A learning heuristic for space mapping and searching self-organizing systems using adaptive mesh refinement

Carolyn L. Phillips[a]

[a]*Mathematics and Computer Science Division*
Argonne National Laboratory, Lemont, IL, 60439, USA

Abstract

In a complex self-organizing system, small changes in the interactions between the system's components can result in different emergent macrostructures or macro-behaviors. In chemical engineering and material science, such spontaneously self-assembling systems, using polymers, nano- or colloidal scale particles, DNA, or other precursors, are an attractive way to create materials that are precisely engineered at a fine scale. Changes to the interactions can often be described by a set of parameters. Different contiguous regions in this parameter space corresponds to different ordered states. Since these ordered states are emergent, often experiment, not analysis, is necessary to create a diagram of ordered states over the parameter space. By issuing queries to points in the parameter space, e.g performing a computational or physical experiment, ordered states can be discovered and mapped . Queries can be costly in terms of resources or time. In general, one would like to learn the most about a space for the fewest total number of queries. Here we introduce a learning heuristic for issuing queries to map and search a two-dimensional parameter space. Using a method inspired by the adaptive mesh refinement method, the heuristic iteratively issues batches of queries to be executed in parallel, based on what has been learned from previous iterations. By adjusting the search criteria of the heuristic, different types of searches (e.g. a uniform search, exploring boundaries, sampling all regions equally) can be flexibly implemented. We show that this method will densely search the space in the limit of infinite queries while preferentially targeting certain features of space. Using numerical examples, including a study simulating the self assembly of complex crystals, we show how this heuristic can discover new regions and map boundaries more accurately than a uniformly distributed set of queries.

Keywords: self-assembly, optimization, adaptive mesh refinement

1. Introduction

Systems of simple components that adaptively self-organize in response to small changes in their environment have been an area of active research in many fields for decades[1, 2]. In material science and chemical engineering, self-assembly offers a novel avenue for engineering materials with precise structure. The principles that govern how to predict and control the self-assembly process, e.g., the design rules for creating a system that organizes into a desired structure[3], are still being discovered and explored.

The resultant large-scale structure often results from the balance of many small-scale interactions. Complex structures can arise from the frustration between competing interactions. Thus the emergent structure can be exquisitely sensitive to small changes. For example, Shevchenko et al.[4] show that the structural diversity of binary nano particle superlattices results from varying the particle size ratio, tuning the electrical charge, and adjusting the relative concentration of the two species. Noorduin et al.[5] demonstrate experimentally that sequential modulations of environmental conditions, e.g. CO_2 concentration, pH, and temperature, control the directional structure growth of carbonate-silica microstructures, from blossoming to form stems, vases, and corals, to curling to form spiral and leaf structures.

When the interactions between the components can be engineered[6], experimentation can be the only way to determine how the large-scale emergent structure is impacted. The engineer-able degrees of freedom of a system define a continuous parameter space and each experiment is a single point in that space. It is generally far simpler to vary parameters in a computational simulation of a complex system than in an physical realization. Such simulations are still functionally experiments because the outcome of each set of parameters can usually not be predicted analytically. Such simulations are useful for exploring a parameter space, generating intuition and design principles as to how competing forces interact, and guiding experimental effort to the more fruitful regions of parameter space. For example, Engel and Trebin[7] identify parameters where geometrical frustration causes two-dimensional quasicrystals to self-assemble. Nguyen et al.[8] show that small changes in a model of a swarm of self-propelled particles causes the swarm to change form from a ball, to a ring, to a torus, to a shell shape. The torus occurs in a small region of the parameter space. And Vernizzi et al.[9] find that by changing the relative fraction of two components that compose an elastic shell, irregular and regular polyhedra arise spontaneously from buckling of the membrane. They identify a parameter regime where less symmetric shells form that are distinct from the more standard icosahedral or spherical shells.

Experiments, even computational experiments, have a cost in time and resources. Given a parameter space of interest and a experimental method of querying points in the space[1], how should such queries be distributed to maximize the amount of information gained about contiguous regions of ordered structures? Commonly, a two-dimensional parameter space is considered and queries are distributed on a uniform grid[7–11]. However, this distribution strategy means, first, that each region is sampled in proportion to its fractional area, regardless of the interest in the region, and second, the cost of the experiment, and therefore the number that can be performed, uniformly determines the resolution of the distribution of queries. Even though, in practice, information from completed experiments arrives staggered in time, often in batches, information from already completed experiments is not used to improve the exploration of the space.

Here we propose a heuristic for iteratively distributing experiments, or queries, in a parameter space that learns from already completed experiments. This heuristic is inspired by the adaptive mesh refinement (AMR) method used in some finite element method simulations. The objective of the heuristic is to increase the resolution of information about small regions, or region of special interest, improve the boundary resolution between regions, and/or search for new regions along boundaries or in unexplored space, depending on how the search heuristic is implemented. That is, the heuristic is flexible and the search type can be adjusted, even between iterations of the search.

This method differs from other space searching methods in that the nature of the space being searched is different. Methods that use surrogate models, such as Kriging[12], try to model a continuous response surface over a parameter space. Evolutionary algorithms attempt find the minimum of a function in a high dimensional space[13]. In the case of this problem, there is no explicit continuous differentiable function defined over the space. Rather, the function over the space is a classification that subdivides the space into contiguous regions that share a discrete experimental outcome. Also, the objective is not to find a peak, but to discover the regions that are present.

In Section II, we briefly review Adaptive Mesh Refinement and then introduce our space searching heuristic. We introduce several measures for ranking the elements of a mesh. In Section III, we show that, in the limit, this heuristic densely distributes queries in the search space and thus, every contiguous region in the space is guaranteed to be found. In Section IV, we provide numerical examples to show how changing the weights for the different measures creates different kinds of searches. Using five cases, we perform a sensitivity analysis over the weighting of three terms. We then apply the method to an example of a space-searching problem involving the self-assembly of Lennard-Jones-Gauss particles into 2D crystals. In Section V, we provide concluding remarks.

[1]A query could be an experiment, a simulation, consulting a database, or any other action that classifies a parameter point.

2. A Learning Heuristic

First we discuss the assumptions used in this paper. We will assume that a parameter space can be subdivided into a finite number of simple connected contiguous regions. Each point in the parameter space represents a legal combination of parameters. A query performed at a point returns a label, or a classification of the ordered structure formed by that particular combination of parameters. A query at a given point in the parameter space will always return the same label. The cost of each query is independent of the other queries. We also assume that queries are expensive, that is, the cost of simply finely distributing queries uniformly over the entire space is prohibitive.

2.1. Adaptive Mesh Refinement

Adaptive Mesh Refinement (AMR)[14, 15] is a method developed for generating numerical solutions of partial differential equations that is commonly used for problems in computational fluid dynamics, hydrodynamics, astrophysics, and finite element applications. In AMR, a space or object is subdivided into a disjoint set of mesh elements. The mesh elements can be triangles or quadrilaterals in 2D, or tetrahedrons, pyramids, prisms, or hexahedrons in 3D, but are generally always convex polytopes. Using an integrator, sets of real or complex values of functions are calculated at the vertices or nodes of the mesh. In regions where higher numerical resolution is desired, inserting additional nodes refines the mesh. Larger mesh elements are replaced by smaller mesh elements. Standardly, AMR refers both to the algorithms that generate and refine the computational mesh and the integration method that advances the partial differential equation values at each node. A large literature exists of different algorithms for the initial mesh generation and subsequent local refinement of the mesh. We propose a simple adaptive search heuristic that uses the mesh generation and refinement algorithms (but not the integration methods) of AMR to dynamically increase the density of queries in the regions of higher interest. Given a contiguous region of parameter space to be searched, the heuristic is briefly described as follows. (1) An initial coarse conforming mesh is generated over the space, subdividing the space into mesh elements. (2) A query is issued for each unlabeled node of the mesh and, upon return of the query, each unlabeled node is assigned a label. (3) A scoring function is applied to the set of mesh elements. (4) A set of highest ranked mesh elements are selected for refining, generating a new set of unlabeled nodes. (2)-(5) are repeated until a stopping criterion is reached. The outline of the heuristic is sufficiently general that there are many possible implementation choices with respect to mesh type, initial mesh generation, mesh refinement method, the scoring function, and number of elements to be refined per iteration. In the remainder of this section we introduce one example implementation of this heuristic in a two-dimensional parameter space.

