Auction Dynamics: A Volume Constrained MBO Scheme

Matt Jacobs Ekaterina Merkurjev Selim Esedoḡlu

October 22, 2017

Abstract

We show how auction algorithms, originally developed for the assigment problem, can be utilized in Merriman, Bence, and Osher's threshold dynamics scheme to simulate multi-phase motion by mean curvature in the presence of equality and inequality volume constraints on the individual phases. The resulting algorithms are highly efficient and robust, and can be used in simulations ranging from minimal partition problems in Euclidean space to semi-supervised machine learning via clustering on graphs. In the case of the latter application, numerous experimental results on benchmark machine learning datasets show that our approach exceeds the performance of current state-of-the-art methods, while requiring a fraction of the computation time.

1 Introduction

Threshold dynamics, also known as the MBO algorithm, is a very efficient algorithm for approximating motion by mean curvature of an interface or network of interfaces. Originally introduced by Merriman, Bence and Osher in [33], the algorithm generates a discrete in time approximation to mean curvature motion by alternating between two simple steps: convolution with a kernel and pointwise thresholding. The principal advantages of the algorithm are: implicit representation of the interface as in the phase field or level set methods, allowing for graceful handling of topological changes; unconditional stability, where the time step size is restricted only by accuracy considerations; and very low per time step cost when implemented on uniform grids.

The goal of this paper is to extend the MBO algorithm and its many beneficial properties to multiphase volume constrained curvature motion. Volume constrained curvature motion arises as L^2 gradient descent for the perimeter of sets functional with the additional proviso that each set must preserve certain volume constraints. As a result, volume constrained curvature motion is central to many interesting problems and applications; for example, finding equal volume tilings of space with minimal surface area (still an open problem in 3

dimensions), and volume constrained segmentation problems in computer vision and machine learning.

We obtain our scheme by appealing to a variational framework for the MBO algorithm developed by Esedoḡlu and Otto in [13]. The framework is based on the heat content energy, a non-local approximation to the perimeter of sets functional. Esedoḡlu and Otto showed that each step of the MBO algorithm is equivalent to minimizing the linearization of the heat content at the current configuration, which may be interpreted as a minimizing movements scheme for the energy. This simple variational approach gives a powerful tool to generalize threshold dynamics to a wide variety of situations, including curvature motion of a multiphase network with non-constant surface tensions [13, 12, 15], segmentation problems on graphs [45, 15, 23], and in this paper to volume constrained curvature flow.

Applying the variational approach to our current situation essentially entails minimizing linearizations of the heat content energy subject to certain volume constraints. Interestingly, the resulting minimization problem is equivalent to a famous combinatorial optimization problem, the assignment problem. The assignment problem is a member of a special family of linear programming problems known as minimum cost flow problems. These problems have many practical applications, such as finding the most efficient route to transport goods across a road network, and finding the best way to allocate resources among a population.

There are many well-known algorithms for solving the assignment problem. We choose to solve the problem using variants of the auction algorithm introduced by Bertsekas in [5]. Our resulting scheme consists of alternating two steps: convolution with a kernel, and assigning set memberships via an auction; hence the name *auction dynamics*. There are many reasons for favoring the auction approach. Auction algorithms are easy to code and have an intuitive structure. In practice, the computational complexity of the auction step scales similarly to the convolution step, thus preserving the efficiency of the original approach. Furthermore, the auction mechanism can be generalized to handle the most complex volume constraints, where each set must satisfy upper and lower volume bounds.

The remainder of the paper is organized as follows. We conclude the introduction with a summary of our contributions. Next, in Section 2, we give a summary of previous work. In Section 3, we dive into the heart of the matter, the auction dynamics algorithm. In particular, we give a detailed exploration of the assignment problem and auction algorithms, and develop the auction variants needed for auction dynamics. With the algorithm in hand, we consider two different applications of auction dynamics. In Section 4, we use auction dynamics to compute several examples of volume preserving curvature flow in two and three dimensions. We also provide numerical evidence that our scheme converges to the correct motion in cases where an exact solution is known. In Section 5, we apply auction dynamics to the semi-supervised learning problem, a well-known clustering problem in machine learning. Finally, we wrap up the paper with a brief conclusion in Section 6.

Contributions

The following is a summary of the present paper's contributions:

- We introduce *auction dynamics:* a highly efficient algorithm for computing the dynamics of multiphase volume constrained curvature flow.

- In the course of deriving our algorithm, we show how highly efficient auction algorithms of Bertsekas et. al. can be utilized to solve an assignment problem that naturally arises in Merriman, Bence, and Osher's multiphase threshold dynamics scheme in the presence of constraints on the volumes of individual phases.

  In particular, our work establishes a natural role for auction algorithms in simulating geometric motion, such as multi-phase volume preserving motion by mean curvature – a connection that was previously unnoticed.

- For certain applications, such as semi-supervised machine learning, it is more realistic to impose upper and lower bounds on the volumes of phases than strict equality constraints. In Section 3.3, we present an extension of the auction algorithm to assignment problems with inequality constraints.

  In turn, the new auction algorithm yields a version of the MBO scheme for approximately solving minimal partition problems with inequality constraints on the volumes of the phases.

- The new algorithms presented are unconditionally stable, and impose the (equality or otherwise) volume constraints **exactly** at every iteration, regardless of the time step size. Neither their complexity nor their accuracy in satisfying the volume constraints depends on the smoothness of the interfaces. As such, the algorithms are equally at home in Euclidean space as they are on abstract graphs.

- In Section 4, the new algorithms are demonstrated on volume preserving multi-phase motion by mean curvature in 2D and 3D. In some of the 3D experiments with equal volume constraints on the phases, the new algorithms are able to find the currently known best candidate for the minimal equal-volume partition, the Weaire-Phelan structure, starting from a randomly shifted cubic lattice as the initial condition.

- In Section 5, the new algorithms are demonstrated in the context of semi-supervised machine learning via clustering (based on minimal partitions) on graphs. On benchmark data sets such as the MNIST hand written digits data set, the algorithms allow us to demonstrate that volume constraints, even in the form of fairly loose inequality constraints, result in dramatic improvements in the accuracy of recognition. In addition, the efficiency of the new algorithms means completing the recognition task in a fraction of the time taken by alternative techniques.

2 Previous Work

There are a number of related numerical schemes for volume preserving versions of motion by mean curvature in the literature, mostly restricted to the two-phase setting. Here, we briefly discuss the closest ones and highlight essential differences.

Volume preserving mean curvature motion incurs the normal speed $v_\perp = \kappa - \bar{\kappa}$ for each phase with piecewise smooth boundary, where κ denotes mean curvature and $\bar{\kappa}$ its average over the boundary of that phase. A typical computational approach, e.g. in the level set [37] literature, is a literal implementation of this normal speed, which entails approximating the mean curvature and its average explicitly, see e.g. [38, 50]. As noted in [42], the approximation of the average curvature especially is prone to inaccuracies and can lead to an accumulation of errors. Instead, [42] notes that it can be regarded as a Lagrange multiplier for the volume constraint, the appropriate value of which can be determined by e.g. a line search procedure so that the constraint is satisfied exactly at the end of the time step; it then demonstrates how to implement this in the two-phase setting using threshold dynamics, thus resulting in an unconditionally stable algorithm (unlike its level set counterparts). The convergence of this scheme was studied in [28] along with a simple extension from [1] to the multiphase case where only one phase must satisfy a volume constraint. In [43], another explicit time stepping level set approach, the Lagrange multiplier is estimated in terms of the surface area and the mismatch in the constraint at the beginning of a forward Euler time step. It is then observed that the constraint is approximately satisfied in simulations in $\mathbb{R}^2$ and $\mathbb{R}^3$.

In this work, with an eye towards applications such as machine learning that are formulated in the context of abstract graphs on which one may not speak of let alone assume "smoothness" of interfaces, we seek unconditionally stable algorithms that impose the volume constraint exactly, regardless of the time step size, by computing the precise value of the Lagrange multipliers in the multi-phase setting. In [12, 47], it is shown that in the two-phase situation, the search for the multiplier in [42] can be replaced by a sort operation that allows satisfying the volume constraint exactly, without assuming anything about the interface, even in the context of clustering on graphs. In this paper, we present a similarly efficient, robust, and exact algorithm in the full generality of the multi-phase setting.

Threshold dynamics has been previously utilized for solving minimal partition problems in several contexts. In computer vision, threshold dynamics has been used for image segmentation in [14] via the Mumford-Shah model. In machine learning, graph based analogues of threshold dynamics have been introduced and used to solve the semi-supervised learning problem [4, 17] (see [32] for additional applications). [45] formulates some of the theory in [13] in the graph context, and [15, 23] contain some extensions that may be useful for that setting. Finally, the effect of volume constraints on classification accuracy for many of the benchmark data sets we present in Section 5 were recently explored in [2] using different numerical methods.

3 Auction Dynamics

Given a torus $D = [0,1]^d$, an N-phase partition $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_N)$ of D is a collection of closed subsets $\Sigma_i \subset D$ satisfying $\bigcup_{i=1}^N \Sigma_i = D$ and $\Sigma_i \cap \Sigma_j = \partial\Sigma_i \cap \partial\Sigma_j$. Our goal is to compute volume constrained curvature motion of the network of interfaces $\{\partial\Sigma_i \cap \partial\Sigma_j\}_{i\neq j}$. To set the stage, the unconstrained motion arises as L^2 gradient descent for the potentially anisotropic surface energy:

$$E(\mathbf{\Sigma}, \boldsymbol{\sigma}) = \sum_{i\neq j} \int_{\partial\Sigma_i \cap \partial\Sigma_j} \sigma_{ij}(n(x)) dH^{d-1}(x), \tag{1}$$

where $n(x)$ is the outward unit normal to a given interface at the point x, and $\sigma_{ij} : \mathbb{R}^d \to \mathbb{R}$ are the surface tensions, a collection of potentially anisotropic norms, $\sigma_{ij}(n) = \sigma_{ji}(n)$. We will restrict our attention to the special case where the surface tensions σ_{ij} are all constant multiples of the same norm (i.e. $\sigma_{ij}(n) = c_{ij}\sigma(n)$ for some collection of constants $\boldsymbol{c} > \mathbf{0}$). Under this assumption, it is natural to impose a triangle inequality on the surface tension constants

$$c_{ij} + c_{jk} \geq c_{ik} \quad \text{for all } i, j, k \text{ pairwise distinct}, \tag{2}$$

which in this case is equivalent to the lower semi-continuity of the surface energy (1). Indeed, the failure of (2) implies that at any interface between phases i and k one may decrease the surface energy (1) by adding an arbitrarily thin layer of phase j. However, it is worth noting that the triangle inequality is not necessary for the stability properties of our scheme.

The foundation of our approach is the variational framework for the MBO scheme introduced in [13]. The framework is based upon a non-local approximation to (1), the heat content energy. The heat content energy is defined on $\mathcal{K}_N$, the convex relaxation of the space of N-phase partitions of D,

$$\mathcal{K}_N = \{\boldsymbol{u} : D \to [0,1]^N : \sum_{i=1}^N u_i(x) = 1\}. \tag{3}$$

For some $\boldsymbol{u} \in \mathcal{K}_N$, a convolution kernel K, a time step δt, and constants c_{ij} the heat content is given by

$$\mathrm{HC}_{\sqrt{\delta t}}(\boldsymbol{u}, \boldsymbol{c}) = \frac{1}{\sqrt{\delta t}} \sum_{i\neq j} c_{ij} \int_D u_i(x)(K_{\sqrt{\delta t}} * u_j)(x) dx \tag{4}$$

where $K_{\sqrt{\delta t}}(x) = \frac{1}{(\delta t)^{d/2}} K(x/\sqrt{\delta t})$. If $\boldsymbol{u}$ is the characteristic function of a partition $\mathbf{\Sigma}$ then as $\delta t \to 0$ the heat content energy converges pointwise to

$$\sum_{i\neq j} c_{ij} \int_{\partial\Sigma_i \cap \partial\Sigma_j} \sigma_K(n(x)) dH^{d-1}(x) \tag{5}$$

where $\sigma_K(n) = \int_{\mathbb{R}^d} |x \cdot n| K(x) dx$, thus explaining the connection between the surface energy (1), the heat content (4), and the convolution kernel K [13]. See [16] for explicit kernel constructions that induce a given surface norm σ.

At the heart of [13] is the discovery that the MBO algorithm may be derived by successively minimizing linearizations of the heat content energy. Let $L_{\delta t}(\boldsymbol{u}, \cdot)$ be the linearization of (4) at some $\boldsymbol{u} \in \mathcal{K}_N$. Then the authors of [13] recover and generalize the MBO scheme by considering the following iteration

$$\boldsymbol{u}^{k+1} = \underset{\boldsymbol{u} \in \mathcal{K}_N}{\arg\min}\, L_{\delta t}(\boldsymbol{u}^k, \boldsymbol{u}). \tag{6}$$

We can see the connection to the MBO scheme by explicitly solving (6). Let

$$\psi_i^k(x) = \sum_{j \neq i} c_{j,i} (K_{\sqrt{\delta t}} * u_j^k)(x), \tag{7}$$

then up to a constant factor the linearization is given by

$$L_{\delta t}(\boldsymbol{u}^k, \boldsymbol{u}) = \sum_{i=1}^{N} \int_D \psi_i^k(x) u_i(x) dx. \tag{8}$$

It is easy to see that the global minimum, $\boldsymbol{u}^{k+1} \in \mathcal{K}_N$, is the characteristic function of a partition $\boldsymbol{\Sigma} = (\Sigma_1, \ldots, \Sigma_N)$ given by pointwise thresholding

$$\Sigma_i^{k+1} = \{x \in D : i = \underset{1 \leq j \leq N}{\arg\min}\, \psi_j^k(x)\} \quad \text{for all} \quad 1 \leq i \leq N. \tag{9}$$

When the surface tensions are constant (i.e. $c_{ij} = c$ for all $i \neq j$), the steps (7), calculating convolution values, and (9), pointwise thresholding, are precisely the classic MBO algorithm.

