DETERMINISTIC ANNEALING ALGORITHM: TUTORIAL, APPLICATION TO PICKUP AND DELIVERY PROBLEM AND COMPUTATIONAL ASPECTS

BY

PRATIK MAYUR PAREKH

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Advisers:

Professor Srinivasa Salapaka
Professor Carolyn Beck

Abstract

The deterministic annealing (DA) method, used for the solution of several nonconvex problems, offers the ability to avoid shallow local minima of a given cost surface and the ability to minimize the cost function even when there are many local minima. The method is established in a probabilistic framework through basic information-theoretic techniques such as maximum entropy and random coding. It arises naturally in the context of statistical mechanics by the emulation of a physical process whereby a solid is slowly cooled and at zero temperature assumes its minimum energy configuration. We start with the introduction to DA method and then present a tutorial to describe the algorithm steps. Also, we discuss the connections of DA method with Statistical Mechanics and Rate-Distortion Theory. Next, we present the application of DA method to pickup and deliver scheduling problem with time windows. Finally, a computational complexity analysis for DA is presented for a given temperature schedule. The case study focuses on the geometric cooling law $T(t) = \rho T(t-1), 0 < \rho < 1$, where $T(t)$ is the temperature at time t.

To my parents, for their endless love and support

Acknowledgments

First, I would like to sincerely thank my advisers Prof. Srinivasa Salapaka and Prof. Carolyn Beck for their guidance. They have been constant source of inspiration and knowledge throughout my program. They have not only guided me through my research problem but have also helped me develop as an individual and critical thinker. I would also like to thank Dimitri for clarifying concepts and helping me with my research problem.

Also, my research thesis and my Master's degree would not have been possible without the support and love of my parents who have encouraged me in all my endeavours.

I would also like to express my gratitude to my research colleagues Thinh, Chandra, Mayank, Ram and Yunwen for making this journey a memorable one. Graduate life in Urbana-Champaign has been a fun-filled journey and I owe this to my friends Manish, Vineet, Anish, Dhruv and Omkar.

Last, I would like to extend my deepest regards to some of the greatest teachers and researchers at UIUC for instilling confidence and motivation in me to keep striving for excellence.

Table of Contents

List of Figures

List of Abbreviations

DA	Deterministic Annealing Algorithm
MEP	Maximum Entropy Principle
SA	Simulated Annealing Algorithm
BA	Blahut Arimoto Algorithm
MC	Markov Chains

List of Symbols

Symbol	Description
T	Temperature
ρ	Multiplication Factor or Cooling Parameter
$\boldsymbol{C}_x$	Input Covariance Matrix
Z_x	Partition Function in context of Gibbs distribution.
α	Tolerance Value in Convergence test of Step 4) of algorithm

Chapter 1

Introduction

[1]Combinatorial resource allocation problems arise in a large number of applications such as facility location problems, data compression, strategy planning, model aggregation, and locational optimization. These problems are typically cast as optimization formulations whose cost surfaces are non-convex with many local minima; therefore finding the global minima is a prohibitive task. Due to the nonconvexity of these cost surfaces many gradient descent methods get trapped in poor local minima, depending on the initialization point. An immediate remedy is to use multiple initialization points and to choose the lowest achievable cost value as the potential global minimum [1]. Clearly, depending on the structure of the cost surface and due to the combinatorial nature of the resource allocations, such an approach is computationally impractical. In this respect, the *simulated annealing* (SA) and *deterministic annealing* (DA) algorithms are effective [1, 2, 3]. The motivating idea behind these methods originates from statistical mechanics: SA and DA emulate a physical process whereby a solid is first heated to its melting point and then is slowly cooled at a rate dictated by its heat transport speed to finally reach its minimum energy configuration [2, 4]. The SA algorithm corresponds to the evolution of a discrete-time inhomogeneous Markov chain (MC) and by starting from an initial point a random walk of bounded or unbounded variance triggers the search of the corresponding state-space and may converge in probability to the minimum energy state. In a physical annealing process, this procedure corresponds to a Brownian motion of a particle. However, the cooling schedule is very critical to the performance of SA. Assuming a given random process, cooling at a fast rate will most probably lead to a nonglobal minimum at zero temperature, while cooling at a slow rate corresponds to a waste of resources. In [5], it was shown that convergence in probability to the global minimum can be achieved if $T(t) = O((\log t)^{-1})$, where $T(t)$ is the temperature at time t. This result was sharpened by Hajek [6], who showed that the SA algorithm converges in probability if and only if $\lim_{t\to\infty} T(t) = 0$ and $\sum_{t=1}^{\infty} \exp(-d^*/T(t)) = 0$ for some appropriately defined positive constant d^*, leading to cooling schedules of the form $T(t) = d/\log t$ for any $d \geq d^*$. Such schedules are generally nonadmissible for real world

[1]This chapter relies on the contents in [25] which is an IEEE copyright owned article. Permission has been taken from IEEE and authors (P. M. Parekh, D. Katselis, C. Beck, S. Salapaka) to include sections in the thesis. The article is accepted for Proceedings of American Control Conference and publication is due in July 2015.

applications. Moreover, the results in [3, 5] use a Gaussian assumption on the aforementioned random walk, leading to bounded variance steps. In [4], it was shown that using Cauchy-distributed (infinite variance) perturbations, cooling schedules of order $O(t^{-1})$ can be achieved. On the other hand, the DA algorithm alleviates the random walk aspect of SA by replacing the stochastic search with an expectation. At the same time, the DA inherits the positive annealing attributes of SA to better identify the global minima of a cost surface, but more importantly it allows for geometric cooling laws leading to performance that is typically significantly faster than the SA.

DA algorithm is based upon the maximum entropy principle (MEP) and does not have a random initialization aspect to it. In this algorithm, we start with one cluster and gradually identify the underlying clusters hierarchically as the algorithm progresses. The annealing aspect attempts to avoid the local minima as the algorithm is running. To promote the understanding of DA, we note that at high temperatures the cost surface is convex under some mild assumptions. Therefore, it can be safely assumed that the initial minimum is unique, thus, global. Through an appropriate cooling schedule, the DA aims at tracking the global minimum of the cost surface, as the temperature is lowered and the surface gradually assumes its nonconvex form. The DA has been successfully used in various applications such as clustering, source compression/vector quantization, graph-theoretic optimization and image analysis [1, 7, 8, 9, 10]. It shares connections with the computation of rate-distortion functions in information theory [8, 14, 15]. More explicitly, it has similarities with the alternating-minimization Blahut-Arimoto (BA) algorithm for the computation of rate-distortion functions and channel capacities [16, 17]. Furthermore, due to the increased interest in tracking algorithms for surveillance and military applications [18, 19], the DA algorithm has been used in the context of the dynamic coverage control problem [9, 11], simultaneous locational optimization and multihop routing [20], and in aggregation of graphs and Markov chains [10]. For these problems, an understanding of the computational effort required by the algorithms is important for efficient implementations.

In Chapter (2), we first review the basic principles of the DA algorithm and its connections with statistical mechanics and rate-distortion theory. In the tutorial, we discuss the flow of DA algorithm and equations governing the cluster locations and associations probabilities[2]. We also discuss the convergence criteria used for stopping the algorithm. Next, in Chapter (3) we present the application of DA method to a pickup or delivery problem with time windows. In this problem, we assume pickup or delivery of boxes to a single depot and each box needs to be serviced within certain time window. The challenge is scheduling the arrival times for the buses such that the boxes get picked up or delivered within the specified time window. The reader should note that, we apply the DA method to the same problem with capacity constraints, i.e. each

[2]The association probabilities will be discussed elaborately in the next chapter

bus has a certain capacity of boxes. The problem and its variants will be explained in Chapter (3). In Chapter (4), we give the analysis of a computational study of the DA algorithm in usual topological spaces while focusing on the geometric cooling schedule $T(t) = \rho T(t-1), 0 < \rho < 1$. A numerical study of the DA algorithm is then performed to promote a better understanding of its practical behavior. Finally, in the last chapter, we summarize the findings and give an overview into possible future work in this area.

Chapter 2

Deterministic Annealing: Tutorial

2.1 Tutorial

[1]To draw connections with the rate-distortion theory later on, we use in the following the terminology that applies to both rate-distortion and clustering setups. We assume that we have a source that produces a sequence of i.i.d. input vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ according to some distribution $p(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}$. We also assume that there exists an encoding function $\boldsymbol{y}(\boldsymbol{x})$ that maps the input vector $\boldsymbol{x}$ to the best reproduction codevector in some finite set $\mathcal{Y}$. In the clustering setup, the input vectors correspond to the training set, while $\mathcal{Y}$ corresponds to the set of some appropriately defined cluster centroids. To clarify the appropriateness notion of the centroids, a distortion definition is necessary. It is also assumed that $\max_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{y}\in\mathcal{Y}} d(\boldsymbol{x},\boldsymbol{y}) < \infty$. With such a measure, appropriate centroids are those minimizing the average distortion for a given training set and a given codebook $\mathcal{Y}$. The most usual distortion measure in the clustering context is the squared error distortion $d(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\|_2^2$. The advantages of this measure is its mathematical tractability and its relationship to least-squares (LS) prediction.

Given a joint probability distribution $p(\boldsymbol{x},\boldsymbol{y})$ defined on $\mathcal{X}\times\mathcal{Y}$, the expected distortion is expressed as $D = \sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{x},\boldsymbol{y})d(\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{y}|\boldsymbol{x})d(\boldsymbol{x},\boldsymbol{y})$, where $p(\boldsymbol{y}|\boldsymbol{x})$ is called *association probability* in the clustering context and *encoding* function in the rate-distortion framework [2]. The optimization problem is the minimization of the Lagragian, $F = D - TH$. Here D represents the distortion, T represents the temperature and H is the measure of Shannon entropy given by the relation, $H = -\sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{x},\boldsymbol{y})\log p(\boldsymbol{x},\boldsymbol{y})$. At high temperature values, we are maximizing entropy and as the temperature is reduced, we start reducing the distortion. As T approaches zero, we directly minimize D to get get a hard clustering solution.

In the DA framework, it can easily be shown that the optimal association probabilities are given by the

[1]This chapter relies on the contents in [25] which is an IEEE copyright owned article. Permission has been taken from IEEE and authors (P. M. Parekh, D. Katselis, C. Beck, S. Salapaka) to include sections in the thesis. The article is accepted for Proceedings of American Control Conference and publication is due in July 2015.

[2]Note that the a posteriori conditional probability $p(\boldsymbol{x}|\boldsymbol{y})$ defines a *decoding* function in information theory.