2.2. Mesh Generation and Refinement

Let the parameter space to be explored be described as Ω, a bounded polygonal region where all parameters have been normalized to be bounded from -1 to 1. Given the predefined geometry of the parameter space to be mapped, and a desired number of initial queries, N, an unstructured mesh of triangles is generated by a Delaunay triangulation algorithm[16]. Nodes are placed on the edges of the parameter space, and then the edges are triangulated. Nodes are then inserted into the centers of circumscribed circles of large triangles until no triangle edge exceeds a hmax, a prechosen positive constant. At the initiation of the mesh, an hmax value is selected that is consistent with the placement of N total nodes. A refinement of the mesh is performed as follows. A set of triangular mesh elements is selected for refinement. For this set, a regular refinement scheme is used. That is, for each triangle, a new node is inserted at the midpoint of each edge, which are then connected by three new edges. Thus the original triangle is replaced by four similar triangles. This step generates hanging nodes, that is, a node that is on the edge of two cells but does not properly belong to one of them. A longest-edge-bisection scheme[17, 18] is applied to neighboring triangles until all hanging nodes are removed. Thus, after refinement, the intersection of two non-disjoint, non-identical triangles consists of either a common vertex or a common side and all mesh elements are triangles. The regular and longest-edge-bisection refinement schemes combined guarantee that the smallest angle in the mesh is bounded to half the smallest angle of the original triangulation.[17, 18] Thus the individual triangles do not deviate too far from equilateral triangles. By using a mesh refinement scheme that generates no hanging nodes, the data structure of each mesh element remains fixed over all iterations. Each mesh element is a triangle with three nodes and three edges. The disadvantage of this mesh refinement scheme is that the number of new nodes generated after each refinement step is not a fixed number relative to the set of triangles initially chosen to be refined. We choose to handle this by adding elements to the list to be regularly refined, until the resultant total refinement generates more than a threshold number of nodes. An example of a refinement scheme applied to a triangular mesh is shown in Figure 1. On the top is an example of an unstructured triangular mesh that has been generated for a square parameter space. The red triangles have been selected for refinement. On the bottom, all new nodes are circled. Each red triangle has been split into four similar triangles and a longest-edge-bisection scheme has been used to resolve hanging nodes in neighboring triangles. Per reference [18], the additional refinements to resolve hanging nodes are guaranteed to terminate.

This heuristic was implemented using the mesh initialization and refinement software in the MATLAB PDE Toolbox[19]. In a test case, an implementation generated 2,611 queries over 50 iterations (returning 52.2 points per iteration on average) in 4.17 minutes. Thus this step is not expected to be rate limiting relative to the time to complete a query to label a node.

2.3. Scoring Function

In this section, we discuss a method for selecting a batch of mesh elements to refine. This selection uses a scoring function. Given the nodes of a mesh element and their labels, the scoring function returns a scalar number that indicates how desirable it is to refine the mesh element. That is, given a mesh element T, the scoring function is,

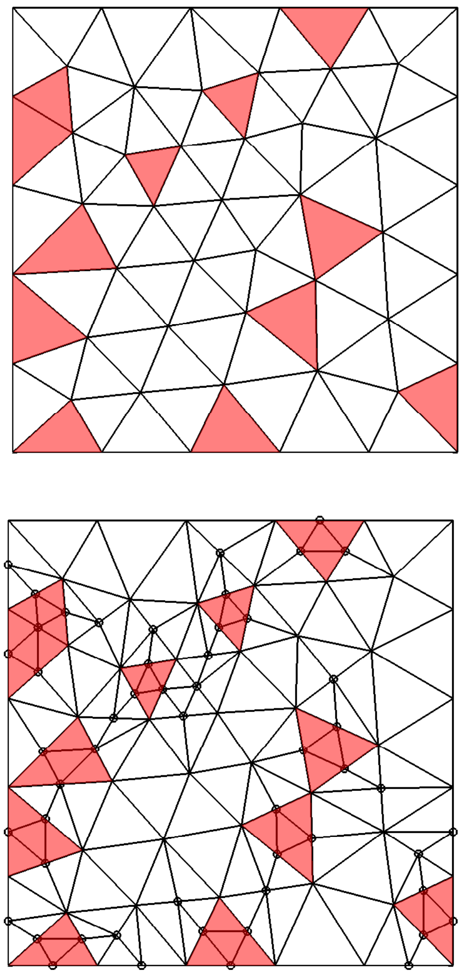


Figure 1: (Top) An example triangulated mesh for a square parameter space. The red triangles have been selected for refinement. (Bottom) The red triangles have been refined. The new nodes are circled.

$$\lambda(T) = \mathcal{F}(\{l\}, \{p\}), \tag{1}$$

where $\{l\}$ and $\{p\}$ are the set of labels and coordinates of the nodes of T, respectively, and where, the more desirable it is the refine the mesh element, the higher the score $\lambda(T)$.

The scoring function can be tailored to score different aspects of a given mesh element. We present several possible scoring function terms, area, number of labels, novelty, and minimum diversity, which can be weighted to create different kinds of searches.

$$\mathcal{F}(\{l\}, \{p\}) = \frac{w_A A(\{p\})}{A_{min}} + \frac{w_L N(\{l\})}{N_{max}} + \frac{w_F F(\{l\})}{F_{max}} + w_{MD} M(\{l\}). \tag{2}$$

2.3.1. Area

The first term is equal to the area of a mesh element, $A(\{p\})$, scaled by the area of the smallest mesh element that is present in the mesh, A_{min}, and weighted by a factor w_A.

2.3.2. Number of Labels

The second term is equal to the number of unique labels contained in the label list $\{l\}$ of a mesh element, and is normalized by the current largest number of unique labels, N_{max}, (where, for triangular mesh elements, $1 \leq N_{max} \leq 3$). The term is weighted by w_L.

2.3.3. Novelty

The third term is a measure of the novelty of the labels of the label list of a mesh element,

$$F(\{l\}) = log\left(\sum_{j=1}^{3} \frac{1}{f(l_j)}\right), \tag{3}$$

where l_j, for $j = 1, 2, 3$, are the three labels of $\{l\}$ and $f(l_j)$ is the frequency of the label l_j over all the nodes of the mesh. This term is normalized by the largest novelty value calculated over all the mesh elements, F_{max}, and weighted by w_F.

2.3.4. Minimal Diversity

The fourth term is a Boolean that measures whether the labels of the mesh element satisfy a minimal level of unique label diversity,

$$M(\{l\}) = \begin{cases} 1 & \text{if } N(\{l\}) > 1 \\ 0 & \text{if } N(\{l\}) \leq 1 \end{cases} \tag{4}$$

and is weighted by w_{MD}.

3. Mathematical Analysis

In this section we will show that as long as the weighting term, $w_A > 0$, the method described in Section II is sufficient to generate a dense set of node points in the parameter space in the limit of the number of node points approaching infinity.