We now wish to extend the variational framework of the heat content energy to volume constrained MBO schemes. We begin by considering the simplest case where each phase must satisfy a volume equality constraint. Suppose we have a partition $\boldsymbol{\Sigma}$ where each phase has some volume $m(\Sigma_i) = v_i$ with respect to the Lebesgue measure m on D. The natural approach is to solve iteration (6) with the additional constraint that the volume of each phase must stay fixed. Thus, the thresholding step is instead replaced with the following minimization problem

$$\underset{\boldsymbol{u} \in \mathcal{K}_N}{\arg\min} \sum_{i=1}^{N} \int_D \psi_i^k(x) u_i(x) dx \quad \text{s.t.} \quad \int_D u_i(x) dx = v_i. \tag{10}$$

If we incorporate the volume constraints with a Lagrange multiplier $\boldsymbol{\lambda}$, we see that the solution to (10) is a partition $\boldsymbol{\Sigma}$ given by a $\boldsymbol{\lambda}^*$ shifted thresholding

$$\Sigma_i = \{x \in D : i = \underset{1 \leq i \leq N}{\arg\min}\, \psi_i(x) - \lambda_i^*\}, \tag{11}$$

where $\boldsymbol{\lambda}^*$ is the optimal Lagrange multiplier. It then follows essentially immediately from [41] that the scheme is consistent with volume preserving weighted curvature flow. Furthermore, if $\hat{K} \geq 0$ and the surface tension matrix C defined

in (12) is conditionally negative semi-definite (CNSD i.e. negative semi-definite on the orthogonal complement of the vector of all 1's)

$$C_{ij} = \begin{cases} 0 & \text{if } i = j, \\ c_{ij} & \text{otherwise} \end{cases} \tag{12}$$

then the scheme is also unconditionally stable [13]. In what follows, we will assume that the surface tension constants $\boldsymbol{c} > \mathbf{0}$ satisfy the triangle inequality (2) and the CNSD matrix condition. These conditions admit a large class of interesting surface tensions, including Read-Shockley surface tensions (see [13] for a further discussion of admissible surface tension constants).

While the scheme thus far seems straightforward, computing the optimal Lagrange multiplier $\boldsymbol{\lambda}^*$ is nontrivial if $N > 2$. This task is particularly difficult if one insists on solving for the Lagrange multiplier and the configuration $\boldsymbol{\Sigma}$ simultaneously (as we do). Our approach is to connect (10) to the assignment problem, a famous linear programming problem with efficient solutions. The assignment problem is typically posed as a maximization problem; thus, as a first step we will replace (10) with the equivalent problem (13)

$$\arg\max_{\boldsymbol{u} \in \mathcal{K}_N} \sum_{i=1}^{N} \int_D a_i(x) u_i(x) dx \quad \text{s.t.} \quad \int_D u_i(x) dx = v_i, \tag{13}$$

where $a_i(x) = (1 - \psi_i^k(x))$. However, rather than working with problem (13) directly, we will consider a discretized version. Discretization is natural, as any implementation of the scheme must be carried out on a finite grid. Discretization also allows us to more clearly connect our approach to the assignment problem, which is typically posed over a finite dimensional vector space. Let $D_n = \{x_1, \ldots, x_n\} \subset D$ be some n point discretization of D. We discretize the volume constraints by requiring each phase to occupy V_i points, where V_i are integers chosen so that $\sum_{i=1}^{N} V_i = n$ and the mass ratios $V_i/n \approx v_i/m(D)$ are as close as possible. Since the convolution values $a_i(x) = (1 - \psi_i^k(x))$ are smooth functions, they have a well defined restriction to D_n. Finally, the discrete analogue of $\mathcal{K}_N$ is the set of functions $\{\boldsymbol{u} : D_n \to [0,1]^N : \sum_{i=1}^{N} u_i(x) = 1\}$, which may also be represented as $\{\boldsymbol{u} \in [0,\infty)^{n \times N} : \sum_{i=1}^{N} u_i(x) = 1\}$. Using the latter representation we arrive at

$$\arg\max_{\boldsymbol{u} \geq \mathbf{0}} \sum_{i=1}^{N} \sum_{x \in D_n} a_i u_i(x) \quad \text{s.t.} \sum_{x \in D_n} u_i(x) = V_i, \sum_{i=1}^{N} u_i(x) = 1. \tag{14}$$

In this form, problem (14) can be viewed as a special case of a family of linear programming problems. This family of problems stems from the minimum cost flow problem, and includes famous problems such as the assignment problem, the transportation problem and the maximum flow problem. We choose to focus on the assignment problem, as it is the simplest of the problems and can be solved with an intuitive economic approach.

3.1 The assignment problem

Given two disjoint sets X and L of equal size r and a weight function $w : X \times L \to \mathbb{R}$, the assignment problem seeks to find a one-to-one matching $M = \{(x_1, \ell_1), \ldots, (x_r, \ell_r)\}$ of X and L (i.e. a bijection), such that the total weight of the matching

$$\sum_{(x,\ell)\in M} w(x, \ell) \tag{15}$$

is maximized. By representing the matching as a binary vector $\mathbf{z}$, where $z_\ell(x) = 1$ if (x, ℓ) are matched and $z_\ell(x) = 0$ otherwise, we can restate the assignment problem as the following optimization problem

$$\max_{\mathbf{z}:X\times L\to\{0,1\}} \sum_{x\in X}\sum_{\ell\in L} w(x,\ell)\ z_\ell(x) \quad \text{s.t.} \sum_{x\in X} z_\ell(x) = 1,\ \sum_{\ell\in L}\ z_\ell(x) = 1. \tag{16}$$

If we relax the binary constraint on $\boldsymbol{z}$, then (16) becomes the following linear programming problem

$$\max_{\boldsymbol{z}\geq\mathbf{0}} \sum_{x\in X}\sum_{\ell\in L} w(x,\ell)\ z_\ell(x) \quad \text{s.t.} \sum_{x\in X} z_\ell(x) = 1, \sum_{\ell\in L} z_\ell(x) = 1. \tag{17}$$

It turns out that the relaxation is exact, and we may substitute (17) for (16). This follows from the fact that the solution to a bounded and feasible linear programming problem always includes a vertex of the feasible polytope. The relaxed linear constraint set is the polytope $\{\boldsymbol{z} \geq \mathbf{0} : \sum_{x\in X} z_\ell(x) = 1, \sum_{\ell\in L} z_\ell(x) = 1\}$. The vertices of the polytope are precisely the vectors $\boldsymbol{z}$ whose entries are binary.

Observe that problem (14) is a special case of (17), with respect to a particular choice of weights $w(x, \ell)$. We can obtain (14) from a generic instance of (17) by letting $X = D_n$, splitting L into N similarity classes $\{S_i\}_{i=1}^N$ each of size V_i and setting $w(x, \ell) = a_i(x)$ for every $\ell \in S_i$. With those choices, (17) becomes

$$\max_{\boldsymbol{z}\geq\mathbf{0}} \sum_{x\in D_n}\sum_{i=1}^{N} a_i(x) \sum_{\ell\in S_i} z_\ell(x) \quad \text{s.t.} \sum_{x\in D_n} z_\ell(x) = 1, \sum_{\ell\in L} z_\ell(x) = 1. \tag{18}$$

The second constraint may be rewritten as $\sum_{i=1}^N \sum_{\ell\in S_i} z_\ell(x) = 1$, thus if we take $u_i(x) = \sum_{\ell\in S_i} z_\ell(x)$ we have nearly reduced problem (18) to problem (14). It remains to reduce the first constraint. It is clear that $\sum_{x\in D_n} z_\ell(x) = 1$ implies $\sum_{x\in D_n} u_i(x) = V_i$. To get the other direction, notice that the assignment of a person x to a particular object $\ell \in S_i$ does not change the value of the problem – it is enough to specify the matching at the level of equivalence classes. Therefore problem (14) encodes the same information as problem (18) and we see that (14) is a special case of (17).

For the remainder of the paper, we will focus on our special case (14) of the assignment problem (see [6] for a similar discussion of the classic formulation (17)). We will interchangeably represent matchings as vectors $\boldsymbol{u}$ in the feasible

polytope and as partitions $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_N)$ of D_n. Our goal for the remainder of this subsection is to develop an intuition for (14), and develop the necessary setup for the auction algorithm in Section 3.2.

It is particularly instructive to give a practical interpretation of (14). Imagine that each phase is an institution that offers a limited number of memberships. For example, the phases may be gyms, or clubs, or different Costco locations, etc. Imagine that the points $x \in D_n$ are people, and each person would like to become a member of some phase. No person wants to have a membership in more than one phase, and each phase only has V_i memberships available. Finally, imagine that the coefficients $a_i(x)$ represent how much person x wants to be a member of phase i. Now we can think of the solution to the assignment problem as the matching of people and phases that maximizes the total satisfaction of the population. Ideally, each person would like to become a member of their favorite phase. However, this is not possible if more than V_i people want to be members of some phase i. The main difficulty of the assignment problem is in understanding how to correctly handle these conflicts.

An interesting approach is to attempt to assign the memberships according to a market mechanism. Imagine that each phase i has a membership price p_i, and if person x is a member of i then they must pay p_i. This can help to resolve conflicts by making the most popular phases more expensive. Assuming that every person acts in their own best interest, x will want to buy a membership at the phase offering the best value, i.e. x wants to be a member of any phase

$$i^* \in i_{cs}(x, \boldsymbol{p}) = \arg\max_{1 \le i \le N} a_i(x) - p_i. \tag{19}$$

We are now led to a very interesting question: does there exist an equilibrium price vector $\boldsymbol{p}_*$ such that assigning memberships according to (19) gives a feasible matching? The answer to this question is yes, and better yet the resulting assignment is optimal.

The connection between the assignment problem and the equilibrium price vector $\boldsymbol{p}_*$ comes from the duality theory of linear programming. As it turns out, the equilibrium price $\boldsymbol{p}_*$ is in fact the optimal solution to the dual of the assignment problem. In addition to the prices $\boldsymbol{p}$, the dual problem introduces a set of variables $\boldsymbol{\pi}(x)$ for each $x \in D_n$. The dual problem is

$$\min_{\boldsymbol{p} \in \mathbb{R}^N,\ \boldsymbol{\pi} \in \mathbb{R}^n} \sum_{i=1}^{N} p_i V_i + \sum_{x \in D_n} \boldsymbol{\pi}(x) \quad \text{s.t. } \boldsymbol{\pi}(x) + p_i \ge a_i(x). \tag{20}$$

Note that the optimal value of $\boldsymbol{\pi}$ is entirely determined by $\boldsymbol{p}$. Given any $\boldsymbol{p}$, the best choice for $\boldsymbol{\pi}$ is to set $\boldsymbol{\pi}(x) = \max_{1 \le i \le N} a_i(x) - p_i$. This shows that $\boldsymbol{\pi}(x)$ is exactly the value of the best deal offered to x by any phase.

Our earlier statements about the equilibrium price vector $\boldsymbol{p}_*$ can be justified by invoking the complementary slackness (CS) condition. According to CS, a feasible assignment $\boldsymbol{u}$ and a feasible dual pair $(\boldsymbol{p}, \boldsymbol{\pi})$ are optimal for their respective problems if and only if

$$\sum_{x \in D_n} \sum_{i=1}^{N} u_i(x)(p_i + \boldsymbol{\pi}(x) - a_i(x)) = 0. \tag{21}$$

Recalling the best choice for $\boldsymbol{\pi}$, (21) implies that in the optimal matching, every person is assigned a membership which satisfies the market strategy (19) using the optimal price vector $\boldsymbol{p}_c$. This implies that the equilibrium price $\boldsymbol{p}_*$ exists and $\boldsymbol{p}_* = \boldsymbol{p}_c$.

Now suppose that we have some price vector $\boldsymbol{p}$ which is not optimal for the dual problem. By CS, it will not be possible to assign every membership according to (19), there will necessarily be conflicts. However, we can attempt to construct a partial assignment (partial matching). A partial assignment matches a subset of people in $S \subset D_n$ to phases $\{1, \ldots, N\}$ while ensuring that no more than V_i people are assigned to any phase. A partial matching can be represented as a partition $\boldsymbol{\Sigma}$ of S into N phases $\boldsymbol{\Sigma} = (\Sigma_1, \ldots, \Sigma_N)$ such that $|\Sigma_i| \leq V_i$ for every phase i. Given a partial matching $\boldsymbol{\Sigma}$, if $x \in \Sigma_i$ then it will be notationally convenient to say that the pair (x, i) is in the matching. A partial assignment $\boldsymbol{\Sigma}$ and a price $\boldsymbol{p}$ satisfies CS if for every phase i and every member $x \in \Sigma_i$, the pair (x, i) satisfies (19).

The most efficient algorithms for the assignment problem have the same basic structure. They generate a sequence of price vectors $\boldsymbol{p}^k$ and partial matchings $\boldsymbol{\Sigma}^k$ such that $\boldsymbol{\Sigma}^k$ and $\boldsymbol{p}^k$ satisfy CS. Each stage of the algorithm either increases the size of the partial matching (a larger subset of D_n matched) or improves the prices (with respect to the dual problem value). Since CS is preserved at every step, if the partial matching becomes a complete matching then it is an optimal solution to the assignment problem.

We will solve the assignment problem using auction algorithms. Auction algorithms have a simple intuitive structure, are easy to code, and have excellent performance. The main advantage of auction algorithms over the well-known Hungarian algorithm [27, 34] is that auction algorithms perform local modifications of $\boldsymbol{\Sigma}^k$ and $\boldsymbol{p}^k$ at every step, whereas the Hungarian algorithm may need to consider global modifications.

3.2 Auction algorithms

In [5], Bertsekas developed the auction algorithm for solving the classic assignment problem (17). Since the original paper, Bertsekas and collaborators have improved upon the computational aspects of the auction algorithm, and extended it to more general minimum cost flow problems (see [7] or [6] for an exhaustive reference on auction algorithms). The most important references for this work are [8] and [9]. In [8], Bertsekas and Castanon modified the auction algorithm to more efficiently handle assignment problems with multiple identical objects as in (14). In [9], Bertsekas, Castanon, and Tsaknakis introduced the reverse auction algorithm for asymmetric assignment problems, which we will use in Section 3.3.

