SHARMA

B.Tech., Indian Institute of Technology, Bombay, 2001
M.S., University of Illinois at Urbana-Champaign, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Systems and Entrepreneurial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Doctoral Committee:

Associate Professor Carolyn Beck, Chair
Assistant Professor Srinivasa Salapaka
Assistant Professor Prashant Mehta
Professor Petros Voulgaris

Abstract

In this work, we consider the static and dynamic resource allocation, coverage and clustering problems, and propose a mathematical framework for formulating and solving such problems. The underlying combinatorial optimization problems are solved in a Maximum Entropy Principle (MEP) based framework. The proposed algorithms are computationally efficient and scalable with respect to the size of the underlying data, and are designed to avoid local minima. A characteristic feature of the proposed framework is its ability to detect natural clusters in the underlying data, without the need to initialize it a priori. The notion of coverage is first derived in the static setting and then adapted to the dynamic problem to address the inherent trade-off between the resolution of the clustering solution and the computational complexity. The proposed algorithm for the static problem is successfully implemented to solve the library design problem in combinatorial drug discovery, by addressing the key issues of diversity, representativeness and scalability. In the dynamic problem, the proposed algorithms successfully address the aspects of coverage and tracking. The determination of cluster centers and their associated velocity field is cast as a control design problem to ensure that the algorithm achieves progressively better coverage with time. The proposed algorithm is shown to be five to seven times faster than the frame-by-frame method, with an ability to identify natural clusters in the underlying dataset.

To Mother and Father.

Acknowledgments

This project would not have been possible without the encouragement and support of a lot of people. I would like to thank my advisers, Prof. Srinivasa Salapaka and Prof. Carolyn Beck, for their constant guidance through the course of my doctoral research. They were always available for discussions and provided me fresh ideas and perspectives on a number of problems. Our numerous discussions, be it work related or otherwise, were always interesting and lively. I would also like to thank Prof. Prashant Mehta for his invaluable advice and guidance through the last two years. Together with him, I would like to thank Prof. Petros Voulgaris and Prof. R. Srikant for serving on my exam committees.

I would like to thank my parents and my brothers for their encouragement and guidance through these years. They have been a constant source of motivation for me throughout my life. I would like to take this opportunity to especially thank my wife Monica for always being on my side and encouraging me. It would be an understatement to say that this work would not have been possible without her.

During my stay at Urbana-Champaign, I made a lot of lifelong friends. I would like to thank all of them, especially Nitin, Anurag, Rekha, Sunita, Gaurav, Bhatia, Vinay, Krishna, Jayashree, and Ganguli. Our numerous discussions, dinners, tea sessions and parties will always have a special place in my heart. I would also like to thank all my former labmates, especially Hui-Hung, Praveen, Mike, Kien, LaThu and Tung for our interesting discussions.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, there has been considerable interest in problems that seek selection of a subset from a given population (*combinatorial resource allocation* problems). These resource allocation problems are sometimes referred to as *locational optimization* problems, and address deployment of static or mobile resources to "cover" a set of sites in a given region. Most of these problems are intimately related to a class of combinatorial resource allocation problems that have been studied extensively in various formulations such as the minimum distortion problem in data compression [25], facility location assignments [16], optimal quadrature rules and discretization of partial differential equations [17], pattern recognition [76], drug discovery [64], neural networks [32], and clustering analysis [31]. Recently they have appeared in control literature, especially in coarse quantization [18, 50, 59], coverage control, mobile sensing networks, and motion coordination algorithms [12]. These areas, either directly or indirectly, bring together concepts from control theory and information theory.

The above class of problems can broadly be classified into two main subproblems, namely - the static resource allocation problems and the dynamic resource allocation problems. As the name suggests, the 'site locations' in the static problem are fixed and the task is to determine the locations of the resources as per the coverage cost function and the given constraints. On the other hand, the problems categorized under the dynamic class, have mobile sites which may change their locations and weights at every instant. The task here is to determine the locations of the mobile resources such that they continuously 'cover' the mobile sites (under the cost function and the

given constraints). Depending on the nature of the application and its emphasis on computational complexity, the dynamic problems can also be considered as an ordered set of static problems (indexed by time). But in most applications, such a naive abstraction provides unsatisfactory results and thus calls for an efficient framework to address the dynamic problems in its entirety.

The static problems discussed above focus on different and seemingly unrelated goals, but they have a number of fundamental common attributes. They are typically formulated under a class of combinatorial optimization problems that searches for an optimal partition of the underlying domain (for instance, a library of compounds for drug discovery, an unknown area of interest for coverage control, or a set of data vectors for data compression), as well as an optimal assignment of values, or elements, from a finite *resource* set to each *cell* in the partition (for instance, candidate drug properties [66], vehicle location [67], or data codebook [57]). Computationally, these problems are typically complex and time intensive if not intractable. Since the number of partitions as well as the number of associations of elements to cells in partitions are combinatorial, these problems are computationally complex (NP-hard [28]), which rules out methods that exhaustively search over all partitions and associations. For instance, in the problem of drug discovery, determining 30 representative compounds from an array of 1000 compounds results in approximately 2.43×10^{57} possibilities. The cost functions in these optimization problems are riddled with many local minima; and therefore it becomes all the more important to design algorithms that do not get trapped in local minima [28]. Since the static resource allocation problem is NP-hard, we cannot expect to find optimal solutions. Hence the proposed algorithms are either based on heuristics, or solve a slightly modified version of the original problem.

Even though all the aforementioned static resource allocation 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 arise due to the diverse array of applications under which such problems are usually formulated and implemented. These differences include the different choice of distance metrics in the definition of *coverage*, the number and types of constraints placed on the resources during the problem formulation, the necessity of computing global versus local optima, and the size and scale of the feasible domain. For example, in the combinatorial drug discovery problem [66], 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 for surveillance and reconnaissance applications [60]. Another instance that highlights the differences in the objectives and constraints of the various resource allocation problems occurs in motion control problems that 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, as opposed to the drug discovery problem, where we are primarily interested in global solutions.

The dynamic resource allocation problems are considerably tougher to address than the static problems. The complexity of such resource allocation problems is further compounded by the addition of dynamics to the sites. These problems have numerous applications such as in finding dynamic clusters in groups of animals in studies of their migration patterns [6], in developing automatic deployment and tracking algorithms for surveillance and military applications [12, 23], in weather forecasting by clustering cyclone patterns [13], and in routing traffic and detecting traffic jams by clustering traffic flow [70]. With recent advances in geographic information systems, geopositioning and wireless sensor networks, there have been a growing number of applications which require efficient algorithms for the dynamic resource allocation problems. The task at hand is to design a velocity field for resources such that cov-

erage is maintained over a time horizon. The mobile resources can be alternatively viewed as *cluster centers* that continuously identify and track groups of moving objects. Thus static *locations* of each data-point are replaced by velocity fields, and from a naive computational viewpoint, the dynamic problem can be regarded as a time-indexed set of static problems. Besides, adding dynamics introduces new complexities to the notions of coverage due to the dynamic nature of cluster sizes, number of clusters, and relative distances in between the individual elements.

In spite of its growing interest and wide-ranging applications, the dynamic resource allocation and clustering problems (and even some static resource allocation problems) are still not characterized or defined in a general framework. The terms *cluster* and *coverage* are subjective and usually have different interpretations, depending on the context of the problem and its application. Most of the algorithms are problem dependent and do not exhibit the flexibility required for addressing a wide array of similar problems, by handling multiple constraints and modified cost functions. The computational complexity inherent due to the combinatorial nature of these problems is usually not addressed efficiently by the existing algorithms and it further exacerbates the drawbacks faced by these algorithms.

1.1 Main Contributions

In this work, we have provided a general framework to formulate and solve the static and dynamic resource allocation and clustering problems. The proposed framework is flexible to accommodate multiple constraints and objectives, depending on the context of the problem. The algorithms developed under this framework are computationally efficient and scalable with respect to the size of the underlying domain. As opposed to some of the existing methods, the solutions obtained under the proposed framework are independent of the choice of initialization. A characteristic feature of

the proposed framework is its ability to detect *natural* cluster-centers in the underlying dataset, without the need to initialize or define clusters a priori. Additionally, the algorithms are less likely to be trapped in local minima and are hierarchical in the way that they progressively seek finer subclusters from larger clusters. The framework for formulating the dynamic problems is used to design algorithms that address both the coverage and the tracking aspects important of clustering dynamic data.

1.2 Organization

This thesis has been organized as follows. We introduce the static resource allocation problem in Chapter 2 and formulate it as a combinatorial optimization problem. We identify the key issues that should be addressed in order to solve such problems and reduce the computational complexity of the candidate algorithms. We present some of the existing algorithms that solve slight variants of this problem and discuss the issues and drawbacks faced by them. Next we propose a Maximum Entropy Principle (MEP) based optimization framework to formulate and solve the static problem. Key features of the proposed framework and its advantages over the existing methods are highlighted. In Chapter 3, we address the combinatorial library design problem in drug discovery and propose algorithms that cater to the specific demands and constraints that emanate from combinatorial chemistry. We employ the basic MEP-based framework proposed in Chapter 2 to resolve the underlying constrained optimization problem in drug discovery. We then propose a scalable algorithm to address the quadratic complexity associated with big datasets. In Chapter 4, we formulate a MEP-based framework to resolve the issue of scalability. We derive a tradeoff between the computation time and error and provide simulation results comparing the proposed scalable and the non-scalable algorithms. The dynamic coverage and clustering problem is presented in Chapter 5. The solution for the instantaneous coverage

problem is sought under the MEP framework. We then present a control theoretic based framework for resolving the key issues associated with the dynamic coverage problem and propose an algorithm which addresses both the tracking and the coverage aspects, while achieving progressively better coverage through the time-horizon. Some extensions to the proposed algorithm and future directions are also discussed in detail. Finally we conclude the thesis by revisiting some key results and highlighting the main features of the proposed frameworks and algorithms for solving static and dynamic resource allocations, coverage and clustering problems.

Chapter 2

Static Resource Allocation Problems

2.1 Introduction

Many problems that require the selection of a subset from a given population can be viewed as resource allocation problems, also referred to as locational optimization, coverage, and clustering problems. This class of problems arise in numerous applications, such as the minimum distortion problem in data compression [25], facility location assignments [16], optimal quadrature rules and discretization of partial differential equations [17], pattern recognition [76], drug discovery [66], neural networks [32], and clustering analysis [31]. In recent years, they have also appeared in control literature, especially in coarse quantization for control over networks [18, 50, 59], coverage control for surveillance and reconnaissance problems, mobile sensing networks, and motion coordination algorithms [12].

Despite the numerous differences, all these problems have some fundamental common attributes, and they aim to solve the same underlying optimization problem in order to obtain

1. An optimal partition of the underlying domain.

2. An optimal assignment of values from a finite set to each cell of the partition.

The mathematical formulation of the underlying optimization problem for addressing the static resource allocation problem is presented below.

2.2 Problem Formulation

In its most basic form, the fundamental static resource allocation problem is stated as follows:

Definition 2.1. Static Resource Allocation: *Given a distribution $p(x_i)$ of the elements $x_i,\ 1 \leq i \leq N$ in a space Ω, find the set of M resource locations $y_j,\ 1 \leq j \leq M$ that solves the following minimization problem:*

$$\underset{y_j,\ 1\leq j\leq M}{\arg\min}\ D(x,y) \quad = \quad \underset{y_j,\ 1\leq j\leq M}{\arg\min} \sum_i p(x_i) \left\{ \min_{1\leq j\leq M} d(x_i, y_j) \right\}, \tag{2.1}$$

where $x = [x_1\ x_2\ \ldots\ x_N]^T \in \mathbb{R}^N\ y = [y_1\ y_2\ \ldots\ y_M]^T \in \mathbb{R}^M$.

Here $d(x_i, y_j)$ represents an appropriate *distance* metric between the element x_i and the resource location y_j. The most natural choice of the distance metric is the Euclidean distance $d(x_i, y_j) = \|x_i - y_j\|^2$, but other metrics are also used, depending on the context of the problem. It should be noted that in almost all the application areas where such problems occur, the number of resources M, is much smaller than the number of elements N. The quantities $p(x_i),\ 1 \leq i \leq N$, where $p(x_i) > 0$ and $\sum_{i=1}^{N} p(x_i) = 1$, represent the weights associated with every element $x_i \in \Omega$. For the case where all the elements are identical, we can choose $p(x_i) = \frac{1}{N}, \forall i$. The quantities $p(x_i)$ can also be interpreted as a continuous probability distribution of the elements over the space Ω. For the continuous case, the above cost function is:

$$D(x,y) \quad = \quad \int_{\Omega} p(x) \left\{ \min_{1\leq j\leq M} d(x_i, y_j) \right\} dx. \tag{2.2}$$

The cost function $D(x,y)$ in (2.1) measures the (weighted) average distance of all elements x_i to their nearest resource location. It is often referred to as the *Coverage* cost associated with the placement of the resources for a given dataset with element locations $x_i,\ 1 \leq i \leq N$. For example, in the context of surveillance and reconnais-

sance problems, the quantities $p(x_i)$ represent the distribution of threat activities at the locations x_i in the given area of interest (represented by Ω). The task of the underlying resource allocation algorithm is to identify the locations y_j of the M resources (which in this case are the Unmanned Air Vehicles (UAVs)), such that they effectively *cover* the given threat locations.

The above resource allocation problem can alternatively be interpreted as an *optimal* partitioning problem, which is formulated as follows:

Definition 2.2. Partitioning Problem: *Given a distribution $p(x_i)$ of the elements x_i $1 \leq i \leq N$ in a space Ω, find the optimal partition of the space Ω into M cells R_j, $1 \leq j \leq M$, and assign to each cell R_j a resource location y_j such that it solves the following minimization problem:*

$$\min_{y_j,\ 1 \leq j \leq M} \sum_{j} \sum_{x_i \in R_j} p(x_i) d(x_i, y_j). \tag{2.3}$$

A solution for the above partitioning problem is illustrated in Figure 2.1, with the elements x_i shown by a square, the resource locations y_j shown by a crossed circle, and the partitions shown by the solid lines.

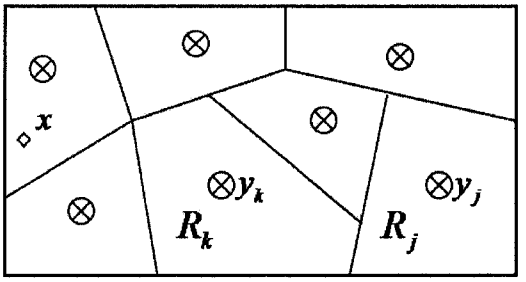


Figure 2.1: An illustrative example depicting the solution of the static resource allocation problem, in which the set of elements x_i $1 \leq i \leq N$ is partitioned into cells R_j, $1 \leq j \leq 7$, and to each R_j, a resource location y_j is ascribed in order to minimize the coverage cost in (2.3).

The minimization problem in (2.1) is non-convex and the cost function surface is typically riddled with numerous local minima. Thus, the algorithms designed

to solve such problems should have the capability to avoid these numerous local minima while searching for the candidate solution. The computational complexity of these algorithms is further exacerbated by the combinatorial nature of the resource allocation problem. For example, in the problem of drug discovery, determining 40 representative compounds (resources) from an array of 5000 compounds (elements) results in approximately 1.21×10^{96} possibilities, thereby ruling out an exhaustive search over all possible partitions. Even in its most basic form, the above resource allocation problem is NP-hard, [28]. Thus, we cannot expect to find the optimal solution in every instance. As a result, the algorithms to solve such problems are either based on heuristics, or solve a slightly modified version of the original problem. In the next two subsections, we briefly discuss two such algorithms, namely the Lloyd's algorithm and the Deterministic Annealing (DA) algorithm. After analyzing the key features of these algorithms and identifying their drawbacks, we then propose a Maximum Entropy Principle based algorithm for formulating and solving the basic static resource allocation and clustering problems.

2.3 Lloyd's Algorithm

Lloyd's algorithm [47] is an iterative method to solve the static resource allocation problem in (2.3). The method identifies two necessary conditions of the optimal solution and then ensures that at each iteration, the partition of domain and the representative elements satisfy these conditions:

1. Nearest Neighbor condition (Voronoi partitions): The partition of the domain is such that each element in the domain is associated to the nearest representative element.

2. Centroid condition: The representative elements are such that the location y_j is in centroid of the jth cell R_j.

In this algorithm, the initial step consists of randomly choosing resource locations and then successively iterating between the steps of: (1) forming Voronoi partitions, and (2) choosing the centroid to be the resource location. It should be noted that the solution depends substantially on the initial allocation of resource locations as in the successive iterations the locations are influenced only by 'near' sites of the domain and are virtually independent of 'far' sites. As a result the solutions from this algorithm 'typically' get stuck to local minima.

2.3.1 Implementation of the Lloyd's Algorithm

The following steps are carried out in order to implement the Lloyd's algorithm:

- Step 1: Initialize the number of clusters M.
- Step 2: Randomly choose the initial locations $y_j, j = 1, 2, \ldots M$ for the resources.
- Step 3: Form Voronoi partitions $\{R_j\}$ by associating each site x_i with the nearest resource location. Ties are broken arbitrarily. The Voronoi partition R_j is defined as follows:

$$R_j = \{x_i | d(x_i, y_j) = \min_{y_k} d(x_i, y_k)\}$$

- Step 4: Move the resource locations to the centroids of their respective partitions, i.e.,

$$y_j = \frac{1}{N} \sum_{x \in R_j} x_i$$

- Step 5: STOP if the resource locations converge, i.e. they do not change their positions in successive iterations or alternatively, the sites do not change their resident clusters in successive iterations.

- Step 6: Go to Step 3.

For practical implementation, the above algorithm is executed multiple times with different initial choice of the resource locations, and the best clustering is chosen as the final solution.

2.3.2 Drawbacks

Since the Lloyd's algorithm is essentially a *local* algorithm, it is unable to steer away from the local minima. Figure 2.2 shows two candidate partitions for the same underlying dataset. Both these partitions, with the choice of resource locations (shown by crossed circles), satisfy the two optimality conditions sought under the Lloyd's algorithm, namely the nearest neighbor condition (Voronoi partitions) and the centroid condition. It is clear from the figures that the resource allocation placement and the

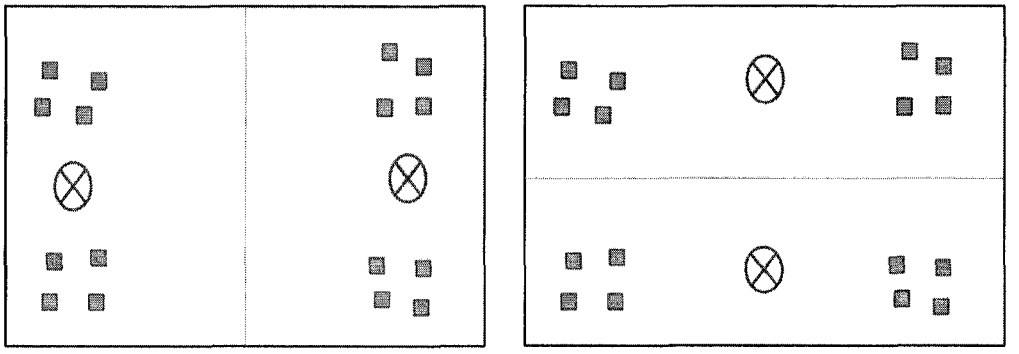


Figure 2.2: Two partitioning solutions obtained under the Lloyd's algorithm. Both the solutions satisfy the optimality steps listed in the Lloyd's algorithm, even though the solution in (b) is just a local minima, while the solution in (a) is the global minima.

partitioning in (a) is considerably better than that in (b) under the cost function defined in (2.3). Still, the Lloyd's algorithm has no way to differentiate between the two scenarios, even though the solution in (b) is just a local minima, as opposed to the global minima in (a). The choice of initial resource allocations considerably alters

the final solution obtained after the iterations in the Lloyd's algorithm.

2.4 Deterministic Annealing Algorithm

In order to address the drawbacks associate with the Lloyd's algorithm, namely it's dependence on the initial choice of resource allocations and its propensity to get stuck in the local minima, several variants were proposed. Deterministic Annealing (DA) algorithm [57] was proposed as a modification of the Lloyd's algorithm and it addressed the main drawbacks often encountered by the latter. The DA algorithm diminishes the local influence of the domain elements by allowing each element $x_i \in \Omega$ to be associated with every resource location y_j through a weighting parameter $p(y_j|x_i)$. This weighting parameter can also be interpreted as an association probability between y_j and x_i, with $\sum_j p(y_j|x_i) = 1 \; \forall \; 1 \leq i \leq N$. Using this formulation, the DA algorithm does away with the hard partitions of Lloyd's Algorithm. The modified cost function obtained under these association weights is as follows:

$$D(x, y) = \sum_{i=1}^{N} p(x_i) \sum_{j=1}^{M} d(x_i, y_j) p(y_j|x_i), \tag{2.4}$$

with the constraint that $\sum_j p(y_j|x_i) = 1$. This cost function is similar to the cost function in (2.1), with soft partitions replacing the previously determined hard partitions. The above term is also referred to as distortion because of its parallels in the Rate-Distortion theory [7].

The choice of the weights $p(y_j|x_i)$ is crucial in determining the extent of trade-off between the diminishing-effect of the local influence and the deviation of the distortion (2.4) from the original cost function (2.1). For instance, a uniform distribution function $p(y_j|x_i) = 1/M$ makes the minimization of (2.4) with respect to y_j independent of initial placement of y_j, however the corresponding distortion function is

considerably different from the cost function in (2.1). On the other extreme, the distribution that assigns $p(y_j|x_i) = 1$ when $d(x_i, y_j) = \min_k d(x_i, y_k)$ else zero, makes the distortion term in (2.4) identical to the cost function (2.1) and also retains the 'local-influence-effect' when minimizing with respect to y_j. Thus the choice of the weighting parameters is a key component of the algorithm, and is determined by optimizing over the space of all possible partitions. In addition to the distortion term,

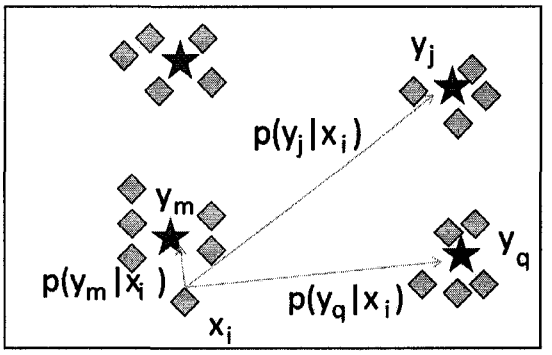


Figure 2.3: Diminishing local influence of association weights. The site x_i is in close vicinity of the resource at location y_m, and considerably farther away from the resources at locations y_j and y_q. As a result, $p(y_m|x_i) \gg p(y_j|x_i)$ and $p(y_m|x_i) \gg p(y_q|x_i)$.

the DA framework also includes an entropy term (H), given by

$$H(y|x) = -\sum_{i=1}^{N} p(x_i) \sum_{j=1}^{M} p(y_j|x_i) \log p(y_j|x_i), \tag{2.5}$$

which measures the randomness of distribution of the associated weights. The entropy is the highest when the distribution of weights over each resource location is the same ($p(y_j|x_i) = 1/M$) for each x_i, i.e., when all x_i have the same influence over every resource location. Maximizing the entropy is the key for selecting a maximally diverse subset of resources. The DA algorithm solves the following optimization problem

$$\min_{y_j} \min_{p(y_j|x_i)} \underbrace{D - T_k H}_{:=F} \tag{2.6}$$

over iterations indexed by k, 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 an analog in statistical physics [56]. 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. T_k can be regarded as the Lagrange parameter in this multi-objective optimization problem.

We use the main features of the DA algorithm and propose a Maximum Entropy Principle (MEP) based framework for solving the static resource allocation problem. Such an approach is flexible enough to accommodate additional constraints and analyze the solution by performing a sensitivity analysis. In Chapter 4, we use the MEP-based framework to resolve the issue of scalability associated with combinatorial optimization problems. We then extend the MEP-based framework to solve the dynamic resource allocation problem in Chapter 5. In the following section, we have presented an overview of the Maximum Entropy Principle and highlighted its main features.

2.5 Maximum Entropy Principle

In this section, we provide a brief overview of the *Maximum Entropy Principle* (MEP)[36]. 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.,

$$\langle f_k(x) \rangle = F_k, \ 1 \leq k \leq m. \tag{2.7}$$

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

$$F_k = \langle f_k(x) \rangle = \sum_{i=1}^{n} p_i f_k(x_i), \quad 1 \leq i \leq n, \tag{2.8}$$

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

MEP postulates that the probabilities should be chosen such that they maximize the Shannon entropy [63] and also satisfy the m expected value constraints.

MEP Statement: The probabilities given by the Gibbs distribution,

$$p_i = \frac{\exp\left\{-\sum_{j=1}^{m} \lambda_j f_j(x_i)\right\}}{Z(\lambda_1, \ldots, \lambda_m)}, \tag{2.9}$$

where Z is called the partition function, defined by

$$Z(\lambda_1, \ldots, \lambda_m) = \sum_{i=1}^{n} \exp\left\{-\sum_{j=1}^{m} \lambda_j f_j(x_i)\right\}, \tag{2.10}$$

maximize the Shannon entropy and satisfy the constraints

$$F_k = \langle f_k(x) \rangle = \sum_{i=1}^{n} p_i f_k(x_i), \quad 1 \leq i \leq n \; 1 \leq k \leq m. \tag{2.11}$$

Proof: We want to determine the probabilities $(p_1, \ldots, p_n)$ such that

$$F_k = \langle f_k(x) \rangle = \sum_{i=1}^{n} p_i f_k(x_i), \quad 1 \leq i \leq n, \tag{2.12}$$

where x_i are the values that the random variable x can take. Since the MEP postulates that the probabilities should be such that they maximize the Shannon entropy [63] and also satisfy the constraints in (2.12), we note the the Shannon entropy defined

by

$$H(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log(p_i), \tag{2.13}$$

is maximized under the constraints in (2.12) by considering the Lagrangian given by

$$L(p_1, \ldots, p_n, \lambda_0, \ldots, \lambda_m) = H - (\lambda_0 - 1)\sum_i p_i - \sum_{j=1}^{m} \lambda_j \sum_i p_i f_j(x_i). \tag{2.14}$$

Setting $\frac{\partial L}{\partial p_i} = 0$, we get

$$p_i = \frac{\exp\left\{-\sum_{j=1}^{m} \lambda_j f_j(x_i)\right\}}{Z(\lambda_1, \ldots, \lambda_m)}, \text{ where,} \tag{2.15}$$

the partition function Z is defined by $Z(\lambda_1, \ldots, \lambda_m) = \sum_{i=1}^{n} \exp\left\{-\sum_{j=1}^{m} \lambda_j f_j(x_i)\right\}$. p_i 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 (2.12) 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

$$F_k = \sum_{i=1}^{n} f_k(x_i) p_i = -\frac{\partial \log Z(\lambda_1, \ldots, \lambda_m)}{\partial \lambda_k}. \tag{2.16}$$

Substituting p_i into (2.13), we get the maximum value of Entropy

$$H_{max} = \lambda_o + \sum_{j=1}^{m} \lambda_j F_j. \tag{2.17}$$

H represents a measure of the 'amount of uncertainty' in the probability distribution. Once H is maximized, it becomes a function of the data of the problem $\{F_k\}$. Thus

$$H_{max} = S(F_1, .., F_m) = \log Z(\lambda_1, .., \lambda_m) + \sum_{k=1}^{m} \lambda_k F_k. \tag{2.18}$$

2.5.1 Sensitivity Analysis

A small change in one of the F_k values will change the maximum attainable H by the amount

$$\frac{\partial S(F_1, ..., F_m)}{\partial F_k} = \sum_{j=1}^{m} \left[\frac{\partial \log Z(\lambda_1, \ldots, \lambda_m)}{\partial \lambda_j}\right] \left[\frac{\partial \lambda_j}{\partial F_k}\right] + \sum_{j=1}^{m} \frac{\partial \lambda_j}{\partial F_k} F_k + \lambda_k. \tag{2.19}$$

On substituting from Equation 2.16, we explicitly get λ_k,

$$\lambda_k = \frac{\partial S(F_1, \ldots, F_m)}{\partial F_k}. \tag{2.20}$$

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 (2.16) w.r.t λ_j and λ_k w.r.t F_j, we obtain the reciprocity laws.

$$\frac{\partial F_k}{\partial \lambda_j} = \frac{\partial^2 \log Z(\lambda_1, \ldots, \lambda_m)}{\partial \lambda_j \partial \lambda_k} = \frac{\partial F_j}{\partial \lambda_k}. \tag{2.21}$$

Similarly, differentiating λ_k w.r.t F_j, we have

$$\frac{\partial \lambda_k}{\partial F_j} = \frac{\partial^2 S}{\partial F_j \partial F_k} = \frac{\partial \lambda_j}{\partial F_k}. \tag{2.22}$$

These reciprocity laws have highly non-trivial physical interpretations in various applications [36, 37].

Sensitivity w.r.t additional parameter

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.

$$\left\langle \frac{\partial f_k}{\partial \alpha} \right\rangle = \frac{1}{Z} \sum_i \exp[-\lambda_1 f_1(x_i) ... - \lambda_k f_k(x_i;\alpha) - \lambda_m f_m(x_i)] \frac{\partial f_k(x_i;\alpha)}{\partial \alpha}, \quad (2.23)$$

which can be simplified to yield

$$\left\langle \frac{\partial f_k}{\partial \alpha} \right\rangle = -\frac{1}{\lambda_k} \frac{\partial}{\partial \alpha} \log Z. \quad (2.24)$$

Additionally, if the parameter α appears in all different f_k, then the above can be generalized to

$$\sum_{k=1}^{m} \lambda_k \left\langle \frac{\partial f_k}{\partial \alpha} \right\rangle = -\frac{\partial}{\partial \alpha} \log Z. \quad (2.25)$$

Now the maximum entropy S is also a function of α. On differentiating w.r.t α, we get

$$-\frac{\partial S}{\partial \alpha} = \sum_{k=1}^{m} \lambda_k \left\langle \frac{\partial f_k}{\partial \alpha} \right\rangle = -\frac{\partial}{\partial \alpha} \log Z. \quad (2.26)$$

It should be noted that S is a function of $(F_1, \ldots, F_m; \alpha)$ while $\log Z$ is a function of $(\lambda_1, \ldots, \lambda_m; \alpha)$. In (2.26), we hold F_k fixed for calculating $\frac{\partial S}{\partial \alpha}$ and λ_k fixed for calculating $\frac{\partial}{\partial \alpha} \log Z$.

2.6 Proposed MEP-based Framework for Static Resource Allocation Problems

The Maximum Entropy Principle provides a convenient way to determine a distribution that achieves a specific value of distortion, and thereby, achieves a pre-specified tradeoff in the context of the static resource allocation problem. More specifically, we seek a probability distribution $p(y|x)$ that maximizes the Shannon Entropy [63],

In the MEP framework, the problem of determining the weighting functions is defined as follows:

Definition 2.3 (Weighting functions in MEP framework:). *Given a distribution $p(x_i)$ of the elements x_i, $1 \leq i \leq N$ in a space Ω and the set of M resource locations y_j, find the probability distribution $p(y_j|x_i)$, $1 \leq i \leq N$, $1 \leq j \leq M$ that solves the following maximization problem:*

$$\max_{p(y_j|x_i)} \underbrace{\left\{ -\sum_{i=1}^{N} p(x_i) \sum_{j=1}^{M} p(y_j|x_i) \log(p(y_j|x_i)) \right\}}_{H(y|x)} \tag{2.27}$$

such that

$$\underbrace{\sum_{i=1}^{N} p(x_i) \sum_{j=1}^{M} d(x_i, y_j) p(y_j|x_i)}_{D(x,y)} = D_0. \tag{2.28}$$

The entropy term quantifies the level of randomness in the distribution of association probabilities, and thus causes the distribution to be maximally non-committal towards any cluster. Therefore in this framework, we first determine the weighting

functions by maximizing the unconstrained Lagrangian,

$$\underset{\{p(y_j|x_i)\}}{\arg\max}\ L = \underset{\{p(y_j|x_i)\}}{\arg\max}\ \left[H(y|x) - \beta(D(x,y) - D_0)\right], \tag{2.29}$$

where $H(y|x)$ and $D(x,y)$ are given by (2.27) and (2.28) respectively, D_0 is the value of distortion the algorithm is aiming for, and β is the Lagrange multiplier. Equivalently, the above problem can also be rewritten as a minimization problem

$$\underset{\{p(y_j|x_i)\}}{\arg\min}\ \underbrace{D(x,y) - TH(y|x)}_{\triangleq F}, \tag{2.30}$$

where the Lagrange multiplier $T = \frac{1}{\beta}$ and the term F are called *temperature* and *Free energy* respectively due to a close analogy to quantities in statistical thermodynamics (where Free energy is the difference between enthalpy (D) and temperature times entropy (TH)) [43].

Using the Lagrangian optimization approach, the solution for the above problem is determined. On differentiating (2.29) with respect to $p(y_j|x_i)$, we obtain the explicit solution for the weights. This is given by the Gibbs distribution,

$$p(y_j|x_i) = \frac{\exp\{-\beta d(x_i, y_j)\}}{\sum_{k=1}^{M} \exp\{-\beta d(x_i, y_k)\}}. \tag{2.31}$$

On substituting this distribution of weights into the Lagrangian (free energy), we obtain the following cost function

$$F(y) = -\frac{1}{\beta}\sum_{i=1}^{N} p(x_i) \log \sum_{k=1}^{M} \exp\{-\beta d(x_i, y_k)\}. \tag{2.32}$$

The resource locations $\{y_j\}$ are specified by minimizing (local) $F(y)$ by setting $\frac{\partial F}{\partial y_j} = 0$

which yields

$$y_j = \sum_{i=1}^{N} p(x_i|y_j)x_i \quad \forall j = 1, 2, \ldots, M, \text{ where,} \tag{2.33}$$

$$p(x_i|y_j) = \frac{p(x_i)p(y_j|x_i)}{\sum_{m=1}^{N} p(x_m)p(y_j|x_m)}. \tag{2.34}$$

Here $p(x_i|y_j)$ denotes the posterior probability calculated using Bayes's rule and the above equations clearly convey the 'centroid' aspect of the solution.