Let Ω be a bounded polygonal region with an initial conforming triangulation of N triangles. Since each refinement of a triangular mesh element is a nested refinement, that is, each refinement is contained inside a single prior refinement at each refinement level; we can construct a forest, i.e., a set of tree graphs, that describes the state of the mesh. We construct one tree for each of the mesh elements of the initial triangulation of Ω. Consider a triangular mesh element $T_0^j, j \in [1, .., N]$ of the initial triangulation. The root of the tree is T_0^j. Each vertex in the tree is a triangular mesh element nested inside T_0^j. Each vertex, except the root, is connected to its preceding triangular mesh element (the triangle from which it was directly refined) by an edge. If it is not a leaf, it is also connected to t vertices representing the triangular elements that result from a refinement of the element. (For regular refinements $t = 4$, for longest-edge-bisection refinements, $t = 2$). Thus, the leaves of the tree represent the current state of the refinement of the original element T_0^j. Figure 2 depicts an example refinement and tree. A triangular mesh element T_0 is shown at three stages of refinement, the initial mesh element, a regular refinement of the initial mesh element, and regular refinement of only the bottom

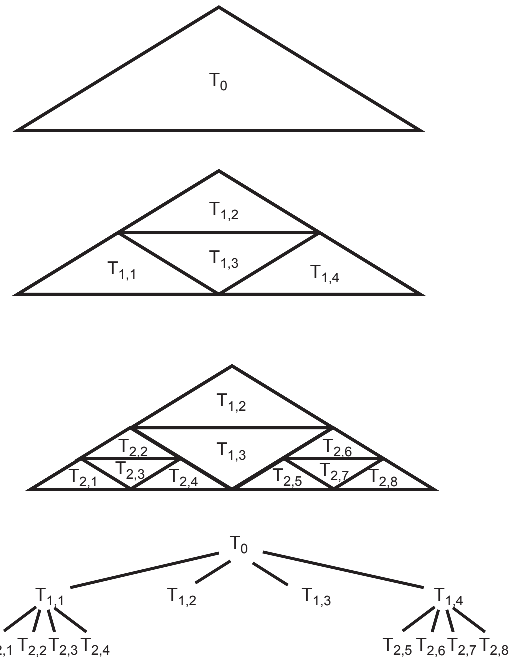


Figure 2: (top) A diagram depicting the three stages of the refinement of a triangular mesh element T_0 and (bottom) the tree that describes the refinement.

two elements $T_{1,1}$, and $T_{1,4}$. The leaves of the tree at the bottom of Figure 2 describe the current state of the mesh element.

Each point p in T_0^j, that is not a node of a triangular mesh element can be found in one and only one leaf of the tree. Let each vertex of the tree be labeled $T_{n,i}^j$ where n is the distance (i.e. number of refinements) from the root node and i is an arbitrary unique index. When a triangle is refined into sub-triangles, the sub-triangles are equal in area, and the ratio of the area of each sub-triangles relative to the original triangle is f, where, $f = 0.5$ and $f = 0.25$ for a longest-edge-bisection refinement scheme and a regular refinement, respectively.

Let $S = \{s_1, s_2, s_3,\}$ be a sequence of triangular mesh elements that are selected to be refined in the order of the sequence, $s_k \in T_{n,i}^j$. Note that in the generation the sequence S, a member of the sequence is only required to exist in the mesh after the refining of the prior members of the sequence. (For completeness, each member of sequence is associated with a refinement method, regular or longest-edge-bisection, to be used for that element.) Let Ψ^k be the set of all the leaves of all the trees after the refinement of element s_k of S. Ψ^k uniquely describes the current state of the refinement.

We introduce the following condition for a sequence S:

Definition 3.1. Largest Triangle Condition *If T is the mesh element with the largest area after the refining of mesh element index s_k of the sequence S, then T will be element $s_{k'}$ of the sequence, where $k < k' < k + K$, and K is a positive integer. If the triangular element T with the largest area is not unique, then one member of the set of the largest elements will be in the sequence with the index k'.*

Theorem 3.2. *If the next triangular mesh element s_{k+1} is selected to be the element that scores the highest per the scoring function Eqn.2 after the refinement of s_k, and $w_A > \gamma > 0$, then the Largest Triangle Condition is satisfied.*

Proof. We provide a proof by contradiction. After the refinement of s_k of a sequence S, assume that A is the maximum area of all the mesh elements of Ψ^k and Γ is the set of all triangular mesh elements T, $T \in \Psi^k$, of area A, thus $\Gamma \subseteq \Psi^k$. Assume that no element $T \in \Gamma$ is found in the sequence S as $k \to \infty$. Or, equivalently $\Gamma \subseteq \Psi^{k'}, \forall k' \geq k$. This means that for all $\Psi^{k'}, k' \geq k$, there must exist at least one other triangular mesh element T' with an area $A' < A$ such that, $\forall T \in \Gamma$, $\lambda(T') > \lambda(T)$, or,

$$\frac{w_A A'}{A_{min}} + \frac{w_L N'}{N_{max}} + \frac{w_F F'}{F_{max}} + w_{MD} M' > \tag{5}$$

$$\frac{w_A A}{A_{min}} + \frac{w_L N}{N_{max}} + \frac{w_F F}{F_{max}} + w_{MD} M \tag{6}$$

which can be rearranged to

$$\frac{w_L (N' - N)}{N_{max}} + \frac{w_F (F' - F)}{F_{max}} + w_{MD}(M' - M) > \tag{7}$$

$$\frac{w_A (A - A')}{A_{min}} \tag{8}$$

The right side of the inequality must be less than the upper bound of the left side, so,

$$w_L + w_F + w_{MD} > \frac{w_A (A - A')}{A_{min}} \tag{9}$$

Let $A_{m'}$ be the area of largest triangular element of Ψ^k that is not contained in Γ. Let $A_0 = max(fA, A_{m'})$, that is, the larger of the area of a triangle in a refinement of a largest triangular element of Ψ^k or the area of the second largest triangular element of Ψ^k. Because the maximum area of all the triangles nested in an initial triangle is a decreasing function as k' increases, the following must also hold

$$w_L + w_F + w_{MD} > \frac{w_A (A - A')}{A_{min}} > \frac{w_A (A - A_0)}{A_{min}} \tag{10}$$

And therefore, for $k' \geq k$,

$$A_{min} > \frac{w_A (A - A_0)}{w_L + w_F + w_{MD}} \tag{11}$$

or there is a lower bound on the area of the smallest mesh element as $k \to \infty$ and because $w_A > \gamma > 0$, the lower bound is bounded away from zero. Let A_{lb} equal the right side of Eqn.11.

We consider the initial set of a triangular mesh elements $\{T_0^j\}, j \in [1, .., N]$, each with area A^j and the tree that result from uniformly and indefinitely refining these elements . The triangular element T_n^j has area less than or equal to $f^n A^j$, where $f = 0.5$. Let $\max(\{A^j\})$ be the maximum initial area of the initial triangulation. For $m > \ln\left(\frac{A_{lb}}{max(\{A^j\}}) \right) / \ln(f)$, all mesh elements $T_{m,i}^j$ have areas less than A_{lb}. We note that m is positive and finite. Each refinement of a single mesh element produces no more than $t = 4$ new elements. Thus there is a finite number of vertices $\{T_{m',i}^j\}$ in the set of trees such that $m' < m$. Specifically, the size of the set $\{T_{m',i}^j\}$, such that $m' < m$ is less than $N \sum_i^{m-1} 4^i < N(m-1)4^{m-1}$. Thus if $k - k' > N(m-1)4^{m-1}$, then all triangular elements of the mesh must have a smaller area A_{lb}. This contradicts Eqn.11.