The basic idea of the auction algorithm is to drive price modifications and augmentations of the partial matching by simulating an auction. In order to

obtain a membership, each person x must submit a bid $b(x)$ to the phase of their choice. At the start of the auction, the price of a membership at each phase i is set to a starting value p_i^0 according to some initial price vector $\boldsymbol{p}^0$. As in a real life auction, if a person x submits a bid $b(x)$ to a phase i, the bid must exceed or match the current price p_i. Using the CS condition (19), we can split the phases into three sets, the high demand phases H, the weak demand phases W, and the equilibrium phases E. The high demand phases $i \in H$ have more than V_i people who would like to purchase a membership, the weak demand phases $i \in W$ have fewer than V_i people and the equilibrium phases $i \in E$ have exactly V_i people. Everyone who wants a low demand or equilibrium membership can submit a bid and immediately be accepted into the initial partial matching, but there is a conflict at the high demand phases. The conflict is resolved by choosing the people who have submitted the largest bids. At any step of the algorithm, if i is a high demand phase, then the set Σ_i consists of the V_i people who have submitted the largest bids for phase i. As people submit bids, the prices of the high demand phases will rise. Eventually, this will incentivize unmatched people to switch their bid to a cheaper phase that may offer a better bang for their buck. The algorithm terminates once all of the phases are in equilibrium.

We now discuss pricing and bidding strategies. Each phase is restricted to setting one uniform membership price, regardless of how large individual bids may be. Assuming that a phase does not want to lose members, the price should be set to the amount that the least committed member is willing to pay. This amount is the lowest bid that a phase received thus far. To make this strategy consistent across all phases, assume that the empty spots in every weak demand phase $i \in W$ are filled by phantom members who all bid the starting price p_i^0.

Finding a bidding strategy that guarantees termination of the algorithm and produces a complete matching satisfying CS turns out to be nontrivial. For a given price vector $\boldsymbol{p}$, if person x is a member of phase i then we must have $i \in i_{cs}(x, \boldsymbol{p})$ to satisfy CS. This suggests that in the course of the auction, an unmatched person x should only submit bids to phases in $i_{cs}(x, \boldsymbol{p})$. The subtlety lies in the question, 'How much should person x be willing to bid?' Obviously, x does not want to overbid, otherwise prices may rise and a different phase will become optimal according to CS. The largest bid, $b(x)$, that x can submit to $i^* \in i_{cs}(x, \boldsymbol{p})$ while being guaranteed not to violate CS is

$$b(x) = p_{i^*} + (a_{i^*}(x) - p_{i^*}) - (a_{i_{\text{next}}}(x) - p_{i_{\text{next}}}), \tag{22}$$

where

$$i_{\text{next}} \in \arg\max_{j \neq i^*} a_j(x) - p_j \tag{23}$$

is x's second most desirable phase. With this bid, x is willing to allow the price of i^* to increase to at most the gap in value between the best and second best choice. While x is matched to i^*, the price p_{i^*} cannot increase beyond $b(x)$. Other prices are non-decreasing, thus for the duration that (x, i^*) is part of the partial matching, this pair satisfies CS.

Unfortunately, this bidding strategy does not work. A problem occurs when there are multiple optimal objects for x, i.e. when $|i_{cs}(x)| > 1$. If this happens,

both $i^*, i_{\text{next}} \in i_{cs}(x, \boldsymbol{p})$ and thus the gap $(a_{i^*}(x) - p_{i^*}) - (a_{i_{\text{next}}}(x) - p_{i_{\text{next}}}) = 0$. In this case, x is unable to raise the price of i^*. This situation may lead to a price war. In a price war, multiple people compete for the same memberships without ever raising the prices, trapping the algorithm in an infinite loop.

To circumvent this difficulty, one must relax the complementary slackness condition. For a given price vector $\mathbf{p}$ and a small $\varepsilon > 0$, a matched pair (x, i) satisfies the ε-complementary slackness condition (ε-CS) if

$$a_i(x) - p_i + \varepsilon \geq \max_{1 \leq j \leq N} a_i(x) - p_i. \tag{24}$$

It is now possible to create a bidding strategy that preserves ε-CS and guarantees that the algorithm will always terminate. As before, an unmatched x will only submit bids to $i^* \in i_{cs}(x, \boldsymbol{p})$; however, now x can bid up to

$$b(x) = p_{i^*} + \varepsilon + (a_{i^*}(x) - p_{i^*}) - (a_{i_{\text{next}}}(x) - p_{i_{\text{next}}}) \tag{25}$$

without overpaying according to ε-CS. Since $(a_{i^*}(x) - p_{i^*}) - (a_{i_{\text{next}}}(x) - p_{i_{\text{next}}}) \geq 0$ this ensures that p_{i^*} increases by at least ε. This mimics real life auctions where any bid must be larger than the current price by at least some fixed amount. Now, starting from any initial price vector $\boldsymbol{p}_0$, the algorithm will be guaranteed to eventually terminate [5]. We now give our version of the auction algorithm, which is equivalent to the "similar object" auction variant in [8]:

Algorithm 1: Membership Auction [8]

Input: $\varepsilon > 0$, volumes $\boldsymbol{V}$, coefficients $\boldsymbol{a}$, initial prices $\boldsymbol{p}^0$ and people $x \in D_n$
Result: Final prices and complete ε-CS matching $(\boldsymbol{\Sigma}, \boldsymbol{p})$.
Initialization: For every $i \in \{1, \ldots, N\}$ mark all x as unassigned, set $\boldsymbol{p} = \boldsymbol{p}^0$, set $\boldsymbol{\Sigma} = \varnothing$;
while *some x is marked as unassigned* **do**
  for *each unassigned $x \in D_n$* **do**
    Calculate $i_{cs}(x, \boldsymbol{p})$ and choose some $i^* \in i_{cs}(x, \boldsymbol{p})$;
    Set $b(x) = p_{i^*} + \varepsilon + (a_{i^*}(x) - p_{i^*}) - (a_{i_{\text{next}}}(x) - p_{i_{\text{next}}})$;
    if $|\Sigma_{i^*}| = V_{i^*}$ **then**
      Find $y = \arg\min_{z \in \Sigma_{i^*}} b(z)$;
      Remove y from Σ_{i^*} and add x to Σ_{i^*};
      Mark y as unassigned and mark x as assigned;
      Set $p_{i^*} = \min_{z \in \Sigma_{i^*}} b(z)$;
    else
      Mark x as assigned and add x to Σ_{i^*};
      if $|\Sigma_{i^*}| = V_i$ **then**
        Set $p_{i^*} = \min_{z \in \Sigma_{i^*}} b(z)$;
      end
    end
  end
end
return $(\boldsymbol{\Sigma}, \boldsymbol{p})$

The output of the auction algorithm is a complete matching $\mathbf{\Sigma}$ satisfying ε-CS, and the final auction prices $\boldsymbol{p}$. Representing the matching as a binary vector $\boldsymbol{u}$ and using ε-CS we may conclude

$$\sum_{i=1}^{N} p_i V_i + \sum_{x \in D_n} \max_{1 \le i \le N} [a_i(x) - p_i] \le n\varepsilon + \sum_{x \in D_n} \sum_{i=1}^{N} u_i(x) a_i(x). \tag{26}$$

Thus, by weak duality, the final assignment $\boldsymbol{u}$ is at most $n\varepsilon$ away from being optimal. In the special case where the coefficients $a_i(x)$ integers, an ε-CS matching is actually optimal for any $\varepsilon < \frac{1}{N}$ [8].

We now give a quick sketch of the complexity. Assume that at the start of the auction, the prices were initialized to zero and the partial matching was empty. Let $C = \max_{i \in \{1,\ldots,N\}, x \in D_n} a_i(x)$ be the largest coefficient. Suppose in the course of the algorithm that the price of some phase i exceeds C. If the algorithm has not yet terminated, then there must be some low demand phase with price zero. This implies that in the remainder of the auction, no person x will ever bid on phase i again, since there must be a phase offering a better value. Thus, we have an upper bound on the price of any phase. Suppose that some phase i is currently priced at p_i, and consider the number of bids required to raise the price. The worst possible case occurs when every currently matched member has bid exactly p_i (such a situation is highly degenerate and rarely appears in practical applications). In this case, it will take exactly V_i bids to raise the price. The price must rise by at least ε; thus, we can conclude that the algorithm will terminate after at most $NV\lceil C/\varepsilon \rceil$ bids, where $V = \max_{1 \le i \le N} V_i$.

A straightforward implementation of the bidding steps in Algorithm 1 requires $O(V + N)$ operations. This can be sped up with special data structures. If we implement a priority queue for each Σ_i, we can complete a bid in $O(\log(V) + N)$ operations. In all of our applications, V is several orders of magnitude larger than N; thus, this gives considerable savings. Combining this with the estimate for the maximum number of bids, we can conclude the algorithm has complexity $O\big(NV(\log(V) + N)C/\varepsilon\big)$. Note that due to the presence of the constant C, this complexity is pseudo-polynomial rather than polynomial.

The complexity can be improved using the idea of epsilon scaling (noted in [5] and analyzed in [19, 20, 21]). Suppose that $(\mathbf{\Sigma'}, \boldsymbol{p}')$ is a matching and a price vector satisfying $r\varepsilon$-CS for some $r > 1$. What happens if we use $\boldsymbol{p}'$ as the initial price vector when we run the auction algorithm with ε? Since any starting price is admissible, the algorithm will still produce a matching and price $(\mathbf{\Sigma}, \boldsymbol{p})$ satisfying ε-CS. However, if r is not too large, then we should expect that $\boldsymbol{p}'$ and $\boldsymbol{p}$ are not too different. This suggests that the auction will not need to modify the prices very much, and thus the algorithm will terminate quickly. Epsilon scaling takes advantage of this idea by running the auction multiple times with successively smaller values of epsilon. The final price vector of the previous run is used as the initial price vector in the next run. Typically, one takes the sequence of decreasing epsilon values to be $\varepsilon_k = C/\alpha^k$ for some integer $\alpha > 1$, stopping once $\varepsilon_k < \frac{\delta}{n}$ for some small δ. Using ε scaling the complexity can be

improved to a weakly polynomial bound. We refer our readers to [7] for the exact details and bounds using ε-scaling. For the problems that we consider, the complexity of the auction algorithm using ε-scaling appears to grow like $O\big(NV(\log(V)+N)\log(nC/\delta)\big)$ (see [6] or [7] for a heuristic explanation of this behavior).

Now we are ready to give the auction dynamics algorithm, Algorithm 2. Recall that our goal is to simulate the evolution of a configuration $\mathbf{\Sigma}$ under volume preserving curvature flow for some time $t = m(\delta t)$. As we saw in the beginning of Section 3, we obtain a consistent and unconditionally stable scheme by solving the iteration

$$\mathbf{\Sigma}^{k+1} = \arg\min_{\Sigma} \mathcal{L}_{\delta t}(\mathbf{\Sigma}^k, \mathbf{\Sigma}) \quad \text{s.t.} \quad |\Sigma_i| = V_i \quad \text{for } 1 \le i \le N \tag{27}$$

m times. This amounts to repeatedly taking convolutions of the configuration $\mathbf{\Sigma}^k$ with a kernel K, and solving the assignment problem. As we have seen above, we can solve the assignment problem efficiently using auctions. The auction dynamics algorithm uses Algorithm 1 along with ε-scaling to quickly and accurately obtain a solution. We give the algorithm below.

Algorithm 2: Auction Dynamics

Input: Discrete domain D_n, initial configuration $\mathbf{\Sigma}$, surface tensions $\boldsymbol{\sigma}$, convolution kernel K, volumes $\boldsymbol{V}$, time step δt, number of steps m, auction error tolerance ε_{min}, epsilon scaling factor α, initial epsilon value ε_0.

Result: Final configuration $\mathbf{\Sigma}^m$

Initialization: Set $\mathbf{\Sigma}^0 := \mathbf{\Sigma}$, set $\bar{\varepsilon} = \varepsilon_{min}/n$;

for k *from* 0 *to* $m-1$ **do**

  Calculate the convolutions: $\psi_i^{k+1}(x) = \sum_{j \neq i} \sigma_{ij} (K_{\sqrt{\delta t}} * \Sigma_j^k)(x)$;

  Calculate the assignment problem coefficients: $\boldsymbol{a}^{k+1} = 1 - \boldsymbol{\psi}^{k+1}$;

  Initialize prices $\boldsymbol{p} = \mathbf{0}$, and $\varepsilon = \varepsilon_0$;

  while $\varepsilon \geq \bar{\varepsilon}$ **do**

    Run Algorithm 1 (Membership Auction):

    $(\mathbf{\Sigma}_{\text{out}}, \boldsymbol{p}_{\text{out}}) = \text{Membership Auction}(\varepsilon, \boldsymbol{V}, \boldsymbol{a}^{k+1}, \boldsymbol{p}, D_n)$;

    Set $\boldsymbol{p} = \boldsymbol{p}_{\text{out}}$;

    Divide ε by α;

    if $\varepsilon < \bar{\varepsilon}$ **then**

      Set $\mathbf{\Sigma}^{k+1} = \mathbf{\Sigma}_{\text{out}}$;

    end

  end

end

return $\mathbf{\Sigma}^m$

3.3 Upper and lower volume bounds

In addition to strict volume preserving curvature flow, auction dynamics can be modified to allow the volume of each phase to fluctuate between some bounds. This will be particularly useful in our applications to machine learning.

Suppose that each phase i must have at least B_i members and at most U_i members for some integers B_i and U_i. To ensure that the resulting problem is feasible, we will require $B_i \leq U_i$ and $\sum_{i=1}^{N} B_i \leq n \leq \sum_{i=1}^{N} U_i$. We will then need to solve the following modified version of the assignment problem:

$$\max_{\boldsymbol{u}\geq \mathbf{0}} \sum_{i=1}^{N} \sum_{x\in D_n} a_i(x)u_i(x) \quad \text{s.t. } \sum_{i=1}^{N} u_i(x) = 1,\ B_i \leq \sum_{x\in D_n} u_i(x) \leq U_i. \tag{28}$$

This version of the problem introduces some complexities that were not present in (14) and will require a more sophisticated approach.