Gibbs distribution [1]:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(-d(\boldsymbol{x},\boldsymbol{y})/T)}{Z_{\boldsymbol{x}}}, \tag{2.1.1}$$

where $Z_{\boldsymbol{x}} = \sum_{\boldsymbol{y}\in\mathcal{Y}} \exp(-\frac{d(\boldsymbol{x},\boldsymbol{y})}{T})$ is a normalizing constant, also known as the *partition function* in statistical physics [15]. After accounting for minimization using the optimal association probabilities, the Lagragian reduces to

$$F^* = -T \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \log \sum_{\boldsymbol{y}\in\mathcal{Y}} \exp(-\frac{d(\boldsymbol{x},\boldsymbol{y})}{T}) \tag{2.1.2}$$

Minimizing F^* for the squared error distortion measure, the optimal locations of the centroids can be expressed as[1]

$$\boldsymbol{y} = \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}|\boldsymbol{y})\boldsymbol{x} = \frac{\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})\boldsymbol{x}}{p(\boldsymbol{y})}. \tag{2.1.3}$$

The mass constrained approach is preferred over the minimization technique used above because it leads to a method independent of initialization [1]. In this approach, we assume unlimited number of codevectors. Let p_y denote the fraction of codevectors coincident at location $\boldsymbol{y}$. Subjected to the constraint $\sum_{\boldsymbol{y}\in\mathcal{Y}} p_y = 1$, the optimization problem reduces to

$$F^* = -T \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \log \sum_{\boldsymbol{y}\in\mathcal{Y}} p_y \exp(-\frac{d(\boldsymbol{x},\boldsymbol{y})}{T}) + \lambda(\sum_{\boldsymbol{y}\in\mathcal{Y}} p_y = 1) \tag{2.1.4}$$

λ represents the lagrange multiplier and it can be shown that it is equal to the temperature in the cooling schedule, i.e $\lambda = T$. By minimizing the lagragian, the optimal association propabilities are given by

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p_y \exp(-d(\boldsymbol{x},\boldsymbol{y})/T)}{Z_{\boldsymbol{x}}}, \tag{2.1.5}$$

where $Z_{\boldsymbol{x}} = \sum_{\boldsymbol{y}\in\mathcal{Y}} p_y \exp(-\frac{d(\boldsymbol{x},\boldsymbol{y})}{T})$ is a normalizing constant. Moreover, it turns out that the optimal locations computation does not change and can be found using (2.1.3). Also, in regard to p_y it can be shown that $p_y = p(\boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})$[1].

Eq. (2.1.3) and (2.1.5) imply that we can construct an alternating-minimization algorithm as follows: Given a codebook $\mathcal{Y}$ we can use (2.1.5) to optimize the association probabilities and with these probabilities we can update the centroid locations based on (2.1.3). Iterating over these two steps till convergence yields the *DA algorithm for clustering*, which is summarized in the following[3]:

1. *Set Limits*: Set a maximum number of codevectors $K_{\max}$ and a minimum temperature $T_{\min}$.

2. *Initialization*: $T > 2\lambda_{\max}(\boldsymbol{C}_x)$, $K = 1$, $\boldsymbol{y}_1 = \sum_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{x}p(\boldsymbol{x})$ and $p(\boldsymbol{y}_1) = 1$. Here $\lambda_{\max}(\boldsymbol{C}_x)$ denotes the

[3]Note the algorithm steps are for the mass-constrained version of the algorithm

maximum eigenvalue of the input covariance matrix $\boldsymbol{C}_x$.

3. If $K < K_{\max}$, create codevectors according to $\boldsymbol{y}_j' := \boldsymbol{y}_j + \boldsymbol{\delta}$ and $\boldsymbol{y}_j'' := \boldsymbol{y}_j - \boldsymbol{\delta}$. Here $\boldsymbol{\delta}$ is a random perturbation. Set $p(\boldsymbol{y}_j') := p(\boldsymbol{y}_j)/2, p(\boldsymbol{y}_j'') := p(\boldsymbol{y}_j)/2$.

4. Update all $2K$ codevectors:

$$\boldsymbol{y}_i = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) p(\boldsymbol{y}_i|\boldsymbol{x}) \boldsymbol{x} / p(\boldsymbol{y}_i) \tag{2.1.6}$$

$$p(\boldsymbol{y}_i|\boldsymbol{x}) = \frac{p(\boldsymbol{y}_i) \exp(-\|\boldsymbol{x} - \boldsymbol{y}_i\|^2/T)}{\sum_{j=1}^{2K} p(\boldsymbol{y}_j) \exp(-\|\boldsymbol{x} - \boldsymbol{y}_j\|^2/T)} \tag{2.1.7}$$

$$p(\boldsymbol{y}_i) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) p(\boldsymbol{y}_i|\boldsymbol{x}). \tag{2.1.8}$$

5. *Convergence Test*: If not satisfied go to step 4).

6. If $T < T_{\min}$ perform the last iteration at $T = 0$ and STOP.

7. *Cooling Step*: $T \leftarrow \rho T$, $\rho \in (0, 1)$.

8. Go to step 3).

The temperature initialization is appropriate, since it has been shown in [1] that above this value no codevector splitting occurs. The idea of codevector splitting or *phase transition* will be further explained in the next section.

Convergence test in Step 5): The convergence test can be implemented in various ways, e.g., as the norm of the difference of subsequent codevectors falling below a predefined threshold or the difference of successive values of the implicit objective function that the DA algorithm minimizes, called *free energy*, falling below a predefined tolerance. In our analysis, the test used is of the form $\|F(\boldsymbol{y}(n)) - F(\boldsymbol{y}(n-1))\| \leq \alpha$, where F denotes the Lagrangian and 'n' denotes the iterate count. Also, $\alpha < 1$, tolerance for convergence test, is chosen to be sufficiently small. In any case, for worst-case computational considerations we impose an empirical upper bound, $n_{\max}$, on the maximum number of iterations in Step 4), which is an implicit function of the clustering model parameters. $n_{\max}$ is considered sufficiently large to be a legitimate overestimate of the iterations needed for the convergence of Step 4).

Remark: The reader may note that the DA algorithm for clustering in this paper does not include the calculation of critical temperatures. Step 7) used in the mass-constrained implementation for calculating the critical temperatures can be expensive in higher dimensions, since it corresponds to the solution of an eigenvalue problem. Therefore, it can be replaced by a simple perturbation. In this case, we always keep two codevectors at each location and perturb them when the temperature is updated. Above the critical

temperature, these codevectors will merge naturally in Step 4). They will split when the system undergoes a phase transition. In other words, at each temperature, we double the number of codevectors and perform the computations of Step 4). In case of the phase transition, the output codevector will not merge and remain distinct.

2.2 Connections to Statistical Mechanics and Rate-Distortion Theory

Consider a system of n particles, which can be in various microstates. Each microstate is designated by a vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$, where the elements can be positions, angular momenta or spins. With each state $\boldsymbol{x}$ we associate a Hamiltonian, i.e., an energy function $\mathcal{H}(\boldsymbol{x})$ [15, 21]. Then, in thermal equilibrium the probability of occurrence of $\boldsymbol{x}$ is given by the *Boltzmann-Gibbs* distribution $w(\boldsymbol{x}) = \exp(-\beta\mathcal{H}(\boldsymbol{x}))/Z_n(\beta)$. Here, $\beta = 1/(\kappa T)$, where κ is Boltzmann's constant and T is the temperature. Moreover, $Z_n(\beta)$ is the earlier mentioned *partition function*, expressed as $Z_n(\beta) = \sum_{\boldsymbol{x}} \exp(-\beta\mathcal{H}(\boldsymbol{x}))$. Through the partition function, an important physical quantity is defined, namely the *Helmholtz free energy* $F = -\ln Z_n(\beta)/\beta$. The Boltzmann-Gibbs distribution can be obtained as the *maximum entropy* distribution under an energy constraint. In this setup, β corresponds to a Lagrange multiplier balancing the entropy and the average energy contributions in F. In the clustering context, the Hamiltonian corresponds to the average distortion D and it turns out that $F = D - TH$, where H is the Shannon entropy given by the formula $H(\boldsymbol{X}, \boldsymbol{Y}) = -\sum_{\boldsymbol{x},\boldsymbol{y}} p(\boldsymbol{x}, \boldsymbol{y}) \log p(\boldsymbol{x}, \boldsymbol{y})$ [1]. Here, $\boldsymbol{X}, \boldsymbol{Y}$ correspond to random vectors in $\mathcal{X}$ and $\mathcal{Y}$, respectively.

The question that arises is how the notion of *maximum entropy principle* in the DA context emerge? By the second law of thermodynamics, the entropy of an isolated system can only increase. This leads to the conclusion that the entropy is maximized at thermal equilibrium. When the system is not isolated, the maximum entropy principle is replaced by the *minimum free energy* principle asserting that F cannot increase, and therefore it reaches its minimum at thermal equlibrium [22]. The DA algorithm is based on the minimization of the average distortion D subject to an assumption on the allowed randomness of the solution, which is dictated by the maximum entropy principle. The annealing process on the temperature plays the role of an external force or the role of exchanging energy between the system and its environment. As the temperature is lowered sufficiently slowly, the system is always kept at equilibrium and therefore at zero temperature it assumes its minimum energy configuration, which is the best hard clustering solution.

We now turn our attention to rate-distortion theory. The basic problem is to determine the minimum expected distortion D achievable at a particular rate R, given a source distribution $p(\boldsymbol{x})$ and a distortion

measure $d(\cdot,\cdot)$. One of the most critical results in this theory is that joint descriptions of sequences of random variables are always more efficient than describing each random variable separately, even when the random variables defining the sequence are independent [14]. For an i.i.d. source with distribution $p(\boldsymbol{x})$ and bounded distortion function, a basic theorem in rate-distortion theory states that the corresponding *rate-distortion function* $R(D)$ can be computed as follows [14]:

$$R(D) = \min_{q(\boldsymbol{y}|\boldsymbol{x}):\sum_{\boldsymbol{x},\boldsymbol{y}} p(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})d(\boldsymbol{x},\boldsymbol{y})\leq D} I(\boldsymbol{X};\boldsymbol{Y}). \tag{2.2.1}$$

Here, $I(\boldsymbol{X};\boldsymbol{Y})$ denotes the mutual information. Clearly, $R(D)$ corresponds to the minimum achievable rate at distortion D.