The *temperature* variable $T = \frac{1}{\beta}$ is fixed by the constraint value D_0 of the distortion. Simple sensitivity analysis of optimal solutions to the distortion constraint yields that lower values of D_0 correspond to lower values of the temperature variable [37]. Clearly for small values of β (large values of T) in (2.30), we mainly attempt to maximize the entropy. Thus choice of weights corresponding to high value (near infinity) of T lead to algorithms that are insensitive to initial allocation of resource locations since their subsequent locations are affected almost equally by all the sites. As β is increased (T is lowered), we trade entropy for the reduction in distortion, and as β approaches to infinity (T approaches zero), we minimize D directly to obtain a *hard* (non-random) solution. Accordingly, in this algorithm, an *annealing* process is incorporated where the minimization problem (2.30) is repeatedly solved at different values $\beta = \beta_k$ where β_k is increased with k.

2.6.1 Phase Transition

For the implementation of the algorithm, we start at $\beta = 0$ and increase it as the algorithm proceeds. At $\beta = 0$, the cost function achieves it global optima with $p(y_j|x_i) = \frac{1}{M}$. This yields all the resource locations $y_j, \forall j = 1, \ldots M$ at the centroid of the dataset. As β is increased, the system undergoes a series of phase transitions,

where *phase transition* refers to the splitting of resource locations and an effective increase in the size of the resource location set. Successive phase transitions identify finer and finer clusters in the dataset.

The critical value of β is reached when the solution is no longer the minimizing solution for the free energy in (2.32). This occurs when the Hessian of $\hat{F}$ is no longer positive definite. The critical value can be determined by using a variational approach presented in [57]. Consider a perturbed set of resource locations $Y + \epsilon\Psi = \{y + \epsilon\psi_y\}$, where ψ_y is the perturbation applied to the resource location y. The first-order (necessary) condition for optimality is

$$\frac{d}{d\epsilon}\hat{F}(y + \epsilon\Psi)|_{\epsilon=0} = 0, \tag{2.35}$$

for every finite perturbation Ψ. This leads to the same condition as in (2.33). The second-order condition for minimization is given by

$$H(\psi) = \frac{d^2}{d\epsilon^2}\hat{F}(y + \epsilon\Psi)|_{\epsilon=0} \geq 0. \tag{2.36}$$

Bifurcation of the solution occurs when the above quantity is equal to zero, i.e.

$$\sum_{j=1}^{M} p(y_j)\psi_y^T \left[I - 2\beta C_{x|y_j}\right]\psi_y + \sum_{i=1}^{N} p(x_i)\left[\sum_{j=1}^{M} p(y_j|x_i)(x_i - y_j)^T\psi_y\right]^2 = 0. \tag{2.37}$$

where $C_{x|y_j} = \sum_{i=1}^{N} p(x_i|y_j)(x_i - y_j)(x_i - y_j)^T$. It can be shown that whenever the first term is non-positive, there exists a perturbation ψ such that the second term is zero. Thus (2.36) is strictly positive if and only if the first term of (2.37) is positive for all perturbations. Thus bifurcation is achieved when coincident resource locations at a point y_c satisfy

$$\det\left[I - 2\beta C_{x|y_c}\right] = 0, \tag{2.38}$$

thereby yielding the critical value $\beta_c{}^{-1} = 2\lambda_{max} C_{x|y_c}$.

2.6.2 Iterative Process for Determining the Resource Locations

Solving the implicit equation (2.33) to determine y_j forms the most computationally expensive step at every value of β_k. It is solved by evolving the following dynamical system till it converges

$$y_j^k(n+1) = \sum_{i=1}^{N} p(x_i|y_j^k(n))x_i \quad 1 \leq j \leq M, n \geq 0, \tag{2.39}$$

where $y_j^k(n)$ represents the value of the estimate of y_j (when the temperature value is given by $\beta = \beta_k$) at the nth step of this iterative procedure. The initial value $y_j^k(0)$ is set to the converged value at the previous value of temperature, that is $y_j^k(0) = y_j^{k-1}(\infty)$ (we represent the converged value $y_j^k(\infty)$ by $y_j(\beta_k)$ when the parameter β needs to be emphasized and simply by y_j otherwise). In fact, the iterative process (2.39) is proved to be equivalent to a Newton's descent method and accordingly, this procedure inherits the convergence properties of Newton's descent methods [54].

Proof. At a given value of $\beta = \beta_k$, the resource location y_j is determined by the following centroidal condition

$$y_j = \sum_{i=1}^{N} p(x_i|y_j)x_i \quad 1 \leq j \leq M. \tag{2.40}$$

This is an implicit equation in y_j and is solved by using the iteration scheme

$$y_j^k(n+1) = \sum_{i=1}^{N} p(x_i|y_j^k(n))x_i \quad 1 \leq j \leq M, n \geq 0, \tag{2.41}$$

where $y_j^k(n)$ represents the value of estimate of y_j when the temperature value given

by $\beta = \beta_k$ in at the nth step of this iterative procedure. The above iteration scheme can be interpreted in terms of a descent algorithm [54]. Note that $\frac{\partial \hat{F}}{\partial y_j^k(n)}$

$$= 2\sum_i p(x_i)p(y_j^k(n)|x_i)[y_j^k(n) - x_i] = 2p(y_j^k(n))\left[y_j^k(n) - y_j^k(n+1)\right] \quad (2.42)$$

$$\Rightarrow \nabla\hat{F} = 2\tilde{P}_2\left[y^k(n) - y^k(n+1)\right]. \quad (2.43)$$

where $\tilde{P}_2 = diag\{p(y_j^k(n))\}$ and $\nabla\hat{F} = \left[\frac{\partial \hat{F}}{\partial y_1^k(n)} \cdots \frac{\partial \hat{F}}{\partial y_M^k(n)}\right]^T$. Thus, y^k can be determined by solving the following iterative scheme

$$y^k(n+1) = y^k(n) - \frac{1}{2}\tilde{P}_2^{-1}\nabla\hat{F} \text{ i.e. } y^k(n+1) = y^k(n) + \alpha_k d_k. \quad (2.44)$$

Convergence for this scheme is guaranteed as long as the descent direction d_k is chosen such that $\langle d_k, \nabla\hat{F}\rangle < 0$. For (2.44), we have $\langle d_k, \nabla\hat{F}\rangle = -\nabla\hat{F}^T(\tilde{P}_2^{-1})^T\nabla\hat{F} \leq 0$, with equality only when $\nabla\hat{F} = 0$. Thus solving the implicit equation (2.40) by using an iterative scheme can be equivalently viewed as a descent algorithm. □

2.6.3 Implementation of the MEP-Based Algorithm

- Step 1: Initialize $\beta = 0$ and set limits on the maximum value of β (i.e. β_{max}), and the maximum number of clusters M. Set the initial number of resources to 1, i.e. initialize $K = 1$ and cooling rate parameter γ to be constant greater than 1.

- Step 2: Determine the locations $\mathbf{y} = \{y_j\}$ of the resources by

$$y_j = \sum_{i=1}^{N} p(x_i|y_j)x_i \quad \forall j = 1, 2, \ldots, K, \text{ where } p(x_i|y_j) = \frac{p(x_i)p(y_j|x_i)}{\sum_{m=1}^{N} p(x_m)p(y_j|x_m)},$$

- Step 3: Use the above value of y_j to update the association weights $p(y_j|x_i)$ via

$$p(y_j|x_i) = \frac{\exp\{-\beta d(x_i, y_j)\}}{\sum_{k=1}^{M} \exp\{-\beta d(x_i, y_k)\}} \quad \forall j = 1, 2, \ldots K, \quad i = 1, 2, \ldots N.$$

- Step 4: Iterate between Steps 2 and 3 till the values converge.

- Step 5: If $\beta > \beta_{max}$, perform a last iteration at $\beta = \beta_{max}$ and STOP. If not, then increase the value of β, i.e. $\beta_{k+1} = \gamma\beta_k$.

- Step 6: If the number of resources (K) is less than M, check the phase transition condition for each cluster. If the condition is satisfied for a cluster y_j, split it by adding a resource location and redistributing the weights between them. Set K = K + 1. After implementing this for each cluster, obtain the expanded set of resource locations $y_{\text{new}} = q_{\text{split}}(y)$ as in (2.45) described below. Set $y = y_{\text{new}}$.

- Step 7: Go to Step 2.

Procedure for splitting resource locations: The implementation of the splitting process in step 6 is carried out by replacing each resource location y_j by two new resource locations (say $y_{j,1}$ and $y_{j,2}$) by giving a local perturbation to each y_j, that is $y_{j,i} = y_{j,i} + \epsilon_i$, $\|\epsilon_i\| < \epsilon$, and ϵ is a small bound determined by the numerical precision of the implementation software. Both these resources are assigned a weight $p(y_j)/2$. These perturbed locations remain close to each other as long as the phase transition condition is not reached, since y is a stable minima under these conditions. At the value of parameter β when the phase transition condition is satisfied for cluster j, i.e.

$$\beta^{-1} = 2\lambda_{max}(C_{x|y_j}), \text{ where } C_{x|y_j} = \sum_i p(x_i|y_j)(x_i - y_j)(x_i - y_j)^T,$$

for some j, the new resources $y_{j,i}$ slide to the new minima of F under the iterations given by (2.39). For those locations y_k for which the phase transition condition is not

reached (at the current value of β), the perturbed locations $y_{k,i}$ are collapsed back to y_k. Thus a new set of distinct resource location y_{new} is formed which comprises of split locations $y_{j,i}$ and the original 'unsplit' locations. We represent this map from $y \to y_{\text{new}}$ by q_{split}, that is,

$$\mathbf{y}_{\text{new}} = q_{\text{split}}(\mathbf{y}). \tag{2.45}$$

2.6.4 Features of MEP-Based Algorithm

We summarize the salient features of the MEP-based algorithm for static resource allocation problems in the following theorem and discuss their importance below.

Theorem 2.4. *In the algorithm presented above, the following are true:*

1. Centroid property: $\lim_{\beta\to 0} y_j = \sum_{i=1}^{N} p(x_i)x_i$ *and* $\lim_{\beta\to 0} \frac{dy_j}{d\beta} = 0$ *for* $1 \le j \le M$.

2. Phase transition property: *The resource locations* $y = \{y_j\}$ *given by (2.33) is a local minimum of free energy* F *at every value of* β *except at* critical temperatures *when* $\beta = \beta_c$ *is given by* $\beta_c^{-1} = 2\lambda_{max}(C_{x|y_j})$ *for some* $1 \le j \le M$, *where*

$$C_{x|y_j} = \sum_i p(x_i|y_j)(x_i - y_j)(x_i - y_j)^T, \tag{2.46}$$

   and $\lambda_{max}(\cdot)$ *represents the largest eigenvalue. Moreover, the number of distinct locations in* $\{y_j(\beta)\}$ *for* $\beta > \beta_c$ *is greater than when* $\beta < \beta_c$.

3. Sensitivity-to-temperature property: *If the hessian of* F *is bounded away from zero, that is if* $\|\frac{\partial^2 F}{\partial y_j^2}\| \ge \Delta$ *for* $1 \le j \le M$, *then* $\|\frac{dy_j}{d\beta}\| \le c(\beta)/\Delta$, *where* $c(\beta)$ *monotonically decreases to zero with* β *and is completely determined by* β *and the size of the space* Ω.

Proof. 1. Centroid Property

$$\lim_{\beta \to 0} y_j = \sum_i^N p(x_i) x_i \text{ and } \lim_{\beta \to 0} \frac{dy_j}{d\beta} = 0, \quad 1 \leq j \leq M \tag{2.47}$$

From (2.33), we have

$$y_j = \frac{\sum_{i=1}^N x_i p(y_j|x_i) p(x_i)}{\sum_{m=1}^N p(y_j|x_m) p(x_m)} = \sum_i p(x_i|y_j) x_i, \tag{2.48}$$

The association probabilities are given by the Gibbs distribution

$$p(y_j|x_m) = \frac{e^{-\beta d(x_m, y_j)}}{\sum_{k=1}^M e^{-\beta d(x_m, y_k)}}. \tag{2.49}$$

Taking limits as $\beta \to 0$, we determine that $\lim_{\beta \to 0} p(y_j|x_m) = \frac{1}{M}$. From (2.48), on taking limits $\beta \to 0$, we note that

$$\lim_{\beta \to 0} y_j = \frac{\sum_{i=1}^N x_i p(x_i) \frac{1}{M}}{\sum_{m=1}^N p(x_m) \frac{1}{M}} = \sum_i p(x_i) x_i \triangleq y_c. \tag{2.50}$$

Since this is true for every resource location, we infer that all the resources are coincident at the centroid of the data as $\beta \to 0$.

In order to prove the second part, we differentiate (2.48), w.r.t β

$$\frac{dy_j}{d\beta} = \sum_i \frac{p(x_i) x_i}{p(y_j)} \frac{dp(y_j|x_i)}{d\beta} - \sum_i \frac{x_i p(x_i) p(y_j|x_i)}{p(y_j)^2} \sum_m p(x_m) \frac{d}{d\beta} p(y_j|x_m) \tag{2.51}$$

where,

$$\begin{aligned} \frac{d}{d\beta} p(y_j|x_i) = -p(y_j|x_i) \left[d(x_i, y_j) + 2\beta (y_j - x_i)^T \frac{dy_j}{d\beta} \right] + \\ p(y_j|x_i) \sum_k p(y_k|x_i) \left[d(x_i, y_k) + 2\beta (y_k - x_i)^T \frac{dy_k}{d\beta} \right]. \end{aligned} \tag{2.52}$$

After simplifying the above expression, we get that

$$\left[I - 2\beta \sum_i p(x_i|y_j)(y_j - x_i)(y_j - x_i)^T\right] \frac{dy_j}{d\beta} =$$
$$-\sum_i p(x_i|y_j)(y_j - x_i) \left\{\sum_k p(y_k|x_i) d(x_i, y_k) - d(x_i, y_j)\right\}$$
$$-2\beta \sum_i p(x_i|y_j)(y_j - x_i) \sum_k p(y_k|x_i)(y_k - x_i)^T \frac{dy_k}{d\beta}. \quad (2.53)$$

At $y_j = y_c$ and as $\beta \to 0$,

$$\lim_{\beta \to 0} \left[I - 2\beta C_{x|y_j}\right] \frac{dy_j}{d\beta} = -2 \lim_{\beta \to 0} \beta \sum_i p(x_i)(y_c - x_i)(y_c - x_i)^T \frac{\Theta(\beta)}{M}, \quad (2.54)$$

where $C_{x|y_c} = \sum_i p(x_i|y_c)(y_c - x_i)(y_c - x_i)^T$ is the covariance matrix and $\Theta(\beta) = \sum_k \frac{dy_k}{d\beta}$. Since $\left[I - 2\beta C_{x|y_j}\right]$ is invertible everywhere except at critical values of β (see the discussion in the part 2 of the theorem below), we get

$$\lim_{\beta \to 0} \frac{dy_j}{d\beta} = -2\beta \left[I - 2\beta C_{x|y_j}\right]^{-1} \sum_i p(x_i)(y_c - x_i)(y_c - x_i)^T \frac{\Theta(\beta)}{M}. \quad (2.55)$$

Thus, $\lim_{\beta \to 0} \frac{dy_j}{d\beta} = 0$ if $\lim_{\beta \to 0} \Theta(\beta)$ is bounded. In order to prove this boundedness of $\Theta(\beta)$, we note from (2.54) that

$$\lim_{\beta \to 0} \sum_j \left[I - 2\beta C_{x|y_c}\right] \frac{dy_j}{d\beta} = -2 \lim_{\beta \to 0} \beta C_{x|y_c} \sum_j \frac{\Theta(\beta)}{M} \quad (2.56)$$
$$\Rightarrow \lim_{\beta \to 0} \Theta(\beta) = 0. \quad (2.57)$$

Hence we infer from (2.55) that $\lim_{\beta \to 0} \frac{dy_j}{d\beta} = 0$.

2. Phase transition property:

   See Section 2.6.1.

3. Sensitivity-to-temperature property:

If the hessian of F is bounded away from zero, that is if $\|\frac{\partial^2 F}{\partial y_j^2}\| \geq \Delta$ for $1 \leq j \leq M$, then $\|\frac{dy_j}{d\beta}\| \leq c(\beta)/\Delta$, where $c(\beta)$ monotonically decreases with β and is completely determined by β and the size of the space Ω.

Proof: On multiplying (2.53) by $p(y_j)$ and summing over the index j, we get

$$\underbrace{\sum_j p(y_j) \frac{dy_j}{d\beta}^T \left[I - 2\beta C_{x|y_j}\right] \frac{dy_j}{d\beta}}_{T_1} =$$

$$\underbrace{\sum_i p(x_i) \sum_j p(y_j|x_i)(y_j - x_i)^T \frac{dy_j}{d\beta} \left\{ \sum_k p(y_k|x_i) d(x_i, y_k) - d(x_i, y_j) \right\}}_{T_2}$$

$$\underbrace{- 2\beta \sum_i p(x_i) \sum_j p(y_j|x_i)(y_j - x_i)^T \frac{dy_j}{d\beta} \sum_k p(y_k|x_i)(y_k - x_i)^T \frac{dy_k}{d\beta}}_{T_3}. \quad (2.58)$$

This implies that $T_1 + T_3 = T_2$. Note that the term T_3 can be rewritten as

$$T_3 = 2\beta \sum_i p(x_i) \left[\sum_j p(y_j|x_i)(y_j - x_i)^T \frac{dy_j}{d\beta} \right]^2, \quad (2.59)$$

thereby implying that T_2 is equal to

$$\sum_j p(y_j) \frac{dy_j}{d\beta}^T \left[I - 2\beta C_{x|y_j}\right] \frac{dy_j}{d\beta} + 2\beta \sum_i p(x_i) \left[\sum_j p(y_j|x_i)(y_j - x_i)^T \frac{dy_j}{d\beta} \right]^2.$$

We note that this is same as the Hessian of F in (2.37) under the perturbation $\frac{dy_j}{d\beta}$ instead of ψ_y. Thus,

$$T_1 + T_3 = H\left(\frac{dy_j}{d\beta}\right) = \left(\frac{dy_j}{d\beta}\right)^T \frac{\partial^2 F}{\partial y_j^2} \left(\frac{dy_j}{d\beta}\right). \quad (2.60)$$

If the Hessian of F is bounded away from zero, that is if $\|\frac{\partial^2 F}{\partial y_j^2}\| \geq \Delta$ for $1 \leq$

$j \leq M$, we obtain that

$$\|T_1 + T_3\| = \|T_2\| = \left\| \mathbf{H}\left(\frac{dy_j}{d\beta}\right) \right\| \geq \Delta \left\| \frac{dy_j}{d\beta} \right\|^2. \tag{2.61}$$

Note that since $\sum_k p(y_k|x_i) = 1$, the term T_2 can be rewritten as

$$\begin{aligned} &\sum_i p(x_i) \sum_j p(y_j|x_i)(y_j - x_i)^T \frac{dy_j}{d\beta} \left\{ \sum_k p(y_k|x_i) d(x_i, y_k) - d(x_i, y_j) \right\} \\ &= \sum_i p(x_i) \sum_j \sum_k p(y_j|x_i) p(y_k|x_i) \left[d(x_i, y_k) - d(x_i, y_j) \right] (y_j - x_i)^T \frac{dy_j}{d\beta} \end{aligned} \tag{2.62}$$

Note that $p(y_j|x_i)$ is of the form $\frac{e^{-\beta d(x_i, y_j)}}{\sum_m e^{-\beta d(x_i, y_m)}}$. On multiplying both the numerator and the denominator by $e^{\beta d(x_i, y_k)}$, we get

$$p(y_j|x_i) = \frac{e^{-\beta(d(x_i,y_j) - d(x_i,y_k))}}{1 + \sum_{m \neq k} e^{-\beta(d(x_i,y_m) - d(x_i,y_k))}} \leq e^{-\beta(d(x_i,y_j) - d(x_i,y_k))}. \tag{2.63}$$

Using this bound in (2.62), we infer that the bound on $\|T_2\|$ has terms of the form $e^{-\beta(d(x_i,y_m) - d(x_i,y_k))}(d(x_i, y_m) - d(x_i, y_k))$, i.e. $\alpha e^{-\beta\alpha}$, where $\alpha = (d(x_i, y_j) - d(x_i, y_k))$. The quantity $\alpha e^{-\beta\alpha}$ has a maximum value of $\frac{e^{-1}}{\beta}$. Since the quantity $(x_i - y_j)$ is bounded, we infer that

$$\|T_2\| \leq c(\beta) \left\| \frac{dy_j}{d\beta} \right\|, \tag{2.64}$$

where $c(\beta)$ is a bound on the remaining terms, and it decreases monotonically with β. Note that $c(\beta)$ is completely determined by the value of β and the bound of the size of the space Ω. From (2.61) and (2.64), we

$$\begin{aligned} \Delta \left\| \frac{dy_j}{d\beta} \right\|^2 \leq \|T_2\| \leq c(\beta) \left\| \frac{dy_j}{d\beta} \right\| \\ \Rightarrow \left\| \frac{dy_j}{d\beta} \right\| \leq \frac{c(\beta)}{\Delta}. \end{aligned} \tag{2.65}$$

□

From the centroid property 1 in the above theorem, we conclude that for very small values of β, the algorithm places all the resources at the weighted centroid of the sites. Since the algorithm starts at low values of β, it is virtually independent of the initial placement of resource locations. Furthermore, at $\beta = 0$, the cost function (2.30) achieves its global minimum with $p(y_j|x_i) = \frac{1}{M}$. This yields all the resource locations $y_j, \forall j = 1, \ldots M$ at the centroid of the dataset. The main rationale for the annealing process, which deforms the free energy F, from the entropy function at $\beta = 0$, to the distortion function at $\beta = \infty$, is to obtain the global minimum at low values of β and track its evolution as β is increased. This heuristic is further supported by the free energy principle in statistical thermodynamics [43], which explains the inclusion of the entropy term as an appropriate choice to add to the energy (distortion-term) for this homotopy.

The phase transition property guarantees that the resource locations obtained at non critical values of parameters are local minima of F. In this sense, the performance of this algorithm is at least as good as other algorithms such as Lloyd's algorithm described before. In fact, as $\beta \rightarrow \infty$, this algorithm is equivalent to a Lloyd's algorithm, albeit that the choice of initial placement of locations (through the annealing process) is designed to achieve better (smaller) minima. The phase transition property also explains the hierarchical nature of this algorithm. As we have seen above, at the start of algorithm when β is small, all the resource locations are the same and at the weighted centroid of site-locations. According to this property, when the parameter β crosses a critical value β_c as β is increased, the number of *distinct* resource locations increase from one to more than one. This can also be viewed as a *splitting* process, whereby a single resource location splits into two or more resource locations. As β is further increased, the next critical temperature is reached whereby another resource location (corresponding to value j in $C_{x|y_j}$ in the theorem) splits again. This

splitting process can be interpreted as identifying *natural* clusters and accordingly critical temperatures can be seen as indicators of cluster resolution. At $\beta = 0$, the algorithm identifies one natural cluster of very low resolution; and at the next critical value of $\beta = \beta_c$, it identifies subclusters (of better resolution) and successively after each critical temperature it identifies finer and finer sub-clusters.

From the sensitivity-to-temperature property, we conclude that in this algorithm for the static case, the resource locations do not move afar in between two critical temperatures. The condition on the norm of the hessian $\frac{\partial^2 F}{\partial y_j^2}$ being bounded away from zero corresponds to non-critical temperature values since critical temperatures are characterized by values of β where the hessian loses its rank.

The bounds that we have specified in this theorem are very conservative and much better bounds can be obtained by making some assumptions on the data. For example, for data with the smallest distance between two distinct resource locations being μ and no sites in the rim around each resource described by $\{x|\rho\mu < \|x-y_j\| \leq (1-\rho)\mu\}$, a conservative bound proportional to $e^{-\beta\mu^2|1-2\rho|}$ is obtained (see Appendix B). In simulations, regardless of the site locations, there is hardly any change in resource locations for temperature values in between two successive critical temperatures. This emphasizes the conservativeness of the bounds obtained and avenues for analysis in making these bounds stricter. This property has important consequences - since there is not much effect of temperature change in resource locations, the *cooling law* for the annealing process or the rule by which we change temperature values can have high *rates*. In our simulations we typically change β geometrically, that is

$$\beta_k \;=\; \gamma^k \beta_0 \text{ for some } \gamma > 1. \tag{2.66}$$

Another consequence of this property forms the basis for extending this MEP based framework to dynamic setting, which we discuss in Chapter 5.

We have employed our proposed MEP-based framework to solve the combinatorial library design problem in drug discovery. In the next chapter, we have highlighted the key features of the proposed framework that make it amenable for solving the drug discovery problem.

Chapter 3

Combinatorial Drug Discovery

3.1 Introduction

In recent years, combinatorial chemistry techniques have provided some of the most important tools for assisting chemists in the exploration of huge chemical property spaces in search of new pharmaceutical agents. With the advent of these methods, a large number of compounds are accessed and synthesized using basic building blocks. Recent advances in high throughput screening approaches such as those using micro/nanoarrays [14] have given further impetus to large scale investigation of compounds for drug discovery. However, combinatorial libraries often consist of extremely large collections of chemical compounds, typically several millions. The time and cost associated with these experiments makes it practically impossible to synthesize each and every combination from such a library of compounds. To overcome this problem, chemists often work with *virtual combinatorial libraries*, which are essentially combinatorial databases containing enumeration of all possible structures of a given pharmacophore with all available reactants. They then select a subset of compounds from this virtual library of compounds and use it for physical synthesis and biological target testing. The selection of this subset is based on a complex interplay between various objectives, which is cast as a combinatorial optimization problem. To address this problem, computational optimization tools have been employed to design libraries consisting of subsets of representative compounds which can be synthesized and subsequently tested for relevant properties, such as structural activity,

bioaffinity, and aqueous solubility.

The lead generation design problem is viewed as a combinatorial resource allocation problem since it requires partitioning the underlying chemical property space spanned by compounds in the virtual library and ascribing a representative compound (i.e. a member of the lead generation library) to each cell in the above partition. The main criterion for constructing a lead generation library has traditionally been molecular *diversity* [79, 8]. As has been noted in the literature [55, 45], diversity as a sole criterion for selection can yield lead generation libraries with compounds that have limited *drug-like* properties, as this selection procedure disproportionately favors outliers or atypical compounds. Though such a library contains maximally diverse compounds, it may be of little significance because of its limited pharmaceutical applications. This drawback can be addressed by designing libraries that are *representative* of all potential compounds and their distribution in the property space [34, 72, 51]. Thus a good lead generation library design will have a manageable number of compounds with adequate diversity so that atypical compounds are duly represented to ensure synthesis of almost all potential drugs. At the same time, it will be representative of the distribution of compounds in the virtual library. In addition to diversity and representativeness, other requirements may also be considered that further constrain the library design, for example, it may be required that the library contains compounds that exhibit specific drug-like properties [45, 68, 78].

The specific challenges, in addition to that of combinatorial optimization, that lead generation design poses are:

1. Designing cost functions that reflect the notions of diversity and representation.

2. Designing algorithms that are scalable in order to handle large datasets.

3. Incorporating specific constraints on compounds of lead generation (such as constraints coming from limits on experimental resources).

Different multi-objective optimization methods have been used previously for designing libraries. Stochastic methods such as Simulated Annealing [77, 26, 2, 4] and Genetic Algorithms [69, 9] have been proposed. Most of the stochastic methods attempt to avoid getting stuck in local minima by generating an ensemble of states at every point and assigning a non-zero probability of transition to an 'inferior' solution. Multi-objective approaches to the problem are useful for including several different design criterion in the algorithm. Taking into consideration the size of these combinatorial libraries, it becomes essential to consider the scaling issues involved with any algorithm that solves the multi-objective optimization problems. Unfortunately, most of the methods mentioned above do not address the key issues of diversity and representativeness simultaneously, and these algorithms are not scalable and often underperform when datasets are large.

In this chapter, an algorithm for lead generation library design (i.e. selecting a subset of compounds from a larger set of virtual compounds) has been presented. The proposed framework accounts simultaneously for diversity as well as representativeness by formulating a combinatorial resource allocation problem with multiple constraints. New procedures are presented that address the issues of scalability and various constraints on the compounds in the lead generation library. We present an algorithm in a MEP-based framework developed in Chapter 2 and modify it to cater to the specific constraints and demands of combinatorial chemistry. The distinctive feature of the MEP-based algorithm and its advantages over the traditional methods have already been discussed in Chapter 2. We propose a scalable algorithm, which preserves the desirable features of the original algorithm, and at the same time addresses the issue of quadratic complexity [3].

3.2 Combinatorial Library Design

Library design refers to the process of screening and then selecting a subset of compounds from a given set of similar or distinct compounds (a virtual combinatorial library) for drug discovery [27]. The combinatorial nature of the selection problem makes it impractical to exhaustively enumerate each and every possible subset to obtain the optimal solution. For example, in order to select 30 representative compounds from a set of 1000, we have ${}^{1000}C_{30}$ possibilities, i.e. approximately 2.42×10^{57} different possible combinations from which to choose. This makes the selection (based on enumeration) impractical and calls for efficient algorithms that make the optimal selection under given constraints. The main aim of library design is to reduce the number of compounds for testing without compromising on the diversity, representativeness and properties such as drug-like behavior in the subset chosen. It facilitates the process of drug discovery by replacing the synthesizing and subsequent testing of all compounds by a much smaller set of representative compounds. Based on the current state of development in the drug discovery process, library design is broadly classified into two main categories:

1. Lead generation library design (broad screening).

2. Lead optimization library design (targeted screening).

The main purpose of these libraries and their design criteria are discussed below.

3.2.1 Lead Generation Library Design

The development of a lead generation library usually involves the design and synthesis of a large number of chemical compounds. It is required that these compounds are diverse from each other. The library containing these diverse compounds is then tested against a host of different biological agents. The main objective in designing

such libraries is to obtain structurally diverse compounds so as to be representative of the entire chemical space. Keeping these requirements in mind, 'diversity' is generally used as the principal screening criterion for designing lead generation libraries[8, 79]. This criterion does not necessarily yield intended results, and may encourage library design that contains compounds so diverse, i.e. singletons or outliers, that they are not 'representative' of any group other than themselves [55, 45]. Such a maximally diverse subset is of little practical significance because of its limited pharmaceutical applications. Hence the criterion of 'representativeness' should also be considered along with 'diversity' [34, 72]. This issue is effectively addressed in this chapter.

3.2.2 Lead Optimization Library Design

Lead optimization libraries are usually designed at a later stage of the drug discovery process, when it is required to select a subset of compounds that are *similar* to a given lead compound(s). This results in an array of compounds which are structurally and chemically similar to the lead. The criterion of similarity is generally used for designing targeted or focused libraries, with a single therapeutic target in mind. Thus the design of lead generation libraries precede that of lead optimization libraries. All the algorithms presented in this chapter are for the computationally intensive lead generation library design, and the computationally less expensive lead optimization problem has not been discussed.

3.3 Lead Generation Library Design Problem

This chapter deals with the problem of designing a library of compounds for the purpose of lead generation. The most common method used to obtain such a library is to maximize the diversity of the overall selection [8, 79]. It is based on the strategy that the more diverse the set of compounds, the better the chance to obtain a lead

compound with desired characteristics. As was noted earlier [55, 45], such a design strategy suffers from an inherent problem which occurs due to the fact that using diversity as the only criterion may result in a set of compounds which are exceptions or singletons. Figure 3.1 shows such a scenario. Here, the circles represent the distribution of compounds in a descriptor space, while the crosses denote compounds chosen according to the maximum diversity principle. As can be seen from the figure, the cluster in the middle is not adequately represented in the final selection. This selection focuses disproportionately on distant compounds, which may be viewed as exceptions. From a drug discovery point of view, it is desirable for the lead genera-

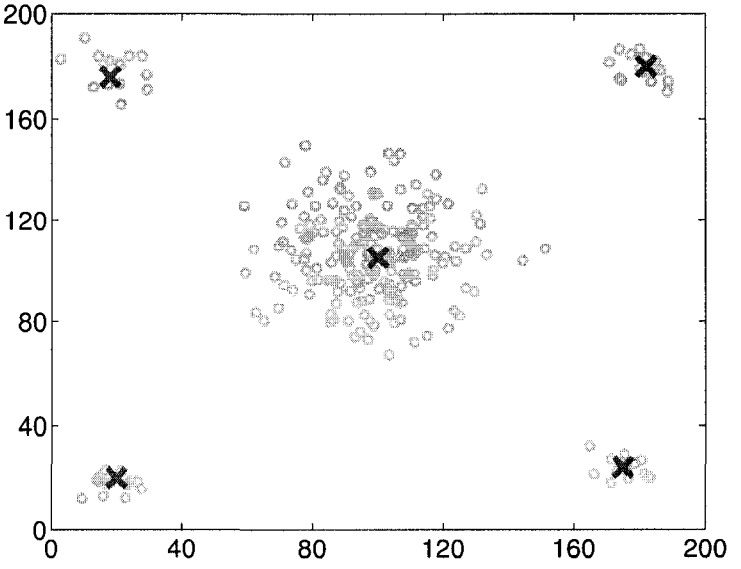


Figure 3.1: A scenario depiciting the inherent problems with the 'diversity' only criterion for lead generation library design.

tion library to contain more compounds from the middle cluster (so as to adequately represent all the compounds) or to at least determine how proportionately representative they are in order to make decisions on how much experimental resources to devote to these lead compounds. A maximally diverse subset is of little practical significance because of its limited pharmaceutical applications. Hence the criterion of representativeness should be considered along with diversity [34, 72, 11]. To ad-

dress this problem, we present an algorithm which selects a lead generation library that achieves high diversity and at the same time specifies how representative (what percent of all the compounds) each member in the library is. The process involves identifying different representative compound locations in an iterative fashion, and is discussed in following sections.

The large size of combinatorial libraries calls for a scalable selection algorithm. We propose a *divide-and-conquer* scheme for the implementation of the algorithm in which the underlying property space of the compounds is recursively and hierarchically partitioned into sub-domains that are approximately isolated from each other. The scheme quantifies the degree of isolation or equivalently the extent of interaction between the sub-domains after each partitioning step. This interaction term is used in specifying bounds on the deviation from the results had the original MEP-based algorithm been used on the entire property space. The implementation of the selection algorithm on each of these subsets that are smaller in size than their parents in the hierarchy substantially reduces the computational time. The proposed algorithm is easily adaptable to include additional criteria needed in the design of lead generation libraries. In addition to similarity and diversity, other criteria include *confinement* which quantifies the degree to which the properties of a set of compounds lie between prescribed upper and lower ranges [5], and maximizing the *activity* of the set of compounds against some predefined targets. Activity is usually measured in terms of the quantitative structure of the given set. The presence of these multiple (and often conflicting) design objectives make the library design a multi-objective optimization problem with constraints.