Previously, we examined and solved the assignment problem from the perspective of the people $x \in D_n$. The limited supply of memberships resulted in competition between the people, which we resolved by introducing prices and simulating an auction. The upper bounds fit nicely into this perspective. The upper bounds indicate that each phase has a limited number of memberships. However it is now possible that the total supply of memberships $\sum_{i=1}^{N} U_i$ exceeds the number of people n. The upper bounds will still induce competition between the people, but any oversupply of memberships means that the set of equilibrium prices will be larger. This will add a wrinkle of difficulty, as not all equilibrium prices will be dual optimal.

The lower bounds are fundamentally different and require a new perspective. Indeed, if some person x sees that there is an available membership in their most desirable phase i, they will immediately join i without caring if some other phase j is deficient (i.e. $|\Sigma_j| < B_j$). Instead, we must think about the lower bounds from the perspective of the phases. Imagine that each phase must sell B_i memberships or they will go out of business. If a phase i is having trouble attracting a sufficient number of people, it will have to introduce an incentive t_i to entice people to join. As a result, the lower bounds induce a competition among the phases. Deficient phases will be forced to offer competing incentives to attract the necessary number of members. Thus, in order to satisfy the lower bounds, we will need to run a reverse auction [9] where the phases bid on the people.

To properly understand the interaction between the prices and incentives, we introduce the dual problem

$$\min_{\boldsymbol{p}\geq \mathbf{0}, \boldsymbol{t}\geq \mathbf{0},\ \boldsymbol{\pi}\ \in \mathbb{R}^n} \sum_{i=1}^{N} p_i U_i - t_i B_i + \sum_{x\in D_n} \boldsymbol{\pi}\,(x) \quad \text{s.t.} \quad p_i - t_i + \boldsymbol{\pi}\,(x) \geq a_i(x). \tag{29}$$

As before, we will use the interplay between the primal and dual problems to drive the search for the optimal solution. The key of course will be the complementary slackness condition. The complementary slackness condition for (28) and (29) states that an assignment $\boldsymbol{u}$ and dual variables $(\boldsymbol{p}, \boldsymbol{t},\ \boldsymbol{\pi}\ )$ are optimal for their respective problems if and only if

$$\sum_{i=1}^{N} \sum_{x \in D_n} u_i(x)(a_i(x) - p_i + t_i - \boldsymbol{\pi}(x)) + \sum_{i=1}^{N} p_i(U_i - \sum_{x \in D_n} u_i(x)) + \sum_{i=1}^{N} t_i(\sum_{x \in D_n} u_i(x) - B_i) = 0. \tag{30}$$

Recall that $\boldsymbol{\pi}$ is determined by $\boldsymbol{p}$ and $\boldsymbol{t}$ and is given by $\boldsymbol{\pi}(x) = \max_{1 \le i \le N} a_i(x) + t_i - p_i$. Now we can recognize that the complementary slackness condition has a simple intuitive explanation. The first sum states that each person should be assigned to the optimal phase based on prices and incentives (this should feel familiar). The second sum states that phases charging membership prices must have the maximum number of members U_i (i.e. no overpriced phases). Similarly, the third sum states that the phases offering incentives must have the minimal number of members B_i (i.e. no over-incentivized phases).

To ensure our auctions do not stall, we will once again turn to the ε-CS condition. For this problem, we will say that a partial matching $\boldsymbol{\Sigma}$ and a price-incentive pair $(\boldsymbol{p}, \boldsymbol{t})$ satisfy ε-CS if every matched pair (x, i) satisfies

$$a_i(x) - p_i + t_i + \varepsilon \ge \max_{1 \le j \le N} a_j(x) - p_j + t_j. \tag{31}$$

As before, we can recognize this ε-CS condition as an ε relaxed version of the first sum in (30). Unfortunately, the other two terms in (30) do not have useful ε relaxations. As a result, we will need to carefully ensure that our auctions will satisfy the other two terms exactly. We will say that a price $\boldsymbol{p}$ (an incentive $\boldsymbol{t}$) is admissible for a matching $\boldsymbol{\Sigma}$ if the second (third) term of (30) is satisfied.

We will solve (28) in two stages. First we will run Algorithm 3, a forward auction algorithm similar to Algorithm 1, where the people compete for memberships. This will produce a complete ε-CS matching satisfying the upper bound constraints but possibly violating the lower bound constraints (we will call this upper feasible). Algorithm 3 differs from Algorithm 1, as it simultaneously runs a mechanism to ensure that no phase is over-incentivized. Note that this extra mechanism is only necessary if one wants to use ε-scaling. In the absence of ε-scaling, phases cannot become over-incentivized as long as $\boldsymbol{t}$ is initialized to $\mathbf{0}$. In the second stage we will feed the result of the first stage into a reverse auction, Algorithm 4, where the phases compete for people. This will produce an ε-CS matching that is both upper and lower feasible. In addition, Algorithm 4 will have a mechanism to prevent phases from becoming overpriced (this mechanism is necessary with or without ε-scaling). As a result, the final output will be a complete and feasible ε-CS matching $\boldsymbol{\Sigma}$ with admissible prices and incentives $(\boldsymbol{p}, \boldsymbol{t})$. This will be enough to conclude that $\boldsymbol{\Sigma}$ solves (28) with error at most $n\varepsilon$. In the special case that the coefficients $\boldsymbol{a}$ are all integers, the argument used in [8] can be easily generalized to show that the solution is optimal if $\varepsilon < \frac{1}{N}$.

Algorithm 3 is a relatively straightforward adaptation of the similar object auctions and the asymmetric assignment auctions found in [7]. On the other

hand, Algorithm 4 appears to have a different structure than the reverse auctions considered in [7]. Indeed, in our reverse auction we choose to work with prices and incentives rather than the profit variable $\boldsymbol{\pi}$. We find that working with prices and incentives leads to a much faster runtime when $N << n$. Since both algorithms are specialized for our current problem, we provide proofs that they terminate and have the desired properties.

Algorithm 3: Upper Bound Auction

Input: $\varepsilon > 0$, bounds $\boldsymbol{B}, \boldsymbol{U}$, coefficients $\boldsymbol{a}$, initial prices $\boldsymbol{p}^0$, initial incentives $\boldsymbol{t}^0$ and people $x \in D_n$

Result: Prices $\boldsymbol{p}$, admissible incentives $\boldsymbol{t}$, and complete ε-CS matching $\boldsymbol{\Sigma}$ satisfying upper bounds.

Initialization: Mark all x as unassigned, set $\boldsymbol{d} = \boldsymbol{p}^0 - \boldsymbol{t}^0$, set $\boldsymbol{\Sigma} = \varnothing$;

while *some x is marked as unassigned* **do**

  for *each unassigned $x \in D_n$* **do**

    Calculate $i_{cs}(x, \boldsymbol{p})$ and choose some $i^* \in i_{cs}(x, \boldsymbol{d})$;

    Set $b(x) = d_{i^*} + \varepsilon + (a_{i^*}(x) - d_{i^*}) - (a_{i_{\text{next}}}(x) - d_{i_{\text{next}}})$;

    if $|\Sigma_{i^*}| = U_{i^*}$ **then**

      Find $y = \arg\min_{z \in \Sigma_{i^*}} b(z)$;

      Remove y from Σ_{i^*} and add x to Σ_{i^*};

      Mark y as unassigned and mark x as assigned;

      Set $d_{i^*} = \min_{z \in \Sigma_{i^*}} b(z)$;

    else if $|\Sigma_i| = B_i$ *and* $d_i < 0$ **then**

      Find $y = \arg\min_{z \in \Sigma_{i^*}} b(z)$;

      Remove y from Σ_{i^*} and add x to Σ_{i^*};

      Mark y as unassigned and mark x as assigned;

      Set $d_{i^*} = \min(\min_{z \in \Sigma_{i^*}} b(z), 0)$;

    else

      Mark x as assigned and add x to Σ_{i^*};

    end

  end

end

Set $\boldsymbol{p} = \max(\boldsymbol{d}, \boldsymbol{0})$, set $\boldsymbol{t} = \max(-\boldsymbol{d}, \boldsymbol{0})$;

return $(\boldsymbol{\Sigma}, \boldsymbol{p}, \boldsymbol{t})$

Proposition 3.1. *Given initial prices and incentives $\boldsymbol{p}^0, \boldsymbol{t}^0$, and an empty matching, Algorithm 3 produces an upper feasible ε-CS matching $\boldsymbol{\Sigma}$ with no over-incentivized phases with time complexity $O(NU(\log(U) + N)(C + G)/\varepsilon)$, where $U = \max_{1 \le i \le N} U_i$ and $G = \max_{1 \le i,j \le N}(p_j^0 - t_j^0) - (p_i^0 - t_i^0)$.*

Proof. Note that no phase can increase beyond U_i members, and no phase can increase beyond B_i members as long as $d_i < 0$. Therefore, the algorithm will not terminate until the matching is complete, upper feasible, and there are no over-incentivized phases. Throughout the auction, the number of unmatched

people is non-increasing and the variable $\boldsymbol{d}$ is entrywise non-decreasing. The monotonicity of these quantities allows us to use the same complexity argument as in Algorithm 1. The above bound will then immediately follow, where the factor G accounts for the prices and incentives not being initialized to zero.

It remains to show that the algorithm preserves ε-CS at every step. The only place where this algorithm differs from Algorithm 1 is when a person x wants to join a phase i, where $|\Sigma_i| = B_i$ and $d_i < 0$. Let $\boldsymbol{d}, \boldsymbol{d}'$ be the values before and after x is added. Since $d_i' \leq \min_{y \in \Sigma_i} b(y)$, every person matched to i must satisfy ε-CS. □

Proposition 3.2. *Given the result of Algorithm 3, Algorithm 4 produces a complete and feasible ε-CS matching $\boldsymbol{\Sigma}$ with no overpriced or over-incentivized phases with time complexity $O(n^2N^2(C+G)/\varepsilon)$ where $G = \max_{i \neq j}(p_j^0 - t_j^0) - (p_i^0 - t_i^0)$.*

Proof. It is clear that the algorithm will not terminate until the matching is complete and lower feasible, and there are no over-priced phases. The algorithm will never add people to an already full phase i with $|\Sigma_i| = U_i$, thus the matching stays upper feasible. A phase only offers incentives if it has fewer than B_i members, and any phase that has offered an incentive will never have more than B_i members. Thus, no phase will become over-incentivized.

Next, we show that $\boldsymbol{\Sigma}$ is a complete ε-CS matching at every step of the algorithm. Consider what happens when a phase i^* is modified. Let $(\boldsymbol{\Sigma}, \boldsymbol{d})$ contain the values before the modification and $(\boldsymbol{\Sigma}', \boldsymbol{d}')$ afterwards.

First, we consider the case where $|\Sigma_{i^*}| < B_{i^*}$. In this case, Σ_{i^*}' must now have B_i points. Let x_f be the last point added to i^*. If $d_{i^*}' = d_{i^*}$, then $\Delta(x_f) < 0$ and we can conclude that every person who had their membership switched to i^* strictly preferred i^* over their previous membership. Since no other entry of $\boldsymbol{d}$ changed, the new pair $(\boldsymbol{\Sigma}', \boldsymbol{d}')$ still satisfies ε-CS. Otherwise, $\Delta(x_f) \geq 0$ and $d_i' = d_i - \Delta(x_f) - \varepsilon$. Clearly, everyone who was in Σ_i is even happier to be in Σ_{i^*}' as $d_{i^*}' < d_{i^*}$ and other entries of $\boldsymbol{d}$ didn't change. Next, we check the other people whose membership didn't change. Let y be some person $y \in \Sigma_j'$ for some $j \neq i^*$. We need to show that $\max_{1 \leq i \leq N} a_i(y) - d_i' - \varepsilon \leq a_j(y) - d_j'$. Only d_{i^*}' is different, so it is enough to show $a_{i^*}(y) - d_{i^*}' - \varepsilon \leq a_j(y) - d_j$. By our choice of x_f, we have

$$a_{i^*}(y) - d_{i^*}' - \varepsilon = a_{i^*}(y) - d_{i^*} + \Delta(x_f) \leq a_{i^*}(y) - d_{i^*} + \Delta(y) = a_j(y) - d_j.$$

Finally, we check the people who were switched to i^*. Let z be one of those people and suppose that z was previously matched to phase r. Since $\Delta(x_f) \geq \Delta(z)$, we may conclude

$$\max_{i \neq i^*} a_i(z) - d_i' \leq a_r(z) - d_r + \varepsilon = a_{i^*}(z) - d_{i^*} + \varepsilon + \Delta(z) \leq a_{i^*}(z) - d_{i^*}'.$$

Next, we consider the case where $|\Sigma_{i^*}| < U_{i^*}$ and $d_{i^*} > 0$. This case is very similar; however, there is one additional thing that can happen. Namely,

Algorithm 4: Lower Bound Auction

Input: $\varepsilon > 0$, bounds $\boldsymbol{B}, \boldsymbol{U}$, coefficients $\boldsymbol{a}$, initial prices $\boldsymbol{p}^0$, initial admissible incentives $\boldsymbol{t}^0$, complete (but possibly lower infeasible) ε-CS matching $\boldsymbol{\Sigma^0}$

Initialization: Set $\boldsymbol{d} = \boldsymbol{p}^0 - \boldsymbol{t}^0$, set $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^0$;

Result: complete and feasible ε-CS matching and admissible prices and admissible incentives $(\boldsymbol{\Sigma}, \boldsymbol{p}, \boldsymbol{t})$.

```
while there exists some i with (|Σ_i| < U_i and d_i > 0) or (|Σ_i| < B_i) do
    for each i* with (|Σ_{i*}| < U_{i*} and d_{i*} > 0) or (|Σ_{i*}| < B_{i*}) do
        for each x ∉ Σ_{i*} do
            Let j be x's current phase;
            Calculate Δ(x) = (a_j(x) − d_j) − (a_{i*}(x) − d_{i*});
        end
        while (|Σ_{i*}| < U_{i*} and d_{i*} > 0) or (|Σ_{i*}| < B_{i*}) do
            Find x ∈ arg min_{y∉Σ_{i*}} Δ(y);
            if |Σ_{i*}| < B_{i*} then
                Remove x from its current phase and add x to Σ_{i*};
                if |Σ_{i*}| = B_{i*} and Δ(x) ≥ 0 then
                    Subtract Δ(x) + ε from d_{i*};
                end
            else
                if Δ(x) + ε ≥ d_{i*} then
                    Set d_{i*} = 0;
                else
                    Remove x from its current phase and add x to Σ_{i*};
                    if |Σ_{i*}| = U_{i*} and Δ(x) ≥ 0 then
                        Subtract Δ(x) + ε from d_{i*};
                    end
                end
            end
        end
    end
end
Set p = max(d, 0), set t = max(−d, 0);
return (Σ, p, t)
```

it is possible that d'_{i^*} can be set to zero before Σ'_{i^*} reaches U_{i^*} members. As before, let x_f be the last person added to i^* in the modification, and let $y_c = \arg\min_{y \notin \Sigma'_i} \Delta(y)$. If x_f exists (possibly no one was added) then $\Delta(x_f) + \varepsilon < d_{i^*} \leq \Delta(y_c) + \varepsilon$. Similar arguments to the above now show that anyone in Σ'_{i^*} satisfies ε-CS. To check that every other person satisfies ε-CS it is enough to show that y_c satisfies ε-CS. Suppose that y_c is matched to a phase j. Then

$$a_{i^*}(y_c) - d'_{i^*} \leq a_{i^*}(y_c) + \Delta(y_c) + \varepsilon - d_{i^*} = a_j(y_c) - d_j + \varepsilon,$$

which is enough to show ε-CS for y_c. Thus, the algorithm preserves ε-CS.