The rate-distortion function $R(D)$ can also be expressed in terms of the relative entropy as follows [14]:

$$R(D) = \min_{q\in\mathcal{B}} \min_{p\in\mathcal{A}} D(p \parallel q). \tag{2.2.2}$$

Here, $\mathcal{A}$ is the set joint distributions with marginal $p(\boldsymbol{x})$ that satisfy the distortion constraints and $\mathcal{B}$ is the set of product distributions $p(\boldsymbol{x})r(\boldsymbol{y})$ with arbitrary $r(\boldsymbol{y})$. These function sets are convex and therefore alternating-minimization (AM) approaches can be applied to compute $R(D)$. Since the relative entropy corresponds to a *Bregman distance*, such AM approaches formulating generalized projections onto convex sets (POCs) are guaranteed to converge [23]. In the information-theoretic framework, the AM method computing $R(D)$ is the Blahut-Arimoto (BA) algorithm [16, 17]. More explicitly, the BA assumes an initial output distribution $r(\boldsymbol{y})$ and computes $q(\boldsymbol{y}|\boldsymbol{x})$ that minimizes the mutual information as

$$q(\boldsymbol{y}|\boldsymbol{x}) = \frac{r(\boldsymbol{y})\exp(-\mu d(\boldsymbol{x},\boldsymbol{y}))}{\sum_{\boldsymbol{y}} r(\boldsymbol{y})\exp(-\mu d(\boldsymbol{x},\boldsymbol{y}))}, \tag{2.2.3}$$

where $\mu > 0$ is a user-defined parameter. It then updates $r(\boldsymbol{y})$ as $r(\boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})$. These two steps are repeated till convergence. The reader may observe the similarities between these two steps and step 4) in the DA algorithm.

Finally, to connect the dots between statistical mechanics, rate-distortion theory and the DA algorithm, we note that if $w(\boldsymbol{x})$ is the Boltzmann-Gibbs distribution characterizing the microstate $\boldsymbol{x}$ and $\pi(\boldsymbol{x})$ is an arbitrary distribution on $\boldsymbol{x}$, it can be shown that [24]

$$D(\pi(\boldsymbol{x}) \parallel w(\boldsymbol{x})) = \beta(F_\pi - F_w), \tag{2.2.4}$$

where F_ϕ denotes the Helmholtz free energy defined with respect to distribution ϕ. This expression certifies the correspondences of both statistical mechanics and rate-distortion theory with the DA framework.

Remark: A last notion that needs to be clarified is that of *phase transitions*. For the system of n particles in statistical physics the free energy for usual models, such as the Ising or the liquid-gas models, can be expressed via mean field approximations by the formula $F = -\ln Z_n(\beta)/\beta$ [22]. Landau theory of phase transitions explains how the shape of the free-energy surface changes as the temperature changes, leading to splittings of existing optima. This attribute is inherited by the DA algorithm. When a phase transition occurs, some of the current codevectors split.

Chapter 3

DA: Application to Pickup and Delivery Problem

3.1 Introduction

In this chapter, we discuss the application of DA algorithm to a Pickup and Delivery Problem with time windows. The problem is a simplified version of the general broader class of the vehicle routing problem which involves the construction of optimal paths between multiple depots to satisfy pickup or delivery requests subjected to capacity and time window constraints. Over the past decades, there is increasing research in the field of freight transportation optimization to come up with efficient techniques to increase overall profits and to reduce the environment impact of fossil fuels. The algorithms must aim at achieving global minima in the least possible computation time.

The simplified version, tackled here, consists of one depot and pickup or delivery of N boxes to or from that single depot by K buses with each box having a certain time range within which it must be serviced[1]. If the box is not serviced in its corresponding time window, the box goes waste and the corresponding cost has to be incurred by the depot. We consider three classes of problems and present the flexibility of DA method to account for capacity and multicapacity constraints. In the problem involving capacity constraints, each bus has the constraint to hold a pre-defined quantity of boxes. For the problem involving multicapacity constraints, we consider each box to be one among p types and each bus has the pre-defined capacity to hold i^{th} type of box where i represents the index for the type of the box.

The problem mentioned above can be considered a resource allocation problem and a exhaustive search for the optimal solution in resource allocation problems is computationally expensive and a difficult problem to solve. The optimization problem, in this case, is non convex in nature and the ability of the algorithm to avoid local minima plays an important role in search for better solutions. Development of computationally efficient algorithms can help in getting solutions quickly and with lesser amount of computational resources. The advantages of DA method include the ability to avoid local minima and the flexibility to account for capacity constraints. In the coming section, we discuss the mathematical formulation and solution approach

[1]By the box being serviced, we mean the box is getting picked up or delivered

to the problem.

3.2 Mathematical Formulation

For our problem, we assume N boxes which need to collected or deposited to one depot within their corresponding time windows on a single day. The boxes here represent packages with pre-defined time windows within which they need to be collected or deposited by the buses for them to not go as waste. In other words, if a box is not serviced within its time window, the depot has to incur the corresponding cost of the box. Boxes are represented as $\{\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3, ..., \boldsymbol{B}_N\}$. The cost associated with box $(\boldsymbol{B}_i)$ is denoted by (C_i), $1 \leq i \leq N$. If we fail to pick up or deliver this box, the loss to the depot is (C_i). Each box $(\boldsymbol{B}_i)$ has its own time window given by the range $[\boldsymbol{t}_i^i, \boldsymbol{t}_i^f]$, where $(\boldsymbol{t}_i^i)$ represents the initial time and $(\boldsymbol{t}_i^f)$ represent the final time. The pickup or delivery must be completed within this window. We have K buses, represented by the set $\{\boldsymbol{Bs}_1, \boldsymbol{Bs}_2, \boldsymbol{Bs}_3, ..., \boldsymbol{Bs}_K\}$, which can be used for pickup. The goal is to pickup boxes in such a way that the least amount of losses is incurred by depot where the total loss is the summation of costs of the boxes which did not get serviced. The decision variables are K bus arrival times represented by $\boldsymbol{T}_j$ where $1 \leq j \leq K$. Also, we assume the time required to load the boxes into the buses is negligible. If all the buses have the same costs, the problem can be viewed as sending the buses at the intersection of time ranges such that the cardinality of the intersections is maximum. However, with large N and different costs, the complexity and difficulty of the problem increases. We consider the types of problems which is as follows:

3.2.1 P1: No constraints

In this problem, all boxes are of the same types and there is no constraint on the holding capacity of buses. The solution aims at getting the bus arrival times so that minimum losses are incurred.

3.2.2 P2: Capacity constraints

In this problem, all boxes are of the same types and we have capacity constraints on the buses. In other words, each bus must have the number of boxes serviced equal to the pre-defined holding capacity. For each bus $\boldsymbol{Bs}_j$, the holding capacity is given by W_j, $1 \leq j \leq K$. The solution aims at getting the bus arrival times so that minimum losses are incurred subjected to capacity constraints given by $\lambda_j = W_j$ where $1 \leq j \leq K$. Here, λ_j denotes the number of boxes actually being serviced by the j^{th} bus.

3.2.3 P3: Multiple Types Capacity constraints

In this problem, all boxes are of the different types and we have p types of boxes. Let the number of l^{th} type of box being transferred by the j^{th} bus be given by λ_{jl}, where $1 \leq j \leq K$ and $1 \leq l \leq p$. The holding capacity constraints for l^{th} type of box and j^{th} bus is denoted by W_{jl}. The solution aims at getting the bus arrival times so that minimum losses are incurred to the depot subjected to the constraints given by $\lambda_{jl} = W_{jl}$ where $1 \leq j \leq K$ and $1 \leq l \leq p$.

3.3 Solution Approach

The solution is implemented using the clustering framework and terminology. The time window information for each box is used as the source information and corresponding output codevectors representing the bus arrival times are found using the DA algorithm. We incorporate the time window information into '$\boldsymbol{x}_i$', $1 \leq i \leq N$ for each box and our goal is to find $\boldsymbol{y}_j$, $1 \leq j \leq K$, which represent the bus arrival times[2], using the $\boldsymbol{x}_i$ information.

The input vectors $\boldsymbol{x}_i$ are created using the time windows for each box $\boldsymbol{x}_i = \frac{(\boldsymbol{t}_i^i + \boldsymbol{t}_i^f)}{2}$, $1 \leq i \leq N$. Here, $(\boldsymbol{t}_i^i)$ represents the initial time and $(\boldsymbol{t}_i^f)$ represent the final time of the time window for the i^{th} box. The output vectors, $\boldsymbol{y}_j = \boldsymbol{T}_j$, represent the bus arrival times to the depot, where $1 \leq j \leq K$. The association probabilities $p(\boldsymbol{y}_j|\boldsymbol{x}_i)$ represent the association between the i^{th} box and j^{th} bus. The importance of the box is characterized by the cost, $p(\boldsymbol{x}_i) = \frac{C_i}{\sum_{1 \leq i \leq N} C_i}$. Here, $p(\boldsymbol{x}_i)$ represent the weights assigned to each $\boldsymbol{x}_i$. As the DA method is executed, we get the cluster locations indicating the bus arrival times and the association probabilities tagging the box to its corresponding bus. If the bus arrival time happens to be outside the time window designated for the box $(\boldsymbol{B}_i)$, we consider the box to be wasted and the depot incurs the corresponding cost (C_i). As discussed in Chapter (2), the association probabilities are governed by the gibbs distribution and calculated using (2.1.6) and the cluster locations are dictated by the implicit equation given by (2.1.7). We follow the same methodology and temperature scheduling for the simulations.

Based on the type of the problem, the algorithm differs in the computation of association probabilities which is discussed below.

3.3.1 P1: No constraints

In this problem, all boxes are of the same types and there is no constraint on the holding capacity of buses. We follow the mass constrained DA algorithm to compute the association probabilities and cluster locations.

[2]Note: $\boldsymbol{y}_j = \boldsymbol{T}_j$

The Lagrangian being minimized is given by

$$\underset{\boldsymbol{y}_j,\, p(\boldsymbol{y}_j|\boldsymbol{x}_i)}{min} \sum_{\boldsymbol{x}_i} p(\boldsymbol{x_i}) \sum_{\boldsymbol{y_j}} p(\boldsymbol{y_j}|\boldsymbol{x_i})(\boldsymbol{x_i} - \boldsymbol{y_j})^2 \tag{3.3.1}$$

The usual DA algorithm directly solves for the association probabilities and cluster locations.