Therefore, an element $T \in \Gamma$ must be in the sequence S, $T = s'_k, k < k' < k + N(m-1)4^{m-1}$.

Next we need to show that it is not necessary that every triangular element added to the sequence S to be the highest scoring element for the Largest Triangle Condition to hold. For example, triangular elements can be added to S to remove hanging nodes.

Corollary 3.3. *Assume that the sequence S is constructed as follows. For $0 < K < K_{max}$, K triangular mesh elements are selected for the sequence S, $\{s_{k+1}, ..., s_{k+K}\}$ such that K' of the elements, $0 < K' \leq K$ are the K' elements that score the highest per the scoring function Eqn (2.1) after the refinement of s_k. The remaining $K - K'$ mesh elements are selected by any arbitrary criteria. This method of constructing the sequence S also satisfies the Largest Triangle Condition as long as $w_A > \gamma > 0$.*

Proof. This holds trivially. If Γ is the set of elements with the largest area after the refinement of s_k, then, per the argument above, after no more than $N(m-1)4^{m-1}$ triangular mesh elements have been selected for the sequence, T has either been selected or T must have the highest score and will be in the set of the K' highest-scoring elements selected in the next set of K elements.

Now we will show that continually refining the triangular mesh element that contains a point p will generate an arbitrarily close node point to p. Let p be a point in a triangular mesh element T that has area A. Per references 4, 5, the refinement scheme used guarantees the minimum angle is bounded away from zero.

Theorem 3.4. *If the minimum angle of T is α, then an upper bound on the distance between the point p and a node of the triangle is $\sqrt{2Asin(\pi - 2\alpha)}/sin\,(\alpha)$.*

Proof. The triangle with the longest internal length that can be constructed with a minimum angle α is an isosceles triangle with two angles α, and one angle $\pi - 2\alpha$. The expression above is the length of the longest edge of such a triangle.

Theorem 3.5. *For a given point $p \in T_0^j$, there is a triangular element $T_{n,i}^j$ containing p such that p is arbitrarily close to a node of $T_{n,i}^j$.*

Proof. The triangular element represented by $T_{n,i}^j$ has area less than or equal to $f^n A^j$, where $f = 0.5$. An upper bound on the distance between a point p in $T_{n,i}^j$ and the nodes of $T_{n,i}^j$ is therefore $f^{n/2}C$, where $C = \sqrt{2A^j \sin(\pi - 2\alpha)}/\sin\alpha$ is constant. Since $f < 1$, for any $\epsilon > 0$, there is an positive integer n, such that $f^{n/2}C < \epsilon$.

Theorem 3.6. *If a sequence S satisfies the Largest Triangle Condition, then every mesh element will be refined within a finite number of refinements.*

Proof. We consider the worst case where the mesh has a single smallest triangle T with area A_s and N triangles of area A_l and show that the smallest triangle must be selected for refinement after a finite number of refinements.

Assume a sequence S has been generated that satisfies 3.1. Consider the set of leaves on the set of trees for the N triangles after the refinement of element s_k of sequence S. If Γ is the set of leaves with maximum area, one element of Γ is guaranteed to be refined in K refinements. An upper bound, therefore, for the number of refinements until T has the largest area of all the mesh element is therefore $KN\sum_{i=1}^{m} 4^i < KNm4^m$. In this many refinement steps plus K, the initially smallest triangle must be selected to be refined.

We have shown that if a sequence of triangular elements S is generated satisfying the condition 3.1, then this sequence will generate a dense set of nodes in the space in the limit of the sequence length approaching infinity. The weighting terms of Eqn.2 can thus be modified to create different search patterns while still satisfying that all contiguous closed regions in Ω must be eventually sampled.

4. Numerical Examples

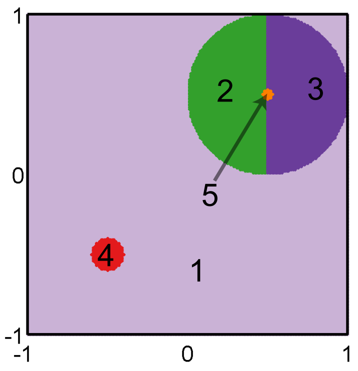
4.1. Example Case

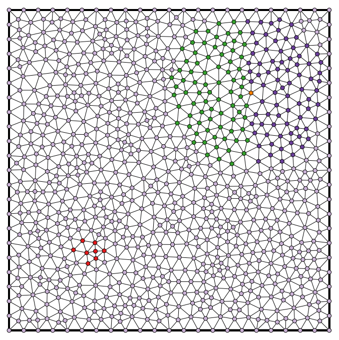
In this section we will demonstrate how the different terms of Eqn (2.1) can create different search patterns. Figure 3(a) shows the two-dimensional five-region test case that will be searched. The five regions are (1) light purple, (2) green, (3) dark purple, (4) red, and (5) orange, with areas 3.1832, 0.3911, 0.3911, 0.0314, and 0.003, respectively. In this test case, region 1 is the largest region, region 5 is the smallest but found at the boundary between region 2 and 3, the second largest regions, and region 4 is the second smallest region which is only adjacent to the largest region 1.

In figure 3(b), a mesh of 852 nodes was generated over this space. This represents a roughly uniform distribution of queries over the space generated with no learning. Region 1-5 are sampled 688, 77, 78, 8, and 1 times respectively, in approximate proportion to their area. This uniform mesh is used for comparison with the different searching methods.

Each iterated search is initialize with the mesh and label information shown in Figure 3(c). Regions 1, 2, and 3, have been sampled 43, 5, and 2 times respectively. The two smallest area regions 4 and 5 have not been sampled in the first iteration.

To explore how the different terms create different search patterns we consider five example searches of the space shown in Figure 4(a). In each search, w_A=1e-4, and the weights w_L, w_F, and w_{MD} are individually set to 1.0. We consider a final search example where three non-zero terms, w_A, w_F, and w_{MD} are combined. Each search is initialized with the roughly uniform mesh of 50 queries shown in Figure 3(c). In each subsequent iteration, each search selects a list of triangular elements to refine

(a)


(b)


