2015 American Control Conference
Palmer House Hilton
July 1-3, 2015. Chicago, IL, USA

Deterministic Annealing for Clustering: Tutorial and Computational Aspects

Pratik M. Parekh, Dimitrios Katselis, Carolyn L. Beck, Srinivasa M. Salapaka

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. In this paper, we first present some background material on the DA method and its connections to statistical physics and rate-distortion theory. A computational complexity analysis is then 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.**

I. Introduction

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 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.

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],

P. M. Parekh is with the Coordinated Science Laboratory and the Mechanical Engineering Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801-2925, E-mail: pparekh2@illinois.edu.

D. Katselis and C. L. Beck are with the Coordinated Science Laboratory and the Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801-2925, Emails: {katselis—beck3}@illinois.edu.

S. M. Salapaka is with the Mechanical Engineering Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801-2925, E-mail: salapaka2@illinois.edu.

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 this paper, we first review the basic principles of the DA algorithm and its connections with statistical mechanics and rate-distortion theory. We leverage a computational analysis 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.

II. DA: The algorithm

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. A distortion mapping $d : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ quantifies the cost of associating an input vector $\boldsymbol{x}$ with a specific output vector $\boldsymbol{y}$. 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[1]. In the DA framework, it can easily be shown that the optimal association probabilities are given by the Gibbs distribution [1]:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \exp(-d(\boldsymbol{x},\boldsymbol{y})/T)/Z_{\boldsymbol{x}}, \tag{1}$$

where T is the temperature and $Z_{\boldsymbol{x}} = \sum_{\boldsymbol{x}\in\mathcal{X}} \exp(-d(\boldsymbol{x},\boldsymbol{y})/T)$ is a normalizing constant, also known as the *partition function* in statistical physics [15]. Moreover, for the squared error distortion measure the optimal locations of the centroids can be expressed as [1]

$$\boldsymbol{y} = \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}|\boldsymbol{y})\boldsymbol{x} = \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})\boldsymbol{x}/p(\boldsymbol{y}). \tag{2}$$

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

Eq. (1) and (2) imply that we can construct an alternating-minimization algorithm as follows: Given a codebook $\mathcal{Y}$ we can use (1) to optimize the association probabilities and with these probabilities we can update the centroid locations based on (2). Iterating over these two steps till convergence yields the *DA algorithm for clustering*, which is summarized in the following:

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 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:

$$\boldsymbol{y}_i = \sum_{\boldsymbol{x}} p(\boldsymbol{x})p(\boldsymbol{y}_i|\boldsymbol{x})\boldsymbol{x}/p(\boldsymbol{y}_i) \tag{3}$$

$$p(\boldsymbol{y}_i|\boldsymbol{x}) = \frac{p(\boldsymbol{y}_i)\exp(-\|\boldsymbol{x}-\boldsymbol{y}_i\|^2/T)}{\sum_{j=1}^{2K} p(\boldsymbol{y}_j)\exp(-\|\boldsymbol{x}-\boldsymbol{y}_j\|^2/T)} \tag{4}$$

$$p(\boldsymbol{y}_i) = \sum_{\boldsymbol{x}} p(\boldsymbol{x})p(\boldsymbol{y}_i|\boldsymbol{x}). \tag{5}$$

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 objective function and $\alpha < 1$ 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 is a variant of the mass-constrained implementation of the DA in [1]. Step 7) 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.

III. 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]:

$$R(D) = \min_{q(\boldsymbol{y}|\boldsymbol{x}): \sum_{\boldsymbol{x},\boldsymbol{y}} p(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})d(\boldsymbol{x},\boldsymbol{y}) \leq D} I(\boldsymbol{X}; \boldsymbol{Y}). \tag{6}$$

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]:

$$R(D) = \min_{q \in \mathcal{B}} \min_{p \in \mathcal{A}} D(p \parallel q). \tag{7}$$

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

$$q(\boldsymbol{y}|\boldsymbol{x}) = \frac{r(\boldsymbol{y}) \exp(-\mu d(\boldsymbol{x}, \boldsymbol{y}))}{\sum_{\boldsymbol{y}} r(\boldsymbol{y}) \exp(-\mu d(\boldsymbol{x}, \boldsymbol{y}))}, \tag{8}$$

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]

$$D(\pi(\boldsymbol{x}) \parallel w(\boldsymbol{x})) = \beta(F_\pi - F_w), \tag{9}$$

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.