Finally, we show that the algorithm terminates. Suppose that, for some i, the quantity d_i decreases by more than $2(C+G)+\varepsilon$ from its starting value. Since $d_i^0 - G \leq 0$, it must be the case that $|\Sigma_i| \leq B_i$. Immediately after d_i is lowered to $d_i^0 - C - 2G - \varepsilon$, phase i must have exactly B_i members. If the algorithm has not terminated, then there must be some j with more than B_j members, and thus $d_j \geq d_j^0 - G$. For any x, we can then conclude that

$$a_i(x) - d_i - (a_j(x) - d_j) \geq a_i(x) - a_j(x) + d_j^0 - d_i^0 + 2C + G \geq 0.$$

It then follows that $|\Sigma_i| = B_i$ for the remainder of the auction, as it will always be easier for other phases to incentivize people to leave phase j rather than phase i.

Notice that the same person cannot switch phases N times unless one of the entries of $\boldsymbol{d}$ has decreased. Thus, a phase i can enter a bidding stage at most Nn times before d_i must decrease by at least ε. This gives us an upper bound of $2N^2n\lceil (C+G)/\varepsilon \rceil$ bidding stages before the algorithm terminates. Quickselect can be used to find the k smallest values of $\Delta(x)$ in time $O(n)$ regardless of k. Thus, the worst case complexity of the algorithm is $O(n^2N^2(C+G)/\varepsilon)$.

□

Both Algorithms 3 and 4 are compatible with ε scaling. The prices and incentives obtained from one iteration of Algorithms 3 and 4 together can be fed into the next iteration. For the instances of (28) that we encounter, the complexity of both algorithms using ε scaling appears to grow like $O(nN(\log(n) + N)\log(nC/\delta))$, where $\delta > 0$ is the maximum error of the final solution.

With the upper and lower bound auction algorithms in hand, we can now give the version of auction dynamics with upper and lower volume bounds, described by Algorithm 5.

3.4 Auction dynamics with temperature

Finally, we conclude this section with a variant of the auction dynamics algorithm that allows us to incorporate random fluctuations due to temperature. There are several reasons to introduce temperature effects into auction dynamics, two of these are:

- When using auction dynamics to solve minimal partition problems (e.g. data segmentation in machine learning, optimal tessellations, etc.), temperature can help the algorithm escape from local minima and find better solutions.
- Low temperature levels can be added to auction dynamics to help avoid degenerate auction coefficients (which slow down the algorithm) without significantly changing the result.

In the classic threshold dynamics algorithm, one may incorporate temperature in the style of rejection free Monte Carlo methods by randomizing the thresholding

Algorithm 5: Auction Dynamics with Volume Bounds

Input: Domain D_n, initial configuration $\boldsymbol{\Sigma}$, surface tensions $\boldsymbol{\sigma}$, kernel K, volume bounds $\boldsymbol{B}, \boldsymbol{U}$, time step δt, number of steps m, auction error tolerance ε_{min}, epsilon scaling factor α, initial epsilon ε_0.

Result: Final configuration $\boldsymbol{\Sigma}^m$

Initialization: Set $\boldsymbol{\Sigma}^0 := \boldsymbol{\Sigma}$, set $\bar{\varepsilon} = \varepsilon_{min}/n$;

for k *from* 0 *to* $m-1$ **do**

  Calculate the convolutions: $\psi_i^{k+1}(x) = \sum_{j \neq i} \sigma_{ij}(K_{\sqrt{\delta t}} * \Sigma_j^k)(x)$;

  Calculate the assignment problem coefficients: $\boldsymbol{a}^{k+1} = 1 - \boldsymbol{\psi}^{k+1}$;

  Initialize prices $\boldsymbol{p} = \boldsymbol{0}$, incentives $\boldsymbol{t} = \boldsymbol{0}$, and $\varepsilon = \varepsilon_0$;

  while $\varepsilon \geq \bar{\varepsilon}$ **do**

    Run Algorithm 3 (Upper Bound Auction): $(\boldsymbol{\Sigma}_{\text{out1}}, \boldsymbol{p}_{\text{out1}}, \boldsymbol{t}_{\text{out1}}) =$ Upper Bound Auction$(\varepsilon, \boldsymbol{B}, \boldsymbol{U}, \boldsymbol{a}^{k+1}, \boldsymbol{p}, \boldsymbol{t}, D_n)$;

    Run Algorithm 4 (Lower Bound Auction): $(\boldsymbol{\Sigma}_{\text{out2}}, \boldsymbol{p}_{\text{out2}}, \boldsymbol{t}_{\text{out2}}) =$ Lower Bound Auction$(\varepsilon, \boldsymbol{B}, \boldsymbol{U}, \boldsymbol{a}^{k+1}, \boldsymbol{p}_{\text{out1}}, \boldsymbol{t}_{\text{out1}}, \boldsymbol{\Sigma}_{\text{out1}})$;

    Set $(\boldsymbol{p}, \boldsymbol{t}) = (\boldsymbol{p}_{\text{out2}}, \boldsymbol{t}_{\text{out2}})$;

    Divide ε by α;

    if $\varepsilon < \bar{\varepsilon}$ **then**

      Set $\boldsymbol{\Sigma}^{k+1} = \boldsymbol{\Sigma}_{\text{out2}}$;

    end

  end

end

return $\boldsymbol{\Sigma}^m$

step. The Monte-Carlo approach suggests randomly assigning each x to a phase i with probability:

$$\mathbb{P}(x \in \Sigma_i^{k+1}) = \frac{e^{-\beta \psi_i^{k+1}(x)}}{\sum_{j=1}^{N} e^{-\beta \psi_j^{k+1}(x)}}, \tag{32}$$

where $\beta = \frac{1}{T}$ is the inverse temperature. In the limit as $T \to 0$ one recovers the original MBO algorithm.

Unfortunately this approach is not compatible with auction dynamics. The volume constraints prevent us from assigning points independently. As a result, we cannot introduce the randomness in the assignment step. Instead, we introduce temperature before the assignment step by perturbing the coefficients $a_i(x) = (1 - \psi_i(x))$. Given some probability distribution $X = X(0, T)$ on the reals with mean zero and variance T, we perturb each coefficient $a_i(x)$ by an independent sample of X. This approach maintains the same basic properties as the randomness strategy (32). As $T \to 0$, we recover the original algorithm and as $T \to \infty$ the points are assigned to phases completely randomly. In our implementations of temperature effects, we choose the random variables to be normal random variables $N(0, T)$.

Table 1: Error for the circles example

Time step / Total time	$t = 0.01$	$t = 0.02$	$t = 0.03$	$t = 0.04$	$t = 0.05$
$\delta t = 0.001$	$1.80 * 10^{-4}$	$4.27 * 10^{-4}$	$1.10 * 10^{-3}$	$2.16 * 10^{-3}$	$4.88 * 10^{-3}$
$\delta t = 0.0005$	$8.60 * 10^{-5}$	$2.74 * 10^{-4}$	$5.13 * 10^{-4}$	$9.78 * 10^{-4}$	$1.84 * 10^{-3}$
$\delta t = 0.00025$	$4.88 * 10^{-6}$	$6.85 * 10^{-5}$	$1.62 * 10^{-4}$	$3.46 * 10^{-4}$	$5.52 * 10^{-4}$

The entries of the table give the error in the value of $R_1(t)$ computed by auction dynamics for various step sizes δt and times t.

4 Experimental results for curvature flow

We demonstrate our auction dynamics algorithm by computing several examples of volume preserving mean curvature motion in two and three dimensions. Since the focus of this work is to develop the necessary theory and algorithms for the volume constrained case, we work with essentially the most basic implementation of auction dynamics with the exception of the following well-known and simple trick to enhance the spatial resolution. The intermediate steps arising in each iteration of auction dynamics yields a smooth level set function (given by $\boldsymbol{\psi} - \boldsymbol{p}$) that can be used (via interpolation) to estimate the fraction of each grid cell occupied by a given phase. This allows for a sub-pixel accurate representation of the characteristic functions of the phases. For applications requiring greater efficiency or accuracy, one may turn to more sophisticated techniques developed for threshold dynamics, e.g. [39, 40], which in principle extend to auction dynamics as well.

4.1 Numerical convergence tests

We begin by demonstrating the accuracy of our method on two simple examples of volume preserving curvature flow with explicit solutions. First we consider the three phase example shown in Figure 1 on the periodic square $[0, 1]^2$. The red phase consists of two disjoint circles of different sizes, with radii $r_1 = 1/6$ and $R_1 = 1/5$ respectively. The blue phase is a translation of the red phase. Under volume preserving curvature flow, the larger red and blue circles will grow while the smaller circles shrink. Since the red and blue phases are identical, the flow may be completely described by the coupled system of equations for the radii of the two red circles:

$$\begin{aligned} R_1'(t) &= \frac{R_1(t) - r_1(t)}{R_1(t)(r_1(t) + R_1(t))} \\ r_1'(t) &= \frac{r_1(t) - R_1(t)}{r_1(t)(r_1(t) + R_1(t))}. \end{aligned} \tag{33}$$

The equations are valid until the smaller circle disappears, and the flow becomes stationary. Using numerical integration on (33) we may compute the flow to arbitrary precision.

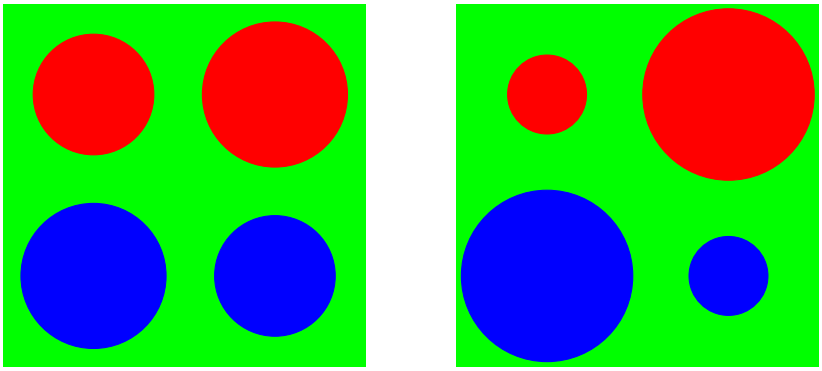

Figure 1: Three phase configuration on a periodic square. The red and blue phases consist of two disjoint circles with different radii, $R_1 = 1/5$ and $r_1 = 1/6$. The green phase is the complement of the union of all of the circles. Under volume preserving curvature flow the two larger circles will grow, while the smaller circles will shrink until they disappear. The left hand side shows the initial configuration, while the right hand side shows the configuration at time $t = .04$.

Table 2: Error for the spheres example

Time step / Total time	$t = 0.008$	$t = 0.016$	$t = 0.024$	$t = 0.032$	$t = 0.040$
$\delta t = 0.004$	$1.64 * 10^{-4}$	$9.22 * 10^{-4}$	$2.00 * 10^{-3}$	$3.63 * 10^{-3}$	$5.28 * 10^{-3}$
$\delta t = 0.002$	$1.41 * 10^{-4}$	$4.90 * 10^{-4}$	$1.05 * 10^{-3}$	$1.80 * 10^{-3}$	$2.80 * 10^{-3}$
$\delta t = 0.001$	$1.08 * 10^{-4}$	$3.18 * 10^{-4}$	$5.88 * 10^{-4}$	$9.60 * 10^{-4}$	$1.43 * 10^{-3}$

The entries of the table give the absolute error in the value of $R_2(t)$ computed by auction dynamics for various step sizes δt and times t.

In Table 1, we compare the results of auction dynamics on the three phase circle configuration to the (essentially) exact solution of (33) computed by numerical integration. The entries of the table display the absolute error in the auction dynamics value of $R_1(t)$ (the radius of the larger red circle) for various step sizes δt and times t. As the step size δt decreases, the auction dynamics solution becomes more accurate. The scheme appears to be first order accurate in time which matches the truncation error analysis for the classical threshold dynamics scheme.