3.3.2 P2: Capacity constraints

We alter the algorithm by assuming the multiplicity of λ_j clusters at the cluster location $\boldsymbol{y}_j$ [13]. In other words, λ_j number of clusters are located at $\boldsymbol{y}_j$. In this case, the partition function used in computing association probabilities changes to

$$Z_x = \sum_j \lambda_j \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2}{T}) \tag{3.3.2}$$

The corresponding association probabilities are given by

$$p(\boldsymbol{y}_j|\boldsymbol{x}_i) = \frac{\lambda_j \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2}{T})}{\sum_k \lambda_k \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_k\|^2}{T})} \tag{3.3.3}$$

After accounting for the association probabilities, the Lagrangian being minimized reduces to

$$\underset{\boldsymbol{y}_j,\, p(\boldsymbol{y}_j|\boldsymbol{x}_i),\, q_j}{min} \{-T \sum_i p(\boldsymbol{x_i}) \log(\sum_j \lambda_j \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2}{T})) + \sum_j q_j(\lambda_j - W_j)\} \tag{3.3.4}$$

Here, q_j $(1 \leq j \leq K)$ denote the Lagrange multipliers corresponding to constraints $\lambda_j = W_j$. The optimal set of cluster locations should satisfy $\frac{\partial F}{\partial \boldsymbol{y}_j} = 0$ and we get the same implicit equation

$$\boldsymbol{y}_j = \sum_{\boldsymbol{x_i}} \frac{p(\boldsymbol{y}_j|\boldsymbol{x}_i)\boldsymbol{x}_i}{p(\boldsymbol{y}_j)} \tag{3.3.5}$$

Reduction of the Lagrangian with respect to q_j or computing $\frac{\partial F}{\partial q_j} = 0$ gives us $\lambda_j = W_j$, $1 \leq j \leq K$. These values of λ_j can be used in the computation of association probabilities in (3.3.3). To summarize, we can follow the geometric temperature scheduling and at each temperature step the association probabilities and cluster locations can be computed using (3.3.3) and (3.3.5) respectively.

Remark: The technique above makes sure that $p(\boldsymbol{y}_j)$, $1 \leq j \leq K$ remains in the same proportion as the corresponding W_j. In other words, by using the formulation above, we get $\frac{p(\boldsymbol{y}_{j1})}{p(\boldsymbol{y}_{j2})} = \frac{W_{j1}}{W_{j2}}$, $1 \leq j1, j2 \leq K$. If

we are aiming towards obtaining $p(\boldsymbol{y}_j) = W_j$, $1 \leq j \leq K$, the Lagrangian being minimized reduces to

$$\min_{\boldsymbol{y}_j,\, p(\boldsymbol{y}_j|\boldsymbol{x}_i),\, q_j} \{-T\sum_i p(\boldsymbol{x_i}) \log(\sum_j \lambda_j \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2}{T})) + \sum_j q_j(p(\boldsymbol{y}_j) - W_j) + r(\sum_j \lambda_j - 1)\} \tag{3.3.6}$$

Here, q_j $(1 \leq j \leq K)$ and r denote the Lagrange multipliers corresponding to constraints $p(\boldsymbol{y}_j) = W_j$ and $(\sum_j \lambda_j - 1)$ respectively. In terms of solving for $\boldsymbol{y}_j$ and $p(\boldsymbol{y}_j|\boldsymbol{x}_i)$, this is a much harder problem to solve. Hence, we adopt the simpler formulation depicted before.

3.3.3 P3: Multiple Types Capacity constraints

The difference in this problem setting compared to (P2) is that we account for only the same type when calculating the association probabilities. We alter the algorithm by assuming the multiplicity of λ_{jl} clusters at the cluster location $\boldsymbol{y}_j$ for the l^{th} type of the box [13]. In this case, the partition function used in computing association probabilities changes to

$$Z_x = \sum_j \lambda_{jl} \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2}{T}) \tag{3.3.7}$$

'l' corresponds to type of the box represented by $\boldsymbol{x}_i$. The corresponding association probabilities are given by

$$p(\boldsymbol{y}_j|\boldsymbol{x}_i) = \frac{\lambda_{jl} \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2}{T})}{\sum_k \lambda_{kl} \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_k\|^2}{T})} \tag{3.3.8}$$

After accounting for the association probabilities, the optimization problem reduces to

$$\min_{\boldsymbol{y}_j,\, p(\boldsymbol{y}_j|\boldsymbol{x}_i),\, q_{jl}} \{-T\sum_i p(\boldsymbol{x_i}) \log(\sum_j \lambda_{jl} \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2}{T})) + \sum_{1 \leq j \leq K} q_{jl}(\lambda_j l - W_j l)\} \tag{3.3.9}$$

In (3.3.9), 'l' corresponds to the type of box represented by $\boldsymbol{x}_i$. Also, q_{jl} $(1 \leq j \leq K,\, 1 \leq l \leq p)$ denote the Lagrange multipliers corresponding to constraints $\lambda_{jl} = W_{jl}$. The optimal set of cluster locations must satisfy $\frac{\partial F}{\partial \boldsymbol{y}_j} = 0$ and we get the same implicit equation

$$\boldsymbol{y}_j = \sum_{\boldsymbol{x_i}} \frac{p(\boldsymbol{y}_j|\boldsymbol{x}_i)\boldsymbol{x}_i}{p(\boldsymbol{y}_j)} \tag{3.3.10}$$

Reduction of the Lagrangian with respect to q_{jl} or computing $\frac{\partial F}{\partial q_{jl}} = 0$ gives us $\lambda_{jl} = W_{jl}$, $1 \leq j \leq K$, $1 \leq l \leq p$. The values of λ_{jl} can be used in the computation of association probabilities in (3.3.8). To summarize, we can follow the geometric temperature scheduling and at each temperature step the association

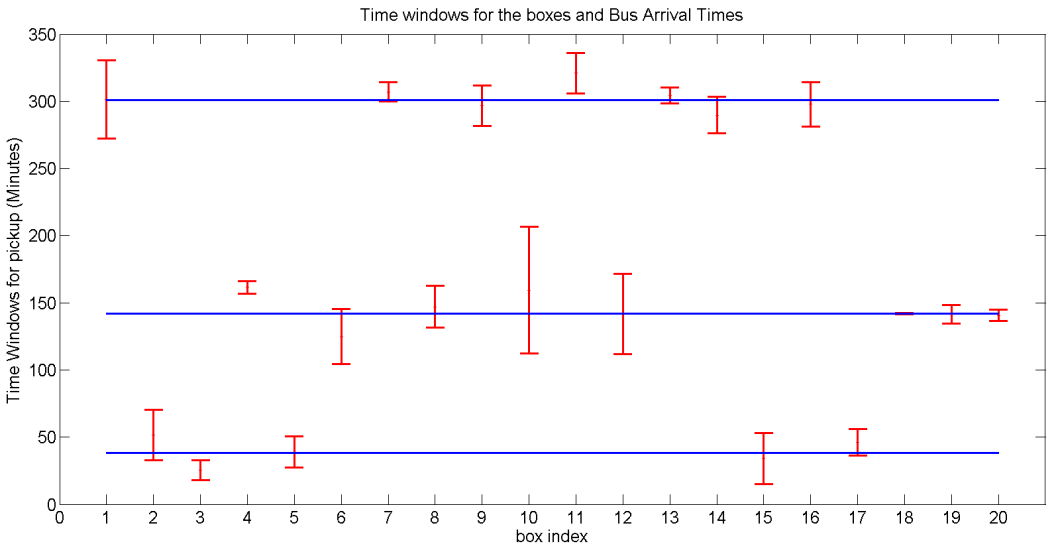

Figure 3.1: Time Windows for 20 Boxes (Red Intervals) and the Arrival Times (Cluster Locations(blue lines)) for 3 buses.

probabilities and cluster locations can be computed using (3.3.8) and (3.3.10) respectively.

3.4 Simulations

For each of the specified problem types, we perform the simulations which are discussed below.

3.4.1 P1: No constraints

For problem (P1), we start with 20 boxes needing to be serviced by 3 buses. The depot incurs the corresponding cost if the box does not get serviced. In Fig. 3.1, blue lines denote the final clustered information[3] after the DA algorithm is run. The red bars denote the time windows for each boxes. If the bus arrival time does not lie in the time window for a box, the box goes waste and the corresponding cost is incurred by the depot. The time data is in minutes and we have chosen the time window to lie within $[0, 350]$ minutes. In Fig. 3.2, the plot shows the corresponding costs for the boxes which were not serviced. Three boxes indexed 3, 4 and 11 do not have any bus arrivals within their time windows. Their corresponding costs (\$11,\$6,\$10), amounting to a total loss of \$26, is 12% of the total costs of all boxes (\$210). The pie chart on the right shows the mass associated with each cluster. In other words, it is a depiction of the masses given by, $p(\boldsymbol{y}_j) = \sum_i p(\boldsymbol{x}_i)p(\boldsymbol{y}_j|\boldsymbol{x}_i)$.

The DA algorithm was also compared with the Lloyd's algorithm. Simulations were performed on different

[3]Bus arrival times

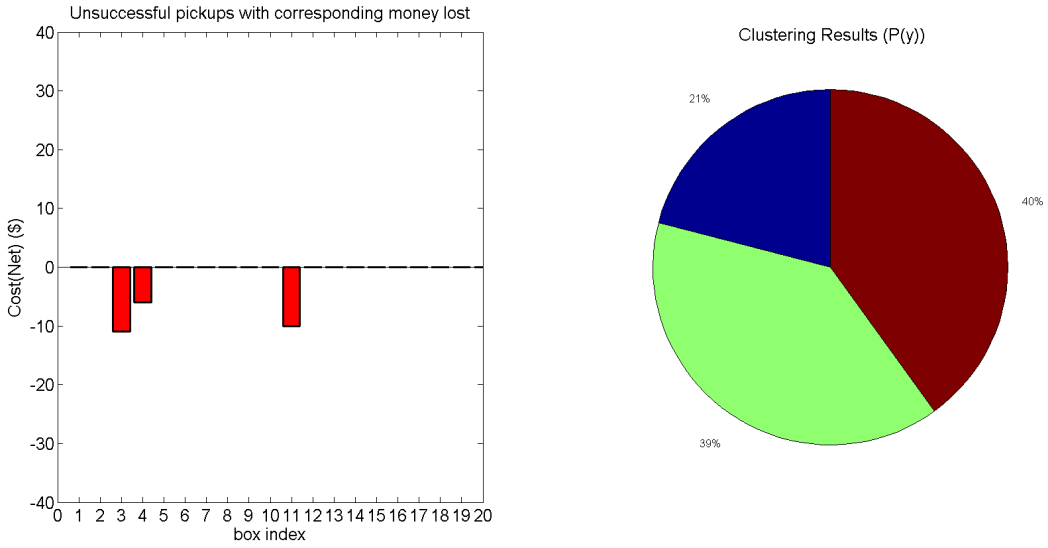

Figure 3.2: (P1) The plot on the left shows the losses due the the boxes not serviced. The negative values indicate that they are contributing towards depot's loss. Three boxes go without service and their corresponding costs are depicted. The pie chart on the right shows the mass associated with each cluster for the three clusters.

datasets with 20 boxes and the bus arrival times (3) were computed using Lloyd's algorithm. DA method on average took relatively greater computational time but Lloyd algorithm was more prone to getting trapped in local minima. Since there is no annealing aspect in Lloyd's algorithm, the algorithm is highly sensitive to initial guess and fails to avoid local minima in its implementation.