IV. DA: Computational Aspects

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:

$$\frac{\partial F}{\partial \boldsymbol{y}_j} = 2\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})p(\boldsymbol{y}_j|\boldsymbol{x})(\boldsymbol{y}_j - \boldsymbol{x}), \tag{10}$$

which gives

$$\boldsymbol{y}_j^{n+1} = \boldsymbol{y}_j^n - \frac{1}{2\boldsymbol{p(y_j^n)}}\frac{\partial F}{\partial \boldsymbol{y}_j}. \tag{11}$$

In eq. (11), n represents the iteration count in Step 4) and $p(\boldsymbol{y}_j^n)$ is given by eq. (5). The iterates are computed using eq. (11), which corresponds to a descent method with descent direction, $\boldsymbol{d}_n = -\boldsymbol{p(y_j^n)}^{-1}\frac{\partial F}{\partial \boldsymbol{y}_j}$. 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, 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]In this paper, 1 flop= 1 basic operation, although the meaning of a flop is slightly different in computer architercture.

$2(d+4)\sum_{j=1}^{K_{\max}-1} j\sigma_j$.

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

$$\sum_{j=1}^{K_{\max}} \sigma_j n_{\max}(52Njd + 66Nj + 6j - N - 2jd) + \sigma_{K_{\max}+1} n_{\max}(52NK_{\max}d + 66NK_{\max} + 6K_{\max} - N - 2K_{\max}d). \quad (12)$$

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

$$\sum_{j=1}^{K_{\max}} 4\sigma_j n_{\max} j(3d-1) + 4\sigma_{K_{\max}+1} n_{\max} K_{\max}(3d-1). \quad (13)$$

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.

V. Simulations

In this section we perform a numerical study of the DA algorithm. The data points and the codevectors are assumed to be in $\mathbb{R}^2$. We choose to implement a convergence test in step 5) based on the difference between successive values of the free energy falling below a predefined threshold, which is assumed to be 10^{-7}. The temperature schedule is initialized to 1.25×10^4 and $T_{\min}$ is set to 0.25 in all simulations. The cooling factor ρ assumes the value 0.91. For all plots, the DA algorithm is executed on multiple data sets.

In Fig. 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. 2 demonstrates the same numbers as Fig. 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. 1. This means that the number of iterations for Step 4) is more sensitive to an increase in $K_{\max}$ than in N.

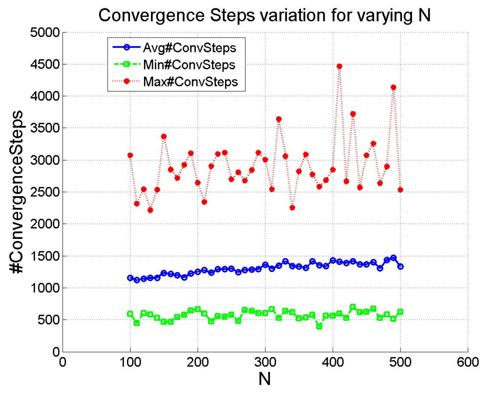

Fig. 1. Minimum, average and maximum numbers of iterations required for the convergence of Step 4) in the DA algorithm vs the number of data points.