3.3.1 Molecular Descriptors

Molecular descriptors are essential for quantifying the properties of different compounds in a chemical space. One, two and three dimensional descriptors are used to

encode the chemical composition, chemical topology and three-dimensional shape [46]. A large number of descriptors can be used to effectively characterize a chemical space. In practice, many of these descriptors are correlated and tools from dimensionality reduction (non-linear mapping and principal component analysis methods [62, 39]) are used to extract lower dimensional representations that preserve the *neighborhood behavior*. It has been previously shown that a 2-dimensional molecular descriptor space exhibits a proper neighborhood behavior which is essential for characterizing similarity and diversity properties between two compounds [4]. In all our simulations, we consider a 2-d or 3-d molecular descriptor space. The Euclidean distance between two points in the space provides a good measure of the degree of similarity between these two compounds. Thus in this scenario of a 2-d descriptor space, the close neighbors of an active compound will also be active. On the other hand, two compounds which are far apart (in terms of the Euclidean distance) can be labeled as diverse. The 2-d descriptor space provides a means of quantifying the properties of different compounds. More details on molecular descriptors and the neighborhood behavior are discussed in [4].

3.3.2 Library Design as a Static Resource Allocation Problems

Due to the underlying combinatorial nature of the compound selection problem, the lead generation library design problem is initially formulated as a class of static resource allocation problems. In order to quantify the different criteria used for designing libraries (namely diversity and similarity), it is required to define appropriate *molecular descriptors* which numerically characterize the various compounds in a chemical space.

In its prototypical form, the problem of selecting representative compounds for

the purpose of library design can be stated as a static resource allocation problem as presented in (2.1).

Definition 3.1. Library Design as a Static Resource Allocation Problem: *Given a distribution $p(x_i)$ of the compounds $x_i,\ 1 \leq i \leq N$ in a descriptor space Ω, find the set of M representative compounds $y_j,\ 1 \leq j \leq M$ that solves the following minimization problem:*

$$\underset{y_j,\ 1 \leq j \leq M}{\arg\min}\ D(\mathbf{x}, \mathbf{y}) \quad = \quad \underset{y_j,\ 1 \leq j \leq M}{\arg\min} \sum_i p(x_i) \left\{ \min_{1 \leq j \leq M} d(x_i, y_j) \right\}, \tag{3.1}$$

where $\mathbf{x} = [x_1\ x_2\ \ldots\ x_N]^T \in \mathbb{R}^N\ y = [y_1\ y_2\ \ldots\ y_M]^T \in \mathbb{R}^M$.

Here $d(x_i, y_j)$ represents an appropriate *distance* metric between the compound x_i and the representative compound y_j in the descriptor space Ω. The quantities $p(x_i),\ 1 \leq i \leq N$, where $p(x_i) > 0$ and $\sum_{i=1}^{N} p(x_i) = 1$, represent the weights associated with every compound $x_i \in \Omega$.

In order to solve the above combinatorial optimization problem, one can employ the MEP-based approach that we proposed in Chapter 2. The solution yields the representative compound locations y_j together with the weighting parameters $p(y_j|x_i)$. But the formulation in (3.1) does not fully address all the issues and incorporates the multiple constraints that are associated with the lead generation library design problem. Additionally, the MEP-based algorithm is computationally costly due to the combinatorial complexity involved in working with large sized libraries. As a result, such large scale problems call for a framework that has the flexibility to accommodate multiple constraints, scales up efficiently with the size of the underlying domain, and at the same time mitigates the computational complexity.

3.4 Constraints on Representative Compounds and Experimental Resources

In the MEP-based formulation that we presented in Chapter 2 (to solve the static resource allocation problem (3.1)), the individual representative compounds are indistinguishable since each of them carry equal weight. However, capacity constraints often distinguish one representative compound from another in practical situations. In order to account for such capacity constraints, it is necessary to modify the MEP-based algorithm. For the specific problem of library design, one such scenario occurs when we want to address the issue of representativeness of individual clusters in the final library design. In order to constrain the size of each cluster, it becomes necessary to distinguish between the various locations of representative compounds. Moreover, quantity constraints on the experimental resources also call for a modification of the MEP-based algorithm. In this section, we present two modifications of the original MEP-based algorithm for addressing these issues.

3.4.1 Incorporating Representativeness

The necessity for quantifying and incorporating representativeness is evident, as discussed in Section 3.3. We incorporate representativeness while identifying diverse compounds by specifying an additional parameter λ_j, $1 \leq j \leq M$ for each compound in the lead generation library. This parameter λ_j quantifies the extent of representativeness of the compound location y_j, i.e., it gives a measure of the number of compounds in the underlying dataset that are represented by the compound location y_j. Thus the resulting library design will have compounds that will cover the entire dataset, where the compounds in the library that cover outliers will have low representative weights and the compounds that cover high volume regular members in the dataset will have high representative weights. In this way, the algorithm can be used

to identify diverse compounds through locations y_j and at the same time determine how representative each compound in the library is.

This representativeness criterion is incorporated into the algorithm by reinterpreting the MEP-based algorithm. The MEP-based algorithm as presented in Chapter 2 assumes that all the representative compounds are identical. However in the new interpretation, each location y_j can be weighted by λ_j which translates to the modified partition function in the algorithm

$$Z_i = \sum_{j=1}^{M} \lambda_j e^{-d(x_i,y_j)/T_k}. \tag{3.2}$$

The parameters λ_j, $1 \leq j \leq M$ give relative weights (and therefore relative representativeness) of locations y_j, $1 \leq j \leq M$, and without loss of generality can be assumed to sum to one. In this denotation, $\lambda_j = 0.2$, for example, would mean that the representative compound y_j in the library represents 20% of the compounds in the dataset. The modified algorithm solves for locations y_j, which maximize the diversity as before, and in addition finds the weights λ_j that specify representativeness by solving the following constrained minimization problem

Definition 3.2. *Capacity-constrained Problem:*

$$\underset{y_j,\ 1\leq j\leq M}{\arg\min} \sum_{i=1}^{N} p(x_i) \left\{ \min_{1\leq j\leq M} d(x_i, y_j) \right\}, \tag{3.3}$$

such that

$$\sum_{j=1}^{M} \lambda_j = 1.$$

Note that although the constraints do not seem to occur in the cost function explicitly, they can be interpreted in terms of the MEP-based algorithm where the partition function is modified as in (3.2). These weighting constraints lead to modified Gibbs distribution and Free Energy terms. Following a procedure similar to that used

in the original algorithm, we get

$$p(y_j|x_i) = \frac{\lambda_j e^{-\beta_k d(x_i,y_j)}}{\sum_j \lambda_j e^{-\beta_k d(x_i,y_j)}}. \tag{3.4}$$

At each value of β, the representative compound location y_j and the weight λ_j are given by

$$y_j = \sum_i p(x_i|y_j)x_i \ , \ \ \lambda_j = \sum_i p(y_j|x_i)p(x_i), \ \ 1 \leq j \leq M. \tag{3.5}$$

Thus we obtain locations for representative compounds in the designed library y_j and their respective representative weights λ_j. Experimental results in Section 3.6 demonstrate the efficacy of this algorithm, where the selection is depicted by location y_j in the descriptor space and the corresponding weights λ_j are shown in pie-charts.

3.4.2 Incorporating Constraints on Experimental Resources

The cost of chemical compounds and experimental resources is significant and presents one of the main impediments in combinatorial diagnostics and drug synthesis. In fact, recent research in nano-instrumentation has led to nanoarrays which bring together the many capabilities crucial for rapid and high-volume research and production - including design, prototyping, assembly, testing and reliable replication at the specificity of biomolecules [48, 14]. One of the main advantages of this approach is its economy, since very small amounts of biological and chemical agents are needed, thereby reducing the cost considerably. Still, different compounds require different experimental supplies which are typically available in limited quantities. In this section, we include these constraints into our algorithm for lead generation, as described below.

The library is classified into q types corresponding to experimental supplies re-

quired by the compounds for testing. We incorporate the supply constraints into the algorithm by translating them into direct constraints on each of the representative compounds. For example, the jth representative compound can avail only W_{jn} amount of the nth resource. This type of a constraint is generally referred to as a multi-capacity constraint. [60]

The modified optimization problem then is given by

Definition 3.3. *Multi-capacity Constrained Problem:*

$$\underset{y_j,\ 1\le j\le M}{\arg\min}\ D(x,y) = \underset{y_j,\ 1\le j\le M}{\arg\min} \sum_n \sum_i p_n(x_i^n) \sum_{j=1}^{M} d(x_i^n, y_j) p(y_j|x_i^n), \tag{3.6}$$

such that

$$\lambda_{jn} = W_{jn} \quad 1 \le j \le M,\ \ 1 \le n \le q, \tag{3.7}$$

where $p_n(x_i^n)$ is the weight of the compound location x_i^n, which requires the nth type of experimental supply, and W_{jn} is the amount of the nth supply that the jth representative compound can avail.

We proceed along the same lines as the MEP-based algorithm by defining the entropy term by

$$H(x,y) = -\sum_n \sum_i p_n(x_i^n) \sum_{j=1}^{M} p(y_j|x_i^n) \log p(y_j|x_i^n), \tag{3.8}$$

and minimizing the corresponding Free Energy, given by $F = D - \frac{1}{\beta_k} H$. This procedure yields the Gibbs distribution of the form

$$p(y_j|x^n) = \frac{\lambda_{jn} e^{-\beta_k d(x^n, y_j)}}{\sum_k \lambda_{kn} e^{-\beta_k d(x^n, y_k)}}. \tag{3.9}$$

Adding the constraints to this equation, we derive the new Lagrangian given by

$$\hat{F} = -\beta \sum_n \sum_i \log(\sum_j \lambda_{jn} e^{-\beta d(x_i^n, y_j)}) p_n(x_i^n) + \sum_j \sum_n q_{jn}(\lambda_{jn} - W_{jn}), \quad (3.10)$$

where q_{jn} $1 \leq n \leq q, 1 \leq j \leq M$ are Lagrange multipliers. Finally, the optimal locations y_j of representative compounds for the lead generation library under experimental constraints are obtained by setting $\frac{\partial \hat{F}}{\partial y_j} = 0$. This gives the following set of equations

$$y_j = \frac{\sum_n p_n(x^n) p(y_j|x^n) x^n dx^n}{\sum_n p_n(x^n) p(y_j|x^n) dx^n}, \quad (3.11)$$

which has the centroidal aspect in which averages are also taken about experimental supplies.

3.5 Scalable Algorithm For Library Design

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. In the original MEP-based algorithm, the computational complexity can be addressed in two steps - first by reducing the number of iterations and second by reducing the number of computations at every iteration. The MEP-based algorithm, as described earlier, exploits the phase transition feature in its process to decrease the number of iterations (in fact in the MEP-based algorithm, typically the temperature variable is decreased exponentially which results in few iterations). The number of computations per iteration in the MEP-based algorithm is $O(M^2N)$ where M is the number of representative compounds and N is the number of compounds in the underlying dataset. In this section, we present an algorithm that requires fewer computations per iteration. This amendment becomes

necessary in the context of the selection problem in combinatorial chemistry as the sizes of the initial dataset are so large that the MEP-based is typically too slow to be practical and often fails to handle the computational complexity.

We exploit the features inherent in the MEP-based algorithm to reduce the number of computations in each iteration. We use the fact that, for a given temperature, the farther an individual data point is from a cluster, the lower is its influence on the cluster. This is evident from the form of the association probabilities $p(y_j|x_i)$ in (2.31). That is, if two clusters are far apart, then they 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 note that ignoring the effects of separated regions (i.e. groups of clusters) on one another will result in a considerable reduction in the number of computations since the points that constitute a separated region will not contribute to the distortion and entropy computations for the remaining points. Thus identification of separated regions in a data-set enables us to process the algorithm much more quickly for large datasets. This saving on the number of computations increases as the temperature decreases since the number of separated regions, which are now smaller, increases as the temperature decreases.

The first step required to identify separated regions is to define and quantify 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. Once groups of *separate* clusters are identified, the final step is to modify the MEP-based algorithm such that it ignores the effects of separate groups on one another. This modification significantly reduces the number of computations to be performed by the algorithm.

3.5.1 Cluster Interaction and Separation

In order to characterize the interaction between different clusters, it is necessary to consider the mechanism of cluster identification during the process of the MEP-based algorithm. As the temperature ($T_k = \frac{1}{\beta}$) is reduced after every iteration, the system undergoes a series of phase transitions. We partition the underlying dataset into clusters by associating each distinct location to all the points in the dataset that are closest, that is, we form the *Voronoi* partition using the nearest neighbor condition. In this way, we achieve finer partitions (smaller clusters), as the temperature decreases. More precisely, at high temperatures that are above a pre-computable critical value, all the representative compounds are located at the centroid of the entire underlying dataset, thereby there is only one distinct location for the representative compounds. As the temperature is decreased, successive phase transitions occur, which lead to a greater number of distinct locations for representative compounds and consequently finer clusters are formed. This provides us with a tool to control the number of clusters we want in our final selection. It was shown in Chapter 2 that a cluster R_i splits at a critical temperature $T_c = \frac{1}{\beta_c}$ when twice the maximum eigenvalue of the posterior covariance matrix, defined by $C_{x|y_j} = \sum_i p(x_i)p(x_i|y_j)(x_i - y_j)(x_i - y_j)^T$ becomes greater than the temperature value, i.e. when $T_c \leq 2\lambda_{max}[C_{x|y_i}]$. This is exploited in the MEP-based algorithm to reduce the number of iterations by jumping from one critical temperature to the next without significant loss in performance.

In the MEP-based algorithm, the location of a representative compound is primarily determined by the data points (compound locations) near to it since far-away points exert small influence, especially at low temperatures. The association probabilities $p(y_j|x_i)$ 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 y_j and x_i. The total interaction exerted by all the data-points in a given

space determines the relative weight of each cluster,

$$p(y_j) := \sum_{i=1}^{N} p(x_i, y_j) = \sum_{i=1}^{N} p(y_j|x_i)p(x_i), \tag{3.12}$$

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

We define the level of interaction that data-points in cluster R_i exert on cluster R_j by

$$\epsilon_{ji} = \sum_{x \in R_i} p(y_j|x)p(x). \tag{3.13}$$

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 value can also be interpreted as the probability of transition from R_i to R_j.

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

$$A = \begin{pmatrix} \sum_{x \in R_1} p(y_1|x)p(x) & .. & \sum_{x \in R_m} p(y_1|x)p(x) \\ \sum_{x \in R_1} p(y_2|x)p(x) & .. & \sum_{x \in R_m} p(y_2|x)p(x) \\ \vdots & \ddots & \vdots \\ \sum_{x \in R_1} p(y_m|x)p(x) & .. & \sum_{x \in R_m} p(y_m|x)p(x) \end{pmatrix}. \tag{3.14}$$

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 3.2 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 identify regions, that is, groups of clusters, which are separate from the rest of the data. The separation is characterized by a quantity which we denote by ϵ. We say a

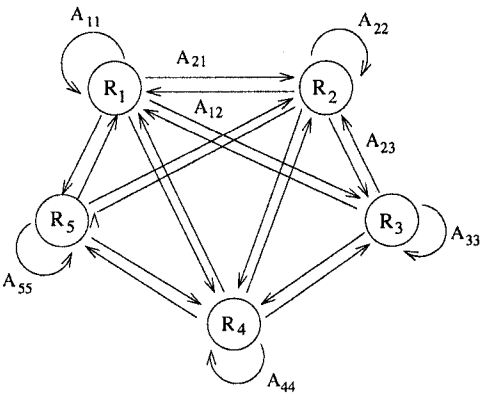


Figure 3.2: Markov chain.

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 the value ϵ determines the number of separate regions formed, if any, which in turn decides the error in distortion due to the proposed algorithm with respect to that of the original MEP-based algorithm.

Alternative ways to identify invariant regions in the underlying space can be motivated by concepts from graph theory. One can consider the Markov matrix as a undirected weighted graph, with each entry A_{ij} representing the weight between node i and node j. Tools from spectral partitioning of graphs [19, 21, 33] are then used to bi-partition (or multi-partition) this resulting graph into invariant sub-graphs. In order to get balanced sub-graphs, partitioning techniques such as Normalized cuts [71] can be used on this weighted graph. However, algorithms for recursive bi-partitioning of graphs result in successive relaxations of the problem, thereby resulting in cumulative errors. Efficient algorithms for multi-way partitions have not been exhaustively studied in the literature. These spectral graph partitioning based algorithms are discussed in more detail later in the chapter.

3.5.2 Trade-off Between Error in Representative Compound Location and Computation Time

As was discussed in Section 3.5, the greater the number of separate regions we use, the smaller the computation time for the modified algorithm. At the same time, a greater number of separate regions results in a higher deviation in the distortion term of the proposed algorithm from the original MEP-based algorithm. This trade-off between reduction in computation time and increase in distortion error is systematically addressed below. We introduce the following notation on a subset V of the domain Ω for a representative compound y_j:

$$G_j(V) := \sum_{x_i \in V} x_i p(x_i) p(y_j|x_i), \tag{3.15}$$

$$H_j(V) := \sum_{x_i \in V} p(x_i) p(y_j|x_i). \tag{3.16}$$

Then from the MEP-based algorithm, the location of the representative compound (y_j) is determined by

$$y_j = \frac{G_j(\Omega)}{H_j(\Omega)}. \tag{3.17}$$

Since the cluster Ω_j is separated from all the other clusters, the representative compound location $y_j^{'}$ will be determined in the modified algorithm by

$$y_j^{'} = \frac{\sum_{x_i \in \Omega_j} x_i p(x_i) p(y_j|x_i)}{\sum_{x_i \in \Omega_j} p(x_i) p(y_j|x_i)} = \frac{G_j(\Omega_j)}{H_j(\Omega_j)}. \tag{3.18}$$

We obtain the component-wise difference between y_j and $y_j^{'}$ by subtracting terms. Note that we use the symbols $\preceq$ and $\succeq$ for component-wise operations. On simplifying

the component-wise terms, we have

$$|y_j - y_j^{'}| \preceq \frac{\max(G_j(\Omega_j^c)H_j(\Omega_j), G_j(\Omega_j)H_j(\Omega_j^c))}{H_j(\Omega)H_j(\Omega_j)}, \tag{3.19}$$

where $\Omega_j^c = \Omega \setminus \Omega_j$. Now note that

$$G_j(\Omega_j^c) \;\leq\; \left(\sum_{x_i \in \Omega_j^c} x_i\right) H_j(\Omega_j^c) = N M_j^c H_j(\Omega_j^c), \tag{3.20}$$

where N is the cardinality of Ω and $M_j^c = \frac{1}{N}\sum_{x_i \in \Omega_j^c} x_i$. We have assumed that $x \succeq 0$ without any loss of generality since the resource allocation problem definition is independent of translation or scaling factors. Thus,

$$\begin{aligned} |y_j - y_j^{'}| &\preceq \frac{\max[N M_j^c H_j(\Omega_j), G_j(\Omega_j)]H_j(\Omega_j^c)}{H_j(\Omega)H_j(\Omega_j)} \\ &= \max\left[N M_j^c, \frac{G_j(\Omega_j)}{H_j(\Omega_j)}\right]\left[\frac{H_j(\Omega_j^c)}{H_j(\Omega)}\right], \end{aligned} \tag{3.21}$$

then dividing through by N and $M = \frac{1}{N}\sum_{x_i \in \Omega} x_i$ gives

$$\frac{|y_j - y_j^{'}|}{MN} \;\preceq\; \max\left[\frac{M_j^c}{M}, \frac{M_j}{M}\right]\eta_j, \;\text{ where } \eta_j = \frac{\sum_{k \neq j}\epsilon_{kj}}{\sum_k \epsilon_{kj}}, \tag{3.22}$$

and ϵ_{kj} is the level of interaction between cluster Ω_j and Ω_k as defined in (3.13).

For a given data-set, the quantities M, M_j and M_j^c are known a priori. For the error in representative compound location $|y_j - y_j'|/M$ to be less than a given value δ_j (where $\delta_j > 0$), we must choose η_j such that

$$\eta_j \leq \frac{\delta_j}{N \max\left[\frac{M_j^c}{M}, \frac{M_j}{M}\right]}. \tag{3.23}$$

3.5.3 Implementation of the Scalable Algorithm

The following steps are the steps in order to implement the scalable MEP-based algorithm:

- Step 1: Initialize $\beta = 0$ and set limits on the maximum value of β (i.e. β_{max}), and the maximum number of clusters M. Set the initial number of resources to 1, i.e. initialize $K = 1$.

- Step 2: Determine the locations of the resources by

$$y_j = \sum_{i=1}^{N} p(x_i|y_j)x_i \quad \forall j = 1, 2, \ldots, K, \text{ where } p(x_i|y_j) = \frac{p(x_i)p(y_j|x_i)}{\sum_{m=1}^{N} p(x_m)p(y_j|x_m)}.$$

- Step 3: Use the above value of y_j to update the association weights $p(y_j|x_i)$ via

$$p(y_j|x_i) = \frac{\exp\{-\beta d(x_i, y_j)\}}{\sum_{k=1}^{M} \exp\{-\beta d(x_i, y_k)\}} \quad \forall j = 1, 2, \ldots K, \quad i = 1, 2, \ldots N.$$

- Step 4: Iterate between Steps 2 and 3 till the values converge.

- Step 5: If $\beta > \beta_{max}$, perform a last iteration at $\beta = \beta_{max}$ and STOP. If not, then increase the value of β, i.e. $\beta_{k+1} = \frac{1}{\gamma}\beta_k$.

- Step 6: Determine the cluster interactions between each pair R_i and R_j and form the transition matrix

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

- Step 7: Lump the transition matrix to identify *almost invariant* regions, which are groups of clusters with strong interactions amongst them. The lumping is done in 2 ways.

- Step 8: For calculating the resource location y_j and the association weights $p(y_j|x)$ in Steps 2 and 3, ignore the effect of x_is that do not belong to the almost invariant region of y_j.

- Step 8: If $\beta > \beta_{max}$, perform a last iteration at $\beta = \beta_{max}$ and STOP. If not, then increase the value of β, i.e. $\beta_{k+1} = \frac{1}{\gamma}\beta_k$.

- Step 9: If the number of resources is less than M, check the phase transition condition for each resource. If the condition is satisfied for a resource y_j, split it by adding a resource location and redistributing the weights between them. Set $K = K + 1$.

- Step 10: Go to Step 4.

In order to find the *almost invariant* groups of cluster from the transition matrix, we can either use spectral graph partitioning methods or an iterative bound based method.

Identification of separate regions in the underlying data provides us with a tool to efficiently scale the MEP-based algorithm. The number of computations at a given iteration is proportional to $\sum_{k=1}^{s} M_k^2 N_k$, where N_k is the number of compounds and M_k is the number of clusters in the kth group. For the original MEP-based algorithm, at any iteration the number of computations is $M^2 N$ where $N = \sum_{k=1}^{s} N_k$. Thus the scalable algorithm saves computations at each iteration. This difference becomes bigger as temperature decreases since corresponding values of N_k decrease. That is, the scalable algorithm effectively runs s parallel MEP-based algorithms each with relatively smaller number of data points N_k. Moreover, as the temperature decreases,

the scalable algorithm runs an increasing number of parallel MEP-based algorithms on progressively shrinking subsets, which results in increasing savings on computations.

3.6 Simulation Results

To demonstrate the advantages of the proposed algorithm, we have applied it on a number of datasets. We show that the algorithm we have proposed simultaneously addresses the key issues of diversity and representativeness in clustering the descriptor space. At the same time, it also addresses scalability issues, which are imperative for huge datasets associated with the drug discovery process.

3.6.1 Case 1: Design for Diversity and Representativeness

As a first step, a fictitious dataset was created to present the 'proof of concept' for the proposed optimization algorithm. The dataset was specifically designed to simultaneously address the issue of diversity and representativeness in the lead generation library design. This data set consists of few points that are outliers while most of the points are in a single cluster. Simulations were carried out in MATLAB. The results for dataset 1 are shown in Figure 3.3.

The pie chart in Figure 3.3 shows the relative weight of each representative compound. Representative compounds with a large weight signify that more compounds are chosen from that area. As was required, the algorithm gave higher weights at locations which had larger numbers of similar compounds. Thus different weights at each representative compound location address the issue of representativeness in the library. At the same time, it should be noted that the key issue of diversity is not compromised. This is due to the fact that the algorithm inherently recognizes the *natural* clusters in the population. As is seen from the figure, the algorithm identifies all cluster locations. The two cluster locations which were quite diverse from the rest

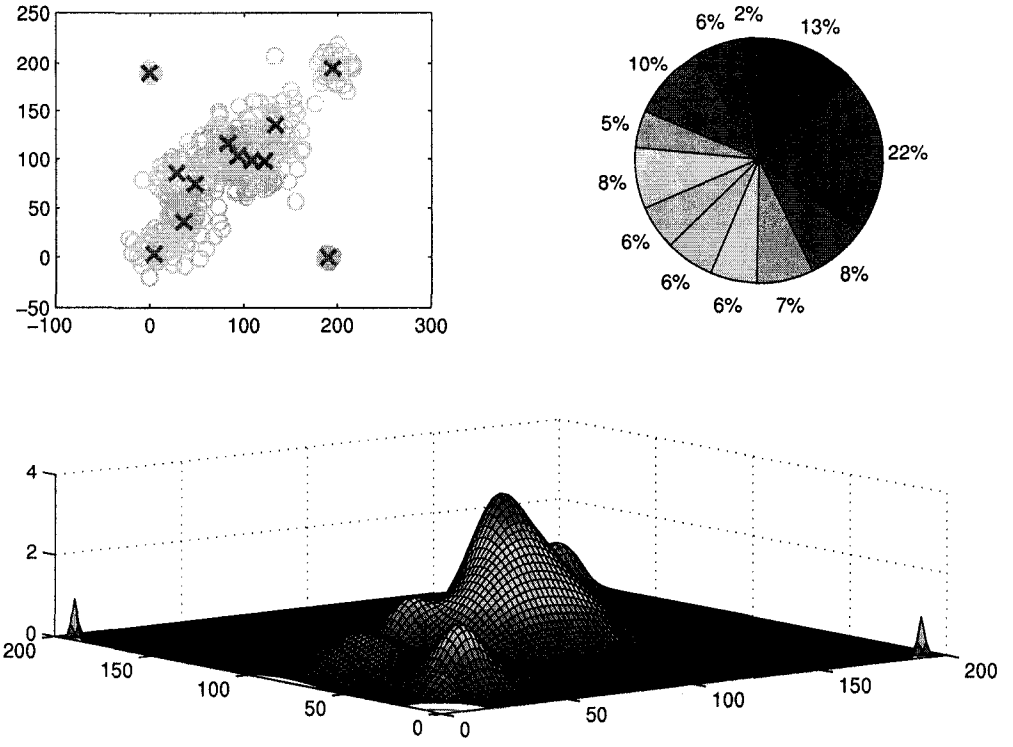


Figure 3.3: Simulation results for data-set 1; (a) The locations $x_i, 1 \leq i \leq 200$ of compounds (circles) and $y_j, 1 \leq j \leq 10$ of representative compounds (crosses) in the 2-d descriptor space. (b) The weights λ_j associated with different locations of representative compounds. (c) The given weight distribution $p(x_i)$ of the different compounds in the dataset.

of the compounds are also identified albeit with a smaller weight. As can be seen from the accompanying pie chart, the outlier cluster was assigned a weight of 2%, while the central lump was assigned a significant weight of 22%.

Results on another dataset are presented in Figure 3.4. In this case too, the dataset was specifically designed to simultaneously address the issue of diversity and representativeness in the design. As with the previous data, the algorithm automatically gave higher weights at locations which had larger numbers of similar compounds. The outlier clusters in this case contained appreciable numbers of data-points. As a result, the outlier clusters were also identified, but with a smaller weight than that of the central clusters (as indicated by the pie chart in Figure 3.4b). Each cluster is appropriately represented in the final selection.

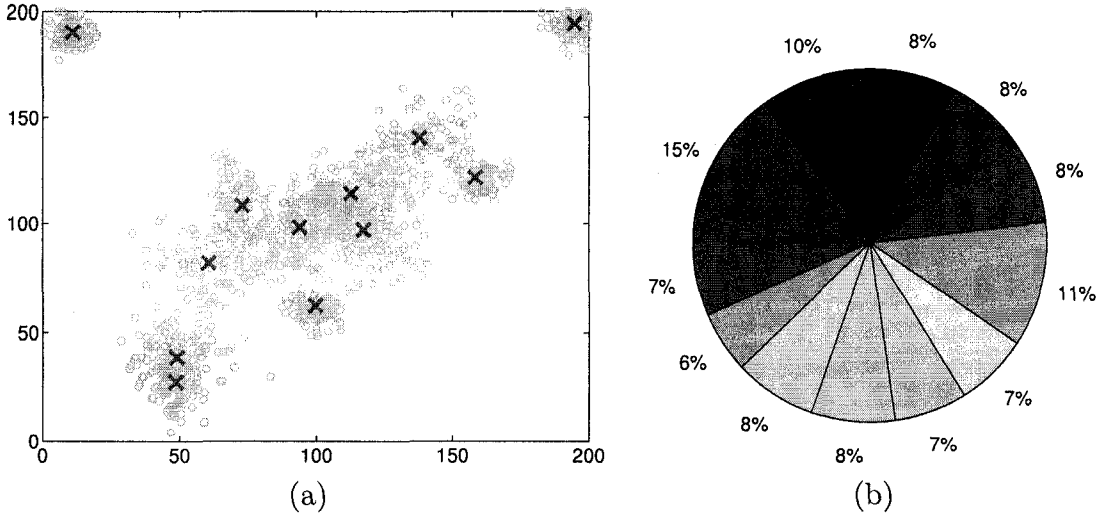


Figure 3.4: (a) Locations x_i of compounds (circles) and representative compounds $y_j, 1 \leq j \leq 12$ (crosses) (b) Weights (λ_j) associated with different locations of representative compounds.

Another feature of this algorithm is the flexibility it provides in dealing with *unique* (i.e. far-away) clusters. By properly assigning the relative importance of clusters a priori, we can choose whether to include or reject these unique clusters in the library design. Though immediate rejection of unique clusters compromises the diversity of the library, there can be scenarios where the properties of these unique cluster compounds are totally undesired in the lead generation library. Thus our approach gives us a means to deal with such scenarios effectively.

3.6.2 Case 2: Scalability and Computation Time

Taking into consideration the huge size of combinatorial libraries, it becomes essential to consider the scaling issues involved with any clustering algorithm. As was pointed out earlier, the identification of separate regions in the lower dimensional descriptor space speeds up the algorithm (as it requires lesser amount of computation) and at the same time allows for larger datasets to be clustered. In order to demonstrate this

fact, the algorithm was tested on a host of synthesized datasets. For the purpose of simulation, these 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. All the points were assigned equal weights (i.e. $p(x_i) = \frac{1}{N}$ for all $x_i \in \Omega$). Figure 3.5 shows the data-set and the representative compound locations obtained by the original MEP-based algorithm. The crosses denote the representative compound locations (y_j) and the pie-chart gives the relative weight of each representative compound (λ_j). Our proposed algorithm starts with one representa-

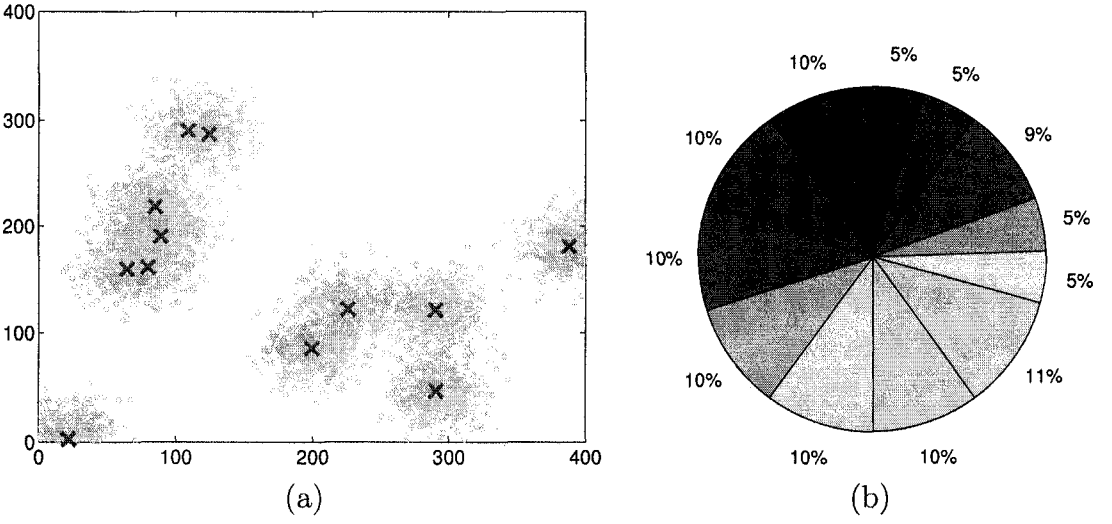


Figure 3.5: (a.) Locations $x_i, 1 \leq i \leq 5000$ of compounds (circles) and $y_j, 1 \leq j \leq 12$ of representative compounds (crosses) in the 2-d descriptor space determined from non-scalable algorithm (b.) Relative weights λ_j associated with different locations of representative compounds.

tive compound 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 3.6a shows the four separate regions identified by the algorithm (as described in Section

3.5.1) at the instant when 12 representative compound locations have been identified. Figure 3.6b offers a comparison between the two algorithms. Here the crosses

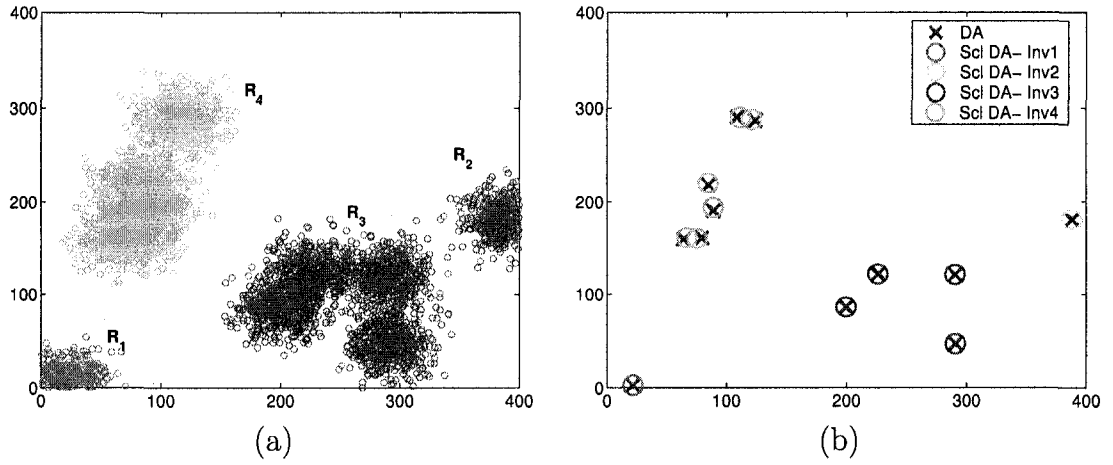


Figure 3.6: (a.) Separated regions R_1, R_2, R_3 and R_4 (denoted by different colors) as determined by the proposed algorithm (b.) Comparison of representative compound locations y_j and y'_j obtained from two algorithms.

represent the representative compound locations (y_j) determined by the non-scalable MEP-based algorithm. The circles represent the locations (y'_j) determined by the modified algorithm that we have proposed. Note that the dataset 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 and its ability to handle larger datasets. The results from the two algorithms have been presented in Table 3.1. As can be seen from Table 3.1, the proposed scalable algorithm uses just one-sixth of the time taken by the original (non-scalable) algorithm and results in only a 5.2% increase in distortion; this was obtained for $\epsilon = 0.005$. Both the algorithms were terminated when the number of representative compounds 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.