Next, we consider an analogous three phase example on the periodic cube $[0, 1]^3$. The first phase consists of two disjoint spheres of radii $r_2 = 1/6$ and $R_2 = 1/5$ centered at $(1/4, 1/4, 1/4)$ and $(1/4, 1/4, 3/4)$. The second phase is a translation of the first phase by $(1/2, 1/2, 1/2)$. As above, the dynamics of the flow can be completely characterized as a coupled system of equations for the radii of the spheres in the first phase (r_2, R_2) given by

$$\begin{aligned} R_2'(t) &= \frac{R_2(t) + r_2(t)}{R_2(t)^2 + r_2(t)^2} - \frac{1}{R_2(t)} \\ r_2'(t) &= \frac{R_2(t) + r_2(t)}{R_2(t)^2 + r_2(t)^2} - \frac{1}{r_2(t)}. \end{aligned} \tag{34}$$

The equations are valid until the smaller sphere vanishes at which point the flow becomes stationary.

In Table 2 we compare the results of auction dynamics on the three phase sphere configuration to the (essentially) exact solution of (34) computed by numerical integration. The entries of the table display the absolute error in the auction dynamics value of $R_2(t)$ (the radius of the larger sphere in the first phase) for various step sizes δt and times t. Again, we see that the scheme is first order accurate in time.

4.2 Tessellations

We begin by considering two different equal volume tessellations of the torus. In Figure 2, the starting configuration is 64 randomly shifted squares of equal volume. After evolving under auction dynamics, the final configuration is a hexagonal lattice, which has optimal isoperimetric quotient among all equal volume tilings of the plane [22]. Thus, the algorithm finds the lowest energy state as one would hope. A more interesting example is given in Figure 3. The starting configuration consists of 17 equal volume rectangles. In the case of 17 subunits, it is impossible to tile the torus with hexagons [31]. Indeed, the final configuration contains a heptagon and a pentagon. Nevertheless, most of the shapes are hexagons and visual inspection suggests that all of the triple junction angles are nearly 120 degrees. Therefore, the final configuration is a plausible local minimizer of the interfacial perimeters.

Next, we consider random Voronoi diagrams in 2 and 3 dimensions. Figure 4 depicts the evolution of a random Voronoi diagram in the plane. The network immediately undergoes topological changes – all of quadruple junctions in the initial configurations split and form triple junctions. Figure 5 shows the evolution of a single "grain" in a random Voronoi diagram in 3 dimensions, Figure 6 shows the same grain and several of its neighbors at the final time. One can clearly see many topological changes in the faces of the grain. Quadruple junctions split and collide throughout the evolution. Both examples clearly show that one must anticipate topological changes in the course of the flow.

Finally, we consider equal volume tilings in 3 dimensions. Our starting configuration is a randomly shifted cubic lattice with 8 phases. Unlike the two dimensional case above, where the flow easily found the optimal solution, the 3 dimensional energy landscape appears to be littered with local minima. Regardless of how the cubes are shifted, the configuration evolves to a final state where each grain assumes the shape shown in Figure 7 – a 12 sided polytope built from 4 trapezoids, 4 rhombi, and 4 hexagons. A simple calculation shows that the isoperimetric quotient of this structure is considerably worse than several well-known tilings of 3-space. On the other hand, if we run the flow in the presence of temperature, the random fluctuations allow us to escape the local minima. Figure 8 shows an experiment with temperature where the final configuration assumes the structure of what is thought to be the most efficient partition of 3-space, the Weaire-Phelan structure [46]. This experiment suggests that auction dynamics with temperature may be a very useful tool for exploring

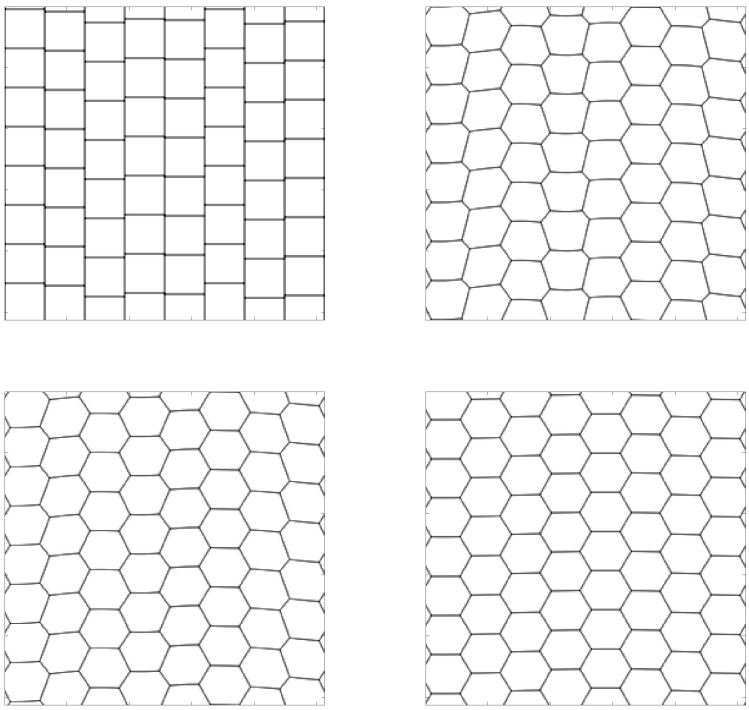

Figure 2: Initial condition: Randomly shifted 8 columns of 8 squares that have identical areas. Periodic boundary conditions.

minimal tilings in 3 dimensions.

5 Semi-Supervised Learning

Given a set of data points $\{x_1, \ldots, x_n\} = \mathcal{V} \subset \mathbb{R}^D$, a fixed collection of labels $\{1, \ldots, N\}$, and a small training subset $F \subset \mathcal{V}$ of points whose ground-truth labels are known, the semi-supervised learning (SSL) problem asks to correctly label the remaining points in $\mathcal{V} \setminus F$. Any solution to the problem is a partition $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_N)$ of $\mathcal{V}$ where Σ_i is the set of points that are assigned label i.

For many real life data sets, information about the sizes of the various classes is often available in advance. For example, the distribution of digits in postal codes and tax returns is very well-known. As a result, we will further assume that each phase should satisfy certain provided volume equality or volume bound constraints. As we will see in Section 5.3, incorporating class size information improves classification accuracy, especially when the training set is small (i.e. $|F| << |\mathcal{V}|$). Notably, there is still a marked improvement even when one can

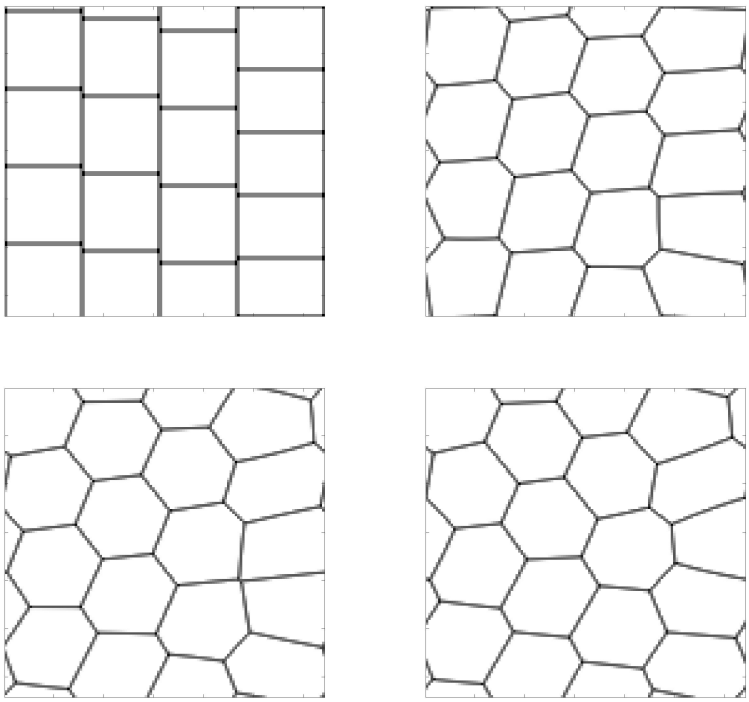

Figure 3: Initial condition: Randomly shifted 17 rectangles that have identical areas. Periodic boundary conditions. After a long time, there is still one phase with five and another with seven neighbors.

only estimate very rough size bounds.

5.1 Variational and graphical models

Variational models for the SSL problem find solutions by minimizing energies of the form

$$E(\mathbf{\Sigma}) = R(\mathbf{\Sigma}) + \mathrm{Fid}(\mathbf{\Sigma}) \tag{35}$$

where R is a regularizing term favoring "smooth" partitions and Fid is a penalty term which incorporates information from the training data F. We will modify this model slightly and only consider the regularizing term R. To incorporate the training data F and class size information we will simply impose the constraints

$$B_i \leq |\Sigma_i| \leq U_i, \quad F_i \subset \Sigma_i \quad \text{for all } 1 \leq i \leq N \tag{36}$$

where B_i and U_i are upper and lower bounds on the class sizes and $F_i \subset F$ is the set of training points labelled i.

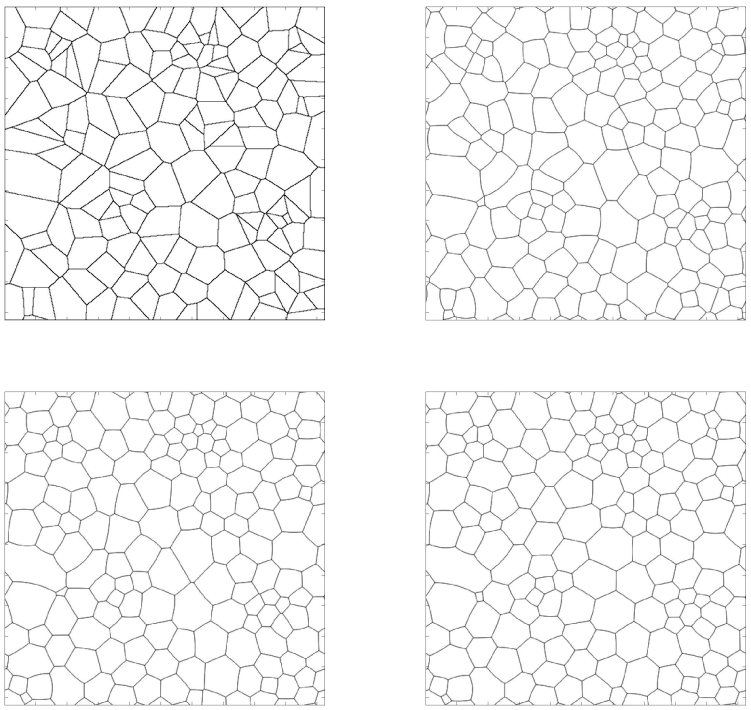

Figure 4: Initial condition: Voronoi diagram of 160 points taken uniformly at random on $[0,1]^2$. Periodic boundary conditions. Each phase preserves its initial area.

In order to define R, we need to give a notion of "smoothness" for partitions of $\mathcal{V}$. To do so, we give $\mathcal{V}$ the structure of a weighted graph $G = (\mathcal{V}, W)$. The weight matrix $W : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is a symmetric matrix where the entries $W(x,y)$ describe the how strongly the points x and y are connected. With this structure, R is typically taken to be some variant of the weighted graph cut:

$$\mathrm{Cut}(\mathbf{\Sigma}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{x\in\Sigma_i}\sum_{y\notin\Sigma_i} W(x,y), \tag{37}$$

which penalizes partitions which place strongly connected points in different classes. Combining the graph cut with the constraints (36) we will find solutions to the SSL problem by solving:

$$\arg\min_{\mathbf{\Sigma}} \frac{1}{2}\sum_{i=1}^{N}\sum_{x\in\Sigma_i}\sum_{y\notin\Sigma_i} W(x,y) \quad \text{s.t.} \quad F_i \subset \Sigma_i, \quad B_i \leq |\Sigma_i| \leq U_i. \tag{38}$$

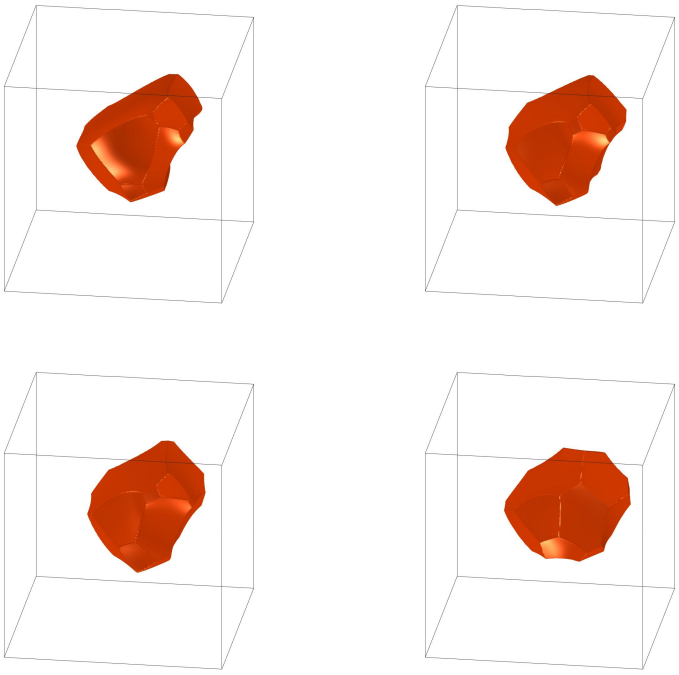

Figure 5: One "grain" from a total of 32. Initial condition: Voronoi diagram of 32 points taken uniformly at random on the 3-torus. Each phase preserves its initial volume.