3.4.2 P2: Capacity constraints

For this problem, we provide capacity to each bus and the algorithm should attempt to match these capacities. We have again chosen the time windows to lie within $[0, 350]$ minutes. We have 100 boxes and 10 buses for pick-up or delivery. The capacities assigned to each bus (cluster) are $\{3, 5, 9, 9, 9, 10, 13, 13, 13, 16\}$. Subjected to these capacities, the algorithm works well and 30% of boxes end up without being serviced. One can see the effectiveness of DA method to cluster data subjected to capacity constraints.

In Fig: 3.3, the blue lines depict the bus arrival times and the red bars indicate the time windows. In Fig: 3.4, the plot shows the corresponding costs for the boxes which were not serviced. Among the 100 boxes, 30 boxes were left without service. The loss incurred by the depot is \$596.22 which is 32% of the total cost of all the boxes($\$1,840.2$). The pie chart on the right shows the mass associated with each cluster. We can see that the masses associated with each cluster are in proportion with the capacity values given as the constraints.

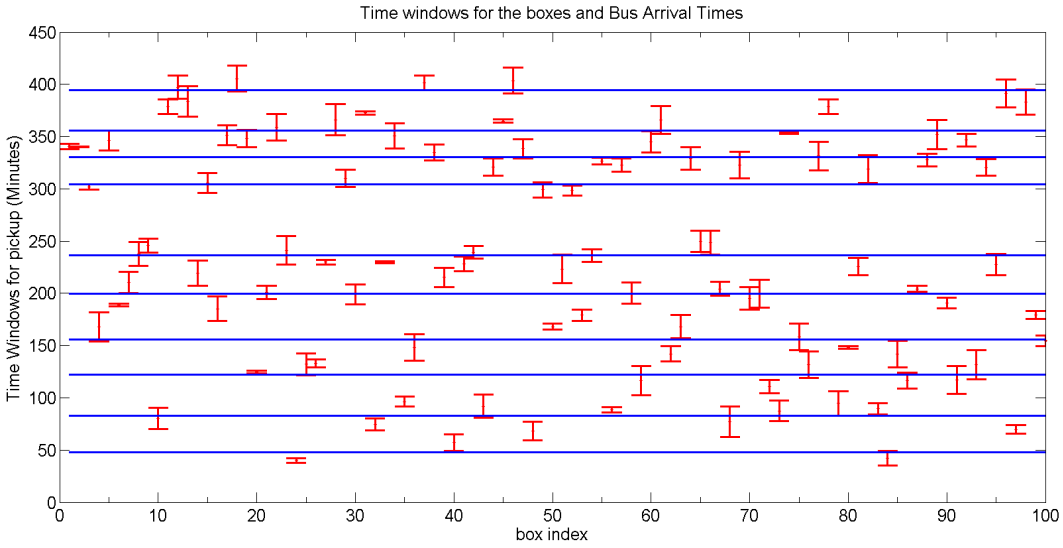

Figure 3.3: (P2)Time Windows for 100 Boxes (Red Intervals) and the Arrival Times (Cluster Locations (blue lines)) for 10 buses.

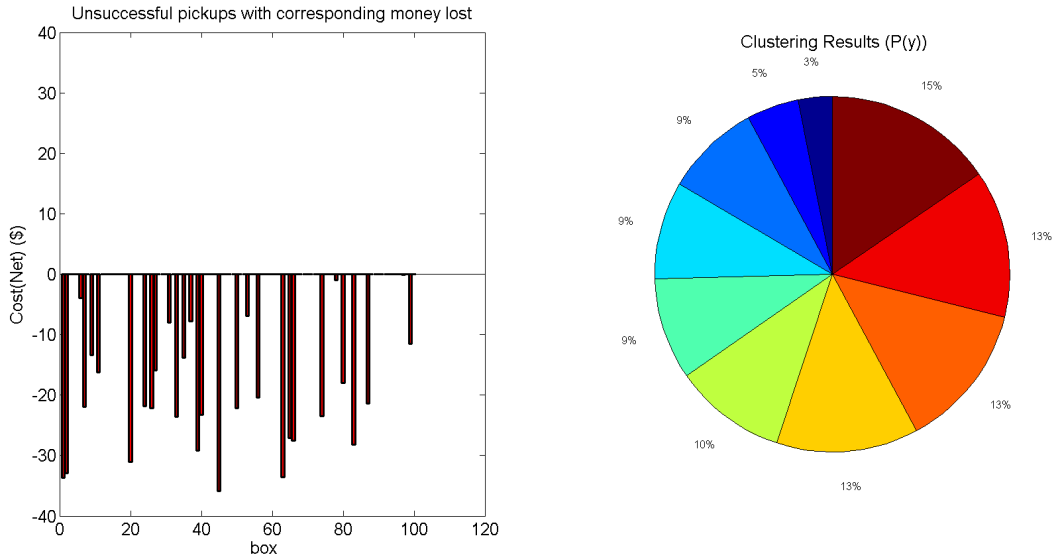

Figure 3.4: (P2) The plot on the left shows the losses due the the boxes not serviced. The negative values indicate that they are contributing towards depot's loss. The pie chart on the right shows the mass associated with each cluster. One can see that they are in proportion to the capacity values given as constraints.

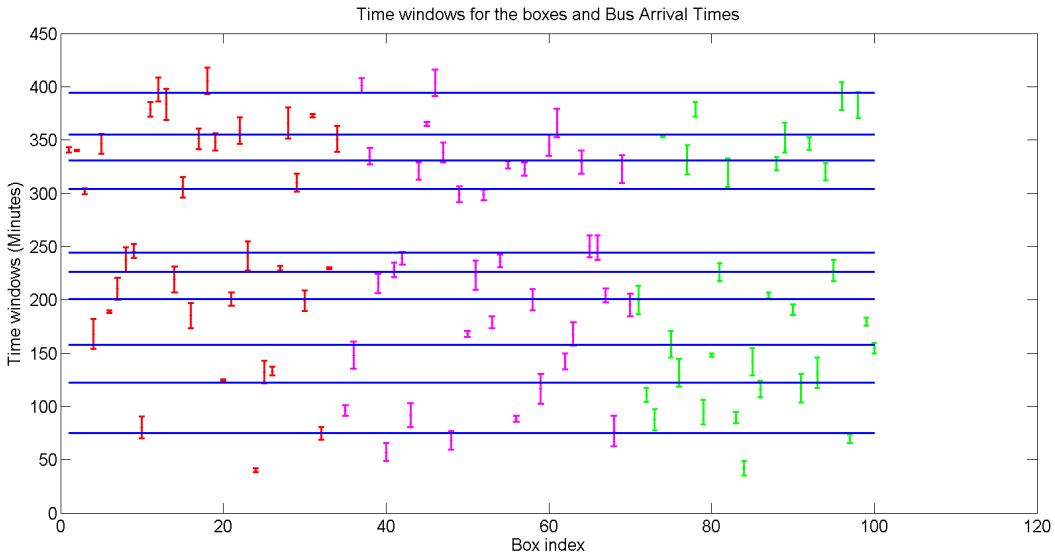

Figure 3.5: (P3)Time Windows for 100 Boxes (3 types) and the Arrival Times (Cluster Locations (blue lines)) for 10 buses. The three colors (red, magenta and green) are used indicate the type of boxes.

3.4.3 P3: Multiple Types Capacity Constraints

For this problem, we have capacities associated with each bus and all types of boxes. In the simulation, we have chosen 100 boxes in total, 3 types of boxes and 10 buses for pick-up or delivery. There are 34 boxes of type 1 (depicted by red color), 36 boxes of type 2 (magenta) and 30 boxes of type 3 (green). The algorithm should attempt to match these capacities. We have again chosen the time windows to lie within $[0, 350]$ minutes. The capacities were randomly chosen and subjected to these capacities, the algorithm works well. In Fig: 3.5, the blue lines depict the bus arrival times and the red, magenta and green bars indicate the time windows for the three types of boxes respectively. Out of 100 boxes, 71 boxes are picked up and the total losses amounts to \$537.43 which is 29.8% of the total costs of all the boxes ($\$1,804.2$).

In Fig: 3.6, the plot shows the total number of boxes picked up by each bus. The colors represent the type of boxes as mentioned earlier. In Fig: 3.7, the plot on the left shows the capacity constraints ($\boldsymbol{W}$) for each type of box and bus. The plot on the right shows the clustered mass information ($\boldsymbol{\lambda}$). One can see that the shape of the plots match each other. Also, the numerical results found that the ratios of the capacity and clustered mass were equal for all the buses, i.e for all types of boxes ($1 \leq l \leq p$), we found $(\frac{W_{j_1 l}}{\lambda_{j_1 l}} = \frac{W_{j_2 l}}{\lambda_{j_2 l}}, 1 \leq j_1, j_2 \leq K)$.

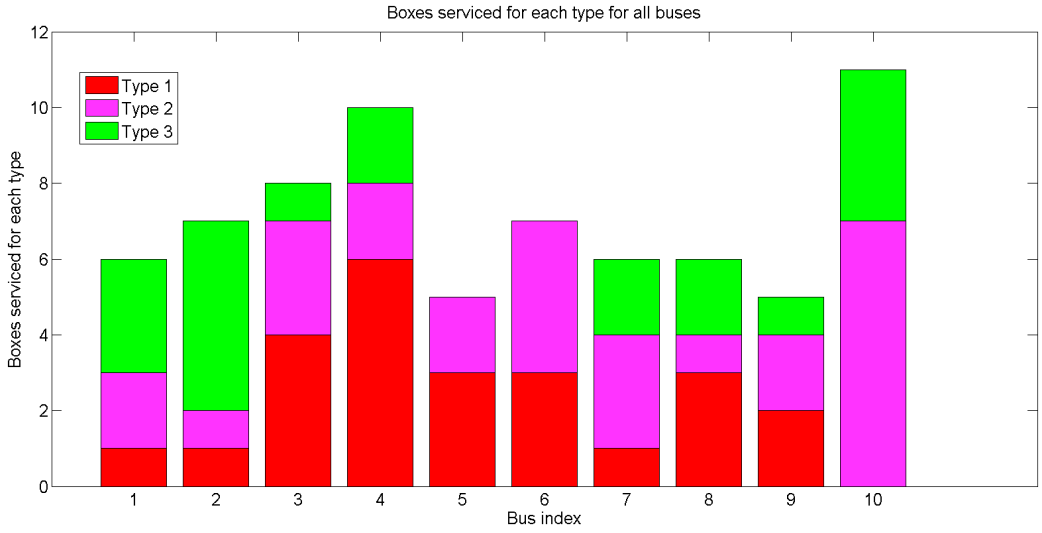

Figure 3.6: (P3) The plot shows the number of boxes serviced by each bus for each type. The three colors indicate the number of boxes serviced for each type (red, magenta and green).