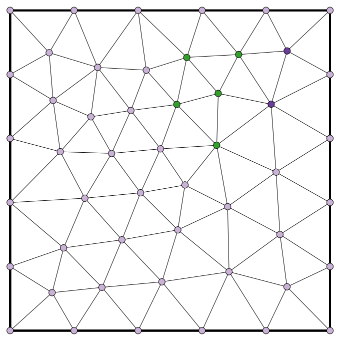
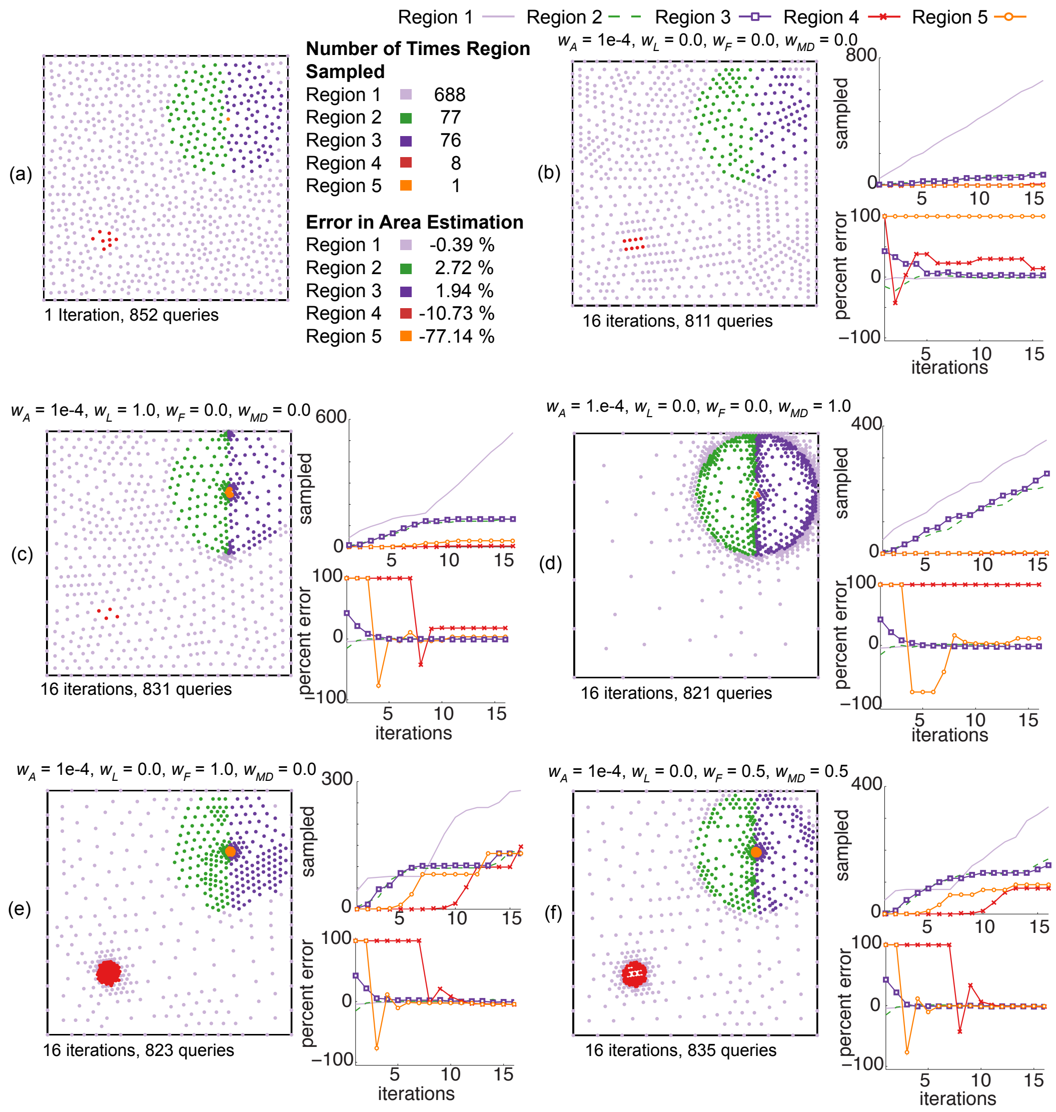
(c)


Figure 3: (a) The 2D five-region test case. (b) A uniform triangulated mesh of 852 nodes. (c) A uniform triangulated mesh of 50 nodes.

based on its scoring function, such that at least 50 queries are generated. Generally a range of 50-60 queries are generated. Each search is performed for sixteen total iterations. Figure 4(b-f) shows the results of these five searches compared to the single-iteration uniform search shown in Figure 4(a).

We use three measurements to compare the searches. (1) The total number of queries in each region. (2) The iteration at which region 4 and 5 are first found, and (3), the error in the approximation of the area of each region. For each search, a plot shows how many times each region has been sampled after each iteration. Note that the scaling of the y-axis of each plot is not kept the same in figure 4(b-f).

Measure (3) is generated by performing a Voronoi tessellation of the nodes in the parameter space and assigning the area of each Voronoi cell to the label at its center.[20] Voronoi cells are bounded by the boundary of the space. The approximated area of each region is the sum of the area of the Voronoi cells



Number of Times Region Sampled

Region 1	🟪	688
Region 2	🟩	77
Region 3	🟪	76
Region 4	🟥	8
Region 5	🟧	1

Error in Area Estimation

Region 1	🟪	-0.39 %
Region 2	🟩	2.72 %
Region 3	🟪	1.94 %
Region 4	🟥	-10.73 %
Region 5	🟧	-77.14 %

Figure 4: A comparison of how varying the weights creates different search patterns for the test case from Figure 3a.

with the regions label. The approximated areas are compared to the true areas to determine the percent error. For each search, a plot shows the percent error of the measurement of each region after each iteration of the search.

Measurement (2) can be directly determined by examining the percent in error of area plot. When region 4 and 5 are not found, the percent error in their area is 100%. By noting at what iteration the percent error of region 4 and 5 drops below 100%, it is straightforward to tell when they are discovered by the search. For example, in Figure 4(b), region 5 is never found, and in Figure 4(c), region 5 is found after 4 iterations.

In Figure 4(a), a roughly uniform mesh of 852 nodes is generated in a single iteration. In Figure 4(b), a roughly uniform mesh of 811 nodes is generated over 16 iterations by using a scoring function that only refines the largest triangular elements. Neither example learns from the results of the queries. Not surprisingly, a mesh initialized in a single iteration produces a more uniform distribution of nodes than a mesh generated by iteratively refining the largest triangular elements. The latter, for example, did not find region 5, while the former sampled it once. In both cases, regions were sampled in approximate proportion to their area and the approximate area of each region has significant error.

In Figure 4(c), the scoring function has an area term and a number of labels term from section 2.C.ii. This scoring function causes the search to preferentially refine triangles with three labels, found where regions 1, 2 and 3 meet and where regions 2, 3, and 5 meet. The focus on these small parts of the space results in very small triangular mesh elements, and at iteration eight, the search transitions to an area-focused search. This can be seen in the sudden increase in queries of region 1 and that region 4 is found at iteration eight. For this search, in any given iteration there are fewer triangular mesh elements with three labels to select than triangles requested for refinement, so elements with two labels are selected as well. Thus the search also explores the boundary of region 2 and 3 and finds region 5. In Figure 4(d), the scoring function has an area term and a minimum diversity term from section 2.C.iv. The search now focuses on the boundary formed by region 1, 2, 3, and 5. The search resolves any boundary with equal preference. A boundary search finds region 4 in two iterations. Over the 821 total queries, the search does not discover region 4.

In Figure 4(e) the scoring function has an area term and a novelty term from section 2.C.iii. The novelty term drives the search to balance the number of queries of each region. For example, region 5 is discovered in three iteration and by the seventh iteration has been sampled the same as region 1. The distribution of nodes in interior of each region is roughly uniform. Noticeably, the balanced number of queries of the five regions causes the area error to drop quickly.

In Figure 4(f), the scoring function has an area term, a novelty term, and a minimum diversity term. This results in the search both balancing the number of queries between regions and preferentially exploring boundaries. Region 5 is discovered in three iterations. Region 4 is not discovered for ten iterations, but then is preferentially sampled. The errors in the areas after 16 iterations are the smallest of all the searches shown.

Region 5 is an order of magnitude smaller than region 4. It was only sampled once by the search of Figure 4(a) and not at all by the search of Figure 4(b). However, for each search shown in Figure 4(c-f), the region was found sooner than region 4. This supports the conclusion that using a non-uniform search pattern to find new regions is the most effective in spaces where regions tend to be adjacent to more than one regions or regions of similar areas. Islands, small area regions adjacent only to a single large area region, are difficult to find.

4.2. *Sensitivity Study*

We perform a study of the sensitivity of the search to the three parameters $\{w_A$, w_F, and $w_{MD}\}$; w_L is set to zero. We consider all combinations of the following parameter sets, $w_A = \{1, 1\text{e-}1, 1\text{e-}2, 1\text{e-}3, 1\text{e-}4\}$, $w_F = \{0, 0.25, 0.5, 0.75, 1.0\}$, and $w_{MD} = \{0, 0.25, 0.5, 0.75, 1.0\}$. For the sensitivity study, we consider the five cases shown in the first column of Figure 5.