Further Examples

The scalable algorithm was applied on a number of different datasets. Results for three such cases have been presented in Figure 3.7. The dataset in Case 2 comprised of 6 randomly chosen cluster centers with 1000 points each. All the points were assigned equal weights (i.e $p(x_i) = \frac{1}{N}$ for all $x_i \in \Omega$). Figure 3.7a shows the dataset and the eight representative compound locations obtained by the proposed scalable algorithm. The pie-chart in Figure 3.7b gives the relative weight of each representative compound location. Figure 3.7c offers a comparison between the two algorithms. As in Case 2, the dataset in Case 3 comprised of eight randomly chosen cluster locations with 1000 points each. Both the algorithms were executed till they identified 8 representative compound locations in the underlying dataset. The clustering results are shown in Figure 3.7a, 3.7b and 3.7c. The dataset in Case 4 comprised of two cluster centers with 2000 points each. Both the algorithms were executed till they identified 16 representative compound locations in the underlying dataset. As was pointed out earlier, the main advantage of the modified algorithm is in terms of the computation time and scalability. Results for the three cases (from the two algorithms) have been

Algorithm	Distortion	Computation Time (sec)
Non-scalable algorithm	300.80	129.41
Proposed algorithm	316.51	21.53

Table 3.1: Comparison between the non-scalable and the proposed scalable algorithm in terms of the distortion achieved and the computation time required for the respective algorithms.

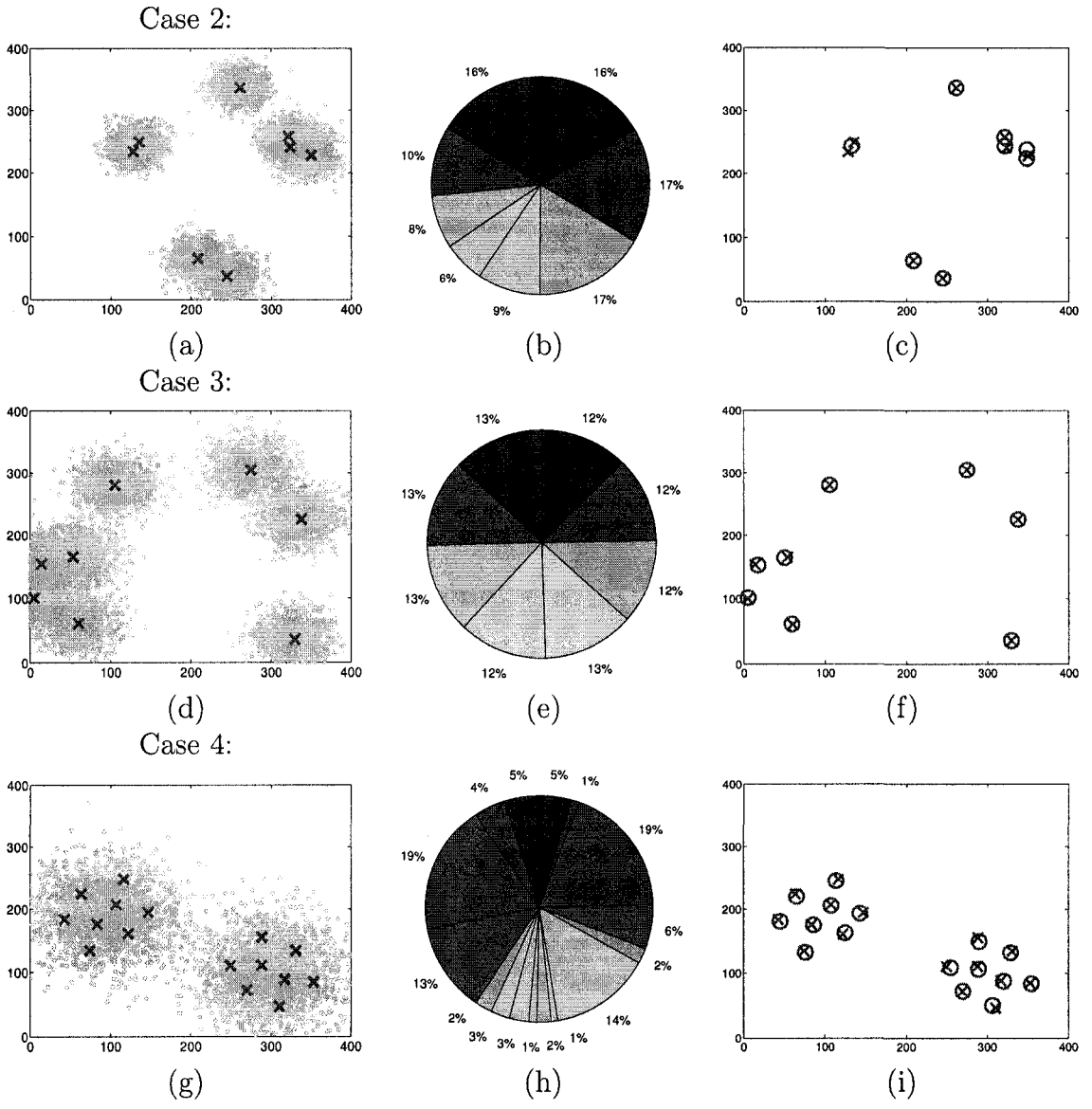


Figure 3.7: (a,d,g) Simulated data-set with locations x_i of compounds (circles) and representative compound locations y_j (crosses) determined by non-scalable algorithm (b,e,h) Relative weights λ_j of representative compound (c,f,i) Comparison of representative locations y_j and y'_j obtained from two algorithms.

presented in Table 3.2. It should be noted that both the algorithms were terminated

Case	Algorithm	Distortion	Computation Time (sec)
Case 2	Non-scalable algorithm Proposed algorithm	290.06 302.98	44.19 11.98
Case 3	Non-scalable algorithm Proposed algorithm	672.31 717.52	60.43 39.77
Case 4	Non-scalable algorithm Proposed algorithm	808.83 848.79	127.05 41.85

Table 3.2: Comparison between the non-scalable and the proposed scalable algorithm in terms of the distortion achieved and the computation time required for the respective algorithms. Results for three different datasets are presented.

after a specific number of representative compound locations had been identified. The proposed algorithm took far less computation time when compared to the non-scalable algorithm while maintaining less than 5 % error in distortion.

The proposed algorithm has also been applied on higher dimensional datasets. Results for one such dataset are presented in Figure 3.8. The dataset was created by identifying eight random locations in a region of size $200 \times 200 \times 200$. These locations were then chosen as the cluster centers. Next, the sizes of each of these clusters were chosen and all points in the clusters were generated by normal distributions of randomly chosen variance. A total of 16000 points comprised this data set. All the points were assigned equal weights (i.e. $p(x_i) = \frac{1}{N}$ for all $x_i \in \Omega$). Figure 3.8 shows the clustering results obtained by the proposed scalable algorithm. Here the crosses represent the representative compound locations (y_j) determined by our algorithm. The main advantage of the modified algorithm is in terms of the computation time

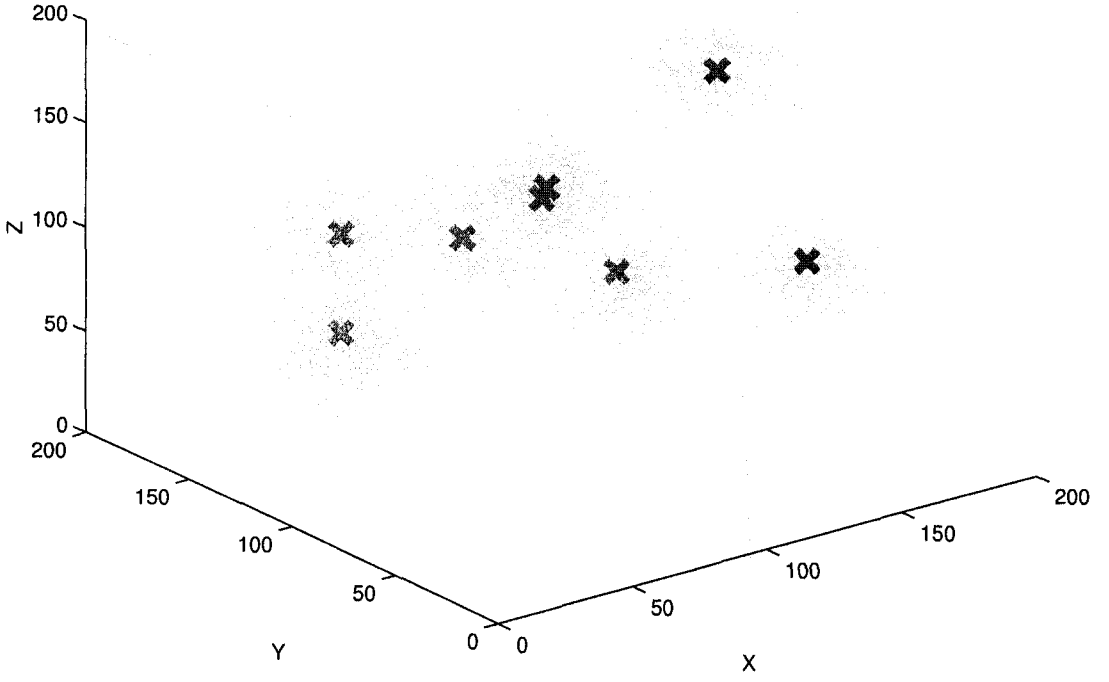


Figure 3.8: Clustering of a 3-d dataset. Locations x_i of compounds (dots) and representative compounds y_j (crosses).

of the algorithm and its ability to tackle larger datasets. The results from the two algorithms have been presented in Table 3.3. As can be seen from Table 3.3, our

Algorithm	Distortion	Computation Time (sec)
Non-scalable MEP-based	161.5	166.4
Proposed Algorithm	172.1	54.2

Table 3.3: Comparison between non-scalable and proposed algorithm.

proposed scalable algorithm uses just 54.2 seconds (as compared to 166.4 used by the non-scalable algorithm) and results in only a 6.5% increase in distortion. It should be

noted that both the algorithms were terminated when the number of representative compounds reached 8. In case of the modified algorithm, the computation time can be further reduced at the expense of error in distortion.

3.6.3 Case 3: Drug Discovery Dataset

This data set was a modified version of the test library set [1]. Each of the 50,000 members in this set were represented by 47 descriptors which included topological, geometric, hybrid, constitutional and electronic descriptors. These molecular descriptors were computed using the Chemistry Development Kit (CDK) Descriptor Calculator and JOELib [29, 75]. The Chemistry Development Kit is an open source Java library for chemoinformatics and bioinformatics while JOELib is a cheminformatics library which supports descriptor calculation. As has been shown previously, these descriptors are well suited for diversity and similarity based searching as they exhibit a neighborhood behavior [53]. This 47-dimensional data was then normalized and projected to a two-dimensional space. The projection was carried out using Principal Component Analysis (PCA) on the higher dimensional data (see Appendix C for details). Simulations were carried out on this two-dimensional dataset. We applied our proposed scalable algorithm to identify 25 representative compound locations from this dataset. The results are shown in Figure 3.9.

As was demonstrated in the previous section, the proposed algorithm gave higher weights at locations which had larger numbers of similar compounds. Compounds which are maximally diverse from all the others are identified with a very small weight. It should be noted that the non-scalable version of the algorithm could not handle the number of computations for this data set (we ran both the scalable and the original MEP-based algorithm using MATLAB on a 512 MB RAM 1.5 GHz Intel Centrino processor) due to the combinatorial nature of the problem.

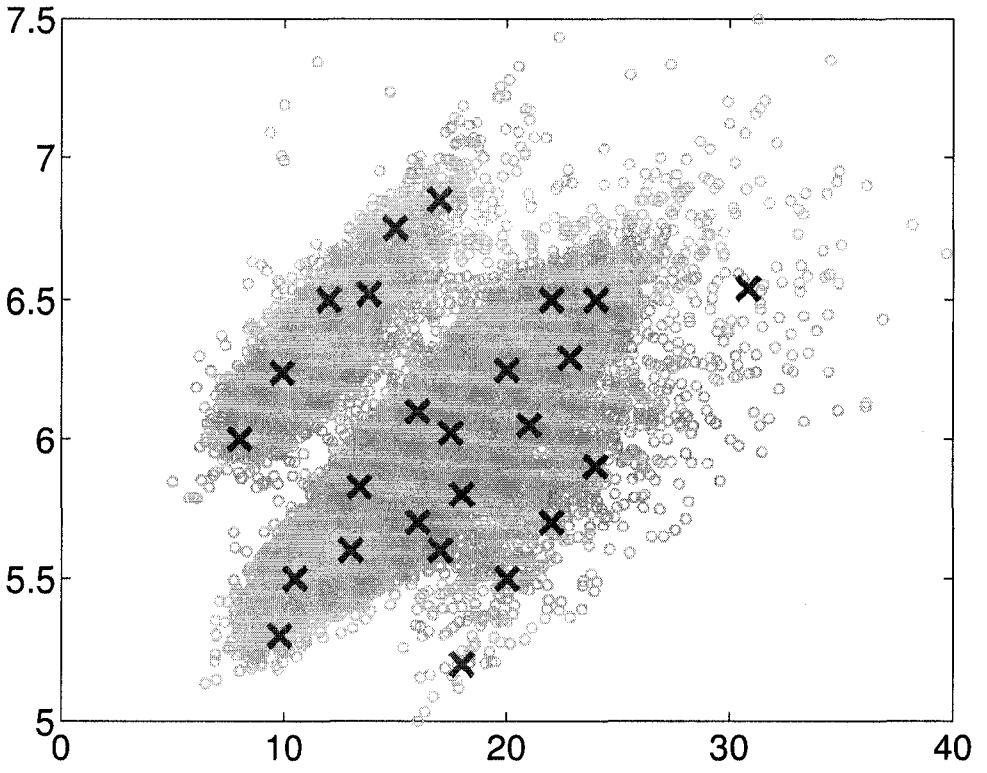


Figure 3.9: Choosing 24 representative locations from the drug discovery dataset.

3.6.4 Case 4: Additional Constraints on Representative Compounds

As was discussed in Section 3, the multi-objective framework of the proposed algorithm allows us to incorporate additional constraints in the selection problem. In this section, we have addressed two such constraints, namely the experimental resources constraint and the exclusion/inclusion constraint. Results on simulated datasets have also been presented, wherein the proposed algorithm was adapted to cater to these additional constraints in the selection criterion.

Constraints on Experimental Resources

A dataset was created to demonstrate the manner in which the algorithm presented in Section 3.4.2 accounts for the constraints placed on experimental resources. Different

compounds require different experimental supplies which are typically available in limited quantities. As was discussed in Section 3.4.2, in addition to diversity and representativeness, we include these constraints on experimental supplies into our algorithm for selecting compounds. In this dataset, the library has been divided into

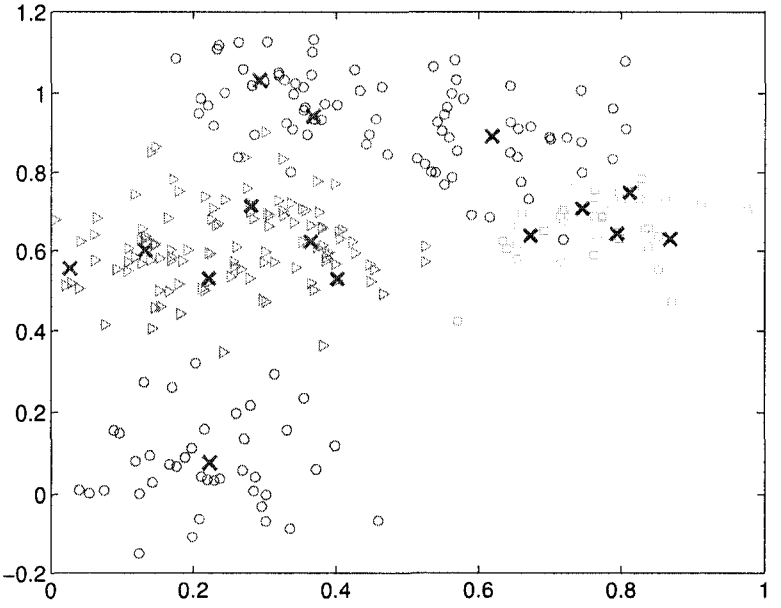


Figure 3.10: Simulation results with constraints on experimental resources.

3 classes based on the experimental supplies required by the compounds for testing, as shown in Figure 3.10 by different symbols. It contains a total of 280 compounds with 120 of the first class (denoted by circles), 40 of the second class (denoted by squares) and 120 of the third class (denoted by triangles). We incorporate experimental supply constraints into the algorithm by translating them into direct constraints on each of the representative compounds (3.7). With these experimental supply constraints in place, the algorithm was used to select 15 representative compound locations (y_j) in this dataset with capacities (W_{jn}) fixed for each class of resource. The crosses in Figure 3.10 represent the selection from the algorithm in the wake of the capacity constraints for different types of compounds. As can be seen from the selection, the algorithm successfully addressed the key issues of diversity and representativeness

together with the constraints that were placed due to experimental resources. To our best knowledge, this is the first algorithm which specifically addresses this type of constraint on experimental resources in a multi-objective optimization framework.

Constraints on exclusion and inclusion of certain properties

In the process of selecting a subset of compounds, there may arise scenarios where we would like to inhibit selection of compounds exhibiting properties within certain pre-specified ranges. This constraint can be easily incorporated in the cost function by modifying the distance metric used in the problem formulation.

Consider a case in a 2-d dataset where each point x_i has an associated radius (denoted by χ_{ij}). The selection problem is fundamentally the same, but with the added constraint that all the selected points (y_j) must be at least χ_{ij} distance removed from x_i. The proposed algorithm can be modified to solve this problem by defining the distance function as follows:

$$d(x_i, y_j) = (\|x_i - y_j\| - \chi_{ij})^2. \tag{3.24}$$

Such a distance function penalizes any selection (y_j) which is in close proximity to the data-points. This is depicted in Figure 3.11a. For the purpose of simulation, a dataset was created with points at 90 locations ($x_i, i = 1 \ldots, 90$). The blue circle around the locations x_i denotes the region in the property space that is to be avoided by the selection algorithm. The objective was to select 6 representative compounds from this dataset such that the criterion of diversity and representativeness is optimally addressed in the selected subset. The selected locations are represented by blue crosses. From Figure 3.12, note that the algorithm identifies the six clusters under the constraint that none of the cluster centers are located in the undesirable property space (denoted by blue circles). The same analysis can be extended to higher

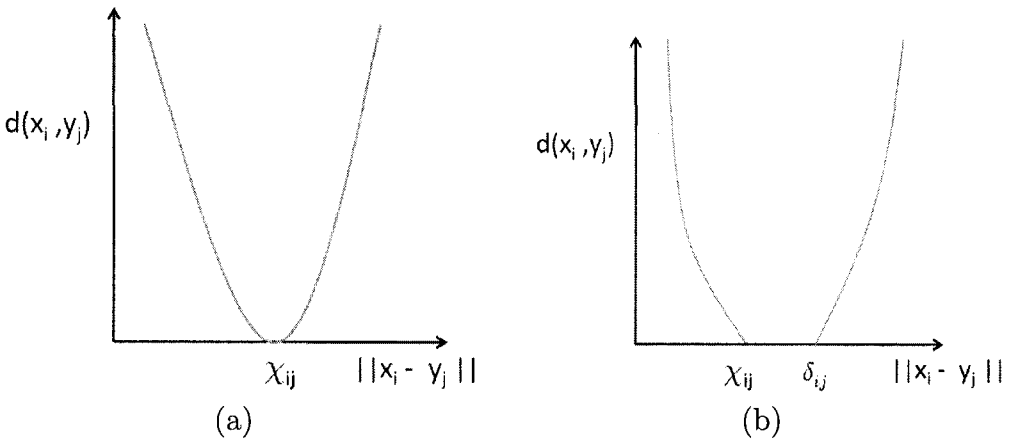


Figure 3.11: (a.) Modified 'distance' metric for exclusion of specific regions (b.) Modified 'distance' metric for exclusion and inclusion of specific regions.

dimensional descriptor spaces, thereby making it possible to exclude ranges of certain properties (e.g molecular weight, solubility). Another distance metric is shown in Figure 3.11b. As can be seen from the figure, it favors representative compounds within a certain radius around a particular compound. The flat region in the figure denotes this radius. Anything outside this region is penalized by this choice of distance metric. Depending on the scenarios encountered, the distance metric can be suitably modified to address further such exclusion and inclusion constraints.

In the preceding sections, we proposed an algorithm for the design of lead generation libraries. The problem was formulated in a constrained multi-objective optimization setting and posed as a resource allocation problem with multiple constraints. As a result, we successfully resolved the key issues of diversity and representativeness of compounds in the resulting library. Another distinguishing feature of the algorithm is its scalability, thus making it computationally efficient as compared to other such optimization techniques. We characterized the level of interaction between various clusters and used it to divide the clustering problem with huge data size into manageable sub-problems with small size. This resulted in significant improvements in the

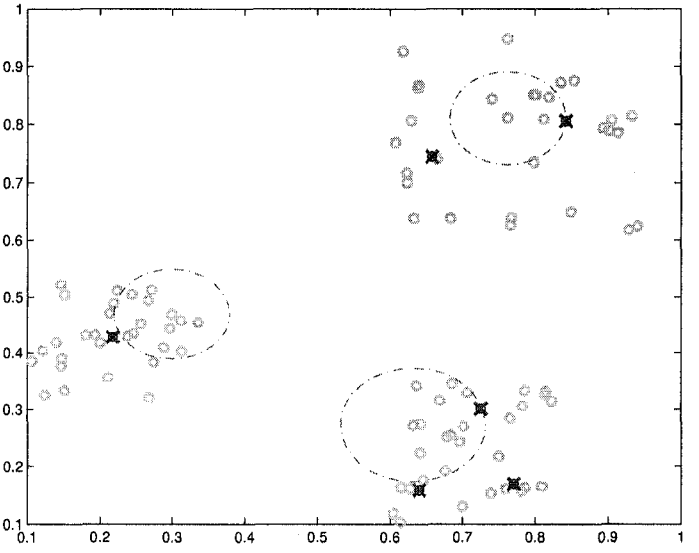


Figure 3.12: Simulation results with exclusion constraint. The locations $x_i, 1 \leq i \leq 90$ of compounds (circles) and $y_j, 1 \leq j \leq 6$ of representative compounds (crosses) in the 2-d descriptor space. Blue circles represent undesirable properties.

computation time and enabled the algorithm to be used on larger sized datasets. The tradeoff between computation effort and error due to truncation is also characterized, thereby giving an option to the end-user.

3.7 Spectral Graph Partitioning

As we had discussed in Section 3.5.1, the markov matrix obtained in (3.14) can be alternatively partitioned using the spectral graph partitioning theory. In fact, by considering the partitions on individual elements, rather than on the clusters, this can be generalized further to solve the static resource allocation problem by using spectral graph partitioning methods. In this chapter, we have formulated the static resource allocation problem as a graph partitioning problem and discussed its key advantages and drawbacks.

3.7.1 Bipartitioning Problem

Consider elements $x_i,\ 1 \leq i \leq N$ in a space Ω, with each x_i representing a node on a graph. Consider a weighted undirected graph $\mathbf{G} = (\mathbf{V},\mathbf{E},\mathbf{W})$ with the vertex set $\mathbf{V}$ and the edge set $\mathbf{E}$. The weight matrix $\mathbf{W} = [w_{ij}]$ satisfies the following properties:

- Symmetry: $w_{ij} = w(x_i, x_j) = w_{ji} = w(x_j, x_i)$.
- Positivity: $w(x_i, x_j) \geq 0\ \forall x_i, x_j \in \mathbf{V}$.

Edge weight w_{ij} is chosen such that it represents the similarity (affinity) between the two nodes.

Now consider the bipartitioning problem where we want to partition the graph $\mathbf{G}$ into two disjoint sets, A and B, such that $A \cup B = V$ and $A \cap B = \varnothing$. This can be achieved by removing the edges that connect the nodes in the sets A and B. The amount of dissimilarity between these two sets can be computed by summing up the weights of all the edges removed. This denotes the value of the $cut(A, B)$:

$$cut(A, B) = \sum_{x_i \in A, x_j \in B} w(x_i, x_j). \tag{3.25}$$

The objective of the optimal bipartitioning problem is to identify the subsets A and B such that this cut value is minimized. From this formulation, it is obvious that the mincut criterion favors cutting small subsets from the original vertex set. Such a scenario is not desirable for a resource allocation problem, where one wants to identify the natural clusters in a given dataset, and obtain balanced partitions.

For the resource allocation problem, we choose the edge weight from a Gaussian kernel function,

$$w_{ij} = w(x_i, x_j) = exp\left[\frac{-d(x_i, x_j)}{\sigma^2}\right], \tag{3.26}$$

where $d(x_i, x_j) = \| x_i - x_j \|^2$ is the Euclidean distance between the two points and the parameter σ is chosen to maintain the threshold above which $w(x_i, x_j) = 0$. Any symmetric, non-negative function monotonically decreasing with increasing distance can be used as a kernel. The parameter σ is a user-defined value, referred to as the kernel width. The choice of a correct width is critical for the performance of the algorithm. The degree of a node is defined as the sum of all the edges connected that node.

$$d_i = \sum_{j=1,N} w(x_i, x_j). \tag{3.27}$$

Define the degree matrix $D = diag(d_i)$. The Laplacian of the graph is defined as:

$$L = D - W. \tag{3.28}$$

The spectral theory of graph partitioning states that the eigenvalues of the graph laplacian L can be used for this bi-partitioning [20, 22]. The Laplacian matrix is symmetric positive semi-definite with pairwise orthogonal eigenvectors. The smallest eigenvalue $\lambda_1 = 0$ and thus the second smallest eigenvector solves the bi-partitioning problem.

It was shown in [71] that the *Normalized Cut* criterion is a better measure of the degree of dissociation between the two sets in order to get balanced partitions.

$$N_{cut}(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \tag{3.29}$$

where $assoc(A, V) = vol(A) = \sum_{x_i \in A, x_j \in V} w(x_i, x_j)$. The criterion is to minimize the N_{cut} value such that the dissociation between A and B is minimized while the association within them is maximized. As it turns out, the problem is NP-complete. However, an approximate solution can be computed by solving a relaxed problem in

real value domain.

Let $\mathbf{q}$ be a N dimensional indicator vector, such that

$$q_i = \begin{cases} 1 \text{ if } x_i \in A \\ \text{-1 if } x_i \in B \end{cases} \tag{3.30}$$

With this definition, we can rewrite the N_{cut} in (3.29) as

$$\begin{aligned} N_{cut}(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\ &= \frac{\sum_{q_i>0,q_j<0} -w_{ij}q_iq_j}{\sum_{q_i>0} d_i} + \frac{\sum_{q_i<0,q_j>0} -w_{ij}q_iq_j}{\sum_{q_i<0} d_i}. \end{aligned} \tag{3.31}$$

Since $\frac{\mathbf{1+q}}{2}$ and $\frac{\mathbf{1-q}}{2}$ are indicator vectors for $q_i > 0$ and $q_i < 0$ respectively, we can rewrite the above quantity as,

$$N_{cut}(\mathbf{q}) = \frac{(\mathbf{1}+\mathbf{q})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}+\mathbf{q})}{k\mathbf{1}^T\mathbf{D}\mathbf{1}} + \frac{(\mathbf{1}-\mathbf{q})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}-\mathbf{q})}{(1-k)\mathbf{1}^T\mathbf{D}\mathbf{1}}, \tag{3.32}$$

where $k = \frac{\sum_{q_i>0} d_i}{\sum_i d_i}$. After simplifications, the bi-partitioning problem is stated as follows:

$$\begin{aligned} &\min_{\mathbf{q}} N_{cut}(\mathbf{q}) = \min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D}\text{ - }\mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} \\ &\quad s.t.\ y(i) \in \{1, -b\},\ \mathbf{y}^T\mathbf{D}\mathbf{1} = \mathbf{0}. \end{aligned} \tag{3.33}$$

Here $y(i)$ is a $N \times 1$ indicator vector which takes on two discrete values. This describes the partition. If y is relaxed to take on all real values, then the solution of the above problem (Rayleigh quotient) can be obtained by solving the generalized eigenvalue problem:

$$(\mathbf{D}\text{-}\mathbf{W})\mathbf{y} = \lambda \mathbf{D}\ \mathbf{y}. \tag{3.34}$$

The unconstrained problem can be transformed into a eigenvalue problem of the normalized Laplacian matrix $\mathbf{L}' = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ as follows,

$$\text{(D-W)}\mathbf{y} = \lambda \mathbf{D}\ \mathbf{y} \Leftrightarrow \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}z = \lambda z, \tag{3.35}$$

where $z = \mathbf{D}^{\frac{1}{2}}\mathbf{y}$. Note that $z_0 = D^{\frac{1}{2}}\mathbf{1}$ is the smallest eigenvector with eigenvalue 0. The matrix $\mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ is positive semidefinite and thus z_0 is the smallest eigenvector. All the eigenvectors are pairwise orthogonal. In terms for y, we obtain that $\mathbf{y}_0 = 1$ is the smallest eigenvector with the eigenvalue 0 and ${z_1}^T z_0 = \mathbf{y}^T\mathbf{D1} = \mathbf{0}$, where y_1 is the second smallest eigenvector of (3.34). As a result, the Rayleigh quotient is minimized by the next smallest eigenvector z_1 and its minimum value corresponds to the eigenvalue λ_1. Thus,

$$z_1 = \underset{z^T z_0 = 0}{\arg\min} \frac{z^T\mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}z}{z^T z}, \tag{3.36}$$

which after transforming back to y gives,

$$y_1 = \underset{\mathbf{y}^T\mathbf{D1}=\mathbf{0}}{\arg\min} \frac{\mathbf{y}^T(\mathbf{D}\text{ - }\mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}. \tag{3.37}$$

Thus the second smallest eigenvector provides a solution to the normalized cut problem, but does not fully satisfy the constraints on y. Under this solution, $y(i)$ takes on a whole range of value, instead of the two discrete value specified in the constraint. For practical implementation, one needs to set a threshold to convert this continuous spectrum of $y(i)$ values to a discrete set. Hence the eigenvector has to be split into two parts to obtain the partitions. Following the same procedure, the third smallest eigenvector partitions the graph into three sets, and so on. But this results in approximation error being accumulated for every partition obtained.

3.7.2 Markov Random Walk Interpretation

An alternative formulation for the spectral graph partitioning (via normalized cut) is presented in a probabilistic framework. A random walk on a graph is finite Markov chain that is time-reversible. Every Markov chain can be viewed as a random walk on a weighted directed graph. Additionally, any time-reversible Markov chain can be viewed as a random walk on an undirected graph. Consider the nodes of the graph as the states of a Markov Chain. The transition probability from one state to the other is proportional to the weight of the edge connecting the two states. In the case of resource allocation problem, each data point represents a node on this graph while the edge weight represents the affinity between two points.

The transition probability p_{ij} of moving from node i to node j (in one time step) is given by

$$p_{ij} = \frac{w_{ij}}{d_i}. \tag{3.38}$$

Thus the matrix $P = D^{-1}W$ is the transition matrix for the random walk defined on the graph G (with row summing to 1). The spectrum of the transition matrix should give us information about the state of the random walk. All the eigenvalues of P lie in the unit circle with the largest value $\lambda_0 = 1 > \lambda_1 > \cdots \lambda_{n-1}$. Also note that the eigenvector corresponding to λ_0 is the vector $\mathbf{1_N}$ (which defines the trivial partition - the whole set). The eigenvalue problem for the transition matrix is

$$Px = D^{-1}Wx = \lambda x. \tag{3.39}$$

It should be noted that any pair (x^*, λ^*) that satisfies this problem has the property that $(x^*, 1 - \lambda^*)$ satisfies the generalized eigenvalue problem of minimizing the

Normalized cut (in (3.34)), given by

$$D^{-1}Wx^* = \lambda^* x^* \Longleftrightarrow (D - W)x^* = (1 - \lambda^*)Dx^*. \tag{3.40}$$

The second smallest eigenvector of the normalized cut corresponds to the second largest eigenvector of the Markov random walk transition matrix.

3.7.3 Stationary Density

The stationary density $p = [p_1\ p_2\ \cdots\ p_N]$ of the transition matrix is $P^T p = p$. It can be easily verified that,

$$p_i = \frac{d_i}{\sum d_j}. \tag{3.41}$$

defines the stationary density. If we start the random walk form this stationary distribution, then the probability that we will transition from set A to set B is given by,

$$P(A \longrightarrow B) = \frac{\sum_{x_i \in A, x_j \in B} p_i Pij}{\sum_{x_i \in A} d_i / \sum_{x_j \in V} d_j} = \frac{\sum_{x_i \in A, x_j \in B} w(x_i, x_j)}{\sum_{x_i \in A} d_i} = \frac{cut(A, B)}{vol(A)}. \tag{3.42}$$

Thus the Normalized cut criterion minimizes the probability that the walk will leave either of the sets A or B. This is closely related to the almost invariant sets of the random walk (lumping of the Markov matrix).