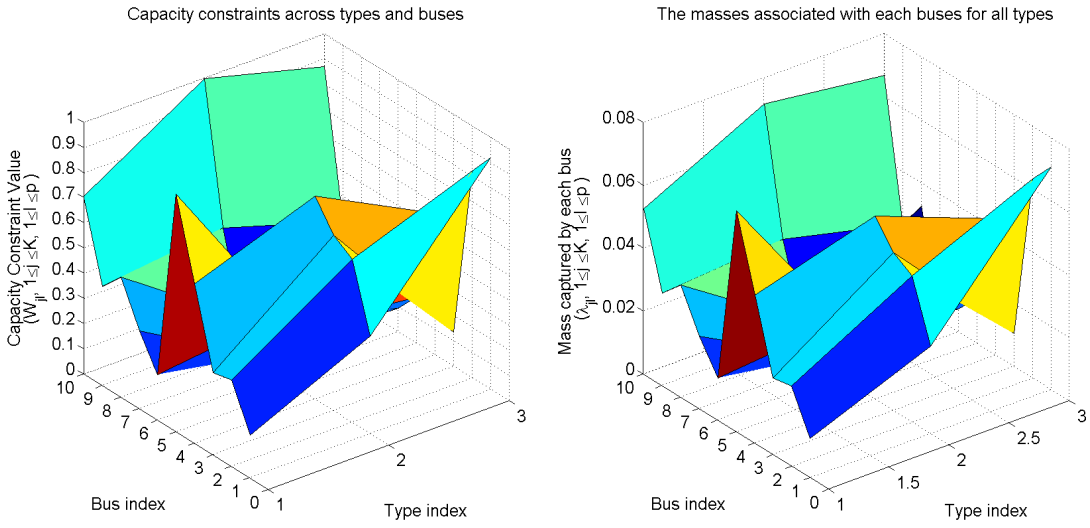

Figure 3.7: (P3) The plot on the left shows capacity constraint value for each type of box and bus. The plot on the right shows the mass associated with each bus and each type of box. We can see that the shapes indicate that the masses are in proportion with the capacities provided as constraints.

Chapter 4

DA: Computational Aspects

4.1 Complexity Analysis

[1] In this section, we present some mild worst case estimates of the previously presented DA algorithm's computational complexity under some simplifying assumptions. The analysis assumes that N and $K_{\max}$ are fixed and the training points are assumed to be $d-$dimensional. It can be shown that the gradient of F with respect to $\boldsymbol{y}_j$ is as follows:

$$\frac{\partial F}{\partial \boldsymbol{y}_j} = 2 \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) p(\boldsymbol{y}_j|\boldsymbol{x})(\boldsymbol{y}_j - \boldsymbol{x}), \tag{4.1.1}$$

which gives

$$\boldsymbol{y}_j^{n+1} = \boldsymbol{y}_j^n - \frac{1}{2\boldsymbol{p}(\boldsymbol{y}_j^n)} \frac{\partial F}{\partial \boldsymbol{y}_j}. \tag{4.1.2}$$

In eq. (4.1.2), n represents the iteration count in Step 4) and $p(\boldsymbol{y}_j^n)$ is given by eq. (2.1.8). The iterates are computed using eq. (4.1.2), which corresponds to a descent method with descent direction, $\boldsymbol{d}_n = -\boldsymbol{p}(\boldsymbol{y}_j^n)^{-1} \frac{\partial F}{\partial \boldsymbol{y}_j}$. Clearly, $\boldsymbol{d}_n^T \frac{\partial F}{\partial \boldsymbol{y}_j} \leq 0$ with equality being true when $\frac{\partial F}{\partial \boldsymbol{y}_j} = 0$ and, hence, Step 4) converges [13].

We are now ready to examine the number of steps of the DA algorithm required for clustering in terms of basic operations (additions, subtractions, multiplications), in each case giving one floating point operation, *flop*[2]. We also count divisions as four flops [7].

Step 1: No cost is associated with this step.

Step 2: The computation of $\boldsymbol{y}_1$ for N training vectors requires Nd multiplications for the formation of the products $\boldsymbol{x}p(\boldsymbol{x})$ and $(N-1)d$ additions for the evaluation of the sum. Therefore, the total cost of this step is $(2N-1)d$ flops.

Step 3: Assume that we fix the temperature and that the number of available codevectors is $K < K_{\max}$. For the formation of the $\boldsymbol{y}_j'$s, we require $2Kd$ additions. For the evaluation of the corresponding probabilities,

[1] This chapter relies on the contents in [25] which is an IEEE copyright owned article. Permission has been taken from IEEE and authors (P. M. Parekh, D. Katselis, C. Beck, S. Salapaka) to include sections in the thesis. The article is accepted for Proceedings of American Control Conference and publication is due in July 2015.

[2] In this paper, 1 flop= 1 basic operation, although the meaning of a flop is slightly different in computer architercture.

we need $2K$ divisions. Therefore, the total cost is $2K(d+4)$ flops. Moreover, note that when we are not at a phase transition temperature, the duplicated codevectors will be merged together in Step 4). The total cost with respect to the evolution of K is discussed at the end of this section.

Step 4: Fix the iteration index. For the computation of $\boldsymbol{y}_i$ we require N multiplications for the formation of the products $p(\boldsymbol{x})p(\boldsymbol{y}_i|\boldsymbol{x})$, Nd multiplications for the formation of $p(\boldsymbol{x})p(\boldsymbol{y}_i|\boldsymbol{x})\boldsymbol{x}$, $(N-1)d$ additions for the computation of the sum and a division for the final computation of $\boldsymbol{y}_i$. This computation is performed for $2K$ codevectors yielding a total cost of $2K(N+2Nd-d+4)$ flops. We now fix $\boldsymbol{x}$. For $p(\boldsymbol{y}_i|\boldsymbol{x})$, note that the numerator is one of the terms appearing in the denominator. Each individual term can be stored to accelerate the code. We therefore focus on the denominator. For each exponent, we require d subtractions, d multiplications, $d-1$ additions and 1 division, i.e., $3d+3$ flops. The evaluation of an exponential requires 8 flops [7]. Thus, the total cost per exponential becomes $24(d+1)$ flops. Multiplication of the exponential with $p(\boldsymbol{y}_i)$ adds a flop. Hence, for each summand of the denominator we require $24d+25$ flops. Moreover, we have $2K$ summands in the denominator and for each $\boldsymbol{y}_i$ and fixed $\boldsymbol{x}$, the denominator remains the same with the numerator being one of the summands in the denominator. Hence, the computation of all summands appearing in the denominator requires $2K(24d+25)$ flops. Also, for computing all the values related to $p(\boldsymbol{y}_i|\boldsymbol{x})$, we require $2K-1$ additions. Therefore, the computation of the denominator of each $p(\boldsymbol{y}_i|\boldsymbol{x})$ for a fixed $\boldsymbol{x}$ requires $(2K(24d+25)+2K-1)=(48Kd+52K-1)$ flops. Now with all the terms available, we just need $2K$ divisions to calculate all $p(\boldsymbol{y}_i|\boldsymbol{x})$'s for a fixed $\boldsymbol{x}$, which amounts to $8K$ more flops. Hence, for all $p(\boldsymbol{y}_i|\boldsymbol{x})$'s with a fixed $\boldsymbol{x}$, we require $(48Kd+60K-1)$ flops. Then, for all $\boldsymbol{x}$, we have to repeat the same calculations N times, which amounts to $(48NKd+60NK-N)$ flops.

Finally, for each $p(\boldsymbol{y}_i)$ we require N multiplications and $N-1$ additions, i.e., $2N-1$ flops, leading to $4NK-2K$ flops for all $2K$ codevectors. Combining all computations, we have $(48NKd+60NK-N)+(2K(N+2Nd-d+4))+(4NK-2K)=(52NKd+66NK+6K-N-2Kd)$ flops.

Therefore, Step 4) requires $n_{\max}(52NKd+66NK+6K-N-2Kd)$ flops in the worst case.

Step 5: In the worst case, $n_{\max}$ tests will be performed for testing convergence. It can be seen that for the calculation of objective function, the number of flops required are proportional to N. Hence, the number of flops required for this step is proportional to $n_{\max}N$.

Step 6: No cost is associated with this step.

Step 7: Assume that the initial temperature is chosen to be $2\lambda_{\max}(\boldsymbol{C}_x)+\delta$ for some small positive constant δ. The total number of temperature values is such that $\rho^m(2\lambda_{\max}(\boldsymbol{C}_x)+\delta)\leq T_{min}$ yielding $m\geq M=\lceil \ln\left(\frac{T_{\min}}{2\lambda_{\max}(\boldsymbol{C}_x)+\delta}\right)/\ln\rho\rceil$. Therefore, the cost of this step is M flops.

Step 8: No cost is associated with this step.

To finish the analysis, we have to take into account the annealing process in all steps and the evolution of K in Steps 3), 4) and 5). To this end, we assume that $K_{\max}$ is achievable within our temperature schedule. Suppose that the sequence of critical temperature values is $T_1^c, T_2^c, T_3^c, \ldots$. Then for all intermediate temperature values in $(T_1^c, 2\lambda_{\max}(\boldsymbol{C}_x)+\delta]$ the value of K is always 1, for all intermediate temperature values in $(T_2^c, T_1^c]$ the value of K is always 2 etc, due to the merging of codevectors is Step 4). If σ_1 is the number of temperature values in $(T_1^c, 2\lambda_{\max}(\boldsymbol{C}_x) + \delta]$, σ_2 is the number of temperature values in $(T_2^c, T_1^c]$ etc, we have a sequence of cardinalities $\sigma_1, \sigma_2, \ldots, \sigma_{K_{\max}}, \sigma_{K_{\max}+1}$, where $\sigma_{K_{\max}+1}$ is the number of temperature values remaining till the end of the annealing process for which $K = K_{\max}$. Note that this is the case no matter if we have more critical temperatures after $T_{K_{\max}}$ since Step 3) is no longer executed. Clearly, σ_i is an increasing function of ρ for all i. With these definitions, we now focus on Steps 3), 4) and 5).