5.2 Auction dynamics on graphs

There are many possible convex relaxations of the graph cut. We will consider the natural analogue of the heat content energy on graphs, the graph heat content (GHC)

$$\mathrm{GHC}(\boldsymbol{u}, W) = \sum_{i=1}^{N} \sum_{x,y \in \mathcal{V}} W(x,y) u_i(x)(1 - u_i(x)), \tag{39}$$

where $\boldsymbol{u} : \mathcal{V} \to \mathcal{K}_N$ is an element of the convex relaxation of the space of N-phase partitions of $\mathcal{V}$. In analogy to the continuum heat content, we may obtain a graph MBO scheme by successively minimizing linearizations of GHC [15, 23]. Up to a constant term, the linearization of (39) at a partition $\boldsymbol{\Sigma}$ is given by

$$\mathcal{L}_{\boldsymbol{\Sigma}}(\boldsymbol{u}) = \sum_{i=1}^{N} \sum_{x \in \mathcal{V}} u_i(x) \sum_{y \notin \Sigma_i} W(x,y). \tag{40}$$

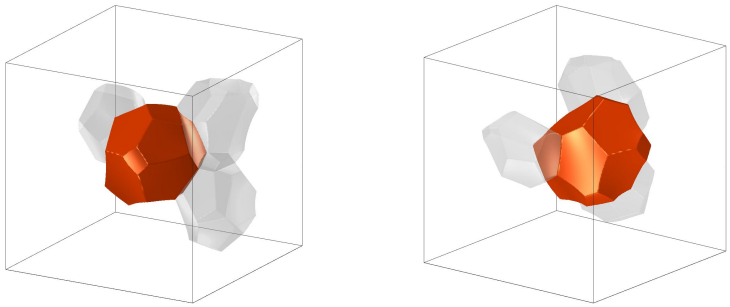

Figure 6: At final time, from a couple of other angles, with a few of its neighbors showing.

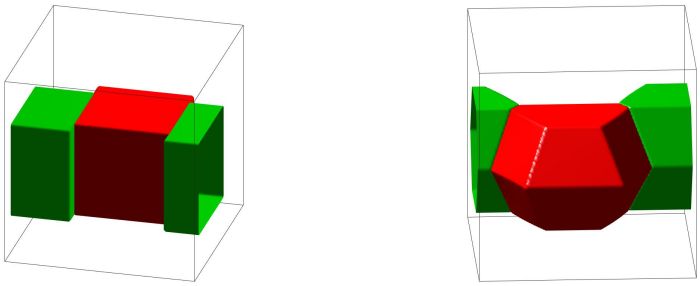

Figure 7: The initial and final configurations of the volume preserving flow on a randomly shifted cubic lattice. Each image shows two of the grains. The final configuration is fixed under the flow, but is not the global minimizer of the surface energy.

Our goal is to obtain a scheme which minimizes (38). As long as W is a positive semi-definite (PSD) matrix, GHC will be concave. Therefore, successively minimizing linearizations of (38) will dissipate the energy. The points $x \in F$ must have their labels fixed, so we only need to minimize the linearizations over $x \in \mathcal{V} \setminus F$. Thus, at every step we are led to solve:

$$\arg\min_{\boldsymbol{u}:\mathcal{V}\setminus\mathcal{F}\to\mathcal{K}_N} \sum_{i=1}^{N} \sum_{x\in\mathcal{V}\setminus F} \psi_i(x)u_i(x) \quad \text{s.t.} \quad B_i - |F_i| \le \sum_{x\in\mathcal{V}\setminus\mathcal{F}} u_i(x) \le U_i - |F_i| \quad (41)$$

where $\psi_i(x) = \sum_{y\notin\Sigma_i} W(x,y)$. Under some very simple transformations, the above problem is equivalent to the upper and lower volume bound assignment problem (28). Thus we may solve (41) using Algorithms 3 and 4.

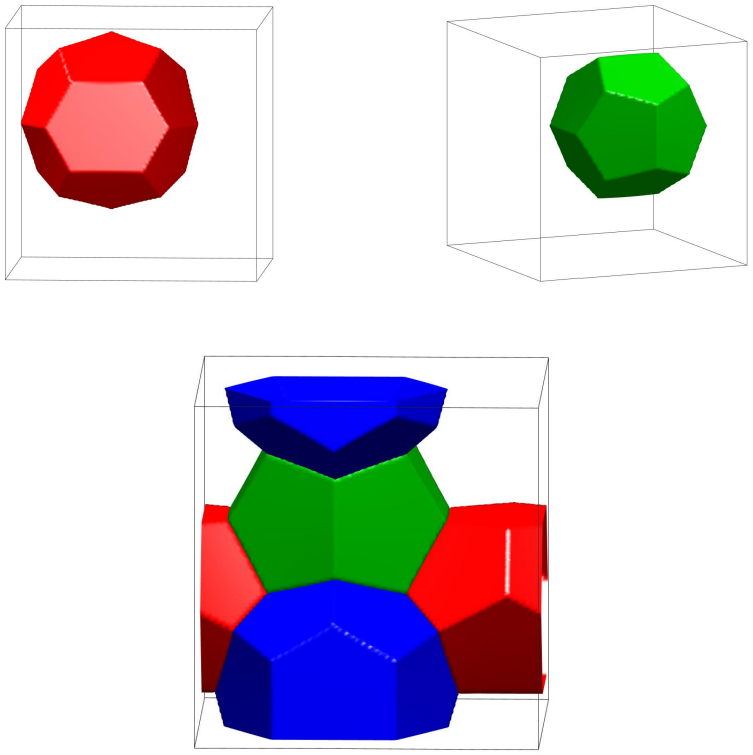

Figure 8: Running the flow on the 8 subunit cubic lattice with temperature fluctuations leads to the Weaire-Phelan structure. The Weaire-Phelan structure contains two distinct subunits shown in the first two images, the truncated hexagonal trapezohedron on the left and the pyritohedron on the right. The bottom image shows how 3 of the subunits fit together.

The set of data points $\mathcal{V}$ is finite and therefore necessarily compact. The energy (38) decreases with each iteration of auction dynamics. Combining compactness and monotonicity, we may conclude that the iterations eventually converge to a local minimum. Due to the concavity of GHC, we cannot guarantee that the local minimum is unique or independent of the initial condition. We can tackle this difficulty by incorporating temperature (as described in Section 3.4). The random temperature fluctuations allow the algorithm to escape local minima and find lower energy solutions. Nonetheless, experimental results show that auction dynamics finds high quality solutions even without temperature (c.f. Tables 5-8).

It remains to construct the weight matrix W. In the graph setting, W

Table 3: Benchmark datasets

Dataset	Dimension	Points	Classes	W construction timing (s)
MNIST	784	70,000	10	149.04
Opt-Digits	64	5,620	10	2.03
Three Moons	100	1,500	3	0.025

essentially plays the role of the convolution kernel K. This suggests setting $W(x, y) = f(|x - y|)$ for some decreasing function f. However, we do not want to connect all points x and y. The data points $\mathcal{V}$ typically cluster near a low dimensional manifold embedded in $\mathbb{R}^D$. To best reflect this manifold structure, we only connect the k nearest neighbors of each point and set the remaining entries of W to zero. Under these assumptions a popular choice for the weights are the Zelnick-Manor and Perona (ZMP) weight functions [49]:

$$W(x, y) = \exp\left(\frac{-|x - y|^2}{\sigma(x)\sigma(y)}\right) \tag{42}$$

where $\sigma(x), \sigma(y)$ are local scaling parameters for x, y respectively. We will take $\sigma(x) = |x - x_r|$ where x_r is the r^{th} nearest neighbor of x for some $r \in \{0, 1, \ldots, k\}$. In general, this construction will not produce a PSD matrix. Thus we take as our final weight matrix $W' = W^T W$.

5.3 Experimental results

To demonstrate the efficiency and accuracy of the auction dynamics approach to the SSL problem, we test against several benchmark machine learning datasets: Opt-Digits, MNIST, COIL, and Three Moons. We consider the performance of our algorithm both with and without temperature and with a wide range of class size constraints. All experiments were run using C code on a single processor core. k-nearest neighbors were calculated using the kd-tree code in the VLFeat library. Table 3 shows the timing information for VLFeat. Initial segmentations were computed using the Voronoi diagram construction described in [23]. All of our subsequent timing information in Table 4 includes the time required to initialize the segmentation and run the auction dynamics iterations.

On each dataset, we build the weight matrix using the ZMP construction detailed above and choose the nearest neighbor and scaling parameters k and r experimentally. The auction error tolerance, $\varepsilon_{\min}$, the scaling parameter, α, and the initial error term ε_0 are set to 10^{-7}, 4 and 0.1 respectively. Without temperature, we run auction dynamics either until convergence or until the relative energy change $\frac{|E_{k+1} - E_k|}{E_{k+1}}$ drops below 10^{-4}. If we introduce temperature, then we run a fixed number of iterations and extract the lowest energy configuration that was found.

5.3.1 Benchmark datasets

Here we detail the various datasets that we tested our algorithm against.

Opt-Digits: Opt-Digits is a database of 5620 handwritten digits [25]. The data is recorded as an 8×8 integer matrix, where each element is between 0 and 16. We construct the weight matrix using the 15 nearest neighbors and local scaling by the 7^{th} nearest neighbor.

MNIST: MNIST is a data set of 70,000 grayscale 28×28 pixel images of handwritten digits (0-9). Each of the digits is centered and size normalized. The data set is separated into 60,000 training images and 10,000 test images. We combine them to create a single set of 70,000 images to test against. We perform no preprocessing on the images. We construct the weight matrix using the 15 nearest neighbors with local scaling based on the 7^{th} nearest neighbor.

COIL: The Columbia Object Image Library (COIL-100) is a database of 128×128 pixel color images of 100 different objects photographed at various different angles [35]. In [36] the authors processed the COIL images to create a more difficult benchmark set. The red channel of each image is downsampled to 16×16 pixels by averaging over blocks of 8×8 pixels. The images are then further distorted and downsampled to create 241 dimensional feature vectors. Then 24 of the objects are randomly selected and randomly partitioned into 6 different classes. Discarding 38 images from each class leaves 250 images per class for a total of 1500 points. We construct the weight matrix using the 4 nearest neighbors and local scaling by the 4^{th} nearest neighbor.

Three Moons: The Three Moons synthetic data set consists of three half circles embedded into $\mathbb{R}^{100}$ with Gaussian noise. The standard construction is built from circles centered at $(0,0), (3,0), (1.5, 0.4)$ with radii of 1,1, and 1.5 respectively. The first two half circles lie in the upper half plane, while the third circle lies in the lower half plane. The circles are then embedded into $\mathbb{R}^{100}$ by setting the remaining 98 coordinates to zero. Finally, Gaussian noise with mean zero and standard deviation 0.14 is added to each of the 100 coordinates. We construct the dataset by sampling 500 points from each of the three circles, for a total of 1500 points. The weight matrix was built using the 15 nearest neighbors with local scaling by the 7^{th} nearest neighbor.

5.3.2 Results and comparison to other methods

In Tables 5-8, we present the results of our algorithm on Opt-Digits, MNIST, COIL and Three Moons. The algorithm is tested both with and without temperature and using several different volume bounds. We set the upper and lower bounds, $\boldsymbol{U}$ and $\boldsymbol{B}$ respectively, to be $B_i = V_i(1-x)$ and $U_i = V_i(1+x)$ where V_i is the ground truth volume of phase i and $x \in \{0, \frac{1}{10}, \ldots, \frac{4}{10}\}$. When temperature is used, we set $T = 0.1$. All reported results were averaged over 100 trials where F was chosen at random in each trial.

In general, we have observed that volume bounds increase the accuracy of the segmentation especially when the training set is extremely small. More notably, this advantage persists even when one can only estimate very rough volume bounds. For example, using only 0.05% of the training data on the MNIST dataset, we obtain a nearly 91.5% accuracy rate even when the upper and lower bounds deviate 30% from the true class size. We also see that incorporating temperature improves accuracy by finding lower energy solutions. Temperature is particularly effective in conjunction with volume bounds, as the bounds help ensure that the lower energy solutions are non-trivial.

A thorough comparison to other methods is presented in Tables 9 and 10. Our procedure achieves an accuracy that is better than or comparable with some of the best recent methods. A notable advantage of auction dynamics is that it is able to perform very well even with a very low number of labeled points. Indeed, we obtain high quality results at fidelity percentages that are out of reach for other state-of-the-art methods for the SSL problem.

Table 4: Timing (in seconds)

Data Set	Bounds Fid. %	1.0 1.0	1.2 0.8	1.4 0.6	no size constraints
MNIST	5%	**9.992** / 0.555	**7.501** / 0.339	**7.202** / 0.298	**7.180** / 0.279
	0.05%	**10.83** / 2.629	**9.353** / 1.857	**9.103** / 1.219	**8.033** / 0.637
OptDigits	20%	**0.741** / 0.022	**0.589** / 0.015	**0.585** / 0.014	**0.584** / 0.014
	0.4%	**0.862** / 0.035	**0.732** / 0.034	**0.723** / 0.034	**0.627** / 0.021
COIL	25%	**0.022** / 0.002	**0.019** / 0.001	**0.019** / 0.001	**0.019** / 0.001
	3%	**0.026** / 0.002	**0.022** / 0.002	**0.021** / 0.002	**0.019** / 0.001
Three Moons	5%	**0.09** / 0.005	**0.08** / 0.005	**0.08** / 0.005	**0.06** / 0.003
	0.25%	**0.09** / 0.008	**0.09** / 0.006	**0.08** / 0.007	**0.06** / 0.006

Bold= with temperature, not bold= without temperature

6 Conclusion

In this paper, we have derived a new, accurate and efficient method for computing volume-constrained curvature motion. Our method is derived from the variational formulation of threshold dynamics based on the heat content energy. Using the variational framework, we demonstrate a novel and surprising connection between volume constrained MBO schemes and the assignment problem. We then propose an efficient scheme for computing the motion based on specially developed variants of auction algorithms.

Our resulting scheme, auction dynamics, has many desirable properties. The interfaces are represented implicitly and thus topological changes are handled effortlessly. The volume constrained heat content energy is a Lyapunov functional for our scheme, thus we can guarantee unconditional gradient stability independently of the time step size. Our auction based approach ensures that

Table 5: Optdigits Results.

Bounds Fid. %	1.0 1.0	1.1 0.9	1.2 0.8	1.3 0.7	1.4 0.6	no size constraints
0.4%	**93.04%** 86.87%	**92.38%** 86.80%	**91.70%** 86.16%	**91.06%** 85.57%	**89.96%** 85.10%	**85.29%** 83.38%
0.5%	**95.96%** 91.76%	**95.18%** 91.06%	**94.66%** 90.39%	**93.84%** 89.87%	**93.06%** 89.31%	**89.76%** 87.98%
0.75%	**98.07%** 95.90%	**97.19%** 95.07%	**96.62%** 94.34%	**96.33%** 93.89%	**95.85%** 93.62%	**94.68%** 93.00%
1%	**98.39%** 97.11%	**97.57%** 96.24%	**97.14%** 95.69%	**96.91%** 95.40%	**96.75%** 95.26%	**96.33%** 95.04%

Bold= with temperature, not bold= without temperature

the volume constraints are satisfied exactly at every iteration, this allows our algorithm to be viable in situations where phase boundaries are rough or poorly resolved.