3.7.4 Multicuts Using Higher Eigenvectors

In [49], it was shown that we can use K eigenvalues of the Laplacian matrix to partition the data into K sets. This partitioning is efficient if the Markov matrix is block stochastic.

The cost function for a multi-way cut is :

$$MNCut = \sum_{i=1,K} \sum_{j=1,K,j\neq i} \frac{cut(A_i, A_j)}{vol(A_i)}. \tag{3.43}$$

The probabilistic interpretation is that the multiway cut tries to find a partition A_i which minimizes the probability of the random walk to escape from each of the sets. Partitioning results based on the multiway cut have been presented in Section 3.7.6.

3.7.5 Comparison of Deterministic Annealing with Spectral Algorithms

Note that the association probability is of the form (Gaussian Kernel centered at y_j) we started with in the spectral graph partitioning algorithm. One key difference between the two algorithms is the problem formulation itself. Both the algorithms are designed to yield 'optimal' partitions but they do so by 'optimizing' different cost functions (3.29 and 2.1). The DA problem addresses the issue of identifying resource location (y_j) for similar points and then forming clusters around these locations based on the association probabilities. Thus, it optimizes on the location y_j associated with each cluster. At the same time, it also optimizes on the association probabilities. On the other hand, the spectral partitioning algorithm solves just for the partitions and does not address the resource location issue.

DA does not aim to solve 'hard clustering' problem (i.e. associating each point to a unique cluster). Instead it solves a fuzzy clustering problem and assigns association probability of every element with every resource location. This is similar to the solution eigenvector of the graph partitioning problem (which takes on real values). For practical implementation, we threshold this vector and obtain a bi-partition.

Another interesting feature of the DA algorithm is the temperature variable (T). It is analogous to the kernel width σ. We start the annealing algorithm at a high value

of temperature and gradually decrease the temperature. This causes phase transition resulting in higher number of clusters. A higher temperature value is equivalent to a high kernel width (σ) (which results in far-off points in the graph with appreciable edge weight). As the temperature is lowered, the far-off connections are penalized as the algorithm solves a local clustering problem. Thus the DA algorithm consists of minimizing F with respect to $\{y_j\}$ and$p(y_j|x)$ starting at high values of T and tracking the minimum of F while lowering T.

The spectral method solves the partitioning problem with one fixed value of σ (which is chosen according to the problem). This choice is critical for the spectral algorithm to give meaningful partitions. In the case of DA, the annealing variable T automatically takes care of this.

One key feature of the spectral partitioning problem is its analogy to the physical process of diffusion [52, 42]. In this setting, two points are considered close only if they are connected by many short paths in the graph. Such a criterion is often important in Image segmentation problems where the aim is to identify (cluster) similar objects from a given image (Figure 3.13).

3.7.6 Simulation Results

In this Section, we have presented some partitioning results for the Multicut Spectral Partitioning problem. Comparisons of the performance of spectral method and the MEP-based algorithm have also been presented.

Figure 3.13 shows the partitioning results obtained from a multicut spectral criterion using 4 eigenvalues. The data has been segmented into four natural clusters by the algorithm. An interesting feature of this partitioning is that it is analogous to the physical diffusion process. The ring (blue) is identified as a single cluster because a diffusion process starting inside the annulus will tend to remain inside after a long time. This feature of the spectral partitioning algorithm makes it useful for image

segmentation. Figure 3.14 shows the partitioning results for a case where the Markov

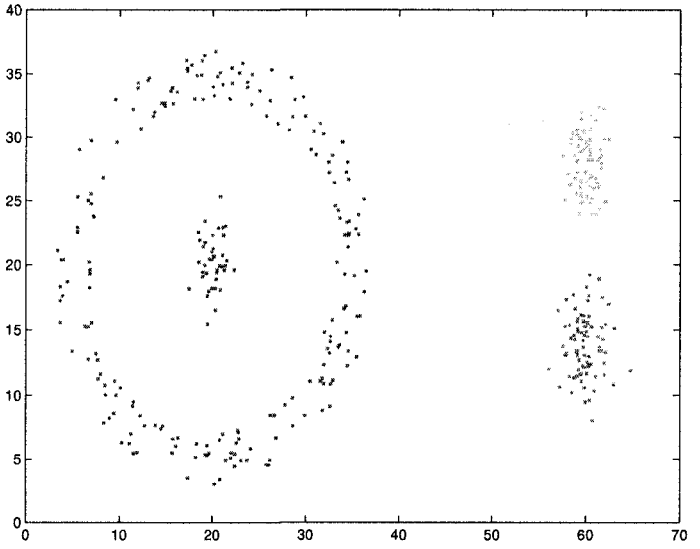


Figure 3.13: Partitioning using Ncut.

chain is lumpable. It consists of 7 natural gaussian clusters which are separated from one another. The spectral method identifies the natural clusters using the first 7 eigenvalues of the Laplacian matrix. Fig 3.15 shows the clustering results obtained from the Deterministic Annealing algorithm.

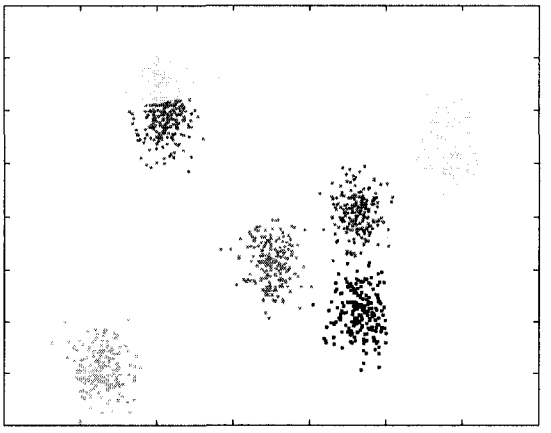


Figure 3.14: Partitioning determined by Spectral Multicut.

For small datasets, the spectral graph partitioning algorithm takes much less time than the DA algorithm. For the dataset shown in Figure 3.14, spectral method took

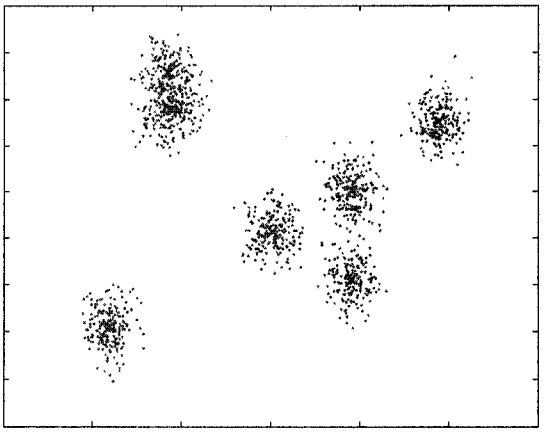


Figure 3.15: Partitioning determined by Deterministic Annealing.

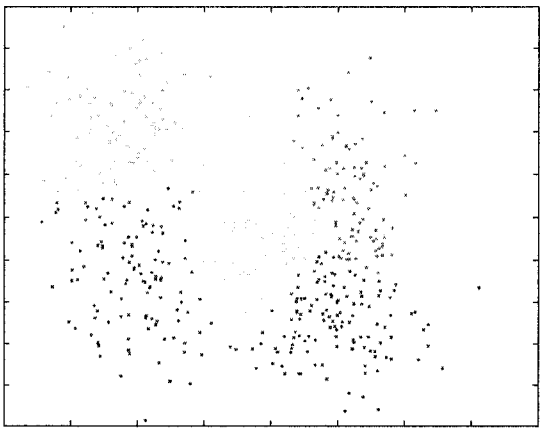


Figure 3.16: Partitioning determined by Spectral Multicut.

just 3.4 seconds while the DA algorithm took 8.36 seconds to determine the partitions.

One of the drawbacks of the Spectral Graph partitioning algorithms is their inability to deal with large datasets. The large size of the Laplacian matrix makes the computation of the eigenvalues very time intensive. To demonstrate this, we used a dataset containing two gaussian clusters with 5000 points each. The DA algorithm was able to identify the two clusters efficiently but the spectral algorithm could not be tested in MATLAB because of memory overflow during the computation of the 2nd eigenvalue of the Laplacian matrix.

The spectral algorithm requires the number of clusters apriori. The partitioning results are not very promising when the number of natural clusters is not known.

Figure 3.17 shows the partitioning results for two clusters. It is clear that the data has 3 (or 4) natural clusters. On the other hand, the DA algorithm identifies the clusters hierarchically (Figure 3.18). This points to the fact that an effective stopping mechanism has to be designed for the spectral methods.

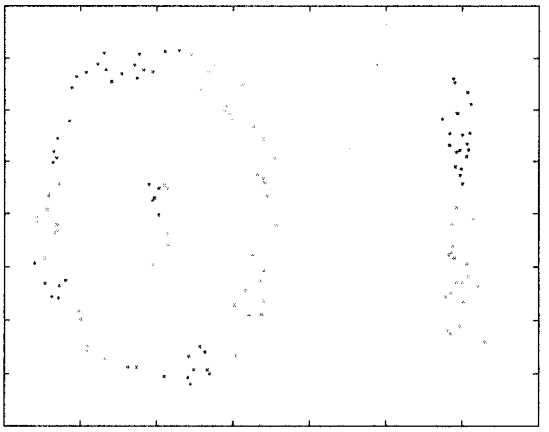


Figure 3.17: Partitioning determined by Spectral Multicut.

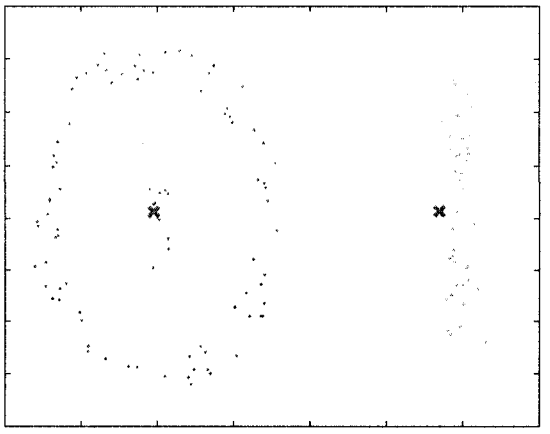


Figure 3.18: Partitioning determined by Deterministic Annealing.

Chapter 4

Scalability in MEP Framework

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. In Chapter 3, we had proposed a Markov matrix based approach for addressing the issue of scalability associated with large scale static resource allocation problems [65].

In this chapter, we develop a MEP-based framework under which the computational complexity, the inherent non-convexity, and the issue of scalability of static resource allocation problems are addressed. The framework provides a way of recursively dividing the dataset into smaller data sets and applying repeatedly an 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).

4.1 Problem Formulation

The scalable resource allocation problem is stated in a MEP framework as follows:

Definition 4.1 (Scalable Resource Allocation Problem in MEP Framework). *Given a distribution $p(x_i)$ of the elements x_i, $1 \leq i \leq N$ in a space Ω, find the set of weighting parameters $p(y_j|x_i)$ and the resource locations y_j which solve the following maximization problem:*

$$\max_{y_j, p(y_j|x)} H(x,y) = -\sum_j \sum_i p(x_i, y_j) \log p(x_i, y_j), \tag{4.1}$$

such that,

$$\sum_j \sum_i p(x_i) p(y_j|x_i) d(x_i, y_j) = F_1, \tag{4.2}$$

$$\sum_j \sum_i p(x_i) p(y_j|x_i) d(x_i, y_j) M_{ij} = F_2, \tag{4.3}$$

$$\sum_j \sum_i p(x_i) p(y_j|x_i) {N_j}^2 = F_3, \tag{4.4}$$

where,

$$H(x,y) = H(x) + H(y|x) = H(x) - \sum_j \sum_i p(x_i) p(y_j|x_i) \log p(y_j|x_i), \tag{4.5}$$

$$M_{ij} = \begin{cases} 0 \text{ if } x_i \in R_j \\ 1 \text{ if } x_i \notin R_j \end{cases}, \tag{4.6}$$

and N_j is the number of elements in cluster R_j, given by $N_j = \sum_i (1 - M_{ij})$.

Here (4.2) 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 basic MEP-based 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 (4.1)-(4.6) reflects one iteration of the algorithm where β_1 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 MEP-based 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 4.3 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 4.4 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:

$$M_{ij} = 1 - e^{-\frac{d_{ij}}{\sigma_j}}, \quad N_j = \textstyle\sum_i e^{-\frac{d_{ij}}{\sigma_j}}, \tag{4.7}$$

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.

Theorem 4.2. *The solution for the above multi-objective optimization problem satisfies the following properties.*

1. Association Weights: *The optimal association weights* $p(y_j|x_i)$ $1 \leq i \leq N$, , $1 \leq j \leq M$ *are given by the Gibbs distribution*

$$p(y_j|x_i) = \frac{e^{-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2}}{Z(\beta_1, \beta_2, \beta_3; \sigma_j)}, \tag{4.8}$$

where the partition function Z *is defined by*

$$Z_i(\beta_1, \beta_2, \beta_3; \sigma_j) = \sum_j e^{-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2}. \tag{4.9}$$

2. Centroid Property: *The optimal resource locations* y_j, $1 \leq j \leq M$ *are given by*

$$y_j = \frac{\sum_i p_{ij} \left\{ W_{ij} x_i - \frac{2\beta_3 N_j}{\sigma_j} \left[\sum_k e^{\frac{-d_{kj}}{\sigma_j}} x_k \right] \right\}}{\sum_i p_{ij} \left\{ W_{ij} - \frac{2\beta_3}{\sigma_j} N_j^2 \right\}}, \tag{4.10}$$

where

$$W_{ij} = \left[\beta_1 + \beta_2 \left(1 + e^{\frac{-d_{ij}}{\sigma_j}} \left(\frac{d_{ij}}{\sigma_j} - 1 \right) \right) \right]. \tag{4.11}$$

3. Size of the Optimal Subsets: *The size of the optimal subsets around each resource location* σ_j *is given by*

$$\frac{1}{\sigma_j} = -\frac{d^T \nu}{d^T d} = -\frac{d^T u}{d^T d} \log \left[\frac{\boldsymbol{\mu}^T b}{u^T b b^T b} \right] - \frac{d^T \log b}{d^T d}. \tag{4.12}$$

Proof. Define the Lagrangian L by

$$L(y_j, p(y_j|x), \beta_1, \beta_2, \beta_3) = H - \beta_1 F_1 - \beta_2 F_2 - \beta_3 F_3, \tag{4.13}$$

where β_1, β_2, and β_3 are the Lagrange multipliers for equality constraints in (4.2), (4.3), and (4.4).

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

$$p(y_j|x_i) = e^{-1-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2}. \tag{4.14}$$

Since the sum of associated weights is 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

$$p(y_j|x_i) = \frac{e^{-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2}}{Z(\beta_1, \beta_2, \beta_3; \sigma_j)}, \tag{4.15}$$

where the partition function Z is defined by

$$Z_i(\beta_1, \beta_2, \beta_3; \sigma_j) = \sum_j e^{-\beta_1 d_{ij} - \beta_2 d_{ij} M_{ij} - \beta_3 N_j^2}. \tag{4.16}$$

Substituting $p(y_j|x_i)$ from (4.15) in (4.13), gives

$$L^* = \sum_i p(x_i) \log Z_i(\beta_1, \beta_2, \beta_3). \tag{4.17}$$

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

$$y_j = \frac{\sum_i p_{ij} \left\{ W_{ij} x_i - \frac{2\beta_3 N_j}{\sigma_j} \left[\sum_k e^{\frac{-d_{kj}}{\sigma_j}} x_k \right] \right\}}{\sum_i p_{ij} \left\{ W_{ij} - \frac{2\beta_3}{\sigma_j} N_j^2 \right\}}, \tag{4.18}$$

where

$$W_{ij} = \left[\beta_1 + \beta_2 \left(1 + e^{\frac{-d_{ij}}{\sigma_j}} (\frac{d_{ij}}{\sigma_j} - 1)\right)\right]. \tag{4.19}$$

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

$$\sum_i p_{ij}\beta_2 {d_{ij}}^2 e^{-\frac{d_{ij}}{\sigma_j}} = 2\beta_3 N_j \left(\sum_k e^{-\frac{d_{kj}}{\sigma_j}} d_{kj}\right) \sum_i p_{ij}. \tag{4.20}$$

We need to solve (4.20) to determine the parameter σ_j. Re-writing (4.20) in vector form, we obtain

$$\beta_2 \begin{bmatrix} p_{1j}d_{1j}^2 \\ \vdots \\ p_{Nj}d_{Nj}^2 \end{bmatrix}^T \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_N \end{bmatrix} = \beta_3' u^T \rho \begin{bmatrix} d_{1j} \\ \vdots \\ d_{Nj} \end{bmatrix}^T \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_N \end{bmatrix}, \tag{4.21}$$

which can be written as

$$(\boldsymbol{\mu}^T \boldsymbol{\rho}) = (\boldsymbol{u}^T \boldsymbol{\rho})(\boldsymbol{b}^T \boldsymbol{\rho}), \tag{4.22}$$

$$\text{where, } \boldsymbol{b} = \beta_3' \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}^T = \beta_3' \boldsymbol{d},\ \boldsymbol{\rho} = \begin{bmatrix} e^{-\frac{d_{1j}}{\sigma_j}} & \cdots & e^{-\frac{d_{Nj}}{\sigma_j}} \end{bmatrix}^T, \tag{4.23}$$

$$\beta_3' = 2\beta_3 \sum_i p_{ij} = 2\beta_3 \boldsymbol{u}^T \boldsymbol{p},\ \boldsymbol{\mu} = \beta_2 \begin{bmatrix} p_{1j}d_{1j}^2 & \cdots & p_{Nj}d_{Nj}^2 \end{bmatrix}^T. \tag{4.24}$$

In order to compute the quantity σ_j, we define $\boldsymbol{\nu} = \log \boldsymbol{\rho}$ and $\gamma_j = \frac{1}{\sigma_j}$. Thus, $\boldsymbol{\nu} = -\frac{1}{\sigma_j} \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}^T$. Note that if we knew $\boldsymbol{\nu}$, then $\gamma_j = -\frac{\boldsymbol{d}^T \boldsymbol{\nu}}{\boldsymbol{d}^T \boldsymbol{d}}$. Then (4.22)

holds iff

$$\boldsymbol{\rho}^T(\boldsymbol{u}b^T\boldsymbol{\rho} - \boldsymbol{\mu}) = 0 \Leftrightarrow \boldsymbol{u}b^T\boldsymbol{\rho} - \boldsymbol{\mu} = \left(I - \frac{\boldsymbol{\rho}\boldsymbol{\rho}^T}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right) y. \tag{4.25}$$

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.,

$$\left(I - \frac{\boldsymbol{u}\boldsymbol{u}^T}{\|u\|^2}\right)\left[\boldsymbol{\mu} + \left(I - \frac{\boldsymbol{\rho}\boldsymbol{\rho}^T}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right) y\right] = 0. \tag{4.26}$$

If we assume that the solution to (4.22) (and (4.26)) exits, then the solution with least norm is given by $\boldsymbol{\rho} = S^\dagger q$,

$$\boldsymbol{\rho} \;=\; (ub^T)^\dagger \left[\boldsymbol{\mu} + \left(I - \frac{\boldsymbol{\rho}\boldsymbol{\rho}^T}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right) y\right] \text{ for some } y, \tag{4.27}$$

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

$$\boldsymbol{\rho} \;=\; \frac{1}{\|u\|^2\|b\|^2}\left[u^T\boldsymbol{\mu} + u^T y - \frac{u^T\boldsymbol{\rho}\boldsymbol{\rho}^T y}{\boldsymbol{\rho}^T\boldsymbol{\rho}}\right] b \tag{4.28}$$

$$\;=\; \Theta b, \text{ where } \Theta(\boldsymbol{\rho}) \text{is a scalar.} \tag{4.29}$$

Substituting $\boldsymbol{\rho}$ in (4.22) gives $\boldsymbol{\mu}^T\Theta b = u^T\Theta b b^T\Theta b$, which implies that,

$$\Theta = \frac{\boldsymbol{\mu}^T b}{u^T b b^T b}, \text{ and } \boldsymbol{\nu} = \log\left[\frac{\boldsymbol{\mu}^T b}{u^T b b^T b}\right] u + \log b. \tag{4.30}$$

We now substitute ν to solve for γ_j

$$\gamma_j = \frac{1}{\sigma_j} = -\frac{d^T\nu}{d^T d} = -\frac{d^T u}{d^T d}\log\left[\frac{\boldsymbol{\mu}^T b}{u^T b b^T b}\right] - \frac{d^T \log b}{d^T d}. \tag{4.31}$$

□

Thus the solution of the Lagrangian maximization problem in (4.13), and hence

that of the scalable resource allocation problem, is given by $p(y_j|x)$, y_j, and σ_j computed in (4.15), (4.18), and (4.31) 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 MEP-based algorithm, the design of the cooling law for the multi-constraint optimization problem considered in this chapter 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.

4.1.1 Critical 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

$$\frac{\partial}{\partial\epsilon}L^*_\epsilon|_{\epsilon=0} = 0 \Rightarrow \sum_i p(x_i) \sum_k p(y_k + \epsilon\Psi_k|x_i)\beta^T\Delta_{ki\epsilon}\Psi_k = 0, \tag{4.32}$$

where,

$$\Delta_{ji\epsilon} = \frac{\partial d}{\partial y_j}|_{y_j+\epsilon\Psi}\Psi_j, \ d = \begin{bmatrix} d_1(x_i,y_j) & .. & d_3(x_i,y_j) \end{bmatrix}^T. \tag{4.33}$$

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

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

$$+\sum_i p(x_i)\left(\sum_k p(y_k|x_i)\beta^T\Delta_{ki0}\Psi_k\right)^2, \tag{4.34}$$

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

$$\left(\beta^T \otimes \Delta^2_{ki0}\right) I_L - \Delta^T_{ki0}\beta\beta^T\Delta_{ki0} = 0. \tag{4.35}$$

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.

4.2 Implementation and Simulation Results

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 $\beta = 0$ and choose an initial value of σ_j such that

$$M_{ij} = 1 - e^{-\frac{d_{ij}}{\sigma_j}} \simeq 0 \ \forall x_i, i = 1, 2, \ldots, N.$$

- Step 2: Set limits on the maximum value of β (i.e.β_{max}), and the number of clusters M. Set the initial number of resources to 1, i.e. initialize $K = 1$.

- Step 3: Initialize the resource location y_j at the centroid of the dataset.

- Step 4: Use the value of y_j and σ_j to compute the weighting functions $p(y_j|x)$ according to

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

  For computing the distance $d(x, y_j)$, truncate the terms that lie outside the radius specified by σ_j.

- Step 5: Compute the optimal σ_j, given by

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

- Step 6: Use the updated values of $p(y_j|x)$ and σ_j from Steps 4 and 5 to update the resource locations y_j, given by

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

where

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

As before, while computing d_{ij}, ignore the terms that lie outside the radius specified by σ_j.

- Step 7: Iterate between Steps 4, 5, and 6 till the values converge.
- Step 8: If $\beta > \beta_{max}$, perform a last iteration at $\beta = \beta_{max}$ and STOP. If not, then increase the value of β, i.e. $\beta_{k+1} = \frac{1}{\gamma}\beta_k$.
- Step 9: If the number of resources is less than M, check the phase transition condition for each resource. If the condition is satisfied for a resource y_j, split it by adding a resource location and redistributing the weights between them. Set $K = K + 1$.
- Step 10: Go to Step 4.

To implement the splitting process in Step 9, we place two identical resources at each resource location and perturb one of them by a small amount δ when we update the value of β. Both these resources are assigned a weight $\frac{p(y_j)}{2}$. Till the phase transition condition is not met, the resources y_j and $y_j + \delta$ remain merged together during the subsequent iterations. At the instant when the phase transition condition is satisfied for cluster j, i.e.,

$$\left(\beta^T \otimes \Delta^2_{ki0}\right) I_L - \Delta^T_{ki0}\beta\beta^T\Delta_{ki0} = 0, \text{ where } \beta^T \otimes \Delta^2_{ki0} = \sum_l \beta_l \frac{\partial^2 d_l(x_i, y_k)}{\partial y_k^2},$$

the resources y_j and $y_j + \delta$ move further apart, while the other pairs remain merged. This effectively causes an increase in the number of resources, and amounts to resource y_j being split into two entities.

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 Steps 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 MEP-based 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

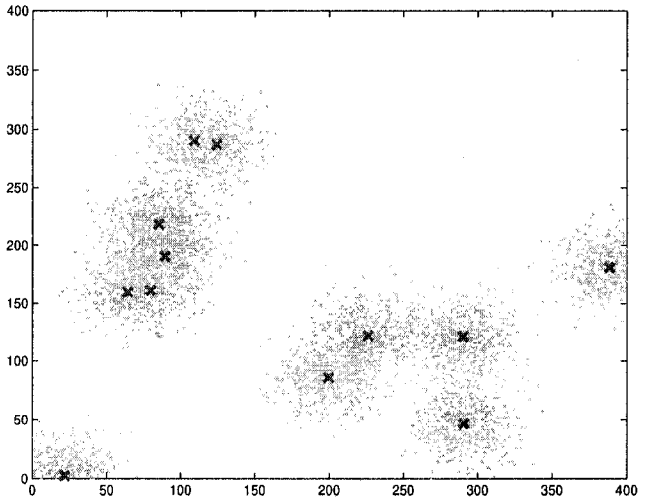


Figure 4.1: Simulated dataset, x_i, $i = 1, \ldots 5000$ (green dots) and resource locations, y_j $j = 1, \ldots, 12$ (blue crosses).

one cluster at the centroid. As the algorithm progresses, the clusters are split and association probabilities are assigned according to Equation 4.15. Figure 4.1 shows the dataset and the final resource locations. The resource locations (y_j) are computed according to Equation 4.18. 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 4.1, 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.

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

Table 4.1: Coverage costs and computation times for two algorithms.

The critical temperature analysis in Section 4.1.1 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.

Before concluding the chapter, we revisit the key contributions in the chapter. 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.

Chapter 5

Dynamic Coverage and Clustering Problems

With recent advances in geographic information systems, geopositioning and wireless sensor networks, there is a growing interest in developing algorithms for deployment of mobile resources that continuously *cover* a set of mobile sites in a given region. In particular, problems related to determining clusters in ensemble of moving objects have received considerable attention lately. These problems can be considered as further generalization of the static resource allocation problem that we discussed in the preceding chapters, with the major difference that the sites and resources are mobile. Such problems have numerous applications, such as in developing automatic deployment and tracking algorithms for surveillance and military applications [12, 23], in finding dynamic clusters in groups of animals in studies of their migration patterns [6], in weather forecasting by clustering cyclone patterns [13], and in routing traffic and detecting traffic jams by clustering traffic flow [70].

The complexity of the dynamic problem addressed in this chapter is further compounded by the inherent dynamics associated with each data point (moving objects). These moving objects could be mobile threat locations in a battlefield scenario, forest fires, unmanned vehicles, or swarms, depending on the context of the underlying problem. The task at hand is to design a velocity field for resources such that coverage is maintained over a time horizon. The mobile resources can be alternatively viewed as *cluster centers* that continuously identify and track groups of moving objects. Thus static *locations* of each data-point are replaced by velocity fields, and from a naive computational viewpoint, the dynamic problem can be regarded as a time-indexed

set of static problems. Besides, adding dynamics introduces new complexity to the notions of coverage due to the dynamic nature of cluster sizes, number of clusters, and relative distances in between the individual elements.

Problems related to the dynamic coverage problems are dealt with in [12, 23, 13, 70], where a special emphasis is given to distributed implementations, that is, under limited information flow between individual elements. The emphasis on distributed algorithms is well justified since the underlying computational costs incurred for centralized schemes are impractical and inviable for implementation in many application areas. For instance, the limited range of sensors prohibit centralized schemes in many sensor network scenarios. However, the distributed algorithms are prone to converge to one of the many local minima that typically riddle the coverage functions. As a consequence, the performance of these algorithms (in terms of the coverage cost) is very sensitive to initial placement of the resource locations.

In spite of their growing interest and wide-ranging applications, dynamic clustering problems are still not characterized or defined in a general framework. The terms *cluster* and *coverage* are subjective and usually have different interpretations, depending on the context of the problem. In some cases, a natural quantification of coverage is not easy to obtain, which results in added computational difficulty. For instance, in [44], the authors propose the concept of a micro-moving cluster (MMC), which denotes a group of objects that are not only close to each other at the current time, but also likely to move together for a while. Each MMC maintains a bounding box, which is a measure of its size. If the size of the bounding box exceeds a certain threshold, the MMC is split. Different clustering algorithms (for static data) can then be implemented on the MMC, instead of the individual points. However, because of the periodic maintenance of bounding boxes, the number of maintenance events dominates the computation time of the algorithm. The approach in [30] relies on a tradeoff between cluster quality and efficiency, but often results in clusters with

large radii [38]. In [40], methods are proposed that automatically determine clusters from the historical trajectories of moving objects. These methods compare clusters at consecutive snapshots and determine the moving clusters. However, the comparison costs for such techniques are high. In [80], a histogram technique based on a clustering paradigm is proposed, where however the authors observe that the task of updating the histograms severely slows down the computation.

In this chapter, we provide a general framework to formulate and solve dynamic clustering problems which solves for both their coverage as well as tracking aspects. More specifically, we adapt the notion of coverage to the dynamic setting, resolve the inherent trade-off between the *resolution* of the clusters in the solution and the computation cost, and provide flexibility to incorporate different dynamic specifications such as resolution and splits, while being able to avoid shallow local minima. The algorithms developed based on this framework are hierarchical in the way that they progressively seek finer subclusters from larger clusters. This hierarchical clustering nature of algorithms is heuristically developed so that they avoid local minima and are insensitive to initial placement of the resources. A characteristic feature of the proposed framework is its ability to detect *natural* cluster-centers in the underlying dataset, without the need to initialize or define clusters a priori. We define the notion of instantaneous coverage in a combinatorial optimization framework, and propose algorithms that achieve progressively better coverage with time. The algorithms are based on the idea of soft clustering, wherein each moving object is assigned to every cluster via a weighting parameter, thereby overcoming the need to define clusters in precise terms. This makes the algorithms easily adaptable to solve a number of related problems in dynamic clustering. The computational complexity of the resulting algorithms is resolved by exploiting the structure of the problem. The algorithm, as it proceeds, becomes more 'local', that is, the computation of clusters becomes less sensitive to 'far off' sites. This feature is exploited to make these algorithms

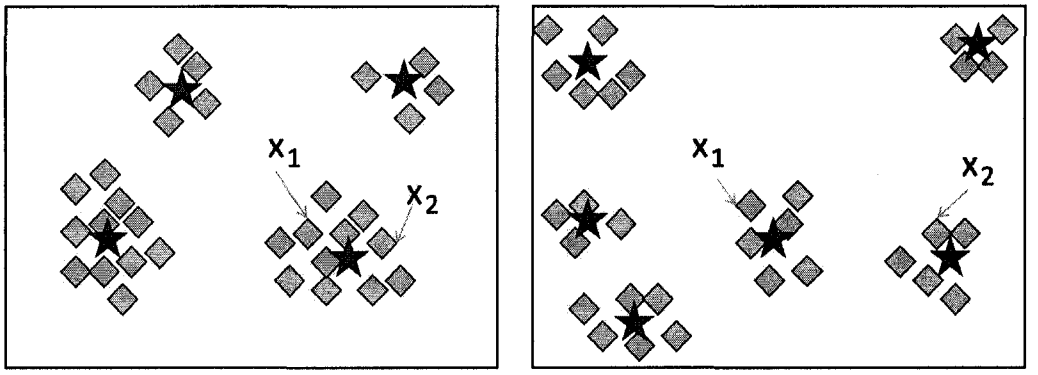
scalable and computationally efficient. Simulation results that employ algorithms based on this framework show improvements in computation times by as much as eight fold over other methods, and cost improvements as large as 13% for the same computation times. Furthermore, the simulations demonstrate the flexibility of this framework, whereby coverage issues such as resolution of clusters are answered, that other existing methods do not accommodate.

This chapter is organized as follows. We discuss the general setting of the dynamic coverage problem in Section 5.1 and highlight the key issues and challenges that need to be resolved in order to solve such problems. We present a frame-by-frame approach for clustering static data and analyze the features that make it amenable to solving the dynamic problem. The solution strategy for solving the clustering problem for dynamic data is presented in Section 5.2. The implementation and simulation results on a variety of datasets are presented in Section 5.3. Directions for increasing the computation efficiency of the algorithm, improving its robustness, and other ideas for future work are discussed in Section 5.4. Finally we conclude the chapter by revisiting the most important results obtained herein and identifying future goals.

5.1 Background and Problem Formulation

5.1.1 Problem Setting

Consider a scenario where the problem is to detect and track a group of moving objects (referred to as *sites*) in a given area. An illustrative schematic is given in Figure 5.1. The motion of these objects is such that they may move in different clusters, change cluster associations, and clusters themselves can split and rejoin over time. For instance, sites x_1 and x_2 are in the same cluster at the time instance shown in (a). Both x_1 and x_2 change their *resident* clusters between the time instances (a) and (b). Alternatively, this problem can also be posed as a coverage problem, where



(a) (b)

Figure 5.1: Clustering moving objects in a given area. (a) and (b) denote two snapshots (each at a distinct time instance) of an area with dynamic sites and resources. The squares and stars denote the positions of the sites and resources respectively. In a typical coverage problem, the task is to identify clusters (within this domain) in real time and track them over time. Objects in different clusters may move away from one another, form new clusters or exhibit other such movement, thereby resulting in an increase or decrease in the number of clusters. Sites x_1 and x_2 reside in the same cluster at the time instance shown in (a). A split causes them to reside in different clusters at the time instance shown in (b).

the aim is to successively identify representative objects (referred to as *resources*) in the area, such that they provide adequate *coverage* of the moving sites at all times. The number of these representative objects is far less than that of the moving sites. Each resource can be thought of as a cluster center. These problems arise in diverse applications such as deployment, tracking and reconnaissance in surveillance applications [12], animal migration study [6], weather prediction [13], traffic flow detection [70] and, medical imaging studies [24].