Step 3: It is easy to verify that the total cost of this step is $2(d+4)\sum_{j=1}^{K_{\max}-1} j\sigma_j$.

Step 4: The cost of this step can be easily seen to be:

$$\begin{aligned}&\sum_{j=1}^{K_{\max}} \sigma_j n_{\max}(52Njd + 66Nj + 6j - N - 2jd) + \\ &\sigma_{K_{\max}+1} n_{\max}(52NK_{\max}d + 66NK_{\max} \\ &+ 6K_{\max} - N - 2K_{\max}d).\end{aligned} \tag{4.1.3}$$

Step 5: The cost of this step turns out to be

$$\sum_{j=1}^{K_{\max}} 4\sigma_j n_{\max} j(3d-1) + 4\sigma_{K_{\max}+1} n_{\max} K_{\max}(3d-1). \tag{4.1.4}$$

Summing the flops for individual steps leads to a worst case upper estimate of the investigated computational complexity. Setting $\sigma_{\max} = \{\sigma_1, \sigma_2, \ldots, \sigma_{K_{\max+1}}\}$, it can be easily seen that the worst case complexity behaves as $O\left(\sigma_{\max} n_{\max} N K_{\max}^2 d\right)$.

Remark: Ideas for the reduction of the DA's computational complexity have been proposed in [7]. The key points in the analysis therein is to use a small number of neighboring codevectors for the update of each codevector in Step 4) and to replace the Gibbs distribution by some simpler, fuzzy-like membership function that is more easily computable and sufficiently accurate at the same time.

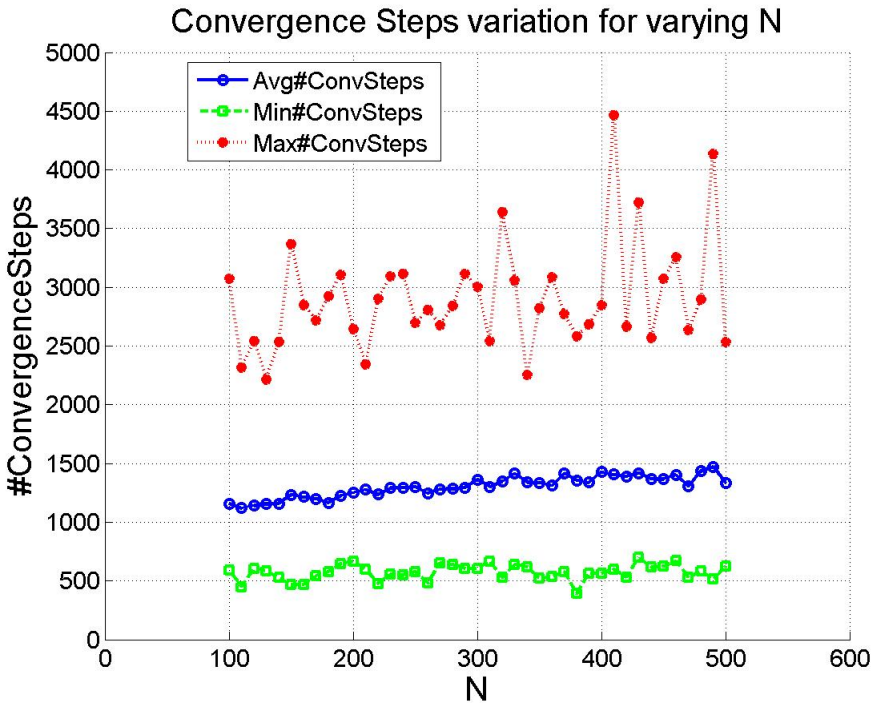

Figure 4.1: Minimum, average and maximum numbers of iterations required for the convergence of Step 4) in the DA algorithm vs the number of data points.

4.2 Simulations

In Fig. 4.1, the minimum, average and maximum numbers of iterations required for the convergence of Step 4) in the DA algorithm versus the number of data points is demonstrated. The comparison is performed over 100 realizations of the data points. N is varied in the range $[100, 500]$ and $K_{\max} = 10$ is kept constant. We observe that the average and minimum numbers of iterations do not significantly change with an increase in N. The maximum number of iterations or the worst case number of iterations to convergence among the 100 runs show a very slight linear increase with an increasing N.

Fig. 4.2 demonstrates the same numbers as Fig. 4.1 with the difference that N is now kept constant at 300, while $K_{\max}$ (maximum number of codevectors) is varied in the range $[5, 25]$. Again 100 different data sets are employed. We observe that all curves show an increasing trend with an increasing $K_{\max}$. Clearly, the slopes in all cases are higher than the corresponding slopes in Fig. 4.1. This means that the number of iterations for Step 4) is more sensitive to an increase in $K_{\max}$ than in N.

To check the exact behavior of the curves in Fig. 4.2 with respect to large orders of magnitude for $K_{\max}$, in Fig. 4.3 $K_{\max}$ is largely varied in the range [1, 150]. In this plot, N is kept constant to 10000. Clearly, the numbers of iterations are highly dependent on $K_{\max}$.

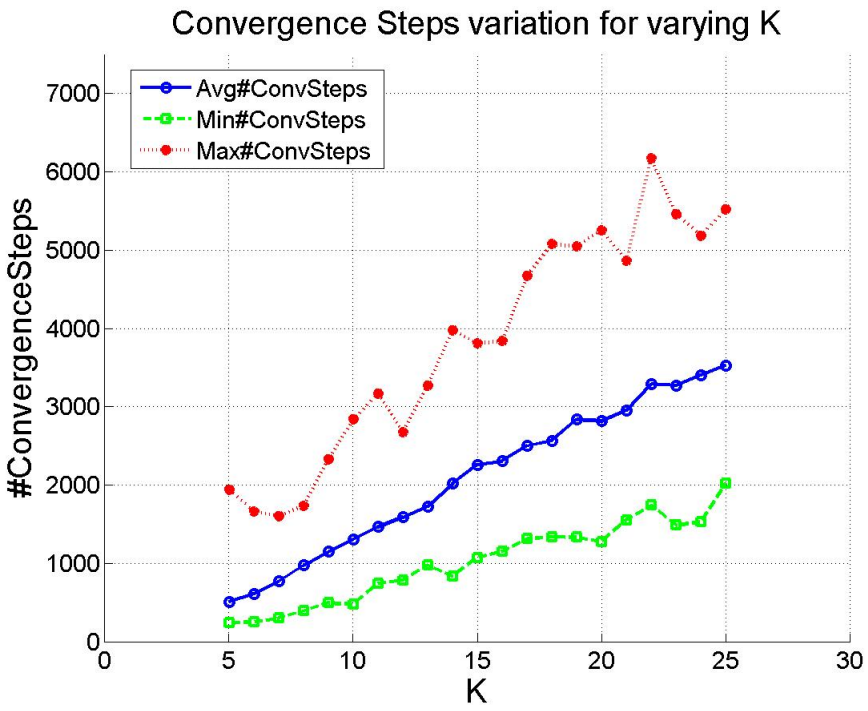

Figure 4.2: Minimum, average and maximum numbers of iterations required for the convergence of Step 4) in the DA algorithm vs the allowed number of codevectors $K_{\max}$.

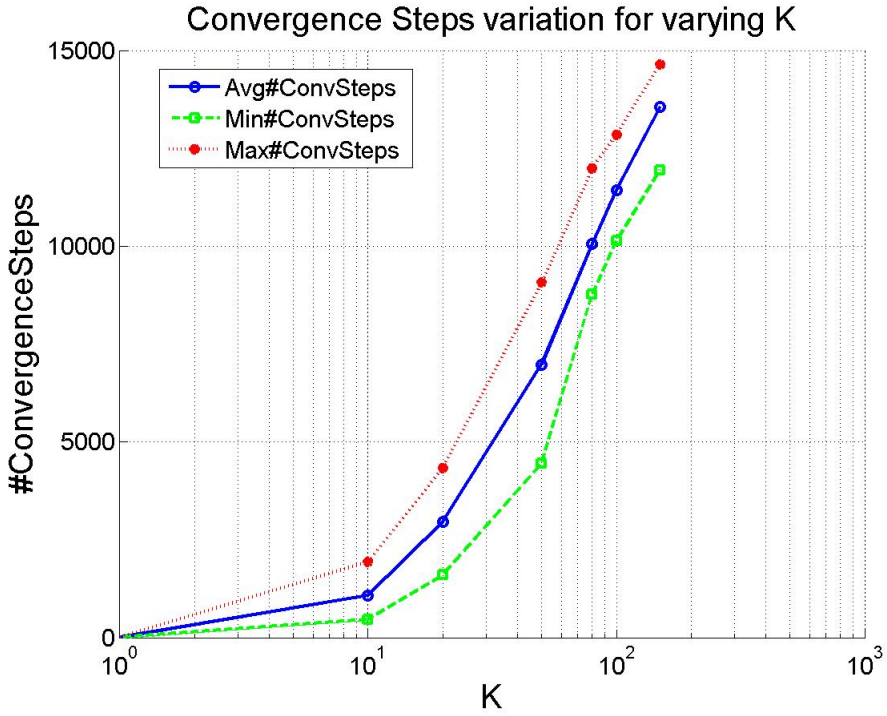

Figure 4.3: Minimum, average and maximum numbers of iterations required for the convergence of Step 4) in the DA algorithm vs the allowed number of codevectors $K_{\max}$.