In addition, auction dynamics is highly flexible and can be used for a wide range of applications. We show how to adapt the algorithm to include random fluctuations due to temperature and solve segmentation problems on weighted graphs. In the application to the SSL problem, our algorithm is particularly effective. We are able to obtain highly accurate solution with training set sizes that are unprecedentedly small compared to other state-of-the-art methods. In the continuum setting, auction dynamics (particularly in conjunction with temperature) shows great promise as a tool for computing minimal partitions of space. We hope that the algorithm will prove to be useful for further exploration in this area.

Acknowledgments. Matt Jacobs and Selim Esedoḡlu were supported by NSF DMS-1317730.

References

[1] Jing An. Volume preserving threshold dynamics for grain networks. Technical report, University of Michigan, 2015.

[2] E. Bae and E. Merkurjev. Convex variational methods for multiclass data segmentation on graphs. *To appear in Journal of Mathematical Imaging and Vision*, 2017.

Table 6: MNIST Results

Bounds Fid. %	1.0 1.0	1.1 0.9	1.2 0.8	1.3 0.7	1.4 0.6	no size constraints
0.05%	**94.84%** 91.00%	**93.89%** 89.83%	**93.17%** 88.10%	**91.48%** 87.12%	**89.66%** 85.87%	**83.49%** 82.62%
0.075%	**96.42%** 94.65%	**95.83%** 93.31%	**94.93%** 91.95%	**94.02%** 90.99%	**92.95%** 90.53%	**90.72%** 89.40%
0.1%	**96.88%** 95.96%	**96.39%** 94.70%	**95.87%** 93.79%	**95.20%** 93.05%	**94.87%** 92.74%	**93.16%** 92.12%
0.2%	**97.28%** 96.85%	**96.92%** 96.15%	**96.79%** 95.99%	**96.70%** 95.88%	**96.64%** 95.84%	**96.54%** 95.83%
0.5%	**97.38%** 97.16%	**97.22%** 96.90%	**97.20%** 96.89%	**97.20%** 96.89%	**97.19%** 96.88%	**97.19%** 96.88%
1.0%	**97.43%** 97.31%	**97.31%** 97.18%	**97.31%** 97.17%	**97.31%** 97.16%	**97.31%** 97.15%	**97.30%** 97.15%

Bold= with temperature, not bold= without temperature

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006.

[4] Andrea L. Bertozzi and Arjuna Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling and Simulation*, 10(3):1090–1118, 2012.

[5] Dimitri Bertsekas. A distributed algorithm for the assignment problem. Technical report, MIT, May 1979.

[6] Dimitri Bertsekas. *Linear network optimization.* MIT Press, 1991.

[7] Dimitri Bertsekas. *Network Optimization: Continuous and Discrete Models.* Athena Scientific, 1998.

[8] Dimitri Bertsekas and David Castanon. The auction algorithm for the transportation problem. *Annals of Operations Research*, 20:67–69, 1989.

[9] Dimitri Bertsekas, David Castanon, and Haralampos Tsaknakis. Reverse auction and the solution of asymmetric assignment problems. *SIAM J. on Optimization*, 3:268–299, 1993.

Table 7: COIL Results

Bounds Fid. %	1.0 1.0	1.1 0.9	1.2 0.8	1.3 0.7	1.4 0.6	no size constraints
3%	**79.23%** 76.26%	**79.17%** 77.47%	**79.27%** 78.33%	**79.54%** 78.91%	**79.47%** 79.138%	**79.41%** 79.40%
4%	**85.05%** 82.70%	**85.10%** 83.80%	**85.17%** 84.53%	**85.24%** 84.96%	**85.13%** 85.01%	**85.12%** 85.10%
5%	**88.58%** 86.71%	**88.72%** 87.79%	**88.73%** 88.32%	**88.64%** 88.56%	**88.60%** 88.57%	**88.55%** 88.54%
10%	**93.73%** 93.12%	**93.73%** 93.68%	**93.73%** 93.73%	**93.72%** 93.71%	**93.73%** 93.73%	**93.74%** 93.74%

Bold= with temperature, not bold= without temperature

[10] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1237–1242, 2011.

[11] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Mach. Learn.*, 46(1):161–190, 2002.

[12] M. Elsey and S. Esedoglu. Threshold dynamics for anisotropic surface energies. *AMS Mathematics of Computation*, 2016.

[13] S. Esedoḡlu and F. Otto. Threshold dynamics for networks with arbitrary surface tensions. *Communications on Pure and Applied Mathematics*, 68(5):808–864, 2015.

[14] S. Esedoḡlu and Y.-H. Tsai. Threshold dynamics for the piecewise constant Mumford-Shah functional. *Journal of Computational Physics*, 211(1):367–384, 2006.

[15] Selim Esedoḡlu and Matt Jacobs. Convolution kernels, and stability of threshold dynamics methods. *SIAM Journal on Numerical Analysis*, 55(5):2123–2150, September 2017.

[16] Selim Esedoglu, Matt Jacobs, and Pengo Zhang. Kernels with prescribed surface tension and mobility for threshold dynamics schemes. *Journal of Computational Physics*, 337(15):62–83, May 2017.

[17] C. Garcia-Cardona, E. Merkurjev, A. L. Bertozzi, A. Flenner, and A. G. Percus. Multiclass data segmentation using diffuse interface methods on

Table 8: Three Moons Results.

Bounds Fid. %	1.0 1.0	1.1 0.9	1.2 0.8	1.3 0.7	1.4 0.6	no size constraints
0.25%	**92.38%** 88.06%	**92.68%** 91.01%	**90.91%** 87.87%	**88.20%** 87.73%	**86.40%** 86.08%	**84.52%** 85.70%
0.5%	**97.66%** 94.84%	**97.64%** 94.31%	**94.80%** 93.16%	**92.79%** 92.24%	**90.29%** 91.38%	**90.22%** 90.97%
0.75%	**98.54%** 96.74%	**98.09%** 96.20%	**95.86%** 94.95%	**94.85%** 93.66%	**94.16%** 93.52%	**93.53%** 93.04%
1%	**98.80%** 97.90%	**98.22%** 97.10%	**96.62%** 95.99%	**95.34%** 95.61%	**95.04%** 95.26%	**94.04%** 94.53%

Table 9: Accuracy Comparison to Other Methods

MNIST (supervised approaches)

Method	Accuracy
boosted stumps* [26, 30]	92.3-98.74%
k-nearest neighbors* [29, 30]	95.0-97.17%
neural/conv. nets* [29, 10, 30]	95.3-99.65%
nonlinear classifiers* [29, 30]	96.4-96.7%
SVM* [29, 11]	98.6-99.32%
Proposed (55% fidelity)	99.14%

- Note that algorithms, marked by *, use substantially more data for training.

graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1600–1613, 2014.

[18] C. Garcia-Cardona, E. Merkurjev, A.L. Bertozzi, A. Flenner, and A.G. Percus. Multiclass data segmentation using diffuse interface methods on graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1600–1613, 2014.

[19] A.V. Goldberg. Solving minimum-cost flow problems by successive approximation. In *STOC 87*, November 1986.

[20] A.V. Goldberg. Efficient graph algorithms for sequentialand parallel computer. Technical report, Laboratory for Computer Science, M.I.T., 1987.

[21] A.V. Goldberg and R.E. Tarjan. Solving minimum cost flow problems by successive approximation. In *Proc. 19th ACM STOC*, May 1987.

[22] Thomas Hales. The honeycomb conjecture. *Discrete and Computational Geometry*, 25(1):1–22, 2001.

[23] M. Jacobs. A fast MBO scheme for multiclass data classification. In *Sixth International Conference on Scale Space and Variational Methods in Computer Vision*, 2016.

[24] T. Joachims et al. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning*, volume 20, page 290, 2003.

[25] C Kaynak. Methods of combining multiple classifiers and their applications to handwritten digit recognition. Master's thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University, 1995.

[26] B. Kégl and R. Busa-Fekete. Boosting products of base classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 497–504, 2009.

[27] Harold Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2, 1955.

[28] T. Laux and D. Swartz. Convergence of thresholding schemes incorporating bulk effects. *ArXiv e-prints*, January 2016.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] Y. LeCun and C. Cortes. The MNIST database of handwritten digits.

[31] Peter McMullen and Egon Schulte. *Abstract Regular Polytopes*. Macmillan, 2002.

[32] E. Merkurjev, J. Sunu, and A. L. Bertozzi. Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video. In *Proceedings of the International Conference on Image Processing*, pages 689–693, 2014.

[33] B. Merriman, J. K. Bence, and S. J. Osher. Diffusion generated motion by mean curvature. In J. Taylor, editor, *Proceedings of the Computational Crystal Growers Workshop*, pages 73–83. AMS, 1992.

[34] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 1957.

[35] S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image library (coil-100). Technical report, Columbia University, 1996.

[36] Alexander Zien Olivier Chapelle, Bernhard Scholkopf. *Semi-Supervised Learning*. The MIT Press, 2006.

[37] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulation. *Journal of Computational Physics*, 79:12–49, 1988.

[38] D. Peng, B. Merriman, S. Osher, H.-K. Zhao, and M. J. Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[39] S. J. Ruuth. *Efficient algorithms for diffusion-generated motion by mean curvature*. PhD thesis, The University of British Columbia, 1996.

[40] S. J. Ruuth. A diffusion generated approach to multiphase motion. *Journal of Computational Physics*, 145:166–192, 1998.

[41] S. J. Ruuth. Efficient algorithms for diffusion-generated motion by mean curvature. *Journal of Computational Physics*, 144:603–625, 1998.

[42] S. J. Ruuth and B. Wetton. A simple scheme for volume-preserving motion by mean curvature. *Journal of Scientific Computing*, 19(1):373–384, 2003.

[43] R. I. Saye and J. A. Sethian. The voronoi implicit interface method for computing multiphase physics. *Proceedings of the National Academy of Sciences*, 108:19498–19503, 2011.

[44] A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12:3311–3370, 2011.

[45] Y. van Gennip, N. Guillen, B. Ostimg, and A. L. Bertozzi. Mean curvature, threshold dynamics, and phase field theory on finite graphs. *Milan Journal of Mathematics*, 82:3–65, 2014.

[46] D. Weaire and R. Phelan. A counter-example to kelvin's conjecture on minimal surfaces. *Philosophical Magazine Letters*, 69:107–110, 1994.

[47] X. Xu, D. Wang, and X. Wang. An efficient threshold dynamics method for wetting on rough surfaces. *arXiv:1602.04688*, February 2016.

[48] K. Yin, X.-C. Tai, and S. Osher. An effective region force for some variational models for learning and clustering. *UCLA CAM Report*, pages 16–18, 2016.

[49] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *Advances in neural information processing systems*, 2004.

[50] H.-K. Zhao, B. Merriman, S. Osher, and L. Wang. Capturing the behavior of bubbles and drops using the variational levl set approach. *Journal of Computational Physics*, 143:495–512, 1998.

Table 10: Accuracy Comparison to Other Methods

MNIST

Method/ % Labeled Nodes	0.25%	0.5%	1.0%
TVP [48]	83.7%	86.3%	90.8%
multiclass MBO [17]	73.0%	90.1%	94.9%
LapRF ($m = 1$) [48]	84.2%	90.9%	95.1%
TVRF ($m = 1$) [48]	93.4%	96.4%	96.8%
LapRF ($m = 2$) [48]	91.0%	94.2%	95.6%
TVRF ($m = 2$) [48]	94.6%	96.6%	96.7%
Proposed (No constraints)	**96.68%**	**97.18%**	**97.30%**
Proposed (Exact volume constraints)	**97.32%**	**97.38%**	**97.43%**

OptDigits

Method/ % Labeled Nodes	0.89%	1.78%	2.67%
LapRLS [3, 44]	92.3%	97.6%	97.3%
sGT [24, 44]	91.4%	97.4%	97.4%
SQ-Loss-I [44]	95.9%	97.3%	97.7%
MP [44]	94.7%	97.0%	97.1%
LapRF ($m = 1$) [48]	79.0%	95.2%	96.8%
TVRF ($m = 1$) [48]	95.9%	97.2%	98.3%
Proposed (No constraints)	**95.39%**	**97.74%**	**98.12%**
Proposed (Exact volume constraints)	**98.31%**	**98.64%**	**98.72%**

COIL

Method/ % Labeled Nodes	3.3%	6.7%	10%
multiclass MBO [18]	72.9%	85.4%	91.5%
convex method [2]	72.7%	85.2%	93.4%
LapRLS [3, 44]	78.4%	84.5%	87.8%
sGT [24, 44]	78.0%	89.0%	89.9%
SQ-Loss-I [44]	81.0%	89.0%	90.9%
MP [44]	78.5%	90.2%	91.1%
LapRF ($m = 1$) [48]	71.7%	87.0%	91.0%
TVRF ($m = 1$) [48]	80.3%	90.0%	91.7%
Proposed (No constraints)	**81.50%**	**91.21%**	**93.63%**
Proposed (Exact volume constraints)	**81.57%**	**91.41%**	**93.73%**

Three Moons

Method/ % Labeled Nodes	1.66%	3.33%	5%
multiclass MBO [17]	68.3%	84.1%	94.3%
LapRF ($m = 1$) [48]	95.1%	96.4%	98.1%
TVRF ($m = 1$) [48]	96.4%	98.2%	98.4%
LapRF ($m = 2$) [48]	96.4%	97.9%	98.5%
TVRF ($m = 2$) [48]	96.4%	98.2%	98.6%
Proposed (No constraints)	**97.46 %**	**98.49%**	**98.79%**
Proposed (Exact volume constraints)	**99.34%**	**99.48%**	**99.51%**