In this chapter, we consider a domain $\Omega \subset \mathbb{R}^2$ with N mobile sites and M resources, where $N \gg M$, on a time horizon in $[0\ \infty)$. The locations of ith mobile site ($i \leq N$) and jth resource ($j \leq M$) at time instance $t \in \mathbb{R}$ is represented by $x_i(t) = [\xi_i(t)\ \eta_i(t)]^T \in \mathbb{R}^2$ and $y_j(t) = [\rho_j(t)\ \omega_j(t)]^T \in \mathbb{R}^2$ respectively. For notational convenience, we sometimes use x_i and y_j in place of $x_i(t)$ and $y_j(t)$, where the time dependence is inherently assumed and is clear from the context. We impose dynamics

by prescribing velocity fields $\phi_i(x,y,t) \in \mathbb{R}^2, i \leq N$ on ith site and $u_j(t) \in \mathbb{R}^2, j \leq M$ on the jth resource, where x and y represent locations of all sites and resources respectively. More precisely we have a domain Ω with N sites $\{x_i\}$ and M resource locations $\{y_j\}$, whose dynamics are given by

$$\begin{aligned} \dot{x}(t) &= \phi(x(t), y(t), t), \;\; x(0) = x_0 \\ \dot{y}(t) &= u(t), \;\; y(0) = y_0 \end{aligned} \quad \Leftrightarrow \quad \dot{\zeta} = f(\zeta, t) \tag{5.1}$$

where $x(t) = [\; x_1(t) \;\; x_2(t) \;\; \cdots \;\; x_N(t) \;]^T, y(t) = [\; y_1(t) \;\; y_2(t) \;\; \cdots \;\; y_M(t) \;]^T,$

and $\zeta = [x^T \; y^T]^T$. This can be considered as a control system where the control field u is to be determined for the M mobile resources, and their initial locations ($\mathbf{y_0}$), such that a notion (yet to be specified) of coverage is maintained.

5.1.2 Challenges and Objectives

One of the main challenges in addressing the dynamic coverage problems stems from the difficulty in quantifying the performance objectives. The notion of coverage becomes an important issue in modeling these problems. We adopt the concept of *distortion* and its variants from the data compression literature (that deal with static coverage problems) as a metric for coverage, and adapt them to make them suitable to dynamic setting. Distortion, in a static coverage problem, is a measure of the (weighted) average distance of any site location to its nearest resource location. Thus assuming a static problem ($\phi(t) \equiv 0, u(t) \equiv 0$), the distortion measure is given by

$$D(x,y) = \sum_{x_i \in \Omega} p(x_i) \left\{ \min_{1 \leq j \leq M} d(x_i, y_j) \right\}, \tag{5.2}$$

where $p(x_i)$ represents the weight of the site x_i and $d(x_i, y_j)$ is a distance metric defined on Ω, for instance it can be given by Euclidean distance $d(x_i, y_j) = \|x_i - y_j\|^2$.

Thus, for a given set of site locations $\{x_i\}$, $1 \leq i \leq N$, the set of resource locations $\{y_j\}$, $1 \leq j \leq M$ that achieves lower distortion results in a better coverage.

Another concept that is difficult to model, irrespective of the static or dynamic setting, is the notion of clusters. More specifically, it is difficult to define the size of a cluster. This difficulty primarily stems from the fact that clusters are hierarchical in nature. Each cluster can be thought of as comprising of smaller subclusters (Figure 5.2), and in the limiting case, every moving object in itself can be thought of as a distinct cluster. On the other hand, the entire domain can be thought of as a cluster. This motivates assigning a notion of *resolution* of a cluster. In most applications, the number of *natural* clusters in the underlying data is not known a priori. Iterative algorithms that identify finer and finer (higher resolution) clusters, do progressively reduce the coverage cost function, but at the expense of increased computation. These issues with modeling coverage function and clusters become even

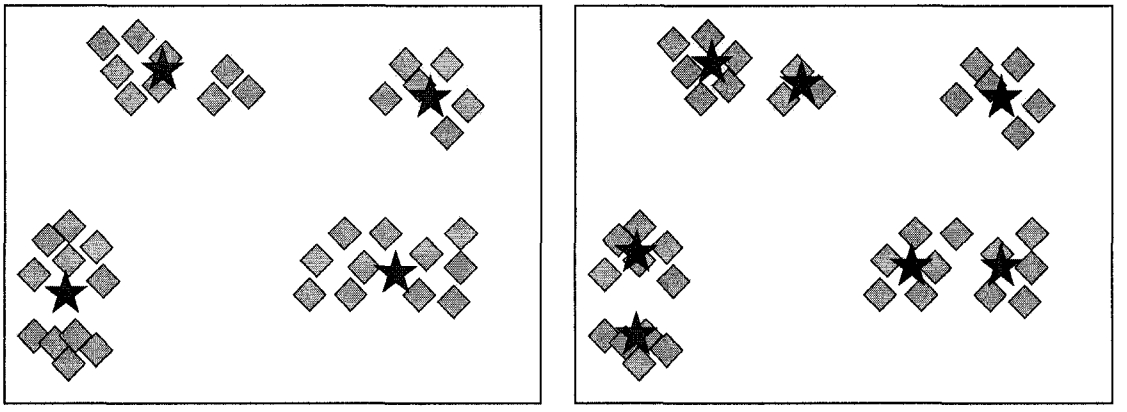


Figure 5.2: The clustering solution in (a) has identified four coarse clusters centers $y_j, 1 \leq j \leq 4$ (shown by blue stars) in the underlying data $x_i, 1 \leq i \leq 37$ (shown by orange squares). On the other hand, the solution in (b) has identified 7 finer clusters at a higher resolution on the same underlying data. The solution in (b) has lower distortion D than (a), but at the expense of higher computation time.

more challenging in the dynamic coverage problems. In contrast to static setting, the

coverage function has to adjust to and appropriately reflect the enlargement, shrinkage, splitting, coalescing, and other such dynamic components of clusters. This adds further variability in the control design, where, for instance, $u(t)$ can be designed to obtain different levels of resolution at different clusters at different times for same initial data and computational cost.

The computational complexity, which forms the main challenge in coverage problems, stems from the inherent non-convex nature of the distortion function (5.2). As discussed in [57], the distortion function, which even in the static clustering problem, is riddled with numerous local minima. Consequently, the problem necessitates designing algorithms that do not get stuck at local minima, while also avoiding expensive divide and search strategies. This complexity is further aggravated by additional time component in the dynamic setting. These challenges necessitate development of a framework for computationally efficient algorithms that adapt the notion of coverage to dynamic setting, address the inherent trade-off between the *resolution* of the clusters in the solution and the computation cost, and have the flexibility to incorporate different dynamic specifications such as resolution and splits, while being able to avoid shallow local minima. Such a framework is proposed in this section. More specifically we formulate a coverage problem and propose algorithms that determine the resource velocity fields $u(t)$, such that they *track* each cluster over time. If a cluster were to split, the resource locations mimic its behavior, thereby maintaining coverage.

5.1.3 Frame-by-Frame Approach

Since the dynamic data is a time-indexed series of static data, the simplest approach is to perform static clustering periodically. However, this simplistic approach has its own pitfalls. If the time between two successive clustering events is short, the algorithm is unnecessarily expensive because none of the spatial clustering from previous time-

steps are exploited for the clustering at the current and future time-steps. On the other hand, if the period is long, the clustering obtained at the previous time-step will not provide adequate coverage, or tracking in the interim to the current time-step. In short, such a simplistic approach ignores the spatio-temporal aspects of clustering moving data. However, this approach gives insights into the various challenges, such as computational complexity in static coverage problems, that are inherited by the the dynamic problems. The MEP-based algorithms that we presented for the static resource allocation problems can be readily used for the frame-by-frame approach. We first analyze the properties of those algorithms and then exploit them to form a basis for framework for the dynamic setting.

5.2 The Dynamic Clustering Problem

Solving the instantaneous problem at each time step does provide the solution for the coverage problem over a finite horizon. However, as discussed earlier, in most applications, this frame-by-frame approach is not computationally viable, since it requires multiple iterations at each time step. That is, at each frame, the static clustering problem must be solved by varying the *temperature* from a very high value to zero, and then resolved repeatedly in each subsequent frame.

In this section, we provide a general framework for formulating the dynamic clustering problem and determining computationally efficient algorithms that resolve issues of numerous local minima, quantifying coverage cost and cluster resolution, spatio-temporal smoothening, achieving trade-offs between computational cost and resolution, and flexibility to account for tracking clusters, their splits and real-time specifications of cluster-resolutions. Broadly stated, the aim of the dynamic clustering algorithm is to successively identify and track clusters of moving objects.

The concept of Free energy serves as a good metric for coverage, and together

with the annealing process, provides a basis for an algorithm that is independent of the initial choice of resource locations and better suited to avoid local minima. The hierarchical nature of the annealing process gives a way of identifying natural clusters and distinguishing their resolutions.

The main conceptual hurdle in adapting these static concepts for the dynamic setting comes from choosing the appropriate cooling rates (as in (2.66)) relative to the time-scales of the site dynamics. For example, we can design the rate of cooling to be much faster than the given dynamics of the sites, to have an algorithm which is similar to the frame-by-frame approach. The sensitivity-to-temperature property in Theorem 2.4 gives a solution to this problem. Since the resource locations are virtually independent of temperatures in between two successive critical temperatures, the design of cooling laws is not critical in between these critical temperatures. Since critical temperatures are indicative of splits and resolution, decreasing temperature values matter only for forcing splits or obtaining high resolution. Note that in a dynamic setting with site-locations changing with time, the cluster splits at time t are still identified by critical temperatures given by $\beta_c^{-1} = 2\lambda_{\max}(C_{x(t)|y_j(t)})$ (once the resource locations $y_j(t)$ are at cluster centers). However, unlike the static case, this condition for splitting can be reached due to dynamics of x and y in addition to changing the values of β. For instance, for a fixed value of the parameter β over a time horizon $[t_0, t_1]$, it is possible that the above splitting condition is achieved since x and y can evolve under the dynamics such that at some time $t_c \in [t_0\ t_1]$, $\beta^{-1} = 2\lambda_{\max}(C_{x(t_c)|y_j(t_c)})$. Thus $C_{x|y_j}$ identifies cluster splits in the dynamic setting, too.

The above interpretation of the sensitivity-to-temperature property allows us to separate the dynamic clustering problem into two main subproblems:

1. Tracking cluster centers.

2. Monitoring splitting conditions.

In our framework, we take decisions to the split resource locations only after they have reached the cluster centers. This establishes a priority between the above two problems, in which tracking cluster centers takes precedence over splitting the resource locations. We design $u(t)$ such that cluster centers are reached, and ensure that they are tracked (within some tolerance). We monitor the cluster splits which result due to site-dynamics or the cooling law. These cooling laws can be designed to force splits in order to obtain higher resolution. The resulting algorithm, as well as the incorporation of user specified decisions on splits and resolutions, are presented later in this section. We first present certain properties of the Free energy and its time derivative, which form the basis for addressing both the problems of tracking cluster centers and monitoring splitting conditions.

5.2.1 Free Energy Properties

The crux of the algorithm for the static setting is that it replaces the notion of distortion in (5.2) by free energy as a metric for coverage. Recall that the free energy is given by

$$F(y) = -\frac{1}{\beta}\sum_{i=1}^{N} p(x_i)\log\sum_{k=1}^{M}\exp\{-\beta d(x_i, y_k)\}. \tag{5.3}$$

In order to achieve the objective of tracking the cluster centers without resorting to the frame-by-frame approach, we formulate a control problem, where we design the velocity field $u(t)$ for the resource locations such that the time derivative of the Free Energy function, $\frac{dF}{dt}$ is non-positive along the trajectories of $x(t)$ and $y(t)$. Such a formulation not only addresses the drawbacks of the frame-by-frame clustering approach, but also helps in preserving the advantageous attributes of the MEP-based clustering algorithm for static data. Moreover, the conditions where the control

authority is lost, that is where $\frac{\partial F}{\partial y} = 0$, provide direct connections to the splitting conditions and cooling laws that we encountered in the static setting. We summarize the properties of free energy and its time derivative in the following theorem.

Theorem 5.1. *Let F given by (5.3) be the free energy for the sites and resources x_i, $1 \leq i \leq N$ and y_j, $1 \leq j \leq M$, whose dynamics are defined by (5.1). Then*

1. Positivity: *$F(\zeta, t) > 0$, for all ζ.*

2. Structured derivative: *The derivative of the free energy term has the following structure:*

$$\frac{dF}{dt} = 2\zeta^T \Gamma(\zeta) f(\zeta), \quad \Gamma = \begin{pmatrix} I_2 \otimes P_1 & -I_2 \otimes P_{12} \\ -I_2 \otimes P_{12}^T & I_2 \otimes P_2 \end{pmatrix},$$

   where $\otimes$ represents the matrix Kronecker product, I_2 is a 2×2 identity matrix, $P_1 = diag(p(x_i)) \in \mathbb{R}^{N \times N}$; $P_{12} = [p(x_i, y_j)] \in \mathbb{R}^{N \times M}, p(x_i, y_j) = p(x_i)p(y_j|x_i)$; and $P_2 = diag\,(p(y_j)) \in \mathbb{R}^{M \times M}$ where $p(y_j) = \sum_i p(x_i, y_j)$. The matrix Γ has following properties

   (a) *Γ is a symmetric positive semidefinite matrix for all ζ.*

   (b) *Γ can be decomposed as $\alpha(I - W)$ where $\alpha > 0$, I is the identity matrix and W is a symmetric doubly stochastic matrix with its spectral radius $\rho(W) = 1$.*

3. Lack of dynamic control authority at cluster centers: *The derivative $\frac{dF}{dt}$ becomes independent of control, that is, $\frac{\partial}{\partial u}\left(\frac{dF}{dt}\right) = 0$ (or equivalently the partial derivative $\frac{\partial F}{\partial y} = 0$) only at those time-instants t_c when the resource locations $y_j(t_c)$ are at the cluster centers, that is only when $y_j(t_c) = \sum_{i=1}^{N} p(x_i(t_c)|y_j(t_c))x_i(t_c)$ for $1 \leq j \leq M$.*

4. Feasibility of tracking cluster centers: *For any* K_o, $\delta > 0$ *and* $t_0 > 0$, *there exists a control* $u(t)$ *such that* $\left\| y(t) - \sum_{i=1}^{N} p(x_i(t)|y_j(t))x_i(t) \right\| < \delta$, *for* $t > t_0 + \Delta t$, *where* $\Delta t < \frac{F(t_0)}{K_o \delta^2}$ *and* $\|u(t)\| < 2\left(1 + K_o + \frac{K_2}{\delta}\|\Phi\|\right) w_1$, $K_2 < 1$ *and* $2w_1 < diameter(\Omega)$.

5.2.2 Proof for Theorem 5.1

Structured derivative. From (5.3), we have

$$F = -\frac{1}{\beta} \sum_i p(x_i) \log \sum_j e^{-\beta((\xi_i - \rho_j)^2 + (\eta_i - \omega_j)^2)}. \tag{5.4}$$

On differentiating w.r.t the components ξ, ρ, η and ω, we obtain

$$\begin{aligned} \left(\frac{\partial F}{\partial \xi}\right)^T &= 2\left[P_1\xi - P_{12}\rho\right], \left(\frac{\partial F}{\partial \eta}\right)^T = 2\left[P_1\eta - P_{12}\omega\right], \\ \left(\frac{\partial F}{\partial \rho}\right)^T &= 2\left[P_2\rho - P_{12}^T\xi\right], \left(\frac{\partial F}{\partial \omega}\right)^T = 2\left[P_2\omega - P_{12}^T\eta\right]. \end{aligned} \tag{5.5}$$

Thus, the partial derivative of F with respect to coordinates $\zeta = [\xi\ \eta\ \rho\ \omega]^T$ is given by:

$$\left(\frac{\partial F}{\partial \zeta}\right)^T = \underbrace{2\begin{pmatrix} I_2 \otimes P_1 & -I_2 \otimes P_{12} \\ -I_2 \otimes P_{12}^T & I_2 \otimes P_2 \end{pmatrix}}_{\triangleq \Gamma} \zeta, \tag{5.6}$$

where $\otimes$ represents matrix Kronecker product, I_2 a 2×2 identity matrix, $P_1 = \text{diag}(p(x_i)) \in \mathbb{R}^{N \times N}$; $P_{12} = [p(x_i, y_j)] \in \mathbb{R}^{N \times M}, p(x_i, y_j) = p(x_i)p(y_j|x_i)$; and $P_2 = \text{diag}\left(p(y_j)\right) \in \mathbb{R}^{M \times M}$, where $p(y_j) = \sum_i p(x_i, y_j)$. Note that each of the above matrices are completely determined by the given weights $p(x_i)$ and the Gibbs distri-

bution $p(y_j|x_i)$. Therefore from (5.1) and (5.6),

$$\frac{dF}{dt} = \left(\frac{\partial F}{\partial \zeta}\right)\dot{\zeta} = 2\zeta^T \Gamma f(t,\zeta). \tag{5.7}$$

Here $\Gamma = \Gamma(\zeta)$ is nonlinear and state dependent matrix but also possesses structure which makes analysis and design easy.

P1: Γ *is a symmetric positive semidefinite matrix for all* ζ.

Proof: From the definition of Γ in equation (5.6) it is clear that it is symmetric. From their definitions, note that the elements $p(x_i)$, $p(x_i, y_j)$ and $p(y_j)$ of Γ have properties that are similar to probability mass functions; i.e. they satisfy $\sum_i p(x_i) = 1$, $\sum_i p(x_i, y_j) = p(y_j)$, $\sum_j p(x_i, y_j) = p(x_i)$ and $\sum_j p(y_j) = 1$. These translate to $P_{12} e_M = [p(x_1), \cdots, p(x_N)]^T \triangleq p_x$, where the notation $e_M = [1\ 1 \cdots\ 1]^T$ represents a vector of length M, $P_{12}^T e_N = [p(y_1), \cdots, p(y_M)]^T \triangleq p_y$, $e_N^T p_x = 1$ and $e_M^T p_y = 1$. This implies that Γ is diagonally dominant since the sum of absolute values of elements of the i^{th} row of P_{12} which is given by $\sum_j |p(x_i, y_j)| = \sum_j p(x_i, y_j)$ adds to $p(x_i)$ which is the diagonal element in the i^{th} row of P_1. Similarly j^{th} row of P_{12}^T adds up to $p(y_j)$ which is the diagonal element in the j^{th} row of P_2. Thus absolute sum of off diagonal terms of each row of Γ is equal to the diagonal term in that row. i.e. $\sum_{j \neq i} |\Gamma_{ij}| = \Gamma_{ii}$. Therefore from Geršgorin theorem (Theorem 6.1.1 in [35]), we have that all the eigenvalues of Γ are located in the discs with centers at the diagonal elements and radii the absolute sum of the off diagonal elements in the corresponding row; i.e.,

$$\begin{aligned} \lambda_i(\Gamma) &\in \cup_{i=1}\{z_i \in \mathbb{C}\ s.t. |z - \Gamma_{ii}| \leq \sum_{j \neq i} |\Gamma_{ij}| = \Gamma_{ii}\}, \\ &\Rightarrow 0 \leq \lambda_i(\Gamma) \leq 2\Gamma_{ii},\ 1 \leq i \leq 2N + 2M. \end{aligned}$$

Therefore Γ is a symmetric matrix with all its eigenvalues nonnegative. This implies Γ is positive semidefinite.

P2: Γ *can be decomposed as* $\alpha(I - W)$ *where* $\alpha > 0$, I *is the identity matrix and* W *is a symmetric doubly stochastic matrix with its spectral radius* $\rho(W) = 1$.

Proof: Note that every element of P_1, P_2 and P_{12} is nonnegative which implies that every off diagonal term in Γ is non positive; i.e.,

$$\Gamma \in Z_{2N+2M} \stackrel{\triangle}{=} \{Q = [q_{ij}] \in \mathbb{R}^{(2N+2M)\times(2N+2M)} \text{ s.t. } q_{ij} \leq 0 \text{ for } i \neq j\}.$$

This implies that there exists a $\bar{W} = [\bar{w}_{ij}] \in \mathbb{R}^{(2N+2M)\times(2N+2M)}$ and $\alpha > 0$ in $\mathbb{R}$ such that Γ is equal to $\alpha I - \bar{W}$ where the spectral radius $\lambda_{max}(\bar{W}) \leq \alpha$ and $\bar{w}_{ij} \geq 0$ for all i, j (This follows from Lemma 2.5.2.1 in [35]). Choose $W = \frac{1}{\alpha}\bar{W}$. This implies $\Gamma = \alpha(I - W)$, $\lambda_{max}(W) \leq 1$ and W is element wise nonnegative. Also W is symmetric since Γ is symmetric. Furthermore

$$\begin{aligned}
\Gamma e_{2N+2M} &= \begin{pmatrix} I_2 \otimes P_1 & -I_2 \otimes P_{12} \\ -I_2 \otimes P_{12}^T & I_2 \otimes P_2 \end{pmatrix} \begin{pmatrix} e_2 \otimes e_N \\ e_2 \otimes e_M \end{pmatrix} \\
&= \begin{pmatrix} e_2 \otimes P_1 e_N - e_2 \otimes P_{12} e_M \\ -e_2 \otimes P_{12}^T e_N + e_2 \otimes P_2 e_M \end{pmatrix} \\
&= \begin{pmatrix} e_2 \otimes (p_x - p_x) \\ e_2 \otimes (p_r - p_r) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\end{aligned}$$

which implies $We_{2N+2M} = e_{2N+2M}$ and since Γ is symmetric, W is symmetric. Therefore $e_{2N+2M}^T W = e_{2N+2M}^T$. This implies W is a symmetric doubly stochastic matrix with $|\lambda_{max}(W)| = 1$.

□

Lack of dynamic control authority at cluster centers. First we transform variables as

$$\bar{y} = y - (I_2 \otimes {Q_2}^T)x \tag{5.8}$$

$$\bar{u} = (I_2 \otimes P_2)(u - (I_2 \otimes Q_2^T)\phi) \tag{5.9}$$

$$\bar{\phi} = (I_2 \otimes P_1)\phi, \tag{5.10}$$

where $Q_1 = [p(y_j|x_i)] \in \mathbb{R}^{N\times M}$ and $Q_2 = [p(x_i|y_j)] \in \mathbb{R}^{N\times M}$, $P_1Q_1 = Q_2P_2 = P_{12}$. In the transformed coordinates, $\frac{dF}{dt}$ is given by

$$\frac{dF}{dt} = \begin{bmatrix} x \\ \bar{y} \end{bmatrix}^T \begin{bmatrix} I_2 \otimes (I - Q_2Q_1^T) & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \bar{\phi} \\ \bar{u} \end{bmatrix} \tag{5.11}$$

$$= x^T R\bar{\phi} + \bar{y}^T\bar{u} = x^T S\phi + \bar{y}^T\bar{u}, \tag{5.12}$$

where $R = (I - Q_2Q_1^T)$ and $S = (I - Q_2Q_1^T)P_1$. Thus, $\frac{dF}{dt}$ becomes independent of control, that is, $\frac{\partial}{\partial u}\left(\frac{dF}{dt}\right) = 0$ when $\bar{y}(t_c) = 0$. From (5.8), we note that

$$\bar{y}(t_c) = 0 \Rightarrow y(t_c) = (I_2 \otimes {Q_2}^T)x \tag{5.13}$$

$$\Rightarrow y_j(t_c) = \sum_{i=1}^{N} p(x_i(t_c)|y_j(t_c))x_i(t_c), \text{ for } 1 \le j \le M, \tag{5.14}$$

which is identical to the centroid condition that we had previously. □

Feasibility of tracking cluster centers. For the case when $\bar{y} \neq 0$, we use Sontag's formula [73, 74] to prescribe a velocity field that ensures that $\frac{dF}{dt} \le 0$. The transformed control value ($\bar{u}$) is given by

$$\bar{u} = -\left(K_o + \frac{x^T S\phi + \sqrt{(x^T S\phi)^2 + (\bar{y}^T\bar{y})^2}}{\bar{y}^T\bar{y}}\right)\bar{y} \tag{5.15}$$

$$\Rightarrow u = (I_2 \otimes Q_2^T)\phi - (I_2 \otimes P_2)^{-1} \left(K_o + \frac{x^T S\phi + \sqrt{(x^T S\phi)^2 + (\bar{y}^T \bar{y})^2}}{\bar{y}^T \bar{y}} \right) \bar{y}, \quad (5.16)$$

where K_o can be chosen arbitrarily such that $K_o \in \mathbb{R}, K_o > 0$. In order to prove that this choice of u is bounded, we first note from (2.33) that y_j is a convex combination of $x_i, 1 \leq i \leq N$, since $0 < p(x_i|y) \leq 1 \, \forall i$ and $\sum_{i=1}^{N} p(x_i|y) = 1$. Thus when the domain area is bounded (i.e. $\|x\| \leq w_1$), we infer that $\|y\| = \left\| \sum_{i=1}^{N} p(x_i|y) x_i \right\| \leq w_1$. Next, note that

$$\|\bar{u}\| \leq \left(K_o + \frac{2\|x\|\|S\Phi\| + \bar{y}^T \bar{y}}{\bar{y}^T \bar{y}} \right) \|\bar{y}\|. \quad (5.17)$$

We assume that y is at least a distance δ from the cluster centers, that is, $0 < \delta < \|\bar{y}\| = \|y - y_c\| < 2w_1$. Since $S = (I - Q_2 Q_1^T) P_1$, where Q_2 and Q_1 are stochastic matrices and $P_1 = diag(p(x))$, with $\lambda_{max}(P_1) \leq K_2 \leq 1$, we get $\|S\| \leq K_2$. Hence,

$$\|\bar{u}\| \leq 2 \left(1 + K_o + \frac{K_2}{\delta} \|\Phi\| \right) w_1. \quad (5.18)$$

This implies that the control $\bar{u}$ is bounded subject to the condition that the velocity field $\|\Phi\|$ is bounded and the resource locations are not at the cluster centers. Thus, under a bounded velocity field Φ and $\bar{y}$ bounded away from zero, there exists a bounded control value given by (5.16) which ensures that $\frac{dF}{dt} \leq 0$.

For a given value of β, if y is not at the cluster centers at time t_1, and reaches the respective cluster centers at a later time t_c, then $\bar{y}(t_1) \neq 0$ and $\bar{y}(t_c) = 0$, i.e. $y_j(t_c) = \sum_{i=1}^{N} p(x_i(t_c)|y_j(t_c)) x_i(t_c)$, for $1 \leq j \leq M$. On substituting the control value $\bar{u}$ from (5.15) in (5.11), we get

$$\dot{F} = -\bar{y}^T K_o \bar{y} - \sqrt{(x^T S\phi)^2 + (\bar{y}^T \bar{y})^2} \;\Rightarrow\; \dot{F} \leq -K_o \bar{y}^T \bar{y} = -K_o \|y - y_c\|^2. \quad (5.19)$$

For $\|y - y_c\| > \delta$ (i.e. when the current location y is at least a distance δ away from the cluster centers), we get that $\dot{F} \leq -K_o \delta^2$. Now consider another function $G(t)$

such that $\dot{G}(t) = -K_o\delta^2$. Since $\frac{d}{dt}(F(t) - G(t)) < 0$ in the interval (t_1, t_2), it implies that $(F(t_1) - G(t_1)) > (F(t_2) - G(t_2))$. Thus for a time $t_2 > t_1$ $(\Delta t = t_2 - t_1)$,

$$F(t_2) - F(t_1) < G(t_2) - G(t_1) = -K_o\delta^2\Delta t. \tag{5.20}$$

This implies that $\Delta t \leq \frac{F(t_1)}{K_o\delta^2}$. Since the choice of K_o is arbitrary, we choose $K_o = \alpha e^{\frac{\gamma}{\delta^2}}$, where $\alpha, \gamma > 0$. This implies that $\Delta t \leq \frac{F(t_1)}{\alpha e^{\frac{\gamma}{\delta^2}}\delta^2}$. Note that the quantity $\frac{1}{\alpha e^{\frac{\gamma}{\delta^2}}\delta^2}$ is upper bounded by $\frac{1}{\alpha e\gamma}$, thereby implying that $\Delta t \leq \frac{F(t_1)}{\alpha e\gamma}$. The bound on $F(t_1)$ is dependent on the size of the underlying space Ω.

$$F(t_1) \leq \frac{N}{\beta}p^x_{max}\log M + 4Np^x_{max}w_1^2, \text{ where } p^x_{max} = \max(p(x_i)). \tag{5.21}$$

□

In spite of the complex nonlinear (non-quadratic) structure of F, its derivative exhibits an algebraic structure that is similar to a derivative of a quadratic function, which makes it available for analysis and design. As shown in the above theorem, this derivative is characterized by the matrix Γ, which has a specific structure that is completely determined by the weights $p(x_i)$ and $p(y_j|x_i)$. These properties prove very important for analysis as well as design of control $u(t)$ to enable tracking of cluster-centers. The assertions 3 and 4 of the above theorem enable the incorporation of the objectives of tracking cluster-centers and monitoring cluster splits and resolution one after the other in the dynamic setting. The assertions 3 and 4 state that one can design a bounded control $u(t)$, that makes $\frac{dF(t)}{dt} < 0$ and ensures that the resources move to a δ-neighborhood of the respective cluster-centers. The quantity $\frac{dF(t)}{dt}$ becomes independent of the control term only when the resources reach the respective cluster-centers. The bound on the time interval to reach the δ-neighborhood of the respective cluster-centers is stated in terms of the quantity δ. The time required to reach

$\delta-$neighborhood is of the order of $O(\frac{1}{\delta^2})$, and the control $u(t)$ is of the order of $O(K_o) + O(\frac{1}{\delta})$. This bound is conservative as it is obtained here independent of the distribution of the mobile sites. The constant K_o can be chosen to accelerate the convergence to the cluster centers, but at the added expense of using higher control effort. The tuning parameter α characterizes this tradeoff. In typical simulations, the convergence rates to cluster centers does not pose a hurdle to the implementation of the algorithm.

Tracking cluster centers

Since tracking cluster centers has higher precedence, the control $u(t)$ is designed to reach the cluster centers whenever the resource locations are not at the cluster-centers. For instance a control law

$$u(t) = (I_2 \otimes Q_2^T)\phi - (I_2 \otimes P_2)^{-1} \left(\frac{x^T S\phi + \sqrt{(x^T S\phi)^2 + (\bar{y}^T \bar{y})^2}}{\bar{y}^T \bar{y}} \right) \bar{y}, \qquad (5.22)$$

drives resource locations y_j to cluster centers of sites x_i. There is a lot of choice in the design of this control where the free energy F is treated as a control Lyapunov function and u is designed to make $\frac{dF}{dt} \leq 0$. Other computationally efficient control laws can be devised by exploiting the properties of Γ [58]. The control $u(t)$ prescribed above is bounded for a bounded velocity field ϕ and drives the resources to a δ-neighborhood of the respective cluster centers in a finite time interval.

Monitoring cluster centers

Once the condition for the lack of dynamic control is reached, the cluster-monitoring is done by taking any of the following decisions

1. Decision to split

A decision to split can be taken when the parameter β satisfies the condition $\beta^{-1} = 2\lambda_{\max}(C_{x(t_c)|y_j(t_c)})$ as discussed in the section above. If time were frozen at $t = t_c$, this splitting condition implies that y is at a local maxima (or inflection point) of F, that is where the Hessian of F is no longer positive definite. The set of resource locations after splitting is determined by the same procedure as in the static case described in Section 2.6.3. These new locations are given by $\mathbf{y}_{new} = q_{\text{split}}(\mathbf{y})$ as defined in (2.45). This does not pose problems for the dynamic implementation of the algorithm as the computation time to determine $q_{\text{split}}(\mathbf{y})$ is small enough when compared to computation of $u(t)$ (more details on this are given in the Section 2.6.3).

2. Decision to track cluster center with or without increasing resolution

At time t if $y_j, 1 \leq j \leq M$ are at cluster-centers and the parameter β does not satisfy the splitting condition, then one can continue to track cluster center by assigning $y_j(t) = \sum_i p(x_i(t)|y_j(t))x_i(t)$ or change resolution by increasing cooling rates with respect to the site-dynamics, which ensue splits and thus result in finer clusters, and therefore higher resolution.

5.2.3 Flexibility of the Framework

The distinct procedures and individual control on these decisions provides a lot of flexibility to the framework. This flexibility is essential since as discussed in Section 5.1, the task of efficiently clustering moving objects occurs in a variety of problems from different application areas. Each of these problems have slightly different constraints and requirements, thereby calling for modifications to the basic framework. This is facilitated by a user-based layer in our clustering framework. This enables the clustering framework to address the issues such as stopping criterion, resolution

of the clustering solution, number of clusters. For instance, a user can decide in real time to investigate clusters with more resolution. These can be effected by changing temperature values after tracking the clusters. This framework can easily accommodate cases where different clusters can be monitored with different resolution. This can be obtained by allowing multiple cooling laws one associated with each resource location. This generalization is discussed further in Section 5.3. The flowchart in Figure 5.3 gives a detailed description of this framework, the salient mechanisms of which are discussed in this section.

5.3 Simulation Results and Discussion

In this section, we present simulation results for a variety of datasets with different underlying dynamics. The test cases are specifically designed to highlight the key features of our dynamic clustering framework, and analyze its performance.

5.3.1 Implementation of the Basic Algorithm

We summarize in Figure 5.3 the steps that the algorithm traces in order to solve the coverage problem. Note that this implementation is for the basic version of the algorithm, without any external user-based directives. All the simulations were carried out in MATLAB. For simulations, the dynamics in (5.1) were run by discretizing them using fourth-order Runge-Kutta method (RK-4) [10]. On using the RK-4 method, the error per step is of the order of $(\Delta t)^5$, while the total accumulated error is of the order of $(\Delta t)^4$. The time steps were chosen so that the solution converges. It should be noted that the time for computing $q_{split}(\mathbf{y})$ is comparable to that of the computation of $u(t)$.