The first case divides the space into nine equal regions. In the second case and third case, the space has ten regions, nine of which are identical in area. In the second case the regions are adjacent and in the third case they are islands. In the fourth and fifth case, the space has ten regions, nine of which have areas that decrease by a power law. In the fourth case the regions are adjacent and in the fifth case they are islands. To measure the quality of the search, we use the summed absolute value of the error of the area of the nine or ten regions, using the Voronoi tessellation described above to measure the area. We subtract from this value the error that results from performing a search where only $w_A > 0$. Thus we are measuring whether the additional terms improve the search relative to an iterated uniform search.

We show cross-sections of the results of the sensitivity study in Figure 5 ($w_F = w_{MD}$). Each plot shows the sum error in the calculation of the areas of the nine or ten regions minus the sum error for the uniform case (e.g. $w_F = w_{MD} = 0$). If the value is negative, the search had less error than the uniform search pattern, and if the value is positive, the search had more error than the uniform search pattern.

We first observe that, for Case 1, no term makes a significant difference. For Cases 2, 3, 4 and 5, we observe that the most important term is w_A. If the space has connected regions (Case 2 and 4), w_A should be small. If the space has islands (Case 3 and 5) w_A should be large. We observe that even if a space primarily has island regions, that the search can be improved by positive w_{MD} and w_F terms. If w_A is chosen correctly for the space, then positive values of w_{MD} and w_F improve the search relative to a uniform search. If w_A is chosen poorly for the space, then positive values of w_{MD} and w_F can sometimes worsen the search relative to a uniform search. On the right two columns of Figure 5, we compare a uniform search of the space to a search performed using $\{w_A = 1\text{e-}2, w_{MD} = 0.5, w_F = 0.5\}$ if the regions are islands and $\{w_A = 1\text{e-}4, w_{MD} = 0.5, w_F = 0.5\}$ if the regions are adjacent. The improved the number queries of the small regions is evident relative to the uniform search.

4.3. *Numerical Example using 2D Lennard Jones-Gauss System*

In this section we consider the application of the method from Section 2 to a complex system that is explored by real queries.

The 2D Lennard Jones Gauss (LJG) system consists of particles confined to move in a plane that interact via a double-well pair potential function that is a function of ϵ, r_0, and σ^2. The simple or complex crystal assembled by the particles is dependent upon the value of the three parameters. In reference [7], Engel and Trebin distributed 5500 queries on a fine grid, ϵ in [0.1, 5.0] at 0.1 increments, and r_0 in [1.01, 2.1] at 0.01 increments, and $\sigma^2 = 0.02$. Each query corresponded to performing a simulated annealing of 1024 particles from a high temperature (liquid/gas state) to a low temperature (crystal state) over $2\text{x}10^6$ molecular dynamics (MD) time steps at zero pressure. In each simulation, the particles condense into one or two droplets. A crystal structure was identified from the particle positions in final frame of the MD trajectory. Seven crystal structures were identified, hexagonal, square, decagonal quasicrystal, dodecagonal quasicrystal, honeycomb, and rhombic. These crystal structures were found over eight regions (the square phase was found in two non-contiguous regions). In reference [21], Phillips and Voth applied a shape matching and machine learning analysis to the data set and discovered an eighth phase, an amorphous triangle/square phase, in a small region between the hexagonal and square phase.

For this study, queries corresponded to a simulated annealing of 1024 particles over $5\text{x}10^6$ molecular dynamics (MD) time steps. All data was generated on NVIDIA Tesla M2070 GPUs, at the Tukey cluster at Argonne National Laboratory.

For a system generating real data, the results of each query must be classified. For this study, data was labeled by using a small training set of fifty examples and a supervised learning method. An S-2 global shape descriptor[21] was calculated

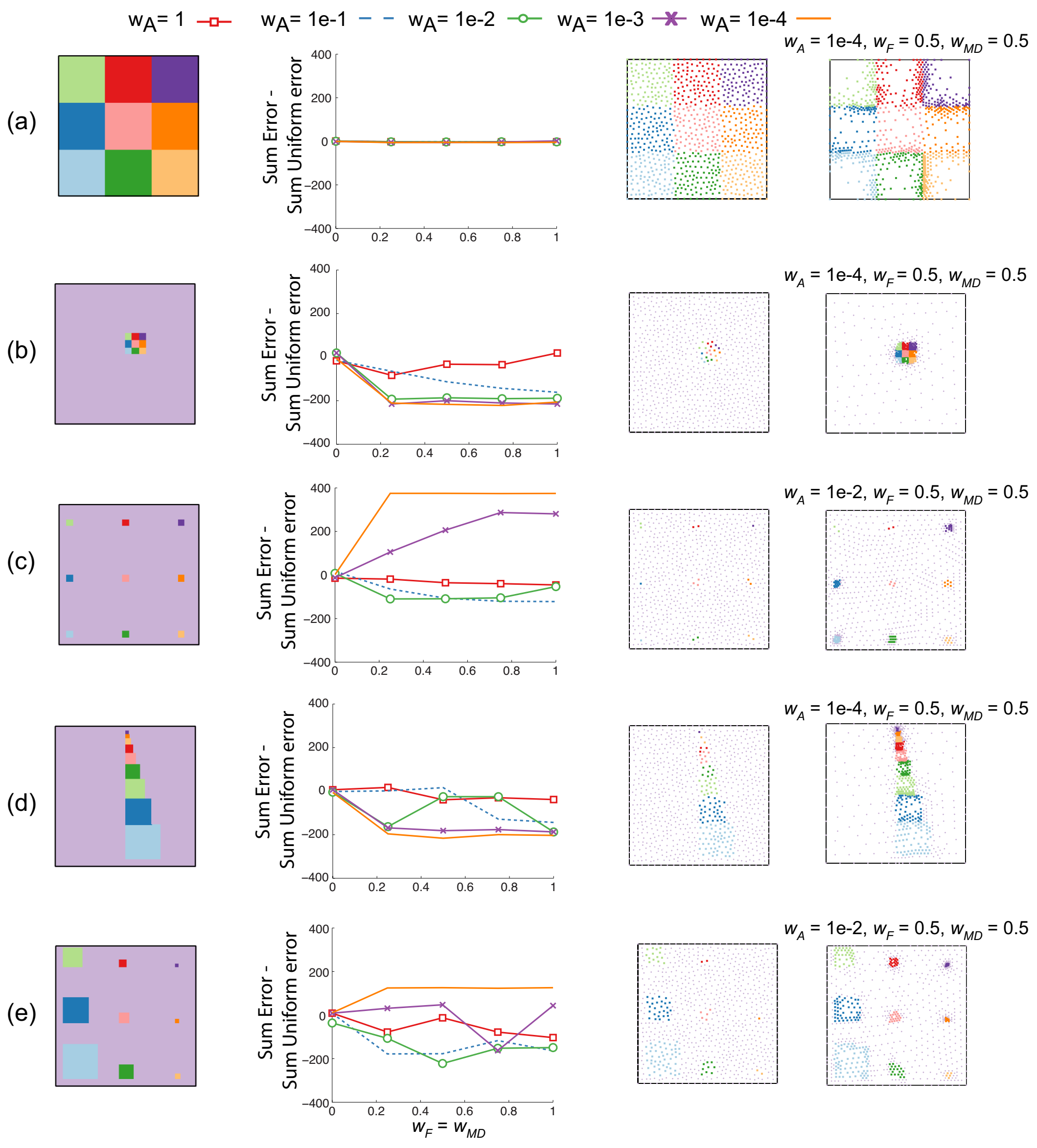


Figure 5: A sensitivity study of the impact of the weights $\{w_A, w_F, w_{MD}\}$ for five cases.

for the coordinate set generated by each query or the training set. Using this descriptor as the feature vector and a euclidean distance metric, a k-nearest neighbors classifier (k=1)[22] was used to classify the result of each query, which was then visually confirmed.