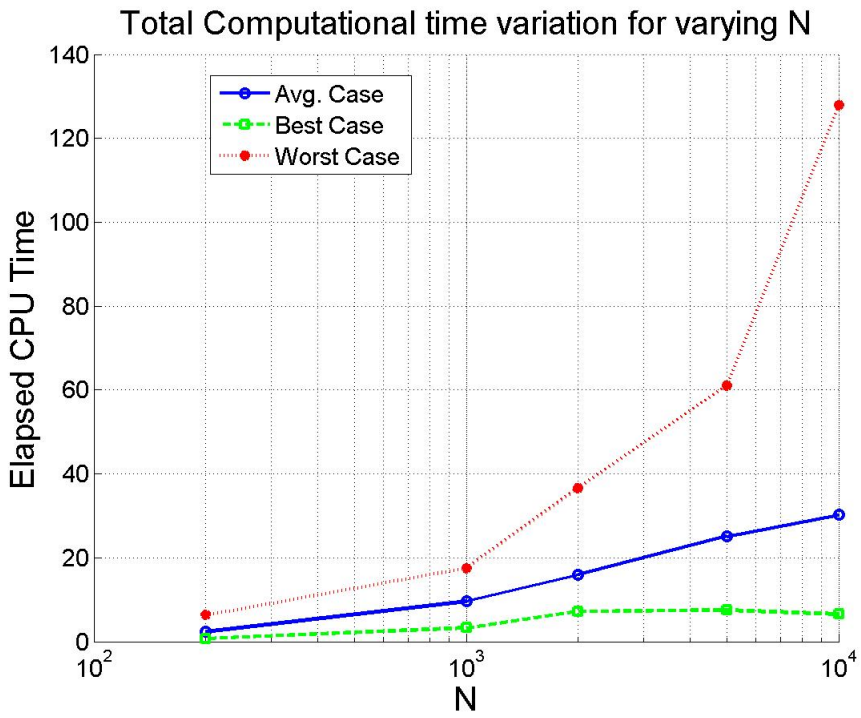

Figure 4.4: $K_{\max} = 10$: Minimum, average and maximum time required for the DA algorithm vs the number of data points N.

Fig. 4.4 demonstrates the minimum, maximum and average execution times of the DA algorithm when N is allowed to vary in the range $[200, 10000]$ and $K_{\max}$ is kept fixed to the value 10. To this end, the Matlab function CPUtime was used. As is demonstrated, the curves show a steady increase.

Finally, in Fig. 4.5 we fix N to 10000 and we vary $K_{\max}$ demonstrating the same curves as in Fig. 4.4. As $K_{\max}$ increases, the computational times show a rapid increase. Furthermore, comparing Figs. 4.4 and 4.5 we observe that the execution times are much more sensitive to an increase in $K_{\max}$ than in N.

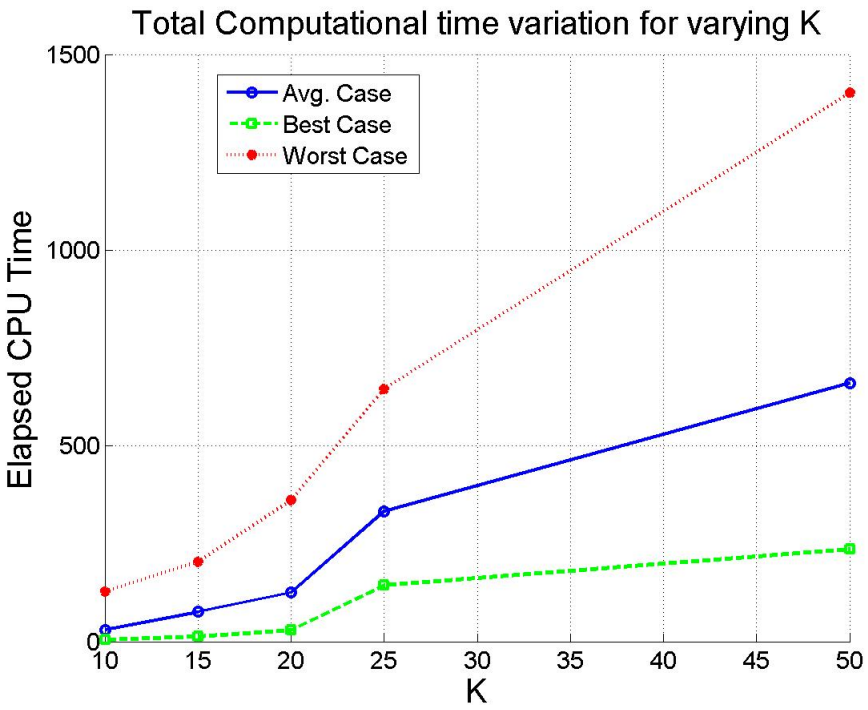

Figure 4.5: $N = 10000$: Minimum, average and maximum time required for the DA algorithm vs $K_{\max}$.

Chapter 5

Conclusion and Future Work

In Chapter (2), tutorial was presented for DA algorithm for clustering. The algorithm steps and convergence criteria were discussed for the mass constrained implementation of DA method. Also, several connections with statistical physics and rate-distortion theory were discussed. In Chapter (3), the DA method was applied to pickup and delivery problem with time windows. For problem (P1), the DA algorithm was compared with Lloyd's algorithm. The DA implementation takes relatively larger computational time but is less sensitive to the initial guess when compared to Lloyd's algorithm. The Lloyd's algorithm fails to avoid local minima and the final solution is immensely affected by the chosen initial guess of cluster locations. The application to the variants of the problem with capacity constraints and multiple types capacity constraints was also discussed. The DA method performed very well with these constraints and gave optimal solutions for all the test cases. Upon applying the DA algorithm to these problems (P1, P2, P3), we can see that in addition to avoiding local optimal solutions through annealing framework, DA algorithm also has the flexibility to solve clustering problems subjected to capacity constraints.

In Chapter (4), upper bounds on the computational complexity of the algorithm were derived under mild assumptions about the data sets. The presented theoretical aspects were accompanied by a numerical study of the behavior of the algorithm to complete the treatment. The most computationally expensive step is Step 4)[1] of the algorithm and number of iterations of this step is more sensitive to increase in K_{max} than in N. The algorithm can be improved by ensuring faster convergence in this step.

Future work will aim at the determination of tighter computational complexity characterizations for the DA algorithm. With regard to pickup and delivery problem, the DA method with modified distance function can be formulated to incorporate the aspect of time windows in a more appropriate way. For the current setting, the mid point of the time windows is considered as the source information. A different distance function can be designed such that the distance of cluster location from any point of time within the window remains the same. In relation to the problem with capacity constraints, we can reformulate the Lagrangian to include the capacity constraints as masses associated with each cluster. Future work can be aimed at

[1]This step involves getting the association probabilities and cluster locations from the implicit equations(2.1.6, 2.1.7, 2.1.8) until convergence is reached which is tested in Step 5)

improving these aspects of DA implementation to the given problem.

References

[1] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210–2239, Nov. 1998.

[2] D. Bertsimas, J. Tsitsiklis, "Simulated Annealing", *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.

[3] B. Hajek, "A Tutorial Survey of Theory and Applications of Simulated Annealing", *Proceedings of 24th Conf. on Decision and Control*, 755–760, NY, 1985.

[4] H. H. Szu, R. L. Hartley, "Nonconvex Optimization by Fast Simulated Annealing", *Proceedings of the IEEE*, vol. 75, no. 11, pp. 1538–1540, Nov. 1987.

[5] S. Geman, D. Geman "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[6] B. Hajek, "Cooling Schedules for Optimal Annealing", *Math. Oper. Res.*, (1988)13: 311–329, .

[7] K. Demirciler, A. Ortega, "Reduced-Complexity Deterministic Annealing for Vector Quantizer Design", *EURASIP Journal on Applied Sig. Proc.*, 2005:12, 1807–1820.

[8] K. Rose, "A Mapping Approach to Rate-Distortion Computation and Analysis", *IEEE Trans. on Inf. Theory*, vol. 40, no. 6, pp. 1939–1952, Nov. 1994.

[9] P. Sharma, S. M. Salapaka, C. L. Beck, "Entropy-Based Framework for Dynamic Coverage and Clustering Problems", *IEEE Trans. on Automatic Control*, vol. 57, no. 1, pp. 135–150, Jan. 2012.

[10] Y. Xu, S. M. Salapaka, C. L. Beck, "Aggregation of Graph Models and Markov Chains by Deterministic Annealing", *IEEE Trans. on Automatic Control*, vol. 59, no. 10, pp. 2807–2812, Oct. 2014.

[11] Y. Xu, S. M. Salapaka, C. L. Beck, "Clustering and Coverage Control for Systems With Acceleration-Driven Dynamics", *IEEE Trans. on Automatic Control*, vol. 59, no. 5, pp. 1342–1347, May 2014.

[12] P. Sharma, S. M. Salapaka, C. L. Beck, "A Scalable Approach to Combinatorial Library Design for Drug Discovery", *J. Chem. Inf. Model.*, vol. 48, no. 1, pp.27–41, 2008.

[13] S. M. Salapaka, A. Khalak, M. A. Dahleh. "Constraints on locational optimization problems", *Proceedings of 42th Conf. on Decision and Control*, vol. 2, pp. 1741–1746, 2003.

[14] T. M. Cover and J. A. Thomas *Elements of Information Theory*, John Wiley & Sons, Inc. 1991.

[15] E. T. Jaynes, "Information Theory and Statistical Mechanics", *Physical Review*, vol. 108, no. 2, pp. 171–190, Oct. 1957.

[16] S. Arimoto, "An Algorithm for Calculating the Capacity of an Arbitrary Discrete Memoryless Channel", *IEEE Trans. on Inf. Theory*, vol. IT-18, pp. 14–20, Jan. 1972.

[17] R. E. Blahut, "Computation of Channel Capacity and Rate-Distorion Functions", *IEEE Trans. on Inf. Theory*, vol. IT-18, pp. 460–473, July 1972.

[18] J. Cortés, S. Martinez, T. Karatas, F. Bullo, "Coverage Control for Mobile Sensing Networks", *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Feb. 2004.

[19] E. Frazzoli, F. Bullo, "Decentralized Algorithms for Vehicle Routing in a Stochastic Time-Varying Environment", *proceedings of Conf. on Decision and Control*, 2004, pp. 3357–3363.

[20] N. V. Kale, S. M. Salapaka, "MaximumV Entropy Principle-Based Algorithm for Simultaneous Resource Location and Multihop Routing in Multiagent Networks", *IEEE Trans. on Mobile Computing*, vol. 11, no. 4, pp. 591–602, Apr. 2012.

[21] W. Krauth, *Statistical Mechanics: Algorithms and Computations*, Oxford University Press, 2006.

[22] D. Arovas, *Lecture Notes on Thermodynamics and Statistical Mechanics*, available at: http://www-physics.ucsd.edu/students/courses/spring2010/physics210a/ LECTURES/210_ COURSE.pdf.

[23] C. Byrne, "Iterative Projection onto Convex Sets using Multiple Bregman Distances", *Inverse Problems*, 15(1999): 1295–1313.

[24] G. B. Bagci, "Some remarks on Rényi relative entropy in a thermostatistical framework", arXiv:cond-mat/0703008v2.

[25] ©[2015] IEEE. Reprinted, with permission, from [P. M. Parekh, D. Katselis, C. Beck, S. Salapaka, "Deterministic Annealing for Clustering: Tutorial and Computational Aspects", Proceedings of American Control Conference, July 2015]