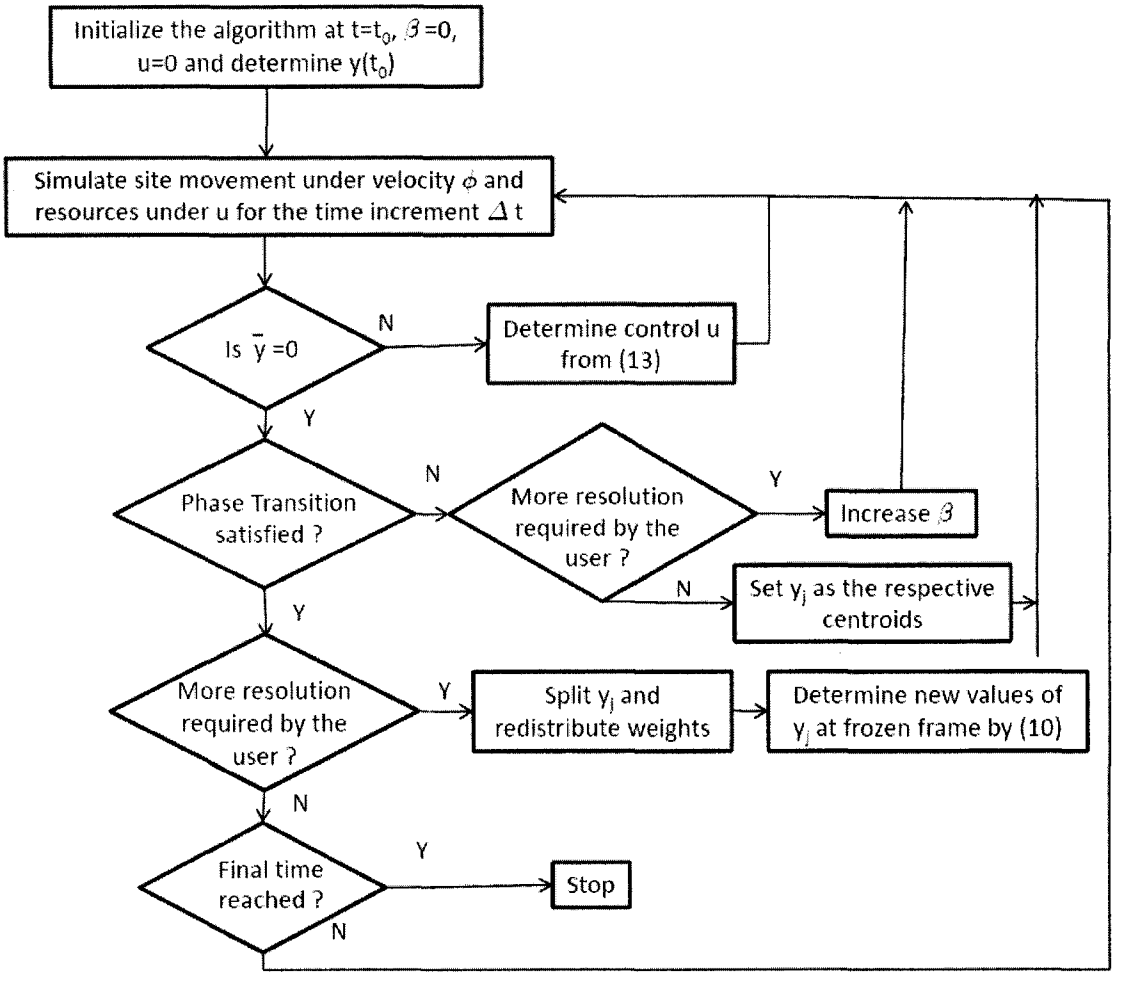


Figure 5.3: Flowchart showing the implementation of the proposed algorithm.

5.3.2 Hierarchical Natural Cluster Identification and Tracking

For the purpose of this simulation, we chose a scenario with 160 mobile sites. The velocity field $\phi(t, x)$ is chosen for each of the mobile sites, such that natural clusters emerge within the time horizon. The time-horizon for this case was 8 seconds. At $t = 0$, the dataset has one natural cluster, with majority of the sites located in the central region (Figure 5.4). The yellow squares show the initial location of the

moving objects (sites). The algorithm begins by placing one resource at the centroid

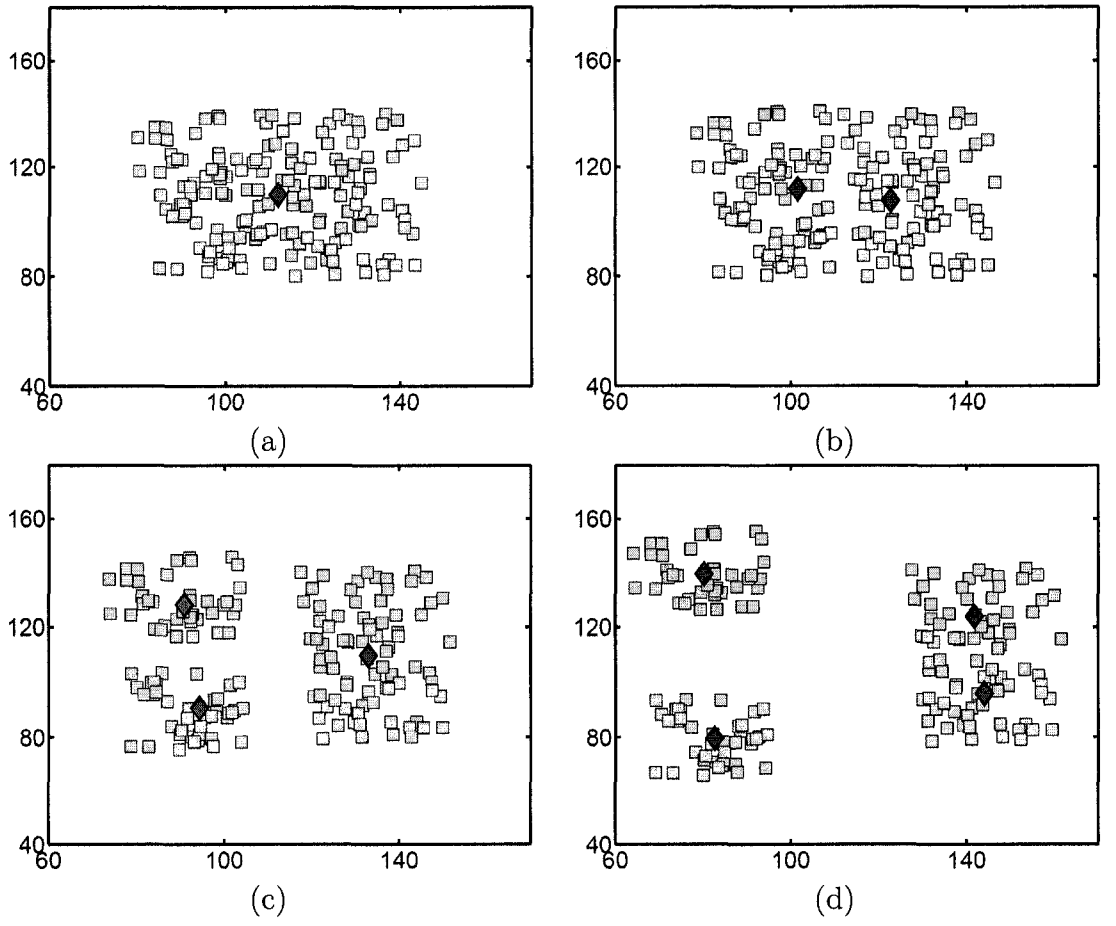


Figure 5.4: Simulation results for dataset 6 showing cluster identification and tracking during the time horizon. Snapshots (a), (b), (c) and (d) show the locations of mobile sites $x_i, 1 \leq i \leq 160$ (shown by yellow squares) and resources $y_j, 1 \leq j \leq M$ (shown by red diamond) at various phase transition instances. All the sites are initially concentrated at the center of the domain area, and then slowly start drifting away from one another. Four natural cluster emerge at the end of the time horizon, as seen in (d). The algorithm progressively identifies and tracks the clusters in a hierarchical manner.

of the site positions (at $t = 0$) (a red diamond denotes this location in Figure 5.4 (a)). As the dynamics evolve, the site locations move according to the equation ($\dot{x} = \phi(x, t)$). The algorithm progressively updates the association probabilities and resource locations, and determines the control value (i.e. the resource velocities from (5.16)) in order to track cluster centers. Figures 5.4 (b), (c) and (d) show the locations

of the sites and the resources in the interim instants. The number of resource locations increases progressively due to successive phase transitions. As seen in the figure, the resource locations maintain coverage by identifying and tracking the natural clusters in the underlying data. At the end of the time-horizon, all the natural clusters are identified. The algorithm successfully avoids the several local minima and provides progressively better coverage. Figure 5.5 (a) shows a plot of the coverage function F with respect to time. Note that at each phase transition, there is a sharp decline in F. This is due to the fact that at every phase transition, a resource location at a new local minimum is added, which results in lower free energy, thereby providing better coverage. As expected, the proposed algorithm is considerably faster than the frame-by-frame approach. The computation times for five datasets are shown in Figure 5.5 (b). Depending on the nature of the underlying data, the proposed algorithm is five to seven times faster than the frame-by-frame method. A comparison of the instantaneous distortion value $\sum_i p(x_i) \min_j d(x_i, y_j)$ obtained by the two algorithms is presented in Figure 5.5 (c). The proposed algorithm identifies and tracks the clusters hierarchically such that the distortion steadily decreases, with sharp decreases at splits. As seen in the figure, the frame-by-frame method for the same number of resources achieves slightly lower distortion, but with frequent spikes due to the spatio-temporal effects. Moreover, the frame-by-frame method uses 5 times the computation time required for the proposed algorithm. Figure 5.5 (d) shows a comparison of the final distortion achieved by the proposed algorithm and the frame-by-frame method (for the same time step and same number of resources). As is seen in the figure, the proposed algorithm achieves a distortion ranging from within 0.5% to 4.3% of the frame-by-frame approach, depending on the dataset. This is achieved using a considerably smaller computation time, as is shown in (b).

In the absence of spatio-temporal smoothening, the clustering solutions obtained at two successive time instants might be considerably disparate from one another,

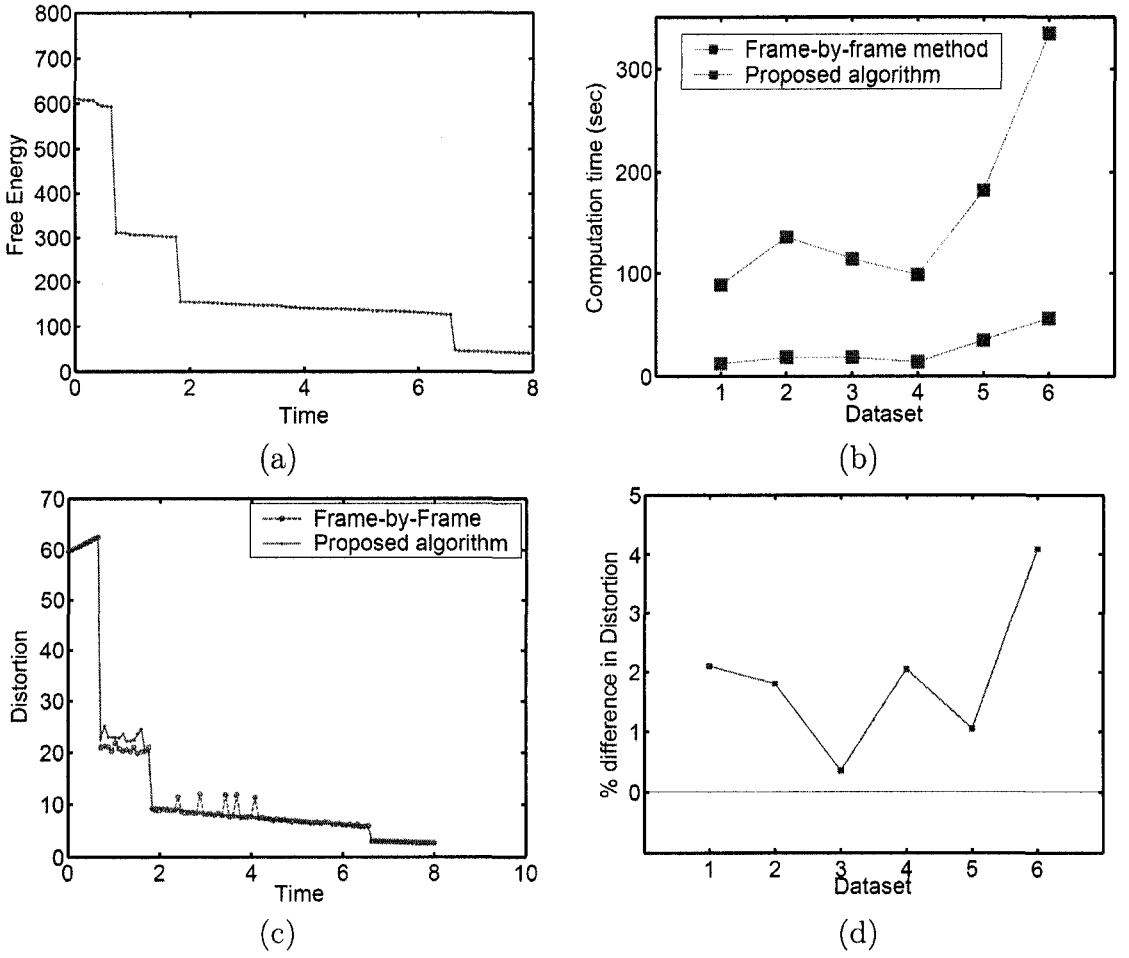


Figure 5.5: (a) Free Energy F with respect to time. The control value u ensures that $\frac{dF}{dt} \leq 0$. The sudden decrease in F is due to the phase transition process. Progressively decreasing Free Energy results in better and better coverage and tracking through the time horizon. (b) Comparison of the computation times for frame-by-frame and proposed algorithm. (c) Comparison of the distortion achieved by the frame-by-frame method and the proposed framework. time. (d) Percentage error in final distortion achieved by the proposed algorithm with respect to the frame-by-frame method.

even though the number of natural clusters in the dataset remains the same. Figure 5.6 shows the clustering solution obtained by the frame-by-frame approach at two successive time instants. Three clusters are identified by the algorithm at each instant, but at considerably different spatial locations. This occurs because no information from the previous clustering solution is used for determining the solution at the next time instant. On the other hand, the proposed framework overcomes this problem by

using a smooth control value everywhere except during cluster splits via phase transition. In order to speed up the frame-by-frame approach, we increase the time-step

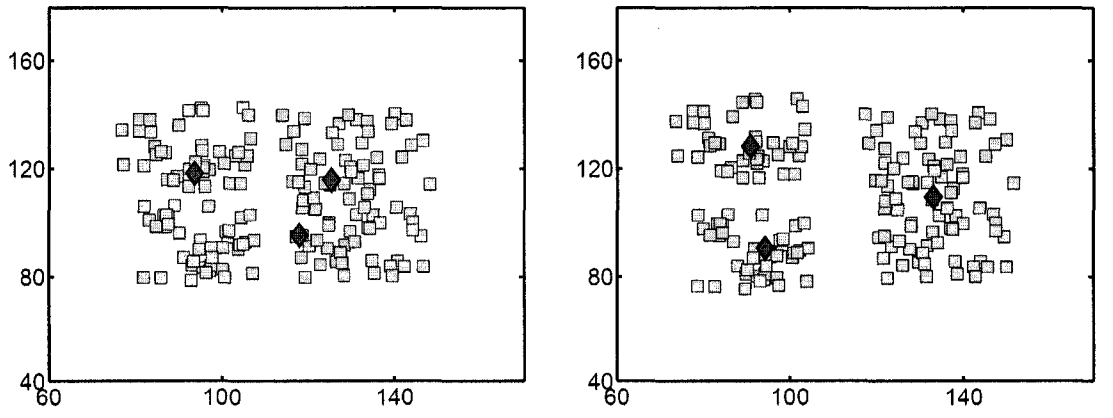


Figure 5.6: Clustering results from two successive frames using the frame-by-frame approach. In both instances, three resource locations were identified by the algorithm, but at considerably different positions. Such a solution violates the spatio-temporal requirement of the dynamic clustering algorithm. This happens because none of the clustering information from previous frame is employed in order to determine the solution at future frames.

between successive frames. During such an implementation, the resource locations at obtained at the previous time instant are used in the interim between successive frames. This results in a clustering solution that deteriorates in the interim because of the lack of new information. Figures 5.7 (a) and (b) show the distortion obtained by a three fold and six fold increase in the time steps. As is seen in the figures, the distortion obtained from the frame-by-frame approach deteriorates considerably with respect to the proposed algorithm. Note that even for a six fold increase in the time step, the computation time for the frame-by-frame is slightly higher than that of the proposed algorithm. In most of the applications, one would not want such a phenomenon to occur. Simulation results for two other datasets are depicted in figures 5.8 and 5.9. The proposed algorithm has also been successfully implemented on considerably larger datasets (with 10,000 mobile sites). Figure 5.10 shows the simulation results on a dataset containing 8000 mobile sites. For this illustrative example, the

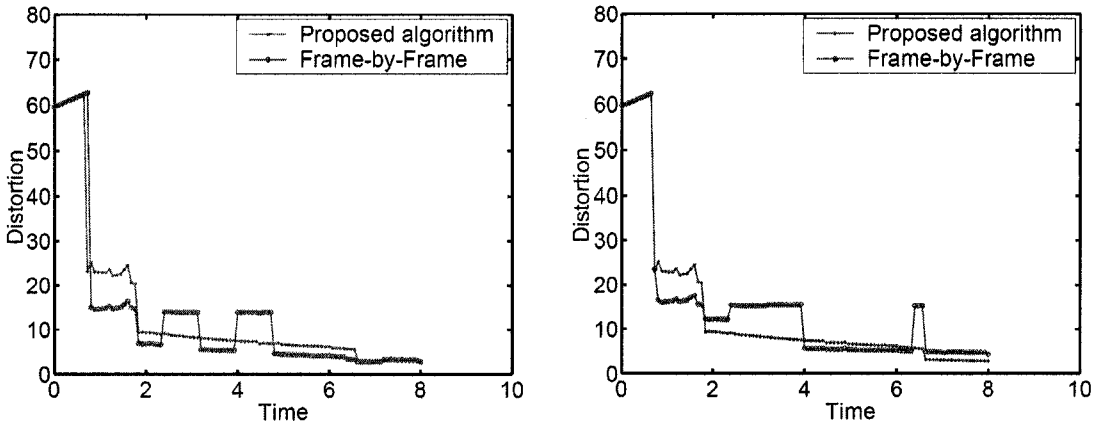


Figure 5.7: Comparison of distortion obtained by the proposed algorithm and the frame-by-frame approach under different time steps (a) Proposed algorithm : Δt = 0.08 sec, frame-by-frame approach: Δt = 0.24 sec. (b) Proposed algorithm : Δt = 0.08 sec, frame-by-frame approach: Δt = 0.48 sec.

locations x_i and the velocities ϕ_i were chosen randomly. For the purpose of visualization and the clarity of presentation, most of the simulation results presented in this section are on datasets with small number of mobile resources. Notwithstanding, all the essential features of the proposed algorithm are preserved irrespective of the size of the underlying dataset.

5.3.3 User-Based Directives

The phase transition phenomenon is caused because of the interplay between inherent dynamics of the moving objets and the decrease in the temperature value. For clustering static datasets, the divisive hierarchical algorithm gets progressively better coverage with a decrease in temperature. This decrease in temperature causes the phase transition phenomenon, leading to an increase in the resolution of the clustering solution. However in the dynamic framework, temperature is not the sole driver for initiating phase transition. The inherent dynamics of the moving sites can change the number of clusters and their locations, indirectly inducing phase transition. In

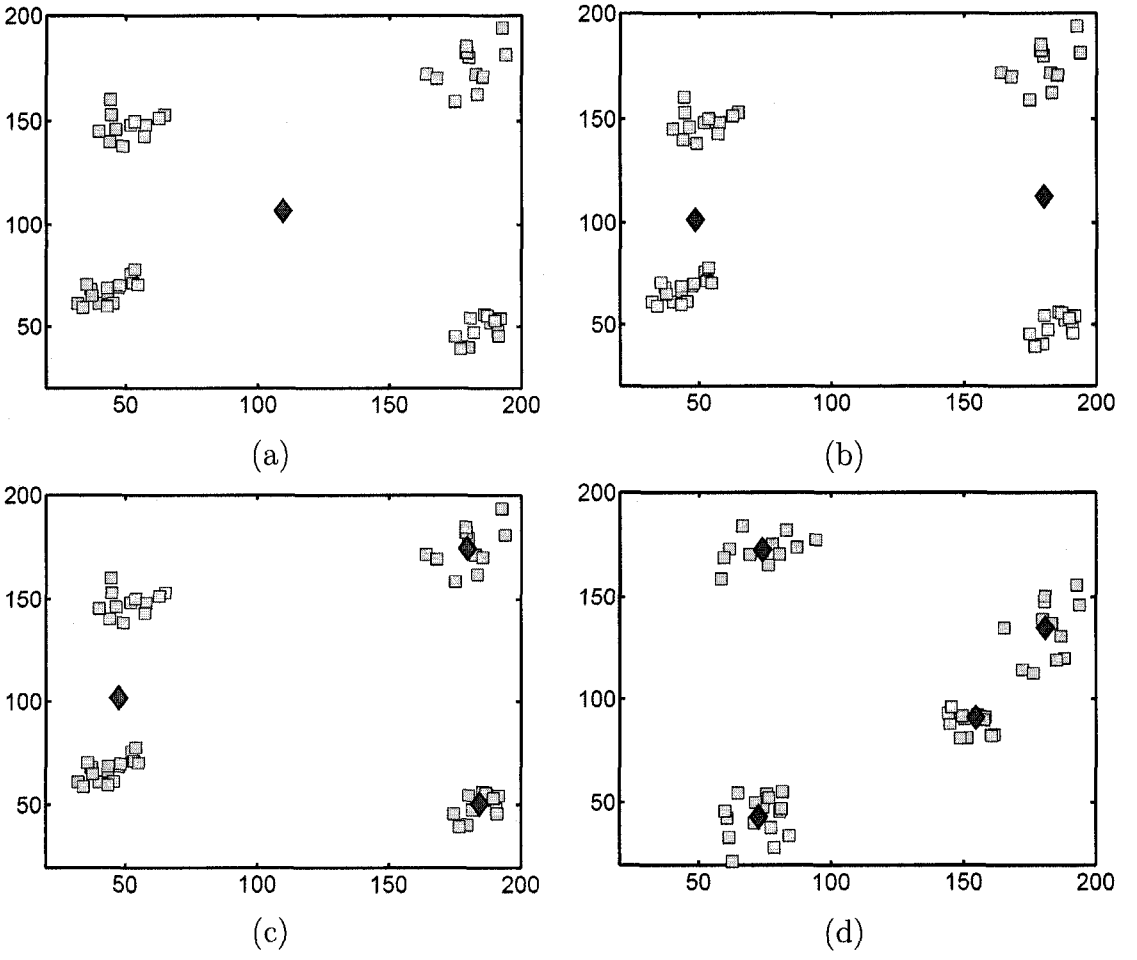


Figure 5.8: Simulation results for dataset 2 showing cluster identification and tracking during the time horizon. Snapshots (a), (b), (c) and (d) show the locations of mobile sites $x_i, 1 \leq i \leq N$ (shown by yellow squares) and resources $y_j, 1 \leq j \leq M$ (shown by red diamond) at various phase transition instances.

addition to this, the nature of the application of the algorithm may call for a tradeoff between cluster tracking and cluster identification. Once all the natural clusters are identified and are being tracked, the user may want a higher resolution clustering and simultaneous tracking. Such user-based directives are incorporated in our dynamic clustering framework, thereby providing a flexibility in its modes of operation. Instead of decreasing the temperature at every instance of $\bar{y} = 0$, the algorithm is modified by adding a user-based directives layer. Depending on the context of the application, the algorithm provides cluster identification and tracking at a particular

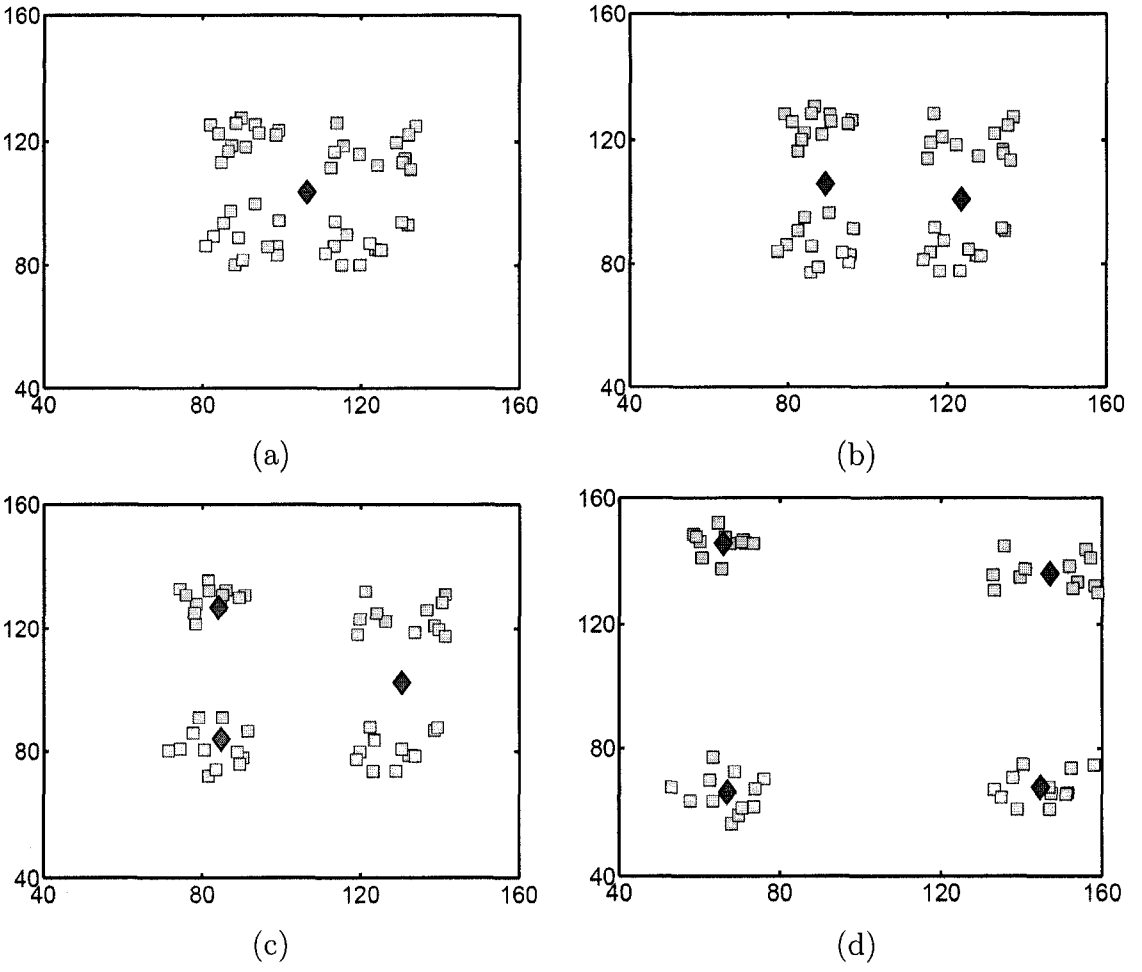


Figure 5.9: Simulation results for dataset 1 showing cluster identification and tracking during the time horizon. Snapshots (a), (b), (c) and (d) show the locations of mobile sites $x_i, 1 \leq i \leq N$ (shown by yellow squares) and resources $y_j, 1 \leq j \leq M$ (shown by red diamond) at various phase transition instances. All the sites are initially concentrated at the center of the domain area, and then slowly start drifting away from one another. Four natural cluster emerge at the end of the time horizon, as seen in (d). The algorithm progressively identifies and tracks the clusters in a hierarchical manner.

resolution. Figure 5.11 shows the simulation results for the same dataset as in Figure 5.9, but under an additional user-based directive on the final clustering resolution. Both the algorithms were executed for the same time-horizon, but resulted in different clustering resolutions.

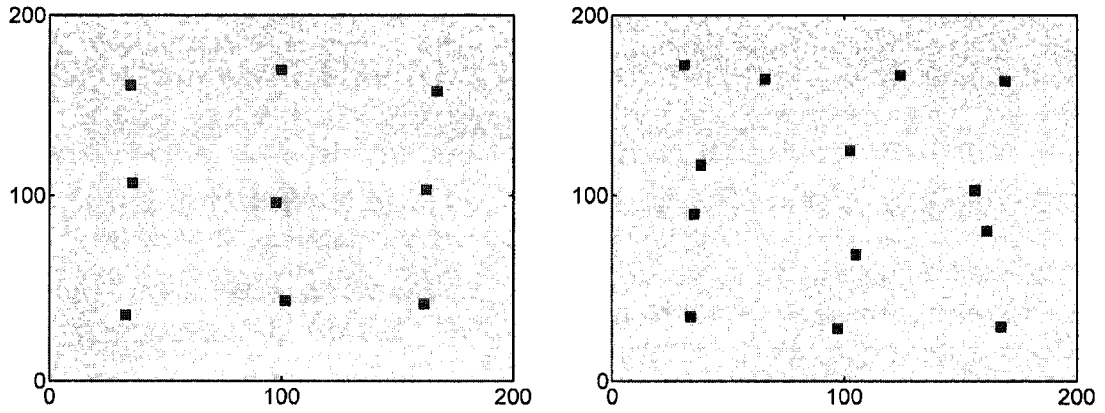


Figure 5.10: Simulation results for a dataset containing 8000 sites with random velocities. Snapshots (a) and (b) show the locations of the mobile sites $x_i, 1 \leq i \leq N$ (shown by yellow dots) and resources $y_j, 1 \leq j \leq M$ (shown by red diamonds) at different instances.

5.4 Extensions

In this section we present some ongoing work that is aimed at analysis as well as further extensions of the framework presented.

5.4.1 Robustness to Modeling Uncertainties

Coverage under transmission errors

In the framework presented above, we have considered ideal transmission whereby the exact locations of the sites x_i are relayed to the resources. The effect of noisy channel transmissions can be accommodated in the following way. To specify the noisy transmission via a communication channel, we define a transition probability $p(z|x)$, the probability that a corrupted location z is transmitted for a site at a location x. This is incorporated into the framework by replacing the instantaneous

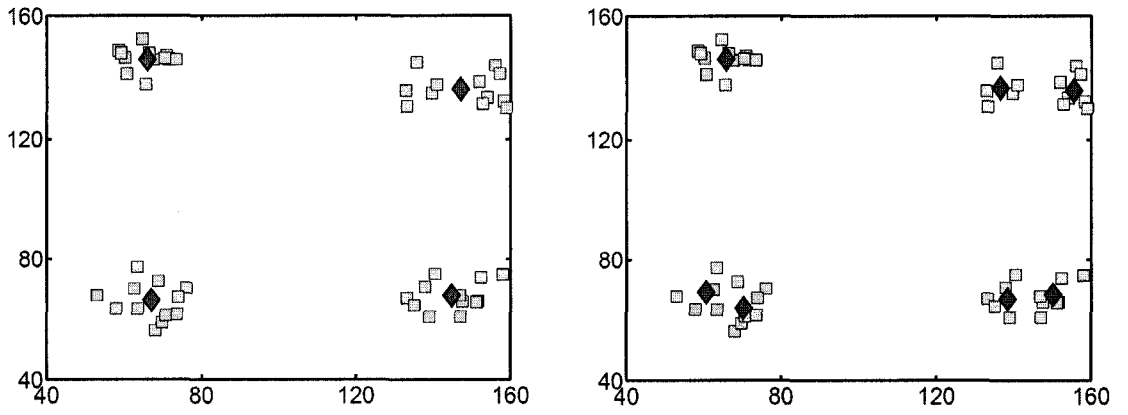


Figure 5.11: Comparison of results obtained in two different cases. (a) Without user-based resolution directives. (b) With user-based resolution directives. In the case (b), the user specified a higher final resolution and thus the algorithm identified and tracked finer clusters (and subclusters) during the same time horizon as that in (a).

distortion term (5.2) by

$$\hat{D}(x,y) = \sum_x \sum_y p(x)p(y|x)d'(x,y), \tag{5.23}$$

where $d'(x,y)$ represents a modified distance that takes into account the characteristics of the noisy channel as

$$d'(x,y) = \sum_z p(x|z)d(x,y), \text{ where } p(x|z) = \frac{p(z|x)p(x)}{\sum_x p(z|x)p(x)}. \tag{5.24}$$

That is, the average weighted distance replaces the notion of distance in the distortion term. Now using the entropy as before, $H = -\sum_x \sum_y p(x)p(y|x)\log p(y|x)$, the association probabilities take the form

$$p(y|x) = \frac{exp(-\beta(\sum_z p(x|z)d(x,y)))}{Z_x}. \tag{5.25}$$

The instantaneous free energy can now be rewritten as

$$\hat{F} = -\frac{1}{\beta}\sum_i p(x_i)\log\sum_j exp\left[-\beta(\sum_k p(x_i|z_k)d(x_i,y_j))\right]. \tag{5.26}$$

This term can now be used as a metric for instantaneous coverage under transmission errors. An analysis similar to that of the proposed algorithm in Section 2 can be employed to obtain dynamic clustering.

Remark: In the framework presented, the velocity fields ϕ of sites are assumed to be known exactly. In case of noisy dynamics, where ϕ is given by perturbations $n(t)$ about a nominal function $\bar{\phi}$, that is $\phi = \bar{\phi}(x,y,t)+n(t)$, and where the measurements of site and resource locations are noisy, control designs based on estimated values of (x,y) give satisfactory performance. This is primarily due to the fact that the shapes of the Gibb's distribution functions are insensitive to slight perturbations in x and y. Another closely related problem is real time estimation of ϕ. Since the sites dynamics are completely realized by *velocity* fields, these fields, when not known a priori, can be estimated and $u(t)$ designed based on the estimated field. The inherent robustness in the algorithm due to the properties of the Gibb's distribution, again yield satisfactory performance. The analysis and quantitative bounds of this perturbation analysis is part of the ongoing work.

5.4.2 Extending the Class of Coverage Problems

Constraints on resources

It should be noted that the resources in the framework presented are identical. However, depending on the problem, the resources can be non-identical, where different constraints apply to different resources, for instance, vehicles can be of different sizes with different coverage capacity. The resources can be made non-identical by introducing weights λ_j to each resource location y_j. This interpretation yields a modified

free energy function given by

$$F = -\frac{1}{\beta}\sum_i p(x_i)\log\sum_j \lambda_j e^{-\beta d(x_i - y_j)}. \tag{5.27}$$

The constraints on resources are implemented by specifying constraints on λ_j. For instance setting them to constants W_j, that is, $\lambda_j = W_j$, yields resource locations that have weights in the same ratios as $\{W_j\}$. More details on this formulation for static problems in the context of facility location, drug discovery, vector quantization are respectively presented in [61, 66, 57]. The same formulation easily extends to the dynamic case where static constraints can be easily incorporated by just redefining the free energy as in (5.27) and minimizing the appropriate Lagrangian with respect to $\{y_j\}$ as well as $\lambda_j\}$. For instance the for the constraints in the above setting the unconstrained Lagrangian is given by

$$L = F + \sum_j \mu_j(\lambda_j - W_j), \tag{5.28}$$

where μ_j are Lagrange parameters and F is given by (5.27). The resource locations y_j, weights λ_j are found by setting $\frac{\partial L}{\partial(\{y_j,\lambda_j,\mu_j\})} = 0$. For dynamic constraints, same process can be applied as described in this chapter except that the free energy $F(t)$ is replaced by the Lagrangian $L(t)$.