A complication of real data is that some queries cannot be granted a non-ambiguous label. Along the boundary between two phases, the crystal found in the data may include a mixture of the phases. This data may represent a true small two-phase region, or the result of an unequilbrated simulation. If one of the two phases is dominant, i.e. represents most of the coordinate set, we classify the query result as that phase. If both phases are approximately equally present, we use a special label to represent this. This label was treated as a unique label for the purpose of calculating the minimum diversity score, but not included in the novelty score.

In Figure 6, we compare the uniform distribution of 5500 queries to a uniform distribution of 1080 queries to 1071 queries issued over 20 iterations. The classification of the data in the top plot of Figure 6, was generated by a machine learning and shape matching method discussed in more detail in reference[21]. Dark purple data corresponds to data with ambiguous labels per that method. In the Figure 6 middle left and right plots, the uniform and refined mesh generated per Section 2, are shown. The 20 iterations were generated using the weights $\{w_A$ = 1e-4, w_L= 0, w_{MD} = 0.5, w_F = 0.5$\}$. The bottom left and right plots of Figure 6 show approximated boundaries of the regions based on the data collected. The boundaries were generated by using a multi-class support vector machine with radial basis function kernel (gamma = 20, C=3).

In Table 1, the number of times each data set queries each region of the 2D LJG phase diagram is compared. The regions

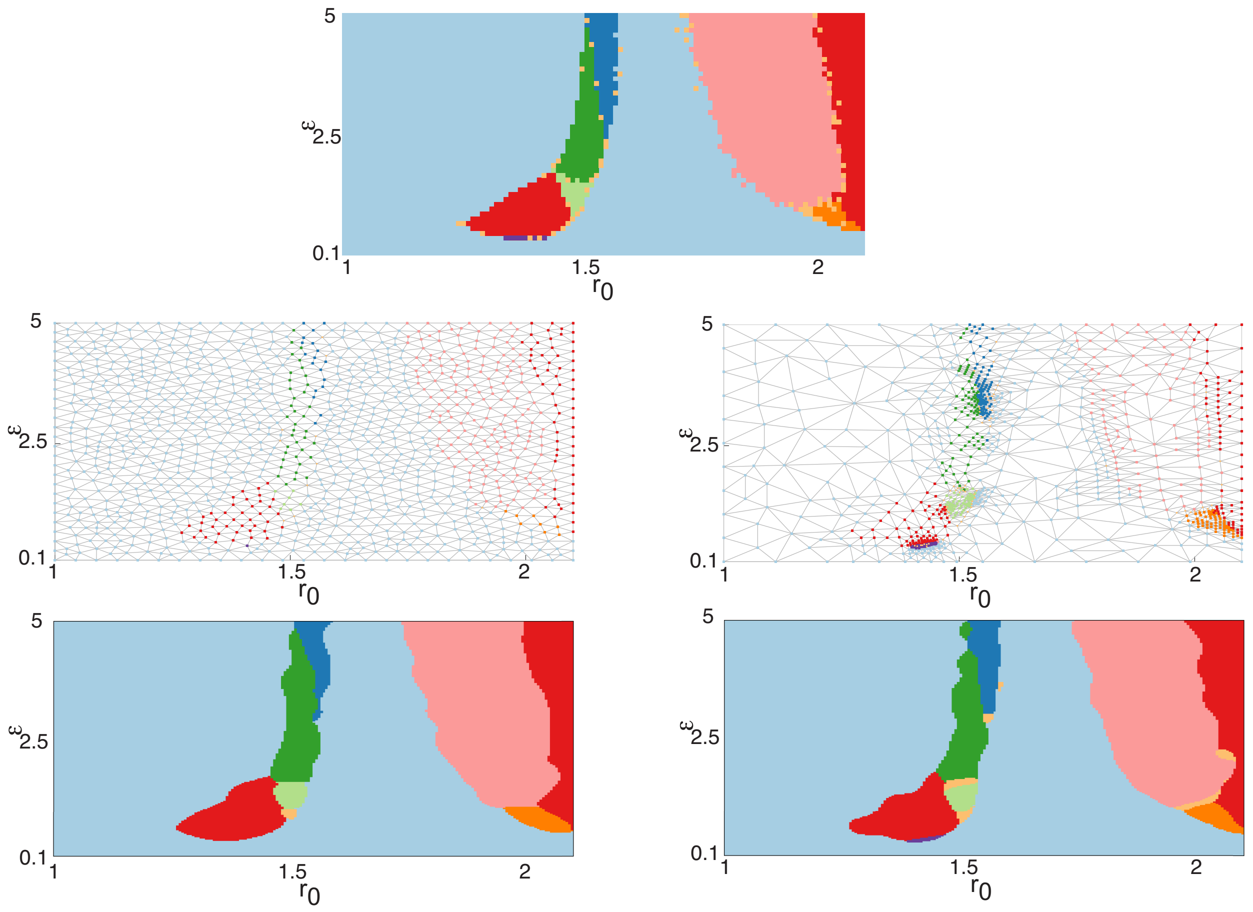


Figure 6: (Top Center) 5500 queries distributed uniformly on a rectangular 55x100 grid. (Left) 1080 queries distributed uniformly. (Left Bottom) A support vector machine generated partitioning of the space using the results of the uniform queries. (Right) 1071 queries distributed over 20 iterations. (Right Bottom) A support vector machine generated partitioning of the space using the results of the iterated queries. The triangular meshes appear stretched because in the space where the mesh us generated, the parameter space is normalized to a unit square.

in the table are listed in order of their area. Both uniform methods sample each region in rough proportion to its relative area fraction of the phase diagram. The iterated method finds and samples the smaller regions with more queries.

In the right bottom of Figure 6, we observe that the boundaries of the regions generated using the iterated method are not as smooth as that of the uniformly distributed data. One way to improve the boundaries of the regions is to change the weighting parameters of the search. In Figure 7, we show the result of 20 iterations, where the weight parameters are $\{w_A$ = 1e-4, w_L= 0, w_{MD} = 0.5, w_F = 0.5$\}$, for the first 15 iterations and $\{w_A$ = 1e-4, w_L= 0, w_{MD} = 1.0, w_F = 0$\}$ for the last 5 iterations. The weight set of the first fifteen iterations prioritizes novelty, boundaries, and area. The weight set of the last five iterations prioritizes just boundaries and area. The result is a significant smoothing of the regional boundaries.

5. Conclusions

In this paper we have introduced a learning heuristic for issuing queries to search for and map regions in a 2D parameter space. This method applies a conforming mesh to the space and determines where the mesh should be refined based on a formula that ranks the mesh elements. We have shown that, as long as the ranking function includes an unbounded mesh element area term and all other terms are bounded, the heuristic will densely fill the space with queries in the infinite limit. We applied this heuristic to several test numerical cases, including a mapping of the parameter space of the 2D Lennard-Jones Gauss particle system, where eight simple and complex crystals are known to self-assemble. We found that choosing different weights for the ranking function terms changes the type of search performed. We found that both spaces with island regions and spaces that only have connected regions can be searched more efficiently than a uniformly distributed query set when using an appropriate weighting for the mesh element area term. We also found that by changing the set of weights being used after a given iteration, the search strategy can be changed from emphasizing discovery to emphasizing mapping boundaries, as meets the wants of the investigator.