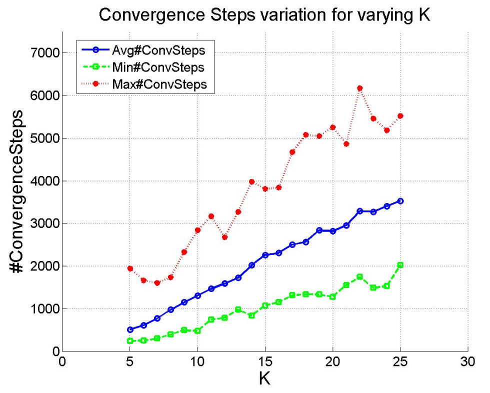

Fig. 2. Minimum, average and maximum numbers of iterations required for the convergence of Step 4) in the DA algorithm vs the allowed number of codevectors $K_{\max}$.

To check the exact behavior of the curves in Fig. 2 with respect to large orders of magnitude for $K_{\max}$, in Fig. 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 $K_{\max}$.

Fig. 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. 5 we fix N to 10000 and we vary $K_{\max}$ demonstrating the same curves as in Fig. 4. As $K_{\max}$ increases, the computational times show a rapid increase. Furthermore, comparing Figs. 4 and 5 we observe that the execution times are much more sensitive to an increase in $K_{\max}$ than in N.

VI. Conclusions

In this paper, the DA algorithm for clustering was revisited. Several connections with statistical physics and rate-distortion theory were discussed. Upper bounds on the computational complexity of the algorithm were derived under quite mild assumptions about the data sets. The presented

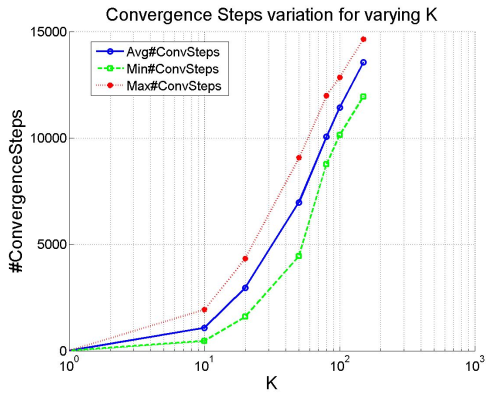

Fig. 3. Minimum, average and maximum numbers of iterations required for the convergence of Step 4) in the DA algorithm vs the allowed number of codevectors $K_{\max}$.

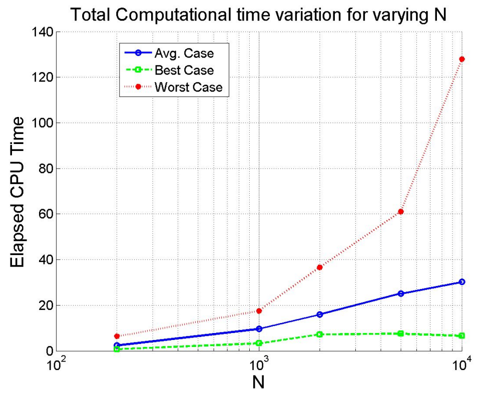

Fig. 4. $K_{\max} = 10$: Minimum, average and maximum time required for the DA algorithm vs the number of data points N.

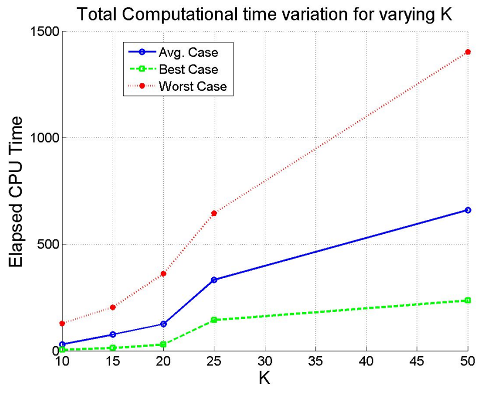

Fig. 5. $N = 10000$: Minimum, average and maximum time required for the DA algorithm vs $K_{\max}$.

theoretical aspects were accompanied by a numerical study of the behavior of the algorithm to complete the treatment. Future work will aim at the determination of tighter computational complexity characterizations for the DA algorithm.

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, "Maximum 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.