Different distance metrics

In this chapter we have assumed square of Euclidean distance $d(x_i, y_j) = \|x_i - y_j\|^2$ as the distance metric. This choice of metric results in cluster centers being weighted centroids which is appropriate in many problems. However, the MEP-based algorithm does not require this specific metric, it is applicable to any other metric. For instance, 'coverage while maintaining distance' is achieved by choosing a metric that penalizes

the resource location when it is too near or too far (see Figure 5.12 for details). From Figure 5.12 (b), we observe that the algorithm identifies six clusters, under

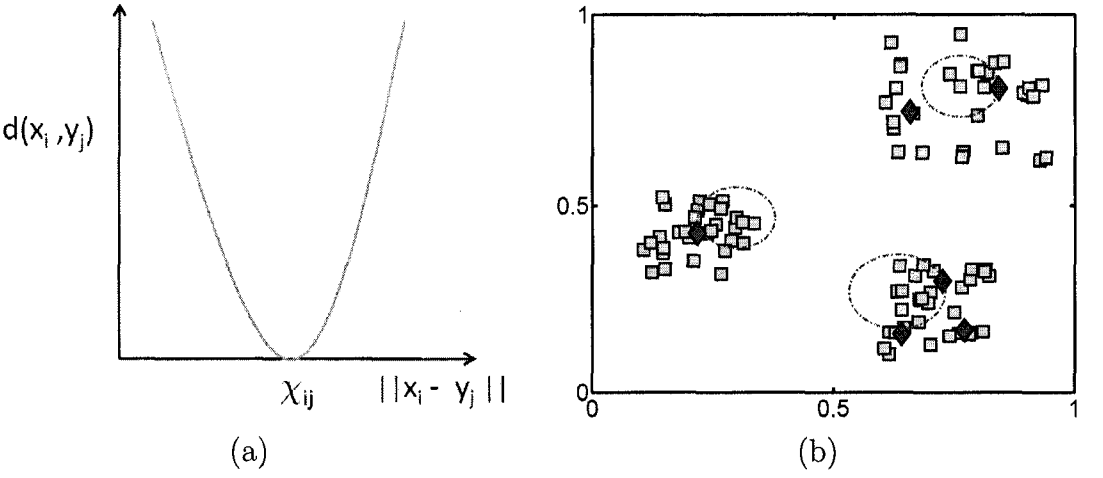


Figure 5.12: (a) Modified distance metric $d(x_i, y_j) = (\|x_i - y_j\| - \chi_{ij})^2$ for obtaining 'coverage while maintaining distance'. (b) Simulation results with modified distance metric for exclusion constraints. The locations $x_i, 1 \leq i \leq 90$ of sites (circles) and $y_j, 1 \leq j \leq 6$ of resources (crosses) in the 2-d descriptor space. Blue circles represent region to be excluded.

the constraint that none of the cluster centers are located in the undesirable region represented by the dotted circles.

Inertial forces into vehicle dynamics

In this chapter, we have presented tracking of cluster centers when velocity fields (one state per direction for each vehicle) of sites are given. The same procedure is applicable even when multiple-state differential equations are given. For instance, in context of vehicle systems, the dynamics of these autonomous mobile agents is usually controlled by thrust actuators, the control term being the amount of thrust in each direction. The corresponding model for a domain with N mobile agents becomes $x_i = [\xi_i \ \eta_i]^T \in \mathbb{R}^2$ and M resource locations $y_j = [\rho_j \ \omega_j]^T \in \mathbb{R}^2$ as before, whose

dynamics are now given by

$$\ddot{x}(t) = \Upsilon(x(t), \dot{x}(t), y(t), \dot{y}(t), t), \;\; x(0) = x_0, \dot{x}(0) = xd_0 \tag{5.29}$$

$$\ddot{y}(t) = u(t), \;\; y(0) = y_0, \;\; \dot{y}(0) = yd_0 \tag{5.30}$$

Our task is to determine the accelerations ($\ddot{y}$) and velocities ($\dot{y}$) for the M mobile resources and their initial locations, such that adequate coverage is maintained. However by writing the above second order differential equations for each vehicle into first order vector differential equations, these equations come under the same form as (5.1), albeit with extra algebraic structure. Under this scenario, the Euclidean distance metric turns out to be of the form

$$d(x_i, y_j) = (x_i - y_j)^2 + \theta(\dot{x}_i - \dot{y}_j)^2,$$

where θ is a constant. The choice of high values of θ gives more relative importance to velocities and hence yields cluster centers that take into account instantaneous headings. Note that the distance function is analogous to sum of kinetic energy $(\dot{x}_i - \dot{y}_j)^2$ and potential energy $(x_i - y_j)^2$ in a mechanical system where site x_i and y_j are connected by springs. The distortion term then is analogous to total energy in the system and θ is reciprocal of natural frequency.

5.4.3 Numerical Issues in the Algorithm

Distributedness and Scalability

The framework presented in this chapter aims at avoiding local minima. As a consequence the computations are global in the sense that computation of each resource location requires values of all the site-locations. However, the contribution of 'far off' site-locations becomes progressively lower as the parameter β is increased. In fact,

the partitions are hard as $\beta \rightarrow \infty$ and consequently the contribution of site-locations that are not 'nearest neighbors' is zero. In this sense, the computation of resource locations change from being *truly global* to *truly local* as β is increased from zero to infinity. In Chapter 3 and 4, this feature is exploited to evolve a scalable algorithm that is a close approximation of the MEP-based algorithm presented here for the static setting. In dynamic setting, the clusters can interact even when β values are high, due to site dynamics. However we propose to monitor values of the norm of the Hessian $\frac{\partial^2 F}{\partial y^2}$, and estimate the *effective* radius around each cluster center y_j, beyond which, the site-locations can be ignored.

Approximation of Gibbs' functions

One of the costly steps in the algorithm presented is computing the exponentials in the Gibb's distribution. One of the ways to address this issue is to substitute the Gibbs distribution in (2.31) by a low complexity distribution that approximates it closely. As shown in [15], these low complexity distributions provide fast computation with performance nearly equivalent to the Gibbs distributions. One such distribution is the variable-width triangle distribution, given by

$$\hat{p}(y_j|x_i) = \frac{R_x - d(x_i, y_j)}{NR_x - \sum_{k=0}^{N-1} d(x_i, y_k)}, \tag{5.31}$$

where R_x is determined for a given value of β by minimizing the L_1 distance between the approximate distribution ($\hat{p}(y_j|x_i)$) and the Gibbs distribution ($p_G(y_j|x_i)$). The computational cost incurred in computing one instance of the low complexity distribution using (5.31) requires $M+7$ flops, where M is the number of resource locations in the span R_x. This total number of flops is determined by counting one flop for addition, subtraction, and multiplication operations and four flops for the division operation. On the other hand, computing one instance of Gibbs distribution in (2.31)

requires $10M + 4$ flops. Figure 5.13 (a) shows a comparison between the exponential function $e^{-\beta z^2}$ and the low complexity triangle function $(R_x - z)/R_x$. The associated probability distributions generated from these functions are compared in Figure 5.13 (b). The parameter R_x is computed by minimizing the Kullback-Leibler divergence [41] (also referred to as relative entropy) between the two distributions, which is also an upper bound the L_1 distance between them, that is

$$D_{KL}(\hat{p}(y_j|x_i) \parallel p_G(y_j|x_i)) \geq \frac{1}{2\ln 2} \parallel \hat{p}(y_j|x_i) - p_G(y_j|x_i) \parallel_1^2, \tag{5.32}$$

where equality is achieved when $\hat{p}(y_j|x_i) = p_G(y_j|x_i)$. This yields the value of $R_x = \sqrt{\frac{9(\pi-2)}{\pi\beta}}$, thereby prescribing a schedule for R_x in terms of the β schedule (see [15] for details). Simulation results and computation times using this approximate Gibbs distribution are presented in Figure 5.14 As seen in Figure 5.14 (c), the use of such

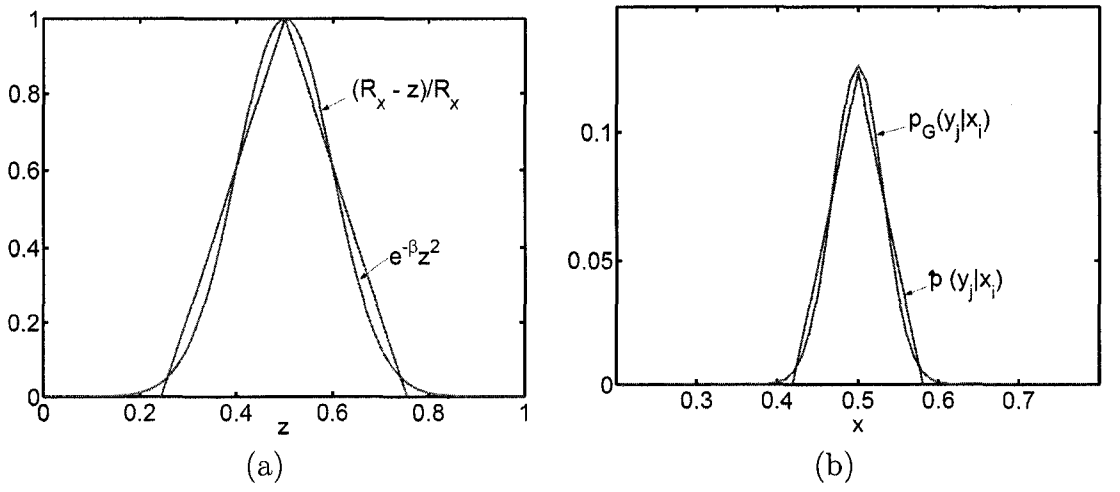


Figure 5.13: (a) Comparison between the numerator term $(e^{-\beta z^2})$ of the Gibbs distribution and the low-complexity triangle function $((R_x - z)/R_x)$ for a given pair (β, R_x). (b) Comparison between the Gibbs distribution $(p_G(y_j|x_i))$ and the low-complexity triangle distribution $(\hat{p}(y_j|x_i))$

low-complexity distributions results in a decrease by $8\% - 18\%$ in the computation time, while slightly deteriorating by $3\% - 6\%$ in the clustering results.

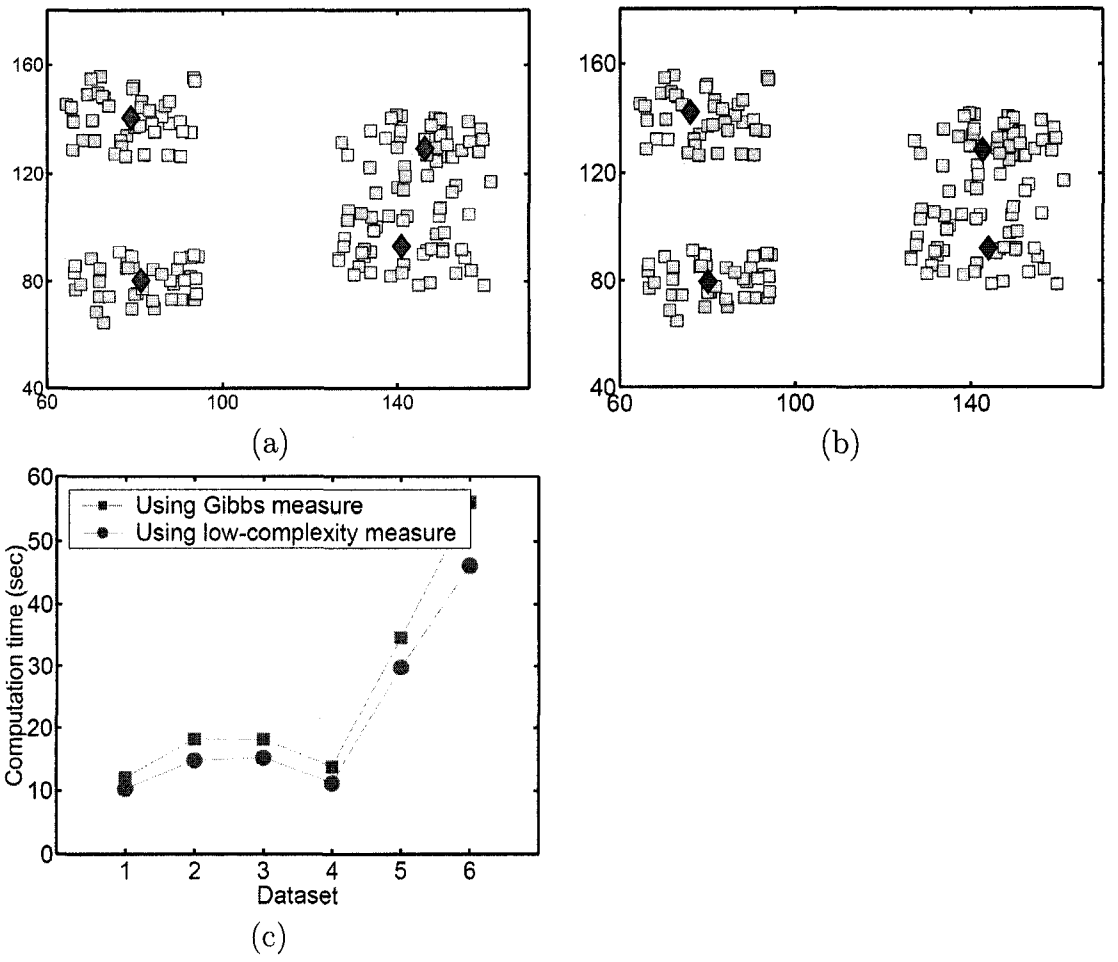


Figure 5.14: Comparison of the results obtained using (a) Gibbs distribution, (b) Low-complexity triangle distribution. Both figures show the instantaneous positions of the sites and resources at the end of the time horizon. The final locations of the resources (shown by red diamonds) do not vary considerably for the two cases.(c) Computation times for six datasets comparing the two cases.

In this chapter, we have proposed an entropy based framework for formulating and solving the dynamic coverage problem. The free energy based notion of coverage from the static setting was adapted to the dynamic setting. As shown in the simulations, the proposed framework resolves both the coverage as well as the tracking aspects of the dynamic coverage problem. The control theoretic approach to determine the velocity field for the cluster centers achieves progressively better coverage with time and is shown to be five to seven times faster than the frame-by-frame method. The hierarchical aspect of the proposed algorithm enables it to identify natural clusters

in the underlying data and characterizes the notion of cluster resolution.

Chapter 6

Conclusions

In this work, we have proposed a Maximum Entropy Principle based framework for formulating and solving a class of resource allocation problems in coverage control, clustering and locational optimization. The proposed framework has been successfully implemented to solve both the static as well as the dynamic problems. The inherent computational complexity due to the combinatorial nature of these problems is addressed efficiently in the proposed framework and the tradeoff between computation time and cluster resolution is characterized. The proposed algorithms are hierarchical in nature and thus equip them to identify natural clusters in the underlying dataset. Additionally, the notion of soft clustering and association weight entropy make the algorithms amenable to avoiding several local minima that riddle the cost surface. The proposed algorithms for the static problem were successfully implemented to solve the combinatorial library design problem is drug discovery. In the dynamic setting, the proposed framework addresses both the coverage as well as tracking aspects of the problem. The determination of cluster centers and their associated velocity field is cast as a control design problem to ensure that the algorithm achieves progressively better coverage with time.

Appendix A

MEP-based Partitioning

Partitioning interpretation

Consider the space of all partitions $\mathcal{Q}$ of Ω. For any instance $q = \{\mathcal{Y}, \mathcal{P}\} \in \mathcal{Q}$, with the partition $\mathcal{P} = \{\Omega_j\}$ and the resource locations $\mathcal{Y} = \{y_j\}$, the coverage cost in (2.1) can be rewritten as

$$D(q) = \sum_i \sum_j \chi_{ij} d(x_i, y_j), \tag{A.1}$$

where χ_{ij} is the indicator function such that $\chi_{ij} = 1, \text{if } x_i \in \Omega_j$ and 0 otherwise. Now consider a probability distribution $P(q)$ on the space of partitions $\mathcal{Q}$. Under this distribution, the average coverage cost (i.e. the expected value of distortion) is given by $\sum P(q)D(q)$, where the summation is over all possible elements in the set $\mathcal{Q}$ (i.e. all possible partitions). Since we have no prior information about the distribution over $\mathcal{Q}$, we use the MEP to determine this probability distribution. Such a distribution thus maximizes the entropy under the constraint that it achieves the average cost, and is given by a Gibbs distribution over partitions

$$P(q) = \frac{e^{-\beta D(q)}}{\sum_{q' \in \mathcal{Q}} e^{-\beta D(q')}},$$

where β can be determined from the value of the average cost [37]. The most probable set of resource locations can be determined by maximizing the marginal probability

$$P(\mathcal{Y}) = \sum_{\mathcal{P}} P(\mathcal{Y}, \mathcal{P}). \tag{A.2}$$

In order to compute (A.2), we note from (A.1) that

$$\sum_{\mathcal{P}} e^{-\beta D(\mathcal{Y},\mathcal{P})} = \prod_{x} \sum_{k} e^{-\beta d(x,y_k)} := Z, \tag{A.3}$$

where Z is called the *partition function* (as it has parallels in statistical physics [56]). Next, we define the *free energy* by the following expression

$$F := -\frac{1}{\beta} \log Z = -\frac{1}{\beta} \sum_{i} \log \sum_{k} e^{-\beta d(x_i,y_k)}. \tag{A.4}$$

Now, the marginal probability $P(\mathcal{Y})$ can be rewritten as

$$P(\mathcal{Y}) = \frac{Z}{\sum_{\mathcal{Y}'} Z} = \frac{e^{-\beta F}}{\sum_{\mathcal{Y}'} e^{-\beta F}}. \tag{A.5}$$

Thus the most probable resource location set $\mathcal{Y}$ is the one that minimizes the *free energy*

$$F = -\frac{1}{\beta} \sum_{i} \log \sum_{j} e^{-\beta d(x_i,y_j)}. \tag{A.6}$$

We now consider the entropy associated with a specific q, given by

$$H = -\sum_{x \in \Omega} p(x) \sum_{j} p(y_j|x) \log p(y_j|x) dx, \tag{A.7}$$

which measures the randomness of the distribution of the associated weights $p(y_j|x)$.

Appendix B

A Less Conservative Bound for Sensitivity With Respect to Temperature

For a dataset where the smallest distance between two distinct resource locations is μ and no sites are in the rim around each resource described by $\{x|\rho\mu < \|x - y_j\| \leq (1-\rho)\mu\}$, the bound on $\left\|\frac{dy_j}{d\beta}\right\|$ is proportional to $e^{-\beta\mu^2|1-2\rho|}$.

Proof. Since $p(y_k|x_i)$ is of the form $\frac{e^{-\beta d(x_i,y_k)}}{\sum_j e^{-\beta d(x_i,y_j)}}$, on multiplying both the numerator and the denominator by $e^{\beta d(x_i,y_k)}$, we get

$$p(y_k|x_i) = \frac{1}{1 + \sum_{j\neq k} e^{-\beta(d(x_i,y_j)-d(x_i,y_k))}} \leq \frac{1}{1 + e^{-\beta d(x_i,y_j)} e^{\beta d(x_i,y_k)}}. \tag{B.1}$$

Since there are no sites x in the rim around each y_j described by $\{x|\rho\mu < \|x - y_j\| \leq (1-\rho)\mu\}$, it implies that $d(x_i, y_j) \leq \rho^2\mu^2$ and $d(x_i, y_k) \leq (1-\rho)^2\mu^2$. Thus,

$$p(y_k|x_i) \leq \frac{1}{1 + e^{-\beta\rho^2\mu^2} e^{\beta(1-\rho)^2\mu^2}} \leq e^{\beta\mu^2(1-2\rho)}. \tag{B.2}$$

Using this bound in (2.62) and the bound obtained (2.61), we infer that the bound on $\|T_2\|$ is of the form

$$\|T_2\| \leq \hat{c}(\beta) \left\|\frac{dy_j}{d\beta}\right\| \Rightarrow \left\|\frac{dy_j}{d\beta}\right\| \leq \frac{\hat{c}(\beta)}{\Delta}, \tag{B.3}$$

where $\hat{c}(\beta)$ monotonically decreases with β and is of the form $e^{-\beta\mu^2(1-2\rho)}$. It is completely determined by the value of β and the bound of the size of the space Ω. □

Appendix C

Principle Component Analysis

In this appendix, we briefly discuss the Principle Component Analysis (PCA) based projection for dimensionality reduction of the drug discovery dataset. In PCA, an orthogonal linear transformation is applied on the dataset to map it into a new coordinate system, such that the maximum variance direction lies along the first coordinate axis (called the principal component), the second highest variance on the next axis and so on. In order to obtain a lower-dimensional projection, the higher components of the mapped data are discarded, while the few principal components are retained. A brief sketch of the algorithm is presented is below.

Algorithm

Given M vectors $x_1, x_2, \ldots, x_M$, such that $x_i \in \mathbb{R}^N,\ 1 \leq i \leq M$,

- Find the mean

$$\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i.$$

- Subtract the mean

$$\Phi_i = x_i - \bar{x}.$$

- Form the augmented matrix

$$A = [\Phi_1\ \Phi_2\ \ldots, \Phi_M].$$

- Compute the sample covariance matrix

$$C = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = AA^T.$$

- Compute the eigenvalues of the covariance matrix and rearrange them such that $\lambda_1 > \lambda_2 > \cdots > \lambda_N$.

- Compute the eigenvectors of the covariance matrix $u_1, u_2, \ldots, u_N$. These eigenvectors form the basis of $\mathbb{R}^N$, and any vector $x - \bar{x} \in \mathbb{R}^N$ can be expressed in the form

$$x - \bar{x} = b_1u_1 + b_2u_2 + \cdots + b_Nu_N.$$

- Dimensionality reduction: Select the K largest eigenvalues, and discard the rest

$$\hat{x} - \bar{x} = b_1u_1 + b_2u_2 + \cdots + b_Ku_K, \ K \ll N$$

Thus the representation of $\hat{x} - \bar{x}$ into the basis $u_1, u_2, \ldots, u_K$ is $[b_1\ b_2\ \ldots\ b_K]$.

The choice of K depends on the truncated residuals. A widely used criterion is to choose K such that

$$\frac{\sum_{i=1}^{K}\lambda_i}{\sum_{i=1}^{N}\lambda_i} > 1 - \epsilon, \tag{C.1}$$

where ϵ is the threshold for the error. In the case of drug discovery dataset, the higher dimensional data was projected on 2-d and 3-d descriptor space using the above algorithm.

References

[1] Mcmaster hts lab competition. HTS data mining and docking competition. http://hts.mcmaster.ca/downloads/82bfbeb4-f2a4-4934-b6a8-804cad8e25a0.html (accessed june 2006).

[2] D. K. Agrafiotis. Stochastic algorithms for maximizing molecular diversity. *J. Chem. Inf. Comput. Sci.*, 37:841–851, 1997.

[3] D. K. Agrafiotis, V. S. Lobanov, and F. R. Salemme. Combinatorial informatics in the post-genomics era. *Nature Reviews*, 1:337–346, 2002.

[4] D.K Agrafiotis. Multiobjective optimization of combinatorial libraries. *IBM Journal of Research and Development*, 45(3):545–566, 2001.

[5] D.K. Agrafiotis and V. S. Lobanov. Ultrafast algorithm for designing focussed combinatorial arrays. *J. Chem. Inf. Comput. Sci.*, 40:1030–1038, 2000.

[6] R. R. Baker. *The Evolutionary Ecology of Animal Migration.* Holmes & Meier Publishers, New York, 1978.

[7] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Prentice-Hall, Englewood Cliffs, N.J., 1971.

[8] J. Blaney and E. Martin. Computational approaches for combinatorial library design and molecular diversity analysis. *Curr. Opin. Chem. Biol.*, 1:54–59, 1997.

[9] R. D. Brown and Y. C. Martin. Designing combinatorial library mixtures using genetic algorithms. *J. Med. Chem.*, 40:2304–2313, 1997.

[10] Butcher. J. C. *Numerical Methods for Ordinary Differential Equations.* John Wiley, 2004.

[11] R. D. Clark. Optisim: An extended dissimilarity selection method for finding diverse representative subsets. *J. Chem. Inf. Comput. Sci.*, 37(6):1181–1188, 1997.

[12] 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.

[13] Chudova D., Gaffney S., Mjolsness E., and Smyth P. Translation-invariant Mixture Models for Curve Clustering. In *Proceedings of the 9th ACM SIGKDD*, 2003.

[14] L. Demers and G. D. Cioppa. Drug discovery: Nanotechnology to advance discovery R&D. *Genetic Engineering News*, 23(15), September 2003.

[15] K. Demirciler and A. Ortega. Reduced-Complexity Deterministic Annealing for Vector Quantizer Design. *EURASIP Journal on Applied Signal Processing*, 2005:1807–1820, 2005.

[16] Z. Drezner. *Facility Location: A Survey of Applications and Methods.* Springer Series in Operations Research. Springer Verlag, New York, 1995.

[17] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, December 1999.

[18] N. Elia and S. Mitter. Stabilization of Linear Systems with Limited Information. *IEEE Transactions on Automatic Control*, 46(9):1384–1400, September 2001.

[19] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23:298–305, 1973.

[20] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23:298–305, 1973.

[21] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, 25:619–633, 1975.

[22] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, 25:619–633, 1975.

[23] E. Frazzoli and F. Bullo. Decentralized algorithms for vehicle routing in a stochastic time-varying environment. *IEEE Conference on Decision and Control*, pages 3357– 3363, 2004.

[24] S. Galic and S. Loncaric. Cardiac image segmentation using spatiotemporal clustering. In *Proceedings of SPIE Medical Imaging*, volume 4322, pages 1199–1206, 2001.

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

[26] A. C. Good and R. A. Lewis. New methodology for profiling combinatorial libraries and screening sets: cleaning up the design process with HARPcik. *J. Med. Chem.*, 40:3226–3236, 1997.

[27] E.M. Gordon, R.W. Barrett, W.J. Dower, S.P.A. Fodor, and M.A. Gallop. Applications of combinatorial technologies to drug discovery. 2. Combinatorial organic synthesis, library screening strategies, and future directions. *J. Med. Chem.*, 37(10):1385–1401, 1994.

[28] R. Gray and E.D. Karnin. Multiple local minima in vector quantizers. *IEEE Transactions on Information Theory*, IT-28:256–361, March 1982.

[29] R. Guha. Chemistry Development Kit (CDK) descriptor calculator GUI (v 0.46). http://cheminfo.informatics.indiana.edu/ rguha/code/java/cdkdesc.html (accessed october 2006).

[30] S. Har-Peled. Clustering Motion. *Discrete and Computational Geometry*, 31(4):545–565, 2003.

[31] J.A. Hartigan. *Clustering Algorithms.* Wiley, New York, NY, 1975.

[32] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, Englewoods Cliffs, New Jersey, 1998.

[33] B. Hendrickson and M. Leland. Multidimensional spectral load balancing. *Sandia Natl. Labs*, (SAND93-0074), 1993.

[34] R. E. Higgs, K. G. Bemis, I. A. Watson, and J. H. Wikel. Experimental designs for selecting molecules from large chemical databases. *J. Chem. Inf. Comput. Sci.*, 37:861–870, 1997.

[35] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

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

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

[38] C. S. Jensen, D. Lin, and B. C. Ooi. Continuous Clustering of Moving Objects. *IEEE Transactions On Knowledge And Data Engineering*, 19(9):1161–1174, 2007.

[39] I. T. Jolliffe. *Principal Component Analysis.* Springer Series in Statistics. Springer Verlag, New York, 2002.

[40] P. Kalnis, N. Mamoulis, and S. Bakiras. On Discovering Moving Clusters in Spatio-temporal Data. *Proceedings of the Int. Symposium in Spatial and Temporal Databases*, pages 364–381, 2005.

[41] S. Kullback. The Kullback-Leibler distance. *The American Statistician*, 41:340–341.

[42] S. Lafon and A. Lee. Diffusion Maps and Coarse-Graining: A Unified Framework for Dimensionality Reduction, Graph Partitioning, and Data Set Parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, September 2006.

[43] L. D. Landau and E. M. Lifshitz. *Statistical Physics, Part 1*, volume 3. Oxford, 3rd edition.

[44] Yang J. Li Y., Han J. Clustering Moving Objects. *Proceeding of the 10th ACM SIGKDD*, 2004.

[45] C. A. Lipinski, F. Lomabardo, B. W. Dominy, and P. J. Feeny. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development setting. *Adv. Drug Del. Review*, 23:2–25, 1997.

[46] D. J. Livingston. The characterization of molecular structures using molecular properties: A survey. *J. Chem. Inf. Comput. Sci.*, 40:195–209, 2000.

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

[48] M. Lynch, C. Mosher, J. Huff, S. Nettikadan, J. Johnson, and E. Henderson. Functional protein nanoarrays for biomarker profiling. *Proteomics*, 4(6):1695–702, 2004.

[49] M. Meila and L. Xu. Multiway cuts and spectral clustering. *Neural Information Processing Systems*, 2003.

[50] S.K. Mitter. Control with limited information. Plenary lecture at International Symposium on Information Theory, Sorento, Italy, 2000.

[51] J. Mount, J. Ruppert, W. Welch, and A. N. Jain. Icepick: A flexible surface-based system for molecular diversity. *J. Med. Chem.*, 42:60–66, 1999.

[52] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis. Diffusion maps, spectral clustering and the reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, 21:113–127, 2006.

[53] D. E. Patterson, R. D. Cramer, A. M. Ferguson, R. D. Clark, and L. E. Weinberger. Neighborhood behavior: A useful concept for validation of "Molecular Diversity" descriptors. *J. Med. Chem.*, 39(16):3049–3059, 1996.

[54] A. Quarteroni, Sacco R., and F. Saleri. *Numerical Mathematics*, volume 37 of *Texts in Applied Mathematics*. Springer, 2000.

[55] D. N. Rassokhin and D. K. Agrafiotis. Kolmogorov-Smirnov statistic and its applications in library design. *J. Mol. Graph. Model.*, 18(4-5):370–384, 2000.

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

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

[58] S. Salapaka. On Combinatorial Optimization Problems with Mobile Sites and Resources. *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 6978–6983, Dec. 2005.

[59] S. Salapaka. Combinatorial Optimization Approach to Coarse Control Quantization. *Decision and Control, 2006 45th IEEE Conference on*, pages 5234–5239, Dec. 2006.

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

[61] S. Salapaka and A. Khalak. Locational Optimization Problems with Constraints on Resources. *Proceedings of 41st Allerton Conference*, pages 1240–1249, October 2003.

[62] B. Schlkopf, A. J. Smola, and K. R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[63] C. E. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois Press, Urbana, Illinois, 1949.

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

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

[66] P. Sharma, S. Salapaka, and C. Beck. A Scalable Approach to Combinatorial Library Design for Drug Discovery. *Journal of Chemical Information and Modeling*, 48(1):27–41, 2008.

[67] P. Sharma, S. Salapaka, and C. Beck. Entropy Based Algorithm for Combinatorial Optimization Problems with Mobile Sites and Resources. *Proceedings of American Control Conference*, pages 1255–1260, 2008.

[68] R. P. Sheridan. The most common chemical replacements in drug-like compounds. *J. Chem. Inf. Comput. Sci.*, 2:103–108, 2002.

[69] R. P. Sheridan, S. G. SanFeliciano, and S. K. Kearsley. Designing targeted libraries with genetic algorithms. *J. Mol. Graph. Model.*, 18:320–333, 2000.

[70] Jiuh-Biing Sheu. A Fuzzy Clustering-Based Approach to Automatic Freeway Incident Detection And Characterization. *Fuzzy Sets Syst.*, 128(3):377–388, 2002.

[71] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[72] M. Snarey, N. Terrett, P. Willet, and D. J. Wilton. Comparison of algorithms for dissimilarity-based compound selection. *J. Mol. Graph. Model.*, 15:372–385, 1997.

[73] E. D. Sontag. A Lyapunov-like characterization of asymptotic controllability. *SIAM J. Control Optim.*, 21:462–471, 1983.

[74] E. D. Sontag. A 'universal' Construction of Artstein's Theorem on Nonlinear Stabilization. *System and Control Letters*, 13(2):117–123, 1989.

[75] C. Steinbeck, C. Hoppe, S. Kuhn, M. Floris, R. Guha, and E. L. Willighagen. Recent developments of the Chemistry Development Kit (CDK) - an open-source JAVA library for chemo and bioinformatics. *Curr. Pharm. Des.*, 12(17):2110–2120, 2006.

[76] C.W. Therrien. *Decision, Estimation and Classification: An Introduction to Pattern Recognition and related topics*, volume 14. Wiley, New York, 1st edition, 1989.

[77] M. Waldman, H. Li, and M. Hassan. Novel algorithms for the optimization od molecular diversity of combinatorial libraries. *J. Mol. Graph. Model.*, 18:412–426, 2000.

[78] J. Wang and K. Ramnarayan. Toward designing drug-like libraries: a novel computational approach for prediction of important structural features. *J. Combin. Chem.*, 1:52–533, 1999.

[79] P. Willett. Computational tools for the analysis of molecular diversity. *Perspect. Drug Discovery Design*, 7/8:1–11, 1997.

[80] Q. Zhang and X. Lin. Clustering Moving Objects for Spatio-Temporal Selectivity Estimation. In *Proceedings of the 15th Australasian database conference*, pages 123–130, 2004.

Author's Biography

Puneet Sharma was born in Lucknow, Uttar Pradesh, India, on October 30, 1979. He graduated from the Indian Institute of Technology, Bombay in 2001, with a Bachelors degree in Mechanical Engineering. He completed his Masters degree in the Department of General Engineering at the University of Illinois in 2003. He has been pursuing his doctoral research at the University of Illinois since then, and will graduate in October 2008, with a Ph.D. degree in Industrial and Enterprise Systems Engineering.