There are many ways this heuristic can be extended and modified. For example, using a tetrahedral mesh, this heuristic can

Table 1: Comparison of the number of queries

Phase	Uniform (grid)	Uniform (mesh)	Iterated (20)	Iterated (15/5)
Hexagonal (light blue)	3690	717	346	347
Dodecagonal Quasicrystal (light red)	947	179	111	136
Square (dark red)	464	101	179	208
Decagonal Quasicrystal (dark green)	152	38	83	73
HoneyComb (dark blue)	108	18	91	82
Pentagonal (light green)	34	7	62	38
Rhombus (dark orange)	29	7	61	66
Amorphous Triangle/Square (dark purple)	6	1	67	50
boundary (light orange)	70	11	71	60
Total	5500	1080	1071	1060

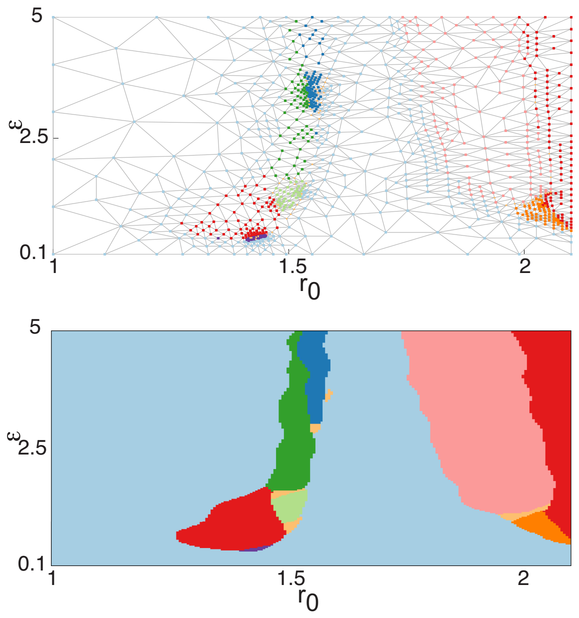


Figure 7: (Top) 1060 queries distributed over 20 iterations. For the last 5 iterations, a weight set of $\{w_A = 1e\text{-}4, w_{MD} = 1.0, w_F = 0.0\}$ was used. (Bottom) A support vector machine generated partitioning of the space using the results of the iterated queries.

extended to a three dimensional parameter space. In the above work, a triangular mesh refined by an heuristic that creates no hanging nodes was chosen for simplicity. A consequence of this choice is that some triangular elements that are not highly ranked are also refined. However, the heuristic can easily be modified to use a structured mesh with rectangular or hexahedral mesh elements, if a method for accommodating hanging nodes, that is, an uneven number of labels per element, can be developed in the ranking formula. Also the ranking function proposed in Equation 2 can be tailored to direct knowledge about the space or other subjective knowledge of what features are of interest in a given search.

The end goal of heuristics such as this is to eventually achieve automated mapping and discovery in low-dimensional spaces, with minimal intervention required from the researcher. To our knowledge, this heuristic is the first of its kind in addressing a classification problem in a continuous space. Full automation of this heuristic requires coupling it with an algorithm that can classify the data from queries as it is acquired and recognize when data is novel and thus warrants a new classification category. Reference [21] and the machine learning tools implemented in Section 4.3 represent first attempts to achieve this coupling. There is a large potential for algorithmic development in this area. We anticipate that methods for organizing and optimizing searches through different kinds of parameter spaces will play an important and growing role in accelerating scientific discovery, especially as computational models mature as tools for ab initio discovery.

6. Acknowledgements

This research was supported in part by a grant from the Office of Naval Research and in part by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357. C.L.P. was funded by the Office of the Director through the Named Postdoctoral Fellowship Program (Aneesur Rahman Postdoctoral Fellowship), Argonne National Laboratory. We acknowledge numerous useful discussions with Todd Munson, Stefan Wild, and Tim Tautges. We than Todd Munson for reviewing the mathematical analysis. We thank Michael Engel for providing the Injavis software package, Aaron Keys for the SMAC software package. Both the k-nearest-neighbor classifier and the support vector machine used software from the scikit-learn open source package[22].

References

[1] George M. Whitesides and Mila Boncheva. Beyond molecules: Self-assembly of mesoscopic and macroscopic components. *Proceedings of the National Academy of Sciences*, 99(8):4769–4774, 2002.

[2] George M. Whitesides and Bartosz Grzybowski. Self-assembly at all scales. *Science*, 295(5564):2418–2421, 2002. doi: 10.1126/science.1070821.

[3] Robert J. Macfarlane, Byeongdu Lee, Matthew R. Jones, Nadine Harris, George C. Schatz, and Chad A. Mirkin. Nanoparticle superlattice engineering with DNA. *Science*, 334(6053):204–208, 2011.

[4] Elena V. Shevchenko, Dmitri V. Talapin, Nicholas A. Kotov, Stephen O'Brien, and Christopher B. Murray. Structural diversity in binary nanoparticle superlattices. *Nature*, 439(7072):9, 2006.

[5] Wim L. Noorduin, Alison Grinthal, L. Mahadevan, and Joanna Aizenberg. Rationally designed complex, hierarchical microarchitectures. *Science*, 340(6134):832–837, 2013. doi: 10.1126/science.1234621.

[6] Sharon C. Glotzer and Michael J. Solomon. Anisotropy of building blocks and their assembly into complex structures. *Nature Materials*, 6(7):557–562, August 2007.

[7] Michael Engel and Hans-Rainer Trebin. Self-assembly of monatomic complex crystals and quasicrystals with a double-well interaction potential. *Phys. Rev. Lett.*, 98:225505, Jun 2007.

[8] Nguyen H. P. Nguyen, Eric Jankowski, and Sharon C. Glotzer. Thermal and athermal three-dimensional swarms of self-propelled particles. *Phys. Rev. E*, 86:011136, Jul 2012.

[9] Graziano Vernizzi, Rastko Sknepnek, and Monica Olvera de la Cruz. Platonic and Archimedean geometries in multicomponent elastic membranes. *Proceedings of the National Academy of Sciences*, 108(11):4292–4296, 2011. doi: 10.1073/pnas.1012872108.

[10] Ines C. Pons-Siepermann and Sharon C. Glotzer. Design of patchy particles using ternary self-assembled monolayers. *Soft Matter*, 8:6226–6231, 2012.

[11] C. R. Iacovella C. L. Phillips and S. C. Glotzer. Stability of the double gyroid phase to nanoparticle polydispersity in polymer-tethered nanosphere systems. *Soft Matter*, 6:1693 – 1703, 2010.

[12] Jack P.C. Kleijnen. Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707 – 716, 2009. ISSN 0377-2217.

[13] Carlos Cotta and Jano van Hemert, editors. *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, volume 153 of *Studies in Computational Intelligence*. Springer, 2008.

[14] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53 (3):484 – 512, 1984.

[15] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, May 1989.

[16] Yao Zheng. Automatic mesh generation: Application to finite element methods, by p. l. george, wiley, new york, 1991. no. of pages: X + 333. isbn 0-471-93097-0. *International Journal for Numerical Methods in Engineering*, 38(14):2483–2484, 1995.

[17] I. G. Rosenberg and F. Stenger. A lower bound on the angles of triangles constructed by bisecting the longest side. *Math. Comp.*, 29:390–395, 1975.

[18] M. Cecilia Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International Journal for Numerical Methods in Engineering*, 20(4):745–756, 1984.

[19] MATLAB and Partial Differential Equation Toolbox. *version 7.12.0.635 (R2011a)*. The MathWorks Inc., Natick, Massachusetts, 2011.

[20] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. `http://control.ee.ethz.ch/ mpt`.

[21] Carolyn L. Phillips and Gregory A. Voth. Discovering crystals using shape matching and machine learning. *Soft Matter*, 9:8552–8568, 2013.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.